Problemas de Busca (a.k.a **NP**) - parte 2

André Vignatti

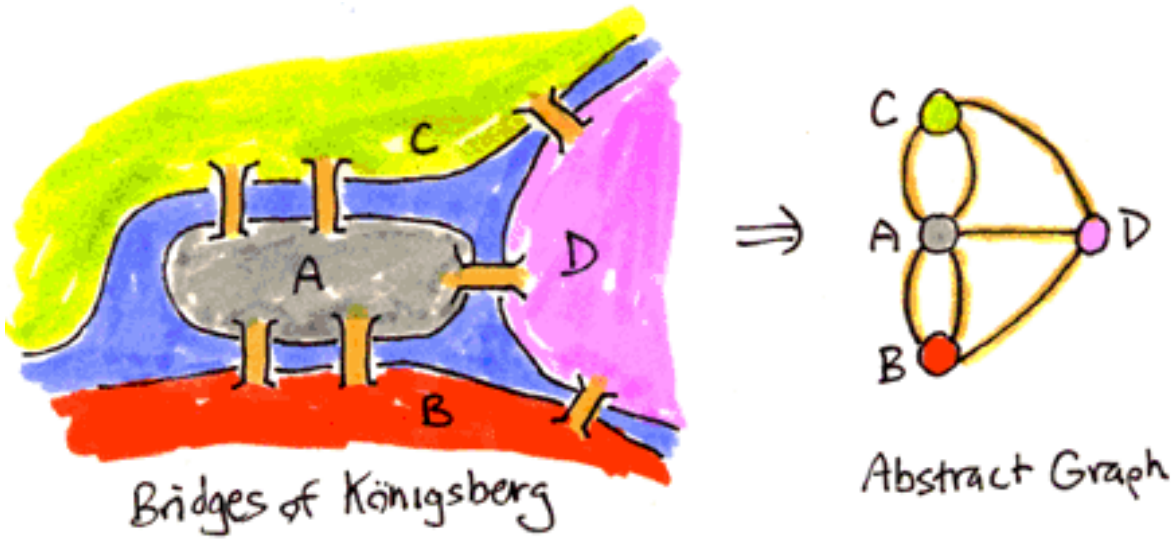
DINF- UFPR

Euler e Rudrata

No verão de 1735 Leonhard Euler, o famoso matemático suíço, estava andando nas pontes da cidade de Königsberg, na Prússia Oriental.

Depois de um tempo, ele notou frustrado que, não importa onde começasse sua caminhada, não importa o quão inteligente ele fazia sua rota, era impossível de atravessar cada ponte exatamente uma vez.

E a partir dessa ambição tola, o campo da teoria dos grafos nasceu.



Euler e Rudrata

Euler queria um caminho que passasse por cada aresta exatamente uma vez (é permitido repetir vértices)

Em outras palavras: quando um grafo pode ser desenhado sem levantar o lápis do papel?

A resposta descoberta por Euler é simples e elegante:

Se e somente se **(a)** o grafo é conexo e **(b)** cada vértice (com exceção dos vértices de início e final) têm grau par.

No caso de Königsberg, era impossível: todos os quatro vértices têm grau ímpar.

Euler e Rudrata

Assim, vamos definir o problema de busca **EULER PATH**:

Problema EULER PATH

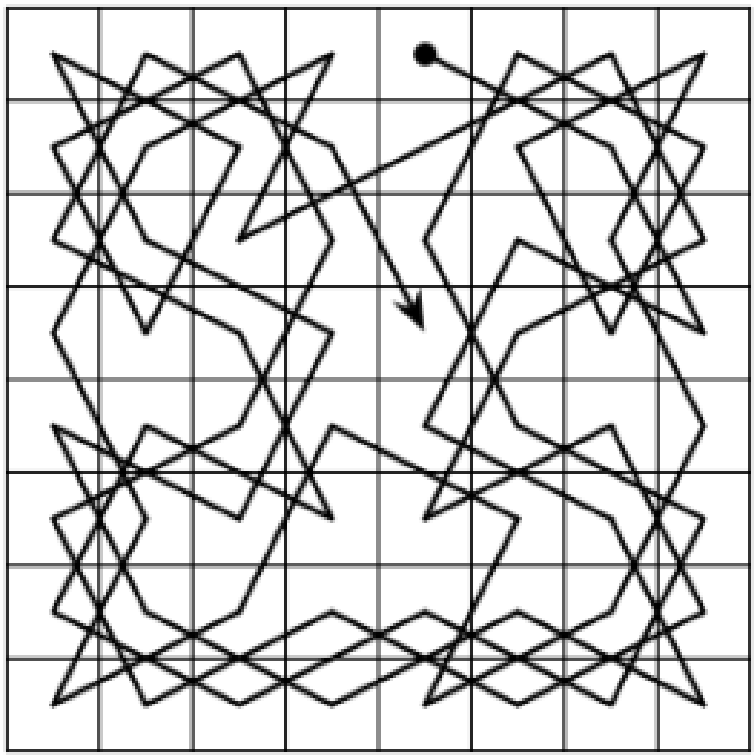
Dado um grafo, encontrar um caminho que contém cada aresta exatamente uma vez (ou dizer que não existe).

