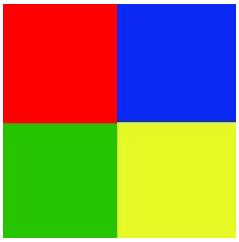
CP365 - Genetic Algorithms

Assignment Overview

For today's program, we're going to simulate evolution to help solve a problem. The problem that we're going to work with is reproducing a fine work of art (OK, maybe we'll start with a simple geometric pattern...but isn't some fine art really just simple geometric patterns???).

Let's start with a simple image and think about how our genetic algorithm will work. Consider the four-color flag here.



Let's pretend that the image is much too difficult to reproduce by hand in an image editing program, so we'll have our evolutionary program create it for us. Each individual in our artificial population will be a single solution to this problem. In this case, a solution is simply a picture. A *good* solution is a picture that closely resembles our target picture.

During each *epoch* in our simulation, we'll need to do several things:

- Evaluate the *goodness* of each of our solutions by comparing them to the target image. We will use a *fitness function* to determine how good each solution is.
- Randomly mutate some of the solutions
- Crossover good solutions with other good solutions. This is like sexual reproduction in a real biological population.
- Leave some of our solutions alone.

Anatomy of a Solution: Each individual solution will be an artificial chromosome. This chromosome will define the parameters of the solution. For our art-recreation domain, each chromosome should be a list of colored polygons. For simplicity, I recommend making each chromosome the same number of polygons...10 might be enough for this first example. Each polygon in the chromosome is defined by a color and a list of (x,y) polygon point coordinates. Again, for simplicity I recommend giving each polygon the same number of vertices (around 4 would probably be fine).

Mutations: For each value (color or vertex coordinates) in an artificial chromosome, there is some chance of random mutation. Mutation changes the old value into a new, random value.

Crossover: For each chromosome, there is some chance of it reproducing with another chromosome. This chance should be higher for chromosomes with higher fitness function ratings.

Fitness Function: Each solution needs to be evaluated each epoch of the simulation. How good a particular solution? For our problem, the *fitness* of a particular solution is a measure of how closely it matches our target image. You could do a pixel by pixel comparison, but this would be fiendishly slow. Instead I recommend taking a random sample of some pixels, find the difference in those pixels, and use that to estimate the difference of the whole image.

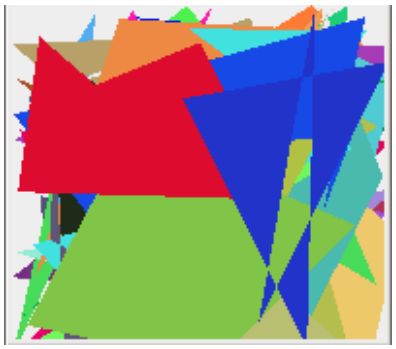
I imagine there are lots of different ways to compare the corresponding pixels. The easiest way that I could think of uses BufferedImage objects for both the target image and the intermediate solution images.

Simulation: First write your GA and all its functions. You may use this code to help you get started. To run the full simulation, start with a population of randomly-created chromosomes (try 50 to begin with). For each epoch or generation in the simulation, perform mutations and crossovers to create a new population. Measure the average fitness of all individuals in the population and report this number each 10 epochs. You should notice that the general trend of the average fitness is going **up**. You may need to play around with your mutation and crossover rates to get a

reasonable performance. I tested with a mutation rate of 0.01 (1 in 100 mutate) and a crossover rate of 0.6 (60% of the new generation is generated from crossovers).

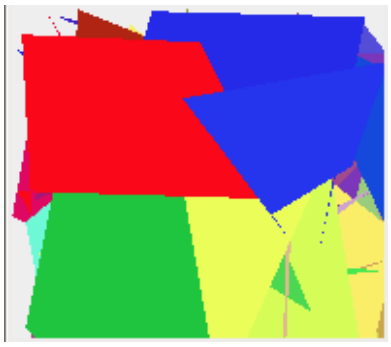
Example: To see how well your algorithm is working, it helps to visualize the highest ranked solution every 10 epochs. You should get something like this:

epoch 100

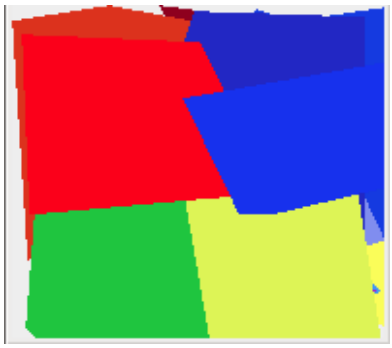


epoch 200

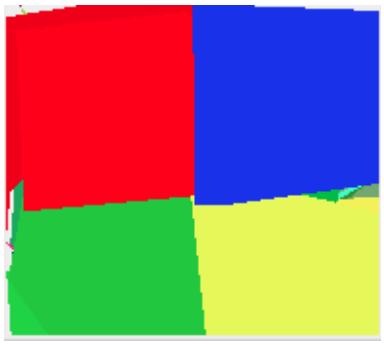
epoch 500



epoch 1000



epoch 2000

epoch 4000 

epoch 10000 

Fine Art: Once you have your algorithm working, try it on some other pictures! You will notice that some images are easier than others to get good representations.

Things to consider:

- Did your algorithm reproduce a work of art nicely? If not, what were some of the problems?
- How much did the mutation and crossover rates affect the success of your algorithm?
- Were you surprised how long it took to run your algorithm? Was it faster or slower than you imagined?
- Do you think GAs would work for other problems? When would they be a good choice? A poor choice?