Bio-Inspired Algorithms

Evolutionary Algorithms

Daniel Vaz
daniel.ramosvaz@esiee.fr

2026-03-26

Ant Colony Optimization

Summary

Ant Colony Optimization

- Meta-heuristic based on a population of ants
- Ants move according to pheromones
- Pheromones evaporate and are deposited by ants
- Creates differentiation by quantity of the pheromones

Approaches

- Ant system, elitist and ranked strategies, MMAS and ACS

Application Principles

- Applications: TSP, scheduling, set cover, network routing
- Different applications require different adjustments
    - Solutions and pheromones, balance, local search, heuristic

Ant Movement (Ant System)

From u, move along uv with probability

$$p_{uv} = \frac{\tau_{uv}^{\alpha} \cdot \eta_{uv}^{\beta}}{\sum_w \tau_{uw}^{\alpha} \cdot \eta_{uw}^{\beta}}$$

Parameters

- τ_{uv}: Amount of pheromones on uv
- η_{uv}: natural interest of uv (e.g. length)
  - Maximizing by default, use $\beta < 0$ for minimizing
- α: importance of pheromone
  - Typically 1 to 3; ignored if 0
- β: importance of natural interest
  - Typically 2 to 5; ignored if 0

Pheromone update (Ant System)

Update: done after every round by

- Evaporation of existing
- Deposit of new pheromones

Evaporation

- By exponential decay: $\tau_{uv} \to (1 - \rho)\tau_{uv}$
- ρ is the evaporation rate

Deposit

- As a function of the number (and quality) of paths through uv
- Add $1/f(s)$ for every ant s traversing the edge

Ant System

Solution construction:

- Generate the solution for each ant
- Using the probabilistic rules
- Possibly improve using local search

Pheromone update

- Evaporation: apply decay to pheromones
- Deposit: for every ant, deposit on used edges

Repeat until some condition is achieved

Ant Colony Optimization

Analysis

Convergence Proofs

Some ACO approaches are known to converge optimally

- E.g. ACS, MMAS
- No bound on the number of iterations

Bounds on expected time

- Mostly for polynomial-time-solvable problems
- Shows that the technique is general

Computational Complexity

Ants and parameters

- Running time is proportional to the number of ants
- Parameters affect how long it takes to converge

Use of local search

- Usually applied to every solution
- Potential to increase the running time

Improvements

- Parallel solution construction
- Construct solutions from pieces of previous solutions

Evolutionary Algorithms

Inspiration

The Peppered Moth

The Peppered Moth

Known example of evolution

Before industrial revolution:

- White moth more frequent (90%)
- Nearly invisible on a birch tree

After start of industrial revolution:

- Black moth became more prevalent
- Result of predation of identifiable color

Natural Evolution

Evolution leads to complex and adaptable agents

- At the base of all natural behavior
- And all bio-inspired algorithms

Idea: use this mechanism for algorithms

- To produce good and complex solutions

Principles of Natural Evolution

Individuals are in competition

- Within a population
- Against other species

Survival of the fittest

- Only the best survive
- And are able to reproduce
- Which passes on their characteristics

Positive feedback mechanism

- Good characteristics survive
- Spread through the population

Evolutionary Algorithms

Properties:

- Randomized
- Population-based

Mechanism:

- Variation: produces new individuals
  - Mutation, recombination (reproduction)
- Selection of best individuals
  - For variation and for survival

Approaches

3 independent strategies developed simultaneously:

- Genetic algorithms
  - Use evolution for combinatorial optimization
- Evolution strategies
  - Continuous optimization, with self-adjusting parameters
- Evolutionary programming
  - To design automata or programs
  - (next lecture)

They are all similar

- Nowadays, they are distinguished by solution representation

Motivation

Applications:

- Timetabling of university classes
  - Handles many different requirements
- Industrial design
  - Very performant, but produces strange structures
  - Explores possibilities beyond rational analysis
- Learning classifier systems
  - Uses EA to create sets of rules
  - Unlike machine learning, very transparent
- Simulation
  - Observe the individuals that survive
  - Informs us about what are adaptive individuals

Genetic Algorithms

Genetic Algorithms

Representation

- Solutions represented by an underlying encoding
- Inspiration: genetic material

Algorithm

- Start with a random population
- Every iteration:
  - Recombine some pairs from the population
  - Apply (rare) mutations to the resulting solutions
  - Select the next population

Genetic Algorithms

Algorithm GeneticAlgorithm

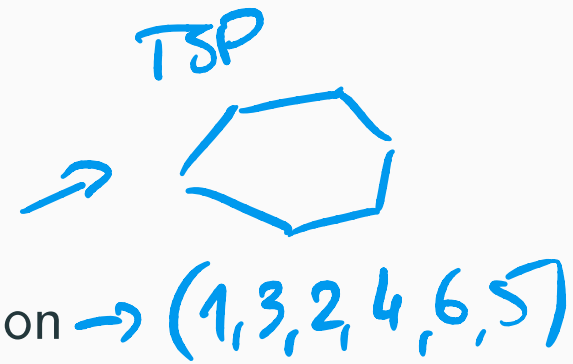
```
1: P := InitializePopulation
2: Evaluate(P)
3: while not Finished do
4:     M := SelectParentPairs(P)
5:     C := Recombine(M)
6:     Mutate(C)
7:     Evaluate(C)
8:     P := SelectNext(P, C)
9: end while
```

Some Notation

Genetic algorithms use specific names

Elements:

- Individual: a solution
- Fitness: objective function (maximizing)
- Phenotype: the characteristics of the solution
- Genome or genotype: the encoding of a solution



Some Notation

Genetic algorithms use specific names

Elements:

- Individual: a solution
- Fitness: objective function (maximizing)
- Phenotype: the characteristics of the solution
- Genome or genotype: the encoding of a solution

Algorithm:

- Population: set of solutions
- Generation: iteration

What do We Need?

