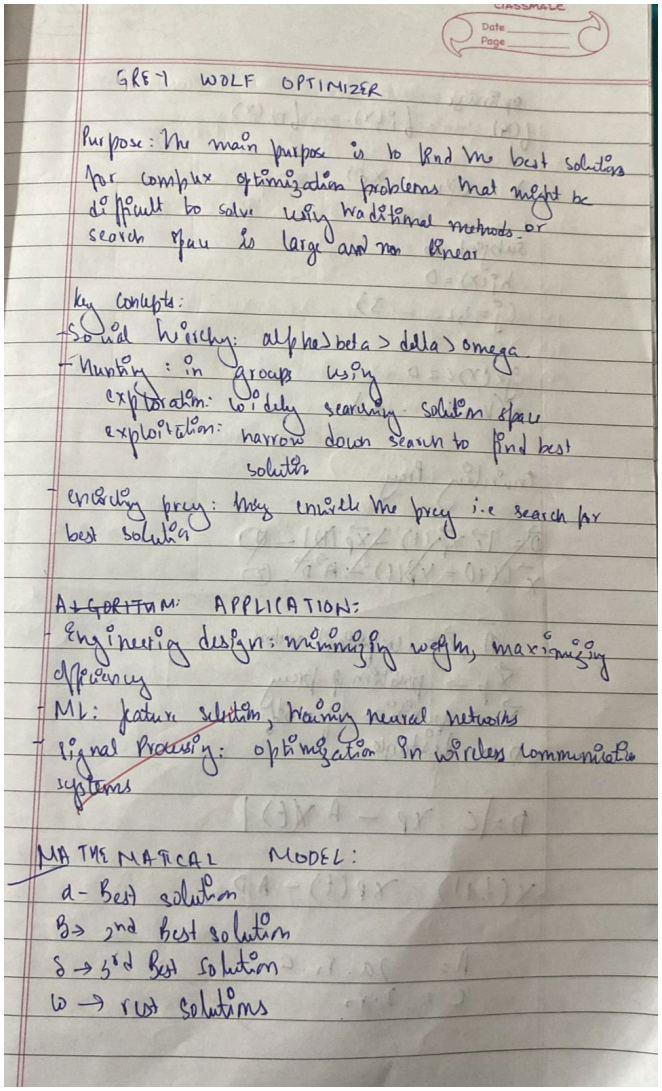
Program 4

Problem statement:Grey Wolf Optimizer (GWO)

Algorithm:



GREY WOLF OPTIMIZER

Purpose: The main purpose is to find the best solutions for complex optimization problems that might be difficult to solve using traditional methods or search space is large and non linear

Key concepts:

- Social hierarchy: alpha > beta > delta > omega
- Hunting: in groups using
  - exploration: widely searching solution space
  - exploitation: narrow down search to find best solution
- encircling prey: they encircle the prey i.e search for best solution

~~ALGORITHM~~ APPLICATION:

- Engineering design: minimizing weight, maximizing efficiency
- ML: feature selection, training neural networks
- Signal Processing: optimization in wireless communication systems

MATHEMATICAL MODEL:

α - Best solution

β → 2nd Best solution

δ → 3rd Best solution

ω → rest solutions

optimize

$f_1(x) \ldots f_i(x) \ldots f_N(x)$

$x = (x_1 \ldots x_d)$

subject to

$h_j(x) = 0$

$(j = 1, 2 \ldots J)$

$g_k(x) <= 0$

$(k = 1, \ldots k)$

Encircling Prey

$\vec{D} = |\vec{C} \cdot \vec{X_p}(t) - \vec{X}(t)|$ — (1)

$\vec{X}(t+1) = \vec{X_p}(t) - \vec{A} \cdot \vec{D}$ — (2)

$\vec{A}, \vec{C}$ = coefficient vectors

$\vec{X_p}$ → position of prey

$\vec{X}$ → position of wolf

D → distance.

