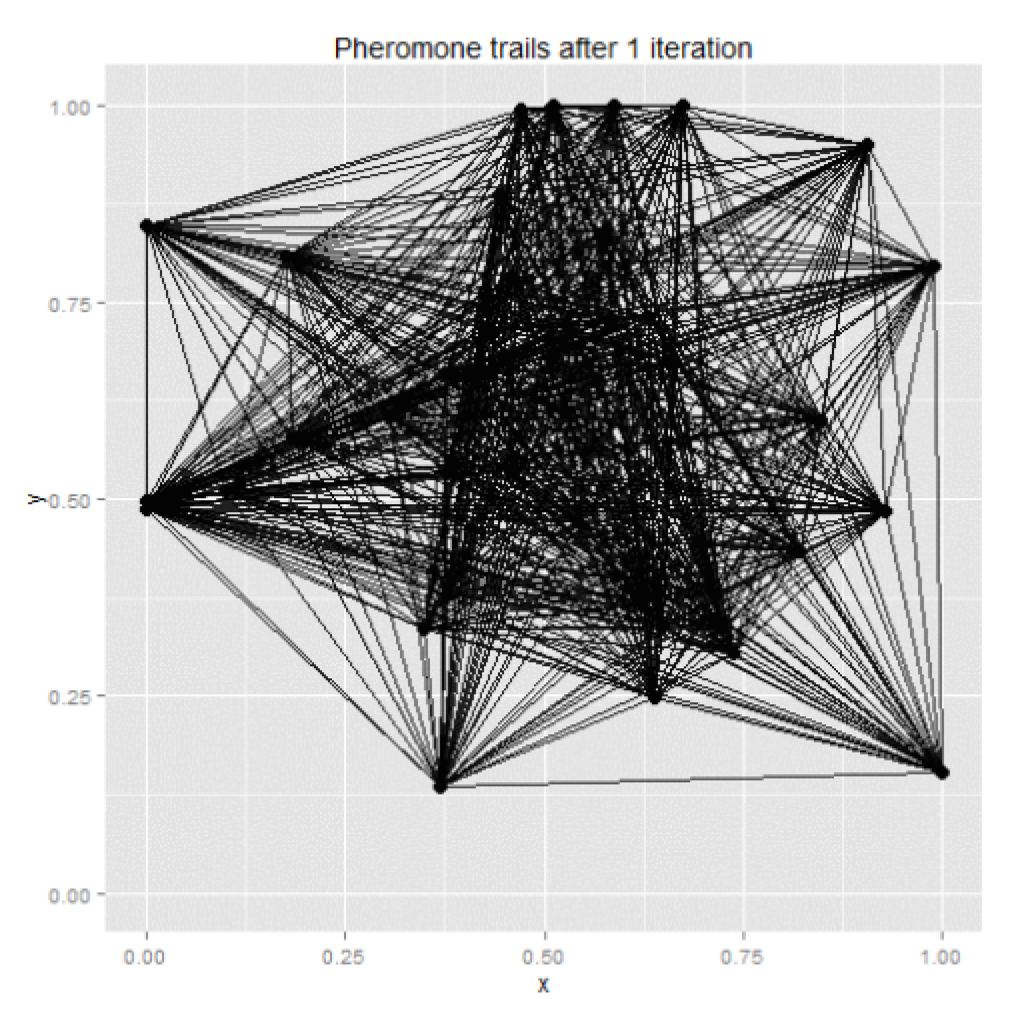
Cheat Sheets

Back to index

Metaheuristics

Ant Colony Optimization



Description

Bio-inspired metaheuristic. It is inspired on the ants' behavior, where an ant guides the others when they find something good.

Types of problems

Intended to solve problems related to graphs. i.e. Useful to find paths.

Representation

For this problem, some terms are defined.

- C_{ij}: Path from the node i to the j
- τ_{ij}: Pheromones in the path from the node i to the node j
- η_{ij}: Heuristic in the path from the node i to node j
- ρ: Evaporation, $[0, 1]$

Movement

- Each ant moves around the graph following the criteria:

$$P(C_{ik}) = \frac{\tau_{ik}^{\alpha} * \eta_{ik}^{\beta}}{\Sigma_{\epsilon N_i} \tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}$$

Note. To optimize distance $\eta_{ik} = \frac{1}{d_{ik}}$ is proposed where d_{ik} is the length of the component C_{ik}.

