Artificial Intelligence Fundamentals Report

Salvatore Correnti, Leonardo Stoppani, Lorenzo Leuzzi,
Michele Montebovi, Mohammed Samim Kashefi
Team Name: Cool Generation
Master Degree in Computer Science (AI curriculum), Academic Year: 2022/23
January 6, 2023
`https://github.com/lilf4p/aif-project.git`

1 Introduction

Genetic algorithms are a family of search algorithms inspired by the principles of evolution in nature. A popular examples of their application is to reconstruct a given image using various techniques. The goal of the project is to explore in depth this problem by using different shapes for approximating different types of images like RGB, grayscale and contours images. Using genetic algorithms, each candidate solution is a set of shapes.

We have been inspired for studying this topic by the book "Hands-On Genetic Algorithms with Python" by Eyal Wirsansky [1], which explores many applications of the genetic algorithms in different fields through the DEAP framework. We started by using the same framework for getting familiar with the topic, next we explored also usages of custom fitness functions, crossover and mutation operators specifically tailored to the problem and also of multi-objective algorithms. After that we implemented a from-scratch version of the algorithms to enhance performance and improve the genetic operators efficiency. Finally we realized a user-friendly interface to choose between the different algorithms, in order to run experiments on a given input image with the desired configuration.

In section 3 we will illustrate the workflow that the team adopted starting from some basic experiments to more complex ones.

1.1 Related works

The paper [2] investigates the application of genetic algorithms in the field of image processing, for example for reconstructing a binary image from a completely random image.

The paper [3] provides new genetic algorithms for Electrical capacitance tomography (ECT) image reconstruction. Results show that the method is efficient and capable of reconstructing high quality images with fast convergence.

1.2 Installation and Usage

The code exposes a command-line interface for repeating all the provided experiments, built with `typer` [4]. For installation and usage, refer to the `README.md` file in the GitHub repository (`https://github.com/lilf4p/aif-project.git`).

2 Methodologies

To conduct our experiments we used the open source framework DEAP, because it allows rapid prototyping and its design differs from most other existing frameworks. In particular, t tries to make algorithms explicit and to use transparent data structures, as opposed to more common black-box frameworks.

For our experiments, in general we encoded our individuals as lists or arrays of real numbers in the range $[0, 1]$, which represents information like point coordinates, colors and opacity. We used several crossover and mutation operators that are common for operating with such arrays, like `Simulated Binary Bounded` crossover and `Polynomial Bounded` mutation and also a random swapping of array items. We also exploited elitism as a strategy for mitigating *premature convergence* of the algorithms [1].

Elitism refers to the practice of preserving the best performing individuals from one generation to the next. This is done to ensure that the population does not lose its best solutions as it evolves over time, and also allows the population not to grow exponentially with each iteration.

As fitness functions we have used both the MSE, SSIM and other custom functions created by us within the experiments of contours reconstruction with lines and points.

3 Experiments

3.1 Results with RGB images

Our first experiments concerned the usage of several polygon types for approximating RGBA images. We have used only regular polygons since they seemed to better suit for these tasks. As fitness functions, we have used MSE and SSIM (Structural Similarity). We limited the size of polygons at least to the size of the image by setting the maximum size of the polygons radius to half of the minimum between height and width of the image.

We ran a number of experiments with different selection, crossover, and mutation methods and the same number of parameters, population size and configuration within 2000 generations for the reconstruction of Mona Lisa RGB image and the best result we got was MSE = 3182 after 3730 seconds 6.

3.2 Results with Grayscale images

After the approach with colored polygons we moved to experiments with grayscale images. One advantage of grayscale images is that their encoding only needs one value for shades of gray instead of the four channels in the RGBA. In this way the algorithms can perform faster by reducing the amount of information stored in memory. We have then changed the enconding of the shapes from $[x_1, y_1, ..., x_n, y_n, r, g, b, alpha]$ to $[x_1, y_1, ..., x_n, y_n, grey]$.

3.2.1 Ellipses

After trying with different polygons like triangles and squares, ellipses seemed to better approximate the input image in less time. Increasing the number of individuals and shapes in population did not improve the results, so we focused on improving the fitness functions by also exploiting multiobjective fitnesses. Figure 3 shows how the algorithm converged in 3000 generations and reached a better approximation combining MSE and SSIM fitness functions, whereas simple MSE converged too fast to a worse approximation. Figure 2 shows the best result found with ellipses and multiobjective fitness.

