ECE 58000 FunWork4

Shaunak Mukherjee

Spring 2024

Problem 1 Exercise 12.7 from the textbook on page 202 in the textbook. We are given two mixtures, A and B. Mixture A contains 30% gold, 40% silver, and 30% platinum, whereas mixture B contains 10% gold, 20% silver, and 70% platinum (all percentages of weight). We wish to determine the ratio of the weight of mixture A to the weight of mixture B such that we have as close as possible to a total of 5 ounces of gold, 3 ounces of silver, and 4 ounces of platinum. Formulate and solve the problem using the linear least-squares method.

Solution to problem 1 We can rewrite this problem in matrix form as below Let: x_1 be the weight of mixture A in oz, x_2 be the weight of mixture B in oz. We aim to find x_1 and x_1 that minimize this objective function. To solve it using the linear least-squares method, we need to rewrite the problem in matrix form. The objective function can be rewritten as, $f(x) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$

$f(x) = \frac{1}{2}\mathbf{x}^T(2\mathbf{A}^T\mathbf{A})\mathbf{x} - \mathbf{x}^T(2\mathbf{A}^T\mathbf{b}) + \mathbf{b}^T\mathbf{b}$

where, $A = \begin{bmatrix} 0.3 & 0.1 \\ 0.4 & 0.2 \\ 0.3 & 0.7 \end{bmatrix}, b = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}$ Now, to find the least squares solution, we solve the normal equations for x* as in page 182: $\mathbf{A^T A x^*} = \mathbf{A^T b}$

we can ompute the matrices $\mathbf{A^T A}$ and $\mathbf{A^T b}$, and then solve for $\mathbf{x}^*$ using matlab using the following script

```
% Given data converted into matrix form
A = [0.3, 0.1; 0.4, 0.2; 0.3, 0.7]; % Coefficient matrix
b = [5; 3; 4]; % Right-hand side vector

% Calculate A_transpose * A
ATA = A.' * A;

% Calculate A_transpose * b
ATB = A.' * b;

% Solve the system of equations using ATA * X = ATb
x_star = ATA \ ATB;

% Extract the values of x and y
x_1 = x_star(1);
x_2 = x_star(2);

% Calculate the ratio of weights of mixture A to mixture B
```

```
ratio = x_1 / x_2;

% Display results
fprintf('Weight of mixture A: %.3f\n', x_1);
fprintf('Weight of mixture B: %.3f\n', x_2);
fprintf('Ratio of weight of mixture A to mixture B: %.3f\n', ratio);
```

The output of the Matlab code is provided below:

```

Weight of mixture A: 10.5665
Weight of mixture B: 0.9606
Ratio of weight of mixture A to mixture B: 11.0000
```

——**Answer**——

Problem 2 Minimize the Griewank function,

$$f(x_1, x_2) = 1 + \frac{x_1^2}{4000} + \frac{x_2^2}{4000} - \cos(x_1)\cos\left(\frac{x_2}{\sqrt{2}}\right)$$

over the search area $\left[\ -7,\ \ 7\ \right] \times \left[\ -7,\ \ 7\ \right]$ using the PSO algorithm and produce plots of the best, average, and the worst objective function values in the population for every generation

