Minimum of Rastrigin Function by Particles Swarm Optimization (DEAP)

From Wikipedia, the free encyclopedia (https://en.wikipedia.org/wiki/Rastrigin_function)

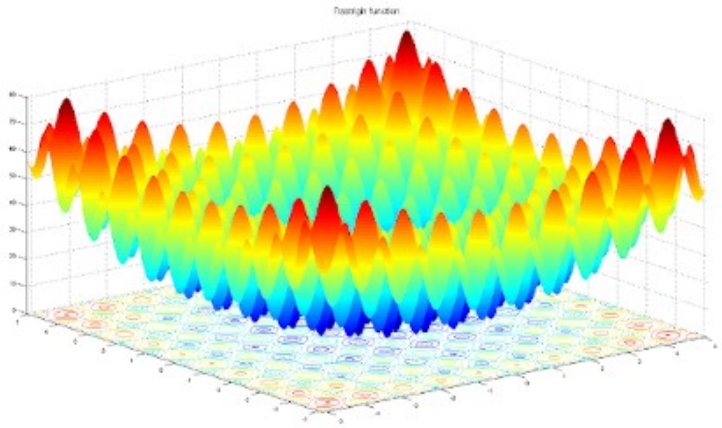
In mathematical optimization, the **Rastrigin function** is a non-convex function used as a performance test problem for optimization algorithms. It is a typical example of non-linear multimodal function. It was first proposed by Rastrigin as a 2-dimensional function and has been generalized by Mühlenbein et al. Finding the minimum of this function is a fairly difficult problem due to its large search space and its large number of local minima.

On an n-dimensional domain it is defined by:

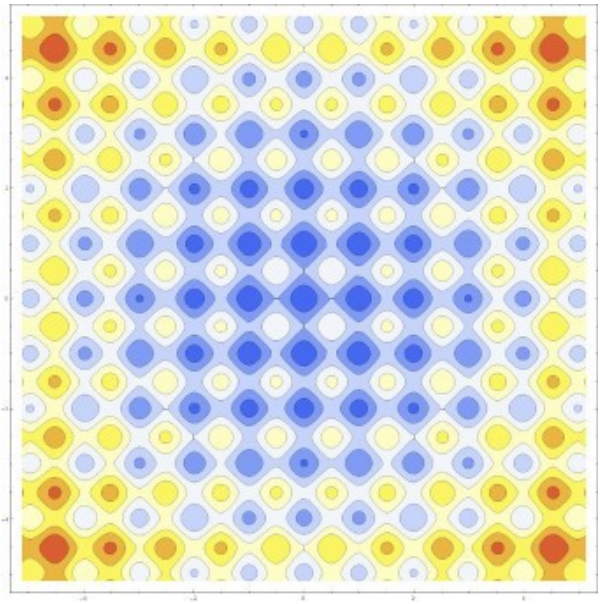
$$f(\mathbf{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos(2\pi x_i)\right]$$

where A = 10 and x(i) ∈ [**− 5.12, 5.12**]. It has a global minimum at **x = 0** where **f(x) = 0**

Rastrigin function of two variables in 3D



Rastrigin function Contour



In this brief notebook we are addressing the **Rastrigin function of two variables minimum problem** by a **Particles Swarm Optimization** implementation using the python library **DEAP (Distributed Evolutionary Algorithms in Python)** (https://github.com/DEAP) with a Swarm of 100 Particles and 1000 Generations.

In [1]:
```
# Libraries import
import operator
import random
import numpy as np
import time

from deap import base,creator,tools
```

In [2]:
```
# Function definition
c = np.zeros((1,1))
def Rastrigin(x):
    fitness = 10*len(x)
    for i in range(len(x)):
        fitness += x[i]**2-(10*np.cos(2*np.pi*x[i]))
    c[0] = fitness
    return c
```

In [3]:
```
# Fitness and Particles
creator.create("FitnessMin",base.Fitness, weights=(-1.0,))
creator.create("Particle",list,fitness = creator.FitnessMin,speed= list,smin=None,s
max= None,best=None)
```

In [4]:
```
# Definition of Particles generation function
def generate(size,pmin,pmax,smin,smax):
    part = creator.Particle(random.uniform(pmin,pmax) for _ in range(size))
    part.speed = [random.uniform(smin,smax) for _ in range(size)]
    part.smin = smin
    part.smax = smax
    return part
```

In [5]:
```
# Definition of Particles update function
def updateParticle(part,best,phi1,phi2):
    u1 = (random.uniform(0,phi1) for _ in range(len(part)))
    u2 = (random.uniform(0,phi2) for _ in range(len(part)))

    v_u1 = map(operator.mul, u1,map(operator.sub, part.best, part))
    v_u2 = map(operator.mul, u2, map(operator.sub,best, part))
    part.speed = list(map(operator.add,part.speed, map(operator.add, v_u1,v_u2)))
    for i, speed in enumerate (part.speed):
        if speed< part.smin:
            part.speed[i]=part.smin
        elif speed > part.smax:
            part.speed[i] = part.smax
    part[:] = list(map(operator.add,part,part.speed))
```

In [6]:
```
# Toolbox and its objects registration
toolbox = base.Toolbox()
toolbox.register("particle",generate, size =2 , pmin = -6 , pmax = 6, smin =-3 , sm
ax =3)
toolbox.register("population",tools.initRepeat,list, toolbox.particle)
toolbox.register("update", updateParticle, phi1 = 2.0 ,phi2=2.0)
toolbox.register("evaluate", Rastrigin)
```

In [7]:
```
# Main function
def main():

    pop = toolbox.population(n=100)
    logbook = tools.Logbook()
    logbook.header = ["gen", "evals"]

    GEN = 1000 #Number of Particles Generations
    best = None

    start_t=time.time()
    for g in range(GEN):
        for part in pop:
            part.fitness.values = toolbox.evaluate(part)
            if not part.best or part.best.fitness < part.fitness:
                part.best = creator.Particle(part)
                part.best.fitness.values = part.fitness.values
            if not best or best.fitness <part.fitness:
                best = creator.Particle(part)
                best.fitness.values = part.fitness.values
        for part in pop:
            toolbox.update(part,best)
        stats = tools.Statistics(lambda ind:ind.fitness.values)
        stats.register("Min", np.min)

        logbook.record(gen= g, evals = len(pop), **stats.compile(pop))
        #Minimum Research Loop Visualization
        #print("Generation e Particles Number =", logbook.stream)
        #print("X, Y = %s , Rastrigin Minimum %s" %(best, best.fitness))

    end_t=time.time()

    print("\n----------------------------------------")
    print("Rastrigin Function Minimum Determination")
    print("----------------------------------------\n")
    print("[X,Y] = ",best)
    print("Best Fitness (Rastrigin Minimum) = ", str(best.fitness)[7:-3])
    print("Elapsed Time (seconds):", end_t-start_t)

    return pop , logbook,best, stats
```

In [8]:
```
# Code execution
if __name__ == "__main__":
    main()
```

```
----------------------------------------
Rastrigin Function Minimum Determination
----------------------------------------

[X,Y] =  [-0.0027465187213622233, 0.0016474240374112492]
Best Fitness (Rastrigin Minimum) =  [0.00203494]
Elapsed Time (seconds): 3.296771764755249
```

In [ ]: