P vs NP

Prof.: Leonardo Tórtoro Pereira

leonardop@usp.br

P vs NP

- Existem problemas que não podem ser resolvidos com um computador
  - São mais comuns do que parecem.
- Problemas que podem ser resolvidos por algoritmos polinomiais são “fáceis”
- Os exponenciais são “difíceis” (hard)

P vs NP

- Polinomial
  - $O(p(n))$
  - $p(n)$ é um polinômio
  - Pesquisa binária - $O(\log n)$
  - Pesquisa sequencial - $O(n)$
  - Insertion Sort - $O(n^2)$
  - Multiplicação de matrizes - $O(n^3)$

P vs NP

- Exponencial
  - $O(c^n)$, $c > 1$
  - Caixeiro Viajante - $O(n!)$
    - Com programação dinâmica - $O(n^2 * 2^n)$
  - Não podem ser resolvidos por algoritmos mesmo em tamanhos pequenos ou moderados

P vs NP

- Nós não temos uma teoria que mostre como obter algoritmos polinomiais para problemas que demandam algoritmos exponenciais
  - Mas não podemos provar que não existem!
- Mas podemos mostrar que os problemas que não possuem algoritmo polinomial conhecido são computacionalmente relacionados.
- Formam a classe conhecida como *NP*.

P vs NP

- Propriedade *NP*
  - Um problema desta classe poderá ser resolvido em tempo polinomial se e somente se:
    - Todos os outros problemas em *NP* também puderem
  - Indício forte de que dificilmente encontraremos um algoritmo eficiente para essa classe de problemas

P vs NP

- Uma característica da classe *NP*
  - Classe de problemas “sim/não” para os quais uma dada solução pode ser verificada facilmente
- A solução em si pode ser muito difícil ou comumente impossível de ser obtida
  - Mas tendo a solução, é possível verificar se está correta em tempo polinomial