Solution to problem 2 Below I show the matlab code for PSO Algo implementation on the function.

```
% Clear workspace
clc
clear
close all

% Define the objective function
syms x1 x2
f_def = 1 + x1.^2 / 4000 + x2.^2 / 4000 - cos(x1) .* cos(x2 / sqrt(2));
f = matlabFunction(f_def);

% Define the range of x1 and x2
x1_range = linspace(-7, 7, 100);
x2_range = linspace(-7, 7, 100);

% Define the PSO parameters
num_particles = 50;          % Size of swarm
i_max_iter = 100;            % Max number of iterations
inertia_w = 0.7;             % Intertial constant < 1.0
cognitive_c1 = 2.0;          % Cognitive constant, usually 2.0
social_c2 = 2.0;             % Social constant, usually 2.0

% Empty arrays to store best, average, and worst objective function values
best_val = zeros(i_max_iter, 1);
```

```
ave_val = zeros(i_max_iter, 1);
worst_val = zeros(i_max_iter, 1);

% Initiate random positions for particles within specified ranges
x1_pos = rand(num_particles, 1) * (max(x1_range) - min(x1_range)) + min(x1_range);
x2_pos = rand(num_particles, 1) * (max(x2_range) - min(x2_range)) + min(x2_range);

% Define article position and particle velocity
p_pos = [x1_pos, x2_pos];
p_vel = rand(num_particles, 2);

% Initialize the best particle position and best global position
p_best = p_pos;
gbest_pos = p_pos(1,:);
gbest_val = f(gbest_pos(1), gbest_pos(2));

% Initialize the figure for subplots
figure('Position', [0, 0, 800 , 350]);
ax1 = subplot(1, 2, 1);
xlabel('x1');
ylabel('x2');
zlabel('Objective Function Value');
title('PSO Optimization Progress');
hold(ax1, 'on');

% Initialize the surface plot
[x1_grid, x2_grid] = meshgrid(x1_range, x2_range);
f_values = f(x1_grid, x2_grid);

% Initialize the contour and particle plot
contourf(x1_grid, x2_grid, f_values, 'LevelStep', 0.3, 'LineWidth', 0.2, 'LineStyle', '--', 'ShowText','on', 'LabelFormat','%0.1f');

p_plot = scatter([], [], 10, 'r', 'filled');  % Empty scatter plot with red filled circles
p_plot.MarkerEdgeColor = 'b';

% PSO main loop
for iter = 1:i_max_iter
    % Update particle positions and velocities
    for i = 1: num_particles

        % Generate vectors
        r = rand();
        s = rand();

        % Algo implementation
        p_vel(i,:) = inertia_w * p_vel(i,:) + cognitive_c1 * r .* (p_best(i,:) - p_pos(i,:)) + social_c2 * s .* (gbest_pos - p_pos(i,:));
        p_pos(i,:) = p_pos(i,:) + p_vel(i,:);


        % Correction factor
        if p_pos(i,1) > max(x1_range)
            p_pos(i,1) = max(x1_range);
```

```
        elseif p_pos(i,1) < min(x1_range)
            p_pos(i,1) = min(x1_range);
        end

        if p_pos(i,2) > max(x2_range)
            p_pos(i,2) = max(x2_range);
        elseif p_pos(i,2) < min(x2_range)
            p_pos(i,2) = min(x2_range);
        end


        % Update personal best
        if f(p_pos(i,1), p_pos(i,2)) < f(p_best(i,1), p_best(i,2))
            p_best(i,:) = p_pos(i,:);
        end
    end

    % Update global best
    for i = 1:num_particles
        if f(p_best(i,1), p_best(i,2)) < gbest_val
            gbest_pos = p_best(i,:);
            gbest_val = f(gbest_pos(1), gbest_pos(2));
        end
    end

    % Update particle plot
    % set(p_plot, 'XData', p_pos(:,1), 'YData', p_pos(:,2), 'ZData', f(p_pos(:,1), p_pos(:,2)));
    set(p_plot, 'XData', p_pos(:,1), 'YData', p_pos(:,2));

    % disp(iter)

    % Pause for a short duration to create cool animation effect (optional)
    pause(0.1);

    % Store best, average, and worst objective function values for this iteration
    best_val(iter) = min(f(p_pos(:,1), p_pos(:,2)));
    best_gbest_val(iter) = min(gbest_val);
    ave_val(iter) = mean(f(p_pos(:,1), p_pos(:,2)));
    worst_val(iter) = max(f(p_pos(:,1), p_pos(:,2)));
end


% Display final result
disp('Optimization finished.');
disp(['Best solution found: x1 = ', num2str(gbest_pos(1)), ', x2 = ', num2str(gbest_pos(2))]);
disp(['Minimum value: ', num2str(gbest_val)]);

% Plotting the best, average, and worst objective function values
generation = 1:i_max_iter;
ax2 = subplot(1, 2, 2);
% plot(ax2, generation, best_gbest_val, 'blue', generation, best_val, 'g', generation, ave_val, 'b', generation, worst_val, 'r');
plot(ax2, generation, best_gbest_val, 'greeno-', generation, best_val, 'blueo-', generation, ave_val, 'blacko-', generation, worst_val, 'redo-', 'MarkerSize', 2);
```

```
title(ax2, 'PSO of Griwank Function');
xlabel(ax2, 'Generation');
ylabel(ax2, 'Objective Function Value');
legend(ax2, 'Global Best', 'Best', 'Average', 'Worst');
grid(ax2, 'on');
```

Below the result output & plot of Global best, objective function's best, average and the worst objective function value with swarm size of 50, with 100 iteration or generations with 0.7 as inertia and 2.0 for c_1 and c_2.

```
Optimization finished.
Best solution found: x1 = -1.4995e-05, x2 = 7.9562e-06
Minimum value: 1.2832e-10
```

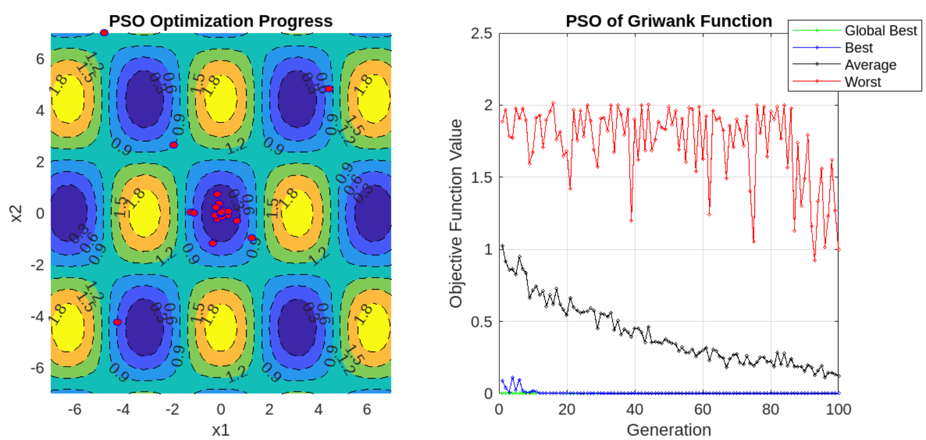

Figure 1: PSO Algo: Griewank Function with 50 swarm size & 100 iterations

Next result output & plot of Global best, objective function's best, average and the worst objective function value with swarm size of 50, with 500 iteration or generations with 0.7 as inertia and 2.0 for c_1 and c_2. It shows how the average and worst objective function values starts to converge at large iterations

```
Optimization finished.
Best solution found: x1 = 8.3276e-09, x2 = 9.8046e-09
Minimum value: 0
```

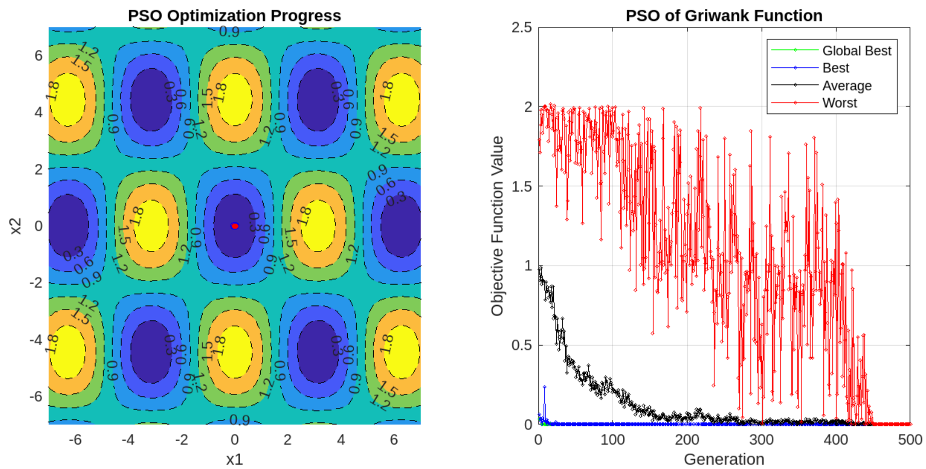

Figure 2: PSO Algo: Griewank Function with 50 swarm size & 500 iterations

———**Answer**———

Problem 3 Minimize the function from Example 14.3 on page 237 in the textbook. The function provided is,

