



Universidade Federal de São Carlos
Departamento de Engenharia de Produção



Otimização Linear Contínua e Discreta

PPGEP, UFSCar - Semestre 01/2025
Prof. Dr. Pedro Munari (munari@dep.ufscar.br)

Tópico 8.2: Classes de Problemas em Otimização

Apresentação - Tópico 8.2

Objetivos deste tópico:

- ▶ Conhecer as diferentes classes de problemas de decisão e otimização;
- ▶ Compreender essas classificações e sua importância;
- ▶ Estudar como provar que um problema é $\mathcal{NP}$ -completo e $\mathcal{NP}$ -difícil.

Vídeo-aula:

- ▶ https://www.youtube.com/watch?v=ApRmVU00Y_o

Complexidade computacional

▷ Alguns problemas em $\mathcal{P}$

- ▶ Problema do caminho mínimo com valores não-negativos:
 - ▶ Algoritmo de Dijkstra requer $O(m^2)$ cálculos elementares;
 - ▶ A função de complexidade é independente dos valores numéricos.

Complexidade computacional

▷ Alguns problemas em $\mathcal{P}$

- ▶ Problema do caminho mínimo com valores não-negativos:
 - ▶ Algoritmo de Dijkstra requer $O(m^2)$ cálculos elementares;
 - ▶ A função de complexidade é independente dos valores numéricos.
- ▶ Problema do caminho mínimo (com alguns valores não-negativos):
 - ▶ Algoritmo de Bellman-Ford requer $O(m^3)$ cálculos elementares para encontrar um caminho mínimo ou detectar um ciclo de custo negativo;

Complexidade computacional

▷ Alguns problemas em $\mathcal{P}$

- ▶ Problema do caminho mínimo com valores não-negativos:
 - ▶ Algoritmo de Dijkstra requer $O(m^2)$ cálculos elementares;
 - ▶ A função de complexidade é independente dos valores numéricos.
- ▶ Problema do caminho mínimo (com alguns valores não-negativos):
 - ▶ Algoritmo de Bellman-Ford requer $O(m^3)$ cálculos elementares para encontrar um caminho mínimo ou detectar um ciclo de custo negativo;
- ▶ Problema de transporte:
 - ▶ O algoritmo primal-dual com mudança de escala possui tempo de execução $O(m^3 \log \theta)$.

Problemas de decisão

Estamos interessados em problemas de otimização, mas a teoria de complexidade é baseada em *problemas de decisão* (sim/não);

Problemas de decisão

Estamos interessados em problemas de otimização, mas a teoria de complexidade é baseada em *problemas de decisão* (sim/não);

- Por exemplo: Existe uma rota com distância total menor que K ?

Problemas de decisão

Estamos interessados em problemas de otimização, mas a teoria de complexidade é baseada em *problemas de decisão* (sim/não);

- ▶ Por exemplo: Existe uma rota com distância total menor que K ?
- ▶ Observe que é fácil converter um problema de otimização para um problema de decisão.

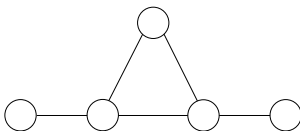
Problemas de decisão

Estamos interessados em problemas de otimização, mas a teoria de complexidade é baseada em *problemas de decisão* (sim/não);

- ▶ Por exemplo: Existe uma rota com distância total menor que K ?
- ▶ Observe que é fácil converter um problema de otimização para um problema de decisão.

Exemplos:

- ▶ CLIQUE: Dado um grafo $G = (V, E)$ e um inteiro k , existe um clique (subgrafo completo) de tamanho k ?



Problemas de decisão

Satisfabilidade (*satisfiability*, SAT):

Problemas de decisão

Satisfabilidade (*satisfiability*, SAT):

- Dada uma fórmula booleana, existe uma atribuição de valores às suas variáveis que a torne verdadeira?

