From Polygons to Image

Chen Li
Master student of Computer Science in UCI

Disi Ji
Master student of Computer Science in UCI

What did we do?

In this project, we aim at using a fixed number of polygons with a fixed number of vertices to approximate a given image. For each polygon, the position of each vertex, RGBA of the shape need to be decided.

Previous work to solve this problem used the random search method, i.e., to randomly modify one polygon each time, and then decide whether to accept the modification by calculating the change of fitness introduced by the it, where fitness is defined to be the distance between the result and the original image.

We used three different methods, including gradient descent, simulated annealing, and genetic algorithm, to improve the performance.

Experiments are conducted to compare the performance

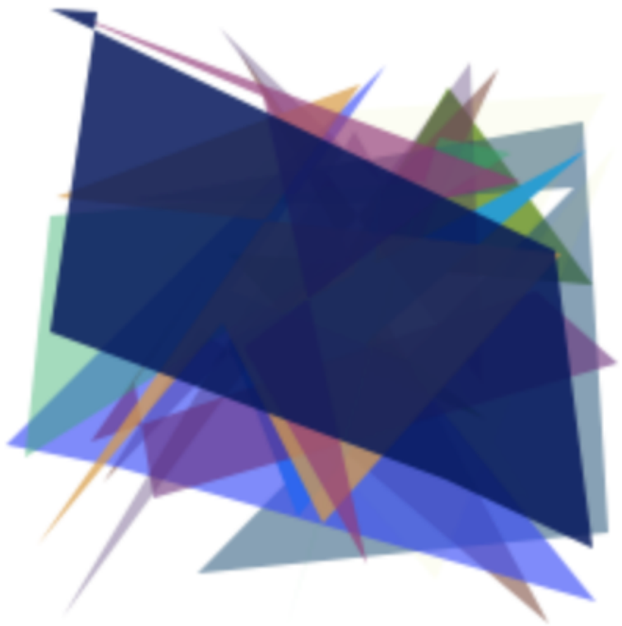
Random Search

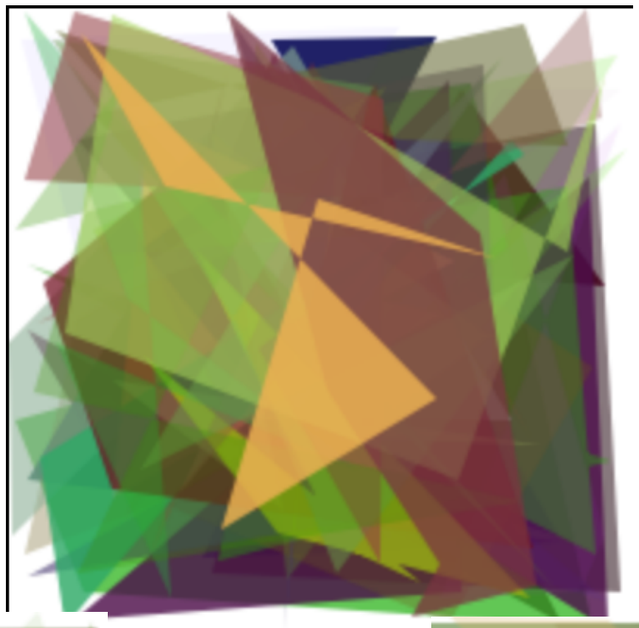
Randomly change attributes of polygons, accept the change when result is better.

