

Introdução à classe de problemas NP- Completos

R. Rossetti, A.P. Rocha, J. Pascoal Faria

FEUP, MIEIC, CAL, 2013/2014

Introdução

■ Considerações Práticas

- 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 $n^{1/n}$, 2^n , $2^{(2^n)}$, $n!$, ou mesmo piores do que isso
- 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 se pode resolver em tempo polinomial, então todos podem ser resolvidos em tempo polinomial!

Tempo Polinomial como referência

- Tempo polinomial é a referência que define e separa a classe de problemas que **podem ser resolvidos eficientemente**. Assim, se um problema pode ser resolvido eficientemente, então significa que o seu tempo de execução é polinomial
- Esta avaliação é geralmente medida em termos do tempo de execução do algoritmo, usando a complexidade no pior caso, como uma **função de n** , que é o **tamanho do *input* do problema**
- Um algoritmo de **tempo polinomial** tem tempo de execução da **ordem de $O(n^k)$** , onde **k é uma constante independente de n**
- Um problema é dito ser “**resolúvel em tempo polinomial**” se houver um algoritmo de tempo polinomial que o resolva
- Algumas funções parecem não ser polinomiais, mas podem ser tratadas como tal: e.g. $O(n \log n)$ tem delimitação superior da ordem de $O(n^2)$
- 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

- Reformulação de problemas de optimização
 - Muitos dos problemas práticos que se pretende resolver são problemas de optimização (maximizar ou minimizar alguma métrica)
 - Um problema é dito ser um “**problema de decisão**” se o seu *output* ou resposta deve ser um simples “SIM” ou “NÃO” (ou derivativos do tipo “V/F”, “0/1”, “aceitar/rejeitar”, etc.)
 - Muitos problemas de optimização podem ser expressos em termos de problemas de decisão:

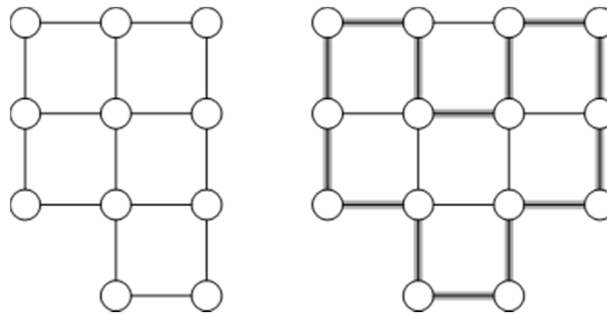
Por exemplo: o problema “qual o menor número de cores que se pode utilizar para colorir um grafo?”, pode ser expresso como “Dado um grafo G e um inteiro k , é possível colorir G com k cores?”
- A classe de problemas **P** é definida por todos os problemas de decisão que podem ser resolvidos em tempo polinomial!



Verificação do Tempo Polinomial

■ Undirected Hamiltonian cycle problem (UHC)

- Dado um grafo G , “é possível determinar se G possui um ciclo que visita todo vértice exactamente uma vez?”



FEUP Universidade do Porto
Faculdade de Engenharia

Classes P e NP - CAL, 2013/14

«#»

Verificação do Tempo Polinomial

- Caso se conheça um ciclo (e.g. $\langle v_3, v_7, v_1, \dots, v_{13} \rangle$), é fácil verificá-lo, por inspecção. Ainda ã sendo possível implementar algoritmo p/resolver o problema, seria fácil “verificar” se G é ou não “Hamiltoniano”
 - O ciclo neste caso é dito ser um “**certificado**”; trata-se de uma informação que permite verificar se uma dada “string” está numa “linguagem” (em *Problemas de Reconhecimento de Linguagem*)
 - Caso seja possível verificar a precisão de um certificado para um problema em tempo polinomial, diz-se que o problema é “verificável em tempo polinomial”
 - Nem todas as “linguagens” gozam da propriedade de serem facilmente verificáveis! Por exemplo: seja o problema definir se um grafo G tem exactamente um ciclo de Hamilton. É fácil verificar se existe pelo menos um ciclo, mas não é simples demonstrar que não há outro!
- A classe de problemas NP é definida por todos os problemas que podem ser verificados por um algoritmo de T polinomial



FEUP Universidade do Porto
Faculdade de Engenharia

Classes P e NP - CAL, 2013/14

«#»

Verificação do Tempo Polinomial

- What does it mean to recognize a language?
 - In formal *Language Theory* it means that given some string you can do some kind of algorithm or computation and either accept or reject the string based on if it is in the language or not.



