Teoria da Complexidade

Cid C. de Souza / IC–UNICAMP

Universidade Estadual de Campinas
Instituto de Computação

1º semestre de 2012

Revisado por Zanoni Dias

Autor

Prof. Cid Carvalho de Souza
Universidade Estadual de Campinas (UNICAMP)
Instituto de Computação
Av. Albert Einstein nº 1251
Cidade Universitária Zeferino Vaz
13083-852, Campinas, SP, Brasil
Email: cid@ic.unicamp.br

Direitos autorais

- Este material só pode ser reproduzido com a autorização do autor.
- Os alunos dos cursos do Instituto de Computação da UNICAMP bem como os seus docentes estão autorizados (e são bem vindos) a fazer uma cópia deste material para estudo individual ou para preparação de aulas a serem ministradas nos cursos do IC/UNICAMP.
- Se você tem interesse em reproduzir este material e não se encontra no caso acima, por favor entre em contato comigo.
- Críticas e sugestões são muito bem vindas!

Campinas, agosto de 2010.

Cid

Reduções entre problemas

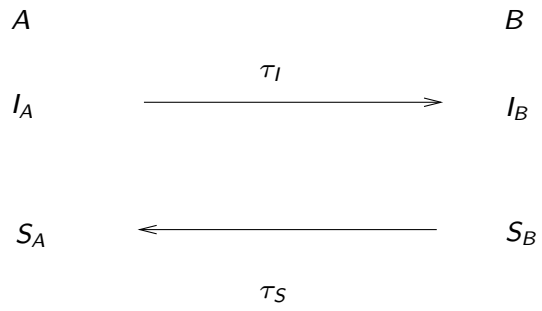
▷ Idéia básica da **redução de Turing**:

Problema A:	Problema B:
• Instância de entrada: I_A;	• Instância de entrada: I_B;
• Solução: S_A.	• Solução: S_B.

▷ **Definição**: uma **redução** do problema A **para** o problema B é um par de transformações τ_I e τ_S tal que, dada uma instância qualquer I_A de A:

- τ_I transforma I_A em uma instância I_B de B **e**
- τ_S transforma a solução S_B de I_B em uma solução S_A de I_A.

▷ **Esquema**:



▷ Quando usar **reduções**?

- **Situação 1**: quero encontrar um algoritmo para A e conheço um algoritmo para B. Ou seja, vou determinar uma *cota superior* para o problema A.
- **Situação 2**: quero encontrar uma *cota inferior* para o problema B e conheço uma *cota inferior* para o problema A.

▷ Exemplo:

- Desejo resolver um sistema linear da forma $Ax = b$.
- Disponho de um programa que resolve sistemas lineares quando a matriz de entrada A é simétrica (i.e., $a_{ij} = a_{ji}$).
- O meu sistema linear não satisfaz esta propriedade.

◦ O que fazer?

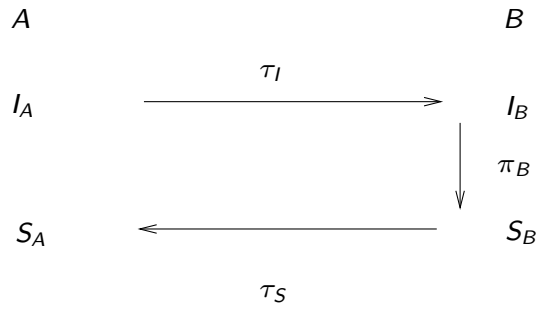
- Transformar a instância do meu problema numa instância que é resolvida pelo algoritmo implementado pelo programa.
- Notar que todo x que é solução de $A^T Ax = A^T b$ também é solução de $Ax = b$ e que $A^T A$ é *simétrica*.

▷ Resolver sistemas lineares da forma $Ax = b$ quando A é **simétrica** é *pelo menos tão difícil quanto* resolver um sistema linear onde A é uma matriz qualquer?

Formalizando ...

$\mathcal{I}_\bullet$: conjunto de todas instâncias do problema $\bullet$;
$\mathcal{S}_\bullet$: conjunto de todas as soluções das instâncias em $\mathcal{I}_\bullet$;

▷ **Definição**: Um problema A é *redutível ao problema* B em tempo $f(n)$ se existe a redução esquematizada abaixo



onde: $n = |I_A|$ e τ_I e τ_S são $O(f(n))$.

▷ **Notação**: $A \propto_{f(n)} B$.

▷ **Observações**:

1. Conhecendo um algoritmo π_B para B, temos imediatamente um algoritmo π_A que resolve *instâncias genéricas* de A:

$$\pi_A \doteq \tau_S \circ \pi_B \circ \tau_I.$$

   A **complexidade de** π_A será dada pela soma das complexidades de τ_I, π_B e τ_S. Ou seja, temos uma *cota superior* para A.

