Cancerous Cell Optimization Algorithm (CCOA) - Full MATLAB Code

%% Cancerous Cell Optimization Algorithm (CCOA)  
function [bestPos, bestVal, convergence] = CancerousCellOptimization(objFunc, dimensions, lb, ub, popSize, maxIter)  
 % Initialization  
 population = lb + (ub - lb) \* rand(popSize, dimensions);  
 fitness = arrayfun(@(i) objFunc(population(i, :)), 1:popSize);  
 [bestVal, bestIdx] = min(fitness);  
 bestPos = population(bestIdx, :);  
 convergence = zeros(1, maxIter);  
 mutationDecay = linspace(0.9, 0.1, maxIter); % gradually decaying mutation strength  
 replicationRate = 0.3; % percentage of cells to replicate  
 suppressionRate = 0.2; % percentage of cells to suppress  
  
 for iter = 1:maxIter  
 % Replication (mitosis of best-performing cells)  
 [~, sortedIdx] = sort(fitness);  
 numReplications = round(replicationRate \* popSize);  
 replicated = population(sortedIdx(1:numReplications), :);  
 replicated = replicated + mutationDecay(iter) \* (rand(size(replicated)) - 0.5);  
 replicated = max(min(replicated, ub), lb);  
  
 % Mutation of the population  
 mutated = population + mutationDecay(iter) \* (rand(size(population)) - 0.5);  
 mutated = max(min(mutated, ub), lb);  
  
 % Combine and Evaluate  
 combined = [population; replicated; mutated];  
 combinedFitness = arrayfun(@(i) objFunc(combined(i, :)), 1:size(combined, 1));  
  
 % Selection (Keep best)  
 [sortedFitness, sortedIdx] = sort(combinedFitness);  
 population = combined(sortedIdx(1:popSize), :);  
 fitness = sortedFitness(1:popSize);  
  
 % Suppress (kill worst-performing cells and regenerate randomly)  
 numSuppress = round(suppressionRate \* popSize);  
 worstIdx = popSize - numSuppress + 1 : popSize;  
 population(worstIdx, :) = lb + (ub - lb) \* rand(numSuppress, dimensions);  
 fitness(worstIdx) = arrayfun(@(i) objFunc(population(i, :)), worstIdx);  
  
 % Update Best  
 [currentBestVal, currentBestIdx] = min(fitness);  
 if currentBestVal < bestVal  
 bestVal = currentBestVal;  
 bestPos = population(currentBestIdx, :);  
 end  
  
 % Ensure convergence curve is monotonic  
 if iter > 1  
 convergence(iter) = min(convergence(iter-1), bestVal);  
 else  
 convergence(iter) = bestVal;  
 end  
 end  
end