NP Completude

Prof^{a} Jerusa Marchi

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

e-mail: jerusa@inf.ufsc.br

Classes de Complexidade

- Se $\mathcal{P}$ difere de $\mathcal{NP}$, então a distinção entre $\mathcal{P}$ e $\mathcal{NP} - \mathcal{P}$ é muito importante
  1. Todos os problemas em $\mathcal{P}$ podem ser solucionados em tempo polinomial
  2. Todos os problemas em $\mathcal{NP} - \mathcal{P}$ são intratáveis
- Dado um problema Π em $\mathcal{NP}$, se $\mathcal{P} \neq \mathcal{NP}$, qual das duas possibilidades pode ser atribuída a Π?

Classes de Complexidade

- Até que se prove que $\mathcal{P} \neq \mathcal{NP}$, não há como mostrar que um problema em particular pertence a $\mathcal{NP} - \mathcal{P}$
- Isso se reflete nas demostrações
  - os resultados são provados de um modo fraco

    "Se $\mathcal{P} \neq \mathcal{NP}$ então o problema $\Pi \in \mathcal{NP} - \mathcal{P}$"

Classes de Complexidade

- Para elaborar tais provas, muitos dos conceitos e técnicas aplicados ao estudo da indecidibilidade são transportados para o estudo da complexidade, porém limitados no tempo
  - Reduções polinomiais
    - Análogo ao conceito de redução utilizado na demonstração da indecidibilidade de problemas

Reduções Polinomiais

- Uma redução polinomial de uma linguagem $L_1 \in \Sigma_1^*$ para uma linguagem $L_2 \in \Sigma_2^*$ é uma função $f : \Sigma_1^* \to \Sigma_2^*$ que satisfaz as seguintes condições:
  1. Há uma máquina de Turing determinística polinomialmente limitada que computa f
  2. Para todos $x \in \Sigma_1^*$, $x \in L_1$ se e somente se $f(x) \in L_2$

Reduções Polinomiais

- **Teorema:** Se uma linguagem L_1 se reduz polinomialmente a uma linguagem L_2 ($L_1 \propto L_2$), então $L_2 \in \mathcal{P}$ implica $L_1 \in \mathcal{P}$ (de modo equivalente $L_1 \notin \mathcal{P}$ implica $L_2 \notin \mathcal{P}$
- **Prova:** Sejam Σ_1 e Σ_2 alfabetos de L_1 e L_2 respectivamente. Seja $f : \Sigma_1^* \to \Sigma_2^*$ uma redução polinomial de L_1 para L_2. Seja M_f uma máquina de Turing determinística polinomialmente limitada que computa f e M_2 uma máquina de Turing polinomialmente limitada que computa L_2. Uma máquina de Turing determinística polinomialmente limitada que reconheça L_1 pode ser construída pela composição de M_f com M_2.

Reduções Polinomiais

- **Prova (cont.):** Uma entrada $x \in \Sigma_1^*$, é iniciamente transformada por M_f, gerando $f(x) \in \Sigma_2^*$. Em seguida, $f(x)$ é fornecida como entrada para M_2, determinando se $f(x) \in L_2$. Desde que $x \in L_1$ sse $f(x) \in L_2$, então a composição $M_f M_2$ é uma Máquina de Turing determinística que reconhece L_1. O fato de $M_f M_2$ operar em tempo polinomial decorre diretamente do fato de M_f e M_2 serem limitadas polinomialmente. Se p_f e p_2 são funções polinomiais que limitam M_f e M_2 respectivamente, então $|f(x)| \leq p_f(|x|)$ e o tempo de execução da máquina $M_f M_2$ é $O(p_f(|x|) + p_2(p_f(|x|)))$, que é polinomial em $|x|$.

Reduções Polinomiais

- Das linguagens para os problemas de decisão:
  - A redução de um problema de decisão Π_1 para um problema de decisão Π_2 significa que:
    - Se Π_2 puder ser resolvido em tempo polinomial, Π_1 também o será
    - Se Π_1 é intratável, Π_2 também é
  - Ou seja, $\Pi_1 \propto \Pi_2$ significa “O problema Π_2 é pelo menos tão difícil quando Π_1”

Reduções Polinomiais

- Exemplo: Redução do problema do Circuito Hamiltoniano para o problema da Satisfazibilidade Booleana
  - Seja dada uma instância do Circuito de Hamilton, ou seja, um grafo $G \subseteq V \times V$, onde $V = \{1, 2, ..., n\}$. Seja f um algoritmo que produza uma fórmula booleana em CNF $f(G)$, tal que G apresente um circuito Hamiltoniano se e somente se $f(G)$ é satisfazível.
  - A fórmula $f(G)$ envolverá n^2 variáveis booleanas, denotadas x_{ij}, com $1 \leq i, j \leq n$
  - x_{ij} tem o seguinte significado: "O vértice i de G é o j-ésimo vértice do circuito de Hamilton de G.

