ACH2043 INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 27

Cap 7.4 – NP-completude

Profa. Ariane Machado Lima
ariane.machado@usp.br

NP-Completude

- Início dos anos 70:
  - Classe de problemas NP para os quais não se conhece uma solução polinomial... mas se existir, poderá ser usada para solucionar em tempo polinomial todos os problemas em NP!
  - Problemas NP-completos

NP-Completude

- Benefícios:
  - A prova de que um problema NP-completo tem uma solução polinomial prova que P = NP
  - A prova de que um problema em NP exige tempo no mínimo exponencial, prova que problemas NP-completos também exigem
  - Se um problema é NP-completo, simplifique-o!

NP-Completude – Definição informal

- Uma linguagem B é NP-completa se satisfaz duas condições:
  - B está em NP
  - Toda linguagem A em NP pode ser decidida usando a solução de B usando “adaptações” polinomiais (para usar a solução de B em A)

NP-Completude – Definição informal

- Uma linguagem B é NP-completa se satisfaz duas condições:
  - B está em NP
  - Toda linguagem A em NP pode ser decidida usando a solução de B usando “adaptações” polinomiais (para usar a solução de B em A)

Função de redução

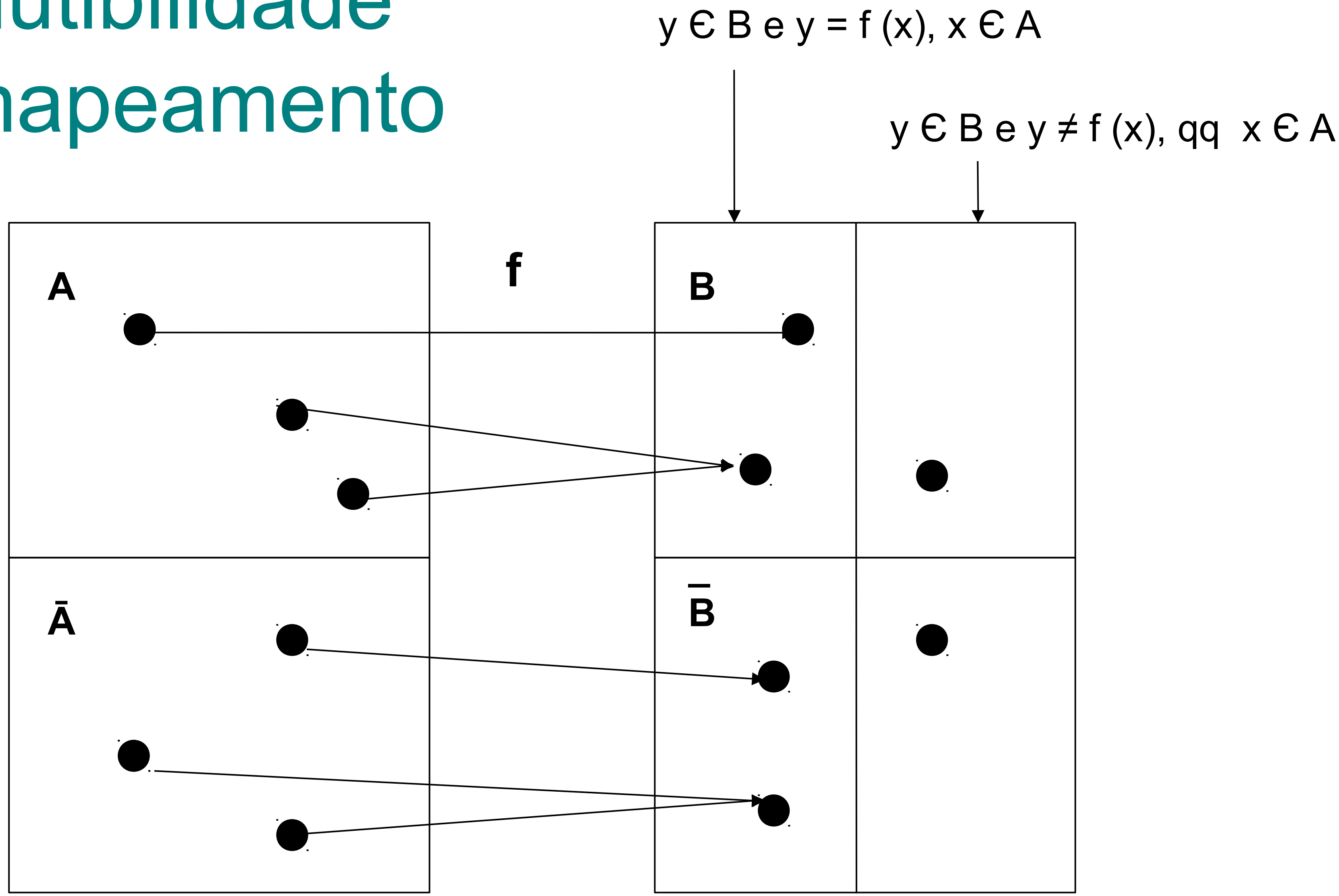
NP-Completude – Definição FORMAL

- Uma linguagem B é NP-completa se satisfaz duas condições:
  - B está em NP
  - Toda linguagem A em NP é redutível em tempo polinomial a B

Redutibilidade em tempo polinomial

- Já vimos redução por mapeamento para provarmos se uma linguagem é ou não decidível
- Como era?

Redutibilidade por mapeamento



Redutibilidade em tempo polinomial

- Agora usaremos a mesma ideia para provar que a decibilidade ocorre em tempo polinomial

Redutibilidade em tempo polinomial

- Uma função f: $\Sigma^* \rightarrow \Sigma^*$ é uma função computável em tempo polinomial se alguma máquina de Turing M de **tempo polinomial**, sobre toda entrada w, pára com exatamente f(w) sobre sua fita

Redutibilidade em tempo polinomial

- A linguagem A é redutível por mapeamento em tempo polinomial à linguagem B ($A \leq_P B$), se existe uma função computável em tempo polinomial $f:\Sigma^* \rightarrow \Sigma^*$ onde para toda w,

w pertence a A <=> f(w) pertence a B.

A função f é denominada a redução de tempo polinomial de A para B.

Teorema

- Se $A \leq_P B$ e $B \in P$, então $A \in P$
- Prova:

Teorema

- Se $A \leq_P B$ e $B \in P$, então $A \in P$
- Prova: Seja M o algoritmo de tempo polinomial que decide B. O seguinte algoritmo N de tempo polinomial decide A:
-
  -

Teorema

- Se $A \leq_P B$ e $B \in P$, então $A \in P$
- Prova: Seja M o algoritmo de tempo polinomial que decide B. O seguinte algoritmo N de tempo polinomial decide A:

N = "Sobre a entrada w:

1. Compute f(w)
2. Rode M sobre a entrada f(w) e dê como saída o que M der como saída "

Teorema

- Se $A \leq_P B$ e $B \in P$, então $A \in P$
- Prova: Seja M o algoritmo de tempo polinomial que decide B. O seguinte algoritmo N de tempo polinomial decide A:

N = “Sobre a entrada w:

1. Compute f(w)
2. Rode M sobre a entrada f(w) e dê como saída o que M der como saída ”

- Por que f deve ser polinomial?
    -

Teorema

- Se $A \leq_P B$ e $B \in P$, então $A \in P$
- Prova: Seja M o algoritmo de tempo polinomial que decide B. O seguinte algoritmo N de tempo polinomial decide A:

N = "Sobre a entrada w:

1. Compute f(w)
2. Rode M sobre a entrada f(w) e dê como saída o que M der como saída "

- Por que f deve ser polinomial?
  - Para que N também seja (não adiantaria M ser polinomial se f não fosse)

Exemplo de redução

- Vamos definir um problema chamado 3SAT e mostrar que ele é redutível em tempo polinomial ao problema do CLIQUE.

