

Ministry of Science and Higher Education of the Russian
Federation National Research University ITMO

Evolutionary calculations

Spring

2024

Laboratory work No. 2

INTRODUCTION TO EVOLUTIONARY COMPUTING

Results after running BitsExample with different dimensions:

Dimension	Run1	Run2	Run3	Run4	Run5	Average
20	30	20	28	42	16	27.2
50	50	2089	1879	1884	2591	1702.2
100	Didn't stop, stuck between 88 and 89	Same as previous	Same	Same	same	Exceeded 17471759

The algorithm works perfectly when having low values for the dimension (the length of the bits string), while it fails to reach the optimal solution when having bigger values for the bits string length. After going through the whole lab, and learning more about techniques that can be used in the evolutionary algorithm, and after delving into the implementation of BitsExample, we notice that the implementation of this algorithm didn't have elitism, which I think that could have solved the problem, because we can clearly see the simplicity of this example, and it only needed to keep the populations that were giving good results and enhance it.

Analyzing Traveling salesman problem:

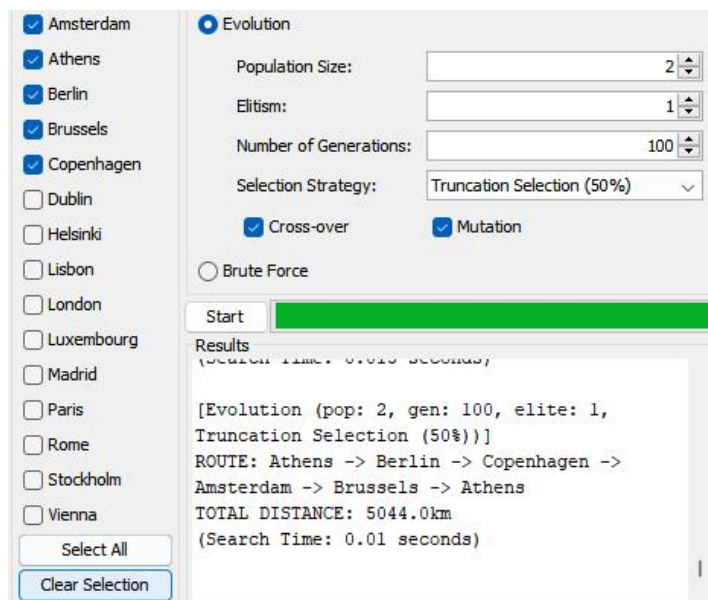
When running the applet window of this example, the available parameters for the genetic algorithm control the behavior and performance of the algorithm. Those settings are as following:

1. Population Size: it determines the number of candidate solutions (individuals) in each generation.
2. Elitism: the number of the best-performing individuals from the current generation that are directly carried over to the next generation without any changes, so when it is not zero, it means that the best solutions found so far are preserved across generations, helping to maintain or improve solution quality (which means that it was not used in the BitsExample, because the best results over generations were fluctuated).
3. Number of Generations: the total number of generations or iterations that the genetic algorithm will go through before termination.
4. Selection Strategy: the method used to choose individuals from the population to serve as parents for the next generation. Available selection strategies are the following:
 - **Truncation Selection (50%)**: involves selecting a fixed percentage of the best-performing individuals from the population based on their fitness scores. For example, selecting the top 50% of individuals. This method favors exploitation by focusing on the best solutions found so far.
 - **Tournament Selection (with a Probability of 0.95)**: Tournament selection involves randomly selecting a few individuals from the population and comparing their fitness scores to determine the winner. The probability parameter (e.g., 0.95) indicates the probability of selecting the fittest individual as the winner of each tournament. This method balances exploration and exploitation by allowing less fit individuals to occasionally become parents, promoting diversity in the population.

- **Stochastic Universal Sampling (SUS):** Stochastic Universal Sampling selects individuals based on their fitness scores, with selection probabilities proportional to their fitness ranks. This method aims to reduce selection pressure compared to roulette wheel selection by ensuring that each individual has a fixed number of selection points along a continuous segment.
- **Sigma Scaling:** Sigma scaling adjusts the fitness scores based on the population's mean and standard deviation, aiming to balance exploration and exploitation. After adjusting the fitness scores, the selection method is applied, which is not explicitly specified.
- **Roulette Wheel Selection:** involves assigning selection probabilities to individuals proportional to their fitness scores, simulating the spinning of a roulette wheel. Fitter individuals have higher chances of being selected, but the method can suffer from stochastic noise and premature convergence.
- **Rank Selection:** it ranks individuals based on their fitness scores and assigns selection probabilities based on their ranks rather than absolute fitness values. This method is less sensitive to outliers and can promote diversity in the population compared to fitness-proportional methods.

