Problema do Carteiro Chinês

Otimização de Rotas na Teoria dos Grafos

Um problema clássico de encontrar o caminho de menor custo que percorre todas as arestas de um grafo pelo menos uma vez, retornando ao vértice inicial.

Introdução

Origem: Formulado pelo matemático chinês Mei-Ko Kwan em 1962.

Inspiração: Situação de um carteiro que precisa entregar correspondências em todas as ruas de uma cidade, percorrendo a menor distância possível.

Aplicações Práticas:

• Coleta de lixo

• Inspeção de estradas

• Remoção de neve

• Entrega de correspondências

Definição do Problema

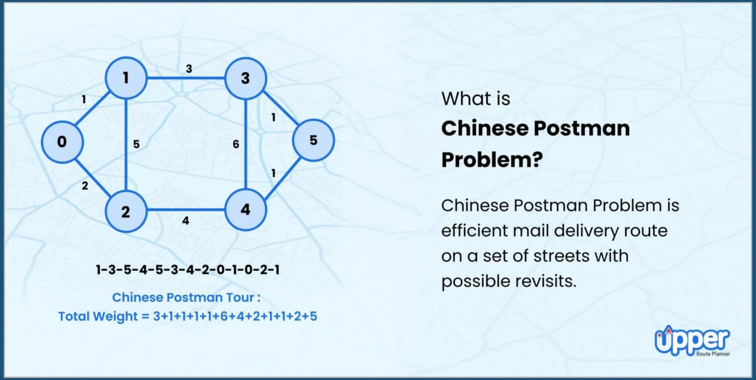
Objetivo: Encontrar um circuito fechado de peso mínimo que percorra todas as arestas de um grafo pelo menos uma vez, retornando ao vértice inicial.

Grafo: Conexo e ponderado $G = (V, E)$

- V: Conjunto de vértices (cruzamentos)
- E: Conjunto de arestas (ruas)
- $w(e)$: Peso de cada aresta (distância/custo)

$$\text{Minimizar: } \sum_{e \in E} w(e) \cdot x(e)$$

Onde $x(e)$ é o número de vezes que a aresta e é percorrida



Classificação:

Grafos não-direcionados: Problema polinomial

Grafos direcionados: Problema polinomial

Grafos mistos: Problema NP-difícil

Conceitos da Teoria dos Grafos

Grafo Euleriano

Um grafo é euleriano se e somente se é conexo e todos os seus vértices têm grau **par** .

Circuito Euleriano

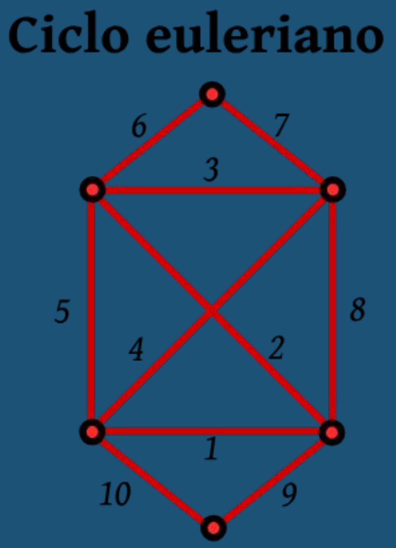
Um caminho que percorre cada aresta do grafo **exatamente uma vez** , retornando ao vértice inicial.

Vértice de Grau Ímpar

Um vértice que possui um número **ímpar** de arestas incidentes.

Teorema de Euler

Um grafo conexo possui um circuito euleriano se e somente se todos os seus vértices têm grau par.



Relação com o Problema do Carteiro Chinês

A solução do Problema do Carteiro Chinês está diretamente relacionada com a existência de **circuitos eulerianos** no grafo.

Grafo Euleriano

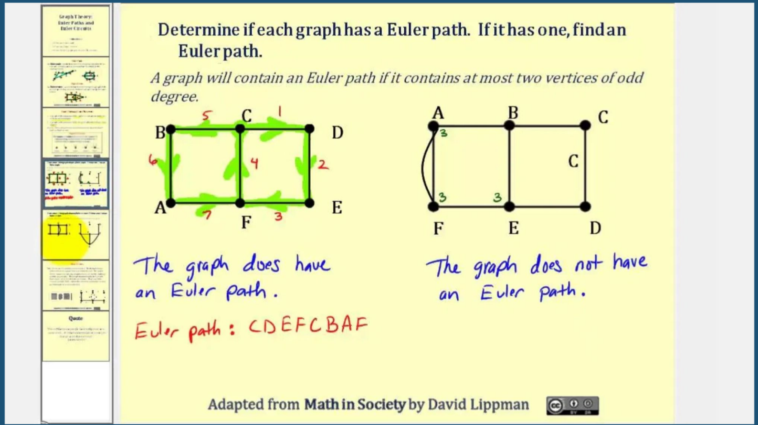
Se o grafo é euleriano (todos os vértices têm grau par), a solução é o próprio circuito euleriano.

Custo mínimo = soma dos pesos das arestas originais.

