

# experimento 8: claude não rodou - a3b80778d

## USER

A base de dados anexada possui quatro variáveis: IDENTIFICADOR, CARGO, PERCENT\_PRI E DISTANCE\_KM. A primeira variável é o código que identifica o funcionário, a segunda apresenta os códigos dos cargos, a terceira é um índice que varia de 0 a 1, sendo o 0 o melhor e, por fim, a última variável é a distância em KM entre o funcionário e o cargo que ele poderá assumir, ou seja, quanto maior a distância, pior. Gostaria que me propusesse uma metodologia para solucionar meu problema de alocação de forma a minimizar o PERCENT\_PRI ou a DISTANCE\_KM ou os dois juntos.

## BOT

Vou ajudar você a desenvolver uma metodologia para resolver este problema de alocação otimizada. Vamos pensar passo a passo:

### 1. \*\*Análise do Problema\*\*

- Temos um problema de alocação de funcionários a cargos
- Cada par funcionário-cargo tem dois critérios de avaliação:
  - \* PERCENT\_PRI (quanto menor, melhor)
  - \* DISTANCE\_KM (quanto menor, melhor)
- Precisamos minimizar um ou ambos os critérios

### 2. \*\*Possíveis Abordagens\*\*

#### a) \*\*Otimização de objetivo único\*\*:

- Minimizar apenas PERCENT\_PRI
- Minimizar apenas DISTANCE\_KM
- Criar uma função combinada (ex:  $\text{peso\_percent} * \text{PERCENT\_PRI} + \text{peso\_dist} * \text{DISTANCE\_KM}$  normalizada)

#### b) \*\*Otimização multiobjetivo\*\*:

- Usar algoritmos de otimização multiobjetivo como NSGA-II
- Encontrar a fronteira de Pareto das soluções

### 3. \*\*Metodologia Proposta\*\*

Sugiro uma abordagem em três etapas:

#### \*\*Etapa 1: Normalização\*\*

```
```python
# Normalizar DISTANCE_KM para escala 0-1
distance_norm = (DISTANCE_KM - min(DISTANCE_KM)) / (max(DISTANCE_KM) - min(DISTANCE_KM))
```
```

#### \*\*Etapa 2: Função Objetivo Combinada\*\*

```
```python
# Criar score combinado
score = w1 * PERCENT_PRI + w2 * distance_norm
# onde w1 + w2 = 1 (ex: w1 = 0.5 e w2 = 0.5)
```
```

#### \*\*Etapa 3: Algoritmo de Otimização\*\*

- Usar Programação Linear Inteira:
  - \* Variável de decisão:  $X_{ij}$  (1 se funcionário  $i$  é alocado ao cargo  $j$ , 0 caso contrário)
  - \* Restrições:
    - \* Cada funcionário deve ser alocado a exatamente um cargo
    - \* Cada cargo deve receber exatamente um funcionário
  - \* Objetivo: Minimizar a soma dos scores

### 4. \*\*Implementação Sugerida\*\*

```
```python
from pulp import *
```

```
# Criar problema
prob = LpProblem("Alocacao_Funcionarios", LpMinimize)

# Criar variáveis de decisão
func_cargos = [(i,j) for i in funcionarios for j in cargos]
x = LpVariable.dicts("alocacao", func_cargos, 0, 1, LpBinary)

# Função objetivo
prob += lpSum([score[i][j] * x[i,j] for (i,j) in func_cargos])

# Restrições
for i in funcionarios:
    prob += lpSum([x[i,j] for j in cargos]) == 1

for j in cargos:
    prob += lpSum([x[i,j] for i in funcionarios]) == 1
...

5. **Vantagens desta Abordagem**
- Flexibilidade para ajustar pesos entre PERCENT_PRI e DISTANCE_KM
- Garantia de solução ótima global
- Facilidade de implementação usando bibliotecas existentes
- Possibilidade de adicionar restrições adicionais

6. **Alternativa Multiobjetivo**
Se precisar considerar os objetivos separadamente:
- Usar NSGA-II para encontrar conjunto de soluções Pareto-ótimas
- Permitir ao tomador de decisão escolher entre diferentes trade-offs

7. **Recomendação Final**
Comece com a abordagem de objetivo único combinado, pois:
- É mais simples de implementar
- Fornece uma única solução clara
- Permite ajuste fino através dos pesos
- Pode ser expandida para multiobjetivo se necessário
```

Você gostaria que eu detalhasse algum destes aspectos ou que desenvolvesse um exemplo prático de implementação?

## USER

execute o código

## BOT

Vou implementar a solução usando Python com PuLP para resolver o problema de otimização. Primeiro, vamos preparar os dados e depois criar o modelo de programação linear inteira.

