CMSC 173 Genetic Algorithm

OBJECTIVE : To be able to revise the Python source of Genetic Algorithm to use a different recombination operator and conduct comparison of results in the class example.
OUTPUT : Revised Python source Code and Short Write up
TYPE OF PROJECT: Individual or Pair ONLY

PROGRAM SPECIFICATIONS

1. Use the PSO python code in the lecture videos "11 - Evolutionary Process Genetic Algorithm Code Part 1 to 13 - Evolutionary Process Genetic Algorithm Code Part 3"
2. Do the following changes in the source code:

Code Modifications:
Code Modifications: 1. Use a single point crossover for the recombination operator. That is, randomly select a cross-over point and then combine the 1st half (2nd half) of parent1 and 2nd half (1st half) of parent2 to generate offsprings. 2. Use the same mutation operator that was presented in the Part 3 of the GA lecture video. 3. Using the Python time module (https://www.programiz.com/python-programming/time), record the time (you can use *super_best_time* at which the best solution was found first. Hint: You should add a timestamp at the "compare with overall best" part of the source code ***aside*** from updating the *super_best_fitness* and *super_best_solution*. Example: If a new best is found at generation 5, then timestamp will be recorded at *super_best_time* = 10s. Then if at generations 6 to 30, no solution was better than the *super_best_solution* from gen 5, then *super_best_time* will still remain at 10s. No need for print outs from gen 6 to 30. Furthermore, if at gen 31 a solution is better than the *super_best_solution*, then timestamp will be activated again and becomes *super_best_time = 100s.* Continuing, if until the last of the generation, example generation 100, none of the new solutions are better than the super_best_solution found at gen 31, then the last recorded *super_best_time* is still *100s.* 4. Everytime the *super_best_time* is updated, please print its value and the generation at which it was recorded*.* See example output below: …. Time and generation at which best was found: 10s at Gen 5 Time and generation at which best was found: 100s at Gen 31 …. 5. There is NO NEED for you to print the initial solutions, the strength, the wheel ranges. There is also no need to print the mutated solutions after the application of the mutation operator. 6. Note that you still need to retain the "all_fitness" list and graph/plot that list according to how it was done in the original source code. 7. For all printouts not mentioned in number 5 above, you still need to print all of them (i.e. print the crossover/recom part, etc). Note that there are additional prints on screen as mentioned in number 4 above.

Mutation Rate: 0.80
Number of generations: {100, 500}
Test for N: {50, 100}

1. Since there are two values for generations and two values for N, then you will have 4 sub-experiments in this section. That is, the following combinations {gen,N}: {100,50}, {100,100}, {500,50} and {500,100} 2. Do 1 run for ***each*** {gen,N} combination. 3. Screenshot the ff below. Please put proper sections to different one combination from another for easy marking. Save these in a pdf file. a. output of #4 (regarding time and generation) for ***each*** {gen,N} combination b. graph of all_fitness for ***each*** {gen,N} combination c. Provide an analysis on the similarities/differences/performances of all the {gen,N} combinations. You may discuss/analyse freely but make sure to do comparisons between and

amongst the different gen and N values in terms of (1) fitness values AND (2) time at which the best solution was first found.

4. Submit the revised python code and name it <familyname_ga>.py (e.g. gamot_ga.py)
5. Zip the pdf file and the python file and name it <familyname_ga>.zip (e.g. gamot_ga_zip). Upload it in the LMS assignment bin