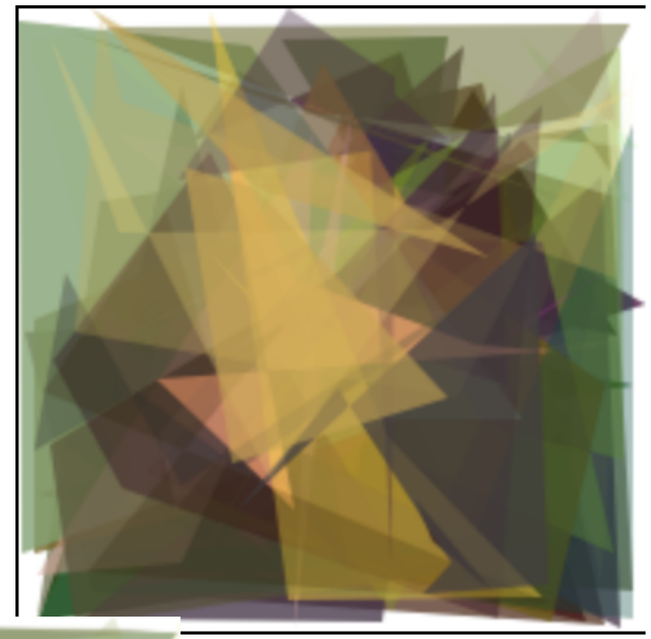
Algorithm 1 Random Search

Shapes[$\#ofpolygons$] ← Randomly initialize each polygon
while ($StoppingCriteria$) **do**
  $i = rand(\#ofpolygons)$
  NewPolygon ← Randomly generate a new polygon
  ShapesNew = Shapes
  $ShapesNew[i] = NewPolygon$
  if ($fitness(ShapesNew) > fitness(Shapes)$) **then**
    Shapes = ShapesNew
  end if
end while

Algorithm	Parameters	Fitness	Mutations	Elapsed Time
original	None	61.60791503267974	100	0.8070001602172852
original	None	81.0986339869281	1000	8.330000162124634
original	None	88.11807843137255	5000	41.742000341415405
original	None	90.5611209150326 7	10000	83.507000207901
original	None	93.96922875816993	50000	570.0190000534058







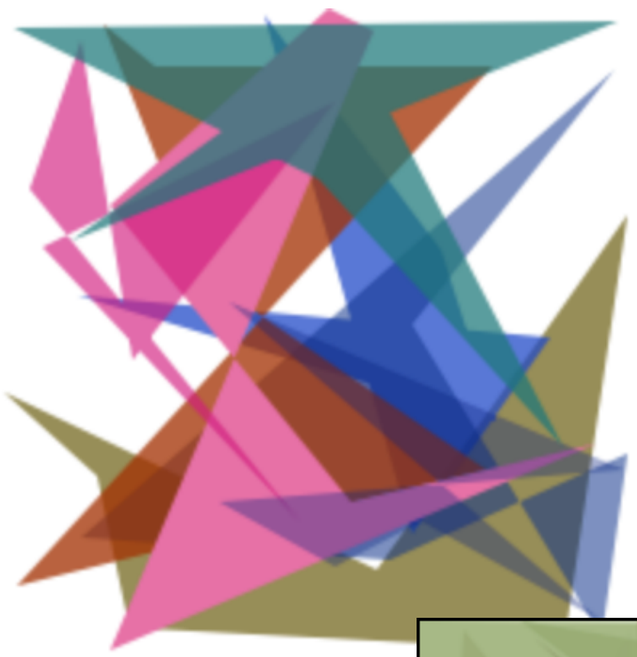
Gradient Descent

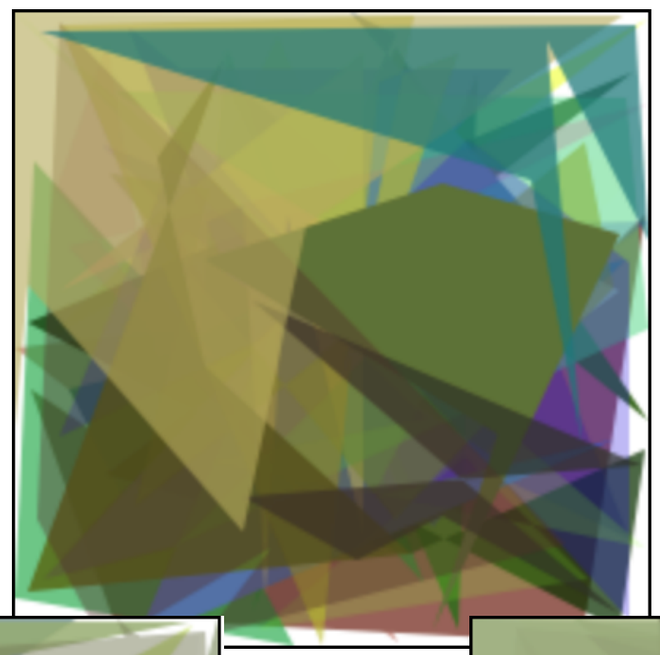
If last modification of shapes results in the increment of fitness, it is very likely that to continue modifying the shape in this direction will still increase fitness.