Update

After all the ants traverse the graph, the pheromones:

Are updated

$$\Delta\tau_{ij}^{a} = \begin{cases} \frac{1}{L_a} & Used \\ 0 & Otherwise \end{cases}$$

And evaporate

$$\tau_{ij} = (1 - \rho) * \tau_{ij}$$

Pseudocode

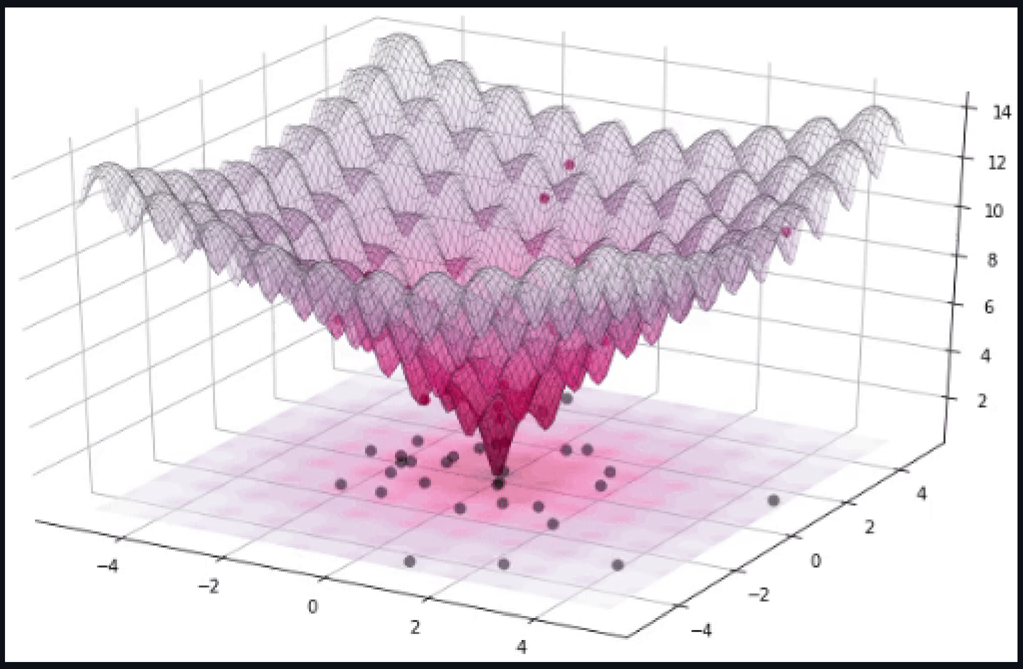
Ant colony

```
At the beginning
    Pheromones <- small value

Move the ants one by one
    Ants: move around the graph

When all the ants record the graph
    Pheromones: are updated
    Ants: deposit pheromones
    Pheromones: evaporate
```

Back to index

Evolutionary algorithms

Differential evolution



Description

This a robust algorithm and introduces the idea of mutating individuals based on 3 others, in order to work as a gravitational force into the minimums.

Types of problems

This algorithm solves continuous multidimensional optimization problems.

Representation

A vector of real values.

$$\vec{x^i} =< x_1^i, x_2^i, \ldots, x_d^i >$$

Mutation

This is the main operation of differential evolution.

For each individual x^i, another one called v^i is generated by performing the following operation.

$$v^i = x^{r1} + (x^{r2} - x^{r3})F$$
$$0 \leq F \leq 2$$

Crossover/Recombination

A new individual is created element by element, randomly picking values from the original one x^i or the mutated one v^i.

This is performed based on a Crossover probability $0 \leq Cr \leq 1$.

$$u_k^i = v_k^i \quad if \quad rand(0,1) \leq Cr \quad else \quad x_k^i$$

Survivor selection

Tournament is used: The best individual between u^i and x^i is selected to be part of the next generation.

Pseudocode

Differential evolution

```
Parameters:
    N -> Population size
    G -> Maximum number of generations
    Cr -> Crossover probability

Return: the best individual

Begin
    Create the initial population of N individuals
    Calculate the population fitness
    While the number of generations is less than G or a good solution hasn't been found
        For each individual in the population.
            Mutation -> Create new individual
            Crossover -> Combine the individuals
            Calculate the fitness of the new one
            Selection -> Select the best between both
        End for
    End while
End
```

Cheat Sheets

Back to index

Evolutionary algorithms

Evolutionary programming

Description

The solutions represent species instead of individuals.