2. Se π_B tem complexidade $g(n)$ e $g(n) \in \Omega(f(n))$ então temos que $g(n)$ também é cota superior para A.
   - *Se* $g(n) \notin \Omega(f(n))$, *a cota superior de* $g(n)$ *ainda vale?*

3. Se $\Omega(h(n))$ é uma cota inferior para o problema A e $f(n) \in o(h(n))$, então $\Omega(h(n))$ também é cota inferior para o problema B.
   - *Por quê exigir que* $f(n) \in o(h(n))$? *O que aconteceria se não fosse?*
   - *Lembrete:* $o(h(n))$ *e* $\Omega(h(n))$ *são* **mutuamente excludentes !**

▷ **Problema do casamento cíclico de cadeias de caracteres (CSM)**

Entrada: Alfabeto Σ e duas cadeias de caracteres de tamanho n:

$$A = a_0 a_1 \ldots a_{n-1} \text{ e } B = b_0 b_1 \ldots b_{n-1}.$$

Pergunta: B é um *deslocamento cíclico* de A?

Ou seja, existe $k \in \{0, \ldots, n-1\}$ tal que $a_{(k+i) \mod n} = b_i$ para todo $i \in \{0, \ldots, n-1\}$?

- Exemplo: para $A = acgtact$ e $B = gtactac$ temos $n = 7$ e $k = 2$.

- Como se resolve este problema?

◇ **Problema do Casamento de Cadeias (SM)**:

Entrada: Alfabeto Σ e duas cadeias de caracteres:
$A = a_0 a_1 \ldots a_{n-1}$ e $B = b_0 b_1 \ldots b_{m-1}$, sendo $m \leq n$.
Pergunta: Encontrar a primeira ocorrência de B em A ou concluir que B não é subcadeia de A.

Ou seja, determinar o menor $k \in \{0, \ldots, n-1\}$ tal que $a_{k+i} = b_i$ para todo $i \in \{0, \ldots, m-1\}$ ou retornar $k = -1$.

- Exemplo: para $A = acgttaccgtacccg$ e $B = tac$ ($n = 15$ e $m = 3$) tem-se $k = 4$.

- **Observação**: O problema SM pode ser resolvido em tempo $O(m + n)$ através do algoritmo de Knuth, Morris e Pratt (1977).

◇ **Redução:** CSM $\propto_n$ SM

- Instância de CSM: $I_{CSM} = (A, B, n)$;
- τ_I constrói a instância de SM:

$$I_{SM} = (A', 2n, B, n), \text{ onde } A' = A \parallel A.$$

  Portanto, τ_I é $O(n)$.
- Se k é a solução de SM para I_{SM}, então k também é solução de I_{CSM}. Logo, τ_S é $O(1)$ e a redução é $O(n)$.

◇ Exemplo:

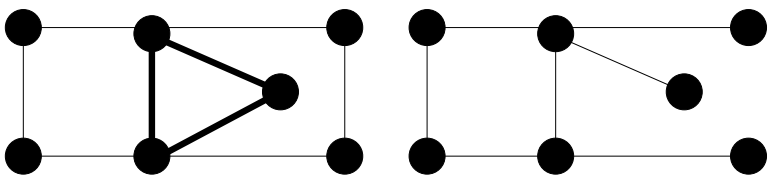
- $I_{CSM} = (acgtact, gtactac, 7)$;
- $I_{SM} = (acgtactacgtact, 14, gtactac, 7)$;
- $S_{SM} = S_{CSM} = \{k = 2\}$.

Exemplos de Reduções (cont.)

▷ **Problema da existência de um triângulo em um grafo conexo não orientado (PET):**

Entrada: Grafo conexo não orientado $G = (V, E)$, sem auto-laços, onde $|V| = n$ e $|E| = m$.
Pergunta: G possui um ciclo de comprimento 3, ou seja, um triângulo?

◦ Exemplos:



(a) Com △. (b) Sem △.

Observações:

- Algoritmo trivial: verificar todas as triplas de vértices (complexidade=$O(n^3)$).
- Existe algoritmo $O(mn)$ que é muito bom para *grafos esparsos*.
- Supor que o grafo é dado pela sua *matriz de adjacências* $A(G)$.
- Se $A^2(G) = A(G) \times A(G)$, então $a_{ij}^2 = \sum_{k=1}^{n} a_{ik}.a_{kj}$. Logo:

$$a_{ij}^2 > 0 \Leftrightarrow \exists\, k \in \{1, \ldots, n\} \text{ tal que } a_{ik} = a_{kj} = 1.$$

- Portanto, o triângulo (i, j, k) existirá se e somente se $a_{ij}^2 > 0$ e $a_{ij} = 1$.
- *Observação*: $a_{ii} = 0$ pois não há auto-laços.