Reduções Polinomiais

- Exemplo: Redução do CH para SAT
  - As cláusulas em $f(G)$ expressarão os vínculos que um circuito de Hamilton deve satisfazer
    - No mínimo um vértice deve aparecer na i-ésima posição do circuito: constrói-se para $j = 1, ..., n$ cláusulas do tipo

$$(x_{1j} \vee x_{2j} \vee ...x_{nj})$$

    - Um único vértice de G pode ser o i-ésimo elemento do circuito: constrói-se para $i, j, k = 1, ..., n$ e $j \neq k$ (se x_{ij} então $\neg x_{ik}$)

$$(\neg x_{ij} \vee \neg x_{ik})$$

Reduções Polinomiais

- Exemplo: Redução do CH para SAT
  - O vértice i deve aparecer exatamente uma vez no circuito: constrói-se para $i = 1, ..., n$

$$(x_{i1} \vee x_{i2} \vee ...x_{in})$$

e para $i, j, k = 1, ..., n$ e $i \neq k$

$$(\neg x_{ij} \vee \neg x_{kj})$$

Reduções Polinomiais

- Exemplo: Redução do CH para SAT
  - Para garantir que a permutação seja de fato um circuito em G: constrói-se para $j = 1, ..., n$ e para cada par (i, k) de vértices tal que (i, k) não seja uma aresta de G cláusulas do tipo

    $$(\neg x_{ij} \vee \neg x_{k,j+1})$$

    se não há uma aresta (i, k) em G, i e k não poderão aparecer em posições sucessivas no suposto circuito

Reduções Polinomiais

- A redução polinomial é interessante, pois é transitiva
  - Se $L_1 \propto L_2$ e $L_2 \propto L_3$ então $L_1 \propto L_3$
- Diz-se que duas linguagens (L_1 e L_2) ou problemas de decisão (Π_1 e Π_2) são *polinomialmente equivalentes* sempre que $L_1 \propto L_2$ e $L_2 \propto L_1$ (ou $\Pi_1 \propto \Pi_2$ e $\Pi_2 \propto \Pi_1$)
  - A relação $\propto$ impõe uma ordem parcial sobre as classes de linguagens equivalentes resultantes (ou classes de problemas)
    - A classe $\mathcal{P}$ constitui a menor classe nesta ordem parcial e pode ser vista como sendo constituida pelas linguagens ou problemas computacionalmente “simples”
    - A classe de linguagens ou problemas $\mathcal{NP}$-Completos forma outra classe de problemas equivalentes que contém as linguagens ou problemas de decisão mais “difíceis” em $\mathcal{NP}$

$\mathcal{NP}$-Completude

- Uma linguagem L é definida como sendo $\mathcal{NP}$-Completa se:
  1. $L \in \mathcal{NP}$
  2. para todas as outras linguagens $L' \in \mathcal{NP}$, $L' \propto L$

$\mathcal{NP}$-Completude

- Em outras palavras, a classe de problemas $\mathcal{NP}$-Completos contém "os problemas mais difíceis em $\mathcal{NP}$"
  - Se qualquer problema $\mathcal{NP}$-Completo puder ser resolvido em tempo polinomial, então todos os problemas em $\mathcal{NP}$ também poderão
  - Se qualquer problema $\mathcal{NP}$ for intratável, então todos os problemas $\mathcal{NP}$-Completos também serão
- Se $\mathcal{P} \neq \mathcal{NP}$ então um problema $\mathcal{NP}$-Completo Π pertence a $\mathcal{NP} - \mathcal{P}$

$\mathcal{NP}$-Completude

- Por definição, provar que um problema é $\mathcal{NP}$-Completo implica reduzir **todos** os problemas em $\mathcal{NP}$ a ele
- *A priori* também não é aparente que haja algum problema $\mathcal{NP}$-Completo
- Esta tarefa pode ser simplificada considerando a transitividade da função $\propto$

