Project #1 - Genetic Algorithm

Evolutionary Computation
Spring 2014
Due Feb. 21st

The goal of this project is to write a genetic algorithm (GA) for a series of benchmark optimization problems. In each case the problem is to optimize, i.e. find the (global) **minimum**, of a real valued function.

To test the GA we'll use 6 starndard, benchmark, real-valued functions:

1. Spherical
2. Rosenbrock
3. Rastrigin
4. Schwefel
5. Ackley
6. Griewangk

Each of these functions is defined at: http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume24/ortizboyer05a-html/node6.html#tabla:DefFunc . (Note the first function labeled as Schwefel on this page is actually the double sum, which we are not using. We are using the Schwefel function defined immediately after the Rastigin function.)

Pay careful attention to the ranges of the functions. You will want to use those ranges both in creating intial individuals and in controlling the generation of neighbors, e.g. you don't want your GA 'wondering' out of the search space. Note that here the functions are all defined with 30 dimensions, e.g. P = 30 in the function definitions.

Project Requirements:

- Write a GA to find the input values (x_1,...,x_{30}) that minimizes each of the six benchmark problems.
- You need to pick the details of the GA, including:
  - Representation
  - Fitness function
  - Algorithm type: Steady state or generational
  - Crossover type: 1-point, 2-point, uniform, arithmetic, etc.
  - Mutation rate
  - etc.

Project Write-up:

- You must write a short paper describing the results of your project that includes the following sections:
- **Abstract** - a short (~200 words) summary of what you did and what the results were.

- **Algorithm descriptions** - clear, complete descriptions of your GA. Be careful to include all of the details someone would need to replicate your work. Examples of necessary details include (there are others):
  - How fitness is measured
  - Exactly how initial random solutions are generated
  - Mutation rates
  - etc.

  Basically everytime you make a decision about how the algorithm works (what type of crossover it will use, how mutaiton is performed, etc.) you should make a note of it.
- **Results** - you should include graphs and/or tables to make it easy to understand the results. Make sure that the graphs and table are clearly labeled.
- **Conclusions** - based on your results draw some specific conclusions about how well the algorithm performed.

- **(CS572 Students Only) Population size versus number of generations/iterations** - For a fixed number of evaluations test whether and, if so, how dividing these evaluations between population size and interations affects the results. For example, given a limit of 5000 evaluations a generational GA could use a population size of 100 for 50 generations, or a population size of 200 for 25 generations, or any other combination whose product is 5000. Determine whether the population size/iterations division matters for these problems.