Aulas 24 e 25

NP-Completude

Prof. Marco Aurélio Stefanes
marco em dct.ufms.br

www.dct.ufms.br/ ~ marco

NP-Completude

- Problema Tratável - algoritmo $O(n^k)$ para algum k
  - o valor de k é pequeno
  - definição é robusta para modelos computacionáis razoáveis
  - polinômios têm propriedades de fechamento sob adição, multiplicação e composição
- Um problema abstrato de decisão consiste de um conj I de instâncias e de uma função $f : I \longrightarrow \{0, 1\}$
- o valor de $f(x)$ é a solução do problema para a instância $x \in I$

Problemas de Decisão

- Exemplo: Problema do Circuito Hamitoniano
  - Um circuito hamiltoniano em um grafo é um caminho fechado que passa por todos os vétices
  - I - Conj. de todos os grafos

$$f(G) = \begin{cases} 1 & \text{Se } G \text{ contém um CH, onde} G \in I \\ 0 & \text{Caso contrário} \end{cases}$$

- Problema de Otimização: queremos maximixar ou minimizar o valor de uma função. Podem ser formulados colocando-se um limitante no valor a ser otimizado. Dessa forma são visto como problemas de decisão. Se o problema de decisão for difícil então o problema de otimização também é.

Problemas NP-Completos

- Codificação de instâncias: Para um problema abstrato ser resolvido por um algoritmo é necessário que as instâncias do mesmo sejam devidamente condificadas
- Um problema concreto de decisão é um problema abstrato de decisão com a diferença de que as instâncias são codificadas por alfabeto finito
- Suposição: $\Sigma = \{0, 1\}$ sempre!
- Um algoritmo decide um problema concreto de decisão em tempo $O(T(n))$ se para cada $n \in N$ e cada $X \in I$ com $|x| = n$, o algoritmo encontrar a solução de x em tempo $O(T(n))$

Classe de Complexidade P

- Um problema concreto é solucionável em tempo polinomial se existe algum k fixo para o qual existe um algoritmo que resolve o problema em tempo $O(n^k)$
- a Classe de complexidade P é o conj. dos problemas concretos de decisão que são solucionáveis em tempo polinomial. (Problemas Tratáveis)
- Algoritmo: Programa escrito numa linguagem como C, etc. com restrição: custo das operações é proporcional ao tamanho dos operandos e memória infinita
  - Polinomialmente equivalente à máquina de Turing

Classe de Complexidade NP

- Verificação de soluções: considere o problema CH. Se a solução de uma instância G for SIM é possível encontrar no grafo um circuito hamiltoniano C. Neste caso, podemos verificar em tempo polinomial em G se C é ou não um circuito hamiltoniano.
- Se a resposta for NÃO, a verificação dessa resposta poder ser mais difícil. Não se conhece nenhum método para verificar em tempo polinomial a resposta NÃO do CH
- a classe de complexidade NP consiste dos problemas para os quais a resposta SIM pode ser verificada em tempo polinomial. i.e. Um problema de decisão está em NP se existe algoritmo A tq.
  1. Para qualquer instância x com solução SIM, existe um $y \in \Sigma^* | \ A(x, y) = SIM$
  2. para qualquer instância x com solução NÃO, para todo $y \in \Sigma^* | \ A(x, y) = NAO$
  3. A é polinomial em $|x|$

Classe de Complexidade co_NP

- $P \subseteq NP$ pois para qualquer problema $Q \in P$, o algoritmo A pode ser obtido do algoritmo para Q, ignorando-se o argumento y
- o complemento de um problema de decisão Q é o problema de decisão obtido invertendo-se o SIM e NÃO da solução de Q
- NCH é o complemento do problema CH: a solução é SIM, sempre que não tiver um circuito hamiltoniano e NÃO sempre que o grafo tiver um circuito hamiltoniano
- A classe co_NP consiste dos problemas de decisão que são complementares de problemas de decisão em NP

Problemas intratáveis

- "intrinsicamente" intratáveis
  Ex. Listar todos os circuitos de um grafo
- Problemas indecidíveis: Não existem algoritmos que resolva o problema
  - Problema da parada
  - Problema do ladrilhamento
- Problemas decidíveis fora de NP
- Algum problema em NP??
  - Problema do CH
  - Problema SAT