$\mathcal{NP}$-Completude

- **Teorema:** Se L_1 e L_2 pertencem a $\mathcal{NP}$, L_1 é $\mathcal{NP}$-Completo, e $L_1 \propto L_2$ então L_2 é $\mathcal{NP}$-Completo
- **Prova:** Uma vez que $L_2 \in \mathcal{NP}$, tudo o que precisamos fazer é mostrar que, para cada $L' \in \mathcal{NP}$, $L' \propto L_2$. Considere qualquer $L' \in \mathcal{NP}$. Uma vez que L_1 é $\mathcal{NP}$-Completo, então é o caso que $L' \propto L_1$. A transitividade de $\propto$ e o fato que $L_1 \propto L_2$ então implica que $L' \propto L_2$

$\mathcal{NP}$-Completude

- Traduzindo o teorema anterior para problemas de decisão, temos um modo direto de provar novos problemas $\mathcal{NP}$-Completos, uma vez que tenhamos um problema reconhecidamente $\mathcal{NP}$-Completo disponível
  - Para provar que Π é $\mathcal{NP}$-Completo, basta mostrar que
    1. $\Pi \in \mathcal{NP}$, e
    2. algum problema reconhecidamente $\mathcal{NP}$-Completo Π' se reduz a Π

$\mathcal{NP}$-Completude

- O problema da Satisfazibilidade Booleana foi o primeiro problema de decisão a ser provado $\mathcal{NP}$-Completo
  - Teorema de Cook - O problema da Satisfazibilidade é $\mathcal{NP}$-Completo

Teorema de Cook

- **Prova:** Como visto, SAT está em NP, pois uma MT não determinística pode testar todas as possíveis combinações de valores verdade em tempo polinomial. Portanto o primeiro dos dois requisitos para $\mathcal{NP}$-Completude está assegurado.
  Para o segundo, seja L_{SAT} a linguagem que codifica o problema SAT. O que deve ser mostrado é que para toda linguagem $L \in \mathcal{NP}$, $L \propto L_{SAT}$. Cada linguagem $L \in \mathcal{NP}$ tem associada a si uma MT não determinística M que reconhece L em tempo polinomial. Seja L_M a linguagem reconhecida por M, assim, devemos mostrar como transformar L_M para L_{SAT}. Então, em essência, apresenta-se uma prova simultânea para toda a linguagem $L \in \mathcal{NP}$ que $L \propto L_{SAT}$.

Teorema de Cook

- Seja $M = (K, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ uma MT não determinística arbitrária que reconhece a linguagem L.
- Seja $p(n)$ o polinômio que limita o tempo de execução de M.
- A redução f_L deve ser derivada de M e p. Então f_L terá a propriedade que para cada $x \in \Sigma^*$, $x \in L$ se e somente se $f_L(x)$ for satisfazível
- Se a entrada $x \in \Sigma^*$ é aceita por M, então há uma computação de M com a entrada x tal que o número de passos é limitado por $p(n)$ onde $n = |x|$.

Teorema de Cook

- f_L inicialmente constrói um conjunto de variáveis booleanas U como segue:
  - Nomeia os elementos de K como $q_0, q_1 = q_a, q_2 = q_r, q_3, ..., q_k$ onde $k = |K| - 1$
  - Nomeia os elementos de Γ como $s_0 = \sqcup, s_1, s_2, ..., s_v$ onde $v = |\Gamma| - 1$

Teorema de Cook

- Há três tipos de variáveis, cada um com um significado distinto:

Variável	Limites	Significado
$Q_{i,k}$	$0 \leq i \leq p(n)$	no instante i
	$0 \leq k \leq r$	M está no estado q_k
$H_{i,j}$	$0 \leq i \leq p(n)$	no instante i, o cabeçote
	$0 \leq j \leq p(n)+1$	está na posição j da fita
$S_{i,j,k}$	$0 \leq i \leq p(n)$	no instante i, o conteúdo
	$0 \leq j \leq p(n)+1$	da fita na posição j
	$0 \leq k \leq v$	é o símbolo k

Teorema de Cook

- As cláusulas em f_L se dividem em 6 grupos que impõem tipos distintos de restrições que fazem com que uma atribuição de valores verdade válida corresponda a uma computação que aceita x

Teorema de Cook

Grupo de Cláusulas	Restrição imposta
G_1	a cada instante i, M está em exatamente um estado
G_2	a cada instante i, o cabeçote está varrendo exatamente 1 posição da fita
G_3	a cada instante i, cada posição da fita contém exatamente um símbolo de Γ
G_4	no instante 0, a computação está na configuração inicial
G_5	no instante $p(n)$, M está no estado q_a, aceitando x
G_6	para cada instante i, $0 \leq i < p(n)$, a configuração de M no instante $i+1$ segue pela aplicação de uma transição δ para a configuração no instante i

