MC906/MO416 Introduction to AI

Lecture 7

UNICAMP

Introduction to AI

Lecture 7 - Evolutionary Computing

Profa. Dra. Esther Luna Colombini
esther@ic.unicamp.br

Prof. Dr. Alexandre Simões
alexandre.simoes@unesp.br

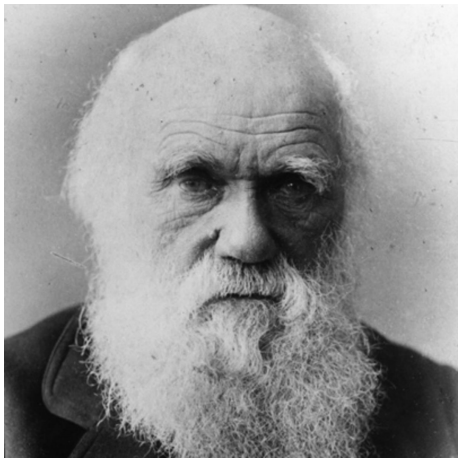
LaRoCS – Laboratory of Robotics and Cognitive Systems

Summary

- Theory of Evolution
- Evolutionary Computing
- Evolutionary Algorithms
- Evolutionary Strategy and Evolutionary Programming
- Genetic Algorithms
- Applications

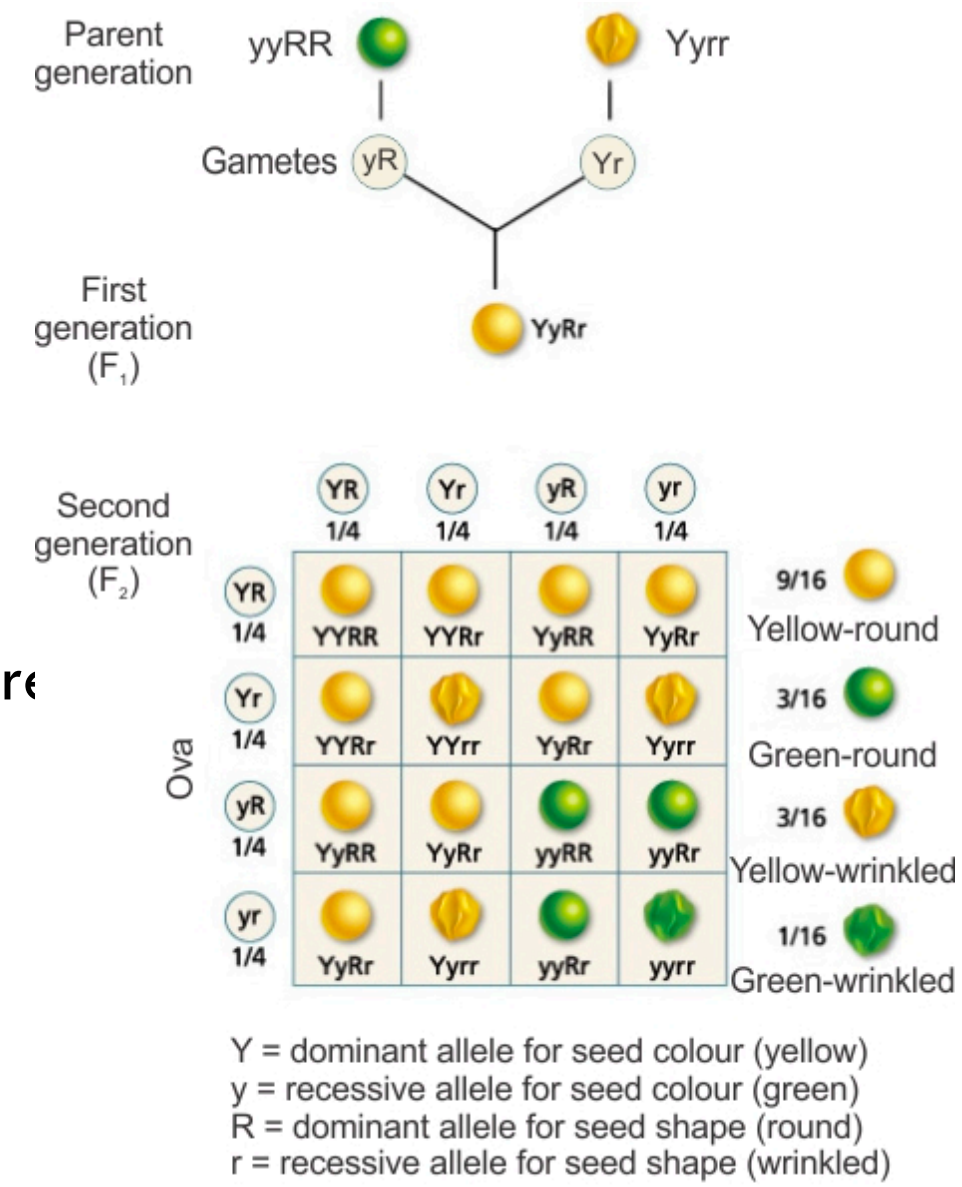
- 1809: Jean-Baptiste Lamarck
  - Law of use and disuse
    - The environment changes -> the individual adapts
    - Through the use and disuse of their aptitudes, nature forces beings to adapt to survive
  - Law of acquired characters
    - Changes in the body's body caused by use or disuse are transmitted to the descendants

- 1859: Charles Darwin
  - There is a diversity of beings due to the contingents of nature (food, climate, ...) and it is by the law of Natural Selection that the beings most adapted to their environments survive
    - against law of the use of disuse
  - The acquired characters are inherited by the following generations



"The better an individual adapts to his environment, the greater his chance of surviving and generating descendants."
(DARWIN, 1859)

- 1865: Gregor Mendel
  - Formalized the inheritance of characteristics, with the theory of DNA (peas)
- 1901: Hugo De Vries
  - Only natural selection is not responsible for the production of new (more adapted) species. There has to be a genetic shift!
  - Formalized the process of generating diversity:
    - Mutation Theory



Optimization

- Numerical optimization
  - operates on the return value of mathematical functions
  - Ex: the discovery of the value of x that maximizes the function
    - $f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$
- Combinatorial optimization
  - aims to discover the best combination of available resources to optimize its use.
  - Ex: what is the best way to allocate competing processes on a CPU, what is the best sequence of cities for the traveling salesman problem
- The evolution of species can be seen as an optimization process
  - over time, living beings become increasingly adapted to an ever-changing environment.

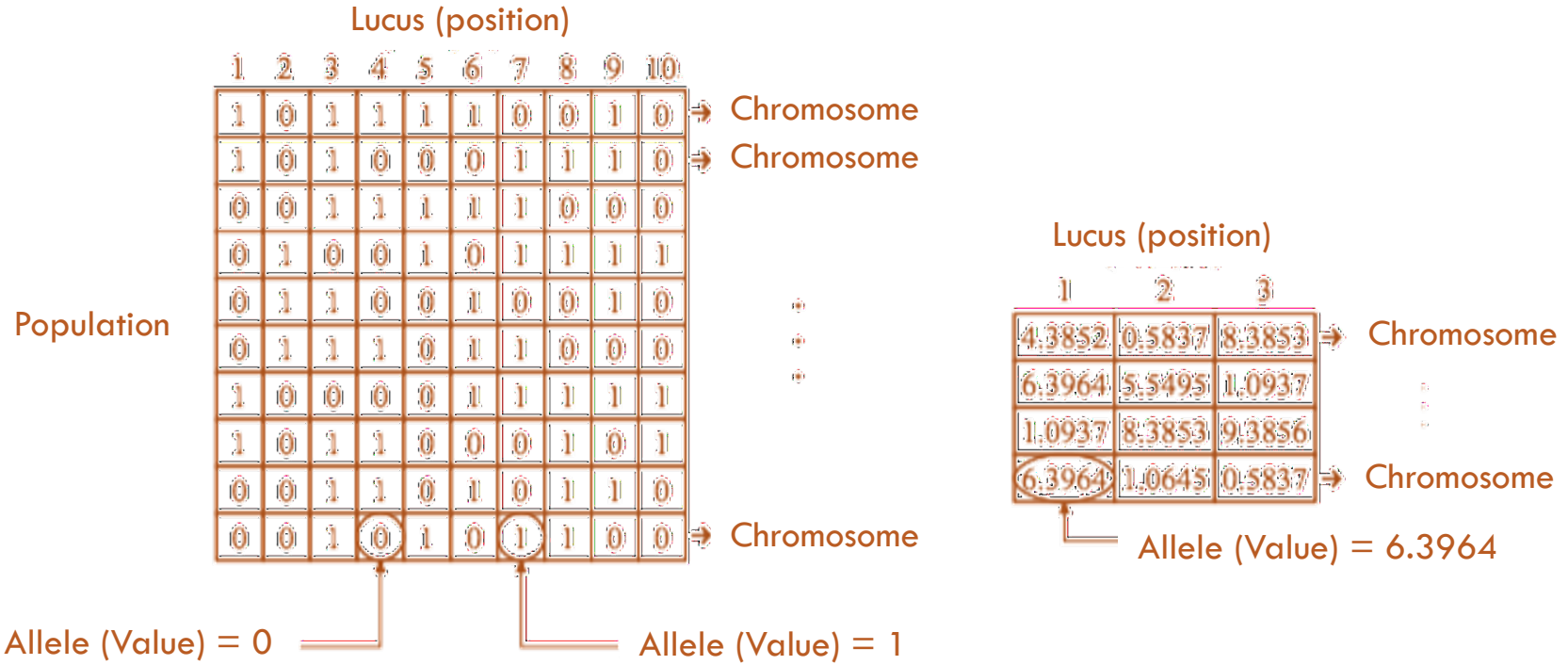
Evolution and Optimization

- Why evolution?
  - Many computational problems
    - involve searching through a large number of possible solutions
    - require the program to be adaptive, able to act in a dynamic environment
- Biological evolution
  - a parallel search in a huge space of problems
  - desired solutions = more adapted organisms

- **Chromosomes:** structure that encodes organisms. Often, the chromosome represents the **individual** in EC
  - The **genotype** is defined by the individual's set of chromosomes
    - It consists of the information present in the data structure that encompasses the genes of an individual
  - The **phenotype** are the characteristics of the organism generated based on the genotype constitute
    - is the result of an individual's genome decoding process

Genotype	Phenotype	Problem
0010101001110101	10869	Numerical optimization
CGDEHABF	Start in city C, then go to G, D, E, H, A, B and finish in F	Traveling salesman
C1R4C2R6C4R1	if condition 1(C1) execute rule 4 (R4), if (C2) execute (R6), if (C4) execute (R1)	Learning rules for agents

- Chromosomes are encoded in a set of symbols called **genes**
  - Individuals or **chromosomes**: corresponds to a data structure that represents or encodes a point in a search space
  - **Alleles**: different values of a gene
  - **Locus**: position of the gene on a chromosome



Basic Principles

- Populations of individuals
  - set of individual candidates for a solution
- **Aptitude** reflects the individual's ability to survive and reproduce. Note associated with the individual who evaluates how good the solution represented by him is.
  - Can be:
    - Same as objective function
    - Result of scheduling the objective function
    - Based on the ranking of the individual of the population

Basic Principles

- Dynamic changes in populations
  - due to the birth and death of individuals
- Variability and heredity
  - new individuals have many of the characteristics of their parents, although they are not identical
  - variability occurs via reproduction or mutation
- Reproduction with genetic inheritance:
  - Sexed: there is exchange of genetic material via crossover or recombination between parents
  - Cloning or Asexual: no exchange of genetic material

- **Selection**
  - Imitates natural selection
  - The best individuals (greatest aptitude) are selected to produce children
  - The probability that a given solution i will be selected is proportional to its suitability

Evolutionary Computing

- Paradigm for solving problems
- Does not require, in order to solve a problem, prior knowledge of a way to find the solution
- Based on evolutionary mechanisms of nature:
  - self-organization and adaptive behavior
- Give up optimal solutions to allow problems to be addressed
- Makes use of evolutionary algorithms
  - Iterative search (optimization) procedure inspired by biological evolutionary mechanisms
- Evolutionary computing is the name used to describe the line of research that deals with evolutionary algorithms

Evolutionary Computing: Characteristics

- It has good exploratory capacity of the search space
- It has been used in several areas of knowledge and has been successful
- Easily parallelable

- Evolutionary algorithms are classified into:
  - Evolutionary Strategies (EE)
    - Developed in the 1960s to optimize real value parameters in dynamic systems
  - Evolutionary Programming (PE)
    - Developed in 1966 to optimize real value parameters in dynamic systems
  - Genetic Algorithms (AG)
    - Developed in the late 1960s as a more general problem-solving strategy
  - Genetic Programming (PG)
    - Developed in the early 1990s as an extension of AGs, where, instead of presenting individuals as fixed-length chromosomal chains, they are actually programs that, when executed, represent candidates for solving the problem
- Differences
  - Representation (encoding): which data structure
  - Genetic operators: crossover and / or mutation
  - Selection operators: deterministic or probabilistic

Evolutionary Algorithms: Comparative

Original propositions of the algorithms

	AG	EE	PE	PG
Representation	Binary strings (original) Real Vectors	Real vectors	Real vectors	Trees
Fitness	Stepped value of the objective function	Objective function value	Value (scaled) of the objective function	Stepped value of the objective function
Mutation	Secondary operator	Main operator	Only operator	One of the operators
Recombination	Main operator	Different variations	None	One of the operators
Selection	Probabilistic	Deterministic	Probabilistic	Probabilistic

Source: Leandro de Castro/Fernando Von Zuben

Evolutionary Algorithms: Standard Algorithm


```
evolutionary program procedure
t = 0
Initialize_Population P(0)
While not (wall_condition) do
      Evaluate_Population P(t)
      P'= Select_Parents P(t)
      P' = Recombination_and_mutation P'
      Evaluate_Population P'
      P(t+1) = Select_s survivors P(t) from P '
      t = t + 1
End
```

Evolutionary Strategy and Evolutionary Programming

Evolutionary Strategy

- Mainly employ mutation operators (Dist. Normal)
- You don't need a lot of information about the problem
- Widely used in parameter optimization (multi-dimensional, multi-modal and non-linear problems)
- Basic algorithm
  - Population with m individuals. Each has n genes
  - Each individual produces k/m descendants with small changes (mutations)
  - Only the best m of the k generated remain alive
  - The size of the population of parents and descendants may be different

Evolutionary Strategy

- Given a population of μ individuals, λ children will be generated. Strategies vary:
  - (μ + λ)-EE: de (μ + λ) individuals, μ will be selected.
  - (μ , λ)-EE: de λ individuals, μ will be selected (λ ≥ μ)
- The various types of evolutionary strategies:
  - Two-member: (1 + 1)-EE
    - The son (mutated individual) is accepted into the new generation if and only if he has better fitness than the father (and is doable)
  - Multi-member: (μ + 1)-EE
    - μ > 1 parents can participate in the generation of a single child
    - The selection operator removes the least fit individual (with the least fitness) from the parents and the generated child
  - Multi-member: (μ + λ)-EE
    - where μ parents produce λ children and the population μ + λ is subsequently reduced to μ individuals. The selection operates in the joint union of parents and children.
  - Multi-member: (μ , λ)-EE
    - only children are selected (for the next generation). That is, the life span of each individual is restricted to one generation

Evolutionary Programming

- Each individual generates a single descendant through mutation
- The best half of the rising population and the best half of the descending population form the new generation
- There is no recombination, only mutation
- Basic algorithm
  - Initial population chosen at random
  - Each solution generates a new solution using mutation
  - Aptitude of each solution is calculated. The fittest are retained
  - After raising μ children from μ parents by mutating each parent, a variation of stochastic selection per tournament selects μ individuals from parents and children