Types of problems

It has evolved to solve continuous multidimensional optimization problems.

Representation

The individual's solution is represented with a vector of d real values where d is the number of features to optimize. In addition to the values, a *mutation step size* is used to guide the change of each individual's mutation.

$$< \vec{x}_i, \vec{\sigma}_1 >$$

Mutation

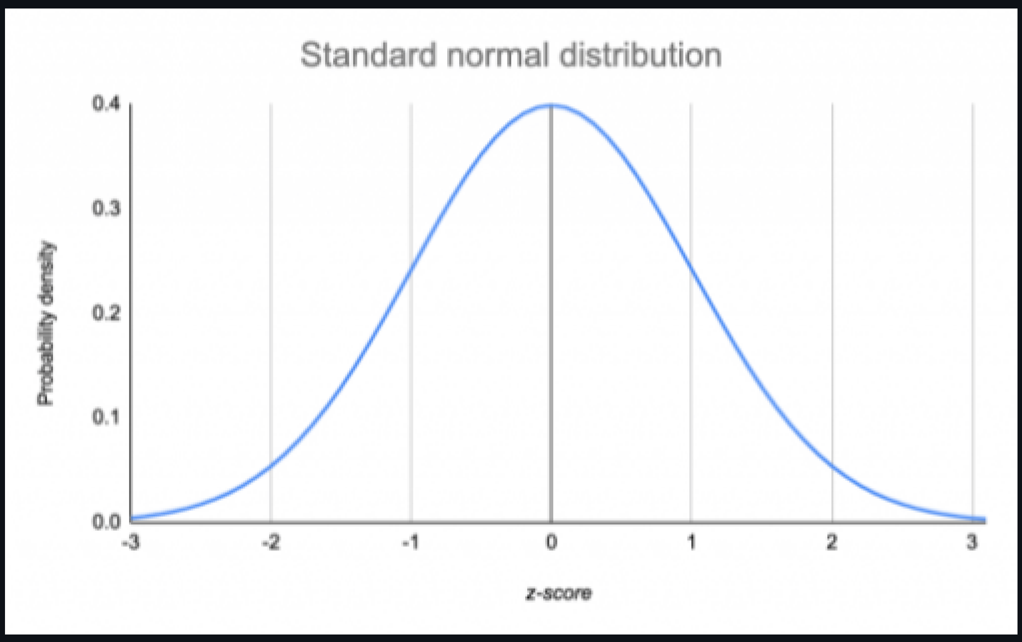
Each value is mutated with a random number based on its step size.

Step size is eventually updated as well.

$$\sigma_i' = \sigma_i(1 + N(0, \alpha))$$

$$x_i' = x_i + N(0, \sigma_i')$$

$$\alpha \approx 0.2$$



Survivor selection

$(\mu + \lambda)$ selection: Where the best μ individuals are selected out of the union of parents and offspring.

Pseudocode

Evolutionary programming

```
Parameters:
    μ -> Population size
    G -> Maximum number of generations

Return: the elite individual

Begin
    Create the initial population
    Calculate the population fitness
    Get the elite
    While the number of generations is less than G or a good solution hasn't beed found
        Mutation of all the species
        Calculate the population fitness
        Survivor selection
        Get the elite or include the elite in the population
    End while
End
```

Evolutionary algorithms

Evolution strategies

Description

The main characteristis of this algorithms is the *self-adaptation* of parameters, since they evolve with the individual itself.

Types of problems

It's designed to solve continuous multidimensional optimization problems.

Representation

The individual's solution is represented with a vector of d real values where d is the number of features to optimize. In addition to the values, a *mutation step size* is used to guide the change of each individual's mutation.

- If all the variables to be calculated are in the same range, a single step size can be used.

$$< \vec{x_i}, \sigma_1 >$$

- Otherwise, a size per featue is recommended.

$$< \vec{x_i}, \vec{\sigma_1} >$$

Parent selection technique

Completely random, this is because the whole population is seen as parent.