$$f(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right) e^{-x^2-y^2} - \frac{e^{-(x+1)^2-y^2}}{3}$$

over the search area $\begin{bmatrix} -3, & 3 \end{bmatrix} \times \begin{bmatrix} -3, & 3 \end{bmatrix}$ using the PSO algorithm and produce plots of the best, average, and the worst objective function values in the population for every generation.

Solution to problem 3 Below I show the matlab code for PSO Algo implementation on the function.

```
% Clear workspace
clc
clear
close all
rng(1,"twister");

% Define the objective function
syms x y
f_def = 3*(1-x).^2 .* exp(-x.^2-(y+1).^2) ...
            - 10*((x/5) - x.^3 - y.^5) .* exp(-x.^2 - y.^2) ...
            - exp(-(x+1).^2 - y.^2)/3;
f = matlabFunction(f_def);

```

```
% Define the range of x and y
x_range = linspace(-3, 3, 100);
y_range = linspace(-3, 3, 100);

% Define the PSO parameters
num_particles = 30;          % Size of swarm
i_max_iter = 100;            % Max number of iterations
inertia_w = 0.7;             % Intertial constant < 1.0
cognitive_c1 = 1.5;          % Cognitive constant, usually 2.0
social_c2 = 1.5;             % Social constant, usually 2.0
phi = 4.1;
kappa = 2/(abs(2-phi-sqrt(phi^2-4*phi)));

% Empty arrays to store best, average, and worst objective function values
best_val = zeros(i_max_iter, 1);
ave_val = zeros(i_max_iter, 1);
worst_val = zeros(i_max_iter, 1);

% Initiate random positions for particles within specified ranges
x_pos = rand(num_particles, 1) * (max(x_range) - min(x_range)) + min(x_range);
y_pos = rand(num_particles, 1) * (max(y_range) - min(y_range)) + min(y_range);

% Define article position and particle velocity
p_pos = [x_pos, y_pos];
p_vel = rand(num_particles, 2);

% Initialize the best particle position and best global position
p_best = p_pos;
gbest_pos = p_pos(1,:);
gbest_val = f(gbest_pos(1), gbest_pos(2));

% Initialize the figure for subplots
figure('Position', [0, 0, 800 , 350]);
ax = subplot(1, 2, 1);
xlabel('x');
ylabel('y');
zlabel('Objective Function Value');
title('PSO Optimization Progress');
hold(ax, 'on');

% Initialize the surface plot
[x_grid, y_grid] = meshgrid(x_range, y_range);
f_values = f(x_grid, y_grid);

% Initialize the particle plot
contourf(x_grid, y_grid, f_values, 'LevelStep', 1, 'LineWidth', 0.1, 'LineStyle', '--', 'ShowText','On', 'LabelFormat','%0.1f');
p_plot = scatter([], [], 50, 'r', 'filled');  % Empty scatter plot with red filled circles
p_plot.MarkerEdgeColor = 'b';


% PSO main loop
for iter = 1:i_max_iter
    % Update particle positions and velocities
    for i = 1: num_particles

