AIML Report

Evolutionary Computation and Optimization

Ramnath Pillai

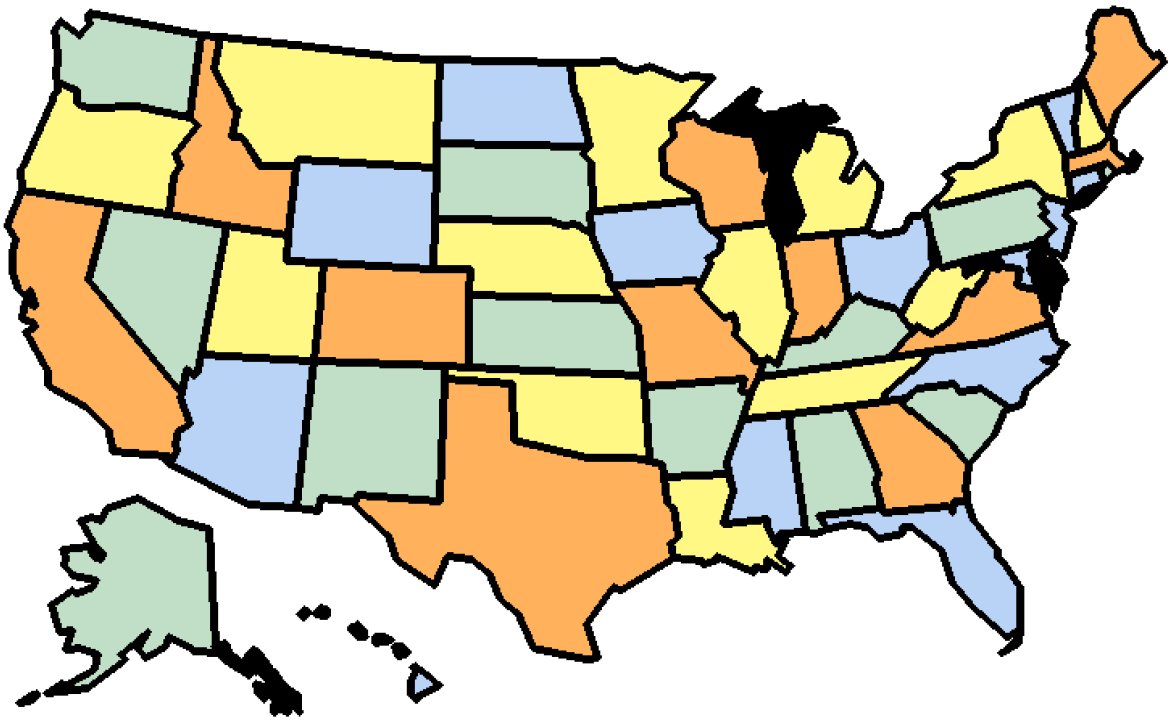
Fall 2015

Assignment 6

Q1. EXTRA CREDIT CODE: Q1/ga_mapcoloring.m

Note: Ideally this code should give an output in less than 200 iterations. The mean number of iterations required to display output is 100.

The equality of colors is enforced as a count of (12,12,13,13) irrespective of its order. This constraint has been enforced by penalizing the best fitness outcomes that have satisfied the adjacency condition but NOT the equality count condition. This penalization is performed in the helper function **checkCondition.m.** Following is a sample output where the color frequencies are 12,12,13,13 as an unordered set.



Q2. REFER TO THE HANDWRITTEN DOCUMENT ATTACHED AT THE END OF THIS ASSIGNMENT

CODE: Q2/Q2.m

Using Quadprog, the optimal value of the function is at [4.9,3.367] and the optimal value is 65.49
Using fmincon, the optimal value of the function is at [4.9,3.367] and the optimal value is 65.49

Q3. a REFER TO THE HANDWRITTEN SHEETS AT THE END OF THIS WORKSHEET

Q3. b FILE: Q3/Q3.xlsx

Some comments on how to run the solver:

- The cost function and constraints are clearly labeled in the Excel File.
- The constraints should ALL be set to <=0.
- One must be careful to choose an initial value of d and t that are within limits
- The already existing values of d and t are the optimal values.
- Optimal value is found to be at **d=1.157** and **t=1 using EXCEL** and the optimal cost is 12.315

