transport

December 12, 2023

A programação linear é uma disciplina essencial para a área de pesquisa operacional, por desempenhar um papel muito importante na otimização da alocação de recursos, incluindo o transporte público. Envolve a criação de modelos para maximizar ou minimizar funções, sujeitos a restrições que representam as limitações do problema. O problema de transporte, uma forma especializada de programação linear, pode ser usado para otimizar a alocação de recursos em rotas de transporte público.

Nesse contexto, o problema de transporte visa alocar veículos em rotas de forma eficiente para atender à demanda dos passageiros, reduzindo custos operacionais. Isso envolve considerar parâmetros como as rotas disponíveis, suas capacidades, a demanda nos bairros e os custos operacionais de cada rota para otimizar a alocação de recursos públicos. Desta forma, a programação linear se mostra uma ferramenta indispensável para melhoria de processos e serviços, em especial no transporte público.

Os parâmetros de entrada são essenciais para encontrar a melhor solução: as rotas representam caminhos para o transporte público, cada uma com sua capacidade máxima de passageiros. Os bairros são as áreas atendidas, com demandas específicas. Custos operacionais variam e influenciam as decisões na alocação de rotas.

O problema de transporte busca equilibrar recursos para atender às demandas dos bairros, sem exceder as capacidades das rotas, minimizando custos. Isso é crucial no planejamento de transporte público, onde a meta é fornecer um serviço eficiente dentro de orçamentos e recursos. O problema de transporte, ao otimizar a alocação de recursos com base nesses parâmetros, é uma ferramenta indispensável para esse propósito.

Pode ocorrer, no entanto, que a demanda de passageiros exceda a capacidade das rotas. Nesse caso, o problema de transporte não tem solução se não houver rotas adicionais disponíveis. Isso pode ser resolvido com a criação de rotas adicionais (no contexto da solução, rotas artificiais), mas isso aumenta os custos operacionais. Neste trabalho, apenas é lidado com a capacidade existente, sem criar rotas artificiais. Isso significa que sempre haverá uma solução, mesmo que não seja a melhor possível.

Quando a situação contrária ocorre, ou seja, quando a capacidade das rotas excede a demanda, a solução se concentra na minimização dos custos operacionais, e "ignora" a capacidade excedente. Isso significa que a solução pode não ser a melhor possível, mas é a melhor possível dentro dos parâmetros fornecidos.

Desta forma, tem-se o seguinte problema de transporte:

- Rotas disponíveis: "Linha Central" "Rota do Parque" e "Via Expressa" Capacidades correspondentes: 1250, 1550, e 1400 passageiros, respectivamente, como o número máximo de passageiros que podem ser atendidos por dia em cada rota.

- Bairros atendidos: “Centro,” “Parque das Flores,” “Vila Nova,” e “Jardim Oceânico.” Demandas correspondentes: 1200, 950, 750, e 1300 passageiros, respectivamente, representando o número de passageiros por dia em cada bairro.
- Custos operacionais por passageiro para cada rota e bairro (estimativa):
  - “Linha Central” tem custos de R$3,50 para o bairro “Centro,” R$4,00 para o “Parque das Flores,” R$3,75 para o “Vila Nova,” e R$4,50 para o “Jardim Oceânico.”
  - “Rota do Parque” tem custos de R$3,00 para o bairro “Centro,” R$3,50 para o “Parque das Flores,” R$4,00 para o “Vila Nova,” e R$3,60 para o “Jardim Oceânico.”
  - “Via Expressa” tem custos de R$3,20 para o bairro “Centro,” R$4,20 para o “Parque das Flores,” R$3,90 para o “Vila Nova,” e R$4,10 para o “Jardim Oceânico.”

Rota	Centro (R)|*ParquedasFlores*(R)|*VilaNova*(R)|*JardimOcenico*(R)			
Linha Central	3,50	4,00	3,75	4,50
Rota do Parque	3,00	3,50	4,00	3,60
Via Expressa	3,20	4,20	3,90	4,10

Esta tabela apresenta os custos operacionais estimados por passageiro para cada rota e bairro em reais (R$).