- Tournament selection
  - a group of q individuals is randomly selected, with or without replacement, from the population. This group becomes part of a tournament, that is, a winning individual is determined depending on their fitness value in relation to their competitors.
  - the best individuals (the one with the greatest fitness) is usually chosen in a deterministic or stochastic way. Only these individuals are placed in the population
  - the process is repeated λ times until a new population is generated

Genetic Algorithms

- 1975: Formalized by Jonh Holland
- **Main operator:** crossover
- **Mutation**: secondary operator applied with low probabilities
- Binary representation (originally) of individuals
- Evolved to variations with whole, real representations of strings
- Probabilistic selection proportional to fitness

Genetic Algorithms: Introduction

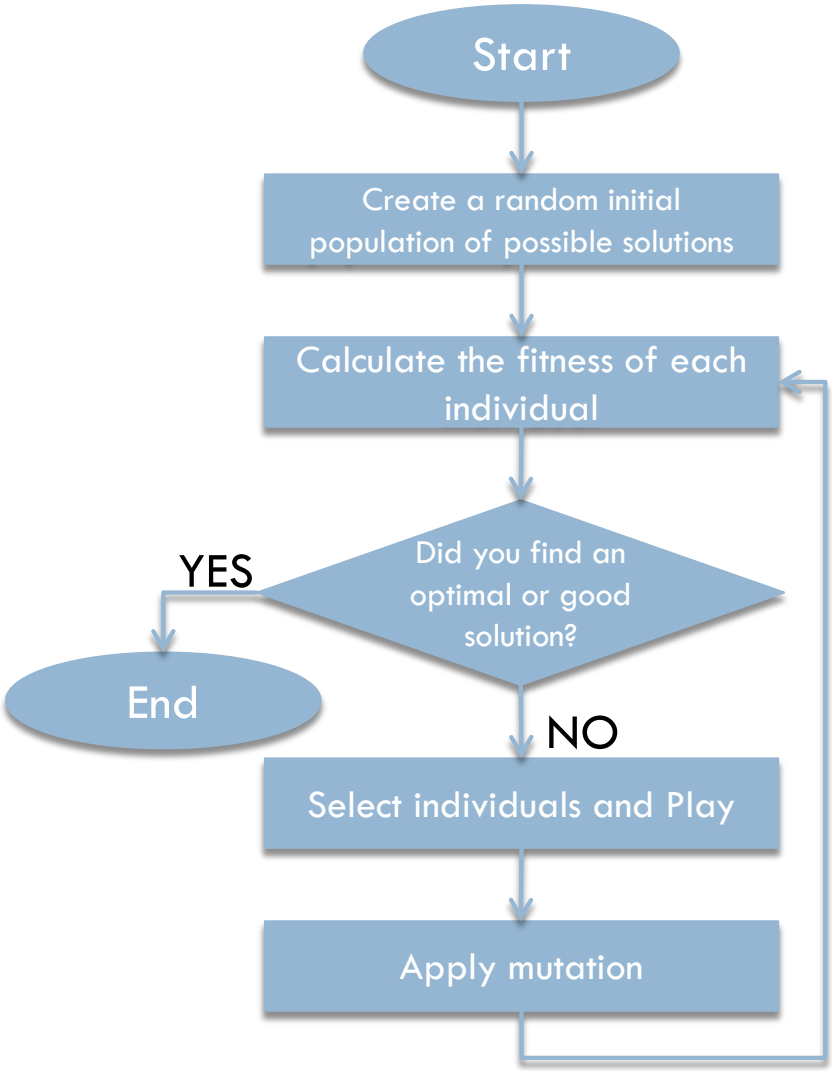
- Genetic Algorithm
  - Individual = solution
  - Causes change in individuals through reproduction and mutation
  - Selects more adapted individuals through successive generations
  - The fitness of each individual is measured by the “fitness function”

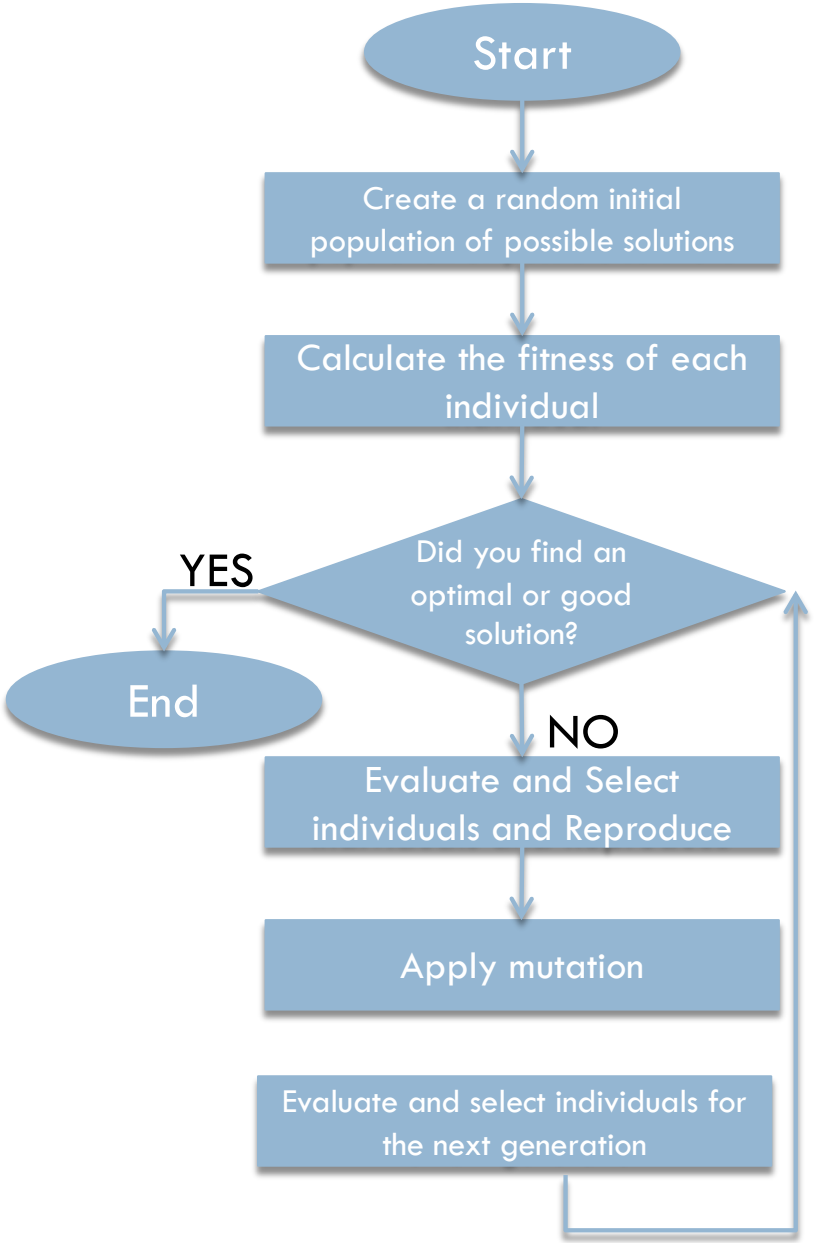
Genetic Algorithms: Characteristics

- Works with a population of solutions simultaneously
- They are easy to implement on computers
- Adapt well to parallel computing
- Easily adapted to other techniques
- Work with continuous or discrete parameters

Genetic Algorithms: Overview

Classic x Modified





GA: Comparation

- Classic AG
  - Binary encoding
  - Reproduction and Natural Selection via roulette algorithm
  - Simple Crossover
  - Mutation
- Modified AG
  - Generation of Sub-populations through operations on members of the initial population and the sub-populations themselves
  - Reproduction and re-classification of the intermediate population
  - Selection for new generation

GA: Basic Concepts

- They manipulate a population of individuals
- Individuals are and should be possible solutions to the problem
- Individuals are combined (crossover) with each other, producing children who may or may not mutate
- Populations evolve through successive generations until they find the optimal solution

GA: Generating Initial Population

- Given a problem, we need to define how individuals will be represented (solutions)
- An individual is a chromosome
  - The parameters of the optimization problem are represented by value chains
  - Examples:
  - Real vectors (2.345, 4.3454, 5.1, 3.4)
  - Bit strings (111011011)
  - Integer vectors (1,4,2,5,2,8)
  - or other data structure (trees, heaps, etc.) -> Genetic Programming

GA: Generating Initial Population

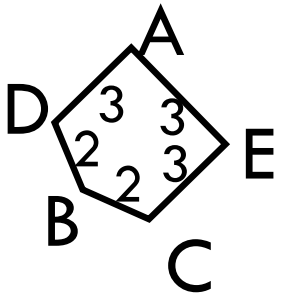
- What should the initial population be?
  - Must be randomly chosen
    - uniform random startup
    - non-uniform random startup
    - random dope initialization
  - Relationships exist between:
    - Convergence speed x Variety
      - The greater the variety, the lower the convergence speed and vice versa
    - Variety x Optimal solution
      - The greater the variety, the greater the chance of finding the optimal solution, however, the lower the convergence speed
- Define fitness role
  - Aptitude is a score associated with the individual who evaluates how good the solution he represents is
  - It can be A(i) = f(i)

GA: Fitness-Fitness Function

- Fitness function (assessment / fitness):
  - It is done through a function that best represents the problem and aims to provide a measure of aptitude for each individual in the current population that will direct the search process
- Assessment functions are specific to each problem
  - Examples

Chromosome	Meaning	Value
0011011	Bynary to Integer	$x = 27$

ADBCE



$A \rightarrow D \rightarrow B \rightarrow C \rightarrow E$

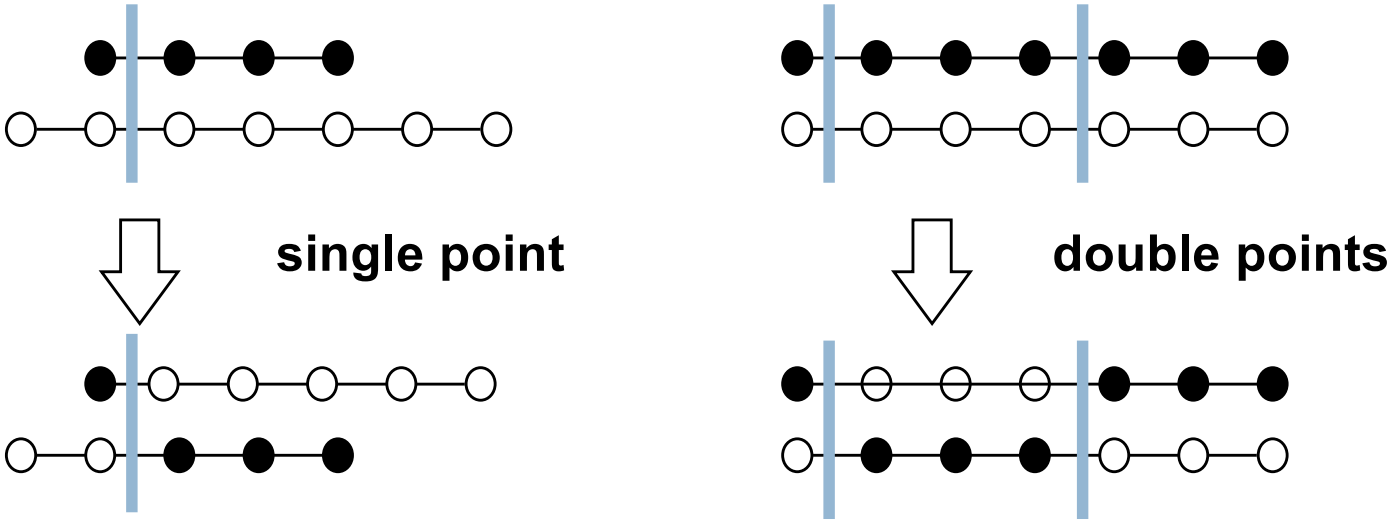
$\sum \text{dist} = 13$

GA: Creating and Evolving Generations

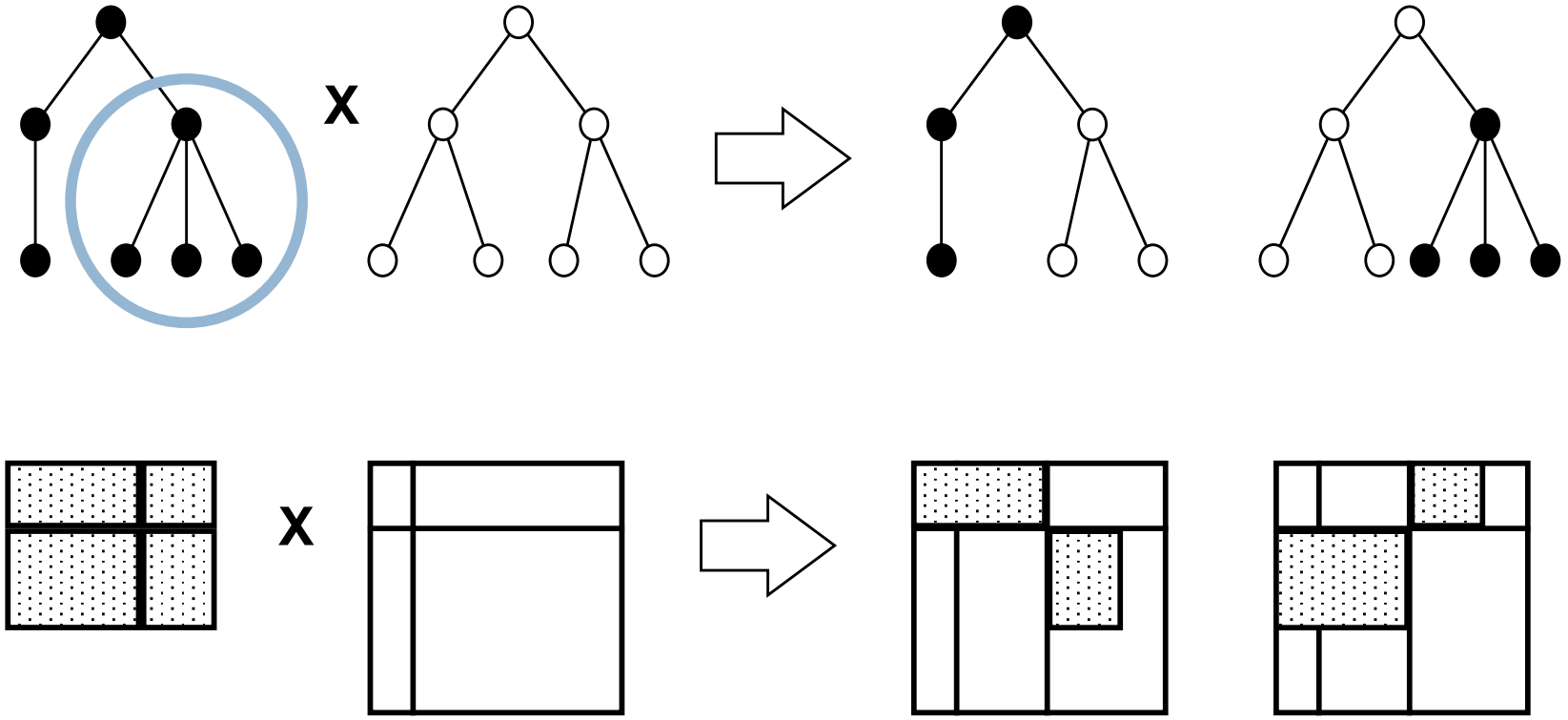
- What are the methods for creating new individuals?
  - Sexual reproduction (crossover)
  - Asexual reproduction
  - Mutation
- How to guarantee to find the solution?
  - Individuals must be selected (the fittest survive) to ensure convergence
  - Premature convergence should be avoided
  - A stopping criterion must be established
- Convergence
  - in the last k generations there has been no improvement in fitness