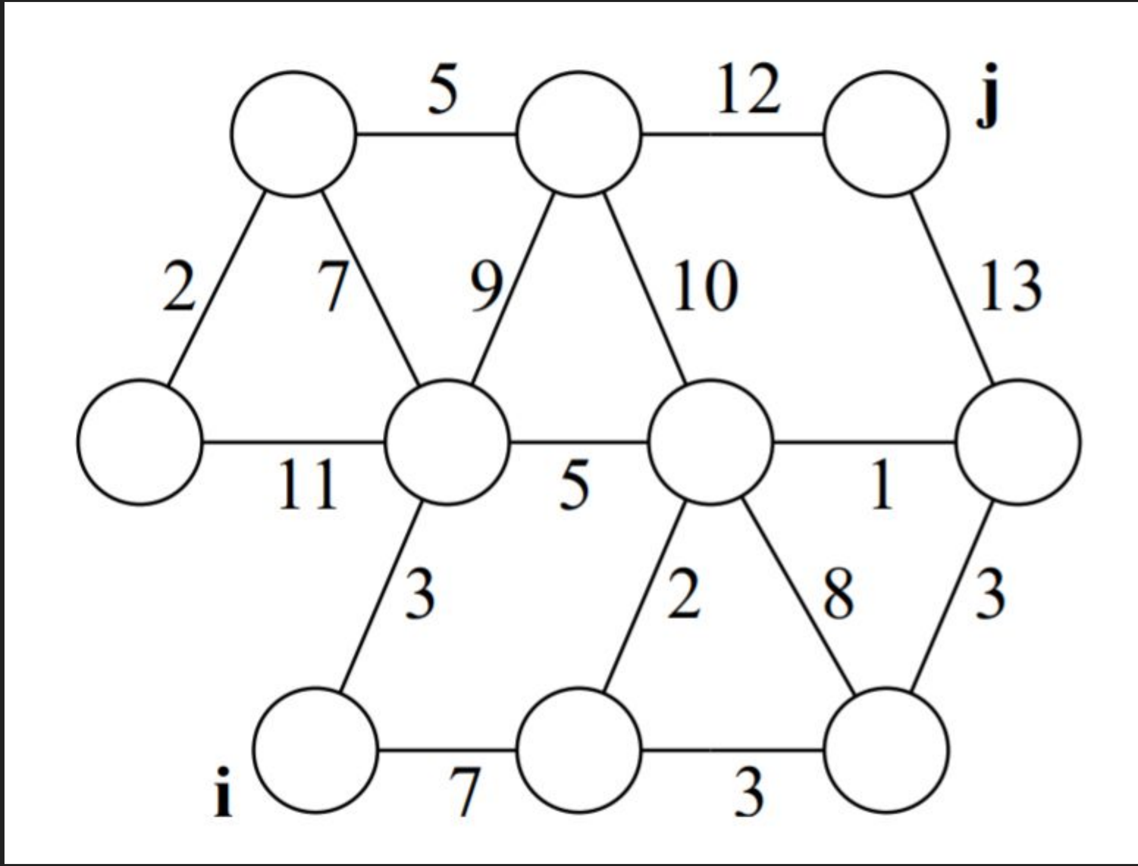
P vs NP

- Vamos ver alguns exemplos pra ilustrar as fronteiras entre problemas fáceis e difíceis

Caminho em grafo

Caminho em Grafo

- Considere um grafo com peso nas arestas, dois vértices i, j e um inteiro $k > 0$.
- Fácil: Existe um caminho de i até j com peso $\leq k$?
  - Há um algoritmo eficiente com complexidade de tempo $O(A \log V)$, sendo A o número de arestas e V o número de vértices (algoritmo de Dijkstra).
- Difícil: Existe um caminho de i até j com peso $\geq k$?
  - Não existe algoritmo eficiente. É equivalente ao PCV em termos de complexidade.



Fonte:
http://www2.dcc.ufmg.br/livros/algoritmos-java/cap9/transp/completo4/cap9.pdf

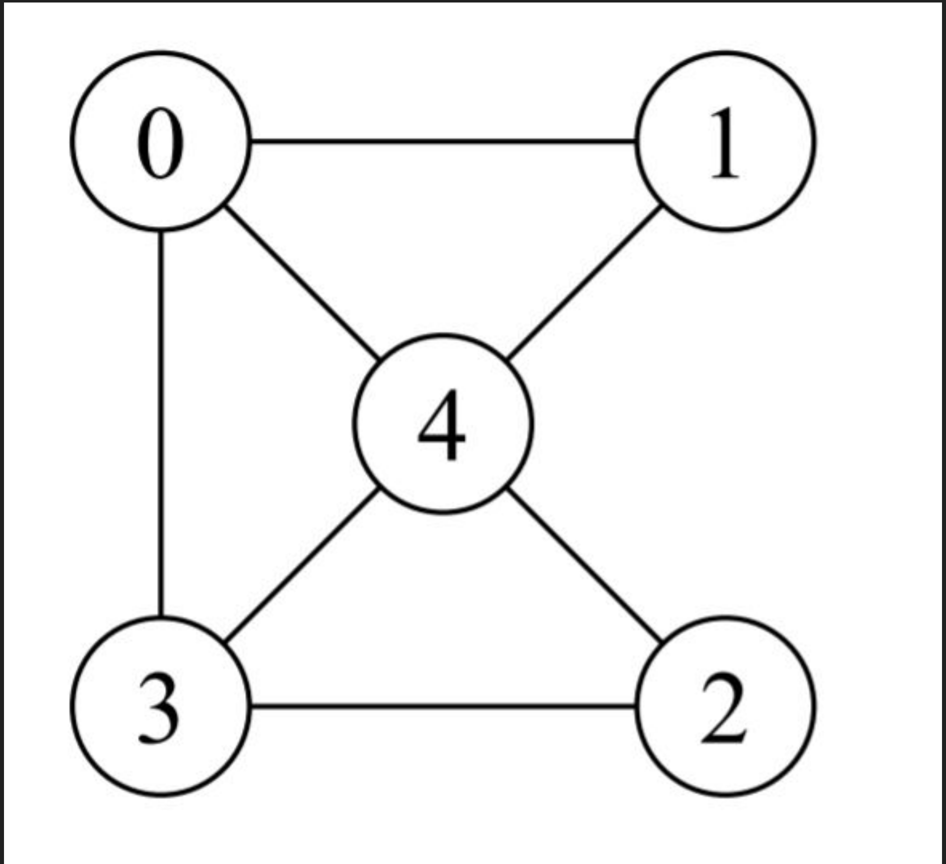
Caminho Hamiltoniano

Caminho Hamiltoniano

- Caminho que permite passar por todos os vértices de um grafo, sem repetir nenhum
  - Se descrever um ciclo, é um ciclo hamiltoniano
- Existe um ciclo de Hamilton no grafo G?
  - Fácil: Grafos com grau máximo = 2 (vértices com no máximo duas arestas incidentes).
  - Difícil: Grafos com grau > 2.