SAT – O problema da satisfazibilidade

- Variáveis boolenas: valores falso ou verdadeiro (0 ou 1)
- Operações boolenas: E (^), OU (v), NÃO (¬)
  - Usaremos ! No lugar de ¬
- Fórmula booleana: expressão contendo variáveis e operações booleanas
  - Ex: Φ = (!x ^ y) v (x ^ !z)
- Fórmula booleana satisfazível: existem valores das variáveis para os quais a fórmula é igual a 1
  - Ex: x = 0, y = 1, z = 0 satisfaz Φ
- SAT = { <Φ>: Φ é uma fórmula booleana satisfazível}

3SAT – O problema da satisfazibilidade

- Literal: uma variável booleana ou sua negação
  - ex: x ou !x
- Cláusula: fórmula contendo apenas “OUs” de literais
  - ex: x1 v !x2 v !x3 v x4
- Forma normal conjuntiva (fnc-fórmula): cláusulas conectadas por “Es” (conjunção de disjunções)
  - Ex: (x1 v !x2 v !x3 v x4) ^ (x3 v !x5 v x6) ^ (x3 v !x6)
- 3fnc-fórmula: todas as cláusulas têm exatamente 3 literais
  - Ex: (x1 v !x2 v !x3) ^ (x3 v !x5 v x6) ^ (x3 v !x6 v x4) ^ (x4 v x5 v x6)
- 3SAT = { <Φ>: Φ é uma 3fnc-fórmula satisfazível}
  - Isto é,

3SAT – O problema da satisfazibilidade

- Literal: uma variável booleana ou sua negação
  - ex: x ou !x
- Cláusula: fórmula contendo apenas “OUs” de literais
  - ex: x1 v !x2 v !x3 v x4
- Forma normal conjuntiva (fnc-fórmula): cláusulas conectadas por “Es” (conjunção de disjunções)
  - Ex: (x1 v !x2 v !x3 v x4) ^ (x3 v !x5 v x6) ^ (x3 v !x6)
- 3fnc-fórmula: todas as cláusulas têm exatamente 3 literais
  - Ex: (x1 v !x2 v !x3) ^ (x3 v !x5 v x6) ^ (x3 v !x6 v x4) ^ (x4 v x5 v x6)
- 3SAT = { <Φ>: Φ é uma 3fnc-fórmula satisfazível}
  - Isto é, cada cláusula de Φ deve ter pelo menos um literal valendo 1

Teorema

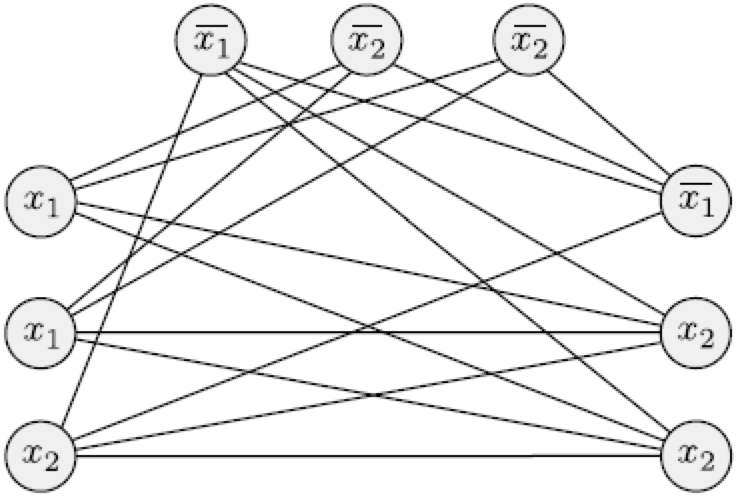
- 3SAT é redutível em tempo polinomial a CLIQUE
- Ideia da prova:
  - converter a fórmula em um grafo
  - $\Phi = (a1 \vee b1 \vee c1) \wedge (a2 \vee b2 \vee c2) \wedge \ldots (ak \vee bk \vee ck)$

Teorema

- 3SAT é redutível em tempo polinomial a CLIQUE
- Prova:
  - Seja uma fórmula na fnc de k cláusulas
  - Os nós em G são organizados em k grupos de três nós (cada grupo corresponde a uma cláusula, cada nó a um literal)
  - Arestas conectam todos os pares de nós, exceto:
    - Nós do mesmo grupo
    - Nós contraditórios (ex: x1 e !x1)
  - Um k-clique corresponde a um conjunto de literais que podem assumir valor 1 e satisfazer a fórmula

Exemplo

Φ = (x1 v x1 v x2) ^ (!x1 v !x2 v !x2) (!x1 v x2 v x2)



3SAT e CLIQUE

- Se $A \leq_P B$ e $B \in P$, então $A \in P$
- $3SAT \leq_P CLIQUE$
- Logo, se CLIQUE for decidível em tempo polinomial então 3SAT também será
- Relacionamos as complexidades de 2 problemas
- É possível relacionar as complexidades de uma classe inteira de problemas...

NP-Completude – Definição FORMAL

- Uma linguagem B é NP-completa se satisfaz duas condições:
  - B está em NP
  - Toda linguagem A em NP é redutível em tempo polinomial a B

NP-Completude – Definição FORMAL

- Uma linguagem B é NP-completa se satisfaz duas condições:
  - B está em NP
  - Toda linguagem A em NP é redutível em tempo polinomial a B (isto é, B é **NP-difícil** ou **NP-hard**)

SAT é NP-completo

- Teorema de Cook-Levin: SAT é NP-completo
- Precisamos provar que
  - SAT está em NP
  - Qualquer problema em NP é redutível em tempo polinomial a SAT (SAT é NP-difícil)

SAT é NP-completo

- SAT está em NP
  - Prova?

SAT é NP-completo - PROVA

- SAT está em NP
  - Uma MT não-determinística pode, em cada ramo da computação, testar uma atribuição de valores. Cada ramo roda em tempo polinomial
  - Ou, similarmente, um verificador polinomial consegue verificar uma dada atribuição de valores

SAT é NP-completo - PROVA

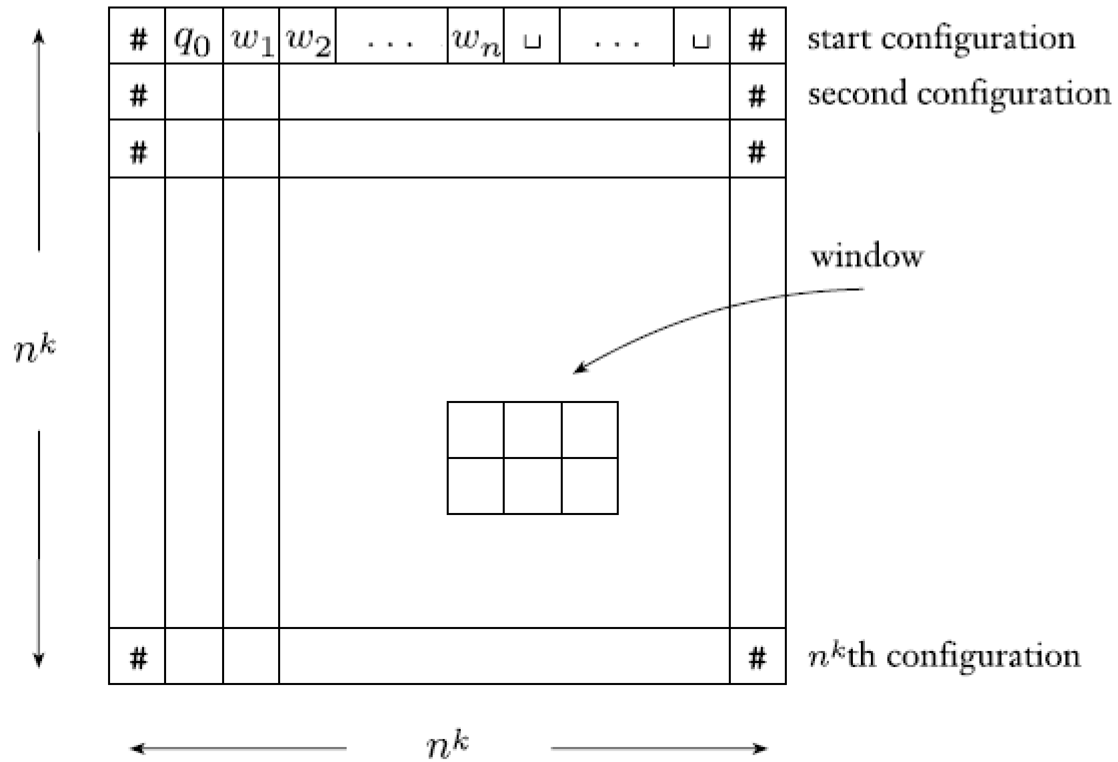
- Qualquer problema em NP é redutível em tempo polinomial a SAT (SAT é NP-difícil)
  - Mais difícil...
  - Ideia: usar uma expressão booleana para simular qualquer máquina de Turing não-determinística
  - Não parece ser tão impossível se lembrar que circuitos de computadores eletrônicos são baseados nas operações booleanas...

