ISE 5113 Advanced Analytics and Metaheuristics
Homework #6

Instructor: Charles Nicholson

Due: See course website for due date

Requirement details

1. Homeworks are to be completed in teams of one, two, or three. You must set up your team in Canvas correctly as an official group. If team members disagree on an answer, you can record solutions corresponding to each member (please clearly mark which solution belongs to which team member).
2. Your primary submission will be a single Word or PDF document and must be of professional quality: clean, clear, concise, yet thoroughly responsive.
3. Any code (e.g., Python) must also be submitted separately. Your code MUST be well-documented. Failure to submit the files will result in a penalty.
4. You cannot use preexisting Python packages for heuristics or metaheuristics. In fact, other than `numpy, copy, random, itertools`, and other basic utilities, you should seek specific permission from the instructor if you have any doubt. That is, *you* are responsible for creating the logic yourself.

You will develop Python code to implement a Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) algorithm for the Schwefel minimization problem.

The Schwefel problem is a generalizable benchmark problem than can be formulated in $n > 0$ dimensions.

Figure 1 presents the formula and a 2D representation of the solution landscape. The feasible region is an n-dimensional hybercube centered at the origin with possible values ranging from -500 to 500 for each dimension. The optimal objective and solution is known.

$$f(x) = 418.982887272433n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$$

Dimensions:	n
Domain:	$-500.0 \leq x_i \leq 500.0$
Global Optimum:	$f(x) \approx 0.0$ at $x = (420.9687, 420.9687, \ldots, 420.9687)$

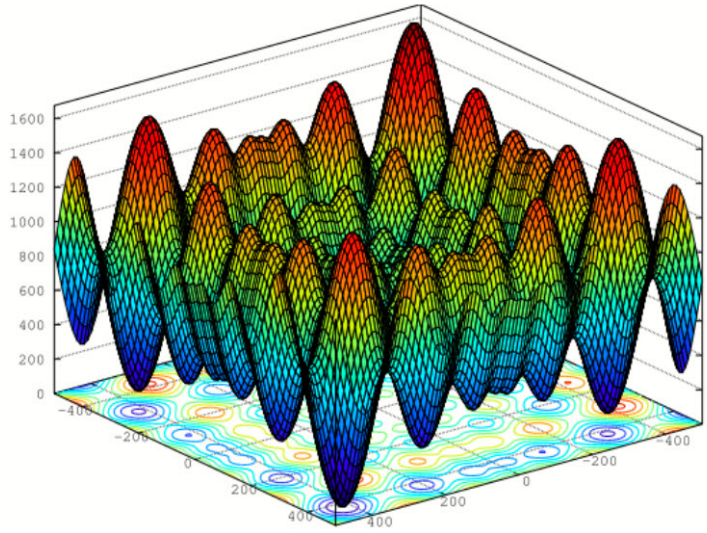

Figure 1: Schwefel function in 2D

Question 1: Genetic Algorithm Implementation (40 points)

Some basic Python code is available to help you begin complete a GA implementation for the Schwefel minimization problem. The code however is incomplete and requires multiple improvements.

(a) Finalize the code.
   i. Create code to generate chromosomes in the initial population.
   ii. Create code to mutate chromosomes.
   iii. Implement logic for crossover rate and mutation rate.
   iv. Implement some type of elitism in the insertion step.
   v. Complete/modify any other logic as you see fit.

(b) Empirically decide on parameters for population size, stopping criterion, cross over rate, mutation rate, selection, and elitism.

(c) Solve the 2D Schwefel.
   i. Create a small population of 8 chromosomes and depict their locations on a graph for the initial random set and the first generation.
   ii. Solve the problem as best as possible and provide information on the quality of the best solution and the performance of the approach overall.

(d) Solve the 200D Schwefel problem as best as possible. Provide information on the quality of the solution and the performance of the approach overall.

Question 2: Particle Swarm Optimization (60 points)

Some basic Python code is available to help you begin a PSO implementation for the Schwefel minimization problem.

(a) Complete the original PSO implementation based on "particle best" and "global best" or constriction is necessary. Make sure to include the following elements:

- limits on particle position (e.g., particles should primarily remain within the feasible region)
- limits on particle velocity
- one or more stopping criteria

Note: inertial weights and/or constriction factors are not required.

(b) Using your code from Part (a), create a swarm of size 5 and solve the 2D Schwefel problem.

i. Record and list in a table the first 3 positions and velocities of each particle.

ii. Determine and highlight the particle that represents the global best particle position in each of these iterations.

iii. Plot the first 3 positions of each of the 5 particles (you can use Python, R, Excel, or even draw it by hand)

(c) Implement a PSO algorithm that uses the "local best" in place of the global best. You may also add inertia weighting or constriction if you like. Implement one of the following neighborhood topologies: ring, star, von Neumann:

(d) Solve the 2D and 200D Schwefel problem as best as possible. Provide information on the quality of the solution and the performance of the approach overall. Compare the results with the Genetic Algorithm results.