CSC 4510
Advanced Topics in Artificial Intelligence (Evolutionary Computation)
Florida Southern College

Assignment 5: Multi-Objective Evolutionary Algorithms
Due: Thursday, April 25, 2019

1. A simple multi-objective problem has two objective functions $f_1(x_1, x_2) = x_1$ and $f_2(x_1, x_2) = x_2^3$, and is subject to the constraint $x_1^2 + x_2^2 \leq 10$.

   (a) Plot the fitness landscape for this problem; you may assume the object variables are all integers. Make sure your axes are clearly labeled and the plot is (at least approximately) drawn to scale.
   (b) Identify the Pareto front, assuming we want to *maximize* both objective functions.

   NOTE: You do not need to do any programming for this problem (the plot can be generated by hand), but you are welcome to do so if you would like.

2. For this problem, you will develop a Python implementation for the fast nondominated sorting and crowding distance algorithms as described in the following paper:

   K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, 2002, pp. 182-197.

   A text file (fitness.txt) is provided that contains the fitness values for multiple data points. For instance, the first line in the text file is '1 15', which means $f_1(\bar{x}_1) = 1$ and $f_2(\bar{x}_1) = 15$. Starter code (nsga2.py) is also provided.

   (a) Write the function (called `ndsort`) to do fast nondominated sorting. Your function must take a fitness array as input and return an array of Pareto front indices as output. You may assume that we want to *minimize* all objectives. In addition, your function should plot the results in fitness space, using the color-coding scheme described below (you may not need all of the colors, but they should be used in this order).

Front	Color
1	blue
2	orange
3	green
4	red
5	purple
6	black

   (b) Write the function (called `crowddist`) to compute crowding distance for every data point. Your function must take a fitness array and a Pareto front array as input and return an array of distances as output. Remember that crowding distance is computed for each Pareto front separately from the others! The code is already setup to save the distances to file (distance.txt), so feel free to check the result after you successfully run the code.