Crossover/Recombination

Two variants are used:

- Intermediate recombination

$$\frac{\vec{p_1} + \vec{p_2}}{2}$$

- Discrete recombination

$$RandomSelection \quad [\vec{p_1}_i, \vec{p_2}_i]$$

Mutation

The mutation of the features consist of adding a random value based on a normal distribution zero-centered with a standard deviation equals to the corresponding σ.

$$x_i' = x_i + N(0, \sigma_1)$$

On the other hand, updating the step size depends of the chosen representation.

- With one step size

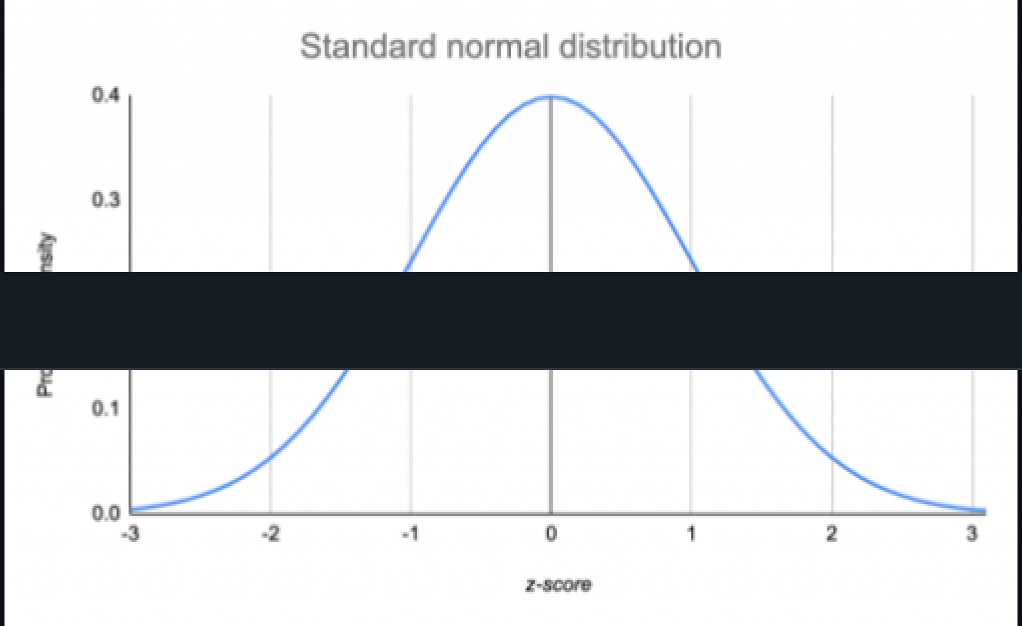
$$\sigma' = \sigma e^{N(0,\tau)}$$

  Where $\tau = \frac{1}{\sqrt{n}}$ and n is the population size.

- With d step sizes

$$\sigma' = \sigma e^{N(0,\tau_1)+N(0,\tau_2)}$$

  Where $\tau_1 = \frac{1}{\sqrt{n}}$, $\tau_2 = \frac{1}{\sqrt{2\sqrt{n}}}$ and n is the population size.



Also, see the **1/5 sucess rule**.

Survivor selection

After creating the offspring λ and calculating their fitness, the best μ are deterministically chosen. There are two approaches.

- (μ, λ) selection: The best μ individuals are selected out of the offspring.
- $(\mu + \lambda)$ selection: Where the μ individuals are selected out of the union of parents and offspring.

Pseudocode

Evolution strategies

```
Parameters:
    N -> Population size
    λ -> Offspring size
    G -> Maximum number of generations

Return: The elite individual

Begin
    Create the initial population
    Calculate the population fitness
    Get the elite
    While the number of generations is less than G or a good solution hasn't been found
        Recombination
        Mutation
        Calculate the population fitness
        Survivor selection (the best individuals)
        Get the elite or include the elite in the population
    End while
End
```

Back to index

Evolutionary algorithms

Genetic algorithms

Description

Probably the most famous algorithm of its kind.