GA: Reproduction/recombination

- Function: combining and/or perpetuating genetic material of the most adapted individuals
- Types:
  - asexual (= duplication) or sexual (crossover)
  - Note: there is a "crossover fee" that controls the amount of reproduction that will be done

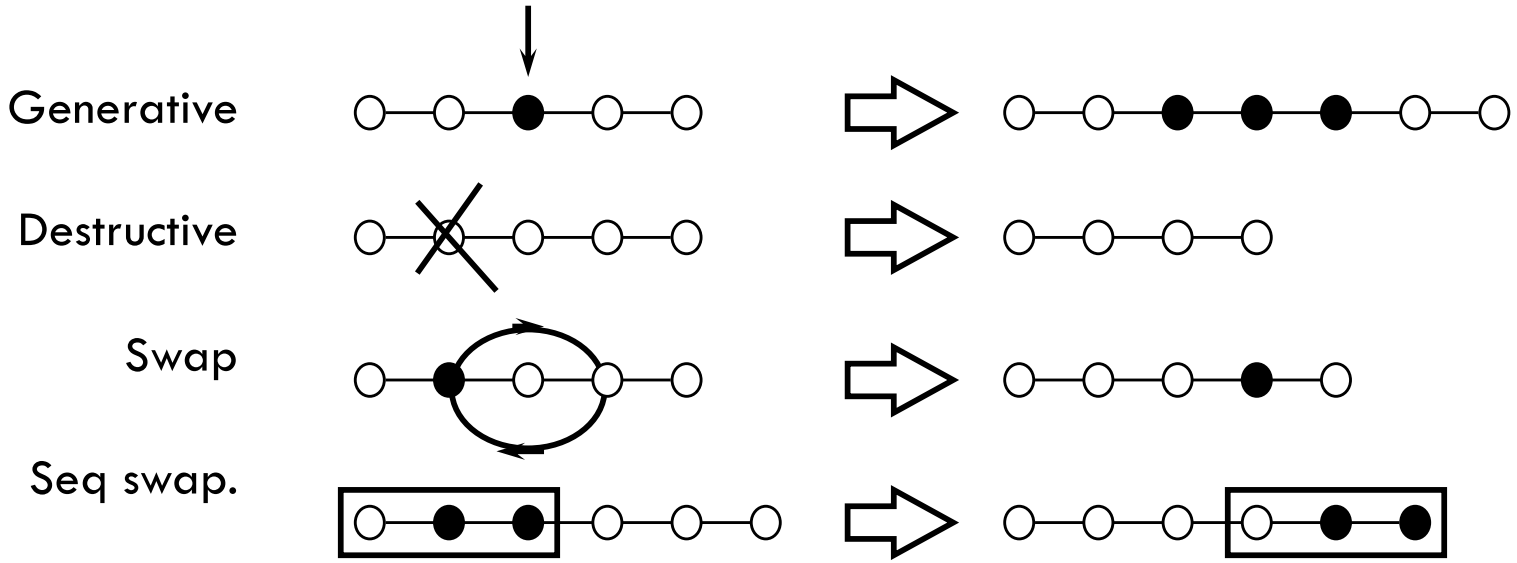


- The more "structured" the representation the more difficult to define the intersection



GA: Mutation

- Objective: to generate diversity (to escape from great places)
- Types: generative, destructive, swap and sequence swap
- Note: There is a "mutation rate" that decreases over time (eg % of the selected population) to ensure convergence



- Other types of mutation
  - **flip**: each gene to be mutated receives a value drawn from the valid alphabet
  - a random value is added to or subtracted from the gene value

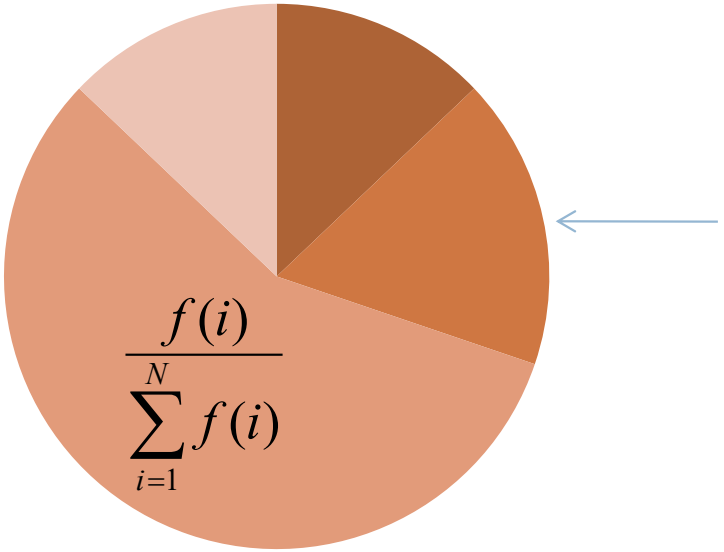
GA: New Generation

- Possible compositions for Population 2:
  - **Exchange of the entire population**
    - This means that population 2 replaces population 1
    - At each cycle, N / 2 pairs are chosen generating M = N descendants
  - **Elitism**
    - Population 2 replaces Population 1 (M = N-1)
    - Add the fittest of population 1 to population 2
  - **Steady State**
    - M <N individuals are generated and these M replace the worst M of the population as a whole 1

- How to select individuals for Breeding?
- **Tournament selection**
  - Randomly choose a number of individuals from the population (tournament size) and make a tournament among them
  - Each tournament consists of comparing the fitness values of the individuals involved, with the winner (and the selected one) being the one with the best fitness value
  - The number of tournaments held is equal to the number of individuals to be selected
  - It does not lead to premature convergence (as long as the size of the tournaments is small)
  - Can be used to define recombination

- **Roulette Method**
  - Select individuals at random, providing reproduction chances to the fittest
  - f(chromosome) = numerical measure of aptitude
  - Selection chances are proportional to aptitude
  - Seeking 4 individuals for reproduction, the roulette wheel is rotated 4 times.
- We could have the following result:

Turn 1 - Individual 4
Turn 2 - Individual 1
Turn 3 - Individual 4
Turn 4 - Individual 3



GA: Roulette Method

- Find the sum of the AT fitness of all individuals in the population: $A_T = \sum_{i=1}^{N} f(i)$
- Generate a random number r:

$$0 \leq r \leq A_T$$

- Select the first individual in the population whose aptitude plus the aptitudes of previous individuals is >= r:

$$\sum_{i=1}^{k \leq N} f(i) \geq r$$

- Example:
  - r = 30

Chromosome	1	2	3	4	5	6	7	8	9	10
Fitness	8	2	17	7	2	12	11	7	3	7
$\sum f(i)$	8	10	27	34	36	48	59	66	69	76

GA: Stopping criteria

- Set a maximum number of generations
- Stop when the system finds the solution (when it is known)
- Stop when loss of diversity occurs (repeat individuals) → Converge !!!

GA: Premature Convergence

- What is Premature Convergence?
  - The system no longer shows any improvement and is still far from the ideal situation. No more diversity of the population
- What can cause Premature Convergence?
  - Several individuals alike (little diversity)
  - A super individual who monopolizes the roulette method
  - A low rate of mutation and / or reproduction
- What can we do to avoid?
  - Changing the fitness function
  - There are techniques that minimize the creation of super-individuals and that increase the selective pressure on the best. They are:
    - Windowing
    - Exponential Transformation
    - Linear Normalization (LN)

GA: Windowing and Exponential Transformation

- Windowing:
  - Get the minimum population rating
  - Discount it from the fitness of each individual
  - The idea is to subtract a constant from f (i), eliminating the worst individual and increasing the chances of the strongest

$$A_i = f(i) - A_{\min}$$

- Exponential Transformation:
  - The new Aptitude f 'value of each individual will be given by:

$$f'(i) = \sqrt[2]{f(i) + 1}$$

- Reduces the influence of the strongest individual

Linear Normalization

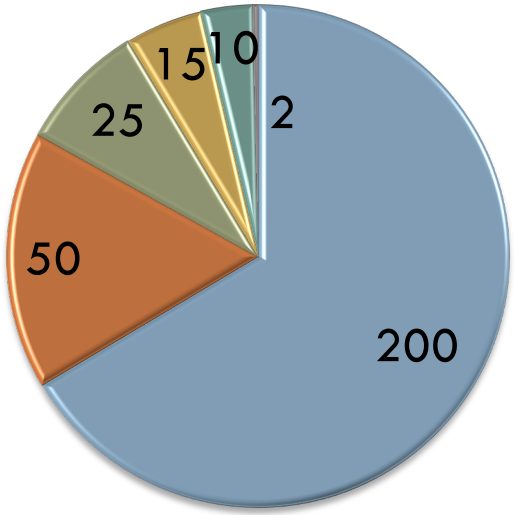
- Order all N individuals in descending order of evaluation (i = 1 less fit)
- Create skills starting from a minimum value and growing linearly to the maximum value for each of the N individuals.

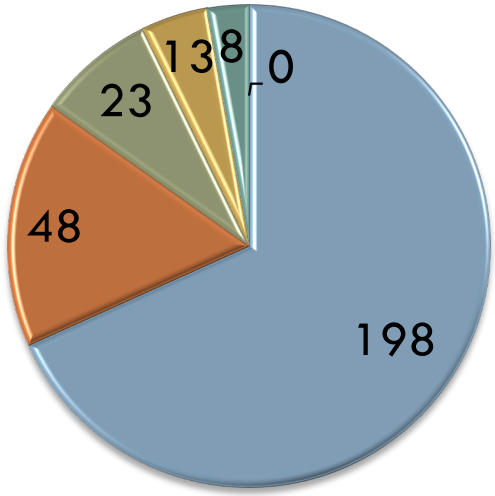
$$A_i = \min + \frac{(\max - \min)}{N-1} \times (i-1)$$

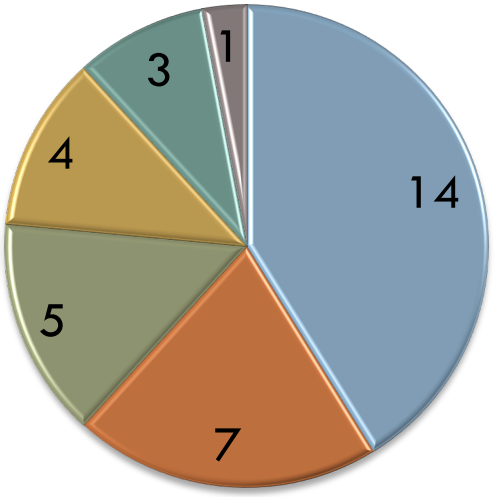
- The max and min values (or the increment constant) are parameters of the technique
- Idea: The greater the increment constant, the greater the selective pressure on the best.

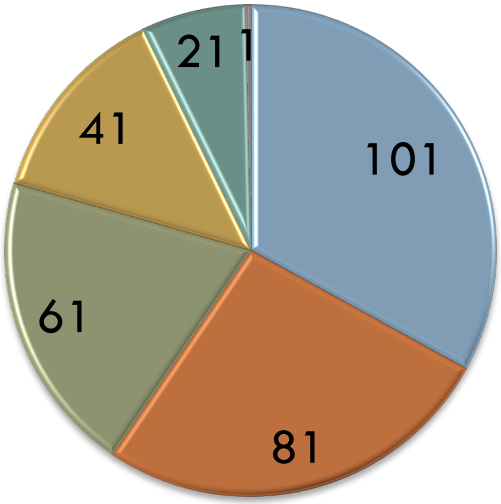
Comparative Example:

Rank	6	5	4	3	2	1
Fitness	**200**	**50**	**25**	**15**	**10**	**2**
Windowing	198	48	23	13	8	0
Exponential Transf.	14	7	5	4	3	1
LN (increment 20)	101	81	61	41	21	1
LN (increment 1)	6	5	4	3	2	1







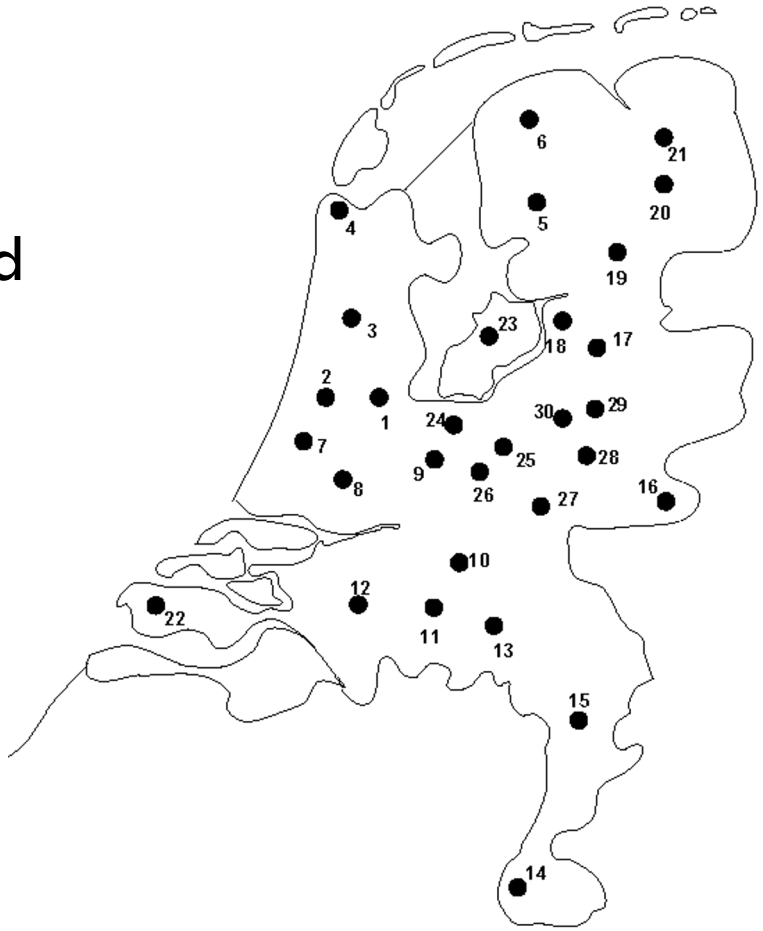


GA: Most usual algorithm

- It has the following characteristics:
  - Chromosome: Binary representation in a vector
  - Reproduction: 1 point crossover
  - New generation: Reproduction (selection) with replacement of the population by Elitism
  - Premature Convergence: Linear Normalization
  - Other details: Mutation is used
  - Performs well in several applications (it is a good starting algorithm)

GA: A simple example

- The Traveling Salesman problem:
  - Find a path between cities so that
  - Each city is visited only once
  - The total travel distance is minimized
- For a problem with 30 cities, we will have something like 30! path possibilities
- $30! \approx 10^{32}$



Example: Representation

- We will consider a simple way of representation:
  - Ordered list of cities visited
  - Each city will be represented by a number

1) São Paulo 2) Campinas 3) SBCampo 4) Sto André 5) Diadema 6) Osasco 7) Guarulhos 8) SCSul

- CityList1 (3 5 7 2 1 6 4 8)
- CityList2 (2 5 7 6 8 1 3 4)

Example: Defining Crossover

- We will use a double point reproduction crossover

Pai1 (3 5 7 2 1 6 4 8)

Pai2 (2 5 7 6 8 1 3 4)