Comparing results with different sets of parameters (running the algorithm with the same set of parameters multiple times to determine the effect of changes):

In order to show the chosen parameters, we will just display a screen shot of the applet window:



In figure 1, 5 cities were chosen, with high number of generations equaling 100.

Figure 1 while having a very big number of generations, and a very few number of population size, the value of Elitism played a big role, so when we don't have Elitism (meaning its value is zero), the algorithm was not able to preserve the optimal value when found through generations, so it was not necessarily reaching the optimal value (5044.0 km), but when set to 1, it guaranteed giving the optimal solutions.

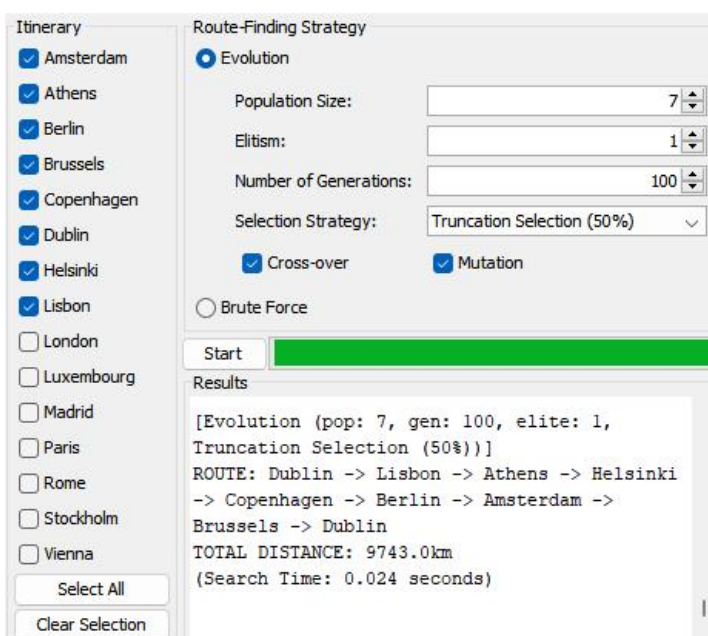


Figure 2 In Figure 2, we increased the number of cities, where we need a size of population around 7. In this case, having a value of Elitism equaling zero, made it impossible for the algorithm to reach the optimal value, even through multiple runs, but when elitism was set to 1, the algorithm worked perfectly (gave the optimal value through all runs). But when setting a higher value for elitism, the algorithm was not able to give the optimal value every time, and the higher value set to elitism -> the less chance the algorithm had for reaching the optimal value.

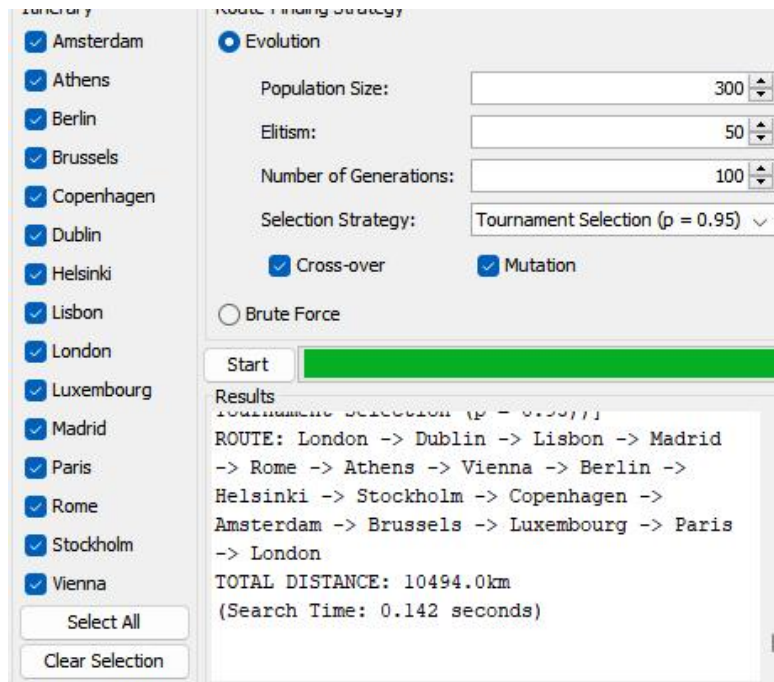


Figure 3

- In Figure 3, after choosing a very big number of cities, the problem became much more complicated, which results in the need of having a much bigger number of population size. This experiment showed that the optimal value for Elitism is not 1, but about 10-50% of the population size.