Problemas de decisão

Satisfabilidade (*satisfiability*, SAT):

- ▶ Dada uma fórmula booleana, existe uma atribuição de valores às suas variáveis que a torne verdadeira?
- ▶ A fórmula deve estar na forma normal conjuntiva (ou disjuntiva): cláusulas definidas apenas com operador $\vee$ (sem repetir elementos) e combinadas com operador $\wedge$;

Problemas de decisão

Satisfabilidade (*satisfiability*, SAT):

- ▶ Dada uma fórmula booleana, existe uma atribuição de valores às suas variáveis que a torne verdadeira?
- ▶ A fórmula deve estar na forma normal conjuntiva (ou disjuntiva): cláusulas definidas apenas com operador $\vee$ (sem repetir elementos) e combinadas com operador $\wedge$;
- ▶ Por exemplo: Existem valores booleanos para x_1, x_2, x_3 tais que a expressão $(x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee x_2)$ seja verdadeira?

Problemas de decisão

Satisfabilidade (*satisfiability*, SAT):

- ▶ Dada uma fórmula booleana, existe uma atribuição de valores às suas variáveis que a torne verdadeira?
- ▶ A fórmula deve estar na forma normal conjuntiva (ou disjuntiva): cláusulas definidas apenas com operador $\vee$ (sem repetir elementos) e combinadas com operador $\wedge$;
- ▶ Por exemplo: Existem valores booleanos para x_1, x_2, x_3 tais que a expressão $(x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee x_2)$ seja verdadeira?
 - ▶ Sim! Basta atribuírmos $x_1 = V$ e $x_2 = V$ (x_3 qualquer).

Problemas de decisão

▷ Classes de problemas

Classe $\mathcal{P}$ (*polynomial time*)

Problemas de decisão

▷ Classes de problemas

Classe $\mathcal{P}$ (*polynomial time*)

- ▶ Vimos que $\mathcal{P}$ é a classe de problemas (de decisão!) que podem ser resolvidos em **tempo polinomial**;
- ▶ Algoritmos com tempo $O(n^2)$, $O(n \log n)$, $O(n^{232})$, etc.

Problemas de decisão

▷ Classes de problemas

Classe $\mathcal{NP}$

Problemas de decisão

▷ Classes de problemas

Classe $\mathcal{NP}$ (*nondeterministic polynomial time*)

Problemas de decisão

▷ Classes de problemas

Classe $\mathcal{NP}$ (*nondeterministic polynomial time*)

- ▶ Problemas de decisão para os quais qualquer resultado **sim** possui um certificado (“solução”) de tamanho polinomial que pode ser **checado em tempo polinomial**.

Problemas de decisão

▷ Classes de problemas

Classe $\mathcal{NP}$ (*nondeterministic polynomial time*)

- ▶ Problemas de decisão para os quais qualquer resultado **sim** possui um certificado (“solução”) de tamanho polinomial que pode ser **checado em tempo polinomial**.
- ▶ Assim, sempre que a decisão é **sim**, é possível verificar a resposta em tempo polinomial;

Problemas de decisão

▷ Classes de problemas

Classe $\mathcal{NP}$ (*nondeterministic polynomial time*)

- ▶ Problemas de decisão para os quais qualquer resultado **sim** possui um certificado (“solução”) de tamanho polinomial que pode ser **checado em tempo polinomial**.
- ▶ Assim, sempre que a decisão é **sim**, é possível verificar a resposta em tempo polinomial;
- ▶ $\mathcal{P} \subset \mathcal{NP}$;

Problemas de decisão

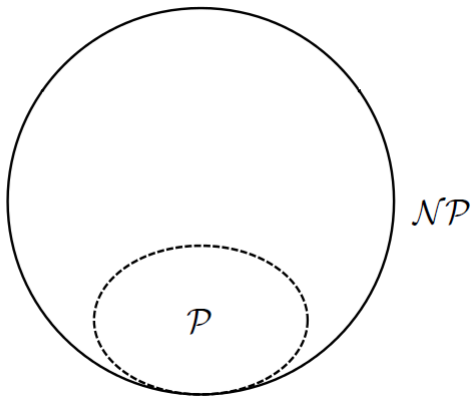
▷ Classes de problemas

Classe $\mathcal{NP}$ (*nondeterministic polynomial time*)