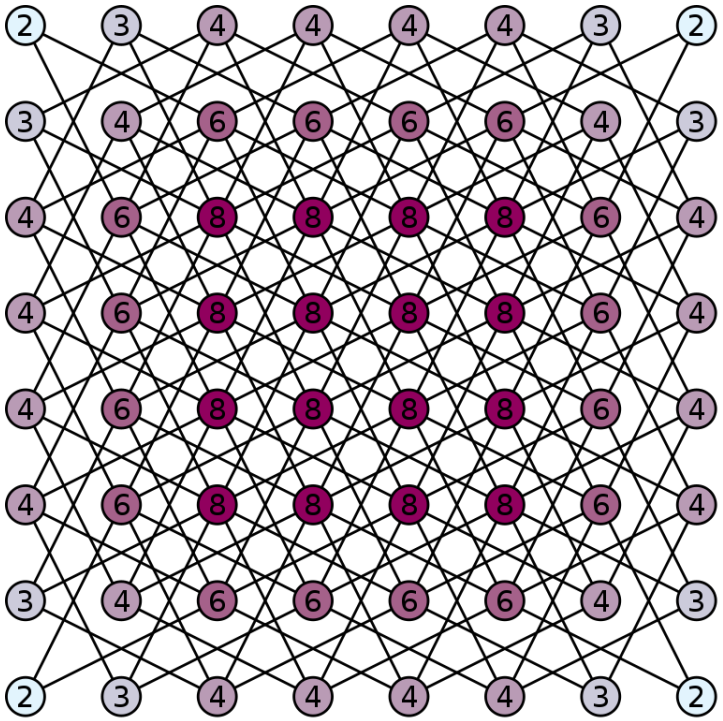
EULER PATH pode ser resolvido em tempo polinomial

Euler e Rudrata

Quase **um milênio antes do verão de Euler** na Prússia Oriental, um poeta da Caxemira chamado Rudrata tinha feito esta pergunta:

No xadrez, o cavalo consegue visitar todos os quadrados do tabuleiro, **sem repetir quadrados**, em uma longa caminhada que termina no quadrado inicial?





Euler e Rudrata

Novamente, temos um problema de grafos:

- são 64 vértices (os quadrados),
- dois vértices são unidos por **aresta** se o cavalo pode ir de um para o outro.

Queremos um ciclo que passa por todos os vértices

E não há razão para ficarmos só nos tabuleiros de xadrez: pode-se fazer a pergunta para **qualquer grafo**

Euler e Rudrata

Na literatura é conhecido como o **Problema do Ciclo de Hamiltoniano**, após o grande matemático irlandês que redescobriu o problema no século 19.

Assim, vamos definir o problema de busca **HAMILTONIAN CYCLE**:

Problema HAMILTONIAN CYCLE

Dado um grafo, encontrar um ciclo que visita cada vértice exatamente uma vez (ou dizer não existe).

Este problema tem **traços do TSP**, e nenhum algoritmo polinomial é conhecido para ele.

Euler e Rudrata

Há duas diferenças entre as definições dos problemas de Euler e Rudrata.

- Euler visitas todas as arestas, Hamilton visitas todos os vértices.
- Euler exige um caminho, Hamilton exige um ciclo.

Qual das diferenças causa a enorme disparidade computacional entre os dois problemas?

RESPOSTA: a primeira, porque a segunda diferença é puramente cosmética.

De fato, o problema **HAMILTON PATH** é equivalente a **HAMILTON CYCLE**, exceto que o objetivo é encontrar um caminho (ao invés de um ciclo).

