```
import numpy as np
import random

# Set the seed for reproducibility (set any number to generate the same random sequence every time)
random.seed(42)  # Seed for Python random module
np.random.seed(42)  # Seed for NumPy random module

# New distance matrix generation code with a larger range of distances (1 to 50)
def generate_random_distance_matrix(n, min_dist=1, max_dist=50):
    distance_matrix = np.random.randint(min_dist, max_dist, size=(n, n))  # Larger range of distances
    np.fill_diagonal(distance_matrix, 0)  # No distance from a point to itself
    return distance_matrix

# Function to calculate the total distance of a route
def calculate_total_distance(route, distance_matrix):
    total_distance = 0
    for i in range(len(route) - 1):
        total_distance += distance_matrix[route[i]][route[i+1]]
    return total_distance

# Nearest Neighbor Algorithm for initial route selection
def nearest_neighbor(distance_matrix, start=0):
    n = len(distance_matrix)
    unvisited = set(range(n))
    unvisited.remove(start)
    route = [start]

    current = start
    while unvisited:
        next_city = min(unvisited, key=lambda city: distance_matrix[current][city])
        route.append(next_city)
        unvisited.remove(next_city)
        current = next_city
    route.append(start)  # Returning to the start point
    return route

# 2-Opt Algorithm for optimization
def two_opt(route, distance_matrix):
    best = route
    improved = True
    while improved:
        improved = False
        for i in range(1, len(route) - 2):
            for j in range(i + 1, len(route) - 1):
                if j - i == 1:
                    continue
                new_route = route[:]
                new_route[i:j] = route[j-1:i-1:-1]  # Reverse the route segment
                new_distance = calculate_total_distance(new_route, distance_matrix)
                best_distance = calculate_total_distance(best, distance_matrix)

                # If we found an improvement, update the best route
                if new_distance < best_distance:
                    best = new_route
                    improved = True
        route = best
    return best

# Get inputs from the user
num_points = int(input("Enter the number of collection points: "))  # Number of points
fuel_cost_per_litre = float(input("Enter the fuel cost per litre (in Rs): "))  # Rs per litre
co2_per_litre = float(input("Enter the CO2 emission per litre (in kg): "))  # CO2 per litre
fuel_efficiency = float(input("Enter the fuel efficiency (km per litre): "))  # km per litre

# Generate random distance matrix
distance_matrix = generate_random_distance_matrix(num_points)

# --- Step 1: Manually set a bad initial route ---
initial_route = [0, 1, 2, 3, 0]  # Manually set a poor initial route (Depot -> Point1 -> Point2 -> Point3 -> Depot)
initial_distance = calculate_total_distance(initial_route, distance_matrix)

# --- Step 2: 2-opt Optimization ---
optimized_route = two_opt(initial_route, distance_matrix)
optimized_distance = calculate_total_distance(optimized_route, distance_matrix)

# Environmental and Cost Calculations
initial_fuel = initial_distance / fuel_efficiency
optimized_fuel = optimized_distance / fuel_efficiency
fuel_saved = initial_fuel - optimized_fuel
cost_saved = fuel_saved * fuel_cost_per_litre
co2_saved = fuel_saved * co2_per_litre
```

```
# Output results
print("\n--- Initial Route and Distance ---")
print(f"Initial Route: {initial_route}")
print(f"Initial Total Distance: {initial_distance} km")

print("\n--- Optimized Route and Distance ---")
print(f"Optimized Route: {optimized_route}")
print(f"Optimized Total Distance: {optimized_distance} km")

print("\n--- Environmental and Cost Impact ---")
print(f"Fuel Used (Initial): {initial_fuel:.2f} liters, Cost: Rs {initial_fuel * fuel_cost_per_litre:.2f}, CO2 Emitted: {initial_fuel * c
print(f"Fuel Used (Optimized): {optimized_fuel:.2f} liters, Cost: Rs {optimized_fuel * fuel_cost_per_litre:.2f}, CO2 Emitted: {optimized_
print(f"Fuel Saved: {fuel_saved:.2f} liters")
print(f"Cost Saved: Rs {cost_saved:.2f}")
print(f"CO2 Emission Reduced: {co2_saved:.2f} kg")
```

```
Enter the number of collection points: 4
Enter the fuel cost per litre (in Rs): 100
Enter the CO2 emission per litre (in kg): 3.0
Enter the fuel efficiency (km per litre): 5

--- Initial Route and Distance ---
Initial Route: [0, 1, 2, 3, 0]
Initial Total Distance: 128 km

--- Optimized Route and Distance ---
Optimized Route: [0, 2, 1, 3, 0]
Optimized Total Distance: 81 km

--- Environmental and Cost Impact ---
Fuel Used (Initial): 25.60 liters, Cost: Rs 2560.00, CO2 Emitted: 76.80 kg
Fuel Used (Optimized): 16.20 liters, Cost: Rs 1620.00, CO2 Emitted: 48.60 kg
Fuel Saved: 9.40 liters
Cost Saved: Rs 940.00
CO2 Emission Reduced: 28.20 kg
```