Maximizing the profits for a tourist bus agency .

Harshal Rudraksha-230002061
Vansh Raj Singh-230002079
Vansh Sabharwal-230002080

Project Description

- **Tourist bus agencies face challenges such as poorly planned routes, underutilization of buses, lower demand on various routes, and high operational costs, such as fuel and maintenance expenses.**
- **Optimization provides a systematic approach to tackle these issues by identifying the most efficient ways to allocate resources, plan routes, and price services.**

Assumptions:

- For simplicity , we didn’t add the constraint of capacity of the bus. i.e. we assumed that the demand of all customers is satisfied that is all customers reach their desired destination .
- maintenance costs other than fuel costs were ignored
- the bus doesnt return back to the previous city visited.

Mathematical Model

Objective Function

Maximize total profit, given by:

$$\text{Maximize} \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} \cdot d_{ij} \cdot x_{ij}$$

This function maximizes revenue based on both demand and distance.

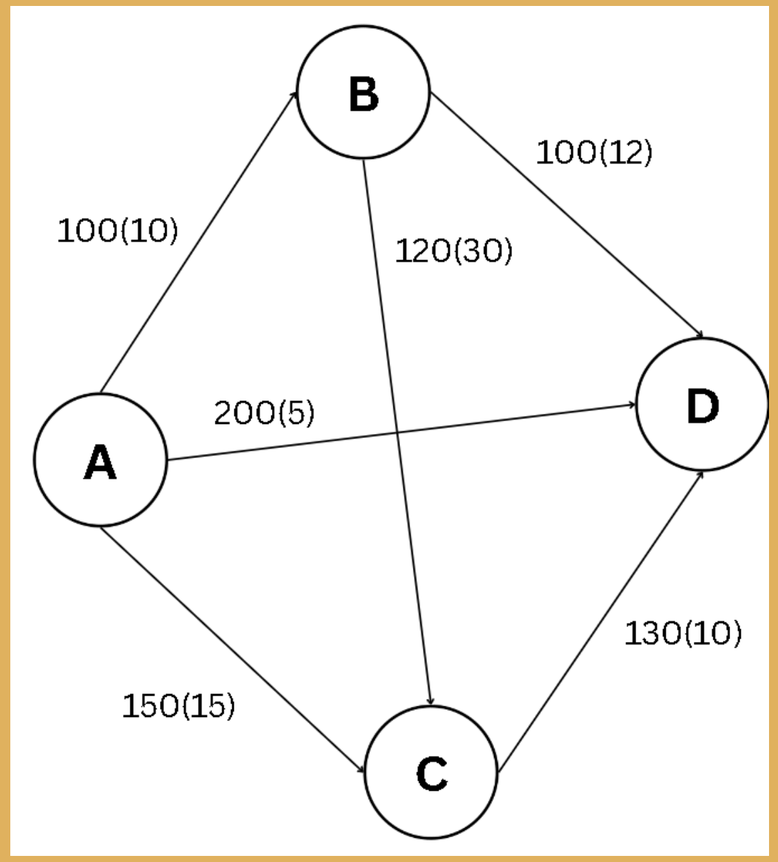
Decision Variables

1. x_{ij}: Binary variable indicating whether the route includes a direct trip from city i to city j (1 if yes, 0 otherwise).

Methodology:

- Use of Linear Programming (LP) for route and capacity optimization
- Incorporating ticket prices on the basis of distance travelled
- Use of genetic algorithm to optimize the total profit
- Use of modules in Python (e.g., itertools,random etc.), or specialized optimization software.

THE FIGURE SHOWS THE POSSIBLE ROUTES FOR THE FOUR CITIES PROBLEM



*DISTANCE(DEMAND)

