

# Introdução à classe de problemas NP-Completo

R. Rossetti, A. P. Rocha, L. Ferreira, J. P. Fernandes, F. Ramos, G. Leão  
FEUP, MIEIC, CAL



FEUP Universidade do Porto  
Faculdade de Engenharia

Introdução à classe de problemas NP-Completo

«#»

## Índice

- A classe de problemas P
- A classe de problemas NP
- As classes de problemas NP-completos e NP-difíceis
- Classificação de problemas por redução



FEUP Universidade do Porto  
Faculdade de Engenharia

Introdução à classe de problemas NP-Completo

«#»

## Preâmbulo

- Em alguns casos práticos, alguns algoritmos podem resolver problemas simples em tempo razoável (e.g.  $n \leq 20$ ); mas quando se trata de inputs maiores (e.g.  $n \geq 100$ ) o desempenho degrada consideravelmente
- Soluções desse género podem estar a executar em tempo exponencial, da ordem de  $n^{\sqrt{n}}$ ,  $2^n$ ,  $2^{(2^n)}$ ,  $n!$ , ou mesmo pior
- Para algumas classes de problemas, é difícil determinar se há algum paradigma ou técnica que leve à solução do mesmo, ou se há formas de provar que o problema é intrinsecamente difícil, não sendo possível encontrar uma solução algorítmica cujo desempenho seja sub-exponencial
- Para alguns problemas difíceis, é possível afirmar que, se um desses problemas pode ser resolvido em tempo polinomial, então todos podem ser resolvidos em tempo polinomial!

## A classe de problemas P

## Tempo Polinomial como Referência

- Considera-se normalmente que um **problema é resolúvel eficientemente** se for **resolúvel em tempo polinomial**, i.e., se houver um **algoritmo de tempo polinomial** que o resolva.
- Um algoritmo é de **tempo polinomial** se o tempo de execução é da ordem de  $O(n^k)$ , no pior caso, em que  $n$  é o **tamanho do input** do problema e  $k$  é uma **constante independente de  $n$**
- Algumas funções parecem não ser polinomiais, mas podem ser tratadas como tal: e.g.  $O(n \log n)$  tem delimitação superior  $O(n^2)$
- Algumas funções parecem ser polinomiais, mas podem não o ser na verdade: e.g.  $O(n^k)$ , se  $k$  variar em função de  $n$ , tamanho do *input*.



## Problemas de Decisão

- Um **problema de decisão** é um problema cujo *output* ou resposta deve ser um simples “SIM” ou “NÃO” (ou derivativos do tipo “V/F”, “0/1”, “aceitar/rejeitar”, etc.)
- Muitos problemas práticos são problemas de optimização (maximizar ou minimizar alguma métrica), mas podem ser expressos em termos de problemas de decisão
- Por exemplo, o problema “qual o menor número de cores que se pode utilizar para colorir um grafo  $G$ ?” pode ser expresso como “Dado um grafo  $G$  e um inteiro  $k$ , é possível colorir  $G$  com  $k$  cores?”



## A classe de problemas P

- A classe de problemas P é constituída por todos os problemas de decisão que podem ser resolvidos em tempo polinomial
- A generalidade dos problemas em grafos que temos abordado pertencem a esta classe (na versão de problema de decisão)
  - Caminho mais curto
  - Árvore de expansão mínima
  - Fluxo máximo
  - Fluxo de custo mínimo
  - Circuito de Euler
  - Problema do carteiro chinês
  - Problemas de emparelhamento
  - ...



Universidade do Porto  
Faculdade de Engenharia

Introdução à classe de problemas NP-Completo

«#»

## A classe de problemas NP



Universidade do Porto  
Faculdade de Engenharia

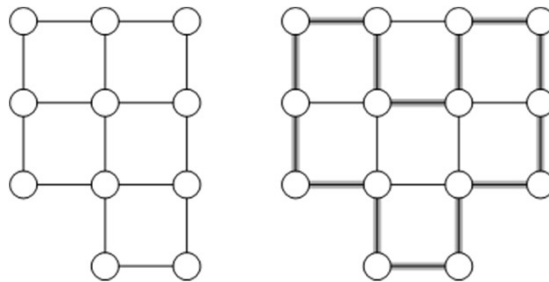
Introdução à classe de problemas NP-Completo

8

«#»

## Problema do Circuito Hamiltoniano

- Problema do circuito Hamiltoniano não dirigido (*Undirected Hamiltonian Cycle - UHC*): verificar se um grafo não dirigido dado  $G$ , é Hamiltoniano, isto é, tem um ciclo (ou circuito) que visita cada vértice exatamente uma vez.