```
```python
import pandas as pd
import pulp as pl
import numpy as np

# Ler os dados e converter as vírgulas para pontos nos números decimais
df = pd.read_csv('Result:', sep=',', decimal=',')

# Obter listas únicas de funcionários e cargos
funcionarios = df['IDENTIFICADOR'].unique()
cargos = df['CARGO'].unique()

# Criar o problema de minimização
prob = pl.LpProblem("Alocacao_Funcionarios", pl.LpMinimize)

# Normalizar DISTANCE_KM para escala 0-1
df['DISTANCE_NORM'] = (df['DISTANCE_KM'] - df['DISTANCE_KM'].min()) / (df['DISTANCE_KM'].max() -
```

```

df['DISTANCE_KM'].min())

# Criar score combinado (peso igual para ambos os critérios)
w1, w2 = 0.5, 0.5
df['SCORE'] = w1 * df['PERCENT_PRI'] + w2 * df['DISTANCE_NORM']

# Criar variáveis de decisão
x = pl.LpVariable.dicts("alocacao",
                        ((i, j) for i in funcionarios for j in cargos),
                        cat='Binary')

# Função objetivo: minimizar a soma dos scores
prob += pl.lpSum(df.loc[df['IDENTIFICADOR'] == i].loc[df['CARGO'] == j, 'SCORE'].iloc[0] * x[i,j]
                 for i in funcionarios for j in cargos)

# Restrição: cada funcionário deve ser alocado a exatamente um cargo
for i in funcionarios:
    prob += pl.lpSum(x[i,j] for j in cargos) == 1

# Restrição: cada cargo deve receber exatamente um funcionário
for j in cargos:
    prob += pl.lpSum(x[i,j] for i in funcionarios) == 1

# Resolver o problema
prob.solve()

# Imprimir status da solução
print(f"Status: {pl.LpStatus[prob.status]}")
print(f"Valor objetivo: {pl.value(prob.objective):.4f}")

# Criar lista de alocações
alocacoes = []
for i in funcionarios:
    for j in cargos:
        if pl.value(x[i,j]) == 1:
            score = df.loc[df['IDENTIFICADOR'] == i].loc[df['CARGO'] == j, 'SCORE'].iloc[0]
            percent_pri = df.loc[df['IDENTIFICADOR'] == i].loc[df['CARGO'] == j, 'PERCENT_PRI'].iloc[0]
            distance = df.loc[df['IDENTIFICADOR'] == i].loc[df['CARGO'] == j, 'DISTANCE_KM'].iloc[0]
            alocacoes.append({
                'Funcionario': i,
                'Cargo': j,
                'Score': score,
                'PERCENT_PRI': percent_pri,
                'DISTANCE_KM': distance
            })

# Criar DataFrame com resultados
resultados = pd.DataFrame(alocacoes)
resultados = resultados.sort_values('Funcionario')

# Imprimir resultados
print("\nAlocações ótimas:")
print(resultados.to_string(index=False))

# Calcular métricas agregadas
print("\nMétricas agregadas:")
print(f"Média PERCENT_PRI: {resultados['PERCENT_PRI'].mean():.4f}")
print(f"Média DISTANCE_KM: {resultados['DISTANCE_KM'].mean():.4f}")
print(f"Média Score: {resultados['Score'].mean():.4f}")
'''

```

Este código:

1. Lê os dados e prepara o formato adequado
2. Normaliza a distância para escala 0-1
3. Cria um score combinado com pesos iguais para PERCENT\_PRI e distância normalizada
4. Define e resolve o problema de programação linear inteira
5. Apresenta os resultados com as alocações ótimas e métricas agregadas

O resultado mostrará:

- A alocação ótima para cada funcionário
- O score final de cada alocação
- Os valores de PERCENT\_PRI e DISTANCE\_KM para cada alocação
- Métricas agregadas do resultado final

Você gostaria que eu ajustasse algum parâmetro específico ou que apresentasse os resultados de outra forma?

Execution Error: FileNotFoundError

Traceback (most recent call last)

Cell In[3], line 6

```
3 import numpy as np
5 # Ler os dados e converter as vírgulas para pontos nos números decimais
----> 6 df = pd.read_csv('Result.', sep=';', decimal=',')
      8 # Obter listas únicas de funcionários e cargos
      9 funcionarios = df['IDENTIFICADOR'].unique()
```

...

FileNotFoundError: [Errno 2] No such file or directory: 'Result.'