Caminho Hamiltoniano

- É um caso especial do PCV. Pares de vértices com uma aresta entre eles tem distância 1 e pares de vértices sem aresta entre eles têm distância infinita.

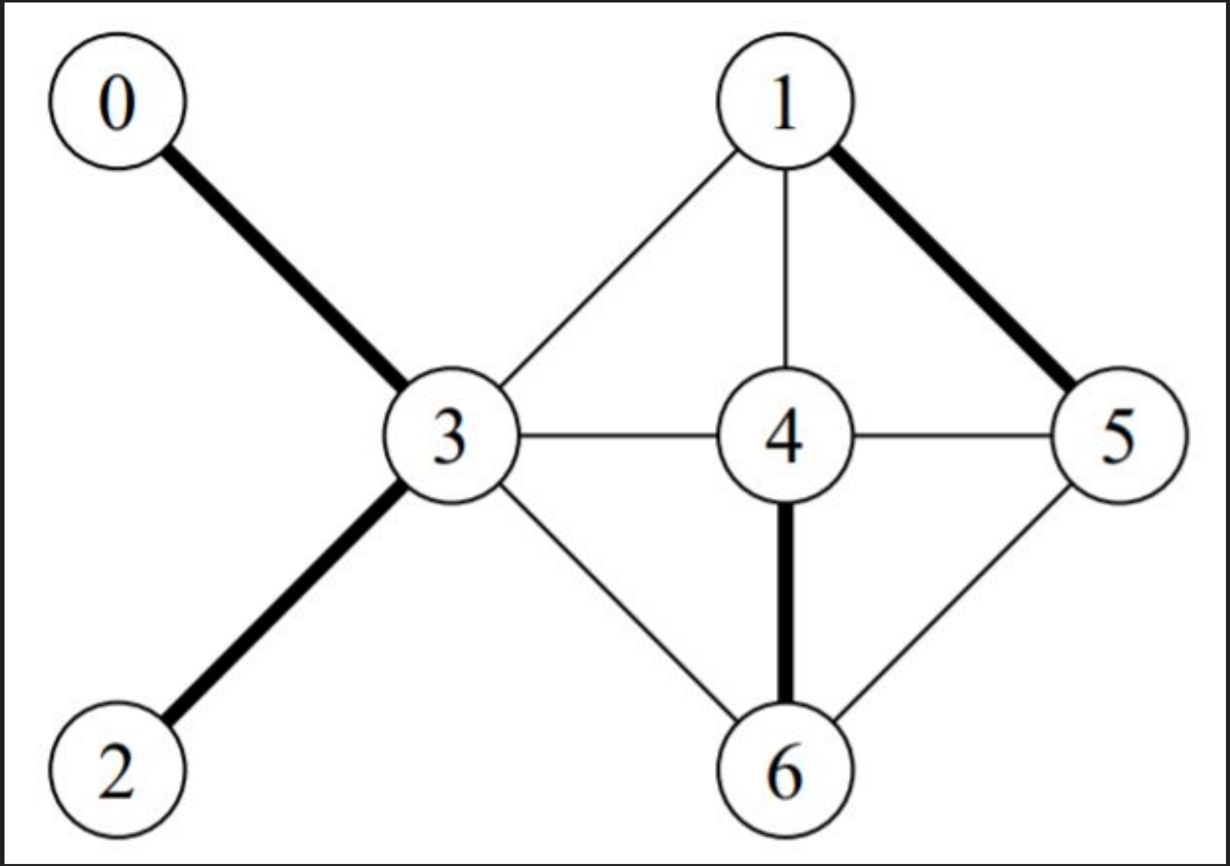


Fonte:
http://www2.dcc.ufmg.br/livros/algoritmos-java/cap9/transp/completo4/cap9.pdf

Cobertura de Arestas

Cobertura de Arestas

- Uma cobertura de arestas de um grafo $G = (V, A)$ é um subconjunto $A' \subset A$ de k arestas tal que todo $v \in V$ é parte de pelo menos uma aresta de A'.
- Uma cobertura de vértices é um subconjunto $V' \subset V$ tal que se $(u, v) \in A$ então $u \in V'$ ou $v \in V'$, isto é, cada aresta do grafo é incidente em um dos vértices de V'.



