# **Evolutionary Optimization**

Imperialist Competitive Algorithm

(Language: Python)

About the algorithm:

The algorithm remotely resembles PSO. Instead of having a single **g\_best** here you can have n number of g\_best (called imperialists) and every imperialist has their own set of particles (called colonies). The colonies are perturbed in by moving towards their imperialist and then updating the particles accordingly.

Implementation:

About each function:

Init\_pop : Initialises the initial population

Func1 : It calculates the value of the objective function

Ranking: It finds out the imperialist among the population

N\_costs: It sets up the probability for giving colonies to their respective imperialist

Col\_div: It divides the colonies among the imperialist using roulette wheel type selection

Update\_emp: It perturbs the colonies according to their respective imperialists

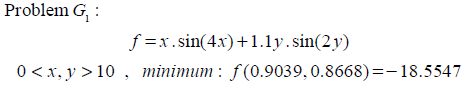
Cost\_emp: It calculated the cost of each empire (colonies+imperialist)

Pro\_emp: It sets up the probability for moving weakest colonies to stronger imperialist.

Improvements:

As the paper suggests that randomness should be introduced into the search space by shifting the direction of movement of points by some angle. However meaning of angle becomes abstract when we talk of optimization problems involving more than two variables. Hence instead of perturbing the direction vector by an angle we can introduce randomness in each of the variables separately.

Results:



Code’s Result:

