Competition 05 - Fleet Optimization Challenge

Management Science - EcoExpress Logistics Fleet

Client Briefing

EcoExpress Logistics needs your consulting expertise to design a sustainable delivery fleet.

Operations Director's Dilemma: "EU regulations demand 40% emission cuts, but we can't sacrifice profitability, service quality, or reliability!"

The Situation

The current fleet of 80 diesel vans faces critical challenges:

- EU Green Deal: 40% emission reduction mandate by 2025
- Rising fuel costs (€2.1/L diesel)
- Competition entering market with 2-hour delivery promise
- Driver shortage (15% unfilled positions)

The Challenge

The current fleet of 80 diesel vans must be replaced to meet EU regulations and remain competitive.

Your task: Design the optimal fleet composition balancing:

1. Minimize Total Cost (purchase + annual operating costs)
2. Maximize Service Score (capacity + reliability + speed)
3. Hard Constraint: Average CO2 emissions ≤ 111 g/km

Key Facts

- Daily demand: 22,000 parcels across 3 cities
- Average daily distance: 1,200 km
- Operating days per year: 360
- Budget guideline: ~€35M capital investment

Available Vehicles

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from typing import Dict, List, Tuple

# Set random seed
np.random.seed(2025)

# Vehicle specifications
```

```
vehicles = pd.DataFrame({
    'Type': ['Electric Truck', 'Hybrid Van', 'Diesel Van', 'E-Cargo Bike',
'Autonomous Pod'],
    'Purchase_Cost': [75000, 45000, 35000, 12000, 95000],  # €
    'Operating_Cost_per_km': [0.18, 0.25, 0.38, 0.05, 0.12],  # €/km
    'CO2_per_km': [0, 95, 185, 0, 0],  # g/km
    'Speed_kmh': [55, 65, 70, 30, 40],  # km/h average
    'Daily_Capacity': [300, 200, 250, 50, 150],  # parcels per vehicle
    'Reliability': [0.93, 0.95, 0.88, 0.98, 0.94],  # uptime %
})

print("EcoExpress Vehicle Options:")
print(vehicles.to_string(index=False))
```

```
EcoExpress Vehicle Options:
          Type  Purchase_Cost  Operating_Cost_per_km  CO2_per_km  Speed_kmh
Daily_Capacity  Reliability
Electric Truck          75000                   0.18           0         55
300          0.93
    Hybrid Van          45000                   0.25          95         65
200          0.95
    Diesel Van          35000                   0.38         185         70
250          0.88
  E-Cargo Bike          12000                   0.05           0         30
50           0.98
Autonomous Pod          95000                   0.12           0         40
150          0.94
```

The Vehicles

- Electric Truck: Zero emissions, high capacity, expensive
- Hybrid Van: Balanced option, good reliability
- Diesel Van: Cheapest, but high emissions (may violate constraint!)
- E-Cargo Bike: Zero emissions, perfect for city centers, low capacity
- Autonomous Pod: Expensive upfront, low operating cost, 24/7 operation

Helper Functions

Calculate Fleet Metrics

This function evaluates ANY fleet composition on both objectives and the constraint.

```
def calculate_fleet_metrics(fleet_composition: Dict[str, int]) -> Dict:
    """
    Calculate metrics for a given fleet composition.

    Args:
        fleet_composition: Dict like {'Electric Truck': 20, 'Hybrid Van':
30, ...}
```

```
    Returns:
        Dict with: total_cost, service_score, avg_co2, total_capacity,
total_vehicles
    """
    # Constants
    DAILY_DISTANCE = 1200  # km
    OPERATING_DAYS = 360
    REQUIRED_CAPACITY = 22000  # parcels/day

    # Initialize
    total_purchase = 0
    total_annual_operating = 0
    total_capacity = 0
    weighted_co2 = 0
    weighted_reliability = 0
    weighted_speed = 0
    total_vehicles = sum(fleet_composition.values())

    # Calculate for each vehicle type
    for vehicle_type, quantity in fleet_composition.items():
        if quantity == 0:
            continue