Fonte:
http://www2.dcc.ufmg.br/livros/algoritmos-java/cap9/transp/completo4/cap9.pdf

Cobertura de Arestas

- O conjunto resposta para $k = 4$ é $A' = \{(0, 3),(2, 3),(4, 6),(1, 5)\}$.
- O conjunto resposta é $V' = \{3, 4, 5\}$, para $k = 3$.
- Dados um grafo e um inteiro $k > 0$
  - Fácil: há uma cobertura de arestas $\leq k$?.
  - Difícil: há uma cobertura de vértices $\leq k$?

Algoritmos Não-deterministas

Algoritmos Não-Deterministas

- Algoritmos deterministas:
  - O resultado de cada operação é definido de forma única.
  - É possível remover essa restrição teoricamente.
  - Apesar de parecer irreal, este é um conceito importante e geralmente utilizado para definir a classe N P.
  - Neste caso, os algoritmos podem conter operações cujo resultado não é definido de forma única.

Algoritmos Não-Deterministas

- Algoritmo não-determinista:
  - capaz de escolher uma dentre as várias alternativas possíveis a cada passo.
  - Algoritmos não-deterministas contêm operações cujo resultado não é unicamente definido, ainda que limitado a um conjunto especificado de possibilidades.

Caracterização das Classes P e NP

Caracterização das Classes P e NP

- P: conjunto de todos os problemas que podem ser resolvidos por algoritmos deterministas em tempo polinomial.
- N P: conjunto de todos os problemas que podem ser resolvidos por algoritmos não-deterministas em tempo polinomial.

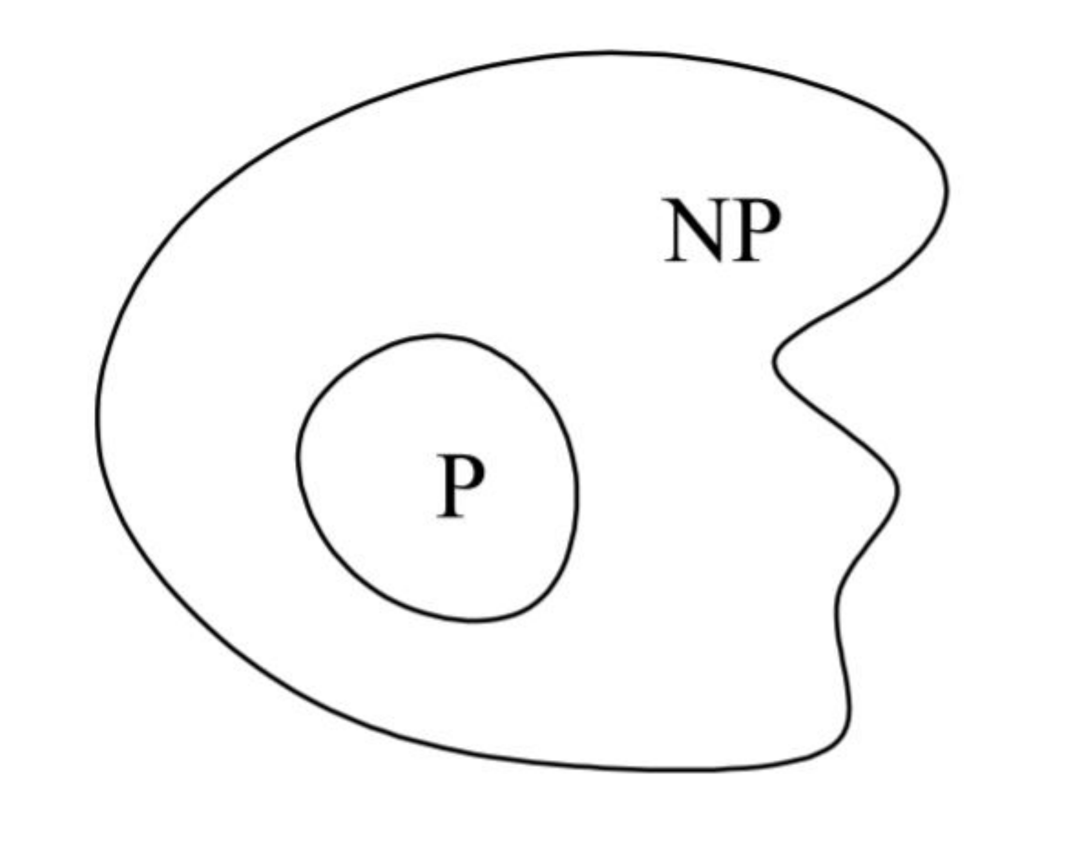
Caracterização das Classes P e NP

- Para mostrar que um determinado problema está em N P, basta apresentar um algoritmo não-determinista que execute em tempo polinomial para resolver o problema.
- Outra maneira é encontrar um algoritmo determinista polinomial para verificar que uma dada solução é válida.