SAT é NP-completo - PROVA

- Seja A uma linguagem em NP, e uma MT não-determinística N que decide A em tempo n^k para alguma constante k
- Um **tableau** para N sobre w é uma tabela n^k x n^k:
  - Linhas: configurações de um ramo de N
  - Primeira e última coluna: “#”
  - Linha 1 tem a configuração inicial
  - Cada linha i é uma configuração originada da configuração i-1
  - Tableau de aceitação: alguma linha tem uma configuração de aceitação

SAT é NP-completo - PROVA

- Um **tableau** para N sobre w é uma tabela n^k x n^k:



SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT:
  - w => f(w)
  - w é a entrada de A, f(w) é uma expressão booleana Φ
  - Variáveis de Φ: $x_{i,j,s}$, onde i,j é a posição de uma célula do tableau e s ($s \in C$, $C = Q \cup \Gamma \cup \{\#\}$).
    - $x_{i,j,s} = 1$ se t[i,j] = s e 0 caso contrário

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Agora precisamos montar uma expressão booleana que corresponda a um tableau de aceitação:
  -
  -
  -
  -
  -

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Agora precisamos montar uma expressão booleana que corresponda a um tableau de aceitação:
  - Φcélula ^ Φinício ^ Φmovimento ^ Φaceita
  -
  -
  -
  -

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Agora precisamos montar uma expressão booleana que corresponda a um tableau de aceitação:
  - Φcélula ^ Φinício ^ Φmovimento ^ Φaceita
  - Φcélula: há exatamente uma variável ligada para cada célula
  - Φinício: a primeira linha possui uma configuração inicial
  - Φmovimento: cada linha i corresponde a uma configuração legalmente originada da configuração da linha i-1 (pela função de transição de N)
  - Φaceita: o tableau é de aceitação

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Φcélula: há exatamente uma variável ligada para cada célula

$$\Phi\text{célula} = \wedge_{1 \leq i\ ,\ j \leq n^k} [(\vee_{s \in C}\ x_{i,j,s}) \wedge (\wedge_{s \in C \text{ e } s \neq t}\ (!x_{i,j,s} \vee !x_{i,j,t}\ )]$$

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Φcélula: há exatamente uma variável ligada para cada célula

$$\Phi célula = \wedge_{1 \leq i\ ,\ j \leq n^k} [(\vee_{s \in C}\ x_{i,j,s}) \wedge (\wedge_{s \in C \text{ e } s \neq t}\ (!x_{i,j,s} \vee !x_{i,j,t}\ )]$$

t[i,j] é não vazia

t[i,j] só possui um símbolo, ou seja, para cada par de variáveis em i,j uma delas pelo menos é falsa

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Φinício: a primeira linha possui uma configuração inicial

$$\Phi\text{início} = x_{1,1,\#} \wedge x_{1,2,q0} \wedge x_{1,3,w1} \wedge x_{1,4,w2} \wedge \ldots \wedge x_{1,n+2,wn} \wedge x_{1,n+3,branco} \wedge \ldots \wedge x_{1,n^k-1,branco} \wedge x_{1,n^k,\#}$$

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Φmovimento: cada linha i corresponde a uma configuração legalmente originada da configuração da linha i-1 (pela função de transição de N)
  - Cada janela 2x3 é legal
  - Ex.

Expresso formalmente, $\delta(q_1, \mathtt{a}) = \{(q_1,\mathtt{b},\mathrm{D})\}$ e $\delta(q_1, \mathtt{b}) = \{(q_2,\mathtt{c},\mathrm{E}), (q_2,\mathtt{a},\mathrm{D})\}$. Exemplos de janelas legais para essa máquina são mostradas na Figura 7.39.

(a)

a	q_1	b
q_2	a	c

(b)

a	q_1	b
a	a	q_2

(c)

a	a	q_1
a	a	b

(d)

#	b	a
#	b	a

(e)

a	b	a
a	b	q_2

(f)

b	b	b
c	b	b

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Φmovimento: cada linha i corresponde a uma configuração legalmente originada da configuração da linha i-1 (pela função de transição de N)
  - Cada janela 2x3 é legal
  - $\Phi_{movimento} = \wedge_{1 < i \leq n^k,\ 1 < j < n^k}$ (a janela (i,j) é legal)
  - $= \wedge_{1 < i \leq n^k,\ 1 < j < n^k} \left(\vee_{a1,\ldots,a6 \text{ é uma janela legal}}\ x_{i,j-1,a1} \wedge x_{i,j,a2} \wedge x_{i,j+1,a3} \wedge x_{i+1,j-1,a4} \wedge x_{i+1,j,a5} \wedge x_{i+1,j,a6}\right)$

SAT é NP-completo - PROVA

- Redução e tempo polinomial de A para SAT (cont.)
  - Φaceita: o tableau é de aceitação

$\Phi_{aceita} = \vee_{1 \leq i\ ,\ j \leq n^k}\ x_{i\ ,\ j\ ,\ q_{aceita}}$

SAT é NP-completo - PROVA

- A redução é mesmo polinomial.
- Vejamos o tamanho de Φ:
  - Número de variáveis:
    - Tableau tem $n^k \times n^k = n^{2k}$ células
    - Cada célula tem l variáveis (l = número de símbolos de fita de N + número de estados de N + 1, ou seja, não depende do comprimento de w, logo é considerado uma constante
    - Número de variáveis: $O(n^{2k})$

  - Tamanho de cada parte de Φ:

SAT é NP-completo - PROVA

Φcélula: há exatamente uma variável ligada para cada célula

$$\Phi\text{célula} = \wedge_{1 \le i,j \le n^k} [(\vee_{s \in C}\, x_{i,j,s}) \wedge (\wedge_{s \in C \text{ e } s \neq t} (!x_{i,j,s} \vee !x_{i,j,t}))]$$

Tamanho:

SAT é NP-completo - PROVA

Φcélula: há exatamente uma variável ligada para cada célula

$$\Phi\text{célula} = \wedge_{1\leq i,j\leq n^k} [(\vee_{s\in C}\ x_{i,j,s}) \wedge (\wedge_{s\in C \text{ e } s\neq t}\ (!x_{i,j,s} \vee !x_{i,j,t}\ )]$$

Tamanho: $O(n^{2k})$

SAT é NP-completo - PROVA

Φinício: a primeira linha possui uma configuração inicial

Φinício = $x_{1,1,\#} \wedge x_{1,2,q0} \wedge x_{1,3,w1} \wedge x_{1,4,w2} \wedge \ldots \wedge x_{1,n+2,wn} \wedge$

