UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science

# Artificial Intelligence
# COS314

Project 3: Genetic Algorithms

**Due: Friday 3 June, 14:00**

For this project you will develop a program to evolve computer generated picture mosaics. A picture mosaic is obtained by sectorizing an original color image into equally sized cells, and then by replacing each cell with a smaller image, selected from a database of images, such that the characteristics of the smaller image closely match that of the original image. You will make use of a genetic algorithm to find the best combination of smaller images to accurately reflect the original image.

The first thing for you to do, is to read about creating picture mosaics, or also referred to as image montages. You may want to look at

- Michael Troebs, *Algorithms to create Image Montages* by Michael Troebs
- ML Michelane, MP Medel, *Understanding Photomosaics*, Dr Dobb's Journal, No 330, Nov 2001, pp 58-63.

You have to find our own database of images to use, but make sure that you have a sufficiently large database. The larger the database, the more accurate the mosaic will be. Also, you will have to write some scripts to covert all of the images to the same size, and to raster (`.ras`) images. Your program will be tested using a `.ras` image, and a database of `.ras` images. You can not assume any size of the images that will be in the database that will be used to test your program.

You may make use of libraries for manipulation of the images. You are also allowed to make use of any data structure libraries that you may need. However, you have to write your own genetic algorithm (GA) code. You are not allowed to use any GA libraries, or any GA code not written by yourself. If we do find that you have not written your own GA code, you will get no marks for this project, and plagiarism actions will be followed. Although the objective of this project is to implement the AI (most of the marks will be for the AI), you also have to provide a good GUI, and write modular, documented, well-designed code.

You may implement the project either in C++, C, Java, or Scala, and must run on Linux. You have to provide a Makefile, ant file, maven file or any other alternative project script that will run on the Linux distribution as provided in the Informatorium.

Submit all your code, project file, and a pdf document wherein you provide instructions on how to compile and use your program. If you make use if any AI in addition to the specifications here, please describe these in our pdf document. The pdf document must be in the root folder. These should be submitted as a compressed tar (`.tgz`) file. Please do not include your images in the tar file, and no executables. Your program should be compiled to the executable, `mosaic`. The program will be executed in the following format:

```
./mosaic image_directory image.ras x y
```

where

- `image_directory` is the name of the directory where all the raster images will be stored. Your program should be written such that the images in the given directory is used.
- `image.ras` is the original image for which a mosaic will be constructed.
- `x` is the number of rows of the grid fitted over `image.ras`.
- `y` is the number of columns of the grid fitted over `image.ras`.

The final output, in other words, the mosaic, is to be stored under the file name *image_mosaic.ras*, where *image* is the name of the image, as provided on the command line.

Now, some guidelines to write the GA to evolve the mosaic:

- Each individual (chromosome) will represent a solution, i.e. a mosaic for the original image. Thus, if the original image is divided into 500 cells, then each individual will consist of 500 images from the image database. That is, each gene of the chromosome will represent one image. You need to define your own mapping from individual to 2D picture mosaic.
- Initialization of the first population: Decide on a number of individuals for the population. Then randomly select from the database an image for each of the genes for each of the individuals. However, you should take note of the following constraints: No pair of adjacent cells of the original image may be replaced with the same image, and you should use as many as possible of the images in the database (try to reduce the number of times that the same image is used in the same chromosome).
- Selection: Use tournament selection as default. For more marks, you may include any other selection method, but the user should be allowed to specify which to use.
- Crossover: Implement two-point cross-over. For more marks, you may include any other selection method, but the user should be allowed to specify which to use. Cross-over occurs at a given probability, which is a value between 0 and 1. If the cross-over probability is 0.9, then after two parents have been selected, then generate a random number between 0 and 1. If this number is less than or equal to 0.9, then perform cross-over. If not, then the parents are not allowed to produce offspring. You may implement additional cross-over operators, from which the user can choose (for extra marks).
- Mutation: This is done by replacing a gene with a new randomly selected image, but such that the constraints above are satisfied. Mutation occurs at a given probability using the same method described above for cross-over. You will find that random mutation on its own will work, but not that well. So, you will need to devise a more efficient way of mutation.
- Selecting the new population: Select the best individuals from the parents and the offspring to form the new population. That is, if the population has 100 individuals, you select the 100 best performing individuals from the parents and the offspring. In addition to this, you are welcome to experiment with different methods.
- And now the fitness function. Here I want you to devise your own fitness function. The fitness function needs to measure how closely the set of images (as given by the genes of the individual) match the original image. For this you need to define a method to measure the distance between the mosaic and the original image. One way of doing this is to calculate for each cell of the original image the average R,G,B values over the pixels of that cell. Then do the same for each image of the chromosome. Then, for each cell use the Euclidean distance or Riemersma's formula (refer to *troeb.pdf* to calculate the similarity between that cell and the corresponding gene (image)). If there are 500 cells, this will result in 500 average values. Now calculate the overage over these to give you a distance between the original and mosaic images. This can serve as a fitness, and the objective is to have this as small as possible. In addition to this, you may want to add penalties if any of the constraints are violated. This is just a suggestion. Play around and see if you can define a better fitness function.

For all of the GA parameters (e.g. population size, cross-over probability, mutation probability, tournament size, etc), you will have to play around to find the best values that will give you a very good mosaic. The user should be allowed to select his own values for these probabilities. You are allowed to add any other operators not mentioned above, as long as it improves the quality of your results.

In your pdf file, you need to describe for each of the points above, exactly how you have implemented it. Please give attention to this and do it properly: write enough detail. This will also be used to give you a mark.