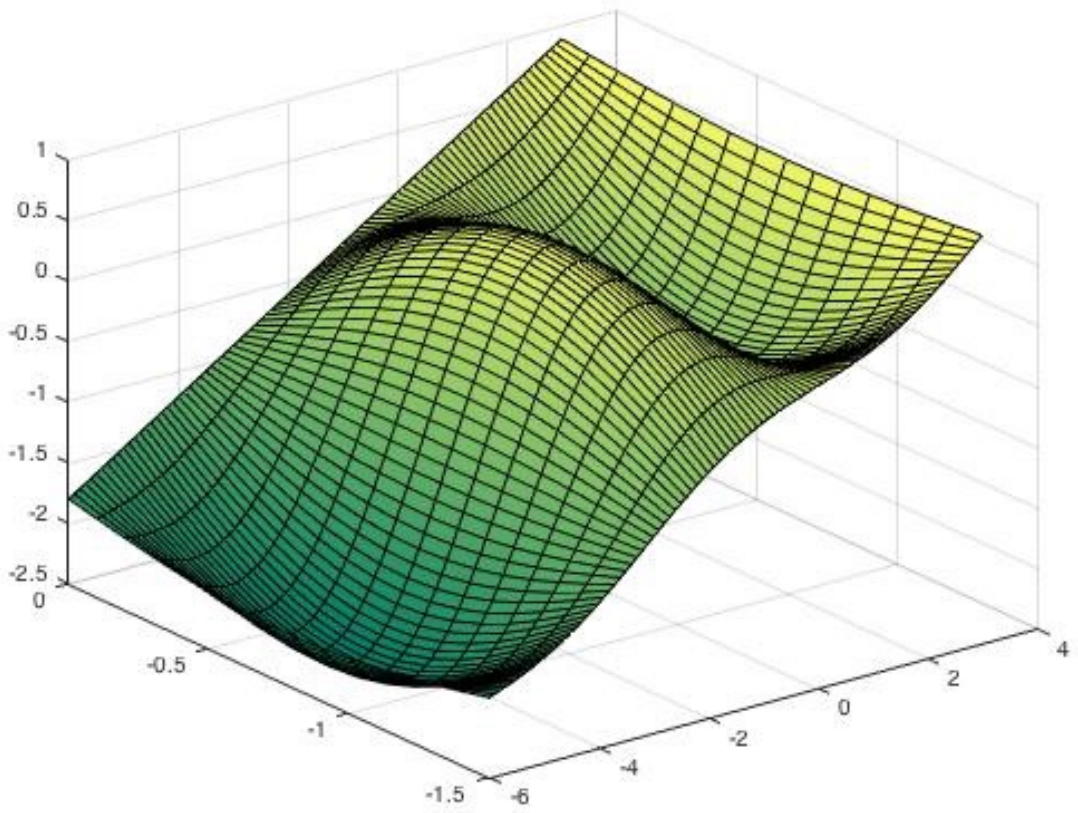
Q3. c The initial value used is [d=2, t=0.6]. Using Active-Set, the optimal value of the function is at [1.157,1] and the optimal value is 12.315

Q3. d Using SQP Algorithm, the optimal value of the function is at [1.157,1] and the optimal value is 12.315.

Q3. e The solutions obtained from Excel and the two different algorithms match. On re running the algorithm with a new set of initial values, [2.5, 0.9], we get the exact same optimal point as the above for all 3 cases. The final solution is consistent among the three algorithms. This is due to the fact that the cost function is

unimodal and therefore, the algorithm does not get caught in different local maxima according as the initial condition is changed.

Q4. a Shown below is the function plot in 3D within the domain mentioned in the question.



Q4. b The stationary points in the given domain is:

- (0,-1.4185) This is a saddle point.
- (-5.0170, -0.7855) This is the Global Minimum and a local minimum
- (1.2660, -0.7855) This is a local minimum
- (-1.2660, -0.7855) This is a local maximum
- (0, -0.1520) This is a saddle point.

The global minimum is computed to be (-5.0170, -0.7855)

Q4. c CODE: Q4.m

CROSSOVER METHOD USED:

Line Crossover: Given 2 parents, a random number alpha is generated between zero and 1 and two new children are generated at ratios alpha:1-alpha and 1-alpha:alpha from either end points, thus having two new children.

MUTATION METHOD USED:

A random number within the desired domain range is assigned to an allele with a 10-sided coin toss probability. i.e. 1/10.

HOW WAS THE GENOME/DATA REPRESENTATION?

- The genotype or the data representation is in the form of a real valued 1x2 Vector where each allele represents a coordinate in 2D space.

HOW WAS INITIAL POPULATION GENERATED?

