Shaurya Gupta OTA Assignment 8

Implement a modified variant of the PSO. This accelerated variant generates the new velocity as per the following expression:

$$v_i^{t+1} = v^t_i + \text{alpha}(\text{epsilon} - 1/2) + \text{beta}(g^* - x^t_i)$$

Where epsilon is a random number between 0 and 1.

Compare the results of PSO and the accelerated PSO for any 5 test functions on the

following parameters for 500 iterations:

a) **Mean function values for 20 runs**

b) **Best function values for 20 runs**

c) **Standard deviation values for 20 runs**

Compare the iteration-wise mean function values plots for both the values.

Program Code for PSO and accelerated PSO:

```
% Parameters
maxIter = 500; % Maximum number of iterations
Npop = 50; % Population size
D = 2; % Dimensionality of the problem
Nruns = 20; % Number of runs
alpha = 1.5; % Stochastic coefficient for APSO
beta = 2.0; % Social coefficient for APSO


% Define test functions and their names
functions = {@quadratic, @cubic, @quartic, @mixed_poly, @biquadratic};
functionNames = {'Quadratic', 'Cubic', 'Quartic', 'Mixed Polynomial',
'Biquadratic'};


% Initialize storage for results (PSO)
psoBestFitHistory = zeros(Nruns, length(functions), maxIter); % Best fitness for
each run
```

```
psoMeanFitHistory = zeros(Nruns, length(functions), maxIter); % Mean fitness for each run


% Initialize storage for results (APSO with modified rule)
apsoBestFitHistory = zeros(Nruns, length(functions), maxIter); % Best fitness for each run
apsoMeanFitHistory = zeros(Nruns, length(functions), maxIter); % Mean fitness for each run


% Standard PSO and Modified APSO
for funcIdx = 1:length(functions)
    f = functions{funcIdx}; % Select the function to optimize
    fprintf('Optimizing Function %s with PSO and Modified APSO\n', functionNames{funcIdx});


    for run = 1:Nruns
        % PSO Initialization
        X = rand(Npop, D) * 10 - 5; % Random initialization in range [-5, 5]
        v = zeros(Npop, D); % Initial velocity is zero
        pbest = X; % Initialize personal best positions
        pbestFit = arrayfun(@(idx) f(X(idx, :)), 1:Npop); % Personal best fitness
        [gbestFit, gbestIdx] = min(pbestFit); % Global best fitness
        gbest = X(gbestIdx, :); % Global best position


        % Modified APSO Initialization (Separate memory for APSO)
        X_apso = X;
        v_apso = v;
        pbest_apso = X_apso;
        pbestFit_apso = pbestFit;
```

```
gbestFit_apso = gbestFit;
gbest_apso = gbest;

% Main loop for iterations
for it = 1:maxIter
    % === Standard PSO Update ===
    w = 0.5 + rand() / 2; % Inertia weight
    for i = 1:Npop
        r1 = rand(1, D); % Random coefficient for cognitive component
        r2 = rand(1, D); % Random coefficient for social component

        % Update velocity
        v(i, :) = w * v(i, :) + 2 * r1 .* (pbest(i, :) - X(i, :)) + 2 * r2 .* (gbest -
X(i, :));

        % Update position
        X(i, :) = X(i, :) + v(i, :);

        % Update personal best
        fit = f(X(i, :));
        if fit < pbestFit(i)
            pbest(i, :) = X(i, :);
            pbestFit(i) = fit;
        end
    end

    % Update global best
    [minFit, minIdx] = min(pbestFit);
```

```
        if minFit < gbestFit
            gbestFit = minFit;
            gbest = pbest(minIdx, :);
        end

        % Store best fitness and mean fitness for PSO
        psoBestFitHistory(run, funcIdx, it) = gbestFit; % Best fitness
        psoMeanFitHistory(run, funcIdx, it) = mean(pbestFit); % Mean fitness

        % === Modified APSO Update ===
        for i = 1:Npop
            epsilon = rand(1, D); % Random number between 0 and 1

            % Modified APSO velocity update rule
            v_apso(i, :) = v_apso(i, :) + alpha * (epsilon - 0.5) + beta *
(gbest_apso - X_apso(i, :));

            % Update position for APSO
            X_apso(i, :) = X_apso(i, :) + v_apso(i, :);

