Lab - 5

classmate
Date
Page

Grey Wolf

Algorithm:-

Initialize the population of wolves (positions) randomly within the search space
Define the max no of iterations (T)
and population size (N)
Define the fitness function to evaluate solutions

Evaluate the fitness of each wolf in the population
Identify the alpha (best solution), beta (second best) and delta (third best) wolves

For t = 1 to T:
  For each wolf i in the population:
    For each dimension d:
    A1 = 2 * a * rand() - a
    C1 = 2 * rand()
    D_alpha = [C1 * X_alpha[d] - X_i[d]
    X2 = X_beta[d] - A2 * D_beta
    A3 = 2 * a * rand() - a
    C3 = 2 * rand()
    D_delta = C3 * X_delta[d] - X_1[d]
    X3 = X_delta[d] - A3 * D_delta
    X_i[d] = (X_i + X2 + X3) / 3
  end for
  end for
  a = 2 - (2 * t / T)
Update alpha, beta and delta wolves based on fitness

Program :-

```
import numpy as np
def grey_wolf_optimizer(fitness_function, num_wolves,
num_dimensions, max_iterations, bounds):
    lower_bound, upper_bound = bounds
    wolves = np.random.uniform(lower_bound, upper_bound,
        (num_wolves, num_dimensions))
    alpha, beta, delta = None, None, None

    fitness = np.array([fitness_function(wolf) for wolf in
            wolves])
    sorted_indices = fitness.argsort()
    alpha, beta, delta = wolves[sorted_indices[:3]]

    a = 2
    for iteration in range(max_iterations):
        for i in range(num_wolves):
            for d in range(num_dimensions):
                r1, r2 = np.random.rand(), np.random.rand()
                A1 = 2*a*r1 - a
                C1 = 2*r2
                D_alpha = abs(C1*alpha[d] - wolves[i,d])
                X1 = alpha[d] - A1*D_alpha

                r1, r2 = np.random.rand(), np.random.rand()
                A2 = 2*a*r1 - a
                C2 = 2*r2
                D_beta = abs(C2*beta[d] - wolves[i,d])
                X2 = beta[d] - A2*D_beta
```

```
        r1, r2 = np.random.rand(), np.random.rand()
        A3 = 2 * a * r1 - a
        C3 = 2 * r2
        D_delta = abs(C3 * delta[d] - wolves[i, d])
        X3 = delta[d] - A3 * D_delta
        wolves[i, d] = (X1 + X2 + X3) / 3
    wolves[i] = np.clip(wolves[i], lower_bound, upper_bound)
fitness = np.array([fitness_function(wolf) for wolf in wolves])
sorted_indices = fitness.argsort()
alpha, beta, delta = wolves[sorted_indices[:3]]
a = 2 - (2 * iteration / max_iterations)

best_solution = alpha
best_fitness = fitness_function(alpha)
return best_solution, best_fitness

if __name__ == "__main__":
    def fitness_function(x):
        return np.sum(x**2)
    num_wolves = 30
    num_dimensions = 5
    max_iterations = 100
    bounds = (-10, 10)
    best_solution, best_fitness =
grey_wolf_optimizer(fitness_function, num_wolves,
num_dimensions, max_iterations, bounds)
    print("Best solution:", best_solution)
    print("Best fitness:", best_fitness)
```

Output: Best solution: [$-7.85e^{-12}$ $-7.36e^{-12}$, $-8.765e^{-12}$, $8.354e^{-12}$ $8.366e^{-12}$]

Best fitness = $3.3249e^{-22}$