- ▶ Problemas de decisão para os quais qualquer resultado **sim** possui um certificado (“solução”) de tamanho polinomial que pode ser **checado em tempo polinomial**.
- ▶ Assim, sempre que a decisão é **sim**, é possível verificar a resposta em tempo polinomial;
- ▶ $\mathcal{P} \subset \mathcal{NP}$;
- ▶ *Nondeterministic*: solucionados em tempo polinomial em uma Máquina de Turing não-determinística;

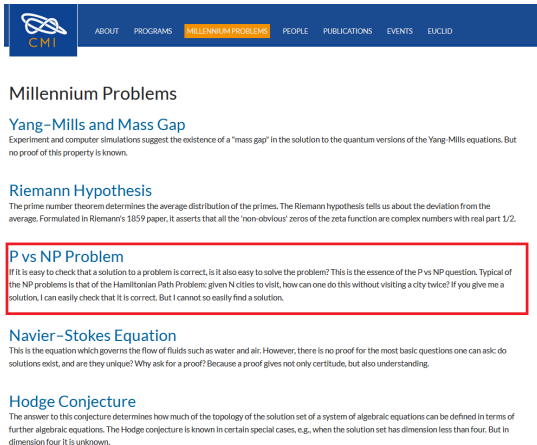
Classes de problemas

▷ Diagrama



Quem quer ser um milionário?

▷ $\mathcal{P} = \mathcal{NP}$?



The screenshot shows the Clay Mathematics Institute website. At the top is a blue navigation bar with the CMI logo and links for ABOUT, PROGRAMS, MILLENNIUM PROBLEMS (highlighted in orange), PEOPLE, PUBLICATIONS, EVENTS, and EUCLID. Below the navigation bar is the heading "Millennium Problems". There are five problem entries, each with a title in blue and a brief description in black. The "P vs NP Problem" entry is enclosed in a red rectangular border. The other problems are Yang-Mills and Mass Gap, Riemann Hypothesis, Navier-Stokes Equation, and Hodge Conjecture.

Millennium Problems

Yang-Mills and Mass Gap
Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang-Mills equations. But no proof of this property is known.

Riemann Hypothesis
The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part 1/2.

P vs NP Problem
If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Navier-Stokes Equation
This is the equation which governs the flow of fluids such as water and air. However, there is no proof for the most basic questions one can ask: do solutions exist, and are they unique? Why ask for a proof? Because a proof gives not only certitude, but also understanding.

Hodge Conjecture
The answer to this conjecture determines how much of the topology of the solution set of a system of algebraic equations can be defined in terms of further algebraic equations. The Hodge conjecture is known in certain special cases, e.g., when the solution set has dimension less than four. But in dimension four it is unknown.

<http://www.claymath.org/millennium-problems>

Problemas de decisão

▷ Classes de problemas

- ▶ Um problema A é **reduzível** a um problema B ($A \propto B$) se para qualquer instância d_A de A podemos construir uma instância d_B de B em tempo polinomial de modo que d_A resulta em **sim** para A se, e somente se, d_B resulta em **sim** para B .

Problemas de decisão

▷ Classes de problemas

- ▶ Um problema A é **reduzível** a um problema B ($A \propto B$) se para qualquer instância d_A de A podemos construir uma instância d_B de B em tempo polinomial de modo que d_A resulta em **sim** para A se, e somente se, d_B resulta em **sim** para B .
- ▶ Um problema B é $\mathcal{NP}$ -completo se (1) $B \in \mathcal{NP}$; (2) dado qualquer $A \in \mathcal{NP}$, então $A \propto B$;

Problemas de decisão

▷ Classes de problemas

- ▶ Um problema A é **reduzível** a um problema B ($A \propto B$) se para qualquer instância d_A de A podemos construir uma instância d_B de B em tempo polinomial de modo que d_A resulta em **sim** para A se, e somente se, d_B resulta em **sim** para B .
- ▶ Um problema B é $\mathcal{NP}$ -completo se (1) $B \in \mathcal{NP}$; (2) dado qualquer $A \in \mathcal{NP}$, então $A \propto B$;
- ▶ “os problemas mais difíceis em $\mathcal{NP}$ ”;