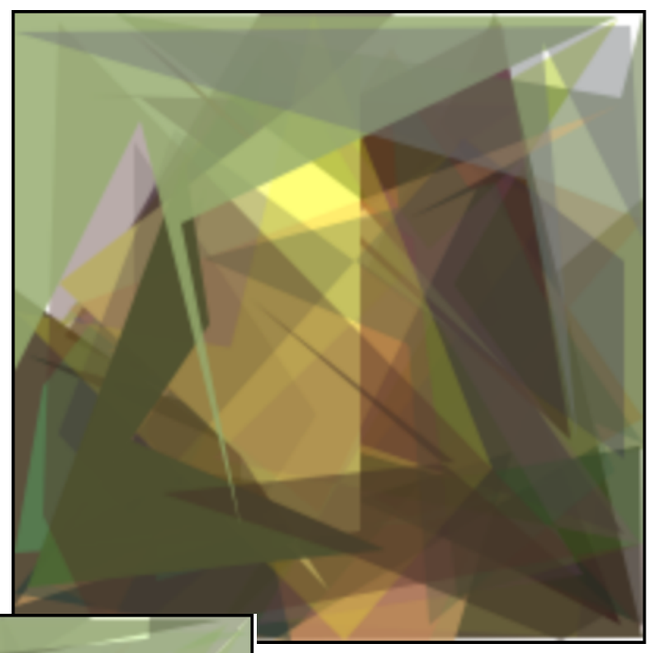
Algorithm 2 Generate NewPolygon with Gradient Descent

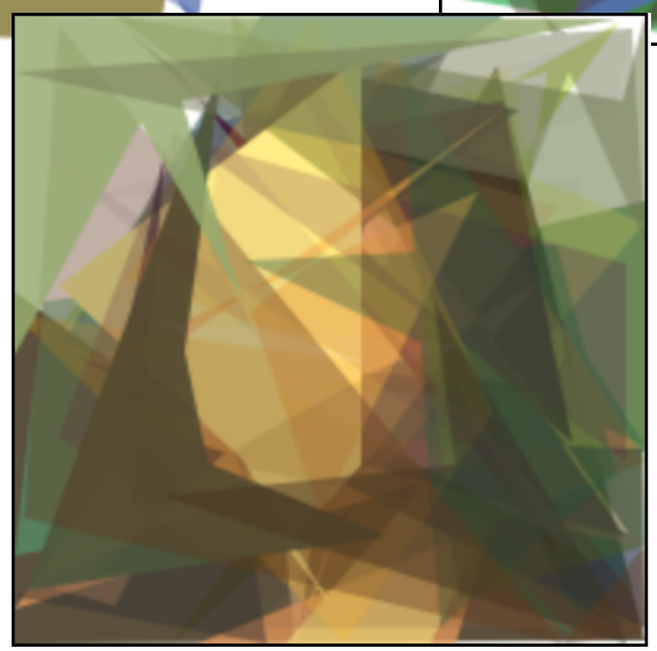
if $(fitness(ShapesNew) > fitness(Shapes))$ **then**
    i = index of the changed the shape
    Step = ShapesNew[i]-Shpes[i]
    NewPolygon += Step
else
    NextPolygon ← Randomly generate a new polygon
end if

gradient	None	67.35556862745098	100	0.803999008178711
gradient	None	83.92560784313726	1000	8.411999940872192
gradient	None	91.20078431372549	5000	41.84599995613098
gradient	None	92.94537581699348	10000	87.25099992752075
gradient	None	95.00325816993464	50000	419.02999997138977