```

```
        % Generate vectors
        r = rand();
        s = rand();

        % Algo implementation
        p_vel(i,:) = inertia_w * p_vel(i,:) + cognitive_c1 * r .* (p_best(i,:) - p_pos(i,:)) + social_c2 * s .* (gbest_pos - p_pos(i,:));

        % Clerc's Cnstriction factor
        % p_vel(i,:) = kappa*(p_vel(i,:) + cognitive_c1 * r .* (p_best(i,:)) - p_pos(i,:)) + social_c2 * s .* (gbest_pos - p_pos(i,:));
        p_pos(i,:) = p_pos(i,:) + p_vel(i,:);

        % Correction factor
        if p_pos(i,1) > max(x_range)
            p_pos(i,1) = max(x_range);
        elseif p_pos(i,1) < min(x_range)
            p_pos(i,1) = min(x_range);
        end

        if p_pos(i,2) > max(y_range)
            p_pos(i,2) = max(y_range);
        elseif p_pos(i,2) < min(y_range)
            p_pos(i,2) = min(y_range);
        end

        % Update personal best
        if f(p_pos(i,1), p_pos(i,2)) < f(p_best(i,1), p_best(i,2))
            p_best(i,:) = p_pos(i,:);
        end
    end

    % Update global best
    for i = 1:num_particles
        if f(p_best(i,1), p_best(i,2)) < gbest_val
            gbest_pos = p_best(i,:);
            gbest_val = f(gbest_pos(1), gbest_pos(2));
        end
    end

    % Update particle plot
    % set(p_plot, 'XData', p_pos(:,1), 'YData', p_pos(:,2), 'ZData', f(p_pos(:,1), p_pos(:,2)));
    set(p_plot, 'XData', p_pos(:,1), 'YData', p_pos(:,2));

    % disp(iter)

    % Pause for a short duration to create cool animation effect (optional)
    pause(0.1);

    % Store best, average, and worst objective function values for this iteration
    best_val(iter) = min(f(p_pos(:,1), p_pos(:,2)));
    best_gbest_val(iter) = min(gbest_val);
    ave_val(iter) = mean(f(p_pos(:,1), p_pos(:,2)));
    worst_val(iter) = max(f(p_pos(:,1), p_pos(:,2)));

end
```

```


% Display final result
disp('Optimization finished.');
disp(['Best solution found: x = ', num2str(gbest_pos(1)), ', y = ', num2str(
    gbest_pos(2))]);
disp(['Minimum value: ', num2str(gbest_val)]);

% Plotting the best, average, and worst objective function values
generation = 1:i_max_iter;
ay = subplot(1, 2, 2);
plot(ay, generation, best_gbest_val, 'greeno-', generation, best_val, 'blueo-',
    generation, ave_val, 'blacko-', generation, worst_val, 'redo-', 'MarkerSize'
    , 2);
title(ay, 'PSO of Function');
xlabel(ay, 'Generation');
ylabel(ay, 'Objective Function Value');
legend(ay, 'Global Best', 'Best', 'Average', 'Worst');
grid(ay, 'on');
```

Next result output & plot of Global best, objective function's best, average and the worst objective function value with swarm size of 30, with 100 iteration or generations with 0.7 as inertia and 1.5 for c_1 and c_2. It shows how the average and worst objective function values starts to converge at large iterations

```
Optimization finished.
Best solution found: x = 0.22828, y = -1.6255
Minimum value: -6.5511
```