Grafo Não-Euleriano

Se o grafo não é euleriano (possui vértices de grau ímpar), é necessário duplicar algumas arestas para torná-lo euleriano.

Custo mínimo = soma dos pesos das arestas originais + custo das arestas duplicadas.



Propriedade importante: O número de vértices de grau ímpar em qualquer grafo é sempre par.

Esta propriedade é fundamental para a solução do problema, pois permite o emparelhamento dos vértices de grau ímpar.

Algoritmo de Solução

Passos para Resolver o Problema:

1. Verificar se o grafo é euleriano: Se todos os vértices têm grau par, o grafo já é euleriano.
2. Identificar os vértices de grau ímpar: Se existem vértices de grau ímpar, identificá-los.
3. Encontrar o emparelhamento de peso mínimo: Criar um grafo completo com os vértices de grau ímpar e encontrar o emparelhamento de peso mínimo.
4. Adicionar arestas duplicadas: Para cada par de vértices no emparelhamento, adicionar arestas duplicadas ao longo do caminho mais curto entre eles.
5. Encontrar um circuito euleriano: Usar o algoritmo de Hierholzer para encontrar um circuito euleriano no grafo modificado.

Algoritmo de Hierholzer

- Escolher um vértice inicial arbitrário
- Seguir um caminho de arestas, removendo cada aresta percorrida
- Quando retornar ao vértice inicial, inserir subcircuitos em vértices com arestas não percorridas

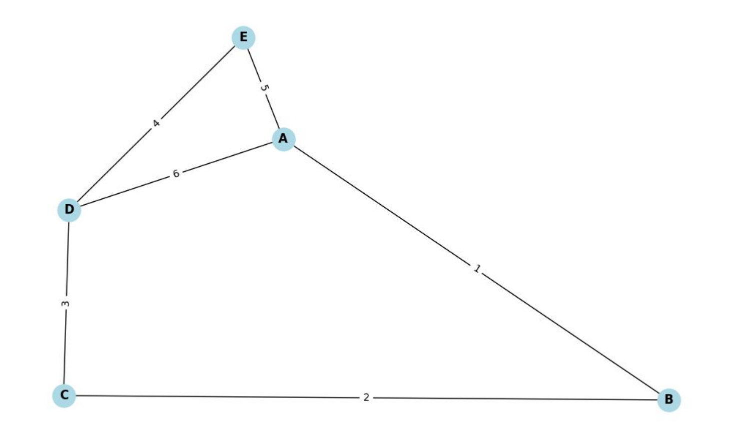
Exemplo Prático

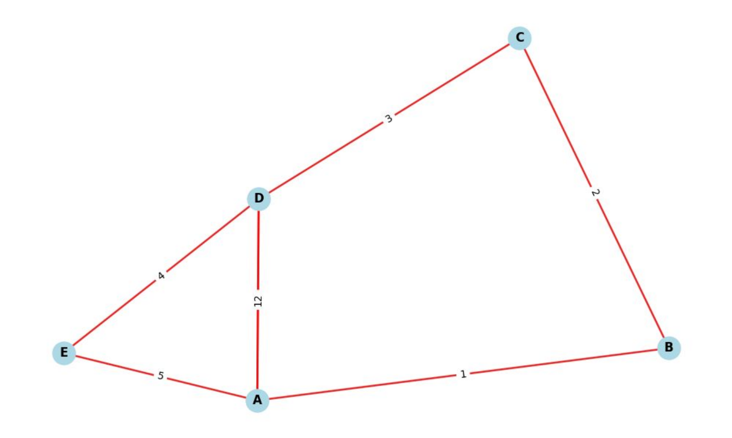
Aplicação do Algoritmo:

1. Análise do grafo original: 5 vértices (A, B, C, D, E) e 6 arestas.
2. Identificação dos vértices de grau ímpar: A, D têm grau ímpar.
3. Emparelhamento de peso mínimo: O par (A,D) forma o emparelhamento de peso mínimo.
4. Adição de arestas duplicadas: Duplicamos a aresta A-D para tornar o grafo euleriano.

Custo das arestas originais:		21
Custo das arestas duplicadas:	6 (A-D: 6)	
Custo total da rota:		**27**

O grafo euleriano resultante possui todas as arestas originais mais as arestas duplicadas A-D.





Implementação em Python

Principais Funções:

find_odd_degree_vertices
: Encontra os vértices de grau ímpar no grafo.

add_minimum_weight_matching
: Adiciona arestas para tornar o grafo euleriano.

find_eulerian_circuit
: Encontra um circuito euleriano no grafo.

