Caixeiro Viajante

Cristiano Campos e Jansen Silva

Universidade Federal Fluminense

July 4, 2019

TSP

O Problema do caixeiro viajante (TSP) consiste em responder a seguinte pergunta: "Dada uma lista de cidades e as distâncias entre cada par de cidades, qual é a rota mais curta possível que visita cada cidade e retorna à cidade de origem?" É um problema NP-difícil na otimização combinatória , importante na pesquisa operacional e na informática teórica .

Histórico

O problema foi matematicamente formulado em 1800 pelo matemático irlandês WR Hamilton e pelo matemático britânico Thomas Kirkman.

Na década de 1930 Merrill M. Flood, introduziu o nome do problema do caixeiro viajante quando estava procurando resolver um problema de roteamento de ônibus escolar na Universidade de Princeton.

Nas décadas de 1950 e 1960, o problema tornou-se cada vez mais popular nos círculos científicos da Europa e dos EUA, depois que a RAND Corporation, ofereceu prêmios por etapas na solução do problema.

Trabalhos clássicos para solução exata do TSP

Ano	Pesquisador	Trabalho
1984	Dantzig et al.	Trabalho de referencia para o PCV
1973	Laporte e Nobert	Métodos Exatos
1980	Crowder e Padberg	B&B
1981	Balas e Christofides	Restrições lagrangeanas para o PCV
1985	Fleiscmann	Algoritmo com uso de plano de cortes
1995	Junger et al.	Relações e B&Cut
1998	Applegate et al.	B&Cut
2001	Applegate et al.	Cortes

Quadro 1: Trabalhos com solução exata para o PCV
Fonte: Goldbarg et al. (2005)

Aplicações

O TSP tem várias aplicações, como planejamento , logística e fabricação de microchips . Aparece como um subproblema em muitas áreas, como o sequenciamento de DNA.

Citação aplicações para o TSP:

- Manipulação de itens em estoque (RATLIFF,ROSENTHAL, 1981).
- Programação de operações de máquinas em manufaturas (KUSIAK E FINKE, 1987)
- Otimização do movimento de ferramentas de corte (CHAUNY et al. , 1987)
- Otimização de perfuração de furo em placas de circuitos impressos (REINELT, 1989).

Problemas de Otimização Combinatória

Definição:

Problemas de Otimização Combinatória são aqueles cuja resolução se deve a otimização (maximização ou minimização) de uma ou várias funções objetivo. Devem ser satisfeitas diversas restrições definidas sobre suas variáveis.

Exemplos:

Existem vários problemas que surgem na Ciência da Computação que se enquadram nesta classe, tais como:

- Coloração de grafos;
- Determinação da árvore geradora de peso mínimo de um grafo;
- Fluxo em redes;
- Problema da mochila (Knapsack Problem);
- Problema do caixeiro viajante (Traveling Salesman Problem-TSP);
- Determinação do caminho mais curto entre dois vértices de um grafo;
- Determinação da clique de tamanho máximo de um grafo.

POC-Subgrupos

• Problemas indecidíveis: são problemas para os quais não existe nenhum algoritmo capaz de resolvê-los. Como exemplo, temos o Problema da Parada.

• Problemas decidíveis: são problemas que possuem um algoritmo cujo tempo de processamento cresce polinomial ou exponencialmente em função do tamanho da instância do problema.

Teorema da NP-Completude

A teoria da NP-Completude divide os problemas decidíveis de Otimização Combinatória em três grandes grupos, de acordo com a variação do esforço computacional (tempo) necessário para se resolver o problema em relação à variação do tamanho da instância do problema.

Teorema da NP-Completude

• Problemas em P: Os algoritmos para resolvê-los possuem tempo de processamento que cresce polinomialmente em função do tamanho da instância do problema. Este tipo de algoritmo é conhecido como eficiente.
Exemplos: Fluxo em redes, determinar a árvore geradora de peso mínimo de um grafo, determinar se um grafo de intervalo próprio é hamiltoniano, etc.

• Problemas em NP : Existência de um algoritmo determinístico para resolvê-lo cujo tempo de processamento cresce exponencialmente em função do tamanho da instância do problema.
Exemplos: Determinar se um grafo possui um ciclo hamiltoniano.

Teorema da NP-Completude

• Problemas intratáveis: São aqueles cujos algoritmos que os resolvem necessitam de um tempo exponencial em relação ao tamanho da instância.

Exemplo: O problema de encontrar todas as cliques maximais de um grafo.

Continuação

Com base na definição de transformação polinomial de problemas, e na pertinência ou não de um problema a NP, duas classes adicionais são definidas, complementando a teoria da NP-Completude.

• Problemas NP-difíceis: Um problema pertence a esta classe se todos os problemas em NP forem polinomialmente redutíveis ele.

