Metaheuristic methods

Genetic and evolutive algorithms

Juan Antonio Díaz García

Genetic and evolutionary algorithms

"The idea behind genetic algorithms is to do what nature does. Let us take rabbits as an example: at any given time there is a population of rabbits. Some of them are faster and smarter than other rabbits. These faster, smarter rabbits are less likely to be eaten by foxes, and therefore more of them survive to do what rabbits do best: make more rabbits. Of course some of the slower, dumber rabbits will survive just because they are lucky. This surviving population of rabbits starts breeding. The breeding results in a good mixture of rabbit genetic material: some slow rabbits breed with fast rabbits, some fast with fast, some smart rabbits with dumb rabbits, and so on. And on the top of that, nature throws in a 'wild hare' every once in a while by mutating some of the rabbit genetic material. The resulting baby rabbits will (on average) be faster and smarter than these in the original population because more faster, smarter parents survived the foxes. (It is a good thing that the foxes are undergoing similar process - otherwise the rabbits might become too fast and smart for the foxes to catch any of them). A genetic algorithm follows a step-by-step procedure that closely matches the story of the rabbits."[1]

[1] Zbigniew Michalewicz, *Genetic algorithms + data structures = evolution programs (Third edition), Springer, 1992, page 14.*

Genetic and evolutionary algorithms[2]

Genetic and evolutionary algorithms are based on *competition* and are iterative optimization algorithms which simulate the evolution of species. Main characteristics:

- An initial population is generated (usually at random).
- Each individual is the encoded version of a tentative solution.
- A fitness function is associated with every individual in the population to represent is suitability to the problem.
- At each step some individuals form the population are selected, following the selection paradigm in which individuals with better fitness are selected with higher probability.
- New offsprings are generated by means of genetic operators (crossover, mutation, etc.) applied to the selected individuals.
- The population is replaced applying a replacement scheme to select the new population form the current population and the generated offsprings.

[2] El-Ghazaby Talbi, {*tinyMetaheuristics: From design to implementation, Wiley, 2009, chapter 3.*

Genetic and evolutionary algorithms

Metaphor	Optimization
Evolution	Problem solving
Individual	Solution
Fitness	Objective function
Environment	Optimization problem
Locus	Element of the solution
Allele	Value of the element (locus)

From: El-Ghazaly Talbi, *Metaheuristics: From design to implementation*, Wiley, 2009, page 200.

Genetic and evolutionary algorithms

Juan Antonio Díaz García

Solution representation:

- Genotype: Represents the encoding.
- Phenotype: Represents the solution.
- Genotype must be decoded to generate the phenotype.
- Genetic operators acts on genotype levels while the fitness function of an individual will use the phenotype of the associated individual.

Genetic and evolutionary algorithms

Main search component to design an evolutionary algorithm:

- **Solution representation**: The encoded solution is referred as *chromosome*, the variables in a chromosome are referred as *genes*. The possible values of variables (genes) are the *alleles* and the position of an element (gene) in the chromosome is the *locus*.
- **Initial population**: The method used to generate the initial population.
- **Fitness function**: It is used to rank the individuals of a population.
- **Selection strategy**: Defines the strategy to select individuals that will generate new offsprings.
- **Reproduction strategy**: Describes the operators (crossover and mutation) to generate new offsprings.
- **Replacement strategy**: The strategy used to select the next generation selecting individuals from the current population and the generated offsprings (survival of the fittest).
- **Stopping criteria**: Rules that define the number of generations to simulate.

Genetic and evolutionary algorithms

Evolutionary heuristic pseudocode

$t \leftarrow 0$
$P_0 \leftarrow$ GENERATATEINITIALPOPULATION($Size$)
EVALUATE(P_0)
$Stop \leftarrow$ **false**
while (**not** $Stop$) **do**
    $t \leftarrow t + 1$
    $P' \leftarrow$ SELECT(P_t)
    $P^{new} \leftarrow$ ALTER(P')
    EVALUATE(P^{new})
    REPLACE($P_t \cup P^{new}$)
    $Stop \leftarrow$UPDATESTOPCRITERION( )
end while

$$\max_{x\in[-1,2]} \quad f(x) = x\sin(10\pi x) + 1$$

1. **Solution representation**: We can use a binary representation of real values in the domain of x. If we want a precision of six places after the decimal point we must divide the domain $[-1,2]$ in 3000000 equal size ranges. Therefore we need at least 22 bits to represent at least 3,000,000 different values for the variable x.

$$2^{21} = 2097152 < 3000000 < 4194304 = 2^{22}$$

Solution decoding: Given the decimal representation x' of a binary number we can obtain the corresponding value of x in the interval $[-1,2]$ as follows

$$x = -1 + \frac{3}{2^{22}-1}x'$$

[3] Zbigniew Michalewicz, *Genetic algorithms + data structures = evolution programs (Third edition)*, Springer, 1992, chapter 1.

Examples:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$$
\begin{aligned}
x' &= 0 \\
x &= -1 + \frac{3}{4194303} \times 0 = -1
\end{aligned}
$$

Juan Antonio Díaz García

Examples:

1	1	0	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	1	0	1	0
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$$
\begin{aligned}
x' &= 3183274 \\
x &= -1 + \frac{3}{4194303} \times 3183274 = 1.276856
\end{aligned}
$$

Examples:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

$$
\begin{aligned}
x' &= 4194303 \\
x &= -1 + \frac{3}{4194303} \times 4194303 = 2
\end{aligned}
$$

Genetic and evolutionary algorithm

Example: Optimization of a simple function

2. **Initial Population**: Generate a population of chromosomes where each chromosome is a random binary vector with 22 bits.
3. **Fitness function**: For each phenotype x, $fitness(x) = f(x)$
4. **Selection strategy**: Select a fraction of the population at random.
5. **Reproduction strategy**:
   - *Crossover*: For each pair of parents, select at random a crossover point

Parent 1	0	1	1	0	1	1	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0
Parent 2	1	0	1	1	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	0	1	0
Offspring 1	0	1	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	0	1	0
Offspring 2	1	0	1	1	1	1	1	0	0	0	1	0	1	1	0	1	1	1	0	1	0	1	0

5. **Reproduction strategy**:
   - *Mutation*: Select randomly a gen of the chromosome and flip it.

0	1	1	0	1	1	1	0	0	1	0	1	**1**	0	1	1	1	0	1	0	1	0

0	1	1	0	1	1	1	0	0	1	0	1	**0**	0	1	1	1	0	1	0	1	0

6. **Replacement**: An offspring is incorporated into the population, if it is distinct (i.e., not identical to an existing individual), and if its fitness value is better than the worst fitness value in the population, by discarding the worst individual.
7. **Stop criterion**: A fixed number of iterations.