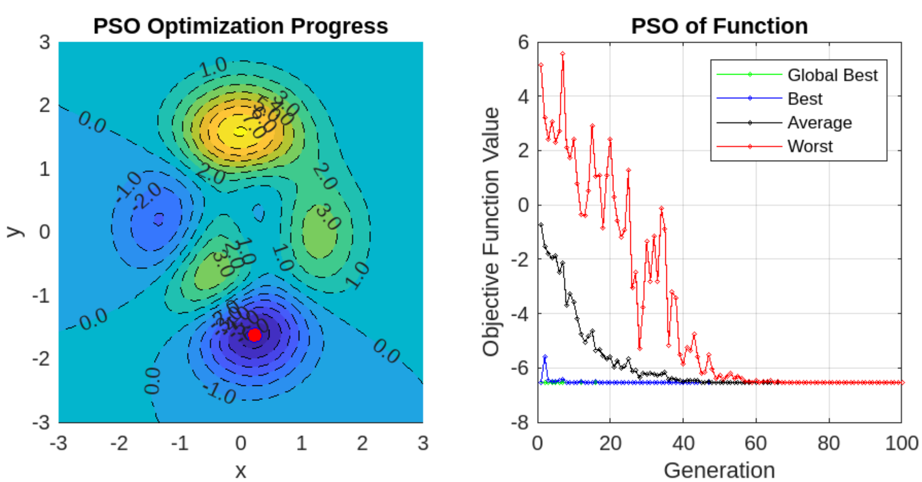

Figure 3: PSO Algo: 30 swarm size & 100 iterations

———**Answer**———

Problem 4 Question- Consider the classic traveling salesperson problem (TSP): Suppose a salesperson must visit clients in different cities, and then return home. What is the shortest tour through those cities, visiting each one once and only once? Implement in MATLAB a simple variant of the genetic algorithm that optimizes the salesperson route for 10 cities. How many different possible paths are there for 10 cities? Use the following representation of any candidate solution: for a given map of the cities to be visited (including the salesperson's home base), assign to each city a positive integer. Thus, for example, if there were six cities, one possible solution might be

$$[1\ 2\ 3\ 4\ 5\ 6]$$

The above represents an order of progression of the salesperson's trip. Note that because the problem requires a round trip, the first coordinate of the vector representing the first city of the trip is also the last city visited by the salesperson. Because the trip is a closed loop, it does not matter from which city the salesperson starts his or her trip. As far as a crossover operator implementation is concerned, use a single-parent operator that produces an offspring by inverting the visiting order between two randomly chosen coordinates of the parent vector. As an example of this implementation of the single-parent crossover operator, suppose that the parent is as above and that the randomly chosen coordinates, that is, the inversion points, are the second and fifth ones. The offspring would then be

$$[1\ 5\ 3\ 4\ 2\ 6]$$

Plot a map with the cities and mark the obtained salesperson's route. The coordinates of the cities are given in the following table.

Locations of the cities										
x	0.4306	3.7094	6.9330	9.3582	4.7758	1.2910	4.83831	9.4560	3.6774	3.2849
y	7.7288	2.9727	1.7785	6.9080	2.6394	4.5774	8.43692	8.8150	7.0002	7.5569