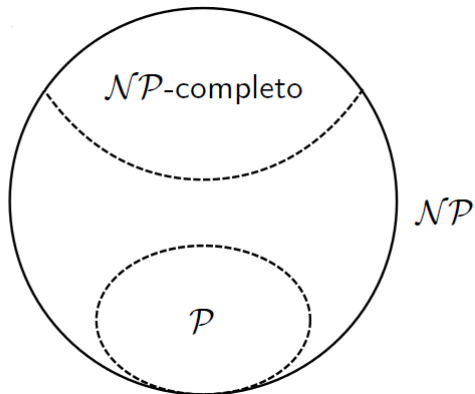
Problemas de decisão

▷ Classes de problemas

- ▶ Um problema A é **reduzível** a um problema B ($A \propto B$) se para qualquer instância d_A de A podemos construir uma instância d_B de B em tempo polinomial de modo que d_A resulta em **sim** para A se, e somente se, d_B resulta em **sim** para B .
- ▶ Um problema B é **$\mathcal{NP}$ -completo** se (1) $B \in \mathcal{NP}$; (2) dado qualquer $A \in \mathcal{NP}$, então $A \propto B$;
- ▶ “os problemas mais difíceis em $\mathcal{NP}$ ”;
- ▶ Se houver um algoritmo em tempo polinomial para um deles, então haverá um algoritmo em tempo polinomial para **todos** em $\mathcal{NP}$!

Classes de problemas

▷ Diagrama



Problemas de decisão

▷ Classes de problemas

- ▶ Dados A e B em $\mathcal{NP}$, de modo que $A \propto B$:

Problemas de decisão

▷ Classes de problemas

- ▶ Dados A e B em $\mathcal{NP}$, de modo que $A \propto B$:
 - ▶ Se $B \in \mathcal{P}$, então $A \in \mathcal{P}$;

Problemas de decisão

▷ Classes de problemas

- ▶ Dados A e B em $\mathcal{NP}$, de modo que $A \leq B$:
 - ▶ Se $B \in \mathcal{P}$, então $A \in \mathcal{P}$;
 - ▶ Se $A \in \mathcal{NP}$ -completo, então $B \in \mathcal{NP}$ -completo;

Problemas de decisão

▷ Classes de problemas

- ▶ Dados A e B em $\mathcal{NP}$, de modo que $A \propto B$:
 - ▶ Se $B \in \mathcal{P}$, então $A \in \mathcal{P}$;
 - ▶ Se $A \in \mathcal{NP}$ -completo, então $B \in \mathcal{NP}$ -completo;
- ▶ Logo, para provar que um problema B é $\mathcal{NP}$ -completo, partimos de um problema A que já sabemos ser $\mathcal{NP}$ -completo e então provamos que $A \propto B$.

Problemas de decisão

▷ Classes de problemas

- ▶ Dados A e B em $\mathcal{NP}$, de modo que $A \propto B$:
 - ▶ Se $B \in \mathcal{P}$, então $A \in \mathcal{P}$;
 - ▶ Se $A \in \mathcal{NP}$ -completo, então $B \in \mathcal{NP}$ -completo;
- ▶ Logo, para provar que um problema B é $\mathcal{NP}$ -completo, partimos de um problema A que já sabemos ser $\mathcal{NP}$ -completo e então provamos que $A \propto B$.
- ▶ *Stephen Cook, 1971*: SAT é $\mathcal{NP}$ -completo.

Problemas de decisão

▷ Classes de problemas

- ▶ Dados A e B em $\mathcal{NP}$, de modo que $A \propto B$:
 - ▶ Se $B \in \mathcal{P}$, então $A \in \mathcal{P}$;
 - ▶ Se $A \in \mathcal{NP}$ -completo, então $B \in \mathcal{NP}$ -completo;
- ▶ Logo, para provar que um problema B é $\mathcal{NP}$ -completo, partimos de um problema A que já sabemos ser $\mathcal{NP}$ -completo e então provamos que $A \propto B$.
- ▶ *Stephen Cook, 1971*: SAT é $\mathcal{NP}$ -completo.
- ▶ Daí para outros problemas, fazemos por exemplo: SAT $\propto$ CLIQUE; CLIQUE $\propto$ VERTEX PACKING; etc.