3.2.2 Text

The next experiments involved the reconstruction of an image using small strings of text. For this specific task DEAP framework was not used, since we have noticed that it is too slow and its general-purpose structure does not allow to optimize the code without rewriting some parts.

In these new experiments, crossover consists in swapping some random rows of the two parents' matrices to generate the child, while mutation consists in drawing a short string of text in a random position of the image. Compared to the previous implementation with bigger shapes, this algorithm focuses on small portions of the image, producing a good approximation with the majority of the details. We initially tried with Peak Signal-to-Noise Ratio, but after different runs we found that the Mean Squared Error performed better. The Figure 5 shows that the convergence of the algorithm was slow with respect to the number of generations but fast considering the elapsed time.

3.2.3 Lines

Another approach we worked on for grayscale images provides an algorithm that tries to reconstruct the outliers of the image using a set of lines with limited length. An individual is initially represented by an empty matrix having the same dimensions as the original image to be reconstructed. After having initialized the matrix, several rows are drawn inside it in a completely random way with a maximum length that is fixed ans passed as input. We used lines of different lengths depending on whether the algorithm had to reconstruct only the outlines or the whole image. After generating random lines, the algorithm draws these lines within the matrix by adding 1 in the region covered by those lines. The fitness function is computed between the absolute value of the difference between a matrix weight map and the original image. The weight map is a function that returns a number between 255 and 0 and is defined as following: $\texttt{int}\left(y = 255/e^(x/2)\right)$. If there are no lines covering a specific pixel the function will return 255, otherwise if there are many lines that cover a specific point the function will return a smaller value. This is a key point for the algorithm because it can generate not only black and white points but several grayscale values and this depends on how many lines cover a specific pixel. Finally, another important part of the algorithm is the mutation, which is relevant for speeding up of the algorithm because the population tries at each iteration to mutate through a random change of rows, but the latter is conserved if and only if the new individual has a better fitness score. In the experiment with the Mona Lisa image the algorithm started with a fitness function of 3785757, and after 300 generations (around 5 minutes), the fitness function was 263723.

3.3 Results with Contours

As part of our investigation, we decided to study the problem of approximating only the contours of a given image: in this setting, the target image is in grayscale format with black points corresponding to the actual contours of the original image, while the rest of the figure is white. For this problem we chose to use a population made up of either points or lines.

Standard metrics (e.g. MSE and SSIM) actually turned out to perform badly for this problem, since they do not capture properly the information in the image. For example, in this case MSE is equivalent to count how many points in the individual are different than the corresponding ones in the target image, but it does not provide any information about how they are structured. As a consequence of this, we developed our custom fitness functions.

For an individual made up of points, our approach is defined as following: if target image contains K black points and each individual contains N points, let $C = (x_c1, y_c1), ..., (x_K, y_K)$ be the coordinates of them and $I = (x1, y1), ..., (x_N, y_N)$ be the coordinates of the points of the individual. We define $d_C((x, y)) := min_{i=1,...,K}|x - x_ci| + |y - y_ci|$ and $d^*_C(I) := \sum_{p \in I} d_C(p)$. Goal of the algorithm is to minimize d^*_C, and in particular it holds that $d^*_C = 0$ if all the points of the individual cover the contours of the original image, i.e. the minimum value for d^*_C is 0. For crossover we have used both `Simulated Binary Bounded` and a random swap of the items of the two chromosomes, while for mutation we have used the `Polynomial Bounded` one (see section 2).

We have tested on several images, and the tests revealed that the points are actually quite capable of approximating the target image and that they tend to distribute themselves quite uniformly throughout the image, maybe also because point coordinates are drawn from a uniform distribution and because `Polynomial Bounded` mutation tends to favorite coordinates near to the original one. Figure 4 in the appendix shows the results obtained after 12000 generations and 110 minutes with a $400x300$ picture of the skyline of Singapore.

The reconstruction of the contours through lines is implemented by the same algorithm described in the Section 3.2 but with lines of shorter length and with an image preprocessing through `OpenCV`. Preprocessing is a main part because it can easily extract the outline of the image, after which the algorithm reconstructs the outlines using the lines and the same fitness function described above.

4 Conclusions

We explored different approaches to image reconstruction through Genetic Algorithms and from the experiments we got quite good results. We observed how `DEAP` works well as first approach, but a from-scratch implementation is necessary, since it is not optimized for usage with `numpy` arrays as individuals and for our specific use-cases. We learnt a lot from this experience, in particular how to design and test new fitness functions that are tailored to these problems, which are actually optimization problems. We learnt how to exploit features that are relevant for the image like the distances of points from the contours of an image, and we are now substantially much more practical with usage of this family of algorithms.