Redução de Problemas

- Um problema de decisão Q é reduzido em tempo polinomial a outro problema de decisão Q' se existe uma algoritmo A que, dada uma instância x de Q, $A(Q)$ é uma instância de Q' tal que
  - A solução para x é a mesma solução para $A(x)$
  - A roda em tempo polinomial em $|x|$
- Notação $Q \leq_p Q'$
- Solução para x é SIM se, e só se, solução para $A(x)$ é SIM
- Se soubermos resolver Q' em tempo polinomial, também sabemos resolver Q

Problemas NP_Completos

Lema Se $Q_1 \leq_p Q_2$ e $Q_2 \in P$, então $Q_1 \in P$

Prova: $Q_2 \in P \Rightarrow \exists$ algoritmo A_2 que resolve Q_2

$Q_1 \leq_p Q_2 \Rightarrow \exists$ algoritmo A que transforma, com as defidas propriedades qualquer instância x_1 de Q_1 numa instância $x_2 = A(x_1)$ de Q_2

Então $A_2(A(x_1))$ é uma solução para Q_1 e é polinomial em $|x_1|$, pois a composição de polinômios é um polinômio.

Um problema Q é NP_Completo se
1. $Q \in NP$
2. $Q' \leq_p Q$ para todo $Q' \in NP$

Um problema é NP_difícil se

$Q' \leq_p Q$ para todo $Q' \in NP$

Problema SAT

Teorema Se algum problema NP_Completo está em P, então $P = NP$. Se algum problema NP_Completo não pode ser resolvido por algum algoritmo polinomial, então nenhum problema NP_completo pode ser resolvido em tempo polinomial

Prova: imediata do lema anterior e das definições

Problema Satisfatibilidade (SAT) $X = \{x_1, x_2, \ldots, x_m\}$ conj de variáveis booleanas

Uma atribuição a X é uma função $t : X \rightarrow \{V, F\}$

Para $x \in X$, x e $\neg x$ são literais sobre x

Exemplo: $X = \{x_1, x_2\}$
Fórmula: $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2)$

Existe uma atribuição a X que satisfaz a fórmula?

Problemas SAT

- A literal x é verdadeira sob t se, e só se, $t(x) = V$
- A literal $\neg x$ é verdadeira sob t se, e só se, $t(x) = F$
- Uma cláusula sobre X é um conjunto de literais sobre X. Ela represetna a disjunção sobre as literias e é satisfeita por uma atribuição se, e só se, pelo menos uma das literias é verdadeira sob a atribuição
- Uma coleção de cláusulas C é satisfatível se, e só se, existe uma atribuição que satisfaz simultaneamente todas as cláusulas em C
- Problema SAT
  Dados X, C, decidir se C é satisfatível

Problema SAT

Teorema de Cook SAT é NP_Completo
$i)$ $SAT \in NP$, pois, dada uma atribuição, é fácil verificar se a atribuição satisfaz todas as cláusulas
$ii)$ $Q \leq_p SAT,\ \forall Q \in NP$

- definir rigorasamente uma linguaguem algoritmica de baixo nível, como as Máquinas de Turing
- $Q \in NP \Rightarrow \exists V|$ para qualquer instância x de Q, se a solução para x é:
  a) SIM, $\exists y|V(x,y)$ =SIM
  b) NÃO, $\forall y\ V(x,y)$ =NÃO

Problema SAT

- Dados x e y encare $y = \langle y_1, y_2, \cdots, y_m \rangle$ como variáveis booleanas $0 \equiv F \quad 1 \equiv V$
- codifique os possíveis passos do algoritmo V como função de y, gerando uma instância de SAT que é satisfatível se, e só se, $\exists y|V(x,y)$ =SIM
- Esta codificação é facilitada, pois V e escrito em uma linguagem bem elementar
- é necessário garantir que a instância de SAT é gerado em tempo polinomial

Este foi o primeiro problema NP_Completo

Problemas NP_Completos

Lema Se L é uma linguagem tal que $L' \leq_p L$ para alguma $L' \in NPC$ então L é NP_difícil. Além disso, se $L \in NP$ então $L \in NPC$