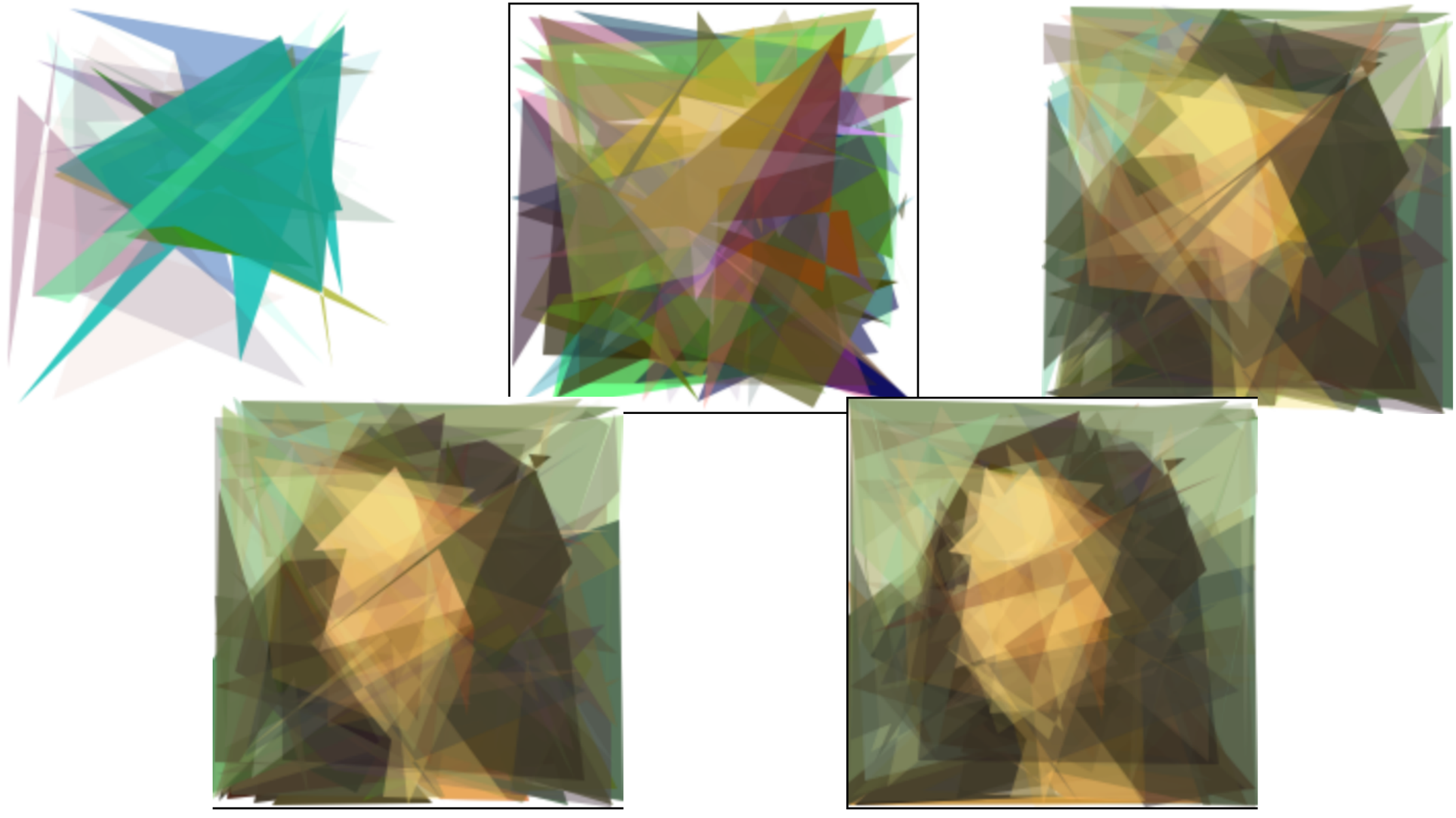
Simulated Annealing

"Metropolis algorithm" is employed, i.e., bad modifications are also accepted with a certain probability.

Algorithm 3 Accept Modifications using Simulated Annealing

$delta = fitness(ShapesNew) - fitness(Shapes)$
if $(delta \geq 0)$ **then**
    Shapes = ShapesNew
else
    $q = Bolzmann(delta)$
    $p = q/(1+q)$
    $t = rand(1.0)$
    if (t<p) **then**
        Shapes = ShapesNew
    else
        break
    end if
end if

Algorithm	Parameters	Fitness	Mutations	Elapsed Time
sa	Temperature: 0.3539967278337753 4	54.91893137254902	100	0.8180000782012 94
sa	Temperature: 5.163216905284117e-16	79.36638562091504	1000	8.353000164031982
sa	Temperature: 3.408207139580972e-90	89.32741176470589	5000	41.78200006484985
sa	Temperature: 8.605284404389222e-190	91.79294444444444	10000	83.48100018501282
sa	Temperature: 4.4e-323	94.28032026143791	50000	425.7650001049042