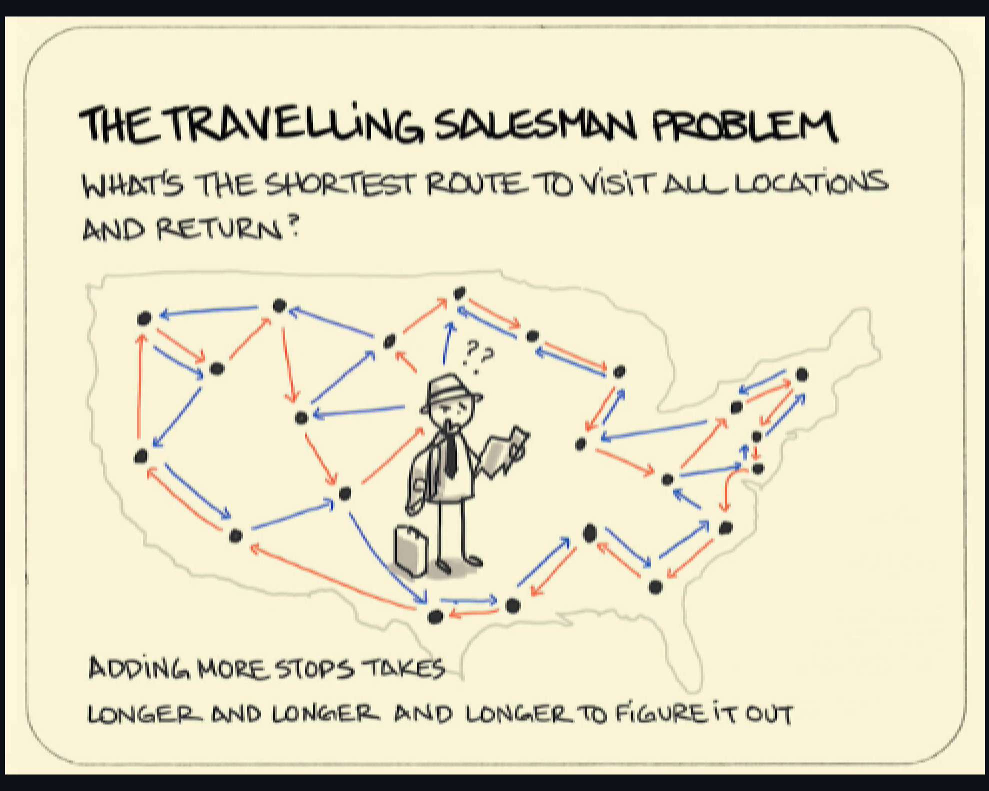
The inspiration comes from the DNA structures. Where there's a population with chromosomes and each one consists of genes.

Types of problems

This algorithm can be used for *constrained* or *unconstrained* problems that are not usually suitable for standard optimization algorithms.

Some of the problems are:

- Traveling salesman problem.



- 8 queens problem.

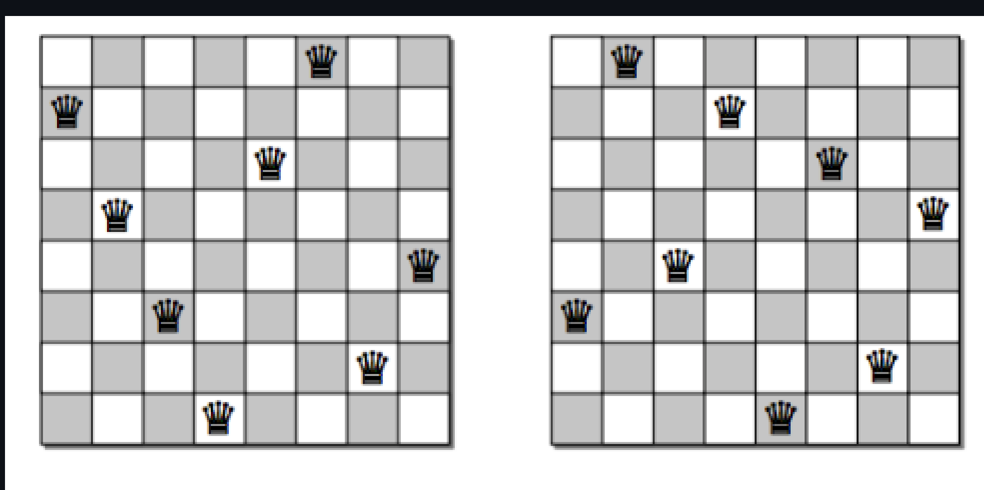


Figure 10.17: Two solutions for the eight-queens problem.

Representation

There are several alternatives:

- Binary representation.

  It's the original approach; the implementation is an array of bools.

- Integer representation.

  Integer array.

