EE496 HOMEWORK 2

Evolutionary Art

Due: April 29, 2019, 23:55

Aycan Doğa Hakyemez **ARC-201** **hdoga@metu.edu.tr**

Submission

- Send your homework compressed in an archive file with name "**eXXXXXXX_ee496_hw2.zip**", where X's are your **7-digit student ID number**. You will **not** get full credit if you fail to submit your project folder as required.
- Your work will be graded on its correctness, efficiency, clarity and readability as a whole.
- You should insert comments in your source code at appropriate places without including any unnecessary detail.
- Late submissions are welcome, but are penalized according to the following policy:
  - 1 day late submission : HW will be evaluated out of 70.
  - 2 days late submission : HW will be evaluated out of 50.
  - 3 days late submission : HW will be evaluated out of 30.
  - Later submissions : HW will NOT be evaluated.
- **Check** what you upload. Do not send corrupted, wrong files or unnecessary files.
- The homework is to be prepared **individually**. Group work is **not** allowed.
- **METU honor code is essential**. Do **not** share your code. Any kind of involvement in cheating will result in a **zero** grade, for **both** providers and receivers.

1. HOMEWORK TASK AND DELIVERABLES

In this homework, you will perform experiments on evolutionary algorithm and draw conclusions from the experimental results. The task is to create an image made of filled circles, visually similar to a given RGB source image (`mona_lisa.jpg`).

The implementations will be in Python language and you will be using OpenCV package to draw the images. You can install it using **`pip install opencv-python`** command. You can use **`import cv2`** to use it in your code.

You should submit a report in which your answers to the questions, the required experimental results and your deductions are presented for each part of the experiment. Moreover, you should submit your Python files where you implement the evolutionary algorithm and the visualizations of the experiments.

The codes should be well structured and well commented. The submissions lacking comments will be simply not evaluated.

The report should be in portable document format (pdf) and named as report.pdf.

2. THE EVOLUTIONARY ALGORITHM

The pseudocode for the algorithm is as follows:

```
Initialize population with num_inds individuals each having num_genes genes
While not all generations (num_generations) are computed:
      Evaluate all the individuals
      Select individuals
      Do crossover on some individuals
      Mutate some individuals
```

2.1. Individuals

Each individual has one chromosome. Each gene in a chromosome represents one circle to be drawn. Each gene has at least 7 values:

- The center coordinates (*x, y*).
- The *radius*.
- The color (red, green, blue, alpha).

The order of circle drawing is important. The circles should be drawn in the descending order of their radiuses. The first circle to be drawn is the one with the largest radius and the last circle to be drawn is the one with the smallest radius. The genes should be sorted accordingly.

The center does not have to be within image boundaries. However, if a circle is not within the image (it lies outside completely), the corresponding gene should be reinitialized randomly until it is.

The radius is a positive integer.

The red (*R*), green (*G*), blue (*B*) values are integers in the range of [0, 255].

The alpha (*A*) is a float in the range of [0, 1].

2.2. Evaluation

In order to evaluate an individual, its corresponding image should be drawn first. Note that the chromosome order is important. The pseudocode is as follows:

```
Initialize img completely white with the same shape as source image.
For each gene in the chromosome:
    Overlay <- img
    Draw the circle on overlay.
    img <- overlay x alpha + img x (1 – alpha)
```

The fitness function is

$$F = -\sum_{k\in\{R,G,B\}}\sum_{\substack{0\le i<width\\0\le j<height}}\left(src_{i,j,k} - img_{i,j,k}\right)^2$$

where $src_{i,j,k}$ is the pixel value, in i^{th} column and j^{th} row, in channel k in the source image. Similarly, $img_{i,j,k}$ represents the pixel value in the individual's image.

2.3. Selection

num_elites number of best individuals will advance to the next generation directly. The selection of other individuals is done with tournament selection with size *tm_size*.

2.4. Crossover

num_parents number of individuals will be used for crossover. The parents are chosen among the best individuals which do not advance to the next generation directly.

Two parents will create two children. Exchange of each gene is calculated individually with equal probability. The probabilities of child 1 having $gene_i$ of parent 1 or parent 2 have equal probability, that is 0.5; child 2 gets the $gene_i$ from the other parent which is not chosen for child 1, where $0 \leq i <$ *num_genes*.

2.5. Mutation

The mutation is governed by *mutation_prob*. While the generated random number is smaller than *mutation_prob* a random gene is selected to be mutated (same as in N Queen Problem in the lecture notes). All individuals except the elites are subject to mutation.

There are two ways a gene can be mutated:

- Unguided:
    - Choose completely random values for x, y, radius, R, G, B, A.
- Guided:
    - Deviate the x, y, radius, R, G, B, A around their previous values.
        - $x - width/4 < x' < x + width/4$
        - $y - height/4 < y' < x + height/4$
        - $radius - 10 < y' < radius + 10$
        - $R - 64 < R' < R + 64$
        - $G - 64 < G' < G + 64$
        - $B - 64 < B' < B + 64$
        - $A - 0.25 < A' < A + 0.25$
    - The values should be corrected to a valid value if they are not.

3. EXPERIMENTAL WORK

You will experiment on several different hyperparameters. Namely,

- Number of individuals (*num_inds*)
- Number of genes (*num_genes*)
- Tournament size (*tm_size*)
- Number of individuals advancing without change (*num_elites*)
- Number of parents to be used in crossover (*num_parents*)
- Mutation probability (*mutation_prob*)
- Mutation type (guided or unguided)

To see the effect of each hyperparameter, run the algorithm for each value in a row while using default values for the other hyperparemeters. The default values are bolded. Use *num_generations*=10000.

num_inds	5	10	**20**	50	75
num_genes	10	25	**50**	100	150
tm_size	2	**5**	10	20	
frac_elites	0.05	**0.2**	0.4		
frac_parents	0.2	0.4	**0.6**	0.8	
mutation_prob	0.1	**0.2**	0.5	0.8	
Mutation type	**guided**	unguided			

For each hyperparameter, explain its effect and give the value which produces the best result. For each experiment, you need to include

- fitness plot from generation 1 to generation 10000
- fitness plot from generation 1000 to generation 10000
- the corresponding image of the best individual in the population at every 1000th generation.

4. DISCUSSION

Suggest three changes to this evolutionary algorithm which may provide faster and/or better convergence. The changes should not be only using a different value for a hyperparameter. Show the result empirically and explain.

5. REMARKS

- Because of the nature of assignment in Python, be careful how you use them. Consider the following:

```
a = [[1, 2], [3, 4]]
b = [a[0], a[0]]
b[0].append(5)
print(b) # prints [[1, 2, 5], [1, 2, 5]]
print(a) # prints [[1, 2, 5], [3, 4]]
```

  At times, you may want to use `deepcopy` from `copy` module.

- You can use `cv2.circle` to draw circles and `cv2.addWeighted` to blend images.
- Because the underlying type for the image is `np.uint8`, be careful of possible overflows.
- You are advised to use classes to implement genes, individuals and populations. It will make debugging easier.
- You are strongly advised not to do your homework on the last day of submission. The experiments may take a while to finish.