$P = NP?$

Caracterização das Classes P e NP

- P ⊆ N P, pois algoritmos deterministas são um caso especial dos não-deterministas.
- A questão é se P = N P ou P != N P.
- Esse é o problema não resolvido mais famoso que existe na área de ciência da computação.
- Se existem algoritmos polinomiais deterministas para todos os problemas em N P, então P = N P.
- Em contrapartida, a prova de que P != N P parece exigir técnicas ainda desconhecidas.



Fonte:
http://www2.dcc.ufmg.br/livros/algoritmos-java/cap9/transp/completo4/cap9.pdf

Caracterização das Classes P e NP

- Acredita-se que N P >> P, pois para muitos problemas em N P, não existem algoritmos polinomiais conhecidos, nem um limite inferior não-polinomial provado.
- Muitos problemas práticos em N P podem ou não pertencer a P (não conhecemos nenhum algoritmo determinista eficiente para eles).

Caracterização das Classes P e NP

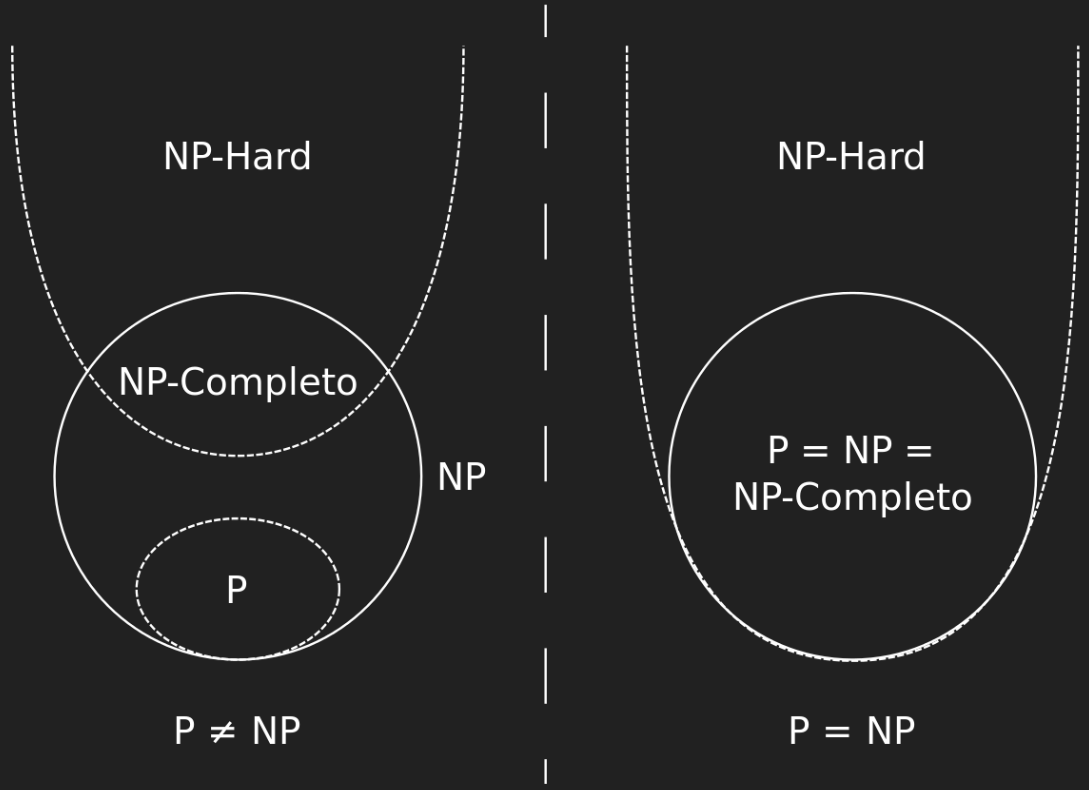
- Se conseguirmos provar que um problema não pertence a P, então temos um indício de que esse problema pertence a N P e que esse problema é tão difícil de ser resolvido quanto outros problemas N P.
- Como não existe tal prova, sempre há esperança de que alguém descubra um algoritmo eficiente. • Quase ninguém acredita que N P = P.
- Existe um esforço considerável para provar o contrário, mas a questão continua em aberto!