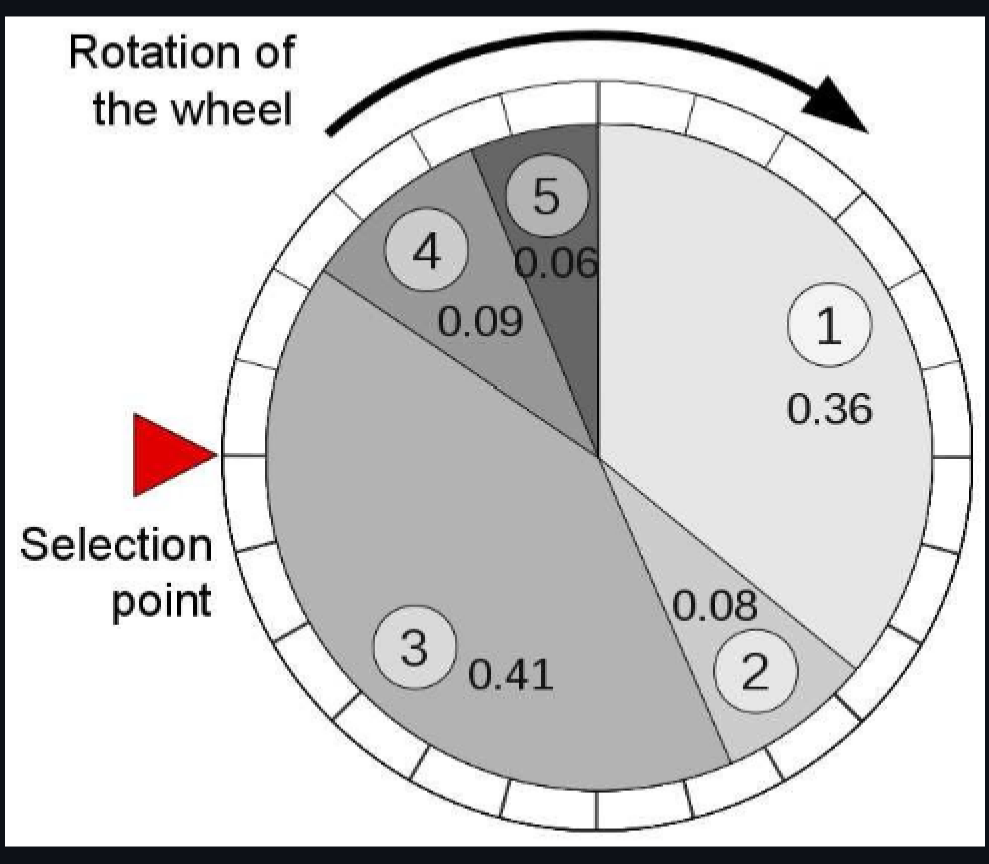
- Real representation.

  Array x where each x_i is a real number.

Parent selection technique

- Roulette/Proportional selection

  We can think of a roulette that has one slice per chromosome of the population. The size of the slice varies depending on the fitnesses; the better it is, the wider its corresponding slice is.



- Tournament selection

  Consists of choosing k random elements and selecting the fittest one.

Crossover/Recombination

The goal is to generate an offspring combining the parents' properties. There are different approaches depending on the representation.

Binary and integer representation

- 1 point crossover.

  Consists of choosing a random pivot point and the new individual will be generated with the left side of the first parent and the right side of the second.

- N points crossover.

  Same idea as 1 point crossover but several sections are used.

- Uniform crossover.

  The new individual is created element by element, randomly picking values from one parent or the other.

Real valued representation

- Discrete reproduction.

  Same as uniform crossover.

- Asymmetric reproduction.

  The offspring is generated with a weighted sum.

$$o_i = \alpha p_1 + (1 - \alpha) p_2$$

  Where α is a value between 0 and 1.

Mutation

The goal is to modify individuals in order to explore the search space. Some of the most used techniques are:

- Bitwise mutation.

  Consists of choosing 1 or more genes and changing their values.

- Random resetting.

  Consists of randomly choosing 1 or more genes and reset their values.

- Uniform mutation.

  Consists of randomly choosing 1 or more genes and replace their values by a number within the constraints.

- Swap mutation.

  Used for the permutation representation and consists of selecting two genes and swapping their values.