$D = |C \cdot X_p - A X(t)|$

$X(t+1) = X_p(t) - A D$

$A = 2a \cdot r_1 \cdot a$

$C = 2 \cdot r_2$

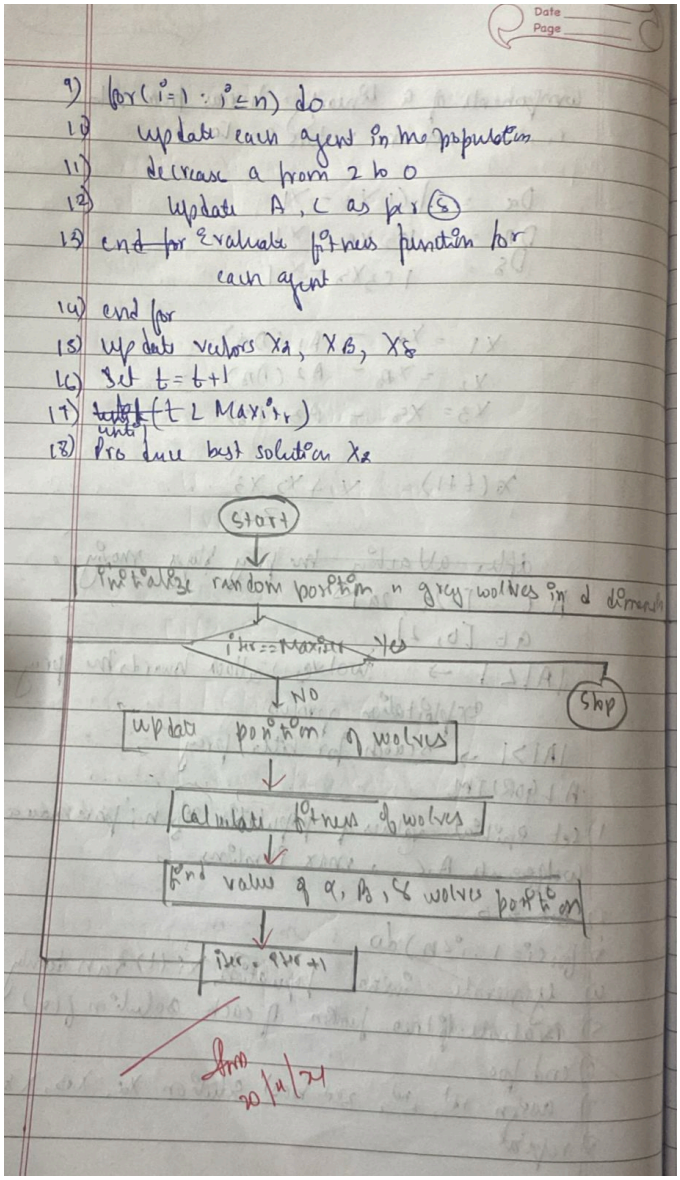
components of a linearly decreases from 2 to 0
r_1, r_2 random vectors in $[0,1]$

$D_\alpha = |C_1 X_\alpha - X|$
$D_\beta = |C_2 X_\beta - X|$
$D_\delta = |C_3 X_\delta - X|$

$X_1 = X_\alpha - A_1 (D_\alpha)$
$X_2 = X_\beta - A_2 (D_\beta)$
$X_3 = X_\delta - A_3 (D_\delta)$

$X(t+1) = \frac{X_1 + X_2 + X_3}{3}$

~~after attacking~~ the prey stops moving
$A \in [-2a, 2a]$
$a \in [0, 2]$
$|A| < 1 \rightarrow$ wolves attack towards the prey
exploitation process
$|A| > 1 \rightarrow$ search for better prey
ALGORITM
1) set initial values of population size n, parameters, coefficients A, C, max iterations
2) set $t = 0$
3) for ($i = 1 : i \leq n$) do
4) Generate initial population $X_i(t)$ randomly
5) Evaluate fitness function of each solution $f(x)$
6) end for.
7) assign 1st, 2nd, 3rd best solution $X_\alpha, X_\beta, X_\delta$
8) repeat