Verificação do Tempo Polinomial

- **Execução** de tempo polinomial é diferente de **verificação** de tempo polinomial!
 - O circuito de Hamilton é verificável em tempo polinomial, mas acredita-se não haver uma solução executável em tempo polinomial que encontre um circuito de Hamilton
- Por que NP e não VP?
 - O termo NP vem de “*nondeterministic polynomial time*,” relacionado com um programa a executar num “computador não determinístico” capaz de realizar palpites; basicamente, tal arquitectura seria capaz de não deterministicamente conjecturar o valor do certificado e verificar, em tempo polinomial, se uma *string* está na linguagem ou não.



Classes P e NP

- $P \subseteq NP$.
 - Assim, 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 seja, poder verificar se uma solução é correcta em tempo polinomial não garante ou ajuda encontrar um algoritmo que resolva o problema em tempo polinomial
 - $P \neq NP$? Muitos autores acreditam que sim, mas não há provas!
- A classe de problemas **NP-Completo**s é a classe dos problemas mais difíceis de resolver em toda a classe NP.
- Pode haver problemas ainda mais difíceis de resolver que não estejam enquadrados na classe de problemas NP; neste caso, são chamados **problemas NP-difíceis (NP-hard)**

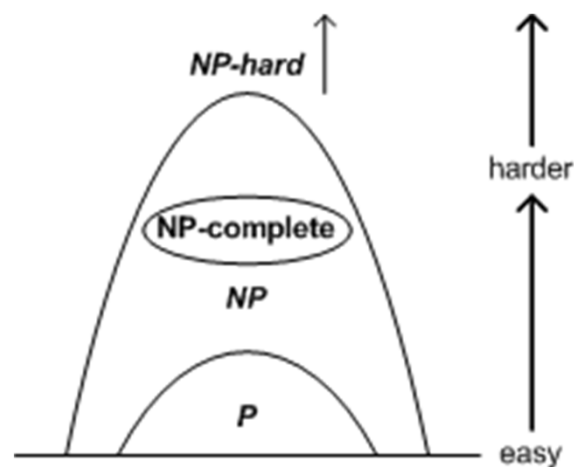


FEUP Universidade do Porto
Faculdade de Engenharia

Classes P e NP - CAL, 2013/14

«#»

Classes P e NP



FEUP Universidade do Porto
Faculdade de Engenharia

Classes P e NP - CAL, 2013/14

«#»

Redução de Problemas NP

- Suponha que há dois problemas, A e B . Sabe-se que A é impossível de ser resolvido em tempo polinomial
- Pretende-se provar que B não pode ser resolvido em tempo polinomial. Como provar ou demonstrar que:

$$(A \notin P) \Rightarrow (B \notin P)$$
- Pode-se tentar provar o contraposto:

$$(B \in P) \Rightarrow (A \in P)$$
 - Em outras palavras, para demonstrar que B não é resolúvel em tempo polinomial, supõe-se que há um algoritmo que resolve B em tempo polinomial, e então deriva-se uma contradição pela demonstração de que A pode ser resolvido em tempo polinomial



Redução de Problemas NP

- Premissa
 - Suponha que há uma subrotina que pode resolver qualquer instância do problema B em tempo polinomial
 - Tenta-se demonstrar que a mesma subrotina pode ser utilizada para resolver A em tempo polinomial
 - Então tem-se “reduzido o problema A no problema B ”!
 - Como se sabe que A não se pode resolver em tempo polinomial, então está-se basicamente a tentar provar que a subrotina não pode existir, implicando que B não pode ser resolvido em tempo polinomial



Redução de Problemas NP

Exemplo

- Sabe-se que o problema UHC é um problema NP-completo! Não se conhece algoritmo de tempo polinomial que o resolva.
- Problema:** suponha um director técnico pede a um dos seus engenheiros para encontrar uma solução polinomial para um problema diferente, nomeadamente o problema de encontrar um ciclo de Hamilton num grafo dirigido (DHC); caso contrário, o demitirá!
- Depois de pensar numa solução, por algum tempo, o engenheiro convence-se de que não se trata de uma solicitação sensata. Será que considerar arestas direccionais tornaria o problema de alguma forma mais fácil? Apesar de ambos (director e eng.) concordarem que UHC é NP-completo, o director está convencido de que DHC é viável e fácil. Como convencê-lo do contrário, e não ser demitido?