Pseudocode

Genetic algorithm

```
Parameters:
    N -> Population size
    G -> Maximum number of generations
    Pr -> Reproduction probability
    Pm -> Mutation probability
Return: The elite individual

Begin
    Create the initial population
    Calculate the population fitness
    Get the elite
    While the number of generations is less than G or a good solution hasn't been found
        Select the parents
        Apply crossover
        Apply mutation
        Calculate the population fitness
        Get the elite or include the elite in the population
    End while
End
```

Back to index

Evolutionary algorithms

Genetic programming

Description

The original goal was to evolve computational programs using syntax trees to solve a problem given a dataset.

This algorithm could instead be positioned in machine learning.

Types of problems

Supervised learning problem.

Representation

The individuals are usually represented as syntax trees.

So, the elements of each individual are:

- Terminals. Leaf nodes in the syntax tree.
- Functions. Internal nodes of a syntax tree, they can be seen as operations.

Initial population

There are three ways to create the very first population.

- Full.

  A set of trees are created with a given depth.

- Grow.

  Each node randomly selects elements from either the set of Functions or the set of Terminals.

- Half-and-half.

  As its name says, it's a mixture of full and grow.

Parent selection technique

Tournament selection is the most used for this technique.

Crossover/Recombination

The classic crossover consists of randomly selecting a crossover point in each parent and using the subtree of the second parent instead of the first's.

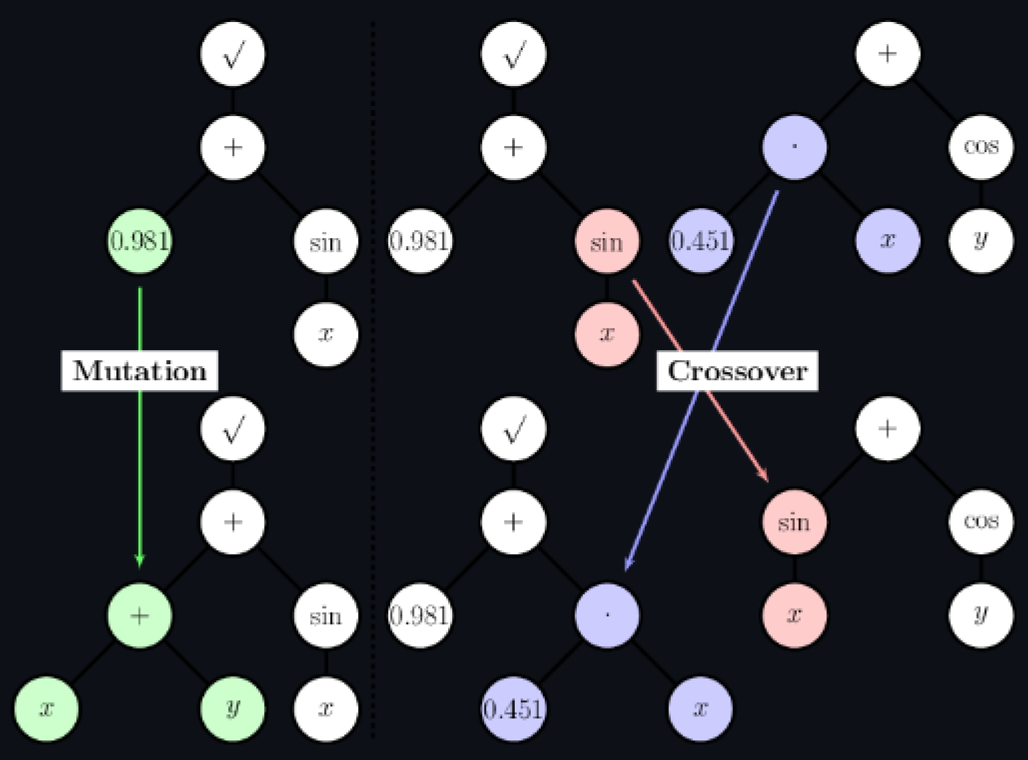
Mutation

- Subtree mutation

  Randomly selects a mutation point in a tree and replaces the subtree for a random one.

- Point mutation

  Consists of replacing a function F for another one with the same arity.