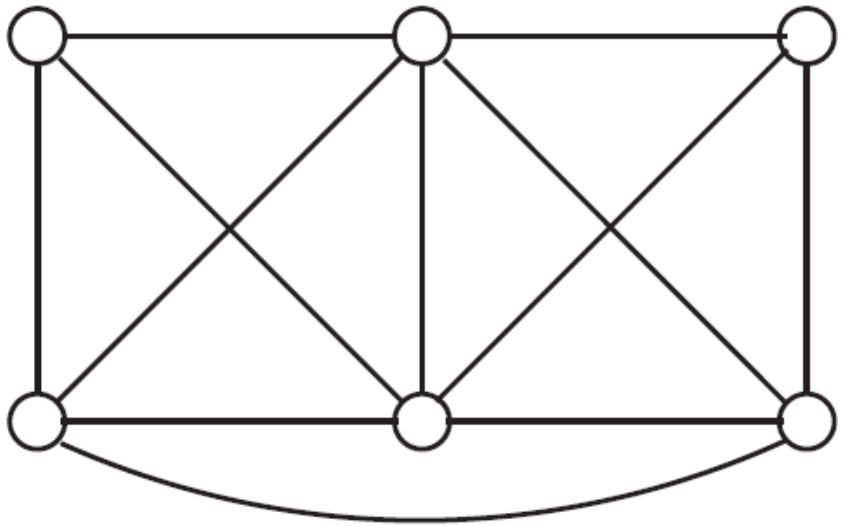
Cortes

Um corte é um conjunto de arestas cuja remoção deixa um grafo desconectado.

Problema MINIMUM CUT

Dado um grafo e um orçamento b, encontrar um corte com $\leq b$ arestas (ou dizer que não existe).

Por exemplo, o menor corte na figura é de tamanho 3.



Cortes

Este problema pode ser resolvido em tempo polinomial:

- **Corretude Probabilística**, pelo algoritmo aleatorizado de contração, apresentado em aula.
- **Corretude Determinística**, através do Fluxo Máximo = Corte Mínimo

Um problema relacionado é o MAXIMUM CUT:

Problema MAXIMUM CUT

Dado um grafo e um orçamento b, encontrar um corte com $\geq b$ arestas (ou dizer que não existe).

Ninguém conhece algoritmo polinomial para o MAXIMUM CUT

Equações Zero-Um

Vamos definir o problema de busca **Equações Zero-Um (ZOE)**:

Problema ZOE

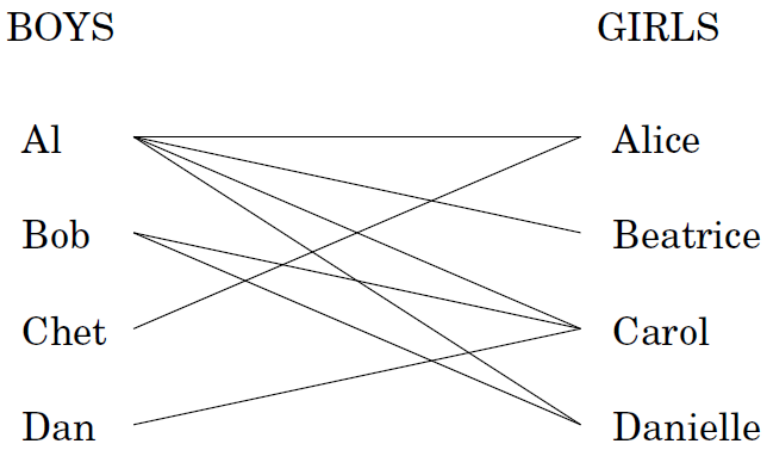
Seja $\mathbf{1}$ o vetor $(1, 1, \ldots, 1)$ com m posições. Dada uma matriz A com dimensões $m \times n$, queremos encontrar x tal que $Ax = \mathbf{1}$, onde cada posição de x assume valor em $\{0, 1\}$ (ou dizer não existe).

Ninguém conhece algoritmo polinomial para resolver esse problema!

Mas se as posições de x puderem ser valores reais no intervalo $[0, 1]$, então é resolvido em tempo polinomial.

Emparelhamento Tridimensional

Lembrando o problema **BIPARTITE MATCHING**: dado um grafo bipartido com n nós de cada lado (os meninos e as meninas), encontrar um conjunto de n arestas disjuntas (ou dizer não existe).