- Population is a randomly generated array of size 500. Each member of the population is a 1x2 real valued vector.

WHAT WAS THE FITNESS METRIC USED

- Fitness function is the negative of the function value itself. i.e. the lesser value the function takes, the higher the fitness value would be.

DIFFERENT CASES CONSIDERED:

- Crossover, no mutation **CODE: Q4_NOMUTATION.m**. Here the code takes an average of 8 iterations to run. However, with mutation the code takes shorter time to converge to the data. This is due to the fact that mutation results in formation of a more varied set of children that could possibly lead to a faster approach to the global minima. It also has the added advantage of escaping from local minima where it could get caught if mutation is neglected. Thus WITH CROSSOVER WITHOUT MUTATION, HIGHER ITERATIONS, LOWER PERFORMANCE.

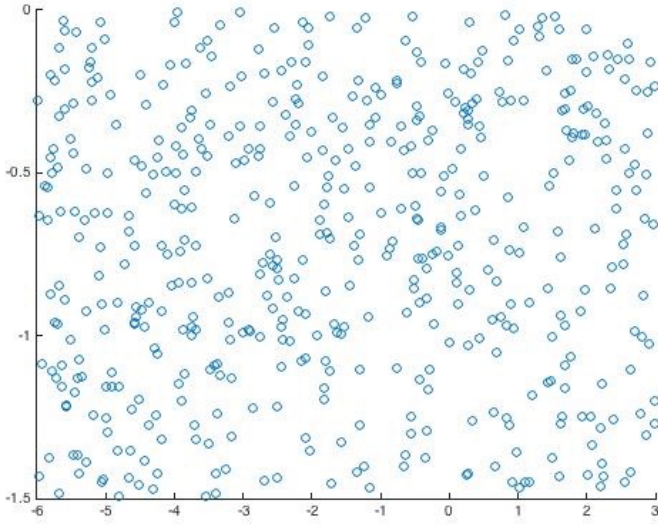
- MUTATION, no crossover: **CODE: Q4_NOCrossover.m**: Here it can be seen that the answer does not converge to the correct minima even after 500 iterations.

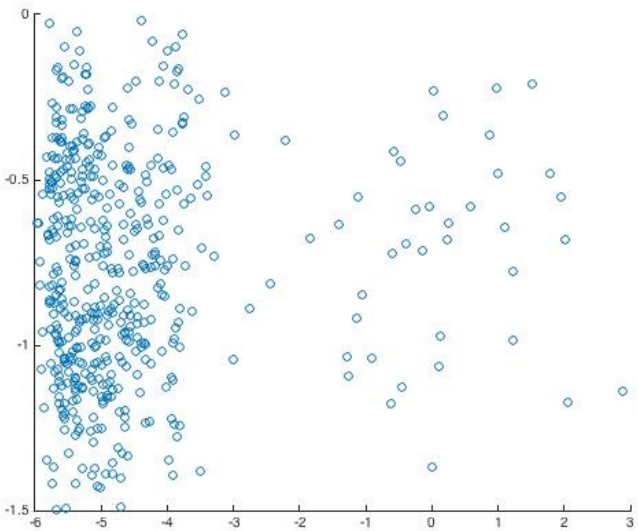
The answer reaches somewhere in the range of the minima but does not actually converge to the desired output. Thus, mutation without crossover is practically obsolete for continuous distributions as this case. THUS LOWER PERFORMANCE, NO CONVERGENCE.

- CROSSOVER AND MUTATION: **CODE: Q4.m** Here the code converges in around 7 to 8 iterations. It can be seen that this is the most efficient case of a Genetic Algorithm. The convergence is accurate as well as the number of iterations is less enough, therefore it is a very efficient strategy to compute the minima of real valued functions without actually calculating the derivative and performing computationally intensive gradient descent.

- NO ELITISM: **CODE: Q4_NOELITE.m** : It can be seen that in a case of no elitism the code takes longer to converge and the performance decreases. This is due to the fact that elitism ensures that the best solutions remain within the set of available solutions while still being available as parents for crossover. This ensures that the best fitness of the next set of solutions is only better or equal in fitness as the current set of solutions. In case of no elitism, the best fitness case is lost during crossover, thus it has to go through a higher number of iterations to attain the global minima and converge to it. No elitism takes an average of 13 iterations whereas 4% elitism gives an average of 7 to 8 iterations.

- PLOTS OF POPULATION AND CONVERGENCE:
CODE: Q4_plotpopulation.m





Following are the 7 iterations its took to converge to the right solution.