Filho 1 (2 5 7 2 1 6 3 4) -> problem with 2

Filho 2 (3 5 7 6 8 1 4 8) -> problem with 8

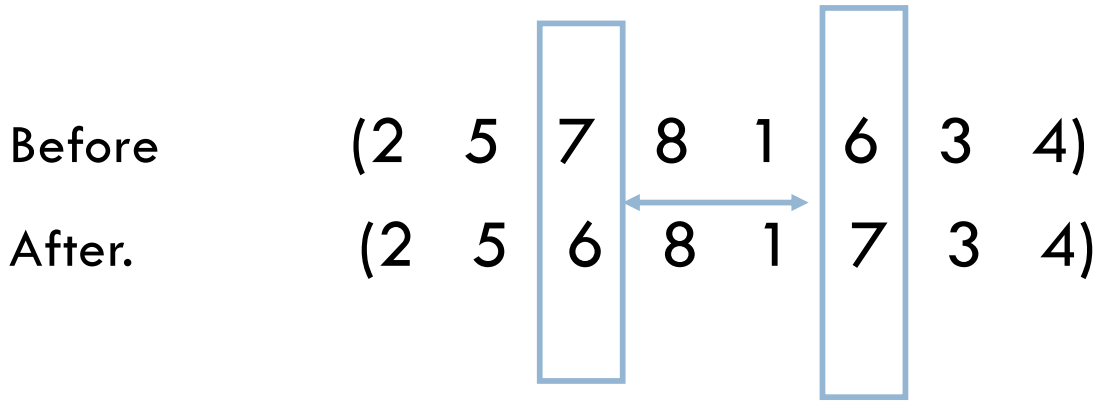
Adjust

Filho 1 (2 5 7 8 1 6 3 4)

Filho 2 (3 5 7 6 2 1 4 8)

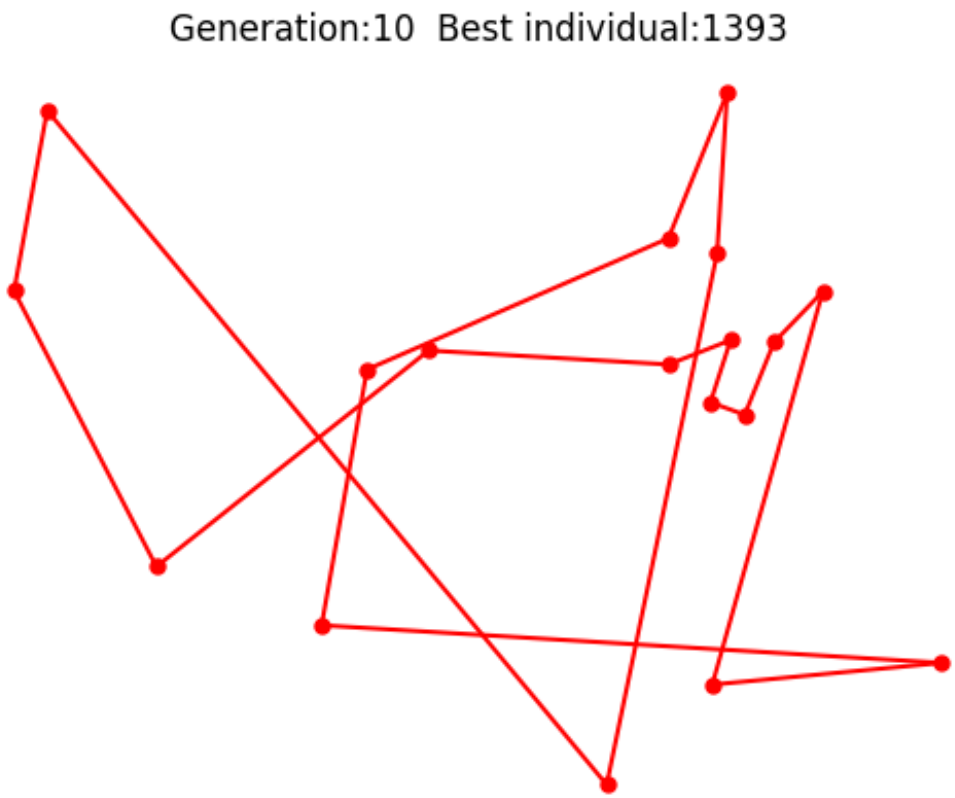
Example: Defining Mutation

- Mutation with swap:



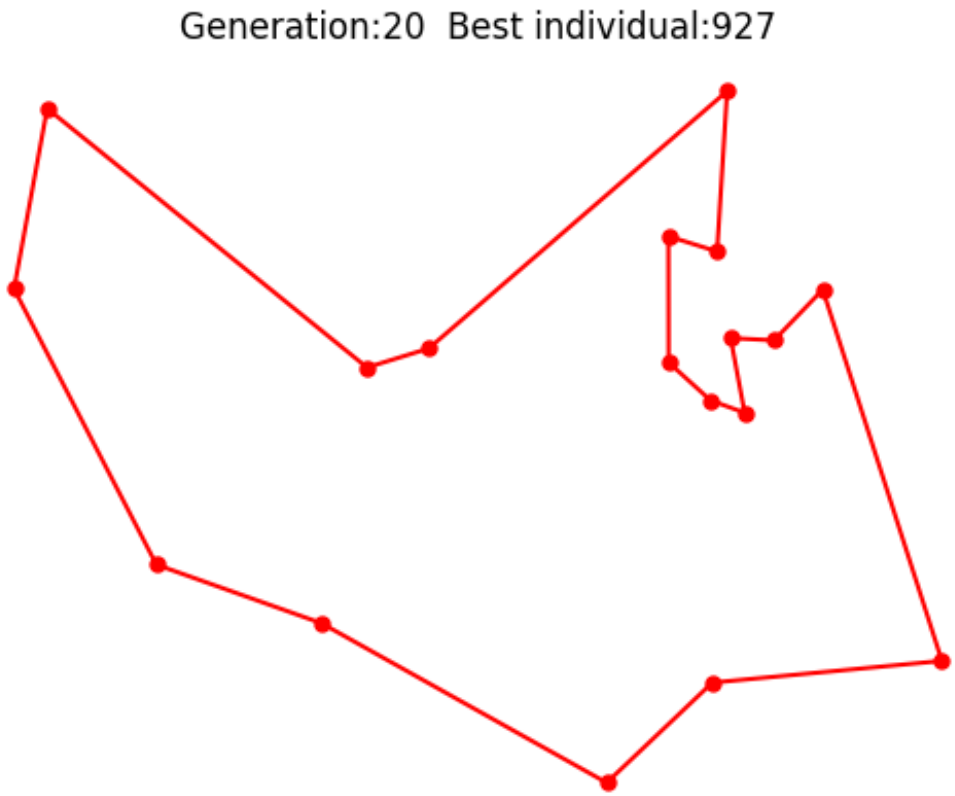
TSP: 18 Cities

Generation 10 - Partial Solution



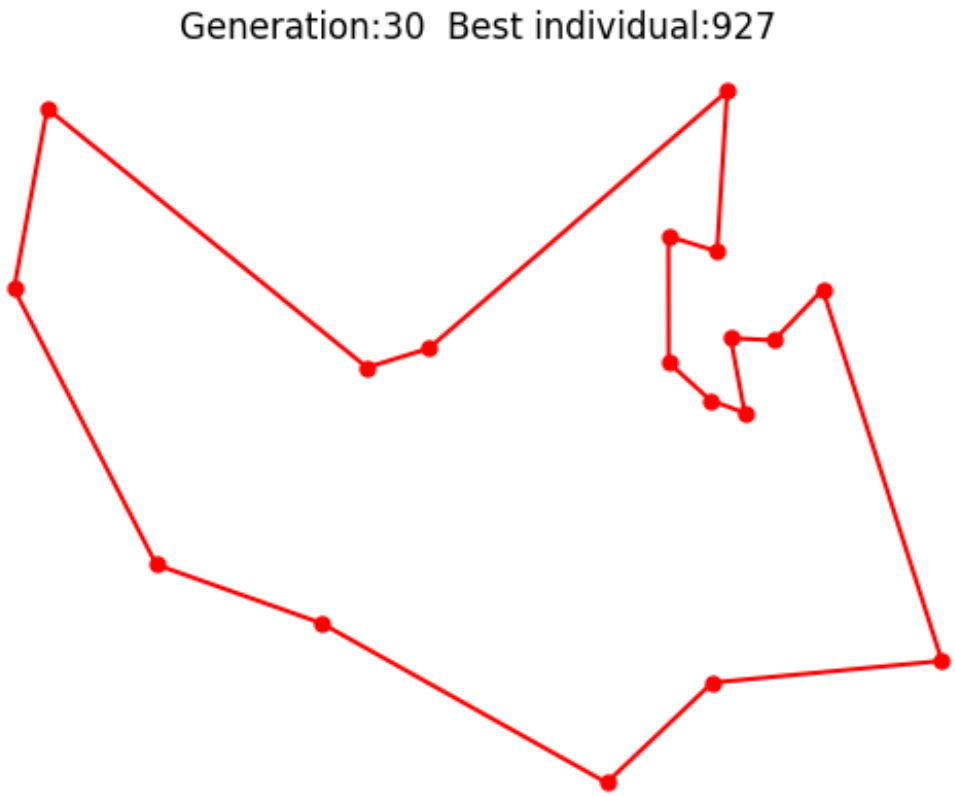
TSP: 18 Cities

Generation 20 - Partial Solution



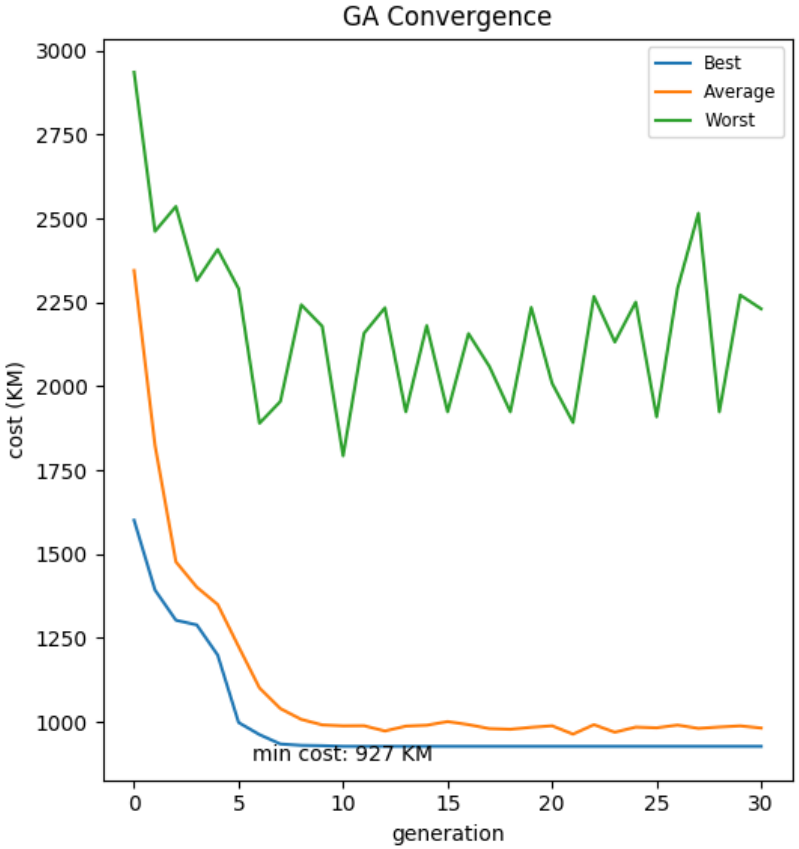
TSP: 18 Cities

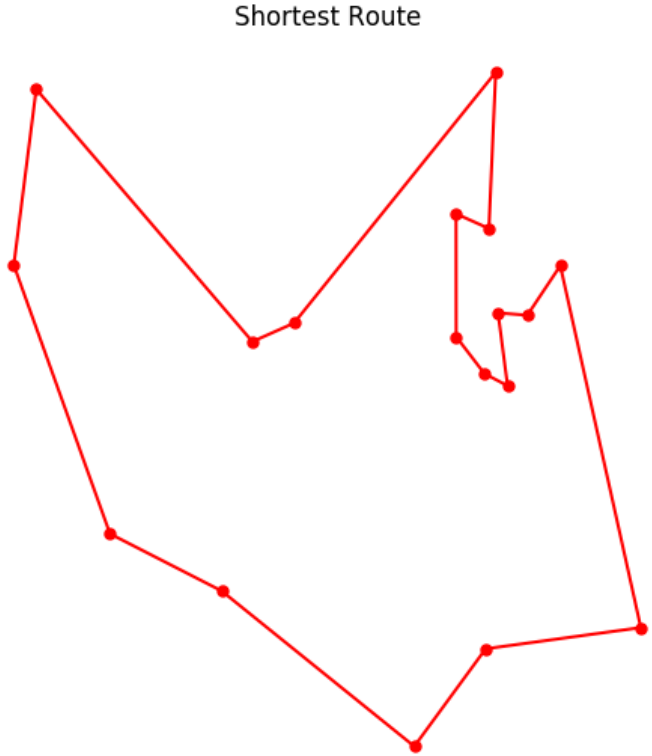
Generation 30 - Partial Solution



TSP: 18 Cities

Solution

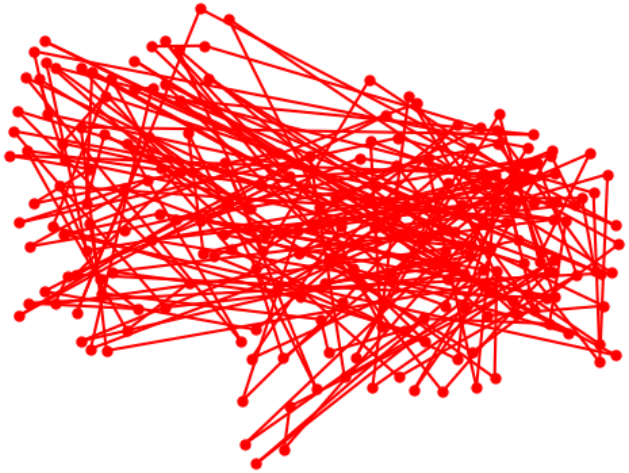




TSP: 223 Cities

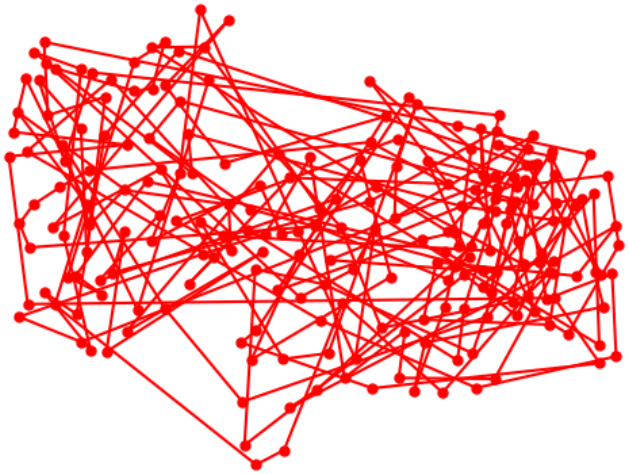
Generation 20/40/60/80 - Partial Solution