Teorema de Cook

- Grupo 1 (G_1): a cada instante i, M está em exatamente um estado

$$\{Q_{i,0} \vee Q_{i,1} \vee ... \vee Q_{i,r}\}, 0 \leq i \leq p(n)$$

$$\{\overline{Q_{i,j}} \vee \overline{Q_{i,j'}}\}, 0 \leq i \leq p(n), 0 \leq j < j' \leq r$$

- Grupo 2 (G_2): a cada instante i, o cabeçote está varrendo exatamente 1 posição da fita

$$\{H_{i,0} \vee H_{i,1} \vee ... \vee H_{i,p(n)+1}\}, 0 \leq i \leq p(n)$$

$$\{\overline{H_{i,j}} \vee \overline{H_{i,j'}}\}, 0 \leq i \leq p(n), 0 \leq j < j' \leq p(n) + 1$$

Teorema de Cook

- Grupo 3 (G_3): a cada instante i, cada posição da fita contém exatamente um símbolo de Γ

$$\{S_{i,j,0} \vee S_{i,j,1} \vee ... \vee S_{i,j,v}\}, 0 \leq i \leq p(n), 0 \leq j \leq p(n)+1$$

$$\{\overline{S_{i,j,k}} \vee \overline{S_{i,j,k'}}\}, 0 \leq i \leq p(n), 0 \leq j \leq p(n)+1, 0 \leq k < k' \leq v$$

- Grupo 4 (G_4): no instante 0, a computação está na configuração inicial

$$\{Q_{0,0}\}, \{H_{0,1}\}, \{S_{0,0,0}\}$$

$$\{S_{0,1,k_1}\}, \{S_{0,2,k_2}\}, ..., \{S_{0,n,k_n}\} \text{//entrada}$$

$$\{S_{0,n+1,0}\}, \{S_{0,n+2,0}\}, ..., \{S_{0,p(n)+1,0}\} \text{//restante branco}$$

- Grupo 5 (G_5): no instante $p(n)$, M está no estado q_a, aceitando x

$$Q_{p(n),1}$$

Teorema de Cook

- Grupo 6 (G_6): este grupo se subdivide em 2 subgrupos de cláusulas
  - o primeiro diz que se o cabeçote de leitura não está varrendo a fita na posição j no instante i, então o símbolo na posição j não muda entre os instantes i e $i+1$.

$$\{\overline{S_{i,j,l}}, H_{i,j}, S_{i+1,j,l}\}, 0 \leq i < p(n), 0 \leq j \leq p(n)+1, 0 \leq l \leq v$$

Teorema de Cook

- Grupo 6 (G_6)
  - o segundo subgrupo assegura que as mudanças feitas de uma configuração para outra estão de acordo com a tabela de transição δ de M
    - Para cada quádrupla (i, j, k, l), $0 \leq i \leq p(n), 0 \leq j \leq p(n) + 1, 0 \leq k \leq r$ e $0 \leq l \leq v$ o subgrupo contém as seguintes três fórmulas

$$\{\overline{H_{i,j}}, \overline{Q_{i,k}}, \overline{S_{i,j,l}}, H_{i+1,j+\Delta}\}$$

$$\{\overline{H_{i,j}}, \overline{Q_{i,k}}, \overline{S_{i,j,l}}, Q_{i+1,k'}\}$$

$$\{\overline{H_{i,j}}, \overline{Q_{i,k}}, \overline{S_{i,j,l}}, S_{i+1,j,l'}\}$$

onde se $q_k \in Q - \{q_a, q_r\}$, então os valores de Δ, k' e l' são tais que $\delta(q_k, s_l) = (q_{k'}, s_{l'}, \Delta)$, com $\Delta = \{+1, -1\}$. Se $q_k \in \{q_a, q_r\}$, então $\Delta = 0$, $k' = k$ e $l' = l$