Caracterização das Classes P e NP

- Existem muitos outros conceitos envolvendo o conjunto NP, inclusive subclassificações como NP-Completo e NP-Difícil.
- Vamos apenas citar brevemente sua existência, pois explicá-los precisaria de mais uma aula inteira :)

Caracterização das Classes P e NP

- NP-Completo
  - Os mais difíceis problemas NP
  - Se um algoritmo resolver eficientemente um, resolve todos (inclusive qualquer NP)
- NP-Difícil
  - É pelo menos tão difícil que os problemas mais difíceis de NP
  - Mas não precisam estar em NP (podem não ser de decisão)

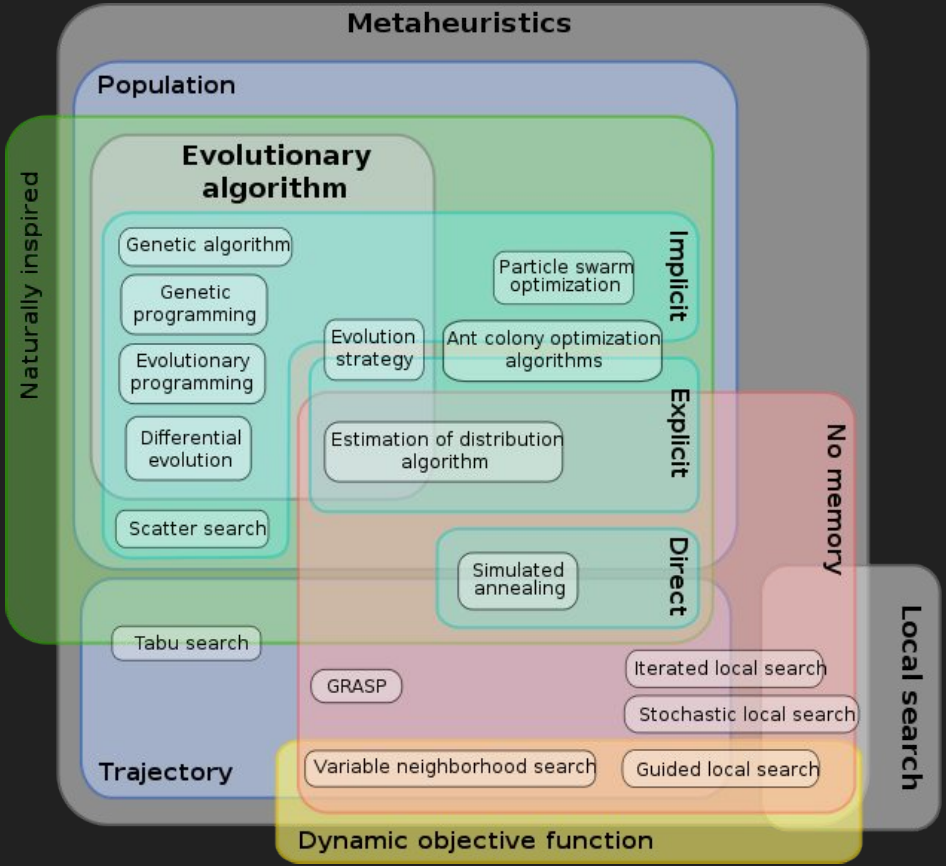


Fonte:
https://pt.wikipedia.org/wiki/NP-dif%C3%ADcil#/media/Ficheiro:P_np_np-completo_np-hard.svg

E como lidar com esses problemas NP?

Como lidar com NP?

- Enquanto não conseguimos encontrar uma solução algorítmica, a melhor saída são aproximações
    - Heurísticas!
- Falamos um pouco em outras aulas, mas são algoritmos que não conseguem garantir a solução ótima. E nem provar que sempre são eficientes...
- Mas costumam ser bons o bastante!



Fonte: https://en.wikipedia.org/wiki/Metaheuristic

Referências

Referências

1. ZIVIANI, N. Projeto de Algoritmos. 2º edição, Thomson, 2004.
2. http://www2.dcc.ufmg.br/livros/algoritmos-java/cap9/transp/completo4/cap9.pdf
3. https://www.researchgate.net/publication/220423686_The_Status_of_the_P_versus_NP_problem
4. https://www.youtube.com/watch?v=qv6UVOQ0F44&ab_channel=SethBling
5. https://arxiv.org/pdf/1203.1895.pdf