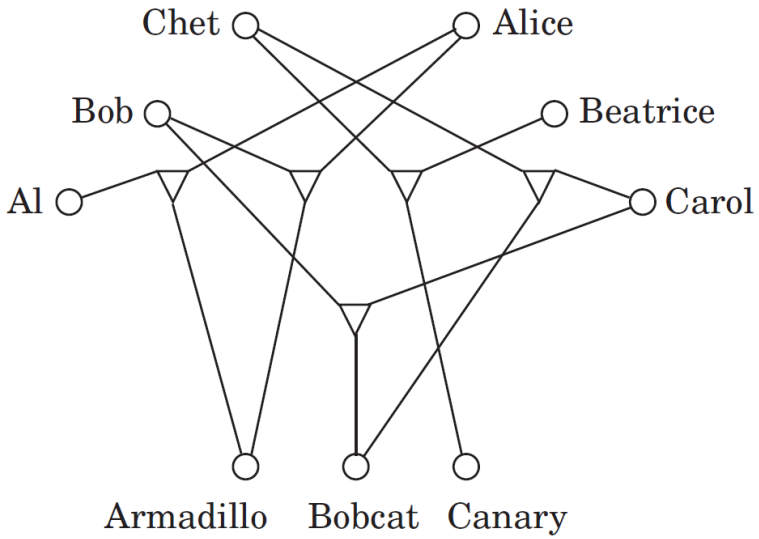
É possível resolver eficientemente este problema.

No entanto, há uma generalização chamada **3D MATCHING**, que não se conhece algoritmo polinomial.

Emparelhamento Tridimensional

No **3D MATCHING**, há n meninos e n meninas, mas também n animais de estimação.

As compatibilidades entre eles são especificados por um **conjunto de triplas**, cada um contendo um menino, uma menina, e um animal de estimação.



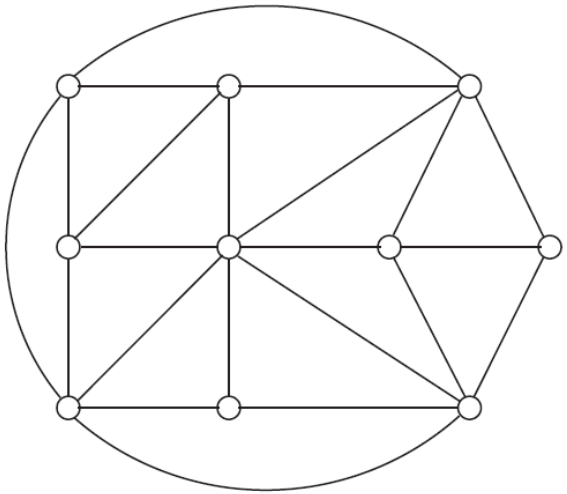
Intuitivamente, uma tripla (b, g, p) significa que o menino b, a menina g, e o animal p se dão bem juntos.
Queremos encontrar n **triplas disjuntas** e, assim, criar n **famílias harmoniosas**.

Conjunto Independente

Problema INDEPENDENT SET

É dado um grafo e um inteiro g, o objetivo é encontrar g vértices **independentes** (que não têm arestas entre eles), ou dizer que não existe.

Você consegue encontrar um conjunto independente de três vértices nesta Figura? E com quatro vértices?



Em **árvores é resolvido eficientemente**, para grafos em geral nenhum algoritmo polinomial é conhecido.

Cobertura por Vértices

Problema VERTEX COVER

É dado um grafo e um orçamento b, queremos encontrar b vértices que cobrem (toquem) todas as arestas (ou dizer que não existe).

É conhecido também como o problema do museu: é possível contratar b guardas para vigiar todos os corredores de um museu?

Você consegue cobrir todas as arestas da figura anterior com 7 vértices? E com 6?