Prova Como L' é NP_Completa, para toda $L'' \in NP \Rightarrow L'' \leq_p L$. Por hipótese, $L' \leq_p L$ e pela transitividade (provar!), temos $L'' \leq_p L$, i.e L é NP_difícil. Caso $L \in NP$, temos que $L \in NPC$

Para Provar que L é NP_Completa

1. Prove que $L \in NP$
2. Escolha uma linguagem NP_Completa L'
3. Escreva jma algoritmo que mapeie toda instância x de L' para uma instância $f(x)$ de L
4. Prove que a função f satisfaz a $x \in L'$ se, e só se, $f(x) \in L$ para $\forall x$
5. Prove que o algoritmo calcula f em tempo polinomial

Problemas NP_Completos

Uma fórmula booleana está na forma normal conjuntiva-FNC - se é expressa em um grupo de conjunções de cláusulas ($\wedge$). A mesma está na 3-FNC se cada cláusula tem exatamente 3 literais distintas.
Exemplo $\phi = (x_1 \vee \neg x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee x_3 \vee \neg x_4)$
O problema 3-SAT é dados X e ϕ na 3-FNC, ϕ é satisfatível?
Teorema 3-SAT é NP_Completo
Prova
$i)$ Mostrar que $3 - SAT \in NP$. Mesmo argumento que $SAT \in NP$
$ii)$ Mostrar que $SAT \leq_p 3 - SAT$. Dada uma instância ϕ do SAT vamos transformá-la em uma instância ϕ' do 3-SAT tq ϕ é satisfatível se, e só se, ϕ' é satisfatível.

3-SAT é NP_Completo

Seja ϕ uma dada fórmula booleana. Fazemos a seguinte substituição local em cada cláusula C_i de ϕ

- Se $C_i = x$, trocamos C_i por
  $S_i = (x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg z)$,
  onde x, y são novas variáveis
- Se $C_i = (x_1 \vee x_2)$, trocamos C_i por
  $S_i = (x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \neg y)$
- Se $C_i = (x_1 \vee x_2 \vee \ldots x_k),\ k > 3$ trocamos C_i por
  $S_i = (x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \wedge (\neg y_2 \vee x_4 \vee y_3)$
  $\ldots (\neg y_{k-3} \vee x_{k-1} \vee x_k)$,
  onde y_i são todas novas variáveis

3-SAT é NP_Completo

- A atribuição às novas variáveis são irrelevantes.
- Note que cada C_i é 1 se e só se, S_i é 1. Assim a fórmula ϕ é satisfatível se, e só se, S é satisfatível.
- Note que cada cláusula aumenta em tamanho em um fator constante, portanto podemos reduzir o SAT ao 3-SAT em tempo polinomial
- Isto, junto com o fato $i)$ mostra que 3-SAT é NP_Competo

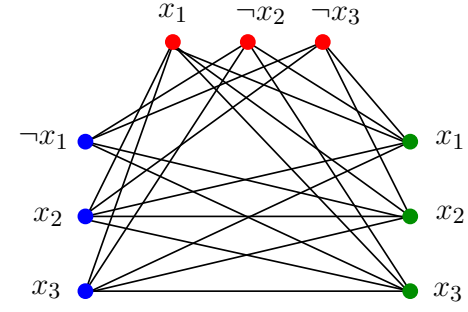
Problemas NP_Completos

Clique : em $G = (V, E)$ é um subconjunto $V' \subset V$ no qual cada par de nós está conectado por uma aresta de E. I.e V' induz um subgrafo completo de G.
Problema do Clique: Dados G e $k \in N$, há um clique de tamanho k em G?

Teorema Clique é NP_Completo
Prova
$i)$ Clique $\in NP$. O certificado é um conj $V' \subset V$. A verificação se V' é um clique é feita testando se para todo par $u, v \in V'$ há uma aresta $(u, v) \in E$. Isto pode ser feito em tempo polinomial.