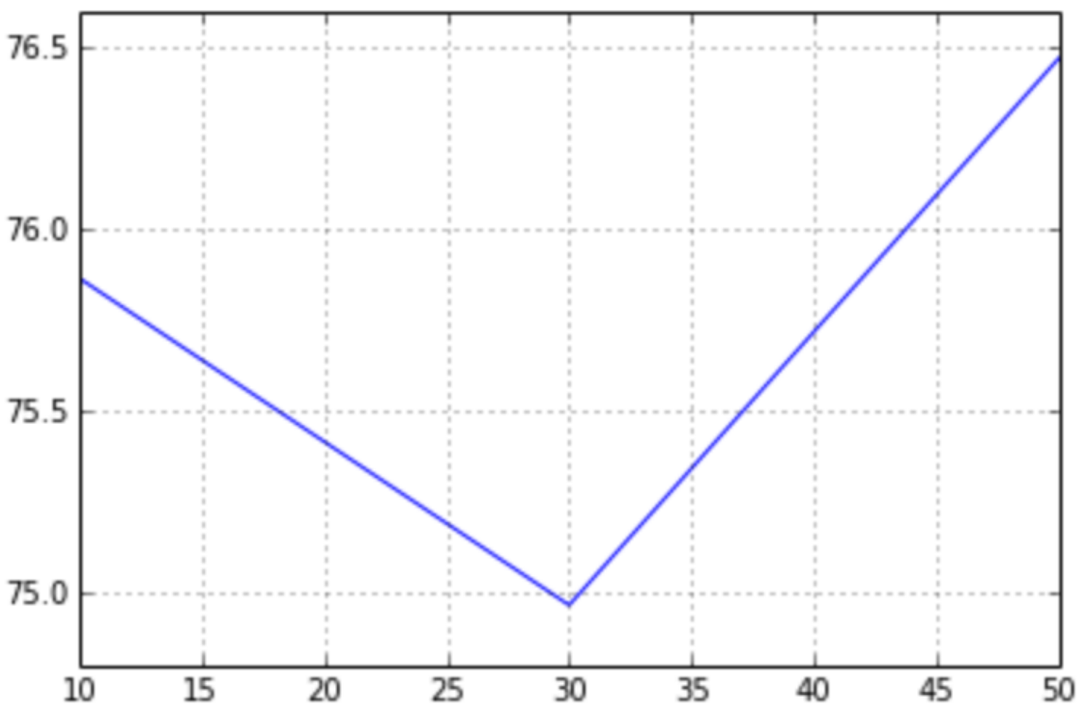
Genetic Algorithm

Selection, Recombination, Mutation

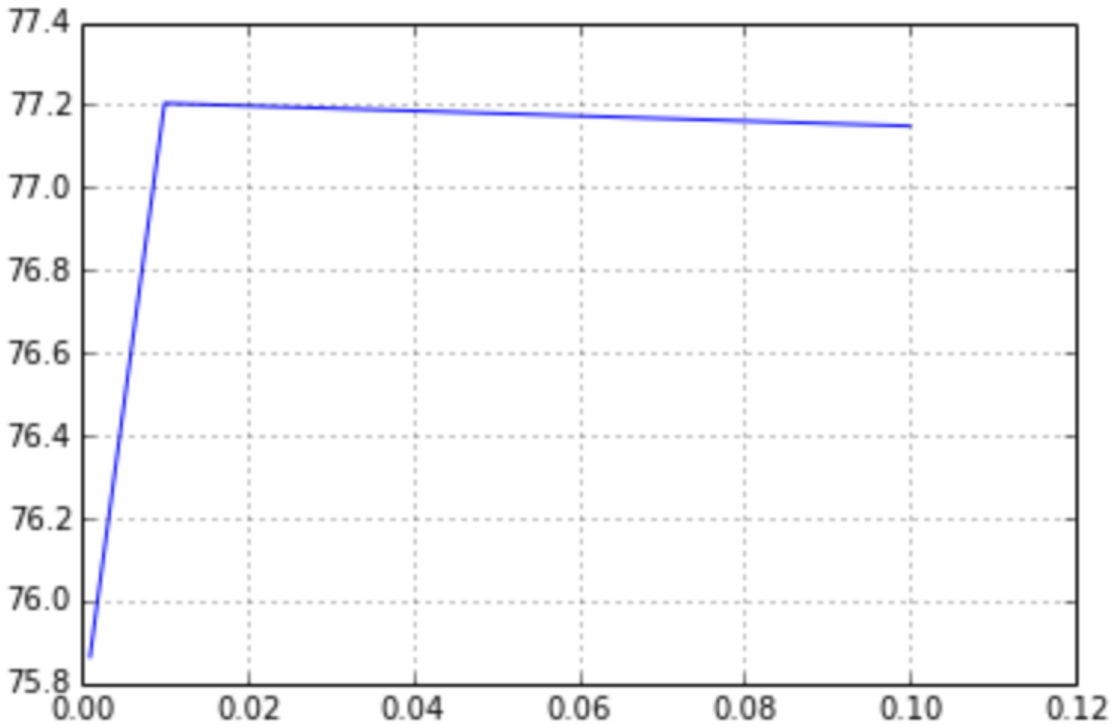
Algorithm 4 Genetic Algorithm

Individuals[population size]← Randomly Initialize DNA of the population
while (!*stop*) **do**
Calculate fitness of the group $fitness[populationsize]$
Calculate probability of being chosen to generate children $probability[i] = \frac{fitness[i]}{sum\_of\_fitness}$
for (i in 0:Population size) **do**
Generate one child through crossover of two chosen parents
end for
Each position in DNA of each individual mutates randomly with p_{mutate}
end while