solve_chinese_postman
: Função principal que resolve o problema.

```python
import networkx as nx
import matplotlib.pyplot as plt
from itertools import combinations

def find_odd_degree_vertices(graph):
    odd_degree_vertices = []
    for node in graph.nodes():
        if graph.degree(node) % 2 != 0:
            odd_degree_vertices.append(node)
    return odd_degree_vertices

def add_minimum_weight_matching(graph):
    odd_vertices = find_odd_degree_vertices(graph)
    if not odd_vertices:
        return graph.copy(), 0
    complete_graph = nx.Graph()
    for u, v in combinations(odd_vertices, 2):
        if nx.has_path(graph, u, v):
            path_length = nx.shortest_path_length(graph, u, v, weight='weight')
            complete_graph.add_edge(u, v, weight=path_length)
    matching = nx.algorithms.matching.min_weight_matching(complete_graph)
    eulerian_graph = nx.MultiGraph()
    for u, v, data in graph.edges(data=True):
        eulerian_graph.add_edge(u, v, **data)
    added_weight = 0
    for u, v in matching:
        path = nx.shortest_path(graph, u, v, weight='weight')
        for i in range(len(path) - 1):
            node1, node2 = path[i], path[i + 1]
            weight = graph[node1][node2]['weight']
            eulerian_graph.add_edge(node1, node2, weight=weight)
            added_weight += weight
    return eulerian_graph, added_weight

def find_eulerian_circuit(graph, start_node=None):
    if not nx.is_eulerian(graph):
        raise nx.NetworkXError("O grafo não é euleriano. Verifique a conectividade e os graus dos vértices.")
    if start_node is None:
        start_node = list(graph.nodes())[0]
    return list(nx.eulerian_circuit(graph, source=start_node))

def solve_chinese_postman(graph, start_node=None):
    original_cost = sum(graph[u][v]['weight'] for u, v in graph.edges())
    eulerian_graph, added_cost = add_minimum_weight_matching(graph)
    circuit = find_eulerian_circuit(eulerian_graph, start_node)
    formatted_circuit = []
    for u, v in circuit:
        weight = eulerian_graph[u][v][0]['weight'] if isinstance(eulerian_graph, nx.MultiGraph) else eulerian_graph[u][v]['weight']
        formatted_circuit.append((u, v, {'weight': weight}))
    total_cost = original_cost + added_cost

    return {
        'total_cost': total_cost,
        'circuit': formatted_circuit
    }
```

Bibliotecas Utilizadas:

- **NetworkX:** Para manipulação de grafos e algoritmos de grafos
- **Matplotlib:** Para visualização dos grafos e circuitos
- **Itertools:** Para combinações de vértices

Complexidade: $O(n^3)$, onde n é o número de vértices de grau ímpar.

A complexidade é dominada pelo algoritmo de emparelhamento de peso mínimo.

Aplicações Práticas

O Problema do Carteiro Chinês tem diversas aplicações no mundo real:

Coleta de Lixo

Otimização de rotas para caminhões de coleta de lixo, que precisam percorrer todas as ruas de uma cidade, minimizando distâncias e custos operacionais.

Remoção de Neve

Otimização de rotas para limpar neve das ruas de uma cidade, minimizando o tempo e o combustível gastos pelos veículos de limpeza.

Inspeção de Estradas

Planejamento de rotas para veículos de inspeção que precisam verificar todas as estradas de uma rede viária, otimizando tempo e recursos.

Entrega de Correspondências

Planejamento de rotas para carteiros que precisam entregar correspondências em todas as ruas de um bairro, minimizando a distância percorrida.

Conclusão

Principais Pontos:

- O Problema do Carteiro Chinês é um problema clássico da teoria dos grafos que busca encontrar o caminho de menor custo que percorre todas as arestas de um grafo pelo menos uma vez.
- A solução envolve a identificação de vértices de grau ímpar e a adição de arestas duplicadas para tornar o grafo euleriano.
- O emparelhamento de peso mínimo entre os vértices de grau ímpar é a etapa crucial para minimizar o custo total da rota.
- A implementação em Python utilizando a biblioteca NetworkX permite resolver o problema de forma eficiente.

Considerações Finais

O Problema do Carteiro Chinês é um exemplo fascinante de como a teoria dos grafos pode ser aplicada para resolver problemas práticos de otimização de rotas, demonstrando a importância da matemática discreta em aplicações do mundo real.

Referências

Artigos e Livros

Kwan, M.-K. (1962). "Graphic programming using odd or even points". Chinese Mathematics. 1: 273–277.

Edmonds, J.; Johnson, E. L. (1973). "Matching, Euler tours and the Chinese postman". Mathematical Programming. 5: 88–124.

Hierholzer, C. (1873). "Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren". Mathematische Annalen. 6(1): 30–32.

Christofides, N. (1973). "The optimum traversal of a graph". Omega. 1(6): 719–732.

Thimbleby, H. (1981). "The directed Chinese Postman Problem". Software: Practice and Experience. 13(6): 1281–1292.

Recursos Online

NetworkX Documentation. "Matching".
https://networkx.org/documentation/stable/reference/algorithms/matching.html

GeeksforGeeks. "Chinese Postman or Route Inspection".
https://www.geeksforgeeks.org/chinese-postman-route-inspection-set-1-introduction/

IME-USP. "Problema do Carteiro Chinês".
https://www.linux.ime.usp.br/~gafeol/mac5711/chinese-postman/tex/main.pdf

Seminário. "Problema do Carteiro Chinês".
Baseado no documento fornecido: SeminarioProblemaCarteiroChines.pdf