Pseudocode

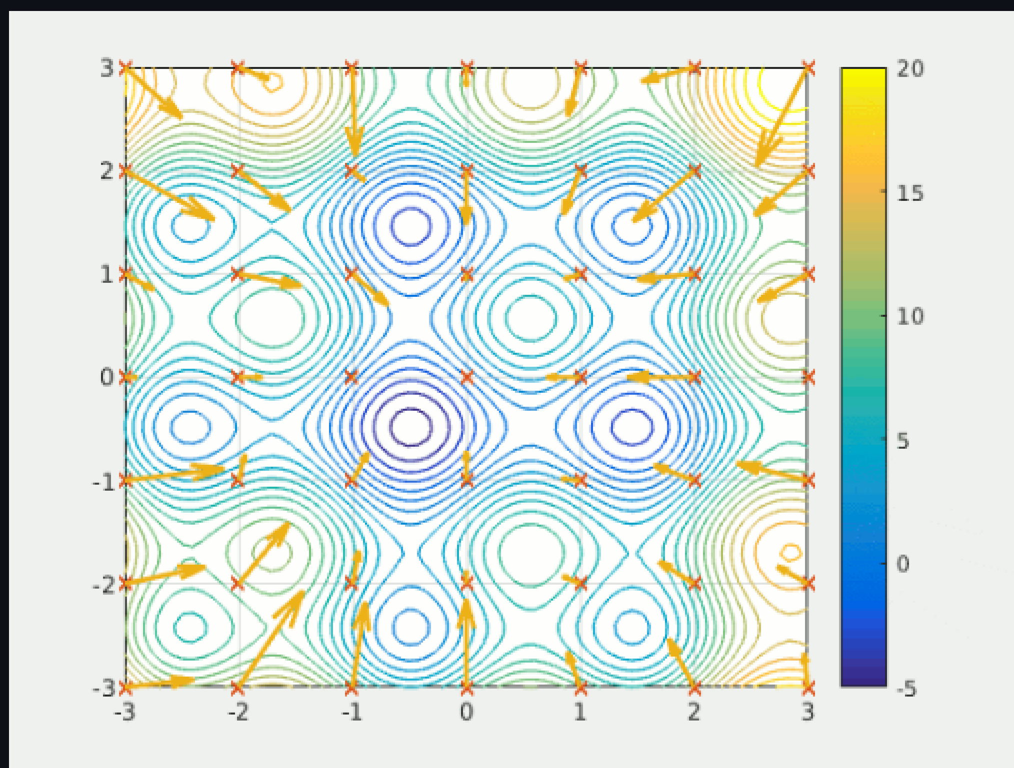
Genetic programming

```
Parameters:
N -> Population size

Return: Best program

Begin
    Create an initial population P of programs
    Calculate the fitness of all programs
    Get the best program
    While a termination criterion is not reached
        Parent selection
        Crossover
        Mutation
        Replace the worst one with the one just created
        Get new fitness
        if fitness(new) > fitness(best)
            Update best
        End if
    End while
End
```

Back to index

Metaheuristics

Particle swarm optimization



Description

Inspired by the movement of a flock when searching for food.

Types of problems

Continuous multidimensional problem optimization.

Representation

Each particle represents a solution. And at the time t, a particle has a vector of positions and another one for velocities.

$$\vec{x^i(t)} =< x_1^i, x_2^i, \ldots, x_d^i >$$

$$\vec{v^i(t)} =< v_1^i, v_2^i, \ldots, v_d^i >$$

Update

At each iteration, both the positions are updated.

Considering that P_{best}^i is the best position where a particle i has been, G_{best}^i is the global best location, r_1 and r_2 are random numbers between 0 and 1, and w, c_1 and c_2 are hyper parameters that can be initiliazed at 0.9 and gradually reduced to 0.1.

$$v^i(t+1) = wv^i(t) + c_1 r_1 (P_{best}^i - x^i(t)) + c_2 r_2 (G_{best} - x^i(t))$$

$$x^i(t+1) = x^i(t) + v^i(t+1)$$

Pseudocode

Particle swarm optimization

```
Parameters:
    N -> Number of particles
    maxIter -> Maximum number of iterations
    func -> Objective function
    bounds -> The search space

Return: the best position

Begin
    Initialize hyperparameters
    Create the particles positions and velocities randomly
    Calculate the objective funcion values
    Calculate the best position of each particle
    Calcule the best position
    While t < maxIter and a good solution hasn't been found
        For each particle i
            Update velocity
            Update position
            Calculate fitness
            Update best position of current particle
            Update best position
        End for
    End while
End
```