Encoding

- Genotype representation of an individual
- How to get the solution from the genotype

Operators

- Mutation
- Recombination
- Individuals formed from recombination, then mutation

Selection

- Parents: which individuals reproduce
- Survivors: which pass to the new generation

Encoding

Encoding: Link between genotype and phenotype

Example: TSP

- Phenotype: tour containing every vertex
- Genotype:
  - Permutation of $[n]$: $(1, 4, 2, 3, 5)$
  - Unordered set of edges: $\{(1, 4), (2, 3), (4, 2), (3, 5), (5, 1)\}$
  - Adjacency matrix of tour "00010 00100 00001 01000 10000"

General principles:

- Simple structure, such as binary
  - Makes operators easier
- Might not be space-optimal

Operators

Operators : How to generate good new individuals

Recombination

- Operator that combines two individuals
- Usually generates one or two new ones
- Intends to aggregate good features
- **Single crossover:** $X[:r] + Y[r:]$, r random

(bit vectors) $Y[:r] + X[r:]$

Operators

Operators : How to generate good new individuals

Recombination

- Operator that combines two individuals
- Usually generates one or two new ones
- Intends to aggregate good features
- **Single crossover:** $X[:r] + Y[r:]$, r random

Mutation

- Modifies an individual
- Small variations to introduce unique features
- **Single bit:** flip bit i for random i

Selection methods

Selection: method to improve population

- Favors good individuals in recombination and new generation

Parent selection

- Which individuals participate in recombination
- Usually random, but using fitness as guide
- **2-tournament:** pick two random individuals, keep best
  - Similar for the second parent

Survivor selection

- Selects which individuals pass to the next generation
- Among previous generation and children
  - Which can be many more than population size
- Uses fitness more directly
- **Best overall:** take the best n out of population+children

Genetic Algorithms

Example

Secret binary vector

Problem: secret binary vector

- Guess a vector $\{0, 1\}^n$ which we do not know
- Fitness: number of bits in common with secret

Encoding: binary string

- Population of 4 individuals

Operators:

- Mutation: bit flip
- Recombination: single-point crossover

Selection:

- Parents: 2-tournament
- Survivors: best overall

Secret binary vector

3 1 3 4

101101, 110101, 111111, 001000

Parent selection: 2-tournament	111|11	00 1 0|00	001|000	10|1|01
Recombination single crossover	111100 001011		001101 101000	
Mutation (prob. 80%) single bit flip	111000	011011	001001	101000
Fitness eval.	4	3	3	5
New population best children + old	101000 5	111000 4	001000 4	001001 3

Genetic Algorithms

Encoding

Encoding

Encoding: Link between genotype and phenotype

- Depends on the problem
- Also on the information we want to change

Discrete representations

- Binary strings, sequence of values, ...
- Ensure that it stays valid (permutation)

Other representations

- Real numbers $\rightarrow$ Evolution strategies
- Decision trees, circuits, automata $\rightarrow$ Genetic programming

Genetic Algorithms

Operators

Operators

Operators : How to generate good new individuals

- Similar to the concept of neighborhood
- Connected to the encoding

Recombination

Recombination: combines two individuals into one

- Can also generate two children
- Goal is to improve fitness by combining good parts
- Should produce valid solutions

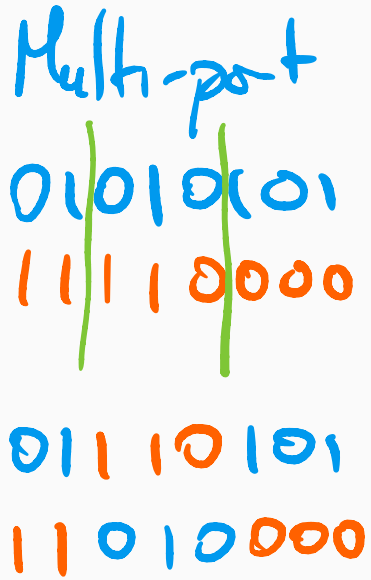
Examples:

- Single-point or multi-point crossover
- Uniform crossover
- Arithmetic crossover
- Permutation insertion

Recombination

Examples: "01010101" + "11110000"

- Single-point or multi-point crossover
- Uniform crossover
- Arithmetic crossover