Teorema de Cook

- Se $x \in L$, então há uma computação em M que aceita x em $p(n)$ passos ou menos, e esta computação, dada a interpretação das variáveis, impõe uma atribuição que satisfaz todas as cláusulas em

$$C = G_1 \cup G_2 \cup G_3 \cup G_4 \cup G_5 \cup G_6$$

- Portanto, SAT é NP-Completo.$\square$

Problemas $\mathcal{NP}$-Completos

- Tendo provado um primeiro problema $\mathcal{NP}$-Completo, é possível reduzi-lo a outro e este a outros, concluindo serem todos $\mathcal{NP}$-Completos
- Identificar problemas $\mathcal{NP}$-Completos e/ou demonstrar que algum problema é $\mathcal{NP}$-Completo é muito importante, pois:
  - Problemas $\mathcal{NP}$-Completos aparecem disfarçados em diversos campos e aplicações
  - Identificá-los economiza tempo e recursos
  - Incentiva o estudo de técnicas "heurísticas" ou "aproximadas" para elaboração de soluções viáveis

Problemas $\mathcal{NP}$-Completos

- Técnicas para tratar problemas $\mathcal{NP}$-Completos
  - Técnicas de Inteligência Artificial
    - algoritmos genéticos
    - métodos de busca heurísticos (A^*, busca gulosa, subida de encosta, têmpera simulada)
  - Técnicas algorítmicas
    - algoritmos aproximados (ou de aproximação)
    - algoritmos de divisão e conquista
    - programação dinâmica

Outros Problemas $\mathcal{NP}$-Completos

- 3-Satisfazibilidade Booleana (3SAT)
  - caso particular do problema da satisfazibilidade onde as cláusulas envolvem 3 literais ou menos.
- MAX SAT
  - Dado um conjunto F de cláusulas e um inteiro K, existe alguma atribuição que satisfaça pelo menos K cláusulas?

Outros Problemas $\mathcal{NP}$-Completos

- Problema do Circuito Hamiltoniano/Caminho Hamiltoniano
- Problema do Caixeiro Viajante
- Caminho mais longo/Caminho mais curto
- Problema da Mochila
- Escalonamento de tarefas/Alocação ótima de registradores

Outros Problemas $\mathcal{NP}$-Completos

- Cobertura Exata
  - Dados um conjunto finito $U = \{u_1, u_2, ..., u_n\}$ (o conjunto universo) e a família de m subconjuntos de U, $\mathcal{F} = \{S_1, S_2, ..., S_m\}$, pergunta-se de existiria uma *cobertura exata* $\mathcal{C} \subseteq \mathcal{F}$ tal que os conjuntos em $\mathcal{C}$ são disjuntos e a sua união é U
    - Exemplo: Seja $U = \{u_1, u_2, u_3, u_4, u_5, u_6\}$ o universo, e seja uma família de subconjunto $\mathcal{F} = \{\{u_1, u_3\}, \{u_2, u_3, u_6\}, \{u_1, u_5\}, \{u_2, u_3, u_4\}, \{u_5, u_6\}, \{u2, u_4\}\}$

$$\mathcal{C} = \{\{u_1, u_3\}, \{u_5, u_6\}, \{u2, u_4\}\}$$

Outros Problemas $\mathcal{NP}$-Completos

- Conjuntos Independentes
  - Dado um grafo não orientado $G = (V, E)$, um conjunto independente V' de G é um subconjunto de vértices tal que não existem dois vértices adjacentes contidos em V'

$$V' = \{u | u \in V \text{ e para todo } u' \in V, (u, u') \notin E\}$$

Outros Problemas $\mathcal{NP}$-Completos

- Clique
  - Dado um grafo $G = (V, E)$, um clique C é um subconjunto de vértices de G tal que para cada par de vértices em C existe uma aresta que os conecta.

$$C = \{u | u \in V \text{ e } \forall u' \in C, (u, u') \in E\}$$

$\mathcal{NP}$-Difícil

- Lembrando: Para provar que um problema Π é $\mathcal{NP}$-Completo, basta mostrar que
  1. $\Pi \in \mathcal{NP}$, e
  2. algum problema reconhecidamente $\mathcal{NP}$-Completo Π' se reduz a Π

$\mathcal{NP}$-Difícil

- Alguns problemas são tão difíceis que apesar de ser possível demonstrar a condição (2) da definição de $\mathcal{NP}$-Completude, não é possível provar a condição (1)
- Estes problemas formam a classe $\mathcal{NP}$-Difícil ou $\mathcal{NP}$-Hard
  - O termo “intratável” é comumente usado indicar problemas $\mathcal{NP}$-Hard, ainda que em princípio possa haver problemas que requerem tempo exponencial mesmo não estando em $\mathcal{NP}$-Hard no sentido formal