Redução de Problemas NP

Solução

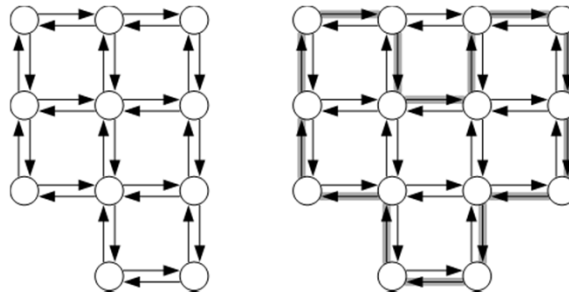
- Pode-se tentar convencer o director de que “se houvesse uma solução eficiente para DHC, demonstrar-se-ia que seria então possível resolver UHC em tempo polinomial!”
- Em particular, usar-se-ia a mesma subrotina usada no DHC para resolver o UHC (transformado numa instância de DHC). Uma vez que ambos conhecem que UHC não é resolúvel, então tal rotina não pode existir. Portanto, DHC também não é resolúvel em tempo polinomial!



Redução de Problemas NP

■ Solução

- Dado um grafo G , criar um grafo dirigido G' pela substituição de cada aresta $\{u, v\}$ por duas arestas dirigidas: (u, v) e (v, u)
- Cada caminho simples em G é, portanto, um caminho simples em G' , e vice-versa. Portanto, G terá um ciclo de Hamilton se, e somente se, G' também o tiver!



Universidade do Porto
Faculdade de Engenharia

Classes P e NP - CAL, 2013/14

<#>

Redução de Problemas NP

■ Solução

- Considera-se q/ a rotina `bool DHC (aDigraph)` resolve o problema!
- Redução UHC \rightarrow DHC

```
bool UHC (G) {
    create digraph G' with the same number of vertices as G
    foreach edge (u, v) in G
        Add edges (u, v) and (v, u) in G'
    return DHC (G')
}
```

- Note-se que nenhum dos problemas foi efectivamente resolvido. Apenas demonstrou-se como converter uma solução para o DHC numa solução para o UHC. Este procedimento é chamado “redução” e é crucial para a teoria dos problemas NP-completos.



Universidade do Porto
Faculdade de Engenharia

Classes P e NP - CAL, 2013/14

<#>

Redução de Problemas NP

■ Definição

- Dados dois problemas, A e B , diz-se que A é polinomialmente redutível a B se, dada uma subrotina de tempo polinomial para B , pode-se utilizá-la para resolver A em tempo polinomial. Quando tal se verifica, expressa-se por

$$A \leq_p B$$

- Lema:** Se $A \leq_p B$ e $B \in P$ então $A \in P$
- Lema:** Se $A \leq_p B$ e $A \notin P$ então $B \notin P$
- Lema:** Se $A \leq_p B$ e $B \leq_p C$ então $A \leq_p C$ (transitividade)



Redução de Problemas NP

■ Definição +formal da classe dos problemas NP- completos

- Um problema de decisão $B \in NP$ é NP-completo se

$$A \leq_p B \mid \forall A \in NP$$
- Assim, se B pode ser resolvido em tempo polinomial, então qualquer outro problema A em NP é resolúvel em tempo polinomial
- Lema:** B é NP-completo se
 - (1) $B \in NP$, e
 - (2) $A \leq_p B$ para algum problema A , se A é NP-Completo



Redução de Problemas NP

- Procedimento:

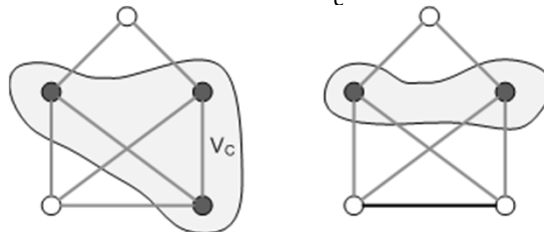
Dado um problema X , prove que o mesmo é pertencente à classe dos problemas NP -completos.

 1. Provar que X está em NP
 2. Seleccionar um problema Y que se sabe ser NP -completo
 3. Definir uma redução de tempo polinomial de Y para X
 4. Provar que, dada uma instância de Y , Y tem uma solução se, e se somente, X tem uma solução



Vertex Cover

- Uma cobertura de vértices de um grafo $G = (V, E)$ é um subconjunto $V_C \subseteq V$, tal que toda aresta $(a, b) \in E$ é incidente em pelo menos um vértice $u \in V_C$.