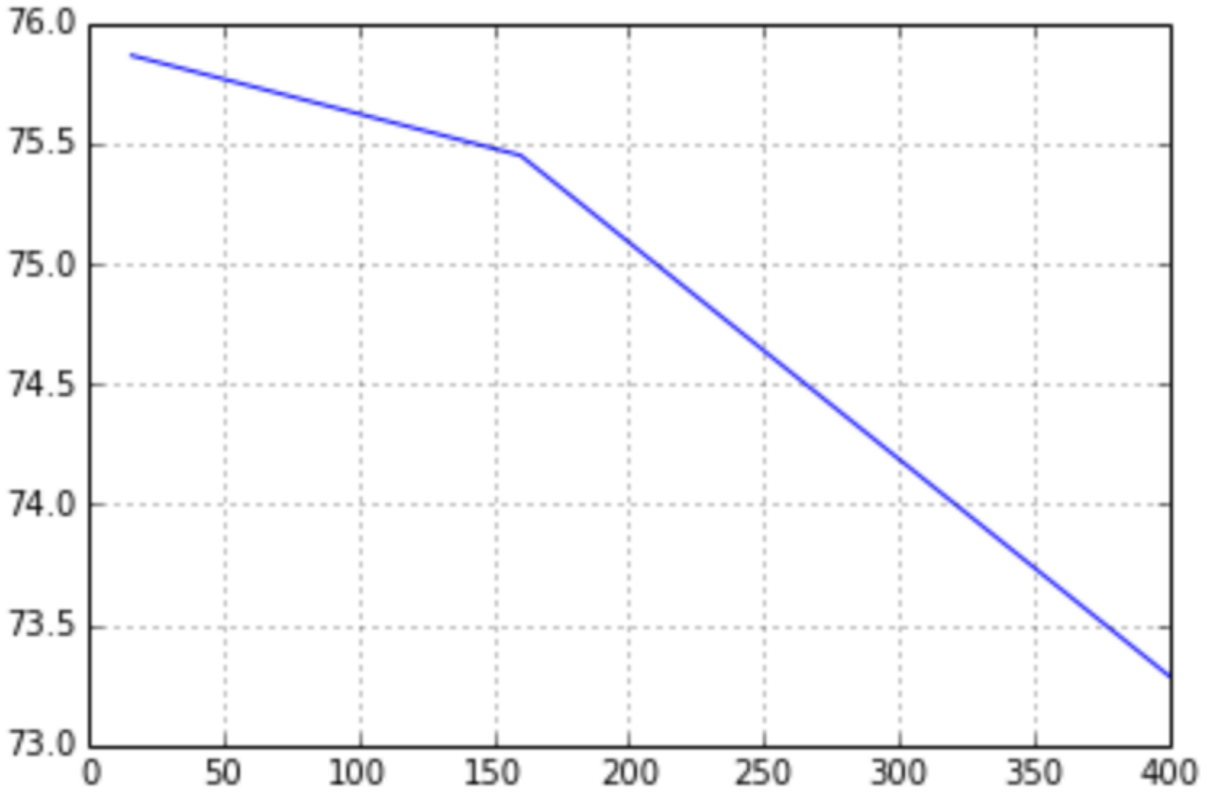
Different Population



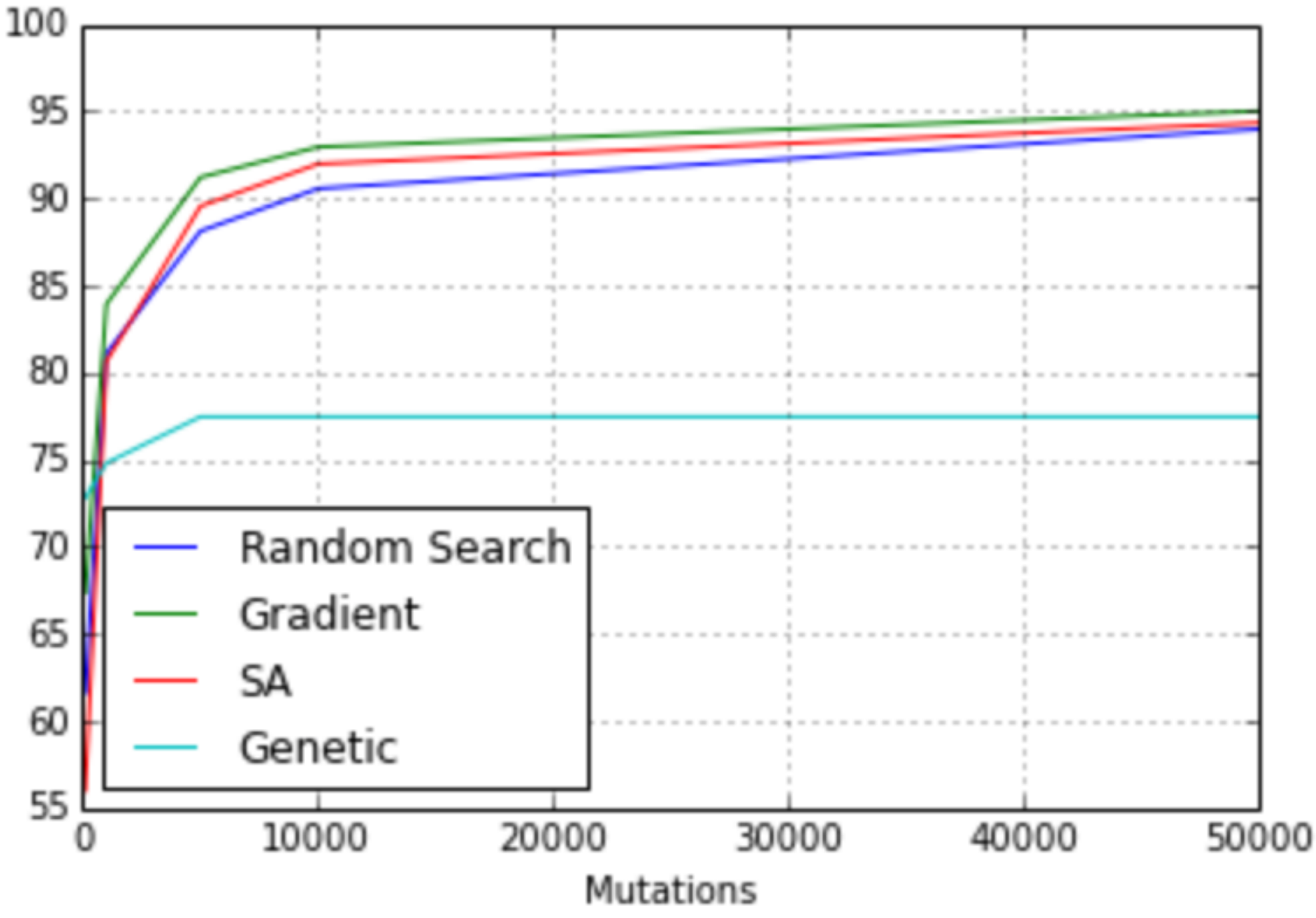
Different Mutate Rate



Different Crossover Length



Compare


100
95
90
85
80
75
70
65
60
55
0
10000
20000
30000
40000
50000
Mutations
Random Search
Gradient
SA
Genetic

Future Work

1. Optimize genetic algorithm
2. Try different error function
3. Apply more algorithms