## Verificação em Tempo Polinomial

- Não se conhece nenhum algoritmo eficiente (de tempo polinomial) para resolver o problema anterior
- No entanto, dado um ciclo candidato  $C$ , é fácil verificar em tempo polinomial (linear) se cumpre a propriedade pretendida
- Neste contexto,  $C$  diz-se ser um “**certificado**” de uma solução (uma “prova” de que o grafo é Hamiltoniano)
- Diz-se que o problema é **verificável em tempo polinomial**, se for possível verificar em tempo polinomial se um certificado de uma solução é correto
- Nem todos os problemas têm esta característica: e.g., o problema de determinar se um grafo  $G$  tem exactamente um ciclo de Hamilton. É fácil certificar que existe pelo menos um ciclo, mas não é fácil certificar que não há mais!

## Classe de problemas NP

- A classe de problemas NP (*nondeterministic polynomial*) é definida por todos os problemas que podem ser verificados por um algoritmo de tempo polinomial
  - “Verificados” no sentido explicado no slide anterior
  - Não confundir **resolução** em tempo polinomial (P) com **verificação** em tempo polinomial (NP)!
  - \* O termo “*nondeterministic*” provém da definição inicial da classe NP em termos de máquinas de Turing não deterministas, capazes de não deterministicamente conjecturar o valor do certificado (em geral conjecturar uma *string*) e verificá-lo depois, em tempo polinomial (em geral, verificar se a string faz parte da linguagem).



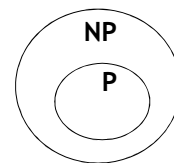
Universidade do Porto  
Faculdade de Engenharia

Introdução à classe de problemas NP-Completo

«#»

## Relação entre as classes P e NP

- $P \subseteq NP$ 
  - Se um problema é resolúvel em tempo polinomial, então pode-se certamente verificar se uma solução é correcta em tempo polinomial
- Não se sabe certamente se  $P = NP$  ou  $P \neq NP$ !
  - Poder verificar se uma solução é correcta em tempo polinomial, não garante ou ajuda a encontrar um algoritmo que resolva o problema em tempo polinomial
  - Acredita-se que  $P \neq NP$ , mas não há provas!



Universidade do Porto  
Faculdade de Engenharia

Introdução à classe de problemas NP-Completo

«#»

## As classes de problemas NP-completos e NP-difíceis



## Problemas NP-completos (NPC)

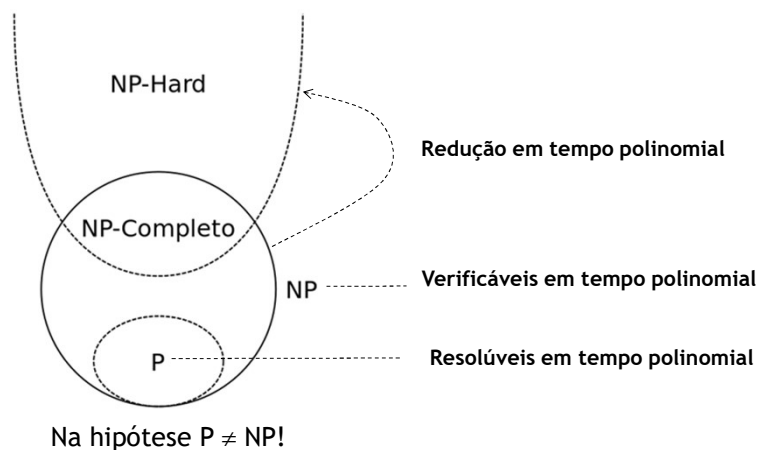
- A classe de problemas NP-completos é a classe dos problemas “mais difíceis” de resolver em toda a classe NP.
  - São pelo menos tão difíceis como qualquer outro problema em NP
- Mais precisamente, um problema de decisão  $A$  é NP-completo se (i)  $A \in NP$ ; e (ii) qualquer problema  $A' \in NP$  é **reduzível em tempo polinomial** a  $A$  ( $A' \leq_p A$ )
  - A redução envolve converter os dados de entrada de  $A'$  em dados de entrada de  $A$ , e os dados de saída de  $A$  em dados de saída de  $A'$
  - As reduções serão estudadas adiante
  - Atualmente, para provar que  $A$  é NPC, basta encontrar um problema  $A'$  NPC já conhecido e provar que  $A'$  é reduzível a  $A$  em tempo polinomial
- E.g., o problema do circuito Hamiltoniano é NP-completo