Uniform

01110001

Arithmetic

01010101
+ 11110000

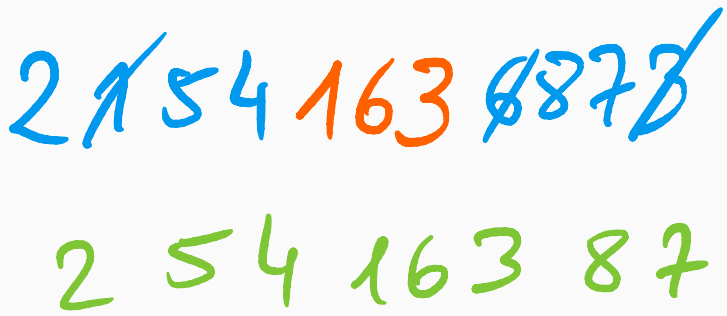
12120101 mod 2

10100101

Recombination

Examples: (2,1,5,4,6,8,7,3) + (4,5,8,1,6,3,7,2)

- Permutation insertion



Recombination Operators

Recombination of parents X, Y

Single-point crossover:

- Pick a random index i
- Swap positions after i from Y to X
- Individuals $X[:i] + Y[i:]$ and $Y[:i] + X[i:]$

Multi-point crossover:

- Similar with $2, 3, \ldots$ cuts
- E.g. $X[:i_1] + Y[i_1:i_2] + X[i_2:]$

Recombination Operators

Recombination of parents X, Y

Uniform crossover:

- For each index, randomly take $X[i]$ or $Y[i]$

Arithmetic crossover:

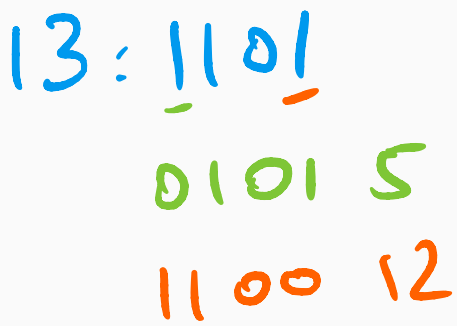
- For each i, take $X[i] + Y[i] \bmod 2$

Permutation insertion:

- Pick a random subsequence S of Y, index of X
- Insert S into X at position i and remove repeated from X

Mutation

- Modifies an individual
- Small variations to introduce unique features
  - Change with small probability to not spoil solutions
- Similar to a neighborhood operator
- Exploration connected to encoding
  - Integers: different bits cause different change



Mutation

Mutation

- Modifies an individual
- Small variations to introduce unique features
  - Change with small probability to not spoil solutions
- Similar to a neighborhood operator
- Exploration connected to encoding
  - Integers: different bits cause different change

Examples

- Single bit flip, independent bit flip
- Exchanges (binary, permutation)
- Partial reversal

Mutation

d

Examples: "01011011" / (3,1,5,6,2,7,4,8)

- Single bit flip, independent bit flip
- Exchanges (binary, permutation)
- Partial reversal

indep. bitflip (prob. 1/d)

00010011

exchange

3 1 7 6 2 5 4 8

reversal

3 1 5 4 7 2 6 8

Mutation Operators

Mutation of genome X

Bit flip: Change the value of bits of the solution

- **Single:** a single bit chosen at random
- **Independent:** each bit flipped with some probability

Exchange (permutation):

- Exchange two elements chosen at random

Partial reversal (permutation):

- Pick two indices i and j at random
- Reverse the permutation between i and j
- New solution: $X[:i] + X[j:i:-1] + X[j:]$

Genetic Algorithms

Selection

Selection

Selection: method to improve population

- Favors good individuals in recombination and new generation
- Maintains genetic diversity

Important properties:

- Selection pressure: how much do we favor the best individuals
  - Influences how quickly we converge
  - Fast convergence can be negative
- Genetic drift: loss of variety over time
  - Influences the quality of solutions
  - Sometimes the best solutions can disappear

Parent Selection

Parent selection: selects individuals for recombination

- Sufficient to select $2m$ parents to get m children
- Balance of selection pressure and genetic drift

Methods

- Proportional selection
  - Roulette method
  - Stochastic universal sampling

Parent Selection

Parent selection: selects individuals for recombination

- Sufficient to select $2m$ parents to get m children
- Balance of selection pressure and genetic drift

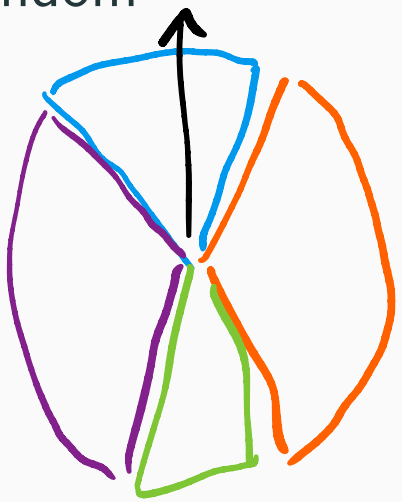
Methods

- Proportional selection
  - Roulette method
  - Stochastic universal sampling
- Rank selection
- Tournament selection

Roulette method

Method

- Probability proportional to the fitness
  - Weight of individual i: $p_i := f(x_i) / \sum_j f(x_j)$
- Select each parent independently at random
  - Each i sampled with probability p_i
- Repeat $2m$ times