Despite the algorithms were tested on a set of sample images, we observed that the quality of the approximations seems to remain high when scaling to higher resolutions, for example with the contours approaches. In this direction we decided to build an interface which allows to easily repeat our best experiments and run custom ones. We think this can help to expand our studies and better understand the generalization capabilities of our algorithms.

References

[1] E. Wirsansky. *Hands-On Genetic Algorithms with Python: Applying genetic algorithms to solve real-world deep learning and artificial intelligence problems.* Packt Publishing, 2020. ISBN: 9781838559182. URL: `https://books.google.it/books?id=A0vODwAAQBAJ`.

[2] Seyedali Mirjalili et al. "Genetic Algorithm: Theory, Literature Review, and Application in Image Reconstruction". In: *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications.* Ed. by Seyedali Mirjalili, Jin Song Dong, and Andrew Lewis. Cham: Springer International Publishing, 2020, pp. 69–85. ISBN: 978-3-030-12127-3. DOI: `10.1007/978-3-030-12127-3_5`. URL: `https://doi.org/10.1007/978-3-030-12127-3_5`.

[3] Mou Changhua et al. "Image reconstruction using a genetic algorithm for electrical capacitance tomography". In: *Tsinghua Science and Technology* 10.5 (2005), pp. 587–592. DOI: `10.1016/S1007-0214(05)70123-1`.

[4] *Typer.* URL: `https://typer.tiangolo.com/`.

A Appendix

A.1 Relationship with the course

Genetic Algorithms are an example of simple yet effective search algorithms for solving problems in many different fields and were introduced in the course as part of Problem Solving section when discussing about Search in Complex Environments. We decided to go in deep by exploring one popular application of them that is substantially an example of optimization problem where the fitness function is designed to have a minimum when the target image is completely reconstructed, and we have experimented by ourselves how to design and optimize genetic algorithms by studying the characteristics of the problem.

A.2 GitHub metrics

The project can be found on the Github repository at the following url `https://github.com/lilf4p/aif-project.git`. Instructions to run the program can be found in the README file with details about implementation.

Number of commits for each member of the team (in alphabetical order) in the period from 23 October 2022 to 6 January 2023:

- Leonardo: 30
- Lorenzo: 21
- Michele: 23
- Mohammed: 37
- Salvatore: 43

Figure 1 shows the history of commits from 23 October 2022 to 3 Janauary 2023. The total number of commits on the repository amounts at 184.

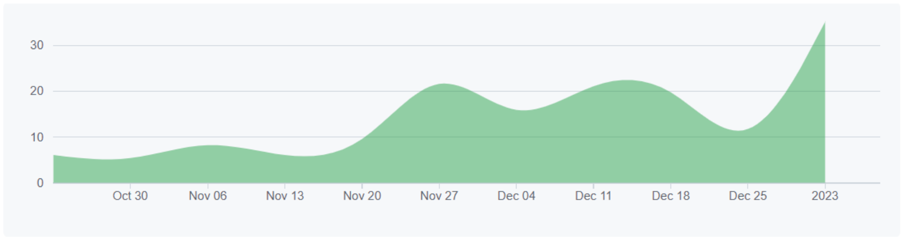


Figure 1: History of commits

A.3 Contributions

Leonardo worked on the grayscale case with polygons and ellipses and explored also multi-objective fitnesses for this problem and managed the repository.
Lorenzo completed the work on the grayscale case by exploring reconstruction with text.
Mohammed worked on the RGB case with different polygons and metrics.
Salvatore worked on reconstructing contours of an image, and in particular he developed the above-described distance-based method for individuals made up of points.
Michele worked on both contours and grayscale reconstruction with individuals made up by lines by developing its weightMap-based method for describing the quality of an approximation.

A.4 Images and Graphs

Here are listed the images and graphs which were referred in the Section 3.