Brute Force Approach for 4 cities

```python
# Define function to calculate profit for a given path
def calculate_profit(path):
    profit = 0
    total_fuel_cost = 0
    for i in range(len(path) - 1):
        start, end = path[i], path[i + 1]
        distance = distances[(start, end)]
        demand = demands[(start, end)]
        revenue = demand * distance * ticket_price_per_km
        fuel_cost = distance * fuel_cost_per_km
        total_fuel_cost += fuel_cost
        profit += (revenue - fuel_cost)
    return profit

# Generate all valid paths from A to D without revisiting cities
start_city = 'A'
end_city = 'D'
valid_paths = [path for path in itertools.permutations(cities) if path[0] == start_city and path[-1] == end_city]

# Calculate the profit for each path and find the optimal one
optimal_path = None
max_profit = 0
for path in valid_paths:
    profit = calculate_profit(path)
    if profit > max_profit:
        max_profit = profit
        optimal_path = path

# Print the optimal path and profit
print("\nOptimal path and profit:")
print(f"Optimal path: {' -> '.join(optimal_path)}")
print(f"Maximum profit: {max_profit:.2f} rupees")
```

```
Optimal path and profit:
Optimal path: A -> B -> C -> D
Maximum profit: 29325.00 rupees
```

Key Steps in Genetic Algorithm

- Initialization: The initial population is generated randomly.
- Selection: Tournament selection ensures that the fittest individuals (here, the more profitable paths) are more likely to be chosen as parents.
- Crossover: Offspring inherit a mix of characteristics from parents.
- Mutation: Adds randomness to avoid local optima and maintain diversity.
- Iteration: Repeats the process for a fixed number of generations.
- Evaluation: Tracks the best solution found throughout the generations.

Optimal Approach for 4 cities

```python
# Function to calculate profit for a given path
def calculate_profit(path):
    profit = 0
    total_fuel_cost = 0
    for i in range(len(path) - 1):
        start, end = path[i], path[i + 1]
        distance = distances.get((start, end), float('inf'))
        demand = demands.get((start, end), 0)
        revenue = demand * distance * ticket_price_per_km
        fuel_cost = distance * fuel_cost_per_km
        total_fuel_cost += fuel_cost
        profit += (revenue - fuel_cost)
    return profit

# Generate a random path
start_city = 'A'
end_city = 'D'
def generate_random_path():
    middle_cities = cities[1:-1]  # Exclude start and end cities
    random.shuffle(middle_cities)
    return [start_city] + middle_cities + [end_city]

# Genetic Algorithm Parameters
population_size = 100
generations = 200
mutation_rate = 0.1

# Initialize population
population = [generate_random_path() for _ in range(population_size)]

# Genetic Algorithm
for generation in range(generations):
    # Calculate fitness for each path
    fitness_scores = [(path, calculate_profit(path)) for path in population]
    fitness_scores.sort(key=lambda x: x[1], reverse=True)

    # Selection: take the top 50% paths
    selected_paths = [path for path, _ in fitness_scores[:population_size // 2]]

    # Crossover: create new paths from selected ones
    offspring = []
    while len(offspring) < population_size // 2:
        parent1, parent2 = random.sample(selected_paths, 2)
        cut = random.randint(1, len(cities) - 2)
        child = parent1[:cut] + [city for city in parent2 if city not in parent1[:cut]]
        offspring.append(child)

    # Mutation: randomly swap cities in paths
    for path in offspring:
        if random.random() < mutation_rate:
            i, j = random.sample(range(1, len(cities) - 1), 2)
            path[i], path[j] = path[j], path[i]

    # Update population
    population = selected_paths + offspring

# Get the best path
best_path = max(population, key=calculate_profit)
max_profit = calculate_profit(best_path)

# Print the optimal path and profit
print("\nOptimal path and profit:")
print(f"Optimal path: {' -> '.join(best_path)}")
print(f"Maximum profit: {max_profit:.2f} rupees")
```

```
Optimal path and profit:
Optimal path: A -> B -> C -> D
Maximum profit: 29325.00 rupees
```

OUTPUT FOR TWELVE CITIES WITH RANDOMLY ASSIGNED DISTANCES AND DEMANDS

```
Generation 0, Best Profit: 376995.00
Generation 100, Best Profit: 394175.00
Generation 200, Best Profit: 394175.00
Generation 300, Best Profit: 394175.00
Generation 400, Best Profit: 394175.00
Generation 500, Best Profit: 394175.00
Generation 600, Best Profit: 394175.00
Generation 700, Best Profit: 394175.00
Generation 800, Best Profit: 394175.00
Generation 900, Best Profit: 394175.00

Optimal path and profit:
Optimal path: A -> F -> K -> H -> I -> J -> B -> C -> G -> D
Maximum profit: 394175.00 rupees
```

Thank You