- Besides, when minimizing the number of generations (set to 50), the effect of the selection strategy became clearer, so the tournament selection strategy and other similar strategies that are almost preserving the

good samples through generations, kept giving the optimal value, while strategies that are more likely to lose good samples through iterations (like roulette wheel and stochastic universal sampling), were not able to find the optimal result.




- Through these experiments, the affect of sampling techniques (cross-over and mutation) were studied. In general, selecting both of those two techniques was giving the best results, but when choosing only mutation, the results were still good, but not the same as when having both techniques chosen. But the worst results were given when choosing only cross-over. This means that most of the good samples were resulting from enhancing the samples, not mixing them together (mutation operator is more important than cross-over).

Analyzing Mona Lisa problem:

In this example, first, we will explain the editable parameters available:

1. Population Size: the number of individuals (solutions) in each generation.
2. Elitism: (explained previously).
3. Selection Pressure: Selection pressure determines how much emphasis is placed on selecting the fittest individuals for reproduction. A higher selection pressure value (closer to 1) results in a higher likelihood of selecting the fittest individuals, potentially leading to faster convergence but also increasing the risk of premature convergence and loss of diversity.
4. Add Polygon: The probability of adding a new polygon to the image representation during the evolutionary process. Adding polygons increases the complexity of the image but also allows for more detailed representations.
5. Remove Polygon: The probability of removing an existing polygon from the image representation during the evolutionary process. Removing polygons reduces the complexity of the image but may also result in loss of detail.
6. Reorder Polygons: The probability of reordering the order of polygons in the image representation during the evolutionary process. Reordering polygons can potentially improve the arrangement of shapes in the image.
7. Crossover: Crossover is a genetic operator that combines the genetic material of two parent solutions to produce offspring solutions. The crossover probability determines the likelihood of applying crossover during reproduction. A value of 0 means no crossover will be performed, while a value closer to 1 means crossover will be performed more frequently.
8. Add Vertex: The probability of adding a new vertex to an existing polygon during the evolutionary process. Adding vertices can increase the complexity and detail of the polygons.
9. Remove Vertex: The probability of removing a vertex from an existing polygon during the evolutionary process. Removing vertices simplifies the polygons and reduces complexity.

10. Move Vertex: The probability of moving a vertex within its polygon during the evolutionary process. Moving vertices can change the shape and position of polygons in the image.
11. Change Colour: The probability of changing the color of polygons during the evolutionary process. Changing colors can introduce variation and potentially improve the overall appearance of the image.

Solution	Iteration	Fitness	Number of polygons and vertices	Drawing
Bad	18622	289036.98	16 p, 120 v	
Average	24989	213789.7	49 p, 325 v	
Good	24726	218019.3	32 p, 204 v	

Questions:

1. What type of solution structure does each of the considered problems belong to?

Bits Problem: The solution structure for the Bits problem consists of bit strings of fixed length. Each bit string represents a potential solution, where each bit can be either 0 or 1. The goal is to evolve these bit strings to reach a target solution with a specific pattern of 1s and 0s.

Traveling Salesman Problem: The solution structure for the Traveling Salesman Problem involves representing a sequence of cities to visit. Each solution (tour) consists of a permutation of the cities, defining the order in which they are visited. The objective is to find the optimal permutation that minimizes the total distance traveled, so the solution structure is permutation-based.

Mona Lisa Problem: Here, it is more complex than the previous two examples. It involves representing an image using a combination of geometric shapes, which are polygons. Each solution is a collection of polygons, each with specific attributes like position, color, and transparency. The goal is to evolve these polygons to approximate the target image as closely as possible.