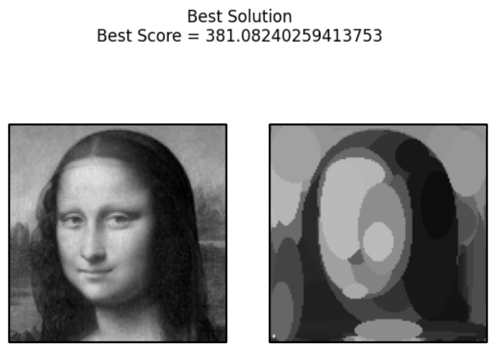
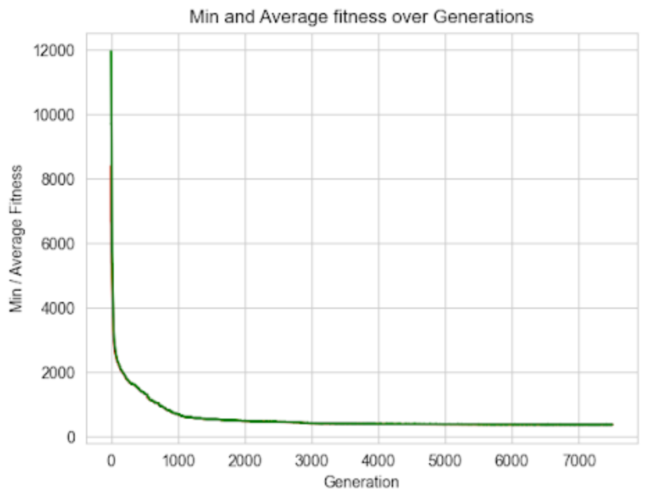
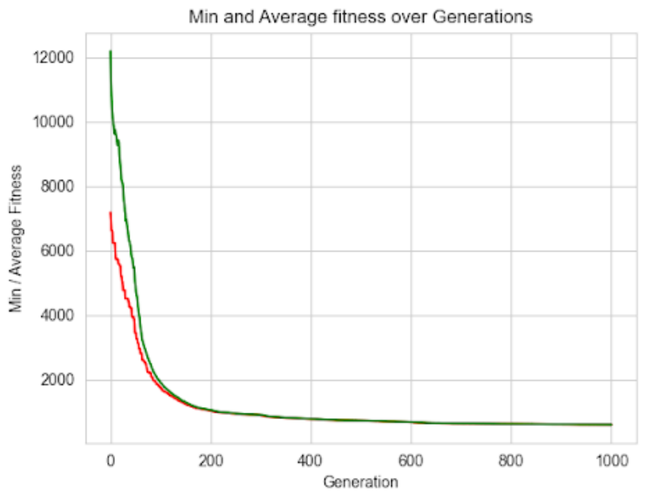


Figure 2: Best result with ellipses in grayscale.

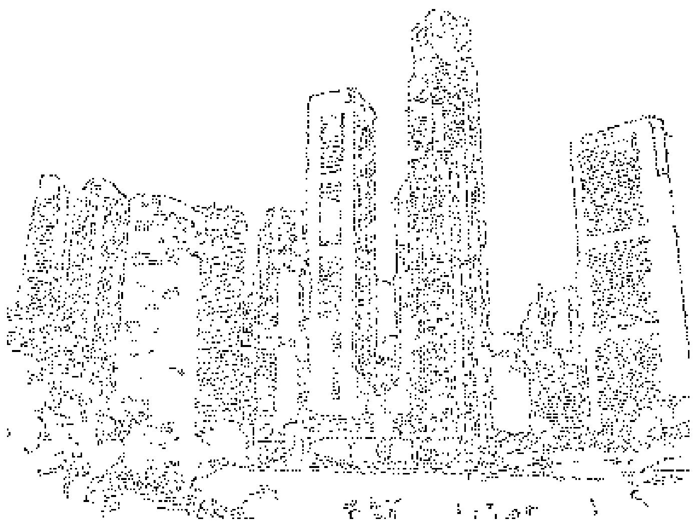


(a) Multiobjective MSE and SSIM.

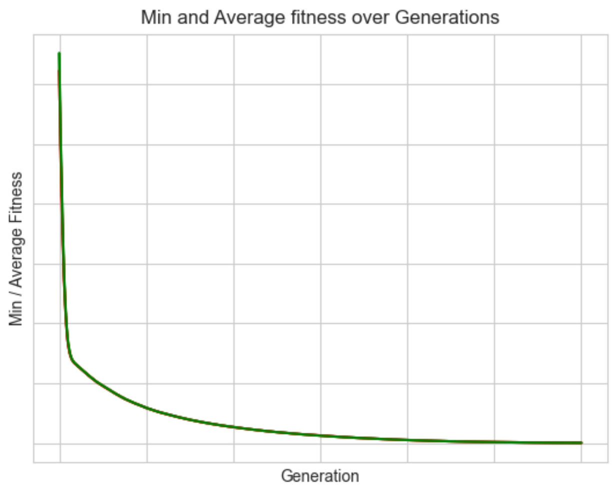


(b) Simple MSE.

Figure 3: Fitness value over generations for grayscale ellipses.