Solution to problem 4

Below I show the matlab code for GA Algo implementation on the TSP Problem.

```
% Clear workspace and command window
close All
clear
clc

% Locations of the cities
x = [0.4306 3.7094 6.9330 9.3582 4.7758 1.2910 4.8383 9.4560 3.6774 3.2849];
y = [7.7288 2.9727 1.7785 6.9080 2.6394 4.5774 8.4369 8.8150 7.0002 7.5569];

num_cities = 10; % Given

% City labels
```

```
city_labels = {'City 1', 'City 2', 'City 3', 'City 4', 'City 5', 'City 6', 'City 7', 'City 8', 'City 9', 'City 10'};

% Plot the cities
figure;
scatter(x, y, 'o', 'blue');
hold on;
text(x, y, city_labels, 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
title('Locations of the cities');
xlabel('x');
ylabel('y');
grid on;

% Possible paths to visit the cities
possible_path = factorial(num_cities - 1);
disp(['Number of different possible paths for visiting ', num2str(num_cities), ' cities once and only once: ', num2str(possible_path)]);

% Genetic Algorithm Parameters
pop_size = 50;  % popation size
num_gens = 100; % Number of generations
p_m = 0.01;     % Rate of mutation (optional)

% Initialize population (pop)
pop = zeros(pop_size, num_cities);
for i = 1 : pop_size
    pop(i, :) = randperm(num_cities);
end

% Evaluate fitness
for generation = 1:num_gens
    fitness = zeros(pop_size, 1);
    for i = 1:pop_size
        fitness(i) = evaluate_fitness(pop(i, :), x, y);
    end

    % Selection process
    [~, sorted_indices] = sort(fitness, 'descend');
    selected_indices = sorted_indices(1:pop_size/2);
    parents = pop(selected_indices, :);

    % Crossover: Single-point crossover
    for i = 1:2:pop_size
        parent1 = parents(mod(i, pop_size/2) + 1, :); % Parent #1
        parent2 = parents(mod(i + 1, pop_size/2) + 1, :); % Parent #2
        x_over = randi([1, num_cities - 1]); % Crossover

        offspring1 = [parent1(1:x_over), setdiff(parent2, parent1(1:x_over), 'stable')];
        offspring2 = [parent2(1:x_over), setdiff(parent1, parent2(1:x_over), 'stable')];
        pop(i, :) = mutate(offspring1, p_m); % Apply mutation
        pop(i + 1, :) = mutate(offspring2, p_m); % Apply mutation

        % pop(i, :) = mutate(offspring1, p_m); % No mutation
        % pop(i + 1, :) = mutate(offspring2, p_m); % No mutation
```

```
    end
    % Store best fitness of this generation
    best_fit_per_gen(generation) = max(fitness);
end


% Report the best route
best_route = pop(1, :);
total_distance = min(1./best_fit_per_gen);
best_fitness = evaluate_fitness(best_route, x, y);
disp(['Best TSP City Route: [' num2str(best_route),']']);
disp(['Best fitneess: ', num2str(best_fitness)]);
disp(['Total distance: ', num2str(total_distance)]);

% Plot the best route
plot_route(x, y, best_route);
title(['Best Fitness: ', num2str(best_fitness)]);

% Plot generation vs best fitness
figure;
plot(1:num_gens, 1./best_fit_per_gen,'blue', 'LineWidth', 1.0);
title('Generation vs Total distance');
xlabel('Generation');
ylabel('Total distance');


% Function to evaluate fitness
function fitness = evaluate_fitness(route, x, y)
    distance = 0;
    num_cities = length(route);
    for i = 1:num_cities-1
        distance = distance + sqrt((x(route(i)) - x(route(i+1)))^2 + (y(route(i)) - y(route(i+1)))^2);
    end
    distance = distance + sqrt((x(route(end)) - x(route(1)))^2 + (y(route(end)) - y(route(1)))^2);
    fitness = 1/distance;
end

% Function to mutate a route
function mutated_route = mutate(route, p_m_rate)
    num_cities = length(route);
    for i = 1:num_cities
        if rand < p_m_rate
            j = randi([1, num_cities]);
            route([i, j]) = route([j, i]);
        end
    end
    mutated_route = route;
end


% Function to show the path with arrows
function plot_route(x, y, route)
    plot(x(route), y(route), 'blue', 'LineWidth', 0.1);
    scatter(x(route), y(route), 'filled', 'MarkerEdgeColor', 'blue');
end
```

Next result output & plot of the map with the cities travelled by best route the obtained salesperson's route.

```
Number of different possible paths for visiting 10 cities once and only once:
    362880
Best TSP City Route: [7    8    4    3    5    2    6    1   10    9]
Best fitneess: 0.036747
Total distance: 27.2133
```