$x_{1,n+3,branco} \wedge \ldots \wedge x_{1,n^k-1,branco} \wedge x_{1,n^k,\#}$

Tamanho:

SAT é NP-completo - PROVA

Φinício: a primeira linha possui uma configuração inicial

Φinício = $x_{1,1,\#} \wedge x_{1,2,q0} \wedge x_{1,3,w1} \wedge x_{1,4,w2} \wedge \ldots \wedge x_{1,n+2,wn} \wedge$

$x_{1,n+3,branco} \wedge \ldots \wedge x_{1,n^k-1,branco} \wedge x_{1,n^k,\#}$

Tamanho: $O(n^k)$

SAT é NP-completo - PROVA

Φmovimento = $\wedge_{1 < i \leq n^k,\ 1 < j < n^k}$ (a janela (i,j) é legal)

= $\wedge_{1 < i \leq n^k,\ 1 < j < n^k}$ $(\vee_{a1,\dots,a6 \text{ é uma janela legal}}\ x_{i,j-1,a1} \wedge x_{i,j,a2} \wedge x_{i,j+1,a3} \wedge x_{i+1,j-1,a4} \wedge x_{i+1,j,a5} \wedge x_{i+1,j,a6})$

Tamanho:

SAT é NP-completo - PROVA

Φmovimento = $\wedge_{1 < i \leq n^k,\ 1 < j < n^k}$ (a janela (i,j) é legal)

$= \wedge_{1 < i \leq n^k,\ 1 < j < n^k} \left(\vee_{a1,\ldots,a6 \text{ é uma janela legal}}\ x_{i,j-1,a1} \wedge x_{i,j,a2} \wedge x_{i,j+1,a3} \wedge x_{i+1,j-1,a4} \wedge x_{i+1,j,a5} \wedge x_{i+1,j,a6} \right)$

Tamanho: $O(n^{2k})$

SAT é NP-completo - PROVA

Φaceita: o tableau é de aceitação

$\Phi aceita = \vee_{1 \leq i , j \leq n^k} x_{i , j , qaceita}$

Tamanho:

SAT é NP-completo - PROVA

Φaceita: o tableau é de aceitação

$\Phi_{aceita} = \vee_{1 \leq i\ ,\ j \leq n^k}\ x_{i\ ,\ j\ ,\ qaceita}$

Tamanho: $O(n^{2k})$

SAT é NP-completo - PROVA

Tamanho total de Φ:

SAT é NP-completo - PROVA

Tamanho total de Φ: $O(n^{2k})$

E está completa a prova.

Teorema

Se B for NP-completa e B Є P, então P = NP

Ideia da Prova:

Teorema

Se B for NP-completa e B Є P, então P = NP

Ideia da Prova: pela redutibilidade em tempo polinomial

Teorema de Cook-Levin escrito de outra forma

SAT $\in$ P se e somente se P = NP

3-SAT é NP-completa

- SAT $\leq_p$ 3-SAT
- Mas aqui, veremos simplesmente que a prova do terema de Cook-Levin (SAT é NP-completa) poderia ser adaptada para 3fnc-fórmulas.

3-SAT é NP-completa

- $\Phi = \Phi\text{célula} \wedge \Phi\text{início} \wedge \Phi\text{movimento} \wedge \Phi\text{aceita}$
- $\Phi\text{célula} = \wedge_{1 \leq i\,,\, j \leq n^k} [(\vee_{s \in C}\, x_{i,j,s}) \wedge (\wedge_{s \in C \text{ e } s \neq t} (!x_{i,j,s} \vee !x_{i,j,t}\,)]$
- $\Phi\text{início} = x_{1,1,\#} \wedge x_{1,2,q0} \wedge x_{1,3,w1} \wedge x_{1,4,w2} \wedge \ldots \wedge x_{1,n+2,wn} \wedge$

  $x_{1,n+3,\text{branco}} \wedge \ldots \wedge x_{1,n^k-1,\text{branco}} \wedge x_{1,n^k,\#}$
- $\Phi\text{movimento} = \wedge_{1 < i \leq n^k,\ 1 < j < n^k}$

$(\vee_{a1,\ldots,a6 \text{ é uma janela legal}}\, x_{i,j-1,a1} \wedge x_{i,j,a2} \wedge x_{i,j+1,a3} \wedge x_{i+1,j-1,a4} \wedge x_{i+1,j,a5} \wedge x_{i+1,j,a6})$

- $\Phi\text{aceita} = \vee_{1 \leq i\,,\, j \leq n^k}\, x_{i\,,\, j\,,\, \text{qaceita}}$

3-SAT é NP-completa

- Duas questões a serem acertadas
  - Cada subfórmula deve ser uma conjunção de disjunções (E de OUs)
  - Cada subfórmula deve conter exatamente 3 literais

3-SAT é NP-completa

- Cada subfórmula deve ser uma conjunção de disjunções (E de OUs)
  - Aplica-se a distributiva (revisão no cap. 0):

  P v (Q ^ R) = (P v Q) ^ (P v R)

3-SAT é NP-completa

- Cada subfórmula deve conter exatamente 3 literais
  - Se uma cláusula tiver apenas um ou dois literais: replique um deles até completar três
    - Ex: (x1) = (x1 v x1 v x1)
    - (x1 v x2) = (x1 v x2 v x1) = (x1 v x2 v x2)
  - Se uma cláusula tiver mais que l > 3 literais: divida-a em l-2 cláusula com variáveis extras da seguinte forma:

$(x_1 \vee x_2 \vee \ldots \vee x_l) =$

$(x_1 \vee x_2 \vee z_1) \wedge (!z_1 \vee x_3 \vee z_2) \wedge \ldots \wedge (!z_{l-3} \vee x_{l-1} \vee x_l)$

3-SAT é NP-completa

- Assim, a fórmula que representa um tableau de aceitação pode ser escrita como uma 3fnc-fórmula
- Logo, fica provado que a linguagem 3-SAT é NP-completa

Problemas NP-completos adicionais

- Há vários problemas NP-completos
- A maioria dos problemas NP está em P ou é NP-completo
- Se você está procurando um algoritmo polinomial para um novo problema NP, primeiro tente provar que ele é NP-completo

Como provar que um problema B é NP-completo

- A princípio, teríamos que provar que
  - B está em NP
  - Todo problema em NP é redutível em tempo polinomial a B
- Foi o que fizemos com SAT e 3SAT
- Mas o "TODO" pode ser complicado na prova para muitos problemas
- Estratégia:
  - partir de um problema A que já se sabe que é NP-completo (ex: SAT ou 3SAT)
  - achar uma redução em tempo polinomial de A para B

Como provar que um problema B é NP-completo

- Se B pertence a NP e

  se A $\leq_p$ B e A é NP-completo

então B é NP-completo

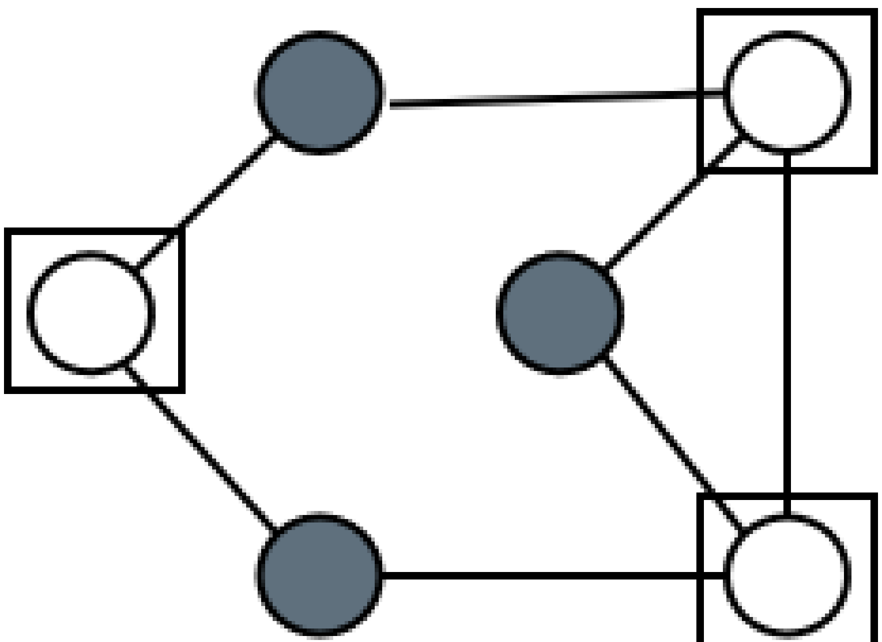
- Ex: CLIQUE é NP-completo:

  CLIQUE pertence a NP e 3SAT $\leq_p$ CLIQUE

- Precisamos identificar estruturas no problema alvo que simulem as variáveis e cláusulas booleanas
- Podemos fazer a redução a partir de qualquer problema NP-completo (embora 3SAT seja bastante usado)
  - Por isso é importante conhecer alguns

O problema da cobertura de vértices

Uma **cobertura** de um grafo é qualquer conjunto de vértices que contenha pelo menos uma das pontas de cada aresta. Em outras palavras, um conjunto X de vértices é uma cobertura se toda aresta do grafo tem pelo menos uma de suas pontas em X.



O problema da cobertura de vértices

- O tamanho de uma cobertura de vértices é igual ao número de vértices da cobertura

- COB-VERT = { <G, k> | G é um grafo não-direcionado que tem uma cobertura de vértices de tamanho k}

- Teorema: COB-VERT é NP-completa

COB-VERT é NP-completa - PROVA

- COB-VERT pertence a NP
  - Certificado: uma cobertura de tamanho k
- 3SAT $\leq_p$ COB-VERT (Ideia da prova)
  - Um grafo que simule uma 3fnc-fórmula Φ
  - Φ será satisfazível sse o grafo tiver uma cobertura de tamanho k
  - Φ: variáveis que assumem valor verdadeiro ou falso
    - G: dois nós de cada aresta (um deles ao menos tem que aparecer na cobertura – verdadeiro ou falso)
  - Φ: cada cláusula com 3 literais, pelo menos um verdadeiro
    - G: grupos de 3 nós conectados por arestas, pelo menos 2 nós inclusos na cobertura

COB-VERT é NP-completa – PROVA – A redução

- Para cada variável x, nós adicionais x e !x conectados por uma aresta (só um literal será verdadeiro, só um nó participará da cobertura – o correspondente ao literal verdadeiro)
- Cada cláusula vira 3 nós conectados entre por arestas, e arestas conectando-os a nós adicionais idêntidos – escolhe-se um correspondente a um literal verdadeiro, e os outros 2 vão para a cobertura
- Número total de nós: 2m + 3l, onde m é o número de variáveis e l é o número de cláusulas
- k = m + 2l
- Figura 7.45

COB-VERT é NP-completa – PROVA – A redução funciona

- Φ será satisfazível sse o grafo tiver uma cobertura de tamanho k

COB-VERT é NP-completa – PROVA – A redução funciona

- Φ é satisfazível => o grafo tem uma cobertura de tamanho k
  - Cada nó adicional referente ao literal verdadeiro vai para a cobertura
  - Em cada cláusula, selecionamos um literal verdadeiro. Colocamos os outros 2 nós correspondentes na cobertura
  - Até agora, temos k nós. Esses k nós cobrem todas arestas:
    - Arestas entre nós adicionais estão cobertas
    - Arestas entre os 3 nós de cada cláusula estão cobertas
    - Arestas entre nós adicionais e nós de cláusulas estão cobertas

COB-VERT é NP-completa – PROVA – A redução funciona

- Φ é satisfazível <= o grafo tem uma cobertura de tamanho k
  - A cobertura tem que conter um nó correspondente a cada par de nós adicionais e dois nós de cada trio correspondentes às cláusulas
  - Atribuímos Verdadeiro ao literais selecionados dentre os nós adicionais.
  - Essa atribuição satisfaz Φ, pois a cobertura deve conter 2 nós de cada trio (cláusula) para cobrir as arestas que conectam nós adicionais a nós de trios (a aresta do terceiro nó é coberta pelo nó adicional verdadeiro, que satisfaz a cláusula correspondente)

COB-VERT é NP-completa – OUTRA PROVA

- CLIQUE $\leq_p$ COB-VERT (Ideia da prova):
  - Dado um grafo G = (V,E) não-direcionado, onde queremos encontrar um k-clique
  - Redução:
    -
    -
  -

COB-VERT é NP-completa – OUTRA PROVA

- CLIQUE $\leq_p$ COB-VERT (Ideia da prova):
  - Dado um grafo G = (V,E) não-direcionado, onde queremos encontrar um k-clique
  - Redução:
    - G complementar = (V, !E), onde !E é o conjunto de todas as arestas entre nós de V que NÃO estão em G
    -
  -

COB-VERT é NP-completa – OUTRA PROVA

- CLIQUE $\leq_p$ COB-VERT (Ideia da prova):
  - Dado um grafo G = (V,E) não-direcionado, onde queremos encontrar um k-clique
  - Redução:
    - G complementar = (V, !E), onde !E é o conjunto de todas as arestas entre nós de V que NÃO estão em G
    - Ache uma cobertura de vértices em G complementar de tamanho |V|-k
  - Ideia:

COB-VERT é NP-completa – OUTRA PROVA

- CLIQUE $\leq_p$ COB-VERT (Ideia da prova):
  - Dado um grafo G = (V,E) não-direcionado, onde queremos encontrar um k-clique
  - Redução:
    - G complementar = (V, !E), onde !E é o conjunto de todas as arestas entre nós de V que NÃO estão em G
    - Ache uma cobertura de vértices em G complementar de tamanho |V|-k
  - Ideia: há um clique de tamanho k se existem |V|-k nós que, por não terem certas arestas, impedem que o grafo seja completo

CAMHAM é NP-Completo

CAMHAM = {<G,s,t> : G é um grafo direcionado com um caminho hamiltoniano de s a t}

Já vimos que CAMHAM ϵ NP

Agora mostraremos que 3SAT $\leq_p$ CAMHAM

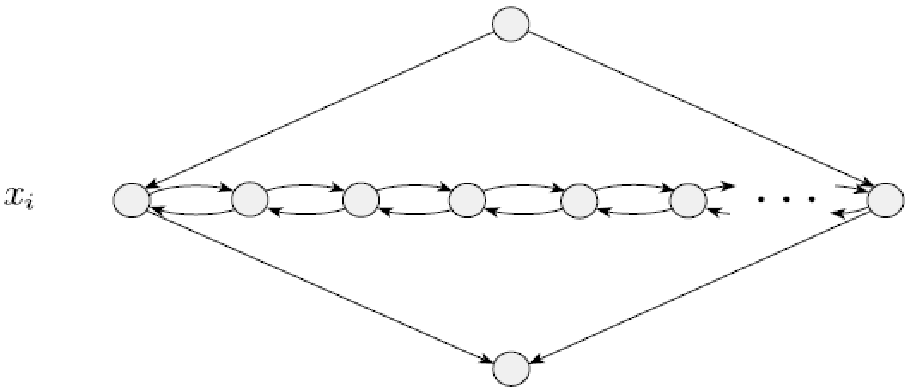
3SAT $\leq_p$ CAMHAM

Φ será satisfazível sse o grafo tiver um caminho hamiltoniano de s para t

Engrenagens de variáveis:

- uma estrutura de diamante para cada variável

- cada diamante i pode ser percorrido da esquerda para a direita (se xi for verdadeiro) ou da direita para a esquerda (se xi for falso)



Φ será satisfazível sse o grafo tiver um caminho hamiltoniano de s para t

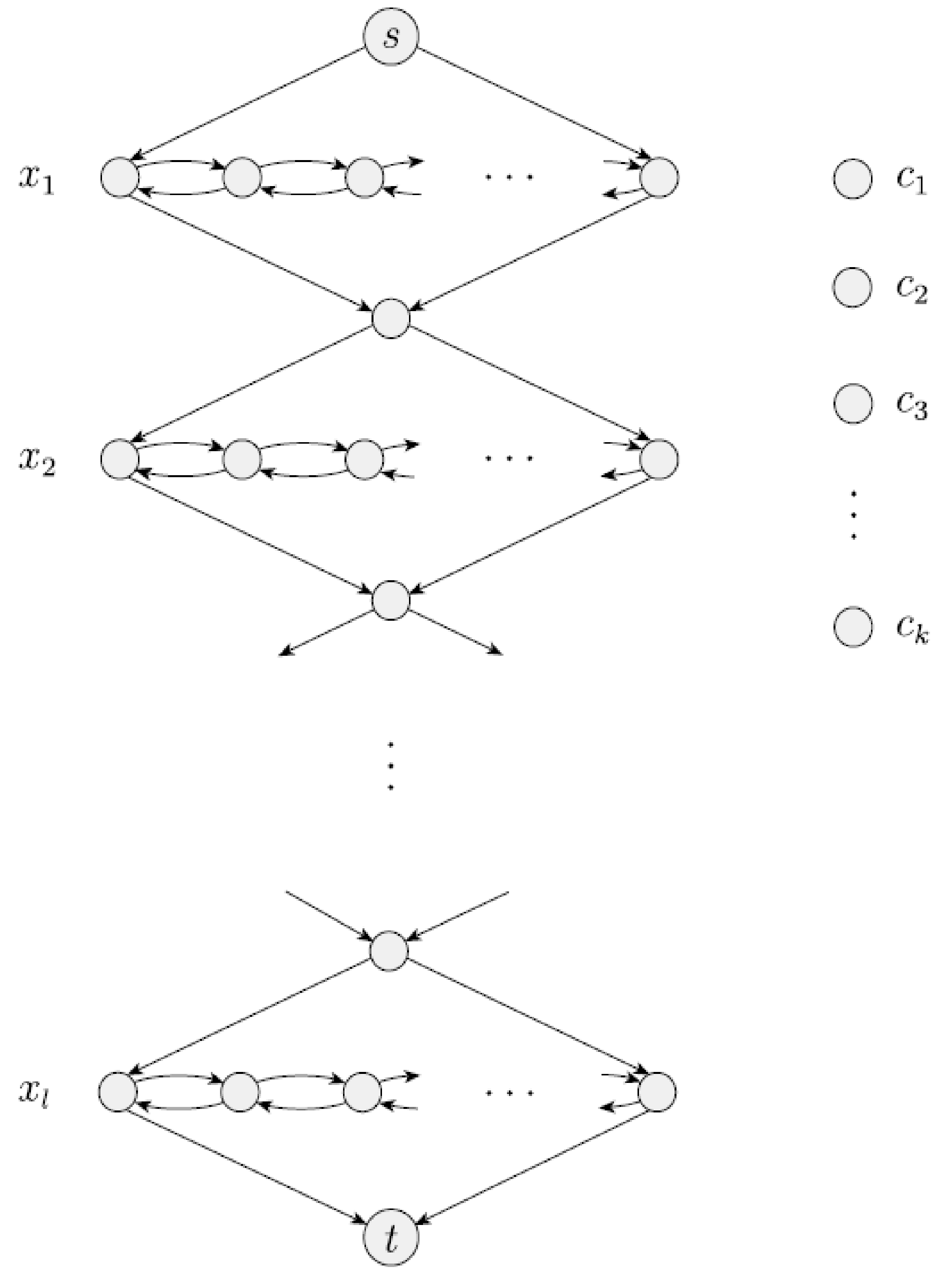
Engrenagens de variáveis:

- uma estrutura de diamante para cada variável

- cada diamante i pode ser percorrido da esquerda para a direita (se xi for verdadeiro) ou da direita para a esquerda (se xi for falso)

- primeiro nó do primeiro diamante: s

- último nó do último diamante: t



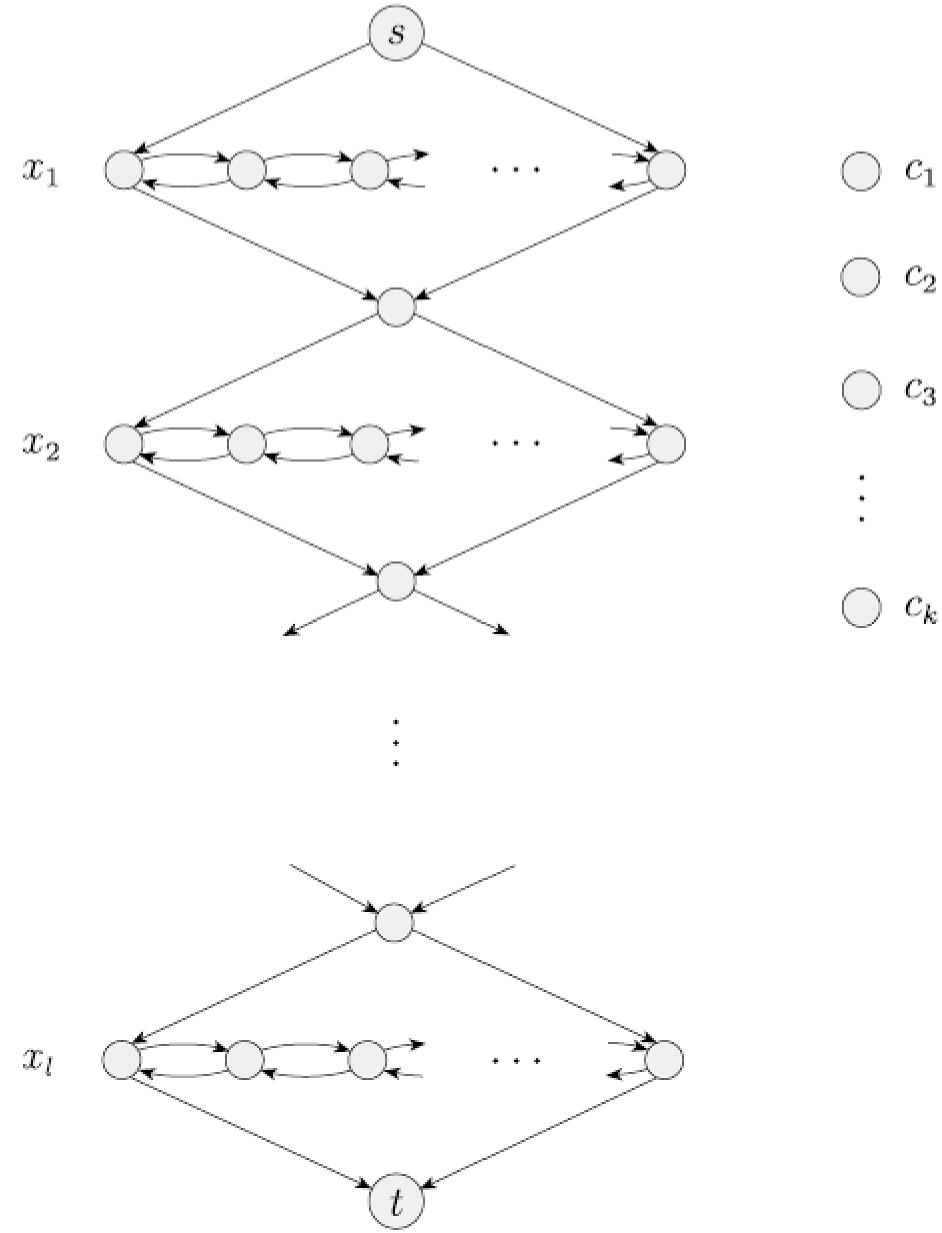
Ligação entre variáveis e cláusulas:

- cada diamante tem uma linha horizontal de 3k+3 nós com arestas indo e vindo entre nós adjacentes:

    - 2 nós das extremidades

    - nó separador, dupla de nós referentes à cláusula c1, nó separador, ..., nó separador, dupla de nós referentes à cláusula ck, nó separador

- se xi (ou !xi) está em cj, o j-ésimo par do i-ésimo diamante conecta-se ao nó cj, o primeiro indo e o segundo voltando (ou o segundo indo e o primeiro voltando)



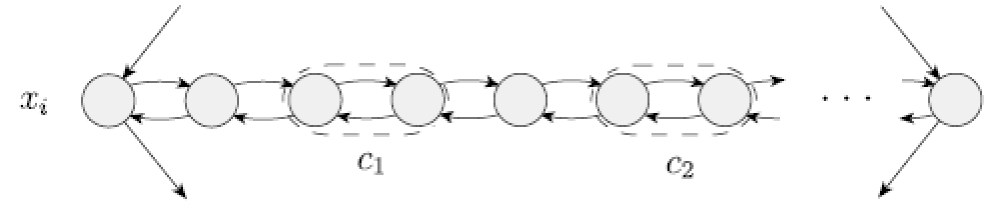
Ligação entre variáveis e cláusulas:

- cada diamante tem uma linha horizontal de 3k+3 nós com arestas indo e vindo entre nós adjacentes:

    - 2 nós das extremidades

    - nó separador, dupla de nós referentes à cláusula c1, nó separador, ..., nó separador, dupla de nós referentes à cláusula ck, nó separador

- se xi (ou !xi) está em cj, o j-ésimo par do i-ésimo diamante conecta-se ao nó cj, o primeiro indo e o segundo voltando (ou o segundo indo e o primeiro voltando)



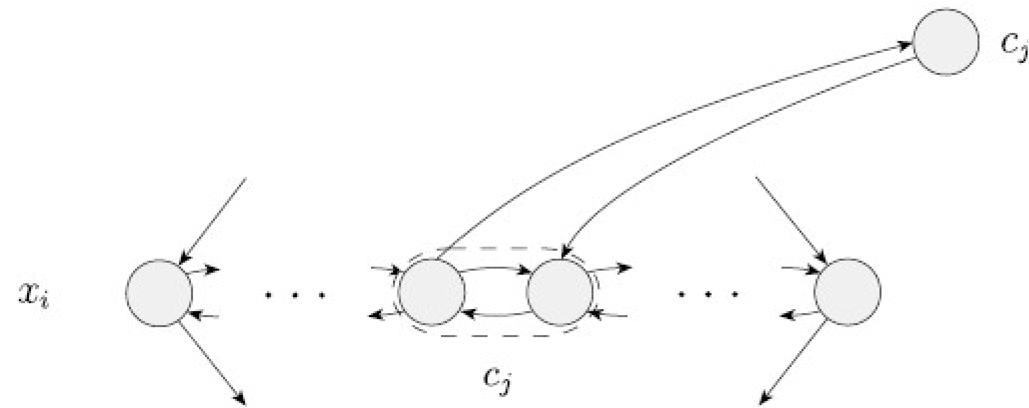


FIGURA 7.51
As arestas adicionais quando a cláusula c_j contém x_i

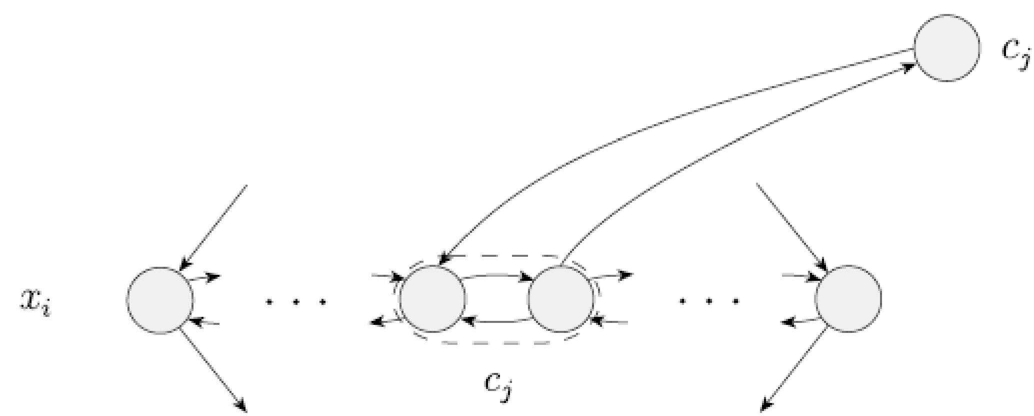


FIGURA 7.52
As arestas adicionais quando a cláusula c_j contém $\overline{x_i}$

$3SAT \leq_p CAMHAM$

A construção é polinomial

A construção funciona, ou seja,

Φ será satisfazível $\Leftrightarrow$ o grafo tiver um caminho hamiltoniano de s para t

3SAT $\leq_p$ CAMHAM

Prova:

Φ é satisfazível => o grafo possui um caminho hamiltoniano de s para t

Para cada variável:

- se xi = v (ou = f) todos os nós do i-ésimo diamante são percorridos (ziguezague da esquerda para a direita pertencerá ao caminho)

Como Φ é satisfazível existe pelo menos um literal verdadeiro em cada cláusula

Para cada cláusula cj, escolher UM dos literais verdadeiros (xi ou !xi) e fazer o desvio do (i-ésimo
nó cj e voltar (sai do prim i
= v, e vice-versa caso cor
posição correta para com

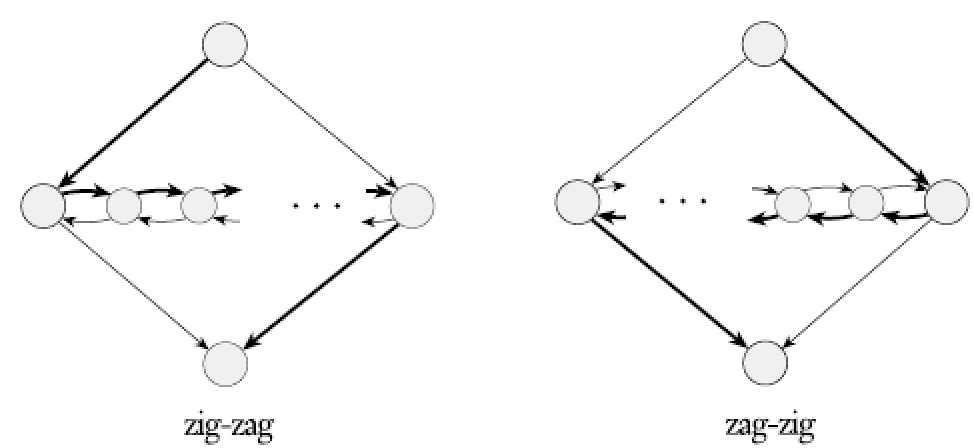


FIGURA 7.53
Zigue-zagueando e zague-zigueando através de um diamante, como determinado pela atribuição satisfeitora