[ ]:
```python
import pandas as pd
import numpy as np
from typing import List

HIGH_COST = 1e12  # Define um custo alto para ser usado como padrão

class TransportPlanner:
    def __init__(self, routes: List[str], neighborhoods: List[str], demands:␣
 ↪List[int], capacities: List[int], costs: List[List[int]]):
        self.routes = np.array(routes)  # Rotas disponíveis
        self.neighborhoods = np.array(neighborhoods)  # Bairros atendidos
        self.demands = np.array(demands)  # Demanda de cada bairro
        self.capacities = np.array(capacities)  # Capacidade de cada rota
        self.costs = np.array(costs)  # Custo por unidade para cada rota e␣
 ↪bairro
        self.total_costs = []  # Lista para armazenar os custos totais

    def allocate_buses(self) -> np.ndarray:
        allocation = np.zeros(self.costs.shape, dtype=int)  # Inicializa matriz␣
 ↪de alocação
        remaining_demand = self.demands.copy()  # Copia a demanda

        # Itera sobre as rotas ordenadas por eficiência de custo
        for route_idx, neighborhood_idx in self.
 ↪get_sorted_routes_by_cost_effectiveness():
```

```
            # Aloca ônibus para o bairro
            self.allocate_to_neighborhood(route_idx, neighborhood_idx,␣
↪allocation, remaining_demand)

            # Verifica se toda a demanda foi atendida
            if self.is_all_demand_met(remaining_demand):
                break

        return allocation

    def get_sorted_routes_by_cost_effectiveness(self) -> np.ndarray:
        # Método para obter rotas ordenadas pela eficiência de custo
        cost_per_unit = np.where(self.demands == 0, HIGH_COST, self.costs /␣
↪self.demands)  # Calcula custo por unidade
        return np.dstack(np.unravel_index(np.argsort(cost_per_unit.ravel()),␣
↪cost_per_unit.shape))[0]

    def allocate_to_neighborhood(self, route_idx: int, neighborhood_idx: int,␣
↪allocation: np.ndarray, remaining_demand: np.ndarray) -> None:
        if remaining_demand[neighborhood_idx] > 0:
            available_capacity = self.calculate_available_capacity(route_idx,␣
↪allocation)  # Calcula capacidade disponível
            assigned_capacity = min(available_capacity,␣
↪remaining_demand[neighborhood_idx])  # Determina a capacidade a ser alocada
            allocation[route_idx][neighborhood_idx] += assigned_capacity  #␣
↪Atualiza a alocação
            remaining_demand[neighborhood_idx] -= assigned_capacity  # Atualiza␣
↪a demanda restante
            self.record_cost(route_idx, neighborhood_idx, assigned_capacity)  #␣
↪Registra o custo

    def calculate_available_capacity(self, route_idx: int, allocation: np.
↪ndarray) -> int:
        return self.capacities[route_idx] - np.sum(allocation[route_idx])

    def record_cost(self, route_idx, neighborhood_idx, assigned_capacity) ->␣
↪None:
        cost = assigned_capacity * self.costs[route_idx][neighborhood_idx]
        self.total_costs.append(cost)

    def is_all_demand_met(self, remaining_demand) -> bool:
        return np.all(remaining_demand == 0)

    def solve(self) -> (pd.DataFrame, List[int]):
        # Propriedade para obter o resultado da alocação
        allocation_result = self.allocate_buses()
```

```
        df = pd.DataFrame(allocation_result, index=self.routes, columns=self.
 ↪neighborhoods)
        return df, self.total_costs


routes = ["Linha Central", "Rota do Parque", "Via Expressa"]
capacities = [1250, 1550, 1400] # Número máximo de passageiros por dia, para␣
 ↪cada rota

neighborhoods = ["Centro", "Parque das Flores", "Vila Nova", "Jardim Oceânico"]
demands = [1200, 950, 750, 1300] # Número de passageiros por dia, para cada␣
 ↪bairro
# Custo operacional por passageiro em cada rota e bairro (estimativa)
# Unidade: Custo em reais por passageiro
costs = [
    [3.50, 4.00, 3.75, 4.50],  # Custos para "Linha Central"
    [3.00, 3.50, 4.00, 3.60],  # Custos para "Rota do Parque"
    [3.20, 4.20, 3.90, 4.10]   # Custos para "Via Expressa"
]

planner = TransportPlanner(routes, neighborhoods, demands, capacities, costs)
df, costs = planner.solve()
print(f"Custos por rota: {costs}, custo total: {sum(costs)}")
df
```

```
Custos por rota: [3600.0, 1260.0, 3894.9999999999995, 0.0, 3800.0, 1125.0,
1755.0], custo total: 15435.0
```

[ ]:

	Centro	Parque das Flores	Vila Nova	Jardim Oceânico
Linha Central	0	950	300	0
Rota do Parque	1200	0	0	350
Via Expressa	0	0	450	950