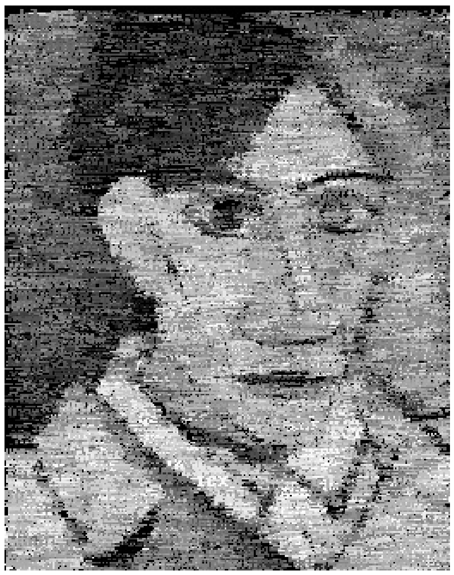


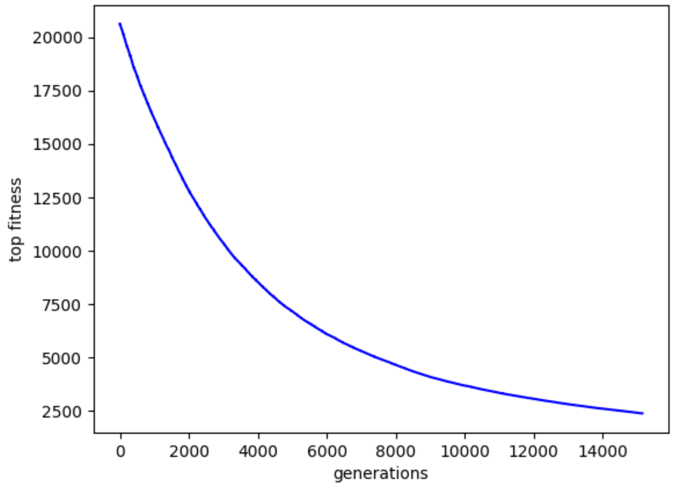
(a) Reconstruct with points a photo of Singapore.



(b) Fitness value over generations.

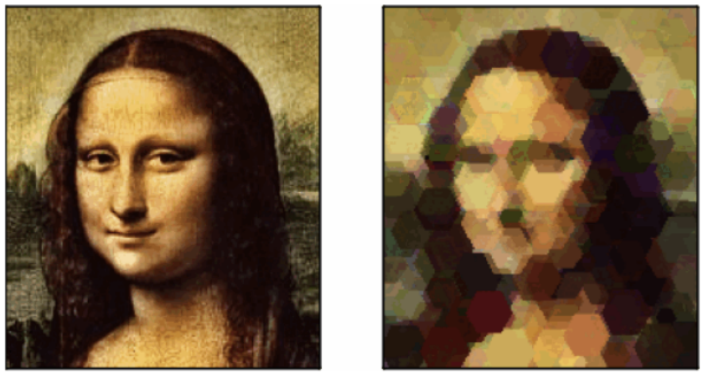
Figure 4: Best result with points.

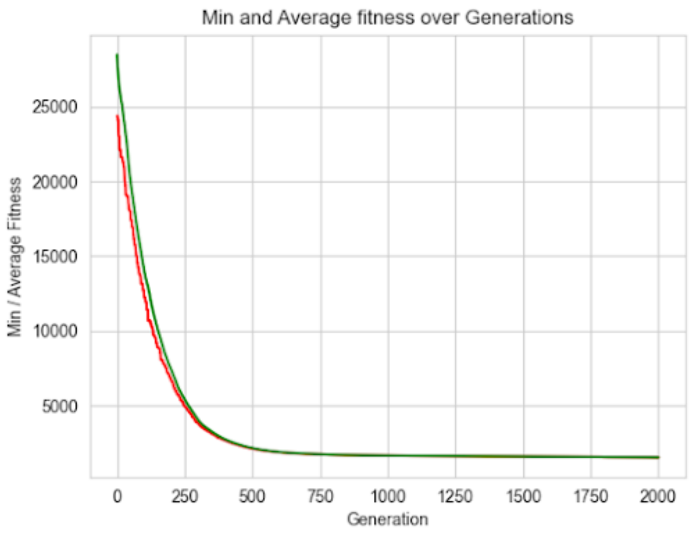

(a) Reconstruct with text a Picasso's paint.



(b) Fitness value over generations.

Figure 5: Best result with text.


(a) Reconstruct with RGB polygons monalisa image.



(b) Fitness value over generations.

Figure 6: Best result with RGB polygons.