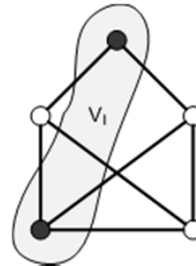


- Vértices em V_C “cobrem” todas as arestas em G .
- Reformulação de VC como um problema de decisão
 - O grafo G tem uma cobertura de vértices de tamanho k ?

Independent Set

- Um conjunto independente de um grafo $G = (V, E)$ é um subconjunto $V_I \subseteq V$, tal que não há dois vértices em V_I que partilham uma aresta de E

- $u, v \in V_I$ não podem ser vizinhos em G .



- Reformulação de VI como um problema de decisão
 - O grafo G tem um conjunto independente de tamanho k ?

Vertex Cover é NP-completo?

- Sabendo-se que o problema de decisão de Conjunto Independente (IS) é NP-completo, ...
- ... pode-se provar que o problema de Cobertura de Vértices também é NP-completo?

Vertex Cover é NP-completo?

- Solução:
 - (1) Provar que VC pertence à classe NP
 - Dado V_C , uma cobertura de vértices de $G = (V, E)$, $|V_C| = k$
 Pode-se verificar em $O(|E| + |V|)$ que V_C é uma cobertura de vértices de G . Como?
 - Para cada vértice $v \in V_C$, remover todas as arestas incidentes
 - Verificar se todas as arestas foram removidas de G .
 - Então, $Vertex\ Cover \in NP!$

Vertex Cover é NP-completo?

- (2) Seleccionar um problema que se conhece ser NP-completo
 - O problema do Conjunto Independente (IS), em grafos, é um problema NP-completo conhecido!
 - Usar IS para provar que VC é NP-completo



Vertex Cover é NP-completo?

- (3) Definir uma redução de tempo polinomial de IS para VC
 - Dada uma instância geral de IS: $G' = (V', E'), k'$
 - Construir uma instância específica de VC: $G = (V, E), k$
 - $V = V'$
 - $E = E'$
 - $(G = G')$
 - $k = |V'| - k'$
 - Esta transformação é polinomial:
 - Tempo constante para contruir $G = (V, E)$
 - $O(|V|)$ para contar o número de vértices

Vertex Cover é NP-completo?

- (4) Provar que G' tem um conjunto independente VI de tamanho k' se, e se somente, VC tem uma cobertura V_c de tamanho k
 - Provar que há um V_I ($|V_I| = k'$) para G' se, e se somente, há um V_C ($|V_C| = k$) para G .
 - Considere dois conjuntos I e J | $I \cap J = \emptyset$, $I \cup J = V = V'$
 - Dada qualquer aresta (u, v) , um dos seguintes casos se verifica:
 1. $u, v \in I$
 2. $u \in I$ e $v \in J$
 3. $u \in J$ e $v \in I$
 4. $u, v \in J$

Vertex Cover é NP-completo?

- (4) Continuação...
 - Assumindo-se que I é um conjunto independente de G' , então:
 - O caso 1 não pode ser (vértices em I não podem ser adjacentes)
 - Nos casos 2 e 3, (u, v) tem *exactamente um* ponto terminal em J
 - No caso 4, (u, v) tem *ambos* os pontos terminais em J
 - Nos casos 2, 3 e 4, (u, v) tem *pelo menos um* ponto terminal em J
 - Então, vértices em J cobrem todas as arestas de G'
 - Também: $|I| = |V| - |J|$ uma vez que $I \cap J = \emptyset$, $I \cup J = V = V'$
 - Assim, se I é um conjunto independente de G' , então J é uma cobertura dos vértices de G' ($= G$)
 - Similarmente, pode-se provar que se J é uma cobertura dos vértices de G' , então I é um conjunto independente de G'

Exemplos de problemas NP-completo

- Alguns exemplos são
 - Ciclos de Hamilton
 - Coloração em grafos
 - Cliques em grafos
 - Subgrafos e supergrafos
 - Árvores de expansão
 - Cortes e conectividade
 - Problemas de fluxo
 - Outros...

Referências e mais informação

- T. Cormen *et al.* (2009) “Introduction to Algorithms.” Cambridge, MA: MIT press.
- 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.