SAT $\propto$ CLIQUE

▷ Ideia da prova

- ▶ Vamos construir um grafo a partir da fórmula booleana (em tempo polinomial!);
- ▶ Seja k o número de cláusulas (expressões entre parênteses);
- ▶ Para cada elemento na fórmula (qualquer x_i ou $\neg x_i$) crie um vértice (se o mesmo elemento aparecer mais de uma vez, cada cópia resulta em um vértice);
- ▶ Para cada par de elementos em cláusulas diferentes *que podem ser Verdadeiros ao mesmo tempo*, introduzimos uma aresta entre eles;

SAT $\propto$ CLIQUE

▷ Ideia da prova

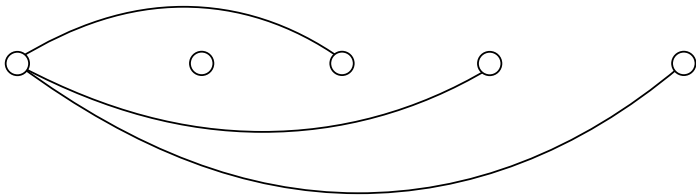
$$x_1 \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge x_3$$



SAT $\propto$ CLIQUE

▷ Ideia da prova

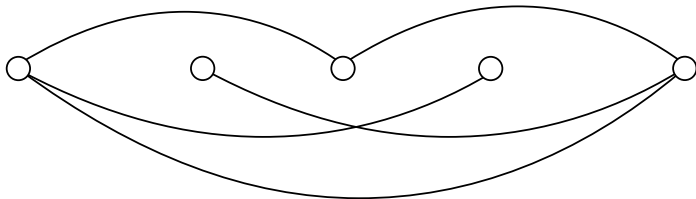
$$x_1 \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge x_3$$



SAT $\propto$ CLIQUE

▷ Ideia da prova

$$x_1 \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge x_3$$



SAT $\propto$ CLIQUE

▷ Ideia da prova

- ▶ Observe que um clique no grafo construído corresponde a um conjunto de elementos, cada um em uma cláusula diferente, e que podem ser definidos como Verdadeiro ao mesmo tempo;
- ▶ Logo, um clique do tamanho do número de cláusulas corresponde a uma atribuição que avalia a fórmula como Verdadeira.

Problemas de otimização

- ▶ As classes que vimos são para problemas de **decisão**;

Problemas de otimização

- ▶ As classes que vimos são para problemas de **decisão**;
- ▶ Como vimos, é fácil fazermos um paralelo entre otimização e decisão, embora sejam problemas diferentes;

Problemas de otimização

- ▶ As classes que vimos são para problemas de **decisão**;
- ▶ Como vimos, é fácil fazermos um paralelo entre otimização e decisão, embora sejam problemas diferentes;
- ▶ Existe uma classificação que envolve outros tipos de problemas, além de problemas de decisão, conhecida como $\mathcal{NP}$ -difícil ($\mathcal{NP}$ -hard);

Problemas de otimização

- ▶ As classes que vimos são para problemas de **decisão**;
- ▶ Como vimos, é fácil fazermos um paralelo entre otimização e decisão, embora sejam problemas diferentes;
- ▶ Existe uma classificação que envolve outros tipos de problemas, além de problemas de decisão, conhecida como $\mathcal{NP}$ -difícil ($\mathcal{NP}$ -hard);
- ▶ “Tão difícil quanto o problema mais difícil em $\mathcal{NP}$ ”;

Problemas de otimização

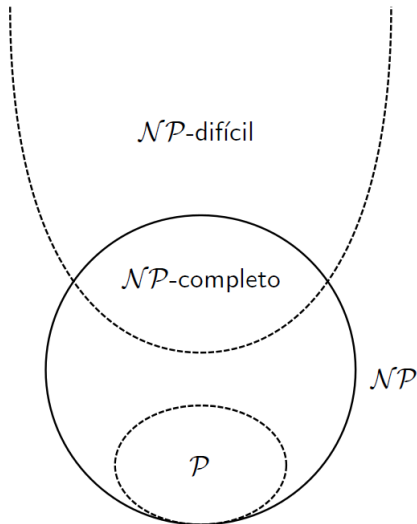
- ▶ As classes que vimos são para problemas de **decisão**;
- ▶ Como vimos, é fácil fazermos um paralelo entre otimização e decisão, embora sejam problemas diferentes;
- ▶ Existe uma classificação que envolve outros tipos de problemas, além de problemas de decisão, conhecida como $\mathcal{NP}$ -difícil ($\mathcal{NP}$ -hard);
- ▶ “Tão difícil quanto o problema mais difícil em $\mathcal{NP}$ ”;
- ▶ Atenção: Um problema $\mathcal{NP}$ -difícil não precisa estar em $\mathcal{NP}$!