Clique é NP_Completo

$ii)$ Mostrar que $3 - SAT \leq_p$ Clique.
Seja $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ uma instância do 3-SAT. Cada C_r possui 3 literais l_1^r, l_2^r, l_3^r. Construimos G_ϕ tq ϕ seja satisfatível se, e só se, G_ϕ tem um clique de tamanho k, como no exemplo
$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$



Clique é NP_Completo

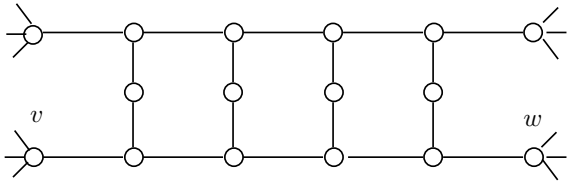
- Suponha que ϕ é satisfatível. Cada C_r contém l_i^r de atribuição 1 e corresponde a um vértice. Escolhendo uma literal verdadeira por cláusula temos k nós.
- Estes k nós pela construção de G_ϕ formam um clique, pois ambas as pontas não podem ser complementos.
- Suponha que G_ϕ tem um clique de k nós. Como não há arestas entre nós de uma tripla, temos exatamente um nó por tripla.
- Podemos atribuir 1 para cada literal correspondente aos nós do clique, pois não há arestas entre complementos. Portanto ϕ é satisfatível.
- Esta transformação pode ser feita em tempo polinomial.

Problemas NP_Completos

Teorema Caminho Hamiltoniano (CH) é NP_Completo

Prova

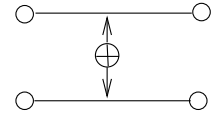
- $i)$ CH $\in NP$. O certificado é uma seqüência S de $|V|$ vértices distintos. A verificação se S é um ciclo hamiltoniano é feita testando se para todo par consecutivo $u, v \in S$ há uma aresta $(u, v) \in E$. Isto pode ser feito em tempo polinomial.
- $ii)$ 3-SAT $\leq_p$ CH. Dada ϕ vamos construir G_ϕ tq G_ϕ é hamiltoniano, se e só se, ϕ é satisfatível. Bloco a ser usado no grafo



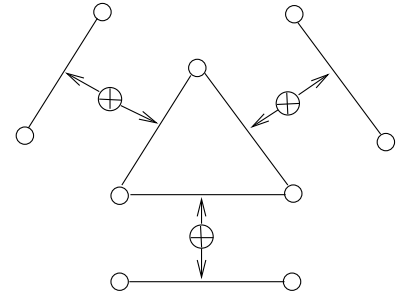
Suponha um CH sem extremo no bloco. Se o caminho entra por v ele deve sair por w.

CH é NP_Completo

Representamos por



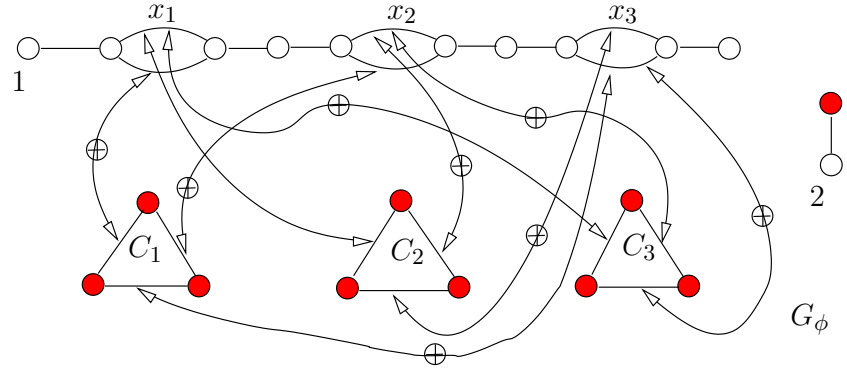
Cada cláusula de ϕ será representada por



Num CH é impossível visitar os 3 blocos entrando por nós do triâgulo

CH é NP_Completo

Para $\phi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$
Temos



Nós vermelhos formam um clique. O CH, se existe, tem extremos 1 e 2

CH é NP_Completo

- Suponha que um CH comece em 1.
  - O trajeto da esquerda para a direita define uma atribuição às variáveis de ϕ.
  - Após este trajeto, o CH deve cobrir todos os triângulos: é possível o CH cobrir no máximo dois blocos em cada triângulo.
  - Portanto, o trajeto precisa percorrer pelo menos um dos blocos(um literal verdadeira) de cada triângulo (cláusula).
- De forma similar, uma atribuição que satisfaz ϕ induz naturalmente um CH no grafo
- É fácil ver que o grafo pode ser construido em tempo polinomial