**Firefly-based Channel Selection**

import numpy as np

from fireflyalgorithm import FireflyAlgorithm

import numpy as np

from numpy.random import default\_rng

class FireflyAlgorithm:

def \_\_init\_\_(self, pop\_size=20, alpha=1.0, betamin=1.0, gamma=0.01, seed=None):

self.pop\_size = pop\_size

self.alpha = alpha

self.betamin = betamin

self.gamma = gamma

self.rng = default\_rng(seed)

def run(self, function, dim, lb, ub, max\_evals):

fireflies = self.rng.uniform(lb, ub, (self.pop\_size, dim))

intensity = np.apply\_along\_axis(function, 1, fireflies)

best = np.min(intensity)

evaluations = self.pop\_size

new\_alpha = self.alpha

search\_range = ub - lb

while evaluations <= max\_evals:

new\_alpha \*= 0.97

for i in range(self.pop\_size):

for j in range(self.pop\_size):

if intensity[i] >= intensity[j]:

r = np.sum(np.square(fireflies[i] - fireflies[j]), axis=-1)

beta = self.betamin \* np.exp(-self.gamma \* r)

steps = new\_alpha \* (self.rng.random(dim) - 0.5) \* search\_range

fireflies[i] += beta \* (fireflies[j] - fireflies[i]) + steps

fireflies[i] = np.clip(fireflies[i], lb, ub)

intensity[i] = function(fireflies[i])

evaluations += 1

best = min(intensity[i], best)

return best

FA = FireflyAlgorithm()

best = FA.run(function=sphere, dim=10, lb=-5, ub=5, max\_evals=10000)

print(best