Generation:20 Best individual:31485

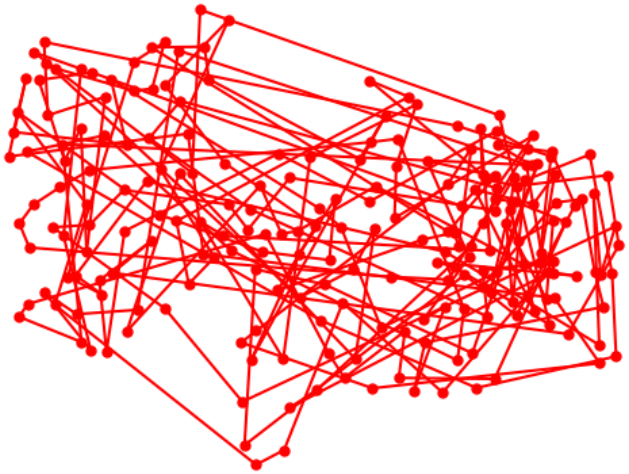


Generation:40 Best individual:19895

Generation:60 Best individual:16986

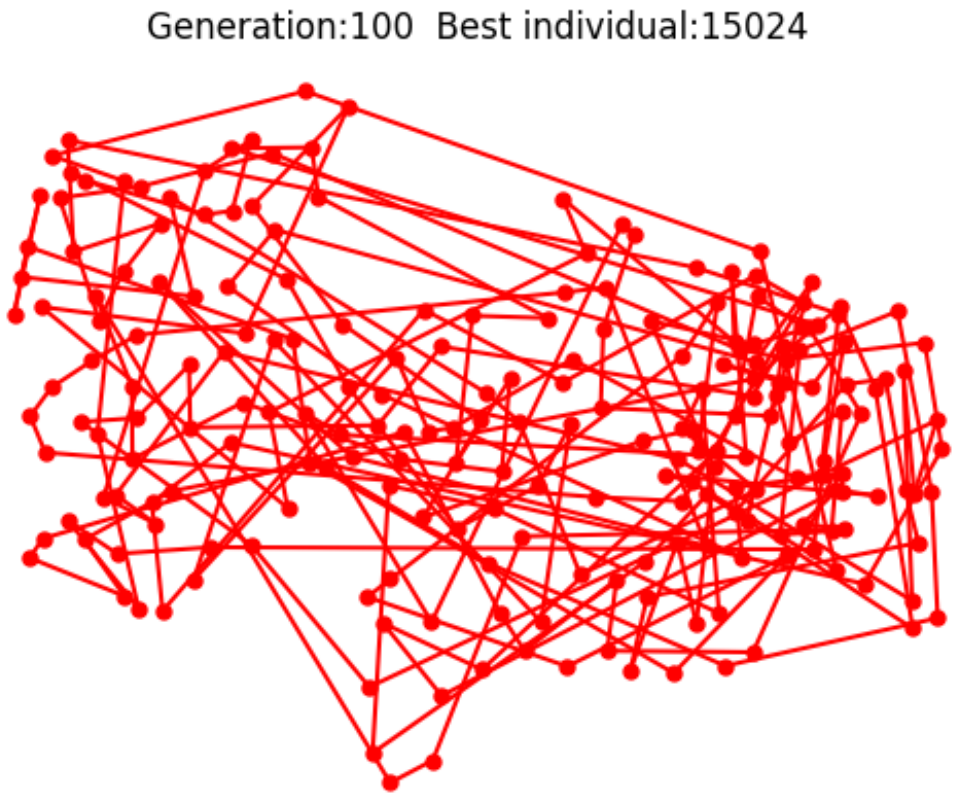


Generation:80 Best individual:15788



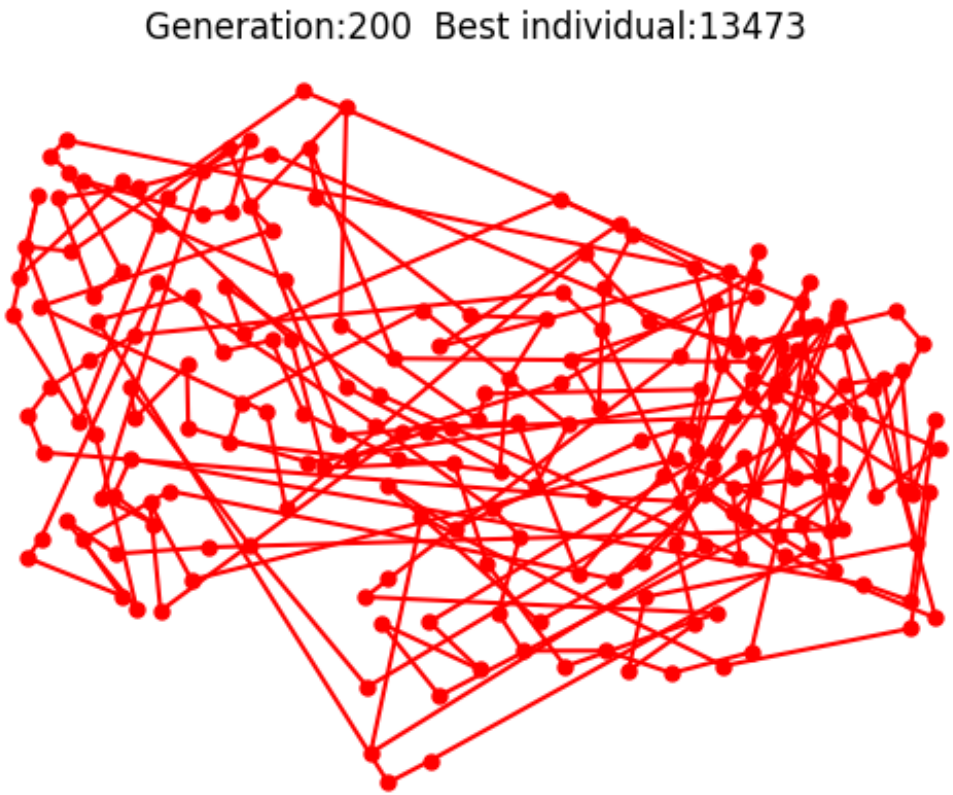
TSP: 223 Cities

Generation 100 - Partial Solution



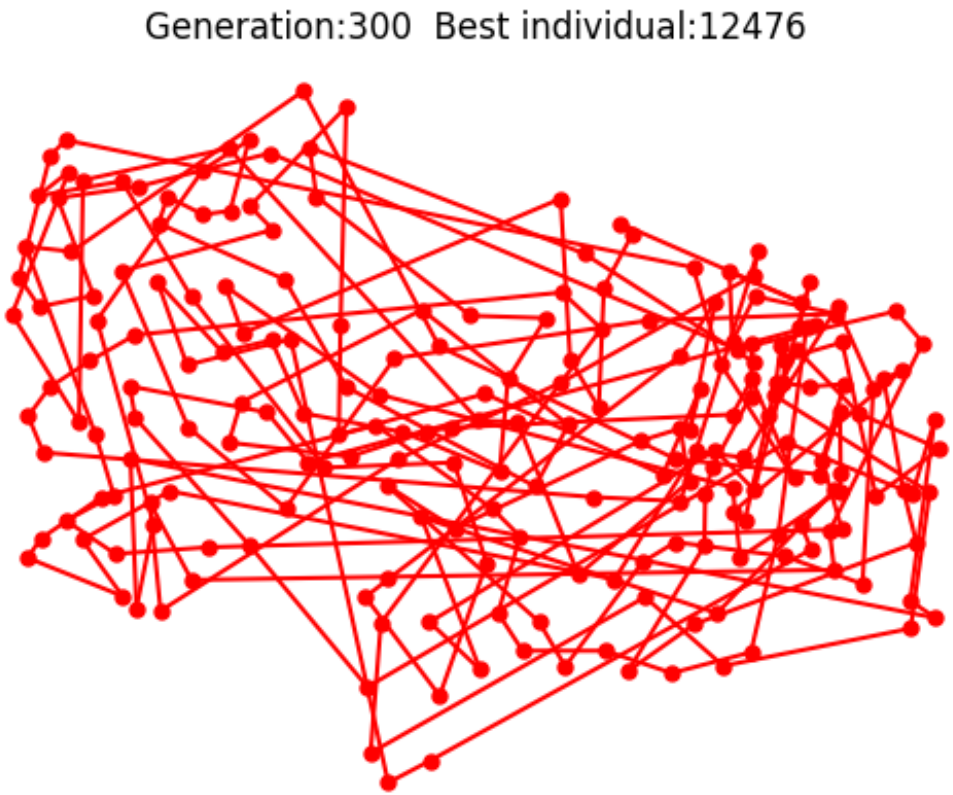
TSP: 223 Cities

Generation 200 - Partial Solution



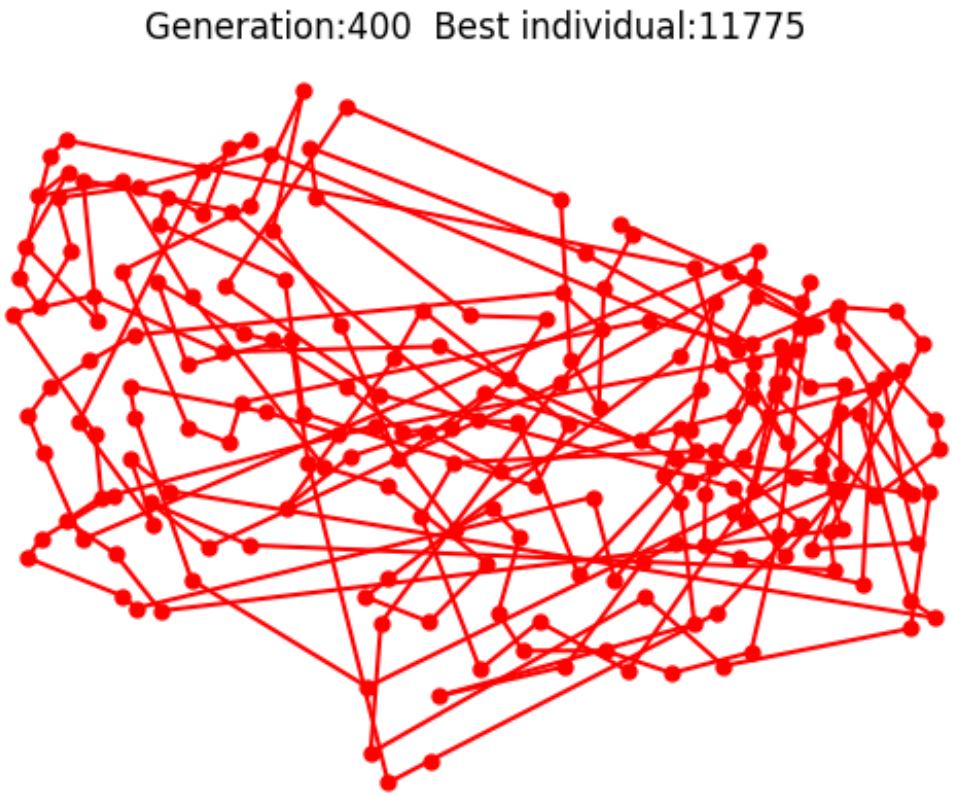
TSP: 223 Cities

Generation 300 - Partial Solution



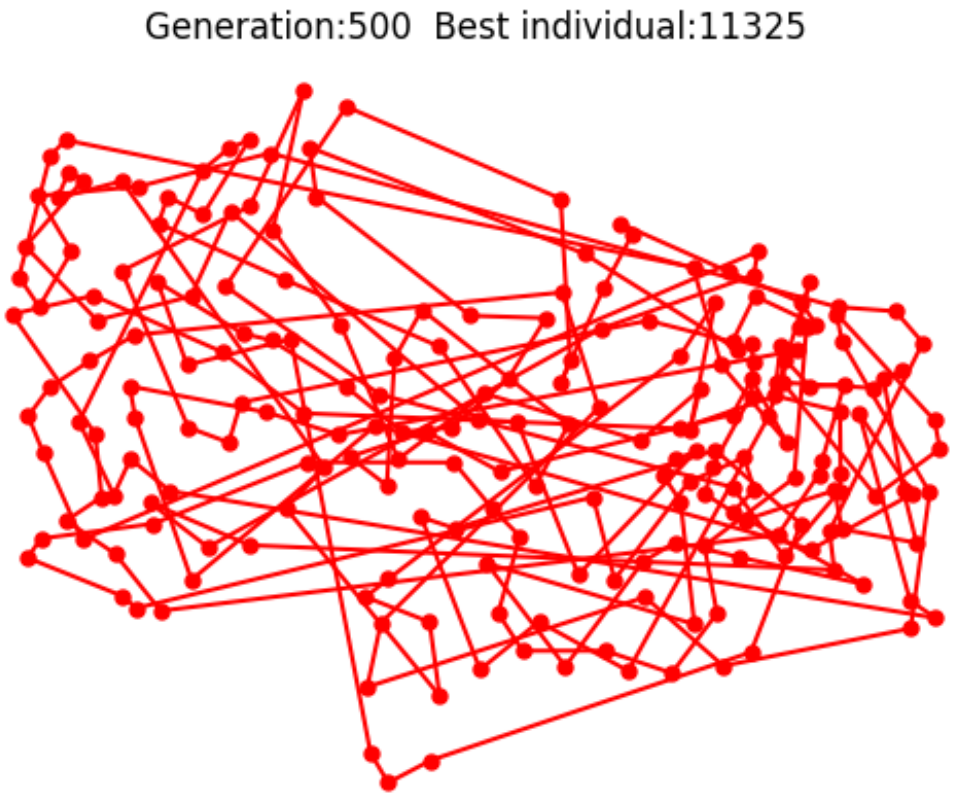
TSP: 223 Cities

Generation 400 - Partial Solution



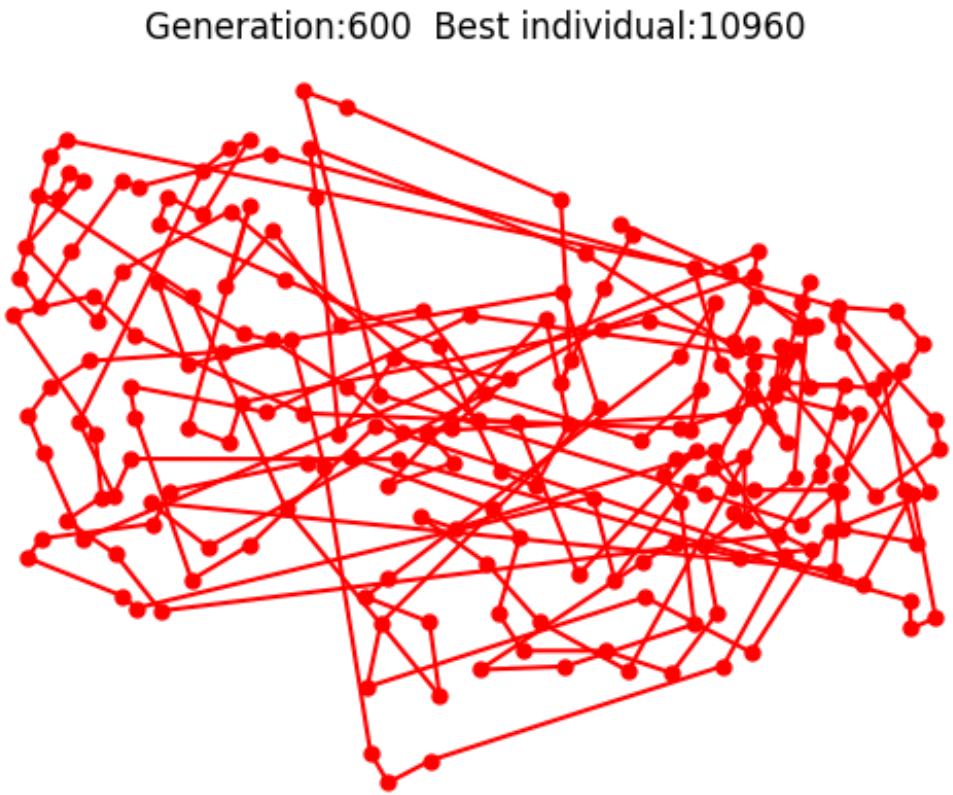
TSP: 223 Cities

Generation 500 - Partial Solution



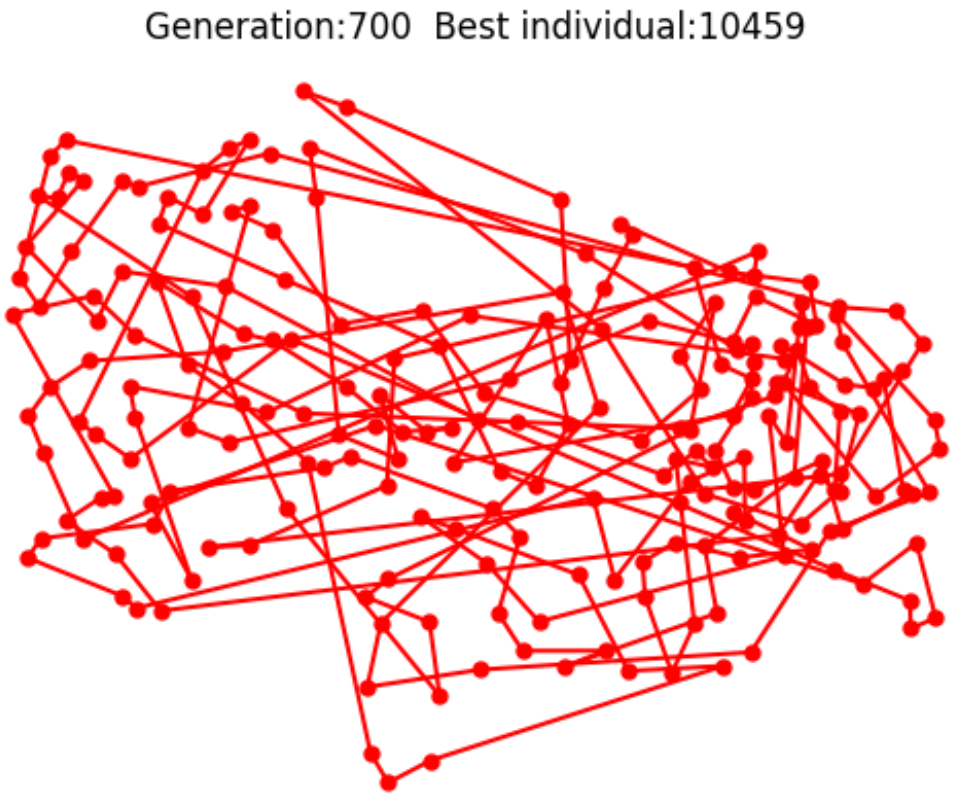
TSP: 223 Cities

Generation 600 - Partial Solution



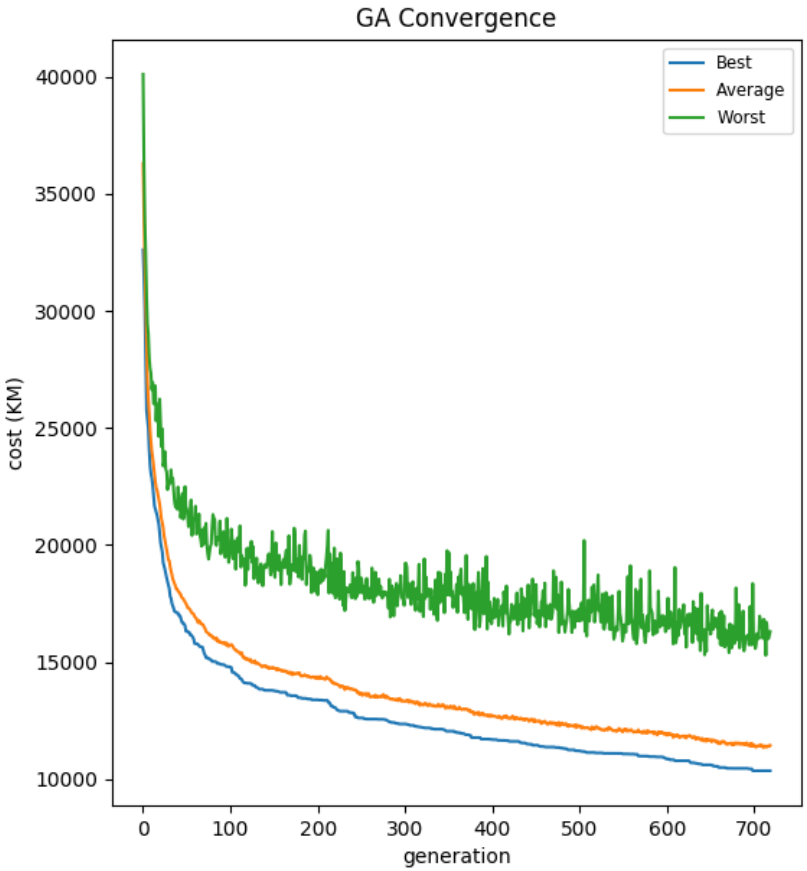
TSP: 223 Cities

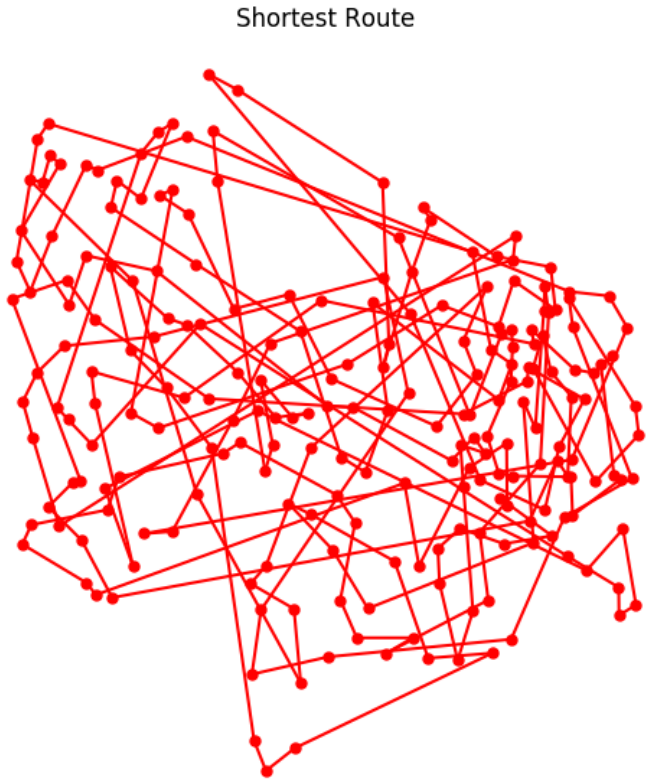
Generation 700 - Partial Solution



TSP: 223 Cities

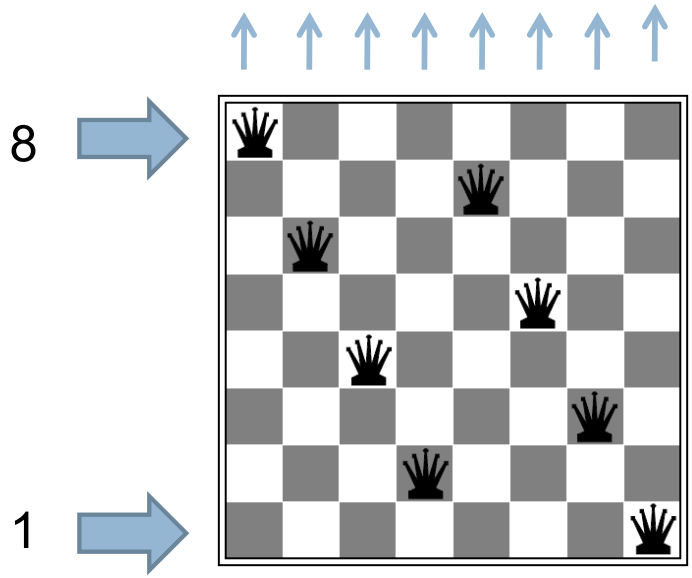
Solution





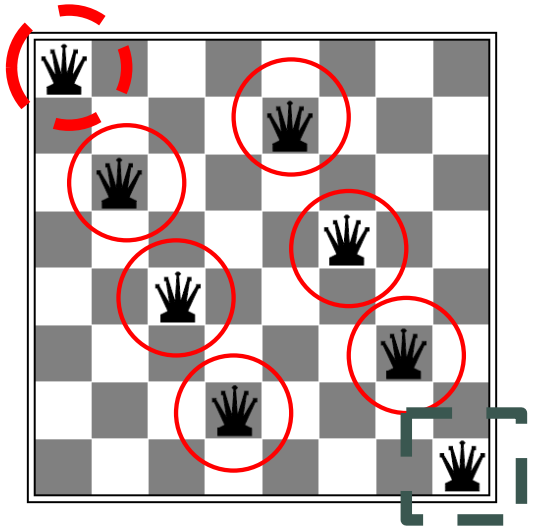
Example: 8-queens

- Problem: 8 queens
  - State: represented in 8 digits, each representing the position of a queen in a column.
  - Ex: [8 6 4 2 7 5 3 1]



Example: 8-queens

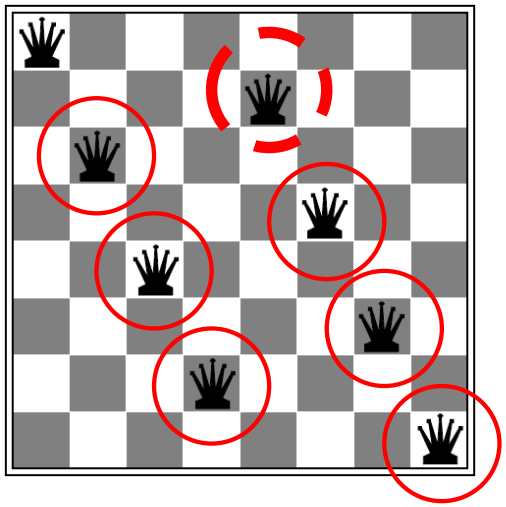
- Fitness function: should return a higher value for better states. Example: number of pairs of non-attacking queens (28 is the solution to the problem).