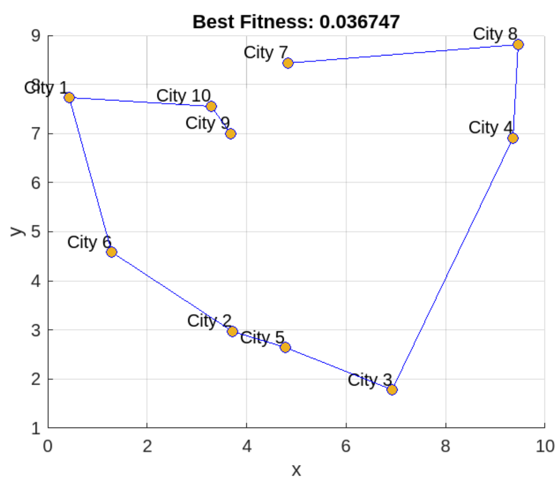

Figure 4: TSP best route trace using GA

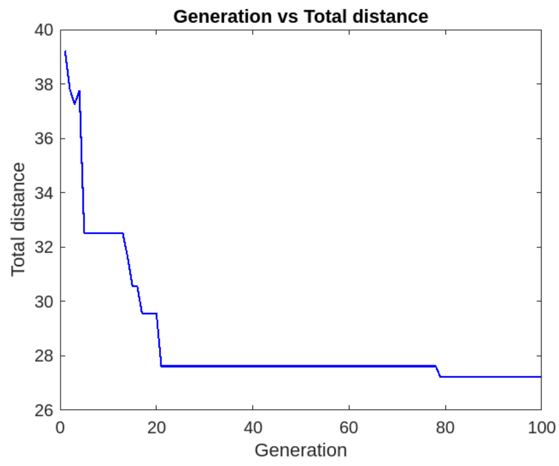

Figure 5: TSP total distance vs generation plot

——**Answer**——

Problem 5 Solve Example 15.4, on pages 253–254 using MATLAB's function linprog. The problem – A manufacturing company has plants in cities A, B, and C. The company produces and distributes its product to dealers in various cities. On a particular day, the company has 30 units of its product in A, 40 in B, and 30 in C. The company plans to ship 20 units to D, 20 to E, 25 to F, and 35 to G, following orders received from dealers. The transportation costs per unit of each product between the cities are given in Table below.

To/From	D	E	F	G	Supply
A	7	10	14	8	30
B	7	11	12	6	40
C	5	8	15	9	30
Demand	20	20	25	35	100

Quantities to be transported from plants to different destinations are represented by the decision variables. Below we will be using MATLAB's function linprog to minimize the total transportation cost. The linear equations are already provided as follows.

$$\begin{aligned} \text{minimize} \quad & -\frac{3}{4}x_4 + 20x_5 - \frac{1}{2}x_6 + 6x_7 \\ \text{subject to} \quad & x_1 + \frac{1}{4}x_4 - 8x_5 - x_6 + 9x_7 = 0 \\ & x_2 + \frac{1}{2}x_4 - 12x_5 - \frac{1}{2}x_6 + 3x_7 = 0 \\ & x_3 + x_6 = 1 \\ & x_1, \ldots, x_7 > 0. \end{aligned}$$

Solution to problem 5 We can use Matlab with linprog **link** to form the matrices and solve the LPP as below

```
% Clear workspace
clear

% Define the costs matrix
costs = [7 10 14 8; 7 11 12 6; 5 8 15 9];

% Reshape costs matrix into a column vector
c_func = reshape(costs', [], 1);

% Define the equality constraints matrix
Aeq = [1 1 1 1 0 0 0 0 0 0 0 0;
       0 0 0 0 1 1 1 1 0 0 0 0;
       0 0 0 0 0 0 0 0 1 1 1 1;
       1 0 0 0 1 0 0 0 1 0 0 0;
       0 1 0 0 0 1 0 0 0 1 0 0;
       0 0 1 0 0 0 1 0 0 0 1 0;
       0 0 0 1 0 0 0 1 0 0 0 1;];
```

```
% Define the equality constraints vector
beq = [30; 40; 30; 20; 20; 25; 35];

% Define lower bound (non-negative constraint)
lb = zeros(12, 1);

% Solve the linear programming problem
% (source- http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/optim/linprog.html)
[c_func, fval, exitflag] = linprog(c_func, [], [], Aeq, beq, lb);

% Display the optimal solution
disp('Optimal solution c_func:')
disp(c_func)

% Reshape the solution vector into a matrix
sol = reshape(c_func, 4, 3)';

% Create a table for the optimal solution
Table = array2table(sol, 'VariableNames', {'D', 'E', 'F', 'G'}, 'RowNames', {'A', 'B', 'C'});


% Display the optimal solution table
disp('Optimal solution tabulated:')
disp(Table)

% Display the total transportation cost
disp('Total transportation cost:')
disp(fval)
```

The output of the Matlab code are provided below:

```
Optimal solution found.
Optimal solution c_func:
    10
     0
    20
     0
     0
     0
     5
    35
    10
    20
     0
     0

Optimal solution tabulated:
         D     E     F     G
         --    --    --    --

    A    10     0    20     0
    B     0     0     5    35
    C    10    20     0     0

```

```
Total transportation cost:
   830
```

——Answer——