Roulette method

Method

- Probability proportional to the fitness
  - Weight of individual i: $p_i := f(x_i) / \sum_j f(x_j)$
- Select each parent independently at random
  - Each i sampled with probability p_i
- Repeat $2m$ times

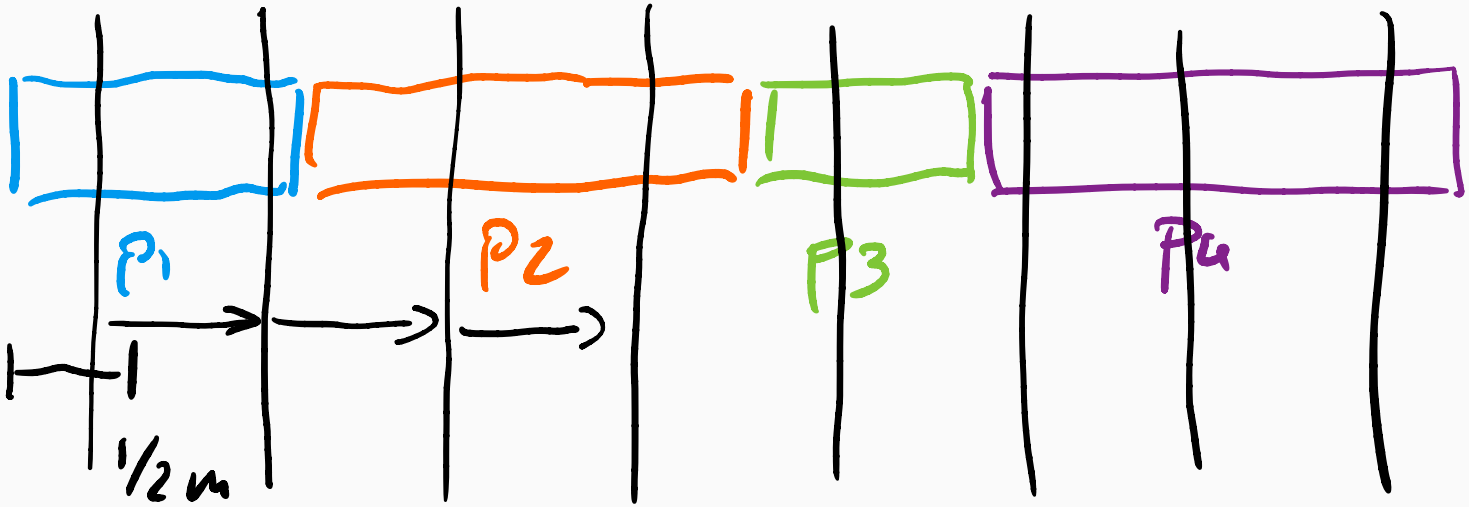
Properties

- Too random, leads to genetic drift

Stochastic universal sampling

Method

- Probability proportional to the fitness
  - Weight of individual i: $p_i := f(x_i)/\sum_j f(x_j)$
- Subdivide interval $[0, 1]$ into individuals
  - $[0, p_1) : 1, [p_1, p_1 + p_2) : 2, [p_1 + p_2, p_1 + p_2 + p_3) : 3, \ldots$
- Equally spaced individuals starting at random $r \in [0, 1/(2m))$
  - Take individuals for $r, r + 1/2m, r + 2/2m, \ldots r + (2m - 1)/2m$



Stochastic universal sampling

Method

- Probability proportional to the fitness
  - Weight of individual i: $p_i := f(x_i)/\sum_j f(x_j)$
- Subdivide interval $[0, 1]$ into individuals
  - $[0, p_1) : 1, [p_1, p_1 + p_2) : 2, [p_1 + p_2, p_1 + p_2 + p_3) : 3, \ldots$
- Equally spaced individuals starting at random $r \in [0, 1/(2m))$
  - Take individuals for $r, r + 1/2m, r + 2/2m, \ldots r + (2m - 1)/2m$

Properties

- Best individual is always kept
- Weak variability
- Needs a single random number

Proportional selection

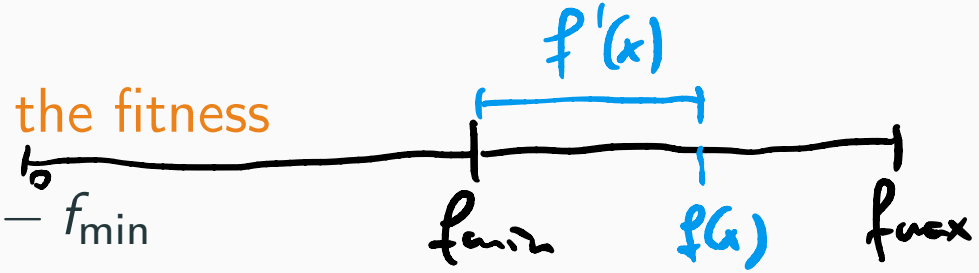
Proportional selection: roulette and stochastic universal sampling

