Algorithmic Research for Recreating Image out of Primitive Shapes

Michael Ehrlich

Contents

Intro

We will start this project by examining different algorithms that can solve our problem.

Our problem requires to generate an Image from a primitive shape with some attributes, the primitive shape image should look as close as it can to the original input image.

From that problem description we can understand that we will start with some random noise or nothing and each iteration of the algorithm we will get closer to our target.

We will look at the following algorithm in this project:

From the project PDF:

Genetic Algorithm, SGD, Simulated Annealing.

Because SGD is used in machine learning as an optimizer, we can examine its **batch, mini-batch and momentum** versions as well, also we can’t forget the popular **Adagrad, RMSProp Atom** and **quasi-Newton,** which are all method to find minimum point in a complex function.

Because we mentioned machine learning we can think about using some **style transfer** algorithm as well.

Filtering Algorithms

In order to decide in which algorithm are we going to use we need to know how to compare them.

I came across paper [1] about best practices in comparing optimization algorithms.
The paper talks about how to practically compare optimization algorithms and how to use benchmarks and graphics in order to establish this. Unfortunately, this project is time constraints and I cannot implement all the popular ones and compare them, but I did learn from this article on what to look at when searching for a previous comparison.

We will focus on mainly on efficiency, reliability and if its suites our problem.

If one algorithm runs faster, uses less memory, and returns a better final function value, on all possible problems, then it can be considered better than the alternative. Of course, in practice such a clear conclusion is rare, therefore interpreting the conclusion of algorithmic comparison can be tricky.

Paper [2] compare several optimization algorithms and tries to classify which type of algorithm can suite which problem. The paper compares several algorithms on water calibration problems, so the general conclusion does not quite fit our desires, but the general idea does.

Suppose we have an objective function $f(x)$, the algorithm we'll choose to use depends on the properties of $f(x)$:

1. $f(x)$ has a single extremum
   a. If its derivatives can be computed, then **gradient-based methods** can be used or variable metric methods, like **DFP** and **BFGS**.
   b. $f(x)$ is not analytically expressed, direct search methods can be used such as **Nedler & Mead 1965.**
2. No assumptions are made about the properties of $f(x)$, so it is a multi-extremum function which is not expressed analytically.

We can try to define a cost function between the original image and the generated one and try to optimize it using methods from 1.a but this will cause funny results as stated in paper [3]. The only way to make $f(x)$ analytically expressed is by using Style Transfer methods which we will cover later.
In addition, our problem is not known for a single extremum because of its generative nature, therefore our problem is of kind 2.

Comparing Relevant Algorithms

The following comparation is only regarding to our specific problem and does not aim to the general case of any problem.

The algorithms I found that can suite our problems are:

1. genetic algorithms.
2. simulated annealing.
3. particle swarm optimisation.
4. Machine learning approach such as style transfer.

Paper [2] has tested only GA from the above. It was efficient and reliable but sometimes converges prematurely to a local maximum but overall had good performance.

<u>Machine Learning:</u>

Although Style Transfer is completely generic to the style and will have great result, it does not suite for our problem due to the lack of data. Our program has access to a primitive shape and a test image therefore is not a machine learning problem, but this was worth to mention.

We are left with heuristic algorithms: GA, SA and PSO, all borrowed from nature and biology.

<u>GA vs SA:</u>

Unlike SA, GA generates not a single candidate solution but an entire population of them. In practice GA gets better results than SA but at a higher computation cost.

This con can almost be ignored due to the GA ability of parallelism, it's trivial to do so because the individual "search agents" comprising each population do not need to exchange messages--i.e., they work independently of each other. Obviously, that means *GA computation can be distributed*, which means **in practice, you can get much better results (closer to the global minimum) and better performance (execution speed).**

<u>GA vs PSO:</u>

PSO and GA are similar in approaches and both gets accurate results although proven that in most cases PSO is more efficient and faster to solve most problems than GA as seen in MIT paper [4] and [5].

Conclusion

We have looked at several of the most popular optimization algorithms out there. First, we had to understand what kind of problem we are facing so we can eliminate the less fitting algorithms. Second, we compared the remaining algorithms to our problem, what we can and cannot do, that way we eliminated the style transfer approach.

At last, we had 3 metaheuristic approaches left that can suite our problem. The Genetic Algorithm was far better than Simulated Annealing in performance and score but failed in comparison of the **Particle Swarm Optimization** due to its superior efficiency.

Overall the chosen algorithm will be the **Particle Swarm Optimization.**

Algorithm Description

```
1. Load the original image.
2. Calculate the average RGB Channels.
3. Create new Image and set its background to the original average image. (This will
   force the shape generation to focus on the details and not as much on the
   background).
4. SET prevImg as the new image created.
5. Until we have reached max shapes:
      a. Initiate swarm with N particles. Each particle will represent a shape.
            i. While PSO is still converging:
           ii. For each particle:
                  1. Calculate fitness value:
                        a. Draw the image on the prevImg and SET it as
                           DrawnImg.
                        b. Calculate difference using RMSE between the drawn
                           image and original.
                        c. Update the particle personal best if necessary.
          iii. Update global history best if necessary.
           iv. Use PSO algorithm equation to move the shapes on prevImg.
      b. SET prevImg as the global best. (now we will build on top of it).
```

References

[1]	Vahid Beiranvand, Warren Hare, and Yves Lucet	2017	**Best Practices for Comparing Optimization Algorithms**
[2]	D.P. Solomatine	1998	**Genetic and other global optimization algorithms - comparison and use in calibration problems**
[3]	Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy	2015	**EXPLAINING AND HARNESSINGADVERSARIALEXAMPLES**
[4]	Rania Hassan, Babak Cohanim, Olivier de Weck		**A COPMARISON OF PARTICLE SWARM OPTIMIZATION AND THE GENETIC ALGORITHM**
[5]	Voratas Kachitvichyanuku	2012	**Comparison of Three Evolutionary Algorithms: GA, PSO, and DE**