            % Update personal best for APSO
            fit_apso = f(X_apso(i, :));
            if fit_apso < pbestFit_apso(i)
                pbest_apso(i, :) = X_apso(i, :);
                pbestFit_apso(i) = fit_apso;
            end
        end
```

```
            % Update global best for APSO
            [minFit_apso, minIdx_apso] = min(pbestFit_apso);
            if minFit_apso < gbestFit_apso
                gbestFit_apso = minFit_apso;
                gbest_apso = pbest_apso(minIdx_apso, :);
            end

            % Store best fitness and mean fitness for APSO
            apsoBestFitHistory(run, funcIdx, it) = gbestFit_apso; % Best fitness
            apsoMeanFitHistory(run, funcIdx, it) = mean(pbestFit_apso); % Mean
fitness
        end
    end
end

% Results Comparison for 20 Runs
meanBestFit_pso = mean(psoBestFitHistory, 1);
meanMeanFit_pso = mean(psoMeanFitHistory, 1);
stdBestFit_pso = std(psoBestFitHistory, 0, 1);

meanBestFit_apso = mean(apsoBestFitHistory, 1);
meanMeanFit_apso = mean(apsoMeanFitHistory, 1);
stdBestFit_apso = std(apsoBestFitHistory, 0, 1);

% Plot iteration-wise mean function values for PSO
figure;
hold on;
for funcIdx = 1:length(functions)
```

```
    plot(1:maxIter, squeeze(meanMeanFit_pso(:, funcIdx, :)), 'LineWidth', 2, 'DisplayName', ['PSO ' functionNames{funcIdx}]);
end
xlabel('Iteration');
ylabel('Mean Function Value');
title('Iteration-wise Mean Function Values for PSO');
legend('show');
hold off;

% Plot iteration-wise mean function values for Modified APSO
figure;
hold on;
for funcIdx = 1:length(functions)
    plot(1:maxIter, squeeze(meanMeanFit_apso(:, funcIdx, :)), '--', 'LineWidth', 2, 'DisplayName', ['Modified APSO ' functionNames{funcIdx}]);
end
xlabel('Iteration');
ylabel('Mean Function Value');
title('Iteration-wise Mean Function Values for Modified APSO');
legend('show');
hold off;

% Output comparison table-like format
fprintf('--- PSO vs Modified APSO Results ---\n');
fprintf('%-20s %-20s %-20s %-20s %-20s\n', 'Function', 'PSO Mean Best', 'APSO Mean Best', 'PSO Std Dev', 'APSO Std Dev');
for funcIdx = 1:length(functions)
    fprintf('%-20s %-20.6f %-20.6f %-20.6f %-20.6f\n', ...
```

```
        functionNames{funcIdx}, ...
        mean(meanBestFit_pso(:, funcIdx, :)), ...
        mean(meanBestFit_apso(:, funcIdx, :)), ...
        mean(stdBestFit_pso(:, funcIdx, :)), ...
        mean(stdBestFit_apso(:, funcIdx, :)));
end



% Test functions
function val = quadratic(pos)
    x = pos(1);
    y = pos(2);
    val = x^2 + y^2;
end

function val = cubic(pos)
    x = pos(1);
    y = pos(2);
    val = x^2 + y^2 + x*y;
end

function val = quartic(pos)
    x = pos(1);
    y = pos(2);
    val = x^4 + y^4;
end
```

```
function val = mixed_poly(pos)
    x = pos(1);
    y = pos(2);
    val = x^2 + y^2 + 2*x*y;
end


function val = biquadratic(pos)
    x = pos(1);
    y = pos(2);
    val = x^2 + y^2 + x^2*y^2;
end
```

Output:

```
>> pso
Optimizing Function Quadratic with PSO and Modified APSO
Optimizing Function Cubic with PSO and Modified APSO
Optimizing Function Quartic with PSO and Modified APSO
Optimizing Function Mixed Polynomial with PSO and Modified APSO
Optimizing Function Biquadratic with PSO and Modified APSO
--- PSO vs Modified APSO Results ---
Function             PSO Mean Best        APSO Mean Best       PSO Std Dev          APSO Std Dev
Quadratic            0.000625             0.006767             0.000783             0.006880
Cubic                0.000597             0.007510             0.000736             0.006515
Quartic              0.000099             0.000618             0.000298             0.001058
Mixed Polynomial     0.000012             0.000056             0.000020             0.000138
Biquadratic          0.000601             0.009389             0.000683             0.008342
fx >>
```

Shaurya Gupta OTA Assignment 8