f = 6 +

Example: 8-queens

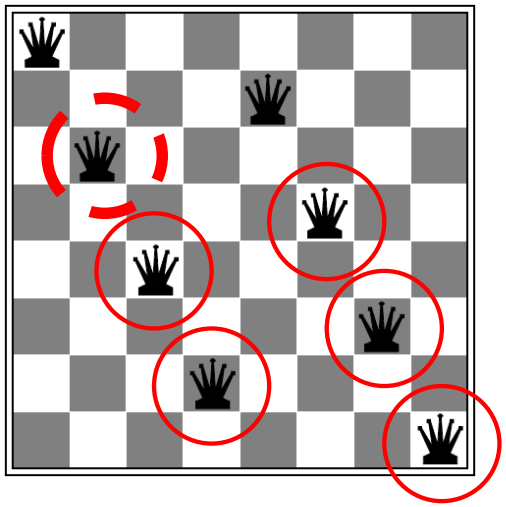
- Fitness function: should return a higher value for better states. Example: number of pairs of non-attacking queens (28 is the solution to the problem).



f = 6 + 6 +

Example: 8-queens

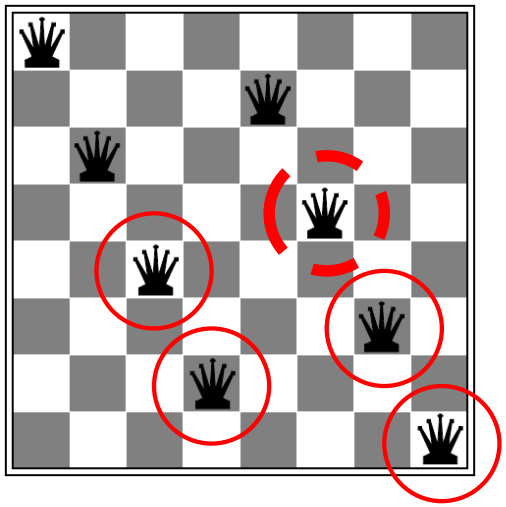
- Fitness function: should return a higher value for better states. Example: number of pairs of non-attacking queens (28 is the solution to the problem).



f = 6 + 6 + 5 +

Example: 8-queens

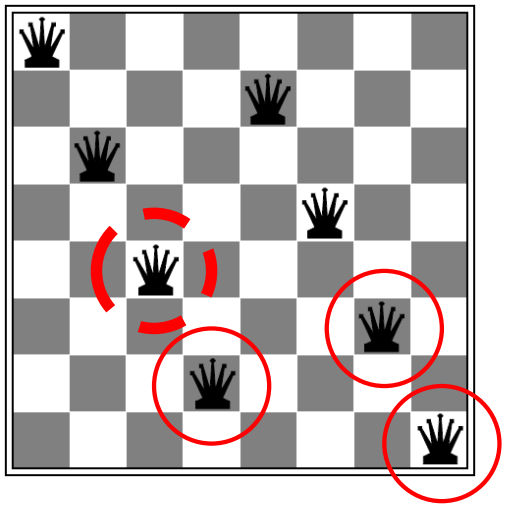
- Fitness function: should return a higher value for better states. Example: number of pairs of non-attacking queens (28 is the solution to the problem).



f = 6 + 6 + 5 + 4 +

Example: 8-queens

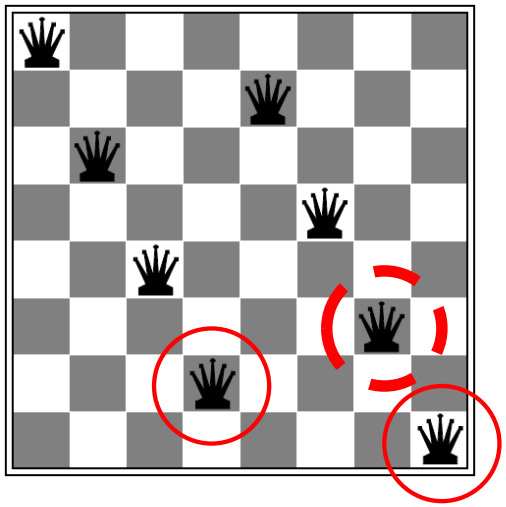
- Fitness function: should return a higher value for better states. Example: number of pairs of non-attacking queens (28 is the solution to the problem).



f = 6 + 6 + 5 + 4 + 3 +

Example: 8-queens

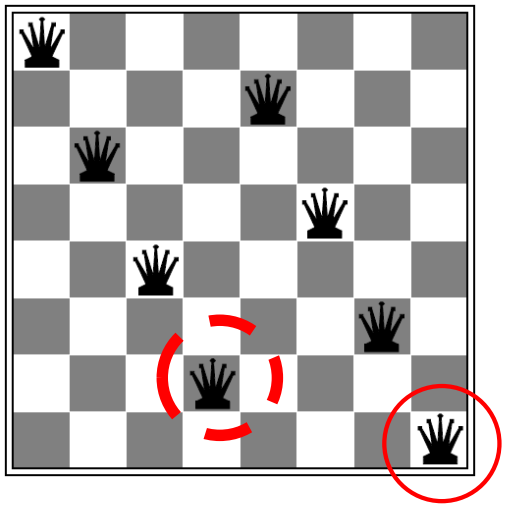
- Fitness function: should return a higher value for better states. Example: number of pairs of non-attacking queens (28 is the solution to the problem).



f = 6 + 6 + 5 + 4 + 3 + 2 +

Example: 8-queens

- Fitness function: should return a higher value for better states. Example: number of pairs of non-attacking queens (28 is the solution to the problem).

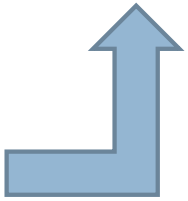


f = 6 + 6 + 5 + 4 + 3 + 2 + 1 = 27

Example: 8-queens

- Pairs for reproduction: chosen according to the fitness function probabilities

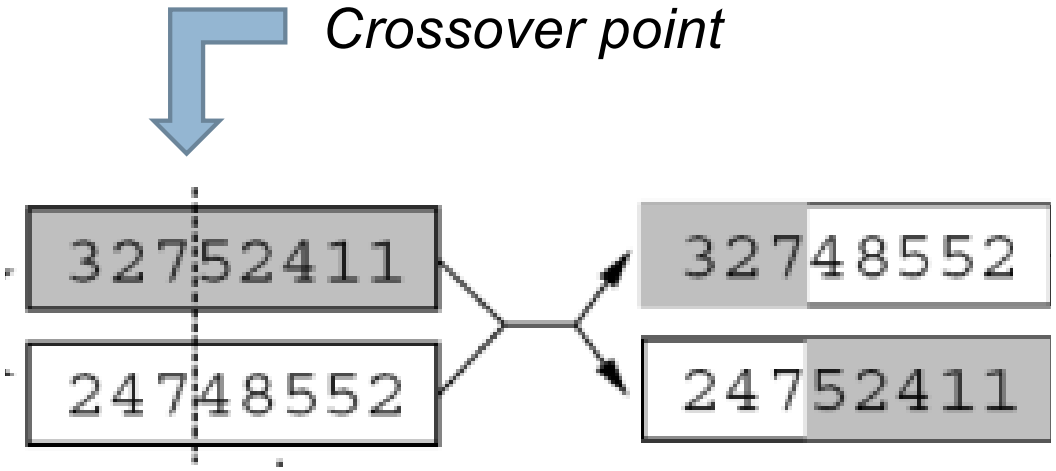
[24748552]	(f=24)	35,8%
[32752411]	(f=23)	34,3%
[24415124]	(f=20)	29,8%
	T = 67	100%



  - Probabilities for choice

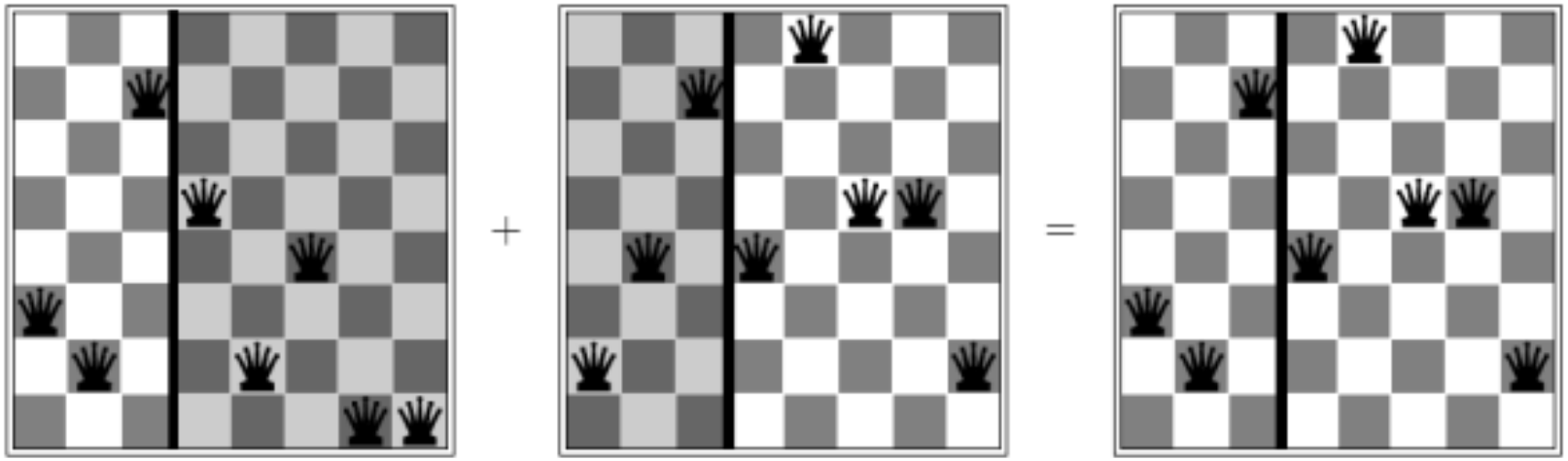
Example: 8-queens

- The single crossover point: chosen at random from the positions in the chain



- Mutation: part of the new state generated can undergo a random mutation in relation to the crossover product.

Example: 8-queens



GA: Parameters

- Population size
- Crossover rate
- Mutation rate
- Generation interval
- Stopping criteria

GA: Evaluation

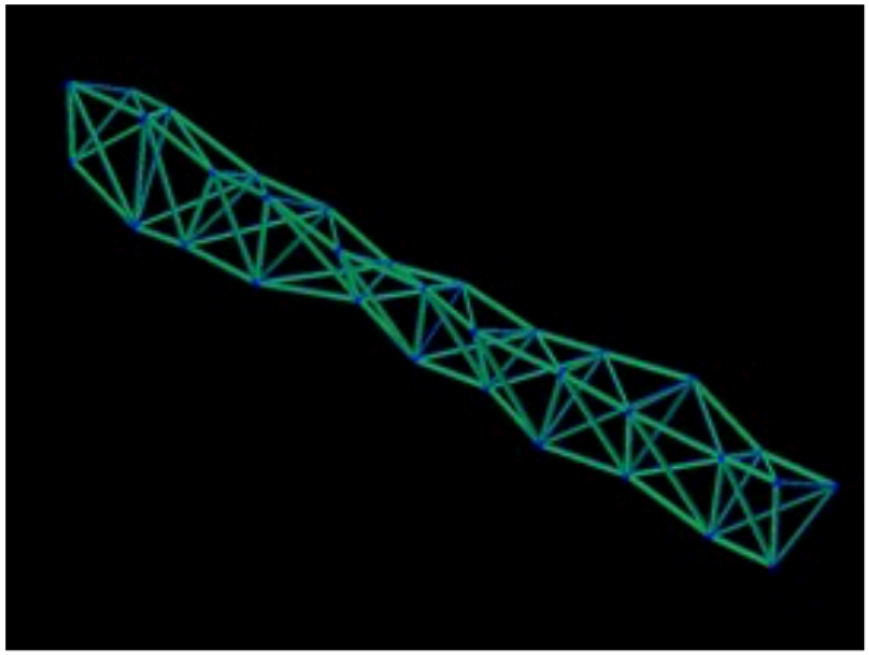
- Benefits
  - Simple (multiple representations, 1 algorithm) and little sensitive to small variations
  - Wide field of applications
  - Still expensive but can be easily parallelized
- Disadvantages
  - Since the method is basically numeric, it is not always easy to introduce knowledge of the domain
  - If not paralleled, it can take a long time to find a solution

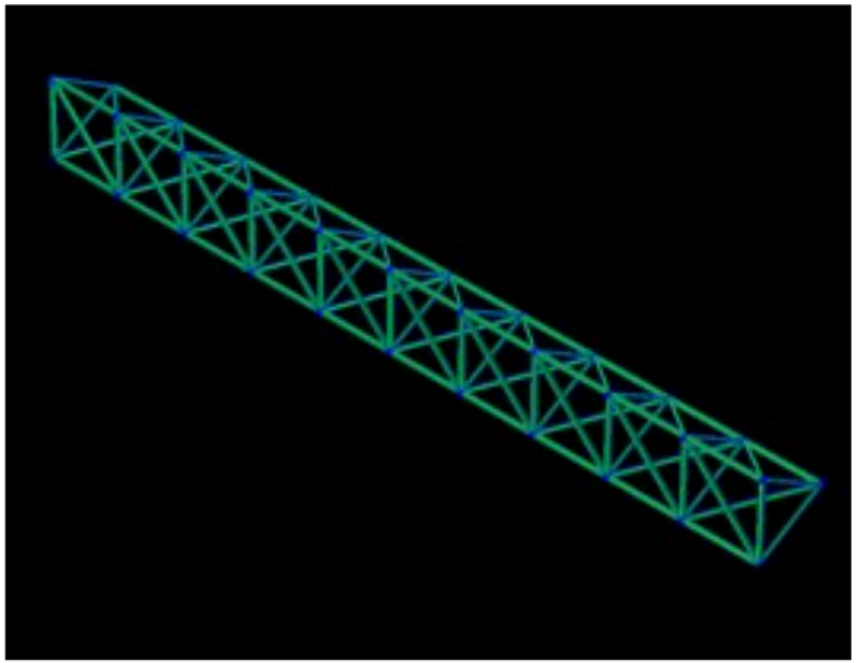
GA: Examples of applications

- Telecommunications Routing
- Olympic Games Planning
- Credit Assessment and Risk Analysis
- Circuit partitioning
- Games
- Optimization issues in general
- Dynamic Systems Control
- Finding New Connectionist Topologies: Artificial Neural Systems Engineering, Modeling of Biological Neural Structures
- Simulation of Biological Models
- Musical composition.

Applications: Generation of antenna designs

- Antenna generated by AG x Antenna generated by designer





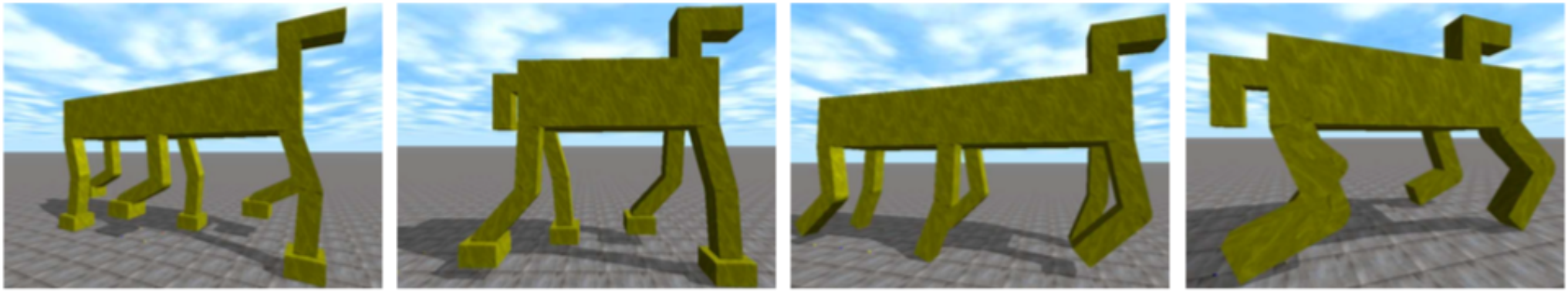
Applications: CMU robots

- Make AIBO go faster
- Learned from Genetic Algorithms
- They left some AIBOs pacing for days
- They walked for x seconds and assessed how far they had walked by positioning in the field
- He was located by LANDMARKS at the ends of the field

http://www-math.uni-paderborn.de/~junge/images/aibos.jpg

Applications: Learn the best architecture for walking

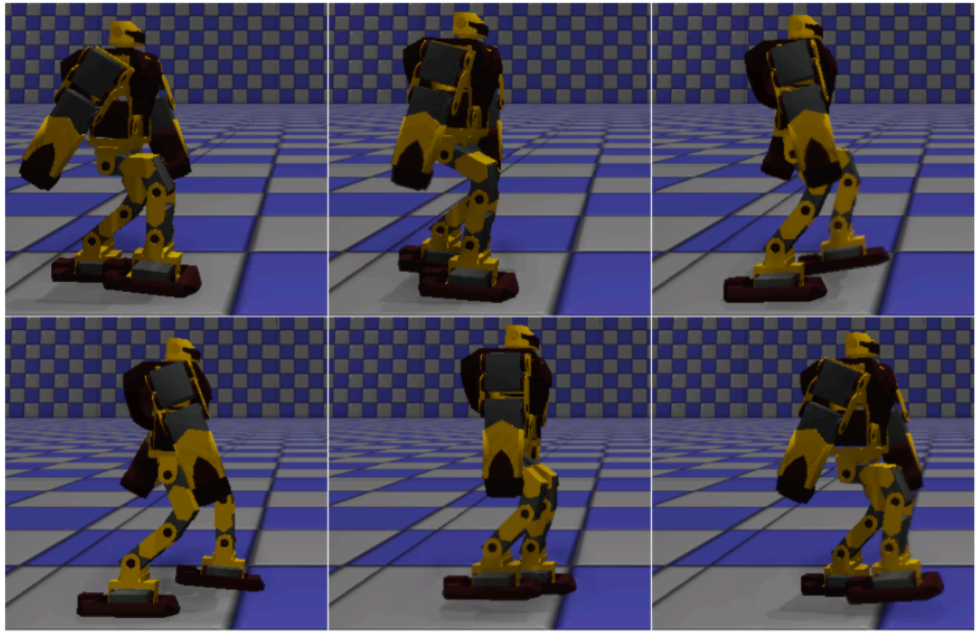
- Milton Heinen – M.Sc. – 2007 - UNISINOS

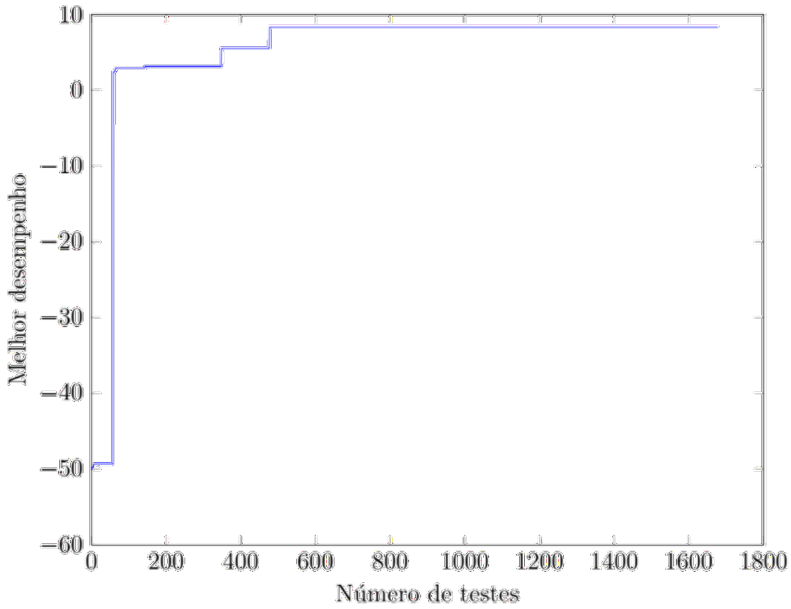


- Fitness: Go further, stable and faster
- Simulated environment and sensors.