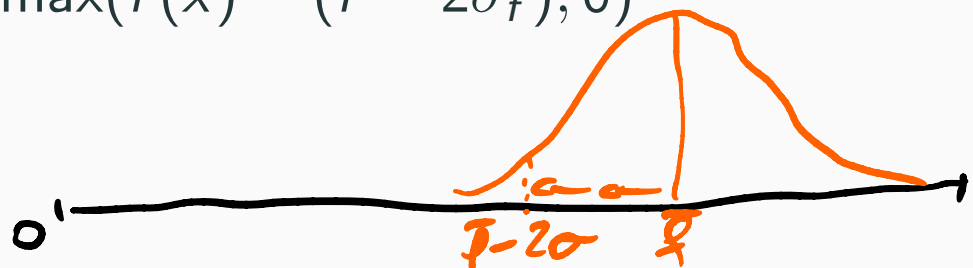
Disadvantage: Selection pressure depends on the values

- Given by ratio of maximum to average
- Weak if values are all similar

Possible improvements:

- Apply a transformation to the fitness
- Windowing: $f'(x) = f(x) - f_{\text{min}}$
- Sigma scaling: $f'(x) = \max(f(x) - (\bar{f} - 2\sigma_f), 0)$





Rank selection

Method

- Sort individuals by fitness
  - Each has a rank r_i in the order
  - Select proportionally to $v_i := (1 - (r_i - 1)/n)^\lambda$
  - λ: controls selection pressure

Properties

- Selection independent of fitness values
- Potentially strong selection pressure

Tournament selection

Method

- Each parent is selected by a tournament
  - Small set of k individuals (usually $k \in \{2, 3\}$)
  - Best one is selected, sometimes best two ($k = 5$)
- Tournament construction
  - Divided: population divided equally
  - Sampled: Each tournament is a sample
  - With or without repetition
  - Stochastic: best wins with some probability p

Tournament selection

Method

- Each parent is selected by a tournament
  - Small set of k individuals (usually $k \in \{2, 3\}$)
  - Best one is selected, sometimes best two ($k = 5$)
- Tournament construction
  - Divided: population divided equally
  - Sampled: Each tournament is a sample
  - With or without repetition
  - Stochastic: best wins with some probability p

Properties

- Independent of fitness values
- k, p control selection pressure
  - Usually k quite small
- Low genetic drift

Parent Selection

Parent selection: selects individuals for recombination

- Sufficient to select $2m$ parents to get m children
- Balance of selection pressure and genetic drift

Methods

- Proportional selection
  - Roulette method
  - Stochastic universal sampling
- Rank selection
- Tournament selection

Survivor Selection

Survivor selection: selects next generation

- Among previous generation and children
  - Which can be many more than population size
- Uses fitness more directly

Methods

- Age-based:
  - m children replace m oldest individuals
  - Generational: $m = n$ (common)
- Fitness-based:
  - Replace worst m individuals of populations
    - Usually with large populations and/or no duplicates
  - Elitism: keep best individual
  - $(\mu + \lambda)$-selection: best n out of population+children
  - (μ, λ)-selection: best n children

Applications

TSP

Genotype: permutation

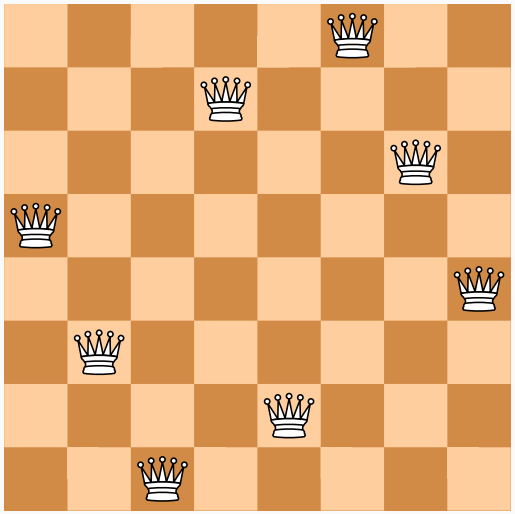
Mutation: exchange two vertices

Recombination: permutation insertion

The Eight-Queens Problem

Problem:

- Place n queens in an $n \times n$ chessboard
- One queen per row, column, diagonal



Algorithm

- Representation: permutation
- Recombination: Permutation insertion
- Mutation: Swap (prob. 80%)
- Parents: 2 out of random 5
- Survivor: Replace worst