        # Get vehicle specs
        vehicle = vehicles[vehicles['Type'] == vehicle_type].iloc[0]

        # Costs
        total_purchase += vehicle['Purchase_Cost'] * quantity
        annual_operating = vehicle['Operating_Cost_per_km'] *
DAILY_DISTANCE * OPERATING_DAYS * quantity
        total_annual_operating += annual_operating

        # Capacity and performance
        total_capacity += vehicle['Daily_Capacity'] * quantity
        weighted_co2 += vehicle['CO2_per_km'] * quantity
        weighted_reliability += vehicle['Reliability'] * quantity
        weighted_speed += vehicle['Speed_kmh'] * quantity

    # Calculate averages
    avg_co2 = weighted_co2 / total_vehicles if total_vehicles > 0 else 999
    avg_reliability = weighted_reliability / total_vehicles if
total_vehicles > 0 else 0
    avg_speed = weighted_speed / total_vehicles if total_vehicles > 0 else
0

    # Total cost (purchase + 3 years operating)
    total_cost = total_purchase + (total_annual_operating * 3)

    # Service score components
    capacity_score = min(1.0, total_capacity / REQUIRED_CAPACITY)  # Caps
at 1.0
    reliability_score = avg_reliability
    speed_score = avg_speed / 70  # Normalize to fastest vehicle
```

```
    # Combined service score (weighted average)
    service_score = 0.5 * capacity_score + 0.3 * reliability_score + 0.2 *
speed_score

    return {
        'total_cost': total_cost,
        'service_score': service_score,
        'avg_co2': avg_co2,
        'total_capacity': total_capacity,
        'total_vehicles': total_vehicles,
        'capacity_score': capacity_score,
        'reliability_score': reliability_score,
        'speed_score': speed_score
    }
```

Example Fleet Evaluation

```
# Example: A balanced fleet
example_fleet = {
    'Electric Truck': 15,
    'Hybrid Van': 25,
    'Diesel Van': 10,
    'E-Cargo Bike': 20,
    'Autonomous Pod': 5
}

metrics = calculate_fleet_metrics(example_fleet)

print("\nExample Fleet Performance:")
print(f"  Total Cost: €{metrics['total_cost']:,.0f}")
print(f"  Service Score: {metrics['service_score']:.3f}")
print(f"  Average CO2: {metrics['avg_co2']:.1f} g/km")
print(f"  Total Capacity: {metrics['total_capacity']:,} parcels/day")
print(f"  Total Vehicles: {metrics['total_vehicles']}")
print(f"\n  EU Compliant: {'✓ YES' if metrics['avg_co2'] <= 111 else '✗
NO'}")
print(f"  Sufficient Capacity: {'✓ YES' if metrics['total_capacity'] >=
22000 else '✗ NO'}")
```

```
Example Fleet Performance:
  Total Cost: €21,912,600
  Service Score: 0.746
  Average CO2: 56.3 g/km
  Total Capacity: 13,750 parcels/day
  Total Vehicles: 75

  EU Compliant: ✓ YES
  Sufficient Capacity: ✗ NO
```

⚠ Capacity Matters!

Your fleet should have enough capacity to handle 22,000 daily parcels. The service score includes a capacity component that penalizes fleets that can't meet demand.

Capacity score = min(1.0, total_capacity / 22,000)

- If capacity = 22,000 → capacity_score = 1.0 (perfect)
- If capacity = 11,000 → capacity_score = 0.5 (only 50%!)

Your Task

Design the optimal fleet composition balancing:

1. Minimize Total Cost (purchase + annual operating costs)
2. Maximize Service Score (capacity + reliability + speed)
3. Hard Constraint: Average CO2 emissions ≤ 111 g/km

Step 1: Initial Solution

Create a fleet composition by your reasoning that:

1. Meets the EU emission constraint (avg CO2 ≤ 111 g/km)
2. Has sufficient capacity (≥ 22,000 parcels/day would be great if achievable)
3. Balances cost vs service score according to YOUR strategic priorities

```
# YOUR FLEET COMPOSITION HERE

my_fleet = {
    'Electric Truck': 0,      # YOUR CHOICE
    'Hybrid Van': 0,          # YOUR CHOICE
    'Diesel Van': 0,          # YOUR CHOICE
    'E-Cargo Bike': 0,        # YOUR CHOICE
    'Autonomous Pod': 0       # YOUR CHOICE
}