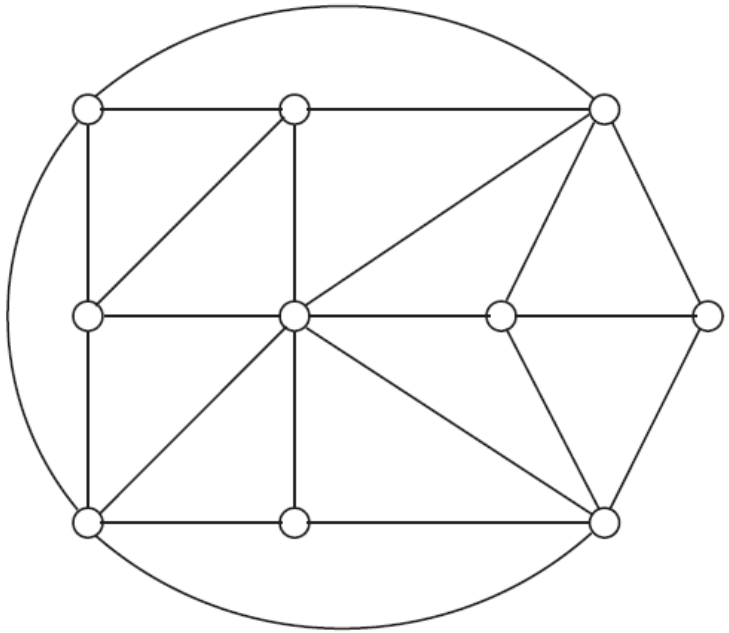
E você percebe a conexão com o problema do **conjunto independente**?

Clique

Problema CLIQUE

Dado um grafo e um objetivo g, encontrar um conjunto de g vértices tal que **todas as arestas possíveis entre eles estão presentes** (ou dizer que não existe).

Qual é o maior clique nesta figura?



Caminho Mais Longo

Sabemos que o problema do caminho mais curto pode ser resolvido eficientemente.

E o que podemos dizer sobre o problema do caminho mais longo (LONGEST PATH)?

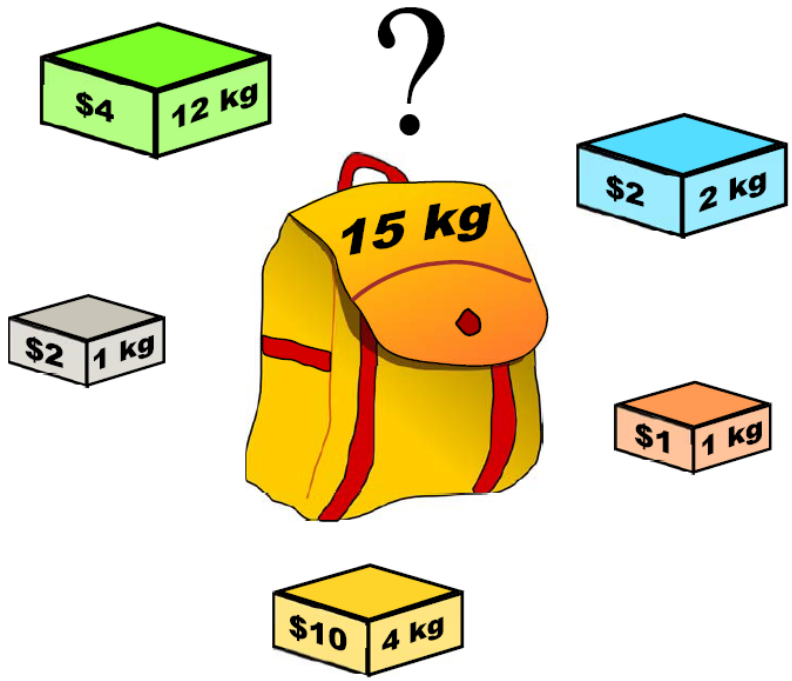
Problema LONGEST PATH

É dado um grafo G com **custos não negativos nas arestas**, dois vértices s e t, e um objetivo g. Queremos encontrar um $(s-t)$-caminho com custo $\geq g$ (ou dizer que não existe).

Naturalmente, para evitar soluções triviais, não permitimos caminhos com vértices repetidos.

Nenhum algoritmo eficiente é conhecido para este problema.

Mochila



Mochila

Problema MOCHILA

São dados:

- n itens
- pesos inteiros $w_1, \ldots, w_n$
- valores inteiros $v_1, \ldots, v_n$
- a capacidade W da mochila
- um objetivo g

Queremos encontrar um conjunto de itens cujo peso total é $\leq W$ e cujo valor total é $\geq g$. Se nenhum conjunto existir, devemos dizer.

Versão mais "'natural" é de otimização, g serve para torná-lo um problema de busca.

Mochila

Existe um **algoritmo de programação dinâmica** para MOCHILA que executa em tempo $O(nW)$, que é exponencial no tamanho da entrada, uma vez que envolve W ao invés de $\log W$.

E temos o algoritmo usual de busca exaustiva, que olha para todos os 2^n subconjuntos de itens.

Existe um algoritmo polinomial para mochila? Ninguém sabe...