Problemas de otimização

- ▶ As classes que vimos são para problemas de **decisão**;
- ▶ Como vimos, é fácil fazermos um paralelo entre otimização e decisão, embora sejam problemas diferentes;
- ▶ Existe uma classificação que envolve outros tipos de problemas, além de problemas de decisão, conhecida como $\mathcal{NP}$ -difícil ($\mathcal{NP}$ -hard);
- ▶ “Tão difícil quanto o problema mais difícil em $\mathcal{NP}$ ”;
- ▶ Atenção: Um problema $\mathcal{NP}$ -difícil não precisa estar em $\mathcal{NP}$!
- ▶ Um problema $\mathcal{NP}$ -difícil que está em $\mathcal{NP}$, é um problema $\mathcal{NP}$ -completo.

Classes de problemas

▷ Diagrama



Classes de problemas

▷ <https://pt.wikipedia.org/wiki/NP-difícil>

Convenção de nomeação NP [[editar](#) | [editar código-fonte](#)]

O sistema de nomeação da família NP é confuso: os problemas NP-difíceis, não são todos NP, a despeito de ter *NP* como o prefixo do seu nome da classe. No entanto, os nomes estão agora estabelecidos e improváveis de mudar. Por outro lado, o sistema de nomeação *NP*- tem algum sentido mais profundo, porque a família NP é definida em relação à classe NP:

NP-difícil

Tão difíceis quanto os problemas mais difíceis em NP. Tais problemas não precisam estar em NP, na verdade, eles nem precisam ser obrigatoriamente problemas de decisão.

NP-completo

Estes são os problemas mais difíceis em NP. Tais problemas são NP-difíceis e em NP.

$\mathcal{NP}$ -difícil

- Precisamos do conceito de oráculo: Dado um problema de decisão ou otimização A , dizemos que um algoritmo possui A como oráculo quando esse algoritmo resolve uma instância de A com uma única instrução;

$\mathcal{NP}$ -difícil

- ▶ Precisamos do conceito de oráculo: Dado um problema de decisão ou otimização A , dizemos que um algoritmo possui A como oráculo quando esse algoritmo resolve uma instância de A com uma única instrução;
- ▶ Um problema A é $\mathcal{NP}$ -difícil se existe um algoritmo de tempo polinomial para um problema B $\mathcal{NP}$ -completo quando esse algoritmo tem A como oráculo;

$\mathcal{NP}$ -difícil

- ▶ Precisamos do conceito de oráculo: Dado um problema de decisão ou otimização A , dizemos que um algoritmo possui A como oráculo quando esse algoritmo resolve uma instância de A com uma única instrução;
- ▶ Um problema A é $\mathcal{NP}$ -difícil se existe um algoritmo de tempo polinomial para um problema B $\mathcal{NP}$ -completo quando esse algoritmo tem A como oráculo;
- ▶ $\mathcal{NP}$ -difícil em geral se refere a problemas de otimização cujo problema de decisão correspondente é $\mathcal{NP}$ -completo;

$\mathcal{NP}$ -difícil

- ▶ Precisamos do conceito de oráculo: Dado um problema de decisão ou otimização A , dizemos que um algoritmo possui A como oráculo quando esse algoritmo resolve uma instância de A com uma única instrução;
- ▶ Um problema A é $\mathcal{NP}$ -difícil se existe um algoritmo de tempo polinomial para um problema B $\mathcal{NP}$ -completo quando esse algoritmo tem A como oráculo;
- ▶ $\mathcal{NP}$ -difícil em geral se refere a problemas de otimização cujo problema de decisão correspondente é $\mathcal{NP}$ -completo;
- ▶ O Problema do Caixeiro Viajante (PCV) é $\mathcal{NP}$ -difícil. Quem conseguir achar um algoritmo de tempo polinomial pra resolvê-lo fica milionário (e muito famoso!).

PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?

PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?
- ▶ Problema de decisão $\mathcal{NP}$ -completo: (2) Dado um grafo $G(V, E)$ **qualquer**, existe um ciclo Hamiltoniano nesse grafo?

PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?
- ▶ Problema de decisão $\mathcal{NP}$ -completo: (2) Dado um grafo $G(V, E)$ **qualquer**, existe um ciclo Hamiltoniano nesse grafo?
- ▶ Basta mostrar que (2) é redutível a (1):

PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?
- ▶ Problema de decisão $\mathcal{NP}$ -completo: (2) Dado um grafo $G(V, E)$ **qualquer**, existe um ciclo Hamiltoniano nesse grafo?
- ▶ Basta mostrar que (2) é redutível a (1): definir custo 1 para as arestas em E ;

PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?
- ▶ Problema de decisão $\mathcal{NP}$ -completo: (2) Dado um grafo $G(V, E)$ **qualquer**, existe um ciclo Hamiltoniano nesse grafo?
- ▶ Basta mostrar que (2) é redutível a (1): definir custo 1 para as arestas em E ; adicionar novas arestas com custo 2 até o grafo ficar completo;

PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?
- ▶ Problema de decisão $\mathcal{NP}$ -completo: (2) Dado um grafo $G(V, E)$ **qualquer**, existe um ciclo Hamiltoniano nesse grafo?
- ▶ Basta mostrar que (2) é redutível a (1): definir custo 1 para as arestas em E ; adicionar novas arestas com custo 2 até o grafo ficar completo; usar (1) como oráculo com $k = n$.

PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?
- ▶ Problema de decisão $\mathcal{NP}$ -completo: (2) Dado um grafo $G(V, E)$ **qualquer**, existe um ciclo Hamiltoniano nesse grafo?
- ▶ Basta mostrar que (2) é redutível a (1): definir custo 1 para as arestas em E ; adicionar novas arestas com custo 2 até o grafo ficar completo; usar (1) como oráculo com $k = n$.
- ▶ Observe que esses passos são feitos em tempo polinomial;

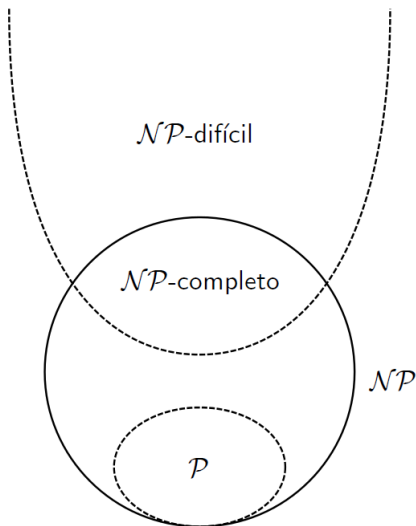
PCV é $\mathcal{NP}$ -difícil

▷ Ideia da prova

- ▶ Problema de decisão relacionado ao PCV: (1) Dado um grafo completo e um número k , existe uma rota factível de custo no máximo k ?
- ▶ Problema de decisão $\mathcal{NP}$ -completo: (2) Dado um grafo $G(V, E)$ **qualquer**, existe um ciclo Hamiltoniano nesse grafo?
- ▶ Basta mostrar que (2) é redutível a (1): definir custo 1 para as arestas em E ; adicionar novas arestas com custo 2 até o grafo ficar completo; usar (1) como oráculo com $k = n$.
- ▶ Observe que esses passos são feitos em tempo polinomial;
- ▶ Além disso, se a resposta do problema (1) é **sim**, então existe um ciclo que visita todos os nós do grafo uma única vez e usando as arestas originais de G (o contrário também é verdadeiro).

Classes de problemas

▷ Diagrama



- ▶ Obrigado pela atenção!
- ▶ Dúvidas?