METU EE 496
Introduction to
Computational Intelligence

Evolutionary
Art

Homework 2 - Evolutionary Algorithms

Due: 23:55, 19/04/2020

Late submissions are welcome, but 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.
- 4 or more days late submission: HW will not be evaluated.

You should prepare your homework by yourself alone and you should not share it with other students, otherwise you will be penalized.

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 (painting.png).

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 single report in which your answers to the questions, the required experimental results (plots, visualizations etc.) and your deductions are presented for each part of the homework. Moreover, you should append your Python codes to the end of the report for each part to generate the results and the visualizations of the experiments. Namely, all the required tasks for a part can be performed by running the related code file. 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 *hw2_name_surname_eXXXXXX* where *name*, *surname* and *X*s are to be replaced by your name, surname and digits of your user ID, respectively.

1 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
```

1.1 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, $radius \in \mathcal{Z}^+$
- The color (red, $R \in \mathcal{Z}$, green, $G \in \mathcal{Z}$, blue, $B \in \mathcal{Z}$, alpha, $A \in \mathcal{R}$) where $(R, G, B) \in [0, 255]^3$ and $A \in [0, 1]$

The order of the tasks to perform circle drawing is important. The circles should be drawn in the descending order of their radii. 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.

1.2 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 <image> completely white with the same shape as the <source_image>.
For each gene in the chromosome:
  overlay <- image
  Draw the circle on overlay.
  img <- overlay x alpha + image x (1 { alpha)
```

The fitness function, F, is

$$F = -\sum_{k \in \{R,G,B\}} \sum_{\substack{0 \leqslant i < \text{width} \\ 0 \leqslant j < \text{height}}} (\text{source\_image}_{i,j,k} - \text{image}_{i,j,k})^2 \tag{1}$$

where $\text{source\_image}_{i,j,k}$ is the pixel value, at i^{th} column and j^{th} row, in channel k in the source image. Similarly, $\text{image}_{i,j,k}$ represents a pixel value in the individual's image.

1.3 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>.

1.4 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 $0 \leqslant i <$ <num_genes>.

Table 1: Parameter configurations to be experimented.

<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			

1.5 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.
    - $x - \frac{\text{width}}{4} < x' < x + \frac{\text{width}}{4}$
    - $y - \frac{\text{height}}{4} < y' < y + \frac{\text{height}}{4}$
    - $radius - 10 < radius' < radius + 10$
    - $R - 64 < R' < R + 64$
    - $G - 64 < G' < G + 64$
    - $B - 64 < B' < B + 64$
    - $A - 0.25 < A' < A + 0.25$
  - The values should be corrected to a valid one in case they are not.

2 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 given in Table 1 while using default values for the other hyperparemeters. The default values are bolded. Use $< num\_generations >= 10000$.

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 1000^{th} generation.

3 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.

4 Remarks

You may find the following useful...

1. 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.

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