## Problemas NP-difíceis (NP-hard)

- Um problema NP-difícil é um problema que satisfaz a propriedade (ii) mas não necessariamente a propriedade (i)
- Ou seja, um problema de decisão  $A$  é NP-difícil se qualquer problema  $A' \in NP$  é **redutível em tempo polinomial** a  $A$
- Por exemplo, o problema da paragem (em máquinas de Turing) é NP-difícil mas não NP-completo
  - Não pertence a NP
  - É NP-difícil, pois se pode converter qualquer problema NP no problema de paragem de uma máquina de Turing (ver referências)

## Classes P, NP, NP-complete e NP-hard





## Classificação de problemas por redução



FEUP Universidade do Porto  
Faculdade de Engenharia

Introdução à classe de problemas NP-Completo

17

«#»

## Como “provar” que um problema $X \notin P$

1. Selecionar um problema  $Y$  não resolúvel em tempo polinomial ( $Y \notin P$ )
    - De acordo com o conhecimento atual, em que se acredita que  $P \neq NP$
  2. Provar que  $Y$  é redutível a  $X$  em tempo polinomial ( $Y \leq_p X$ )
    - Redução de entradas e saídas
- Como a redução é efetuada em tempo polinomial, se  $X$  for resolúvel em tempo polinomial, então  $Y$  também o seria, o que contradiz a hipótese
  - Em geral, a redução de  $Y$  a  $X$  permite provar que  $X$  é pelo menos tão difícil quanto  $Y$



FEUP Universidade do Porto  
Faculdade de Engenharia

Introdução à classe de problemas NP-Completo

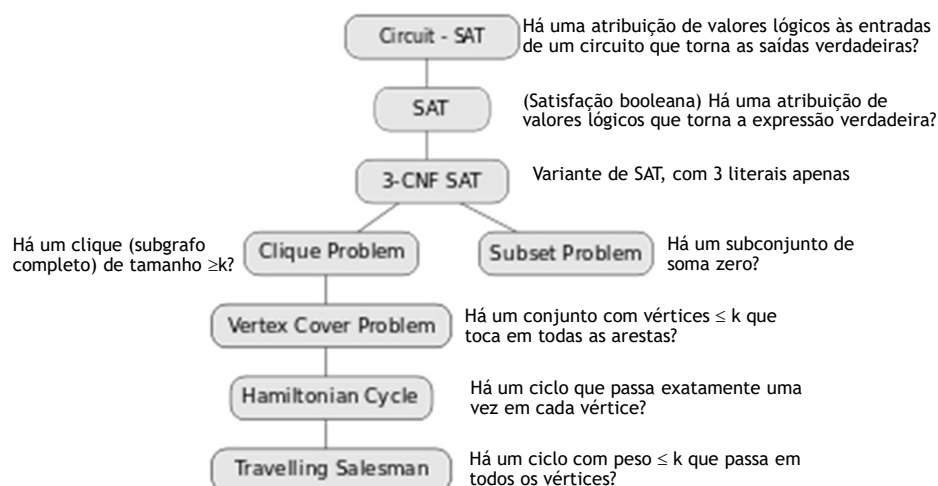
«#»

## Como provar que um problema $X \in \text{NPC}$

1. Provar que  $X$  está em  $\text{NP}$
2. Seleccionar um problema  $Y$  que se sabe ser  $\text{NP}$ -completo
3. Definir uma redução de tempo polinomial de  $Y$  em  $X$  (*conversão de entradas*)
4. Provar que, dada uma instância de  $Y$ ,  $Y$  tem uma solução se, e se somente,  $X$  tem uma solução (*conversão de saídas*)



## Exemplos de problemas NP-completos e reduções normalmente usadas



## Referências e mais informação

- T. Cormen *et al.* (2009) “Introduction to Algorithms.” Cambridge, MA: MIT press.
  - Capítulo 34 NP-Completeness
  - Capítulo 35 Approximation Algorithms
- R. Johnsonbaugh & M. Schaefer (2004) “Algorithms.” Upper Saddle River, NJ: Prentice Hall.
- C.A. Shaffer (2001) “A Practical Introduction to Data Structures and Algorithm Analysis.” Upper Saddle River, NJ: Prentice Hall.