Knapsack Problem

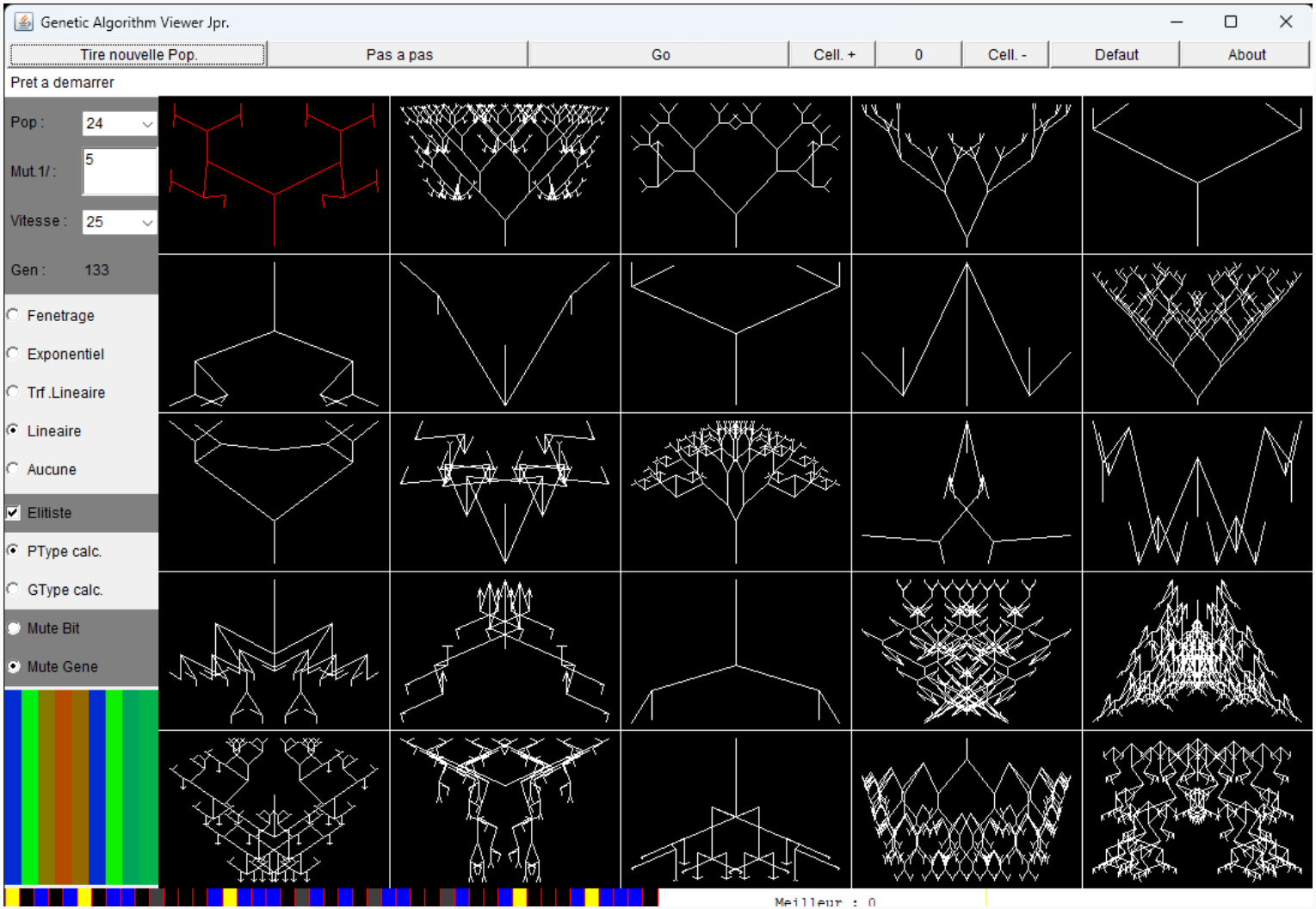
Problem:

- n items, each with weight w_i, value v_i
- Choose items with maximum total value
- Such that total weight $\leq B$