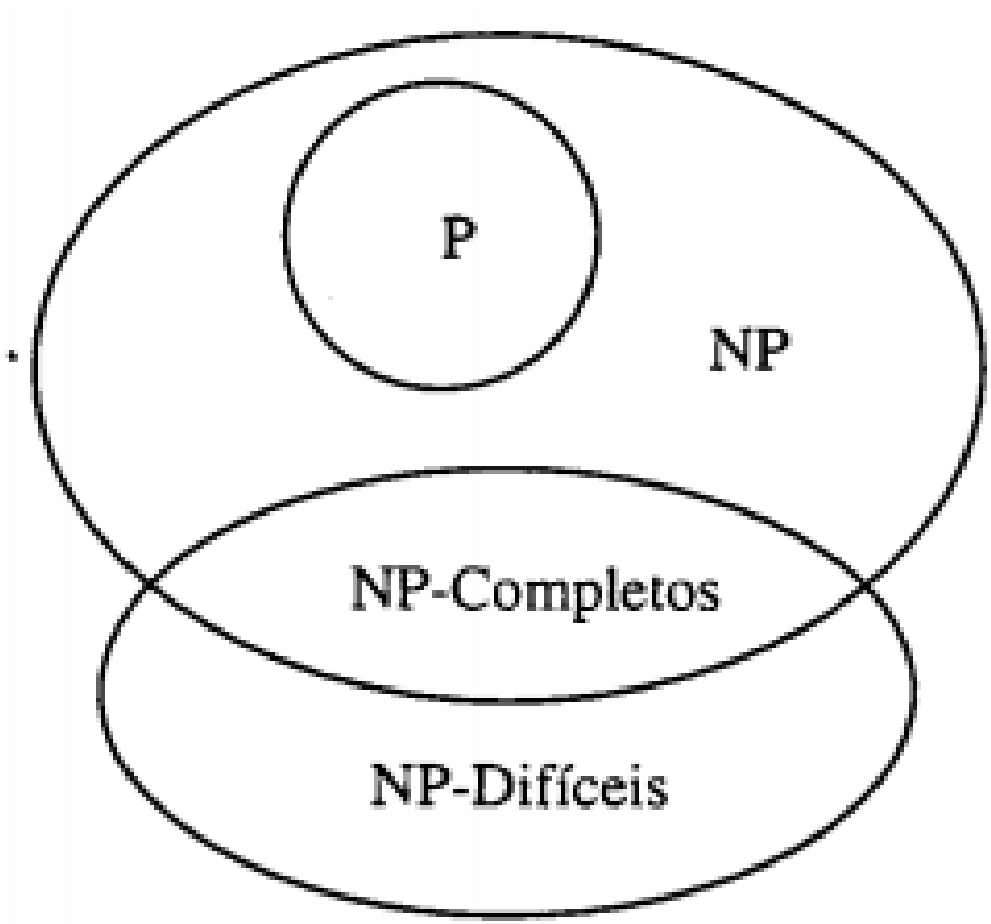
Exemplos: O problema do caixeiro viajante, o problema de se determinar urna clique de tamanho máximo de um grafo, o problema da mochila etc.

Formulação Algébrica

• Problemas NP-completos: São aqueles problemas pertencentes a NP aos quais todos os demais problemas em NP são polinomialmente redutíveis.

Exemplo: O problema de se determinar se um grafo possui um ciclo hamiltoniano com comprimento menor ou igual a um determinado valor.

Diagrama



Crescimento da complexidade

Considere um computador capaz de fazer 1 bilhão de adições por segundo e que no caso de 20 cidades, o computador precisa apenas de 19 adições para dizer qual o comprimento de uma rota. Então sera capaz de calcular $\frac{10^9}{19} = 53$ milhões de rotas por segundo.

Complexidade do TSP

Tabela 1: Comparação entre o número da cidade x tempo de processamento.

Cidades (n)	**Rotas/s**	**(n – 1)! / 2**	**Tempo de processamento**
5	250 milhões	12	insignificante
10	110 milhões	181.440	0,0015 seg.
15	71 milhões	$4{,}35 \times 10^{8}$	10 min.
20	53 milhões	$6{,}0 \times 10^{16}$	36 anos
25	42 milhões	$6{,}2 \times 10^{23}$	253×10^{6} anos

Fonte: Dissertação Aroldo Alexandre (2001)

Formulação Algébrica

.

$$X_{ij} = \begin{cases} 1, & \text{se o caixeiro vai da de cidade i para j} \\ 0, & \text{caso contrário.} \end{cases}$$

Função Objetivo:

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Onde:

c_{ij} é o custo de ir para cidade i a cidade j.

Restrições

$\sum_{i:i\neq j} x_{ij} = 1 \ \forall j$, ou seja, chega apenas um arco a cada cidade j.
$\sum_{i:i\neq j} x_{ij} = 1 \ \forall j$, o que nos diz, que de cada cidade i, sai só um arco.
$x_{ij} \in \{0, 1\} \ \forall i, j$
Somente com essas restrições pode-se obter subrotas.

Eliminando Subrotas

Dado um subconjunto $S \subset N$, deve-se limitar o número de variáveis associadas a essas cidades que podem receber valor diferente de a: $|S| - 1$, ou seja,

$$\sum_{i \in S}^{n} \sum_{j \in S}^{n} x_{ij} \leq |S| - 1$$

$\forall S \subset N$.

Métodos de Solução

Métodos de Otimização X Métodos Heurísticos

Métodos de Otimização

Garantem a otimalidade de uma solução, porém, o tempo necessário para garantir a otimalidade da solução pode ser inviável.

Métodos heurísticos

Não garantem a otimalidade de uma solução, porém, obtém rapidamente boas soluções por meio de métodos intuitivos.

Divisão e Conquista

Quando o problema inicial é "grande", sendo de difícil resolução direta, divide-se o problema em problemas cada vez menores até que possam ser resolvidos.

Branch and Bound

Branching(Ramificação)- O conjunto de soluções viáveis é divididos em subproblemas menores.
Conquista ou Eliminação- É realizada em dois passos.
1. Ir para a melhor solução do subconjunto(giving a bound).
2. Descartar o subconjunto se o limite**(bound)** não pode conter uma solução ótima.

Bibliotecas

- Panadas
- Numpy
- Folium
- Cplex
- Geopy
- GoogleMaps
- Matplotlib
- Networkx