CS 5320/4320

Project #5: Jaya   Points 100   Due: April 30

The undergraduate section will do Part A only. The graduate section will do both the parts. Submission and group instructions are as before.

In this project you will apply the Jaya algorithm for solving the problem from Project #1.

A. [For both UG and G] Implement the Jaya version described in the paper by Rao. Report (a) the best-of-run fitness and the corresponding solution vector from each of 30 independent runs of the algorithm, (b) average of those 30 best-of-run fitnesses and (c) standard deviation of those 30 best-of-run fitnesses.

B. [For G only] Code and execute a <u>new</u> version of Jaya that updates the best (and the worst) as soon as a new individual replaces the current individual (instead of waiting until an entire generation is complete). Graduate students will report the above-mentioned items (a) through (c) – once for the original Jaya (in Part A) and again for the new Jaya (in part B). Conclude whether the original Jaya performs better/worse (using the fitness metrics in (a) through (c)) than the new version.

   In addition, report the following for your <u>new</u> Jaya. Show in a tabular form the total number of updates of the best and, separately, of the worst per generation (note: one generation involves testing exactly N individuals for possible update, where N is the population size) at every generation (if your total number of generations in a run is small) or at suitable intervals of generations (e.g., once every 10 or 20 or 50 or 100 generations). Also report the average (per generation) number of updates of the best and, separately, of the worst in a single run for each of the 30 runs (to compute the average, you will need to process the update counts from EVERY SINGLE generation of a run, not at intervals of 10 or 100 generations). Example output on these update counts:

```
New Jaya
Run#    Gen#    #Best update    # Worst Update
1         1          5                1
          2          3                0
          …….
        100          2                1
          ---------------------------------------------
          Run #1   Average of #Best = (5 + 3 + … + 2)/100 and average of #Worst = (1 + 0 + … + 1)/100

2         1          1                0
          2          2                1
          …….
        100          4                2
          ---------------------------------------------
           Run #2   Average of #Best = (1 + 2 + … + 4)/100 and average of #Worst = (0 + 1 + … + 2)/100
```

   Finally, print the average (over 30 runs) generation-wise count of updates of the best and of the worst.