Senior Capstone 2016 – Image Mosaic – Veishea Grebin

How the Genetic Algorithm Works

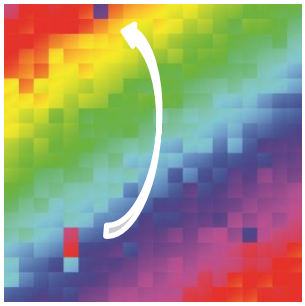
It might help to have ArrayContents.pdf available.

1. Image numbers are stored into the generation children. The image numbers start at 0 and continue until they reach the number of tile images uploaded and then start over. For example if the mosaic is 3x3 and only 4 images are uploaded it will be filled(0 1 2 3 0 1 2 3 0) and then next child will start where it left off (1 2 3 0 1 2 3 0 1). Also the children's tile scores is initialized(empty tiles).
2. Once all children are filled the image numbers in each are randomized.
3. Next the scores are computed to find the best matches. The following is done for every tile in every child. This is checking to see how close the new tile matches the corresponding position on the original image.
   (OriginalTileRed – ChildTileRed)^2 = TileRedScore
   (OriginalTileGreen – ChildTileGreen)^2 = TileGreenScore
   (OriginalTileBlue – ChildTileBlue)^2 = TileBlueScore
   TileRedScore + TileGreenScore + TileBlueScore = TileScore
4. The total child score is computed by adding all the tile scores together for that child. These are then rearranged from best to worst so I know what children have the best arrangements.
5. I then split the generation of 40 into 4 groups of 10. Group 1 with best scores(smaller numbers) and Group 4(bigger numbers) with the worst scores.
6. First I work on Group 3 and 4. I create a separate array for the current child(child 1 in Group 3) that has the image number, score, and location in the child and rearranges them based on their score. I start with the location that has the best score in Child 1 in Group 3 and put that image in that same location in Child 1 in Group 4. For example if image 2 in location 1(green) is the best match(1) it will put that number in location 1 in Child 1 in Group 4(2). Then I find 2 in the Child 1 in Group 4 and put the number that was replaced in that location. If there is more than one image with that number in the second child(C1 G4) I swap it with the one that has the worst score. So if image 2 in position 0 has a worst score than image 2 in position 3 I will replace the image in position 0(3).

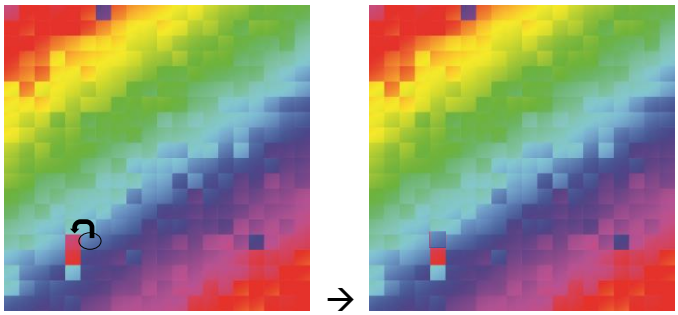
	C1 G3	C1 G4	C1 G4	C1 G4
0	0	2	2	1
1	2	1	2	2
2	3	0	0	0
3	1	2	2	2
	(1)	(1)	(2)	(3)

7. This process continues until I have copied the images with the best scores(half the child) in Child 1 in Group 3 to Child 1 in Group 4. This then continues for Child 2-10 in Group 3 to Child 2-10 in Group 4.
8. Then after Group 3 has is copied into Group 4 I randomly swap 10% of the tiles.
9. I do steps 6-8 again for Group 2 → Group 3.

10. Then steps 6-7 for Group 1→ Group 2.
11. Then Group 1(the top 10 children) is different. I do not change the top child. For the rest I look at the scores and determine if they are “good enough.” I look at each child individually and look at the top half of the best scores. If the score is less than 1000(2000 is still pretty good too but 1000 works better) I leave the image where it is. If not I swap it with the one above or below it(random).
12. After going through the good matches I look at the bottom half. I go through the ordered list, starting at the middle, to find the first score above 1000. Until I get to the bottom 15% I randomly swap bad matches with other bad matches.



Then when I get to the bottom 15% I randomly swap 50% of the time, and copy the tiles around the bad tile(if the new tile is good enough) and replace the bad tile with the good one the other 50% of the time.



13. Next I go back to step 3 and continue until the progress window pops up or current generation gets to 800 or best score is less than 50000(which has only happened when I use the images that were chopped up from the original). If I ever get to 800 it usually means the tiles do not match the original image well enough.