Code:

```
import numpy as np
def objective_function(x):
    return np.sum(x**2)
class GreyWolfOptimizer:
    def __init__(self, objective_function, dim, num_wolves, max_iter, lb=-5, ub=5):
        self.objective_function = objective_function
        self.dim = dim
        self.num_wolves = num_wolves
        self.max_iter = max_iter
        self.lb = lb
        self.ub = ub
        self.positions = np.random.uniform(self.lb, self.ub, (self.num_wolves, self.dim))
        self.alpha_position = np.zeros(self.dim)
```

```
        self.beta_position = np.zeros(self.dim)
        self.delta_position = np.zeros(self.dim)
        self.alpha_score = float("inf")
        self.beta_score = float("inf")
        self.delta_score = float("inf")
    def update_positions(self, a):
        for i in range(self.num_wolves):
            A1 = 2 * a * np.random.rand(self.dim) - a
            C1 = 2 * np.random.rand(self.dim)
            D_alpha = abs(C1 * self.alpha_position - self.positions[i])
            X1 = self.alpha_position - A1 * D_alpha
            A2 = 2 * a * np.random.rand(self.dim) - a
            C2 = 2 * np.random.rand(self.dim)
            D_beta = abs(C2 * self.beta_position - self.positions[i])
            X2 = self.beta_position - A2 * D_beta
            A3 = 2 * a * np.random.rand(self.dim) - a
            C3 = 2 * np.random.rand(self.dim)
            D_delta = abs(C3 * self.delta_position - self.positions[i])
            X3 = self.delta_position - A3 * D_delta
            self.positions[i] = (X1 + X2 + X3) / 3
            self.positions[i] = np.clip(self.positions[i], self.lb, self.ub)
    def evaluate_fitness(self):
        for i in range(self.num_wolves):
            fitness = self.objective_function(self.positions[i])
            if fitness < self.alpha_score:
                self.alpha_score = fitness
                self.alpha_position = self.positions[i]
            elif fitness < self.beta_score:
                self.beta_score = fitness
                self.beta_position = self.positions[i]
            elif fitness < self.delta_score:
                self.delta_score = fitness
                self.delta_position = self.positions[i]
    def run(self):
        a = 2
        for t in range(self.max_iter):
            a = 2 - t * (2 / self.max_iter)
            self.update_positions(a)
            self.evaluate_fitness()
            print(f"Iteration {t+1}/{self.max_iter}: Alpha Score = {self.alpha_score}")
        return self.alpha_position, self.alpha_score
dim = 2
num_wolves = 30
max_iter = 10
lb = -5
ub = 5
```

```
gwo = GreyWolfOptimizer(objective_function, dim, num_wolves, max_iter, lb, ub)
best_position, best_score = gwo.run()
print(f"\nBest Position: {best_position}")
print(f"Best Score: {best_score}")
```

```python
import numpy as np
def objective_function(x):
    return np.sum(x**2)
class GreyWolfOptimizer:
    def __init__(self, objective_function, dim, num_wolves, max_iter, lb=-5, ub=5):
        self.objective_function = objective_function
        self.dim = dim
        self.num_wolves = num_wolves
        self.max_iter = max_iter
        self.lb = lb
        self.ub = ub
        self.positions = np.random.uniform(self.lb, self.ub, (self.num_wolves, self.dim))
        self.alpha_position = np.zeros(self.dim)
        self.beta_position = np.zeros(self.dim)
        self.delta_position = np.zeros(self.dim)
        self.alpha_score = float("inf")
        self.beta_score = float("inf")
        self.delta_score = float("inf")
    def update_positions(self, a):
        for i in range(self.num_wolves):
            A1 = 2 * a * np.random.rand(self.dim) - a
            C1 = 2 * np.random.rand(self.dim)
            D_alpha = abs(C1 * self.alpha_position - self.positions[i])
            X1 = self.alpha_position - A1 * D_alpha
            A2 = 2 * a * np.random.rand(self.dim) - a
            C2 = 2 * np.random.rand(self.dim)
            D_beta = abs(C2 * self.beta_position - self.positions[i])
            X2 = self.beta_position - A2 * D_beta
            A3 = 2 * a * np.random.rand(self.dim) - a
            C3 = 2 * np.random.rand(self.dim)
            D_delta = abs(C3 * self.delta_position - self.positions[i])
            X3 = self.delta_position - A3 * D_delta
            self.positions[i] = (X1 + X2 + X3) / 3
            self.positions[i] = np.clip(self.positions[i], self.lb, self.ub)
    def evaluate_fitness(self):
        for i in range(self.num_wolves):
            fitness = self.objective_function(self.positions[i])
            if fitness < self.alpha_score:
                self.alpha_score = fitness
                self.alpha_position = self.positions[i]
            elif fitness < self.beta_score:
                self.beta_score = fitness
                self.beta_position = self.positions[i]
            elif fitness < self.delta_score:
                self.delta_score = fitness
                self.delta_position = self.positions[i]
    def run(self):
        a = 2
        for t in range(self.max_iter):
            a = 2 - t * (2 / self.max_iter)
            self.update_positions(a)
            self.evaluate_fitness()
            print(f"Iteration {t+1}/{self.max_iter}: Alpha Score = {self.alpha_score}")
        return self.alpha_position, self.alpha_score
dim = 2
```

```
dim = 2
num_wolves = 30
max_iter = 10
lb = -5
ub = 5
gwo = GreyWolfOptimizer(objective_function, dim, num_wolves, max_iter, lb, ub)
best_position, best_score = gwo.run()
print(f"\nBest Position: {best_position}")
print(f"Best Score: {best_score}")
```

Output:

```
Iteration 1/10: Alpha Score = 0.00039706625480835054
Iteration 2/10: Alpha Score = 0.00039706625480835054
Iteration 3/10: Alpha Score = 0.00039706625480835054
Iteration 4/10: Alpha Score = 0.00039706625480835054
Iteration 5/10: Alpha Score = 0.00039706625480835054
Iteration 6/10: Alpha Score = 0.00016150473046534417
Iteration 7/10: Alpha Score = 0.00015472273041535067
Iteration 8/10: Alpha Score = 7.06318886984631e-05
Iteration 9/10: Alpha Score = 4.878663579776153e-05
Iteration 10/10: Alpha Score = 4.838048252992564e-05

Best Position: [-0.0007949   0.00691004]
Best Score: 4.838048252992564e-05
```