2. How are the solutions in the traveling salesman problem coded?

In this problem, the solutions are coded as permutations of city names. Each solution, or candidate route, is represented as a list of strings where each string corresponds to the name of a city. This representation captures the order in which the cities are visited in the route. This permutation-based representation allows the evolutionary algorithm to explore different routes by swapping the order of cities, thereby searching for the shortest path that visits each city exactly once and returns to the starting city. So since we have here constraints on the list formed (having each city once and only once), then operators used in the evolutionary algorithm are as following:

List Order Crossover: This operator performs crossover between two parent routes represented as lists of city names. Where Two random points are chosen in the parent routes. A subset of cities between these points is copied from one parent to the offspring, preserving the relative order of cities in the selected subset. Then, the remaining cities are mapped from the other parent, ensuring that no city is duplicated in the offspring.

List Order Mutation: This operator applies mutation to a single route represented as a list of city names. It randomly selects a position in the route and displaces the corresponding city to a new position by a specified amount. This displacement is done by swapping the selected city with the city at the target position.

3. What is a genotype and what is a phenotype in the task of reproducing a picture?

In the context of reproducing a picture through evolutionary computation:

A) Genotype:

- The genotype represents the genetic information of an individual in the population.
- In picture reproduction, the genotype comprises a structured representation of the image, often encoded as a set of parameters or instructions.
- For example, in the Mona Lisa example, the genotype consists of data structures encoding the positions, colors, and shapes of polygons that compose the image.

Phenotype:

- Since we know that the phenotype is created by rendering the genotype into a visual form that humans can perceive and interpret, Then we can say that in the context of picture reproduction, the phenotype is the outcome—the visual representation—of expressing the genetic instructions encoded within the genotype.
- In other words, the phenotype is the actual image generated from the genetic information encoded in the genotype.

Additional: the followings are photos of the settings used for getting the good, bad and average images in the Mona Lisa problem.

The screenshot shows the 'Watchmaker Framework - Mona Lisa Example' interface. It includes a 'Parameters' section with 'Population Size' set to 15, 'Elitism' set to 2, and 'Selection Pressure' set to 0.56. Below this is the 'Evolution Probabilities' section with sliders for 'Add Polygon' (0.036), 'Remove Polygon' (0.033), 'Reorder Polygons' (0.031), 'Cross-over' (1.00), 'Add Vertex' (0.052), 'Remove Vertex' (0.039), 'Move Vertex' (0.038), and 'Change Colour' (0.059). The 'Fittest Individual' tab is selected, showing a fitness value of 289036.9836739268. Below the fitness value are two images: the original Mona Lisa painting and a low-poly, abstract representation of the painting. The low-poly image is labeled '16 polygons, 120 vertices'. At the bottom, status information indicates 'Population: N/A', 'Elitism: N/A', 'Generations: 18622', and 'Elapsed Time: 00:01:52'.