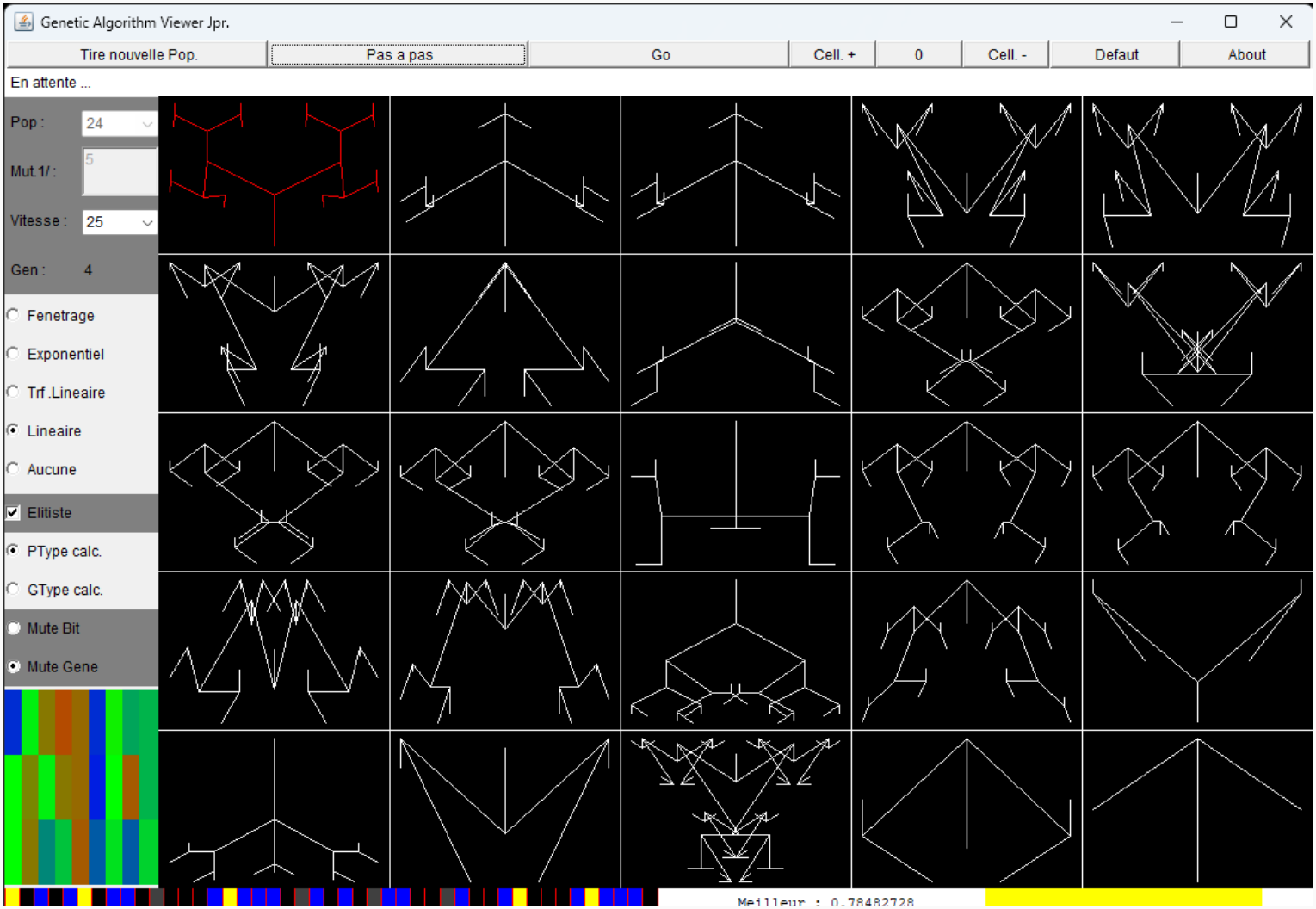
Algorithm

- Representation: binary string of length n
- Recombination: single-point crossover (prob. 70%)
- Mutation: independent bit flip (prob. $1/n$)
- Parents: best out of random 2
- Survivor: generational