# Evaluate your fleet
my_metrics = calculate_fleet_metrics(my_fleet)

print("\nYour Fleet Performance:")
print(f"  Total Cost: €{my_metrics['total_cost']:,.0f}")
print(f"  Service Score: {my_metrics['service_score']:.3f}")
print(f"  Average CO2: {my_metrics['avg_co2']:.1f} g/km")
print(f"  Total Capacity: {my_metrics['total_capacity']:,} parcels/day")
print(f"  Total Vehicles: {my_metrics['total_vehicles']}")
print(f"\n  EU Compliant: {'✓ YES' if my_metrics['avg_co2'] <= 111 else '✗
NO - INVALID!'}")
print(f"  Sufficient Capacity: {'✓ YES' if my_metrics['total_capacity'] >=
22000 else '⚠ LOW'}")
```

Step 2: Pareto Analysis

Generate alternative fleet compositions and find the Pareto frontier.

> Three Approaches (Choose at Least One)
>
> Approach A: Sequential Greedy
>
> - Prioritize one objective (e.g., minimize cost first)
> - Then optimize the other within acceptable range
>
> Approach B: Weighted Greedy
>
> - Generate solutions with different weight combinations
> - Example: 80% cost / 20% service, then 50/50, then 20/80
>
> Approach C: Random + Filter
>
> - Generate 100+ random fleet compositions
> - Filter for feasibility (CO2 ≤ 111)
> - Find Pareto frontier among feasible solutions

```
# GENERATE ALTERNATIVES
# Use one of the three approaches above!

alternatives = []

# Example: Random generation (you should implement your chosen approach)
for i in range(100):
    # Generate random fleet
    random_fleet = {
        'Electric Truck': np.random.randint(0, 35),
        'Hybrid Van': np.random.randint(0, 30),
        'Diesel Van': np.random.randint(0, 25),
        'E-Cargo Bike': np.random.randint(0, 40),
        'Autonomous Pod': np.random.randint(0, 15)
    }

    metrics = calculate_fleet_metrics(random_fleet)

    # Only keep feasible solutions
    if metrics['avg_co2'] <= 111:  # Hard constraint
        alternatives.append({
            'fleet': random_fleet,
            'cost': metrics['total_cost'],
            'service': metrics['service_score'],
            'co2': metrics['avg_co2']
        })

print(f"\nGenerated {len(alternatives)} feasible fleet alternatives")
```

```
# YOUR SOLUTION TO DETERMINE THE PARETO FRONTIER HERE
```

Step 3: Visualization & Recommendation

```
# YOUR SOLUTION HERE
```

Step 4: Create Your Submission

Prepare a one-slide presentation (PDF) containing:

- Your Best Fleet: Total cost and service score (prominently displayed)
- Approach: Which solution generation method(s) did you use?
- Visualization: Scatter plot showing Pareto frontier with your solution marked
- Fleet Composition: Table showing vehicle types and quantities
- Strategy Justification: 2-3 sentences explaining why your fleet is optimal

Tips for Success

Strategy Suggestions

1. Start simple: Try random generation + filtering first to understand the solution space
2. Visualize early: Plot your alternatives to see if you're exploring well
3. Check feasibility: Always verify CO2 ≤ 111 before adding to alternatives
4. Capacity matters: Don't sacrifice too much capacity for cost savings
5. Pareto insight: If your solution isn't on the frontier, ask why
6. Be realistic: Explain trade-offs honestly as there's likely no perfect solution

Common Pitfalls to Avoid

- Forgetting the CO2 constraint: Filter for feasibility BEFORE evaluating
- Ignoring capacity: Low capacity = low service score, even if cheap
- Incorrect dominance check: Must compare BOTH cost AND service score
- Poor visualization: Make sure your chosen solution is clearly visible

Final Checklist

Before submitting your solution, verify:

- ☐ Fleet meets CO2 constraint (≤ 111 g/km)
- ☐ Fleet has reasonable capacity (≥ 22,000 parcels/day recommended)
- ☐ Correctly implemented Pareto frontier filtering
- ☐ Justification is clear and concise (2-3 sentences)
- ☐ One-slide presentation ready with visualization and metrics

Good Luck!

Bibliography