Applications: Optimize bipedal walking parameters

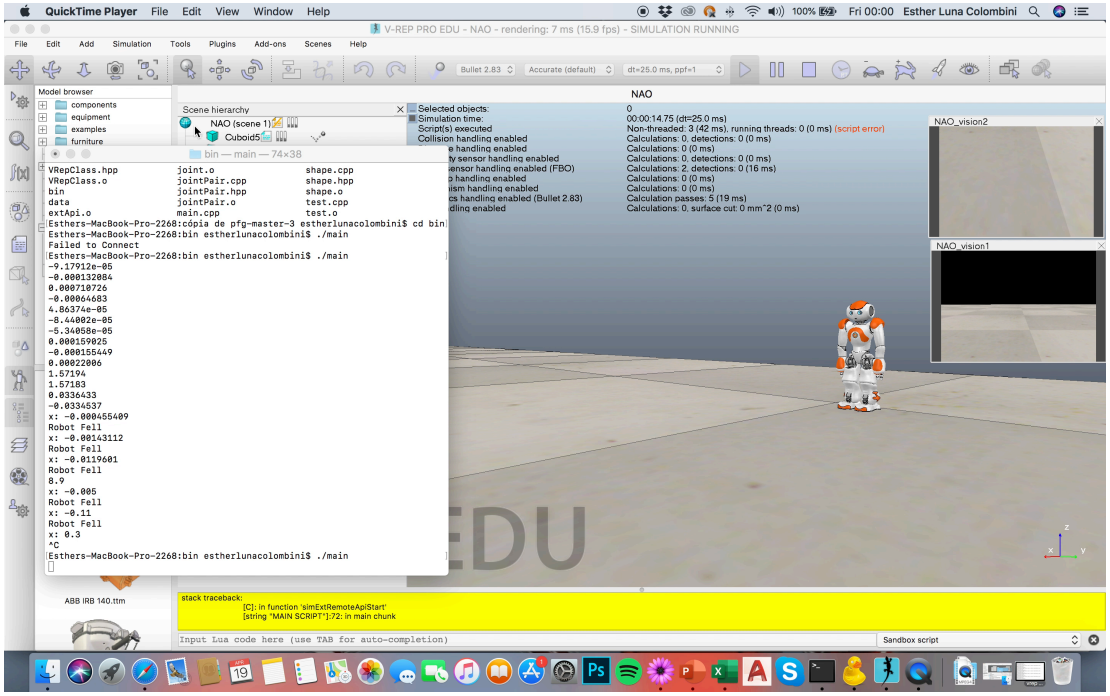
- Best PFG 2012/COMP-ITA
- Truncated Fourier series - AG optimizes parameters

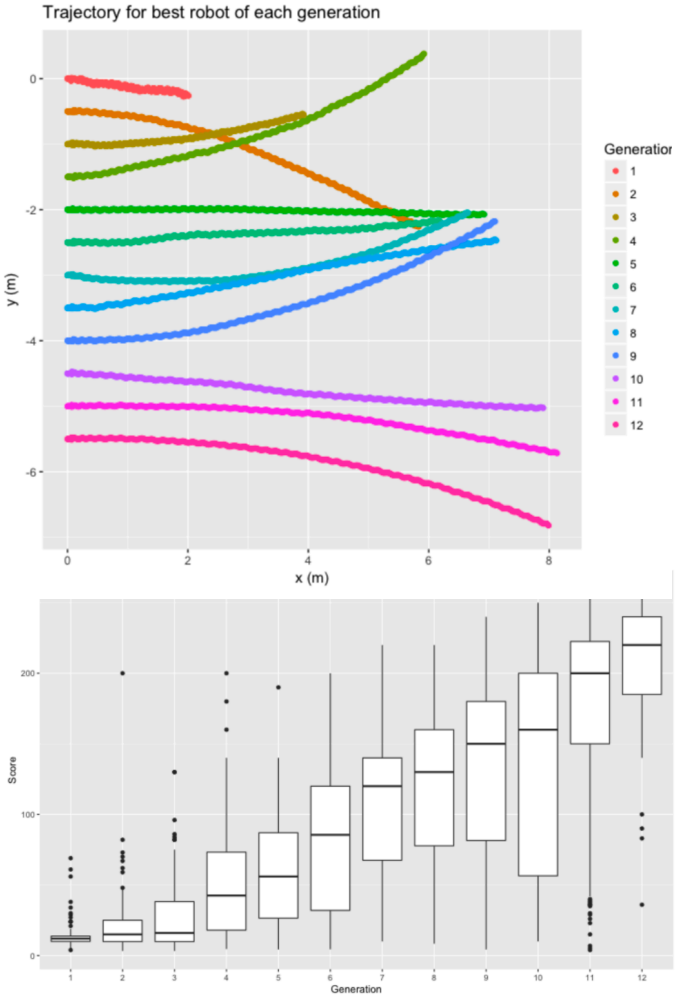




Applications: Optimize bipedal walking parameters

- PFG IC - Outperformed literature results
- Truncanda Fourier Series - AG optimizes parameters

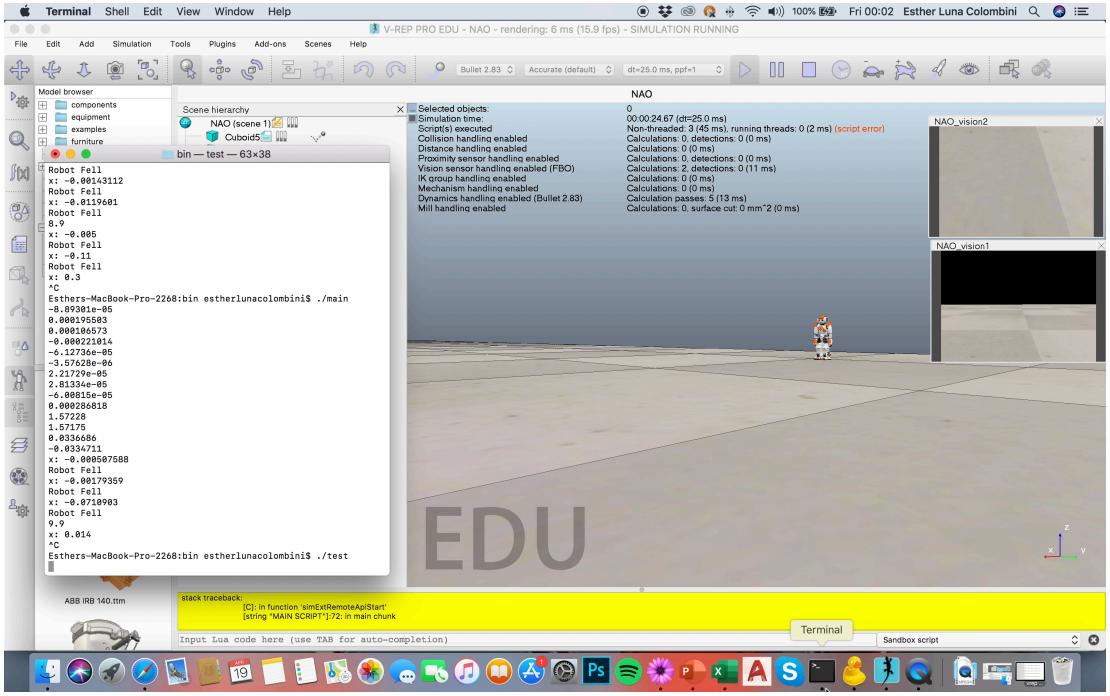


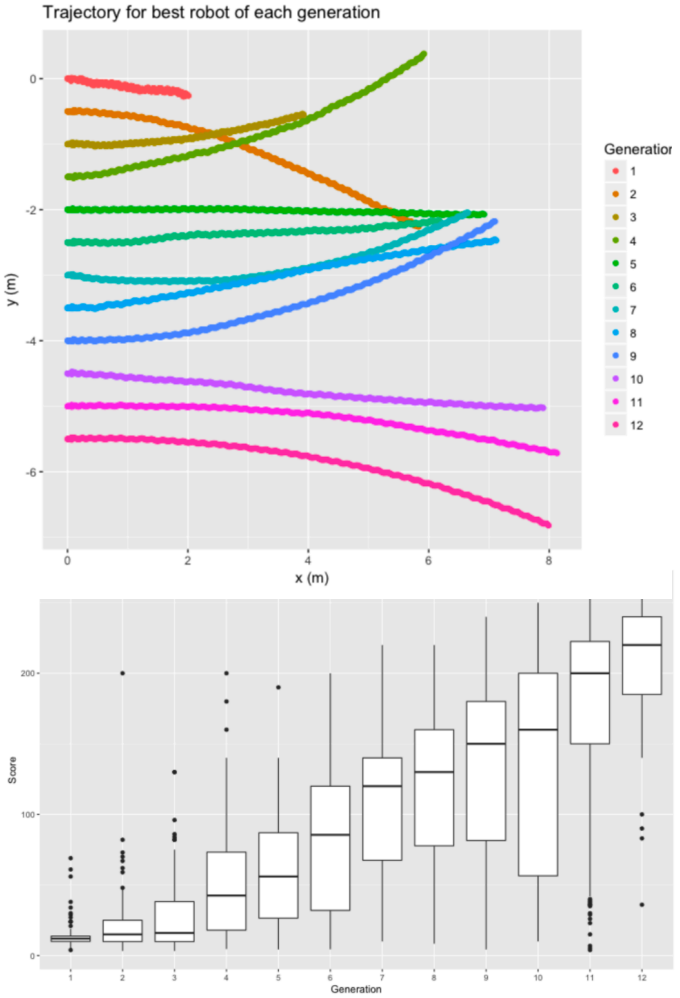


Run	T (s)	Δx (m)	Δy (m)	Vx (m/s)
1	15	8.1	-0.72	0.54
2	15	8.1	-0.98	0.54
3	15	8.1	-0.29	0.54
AVG	15	8.1	-0.663	0.54

Applications: Optimize bipedal walking parameters

- PFG IC - Outperformed literature results
- Truncanda Fourier Series - AG optimizes parameters





Run	T (s)	Δx (m)	Δy (m)	Vx (m/s)
1	15	8.1	-0.72	0.54
2	15	8.1	-0.98	0.54
3	15	8.1	-0.29	0.54
AVG	15	8.1	-0.663	0.54

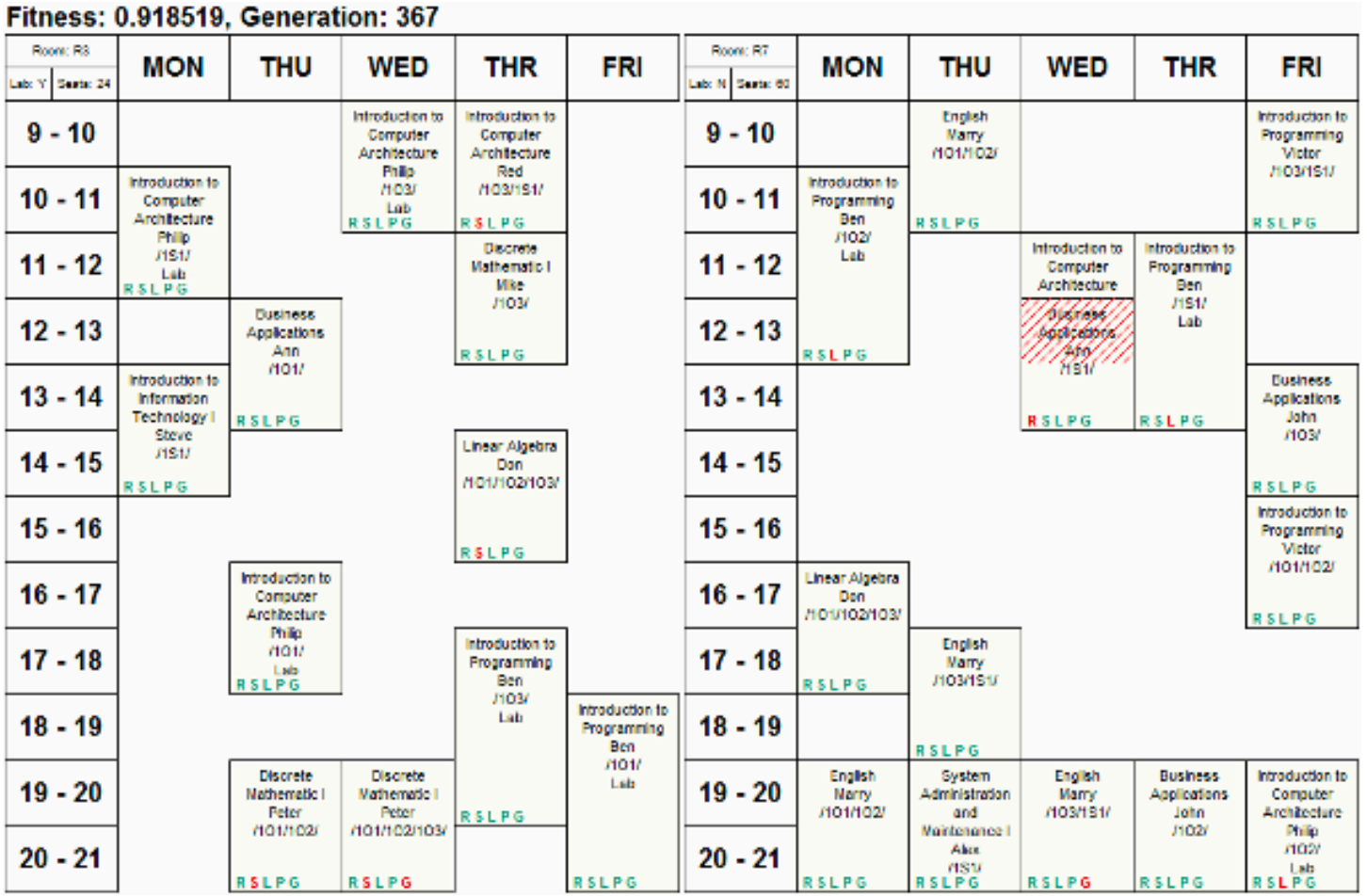
Applications: Query Optimization by Genetic Algorithm

- http://ceur-ws.org/Vol-129/paper16.pdf
- The goal is to retrieve most relevant documents with less number of nonrelevant documents with respect to user query in Information retrieval system using genetic programming.

No.	Selection for Parents depends on	Initial population	Result	Fitness value for Precision Recall
1	Precision	Q_1	(((w_8 or w_2) or (w_{13} and w_8)) or ((w_8 xor w_2) or (w_{13} and w_8)))	1.250000 1.000000
			(((w_8 or w_2) or (w_8 or w_2)) or (w_8 and w_2))	1.250000 1.000000
	Recall		((w_{13} or w_8) or (w_6 or w_2))	1.083333 1.000000
			((w_{13} and w_8) or ((w_6 or w_2) or ((w_{13} or w_8) or (w_6 or w_2))))	1.083333 1.000000
2	Precision	Q_2	((w_8 xor w_2) or (((w_{13} and w_8) or (w_8 xor w_2)) or (w_8 xor w_2)) or (w_8 xor w_2)))	1.250000 1.000000
			((w_8 or w_2) or ((w_{13} and w_8) and (w_8 or w_2)))	1.250000 1.000000
	Recall		(((w_8 xor w_2) or (w_8 xor w_2)) or (w_8 xor w_2))	1.250000 1.000000
			((w_{13} and w_8) or (w_8 xor w_2))	1.250000 1.000000

Applications: Class Scheduling

- https://www.codeproject.com/Articles/23111/Making-a-Class-Schedule-Using-a-Genetic-Algorithm



Exercise

- Assume that you are trying to find the values from X1 to X4 that maximize the following function:
  - f = 5X1 - 3 X2 X3 + X3 - 2 X4
  - You decide to use a genetic algorithm and create the initial population:
    - 0110
    - 1100
    - 1011
    - 0001
- Show and briefly explain how you can create the next two generations of the initial population.

Extra-class activities

Lecture 7

- Atividades
  - Leitura:
    - ZUBEN, F. Computação Evolutiva: Uma Abordagem Pragmática.
    - COPPIN, Ben. Inteligência Artificial, 2013. Capítulo 14.
    - Discuss the differences between the simple hill climbing algorithm and the evolutionary strategy (1 + 1) -EE

References

Lecture 7

- ZUBEN, F. Computação Evolutiva: Uma Abordagem Pragmática.
- COLOMBINI, E. C, TONIDANDEL, F. Notas de Aula de IA.
  - COPPIN, Ben. Inteligência Artificial, 2013. Capítulo 14.