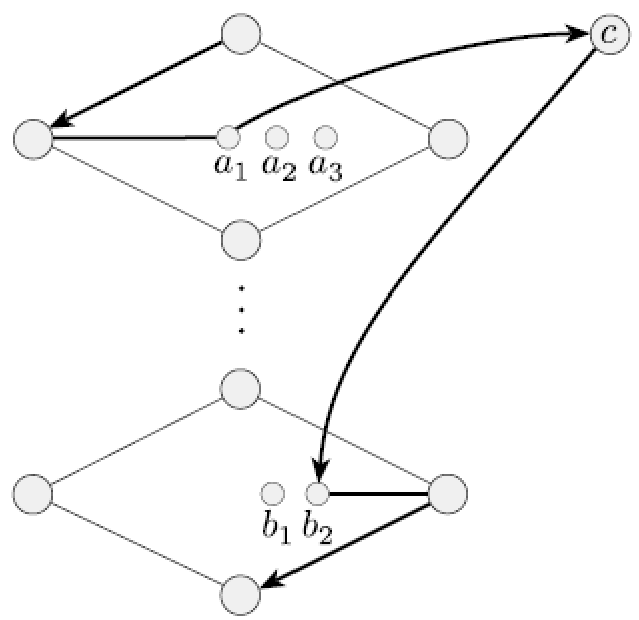
3SAT $\leq_p$ CAMHAM

Prova:

Φ é satisfazível <= o grafo possui um caminho hamiltoniano de s para t

Caminho hamiltoniano **normal**: passa pelos diamantes na ordem do superior para o inferior.

Ex de caminho hamiltoniano não normal:



3SAT $\leq_p$ CAMHAM

Prova:

Φ é satisfazível <= o grafo possui um caminho hamiltoniano de s para t

Mostraremos que:

1-) vale se o caminho hamiltoniano é normal

2-) Todo caminho hamiltoniano neste grafo é normal

3SAT $\leq_p$ CAMHAM

Prova:

Φ é satisfazível <= o grafo possui um caminho hamiltoniano de s para t

1-) vale se o caminho hamiltoniano é normal:

se o caminho passa pelo i-ésimo diamante da esquerda para a direita, então xi = v

se o caminho passa pelo i-ésimo diamante da direita para a esquerda, então xi = f

se o caminho desvia do i-ésimo diamante para cj, então cj é satisfeita (e isso certamente acontece para algum i, pois cj deve ser alcançado)

2-) Todo caminho hamiltoniano neste grafo é normal

Assumimos que existe um caminho hamiltoniano não normal (Fig. 7.54)

a2 ou a3 é um nó separador

Se a2 é separador:

arestas entrando em a2: de a1 e de a3 apenas

Se a3 é separador:

a1 e a2 é do mesmo par

arestas entrando em a2: de a1, a3 e de c

Nos dois casos: a2 não pertence ao caminho hamiltoniano:

- não chegam arestas de a1 e de c, porque já há arestas partindo deles para outros nós

- não chega aresta de a3, pois a3 é o único nó disponível para ser destino de uma aresta saindo de a2

Logo, não existe caminho hamiltoniano não normal

CAMHAM não-direcionado

CAMHAMN = {<G,s,t>: G é um grafo NÃO direcionado no qual existe um caminho hamiltoniano entre os nós s e t}

TEOREMA: CAMHAMN é NP-Completo

PROVA:

- CAMHAMN pertence a NP
- CAMHAM $\leq_p$ CAMHAMN

CAMHAM não-direcionado

CAMHAMN = {<G,s,t>: G é um grafo NÃO direcionado no qual existe um caminho hamiltoniano entre os nós s e t}

TEOREMA: CAMHAMN é NP-Completo

PROVA:

- CAMHAMN pertence a NP (o caminho é o certificado)
- CAMHAM $\leq_p$ CAMHAMN

CAMHAMN é NP-Completo

CAMHAM $\leq_p$ CAMHAMN:

Gd -> Gn

Nós:

s -> ssai

t -> tentra

n ≠ s,t -> nentra, nmeio, nsai

Arestas:

(nentra, nmeio) e (nmeio, nsai) para todo n ≠ s,t

(xsai, yentra) ϵ Gn se (x,y) ϵ Gd

Figura

CAMHAMN é NP-Completo

Gd tem um cam.ham. entre s e t ⇔ Gn tem um cam.ham. entre ssai e tentra

CAMHAMN é NP-Completo

Gd tem um cam.ham. entre s e t => Gn tem um cam.ham. entre ssai e tentra

Caminho em Gd: s, u1, u2, ..., uk, t =>

Caminho em Gn: ssai, u1entra, u1meio, u1sai, u2entra, u2meio, u2sai, ..., ukentra, ukmeio, uksai, tentra

CAMHAMN é NP-Completo

Gd tem um cam.ham. entre s e t <= Gn tem um cam.ham. entre ssai e tentra

Precisamos provar que qualquer caminho hamiltoniano em Gn vai de tripla em tripla (exceto ssai e tentra)

De ssai tem que ir para algum uientra (só eles são conectados com ssai).

Deste uientra, tem que ir uma aresta para uimeio, e de uimeio para uisai, pois esta é a única forma de incluir uimeio.

Depois, deve haver uma aresta de uisai para ujentra, e assim por diante, até que entre em tentra.

SOMA-SUBC é NP-Completo

Já provamos que SOMA-SUBC pertence a NP.

Vamos provar que 3SAT $\leq_p$ SOMA-SUBC

Φ será satisfazível $\Leftrightarrow$ houver uma subcoleção T de S cuja soma é igual ao alvo t

SOMA-SUBC é NP-Completo - redução

- Φ: l variáveis e k cláusulas (Fig. 7.57)
- Variáveis xi de Φ: dois números yi e zi de S, somente um deles pertence à subcoleção T (definindo xi como verdadeiro ou falso)
- Os números de S e o valor de t serão linhas de uma tabela rotuladas por y1, z1, y2, z2, ..., yl, zl, g1, h1, ..., gk, hk, t (em notação decimal).

SOMA-SUBC é NP-Completo - redução

- Cada número de S é representado assim;
  - Parte de cima:
    - Parte da esquerda: um 1 seguido de l-i 0s
    - Parte da direita:
      - j-ésimo dígito de yi = 1 se xi aparece na cláusula cj
      - j-ésimo dígito de zi = 1 se !xi aparece na cláusula cj
      - 0 nos demais dígitos
  - Parte de baixo:
    - Pares de números iguais gj, hj, para cada cláusula cj : um 1 seguido de k-j 0s
- O número alvo t: l 1s seguidos de k 3s

SOMA-SUBC é NP-Completo – a redução funciona

Φ é satisfazível $\Leftrightarrow$ há uma subcoleção T de S cuja soma é igual ao alvo t

SOMA-SUBC é NP-Completo – a redução funciona

Φ é satisfazível => há uma subcoleção T de S cuja soma é igual ao alvo t:

- Selecionamos yi se xi = verdadeiro e zi caso contrário. A soma das primeiras l colunas = l 1s.
- A soma de cada um dos k últimos dígitos da parte de cima está entre 1 e 3, pois há 1 a 3 literais verdadeiros em cada cláusula. Completamos 3 selecionando, se necessário, gj e/ou hj

SOMA-SUBC é NP-Completo – a redução funciona

Φ é satisfazível <= há uma subcoleção T de S cuja soma é igual ao alvo t:

- Observações:
  - não existe vai-um na soma
  - Nas l primeiras colunas, deve-se selecionar yi ou zi mas não ambos
- Se yi foi selecionado, xi = verdadeiro ou falso caso contrário
- A soma é sempre 3 nas k colunas finais:
  - Coluna cj: no máximo 2 vem ge gj e hj , portanto pelo menos 1 deve vir de yi ou zi, satisfazendo cj:
    - Se vier de yi, xi está na cláusula e xi = V
    - Se vier de zi, !xi está na cláusula e xi = F

SOMA-SUBC é NP-Completo – a redução é polinomial

A tabela tem tamanho em torno de $(k+l)^2$, cada entrada é facilmente calculada

Tempo: $O(n^2)$