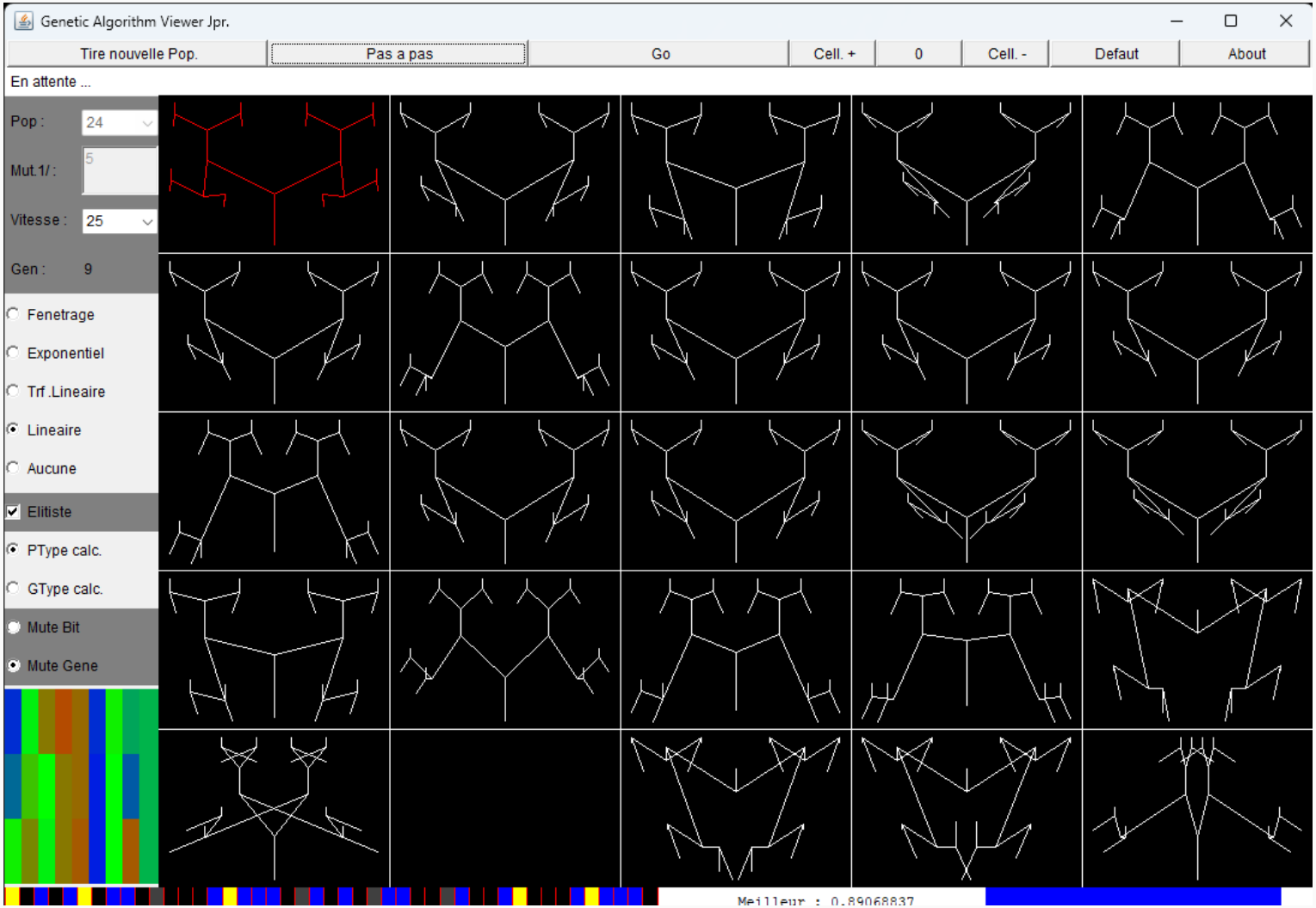
Structure Approximation



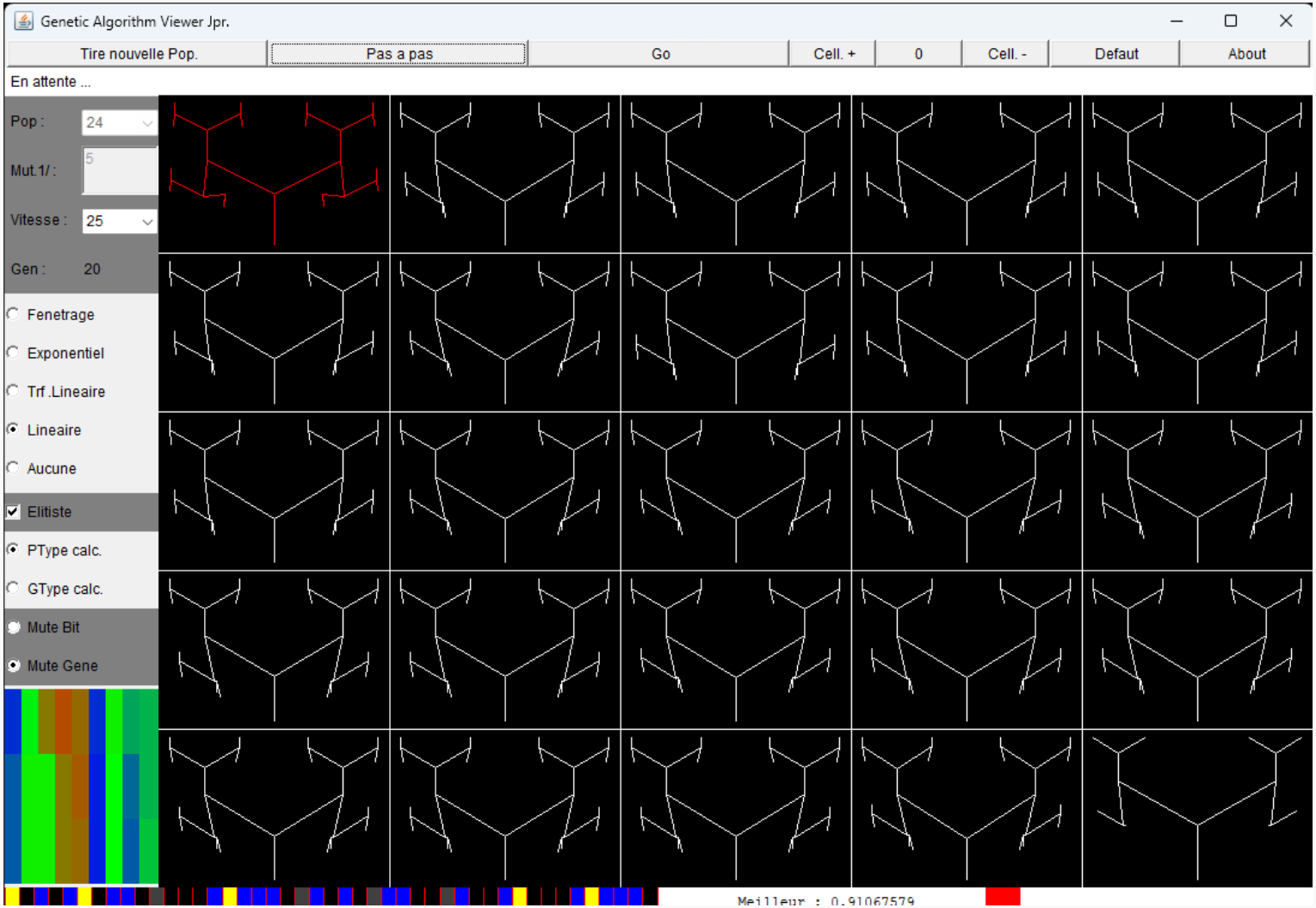
Structure Approximation



Structure Approximation



Structure Approximation



Conclusion

Summary

Evolutionary Algorithms

- Based on natural evolution
- Survival of the fittest
- Population-based

Genetic Algorithms

- Encode individuals using a genotype
- Evolve population over generations
    - Using recombination and mutation, then selection

Applications

- TSP, Eight-Queens, Knapsack, Structure Approximation