Table 1 bad solution

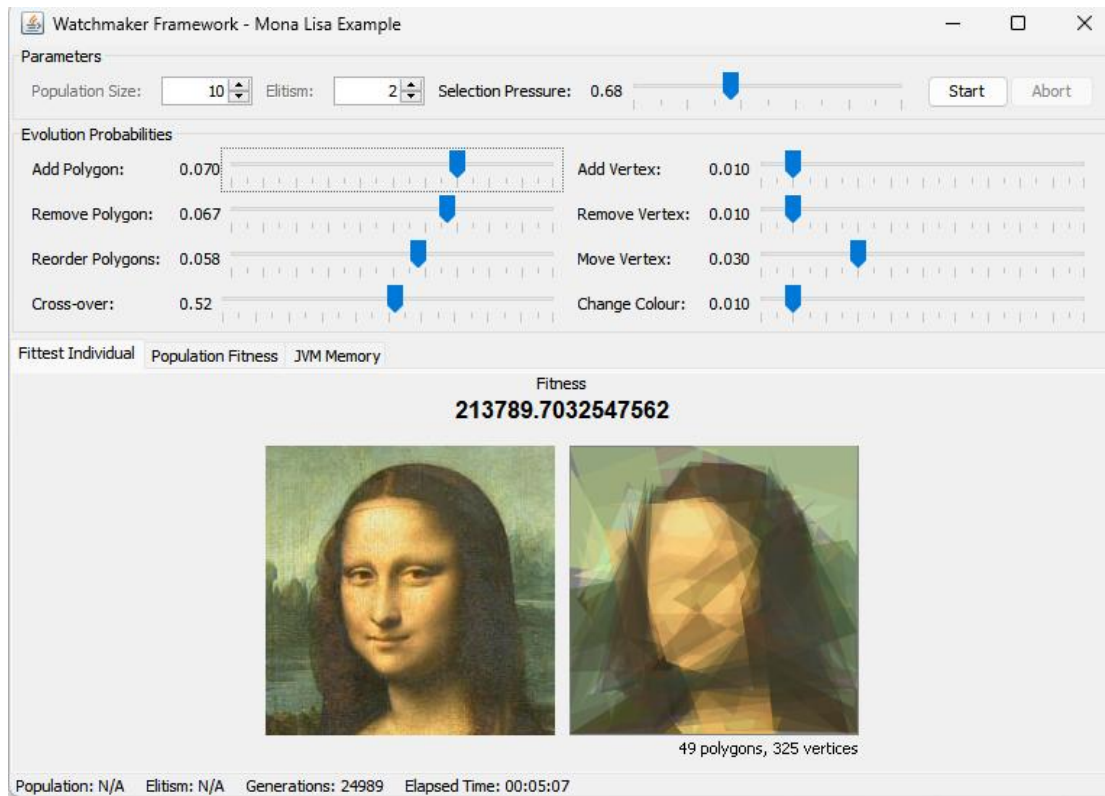


Table 2 average solution

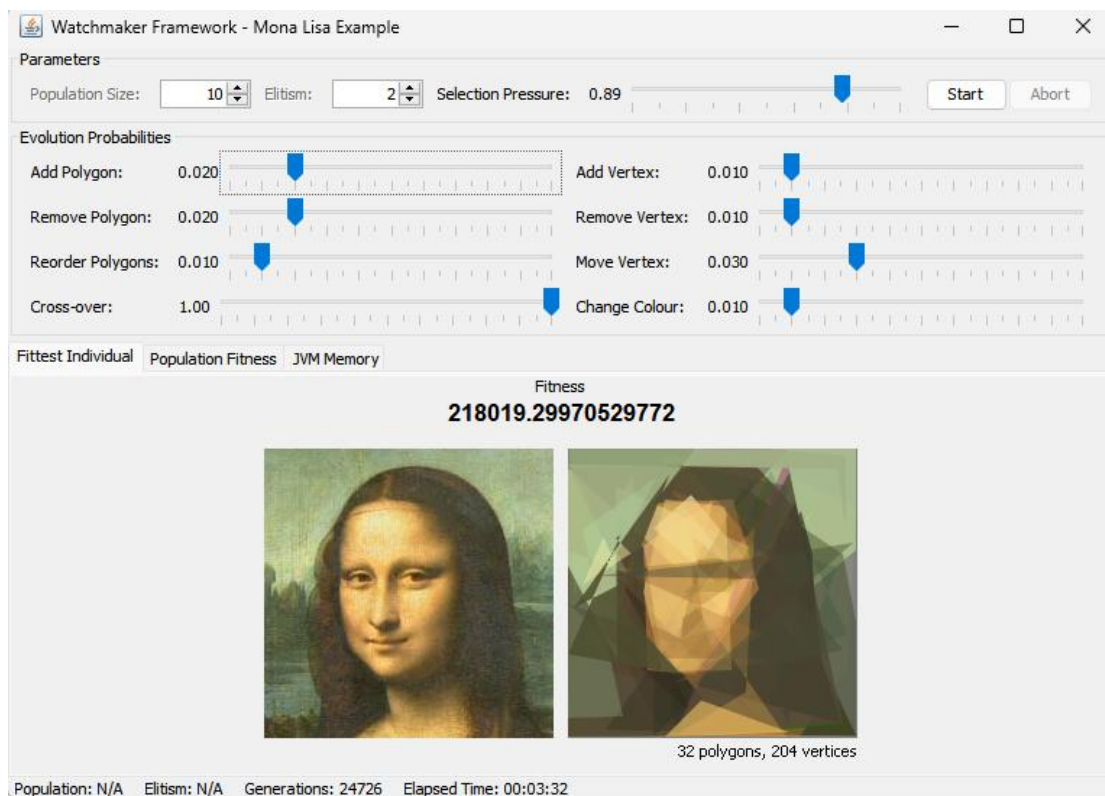


Table 3 good solution