◇ **Problema da Multiplicação de Matrizes Quadradas (MMQ):**

Entrada: Duas matrizes quadradas de números inteiros A e B de ordem n.
Pergunta: Qual é a matriz P resultante do produto $A \times B$?
◇ **Observação**: MMQ pode ser resolvido em tempo $O(n^{\log 7 = 2.807})$ através do algoritmo de Strassen (1969) ou em $O(n^{2.376})$ através do algoritmo de Coppersmith e Winograd (1990).

◇ **Redução:** PET $\propto_{n^2}$ MMQ

- $I_{PET} = A(G)$;
- τ_I constrói a instância de MMQ:

$$I_{MMQ} = (A, A, n), \text{ onde } A = A(G).$$

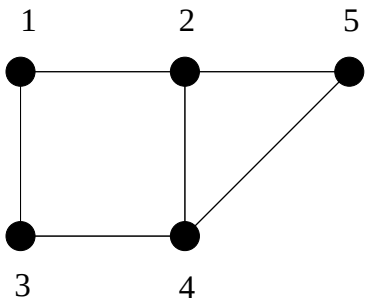
Portanto, τ_I é $O(n^2)$.

- Se $S_{MMQ} = P$ é a solução de MMQ para I_{MMQ}, então a solução de I_{PET} pode ser obtida através do algoritmo τ_S a seguir:

Para $i = 1$ **até** n **faça**
  Para $j = 1$ **até** n **faça**
    Se $(p_{ij} > 0$ **e** $a_{ij} = 1)$, **retorne** `Verdadeiro`.
Retorne `Falso`.

▷ A complexidade de τ_S é $O(n^2)$.

◦ Exemplo: PET $\propto$ MMQ.



$A(G)$

	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	1	1
3	1	0	0	1	0
4	0	1	1	0	1
5	0	1	0	1	0

$P = A(G) \times A(G)$

	1	2	3	4	5
1	2	0	0	2	1
2	0	3	2	1	1
3	0	2	2	0	1
4	2	1	0	3	1
5	1	1	1	1	2

▷ **Multiplicação de Matrizes Quadradas Simétricas (MMQS):**

Entrada: 2 matrizes simétricas A e B de números inteiros de ordem n.
Pergunta: Qual é a matriz P resultante do produto $A \times B$?

◇ Problema MMQ: obter a matriz produto de duas matrizes quadradas arbitrárias (não necessariamente simétricas).
◇ MMQS é mais fácil do que MMQ?

◇ **Observações**:

- MMQS é um caso particular de MMQ: a redução $MMQS \propto MMQ$ é imediata e tem complexidade $O(n^2)$. Portanto *MMQ é pelo menos tão difícil quanto MMQS.*
- Será que *MMQS é pelo menos tão difícil quanto MMQ?* *(menos intuitivo)*

◇ **Redução:** MMQ $\propto_{n^2}$ MMQS

- $I_{MMQ} = (A, B, n)$;
- τ_I constrói a instância de MMQS: $I_{MMQS} = (A', B', 2n)$, onde

$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \quad \text{e} \quad B' = \begin{bmatrix} 0 & B^T \\ B & 0 \end{bmatrix}$$

  Portanto, τ_I é $O(n^2)$.
- A solução do MMQS é dada por:

$$P' = A'B' = \begin{bmatrix} AB & 0 \\ 0 & A^T B^T \end{bmatrix}$$

  Se P é a solução de MMQ, então τ_S pode ser implementada através do seguinte algoritmo:

Para $i = 1$ **até** n **faça**
  Para $j = 1$ **até** n **faça**
    $p_{ij} = p'_{ij}$.

▷ A complexidade da redução é $O(n^2)$.
▷ Pela redução acima, **se** todo algoritmo de MMQ está em $\Omega(h(n))$, **então** todo algoritmo para MMQS está em $\Omega(h(n))$ também.
- Note que $h(n)$ está em $\Omega(n^2)$. *(Por quê?)*

▷ NOTA: se $T(n)$ é a complexidade de um algoritmo para MMQS e $T(2n) \in O(T(n))$ †, então pela redução acima, tem-se um algoritmo de complexidade $O(T(n) + n^2)$ para resolver *MMQ*.

---

†: propriedade atendida por funções "suaves" (p.ex., qualquer polinômio).

Erros comuns ao se usar reduções

1. Usar redução na ordem inversa: por exemplo ao fazer a redução $A \propto B$ e concluir que A é pelo menos tão difícil quanto B.
2. Dada a redução $A \propto B$ achar que toda instância de B tem que ser mapeada numa instância de A (o mapeamento só vai numa direção).
3. Usar o algoritmo produzido por uma redução sem se preocupar com a existência de um outro algoritmo mais eficiente.

   Exemplo:
   - redução do **problema inteiro da mochila (IKP)** ao **problema binário da mochila (BKP)**.
   - A redução pode levar *a uma instância de entrada do BKP de tamanho muito grande!*

Reduções polinomiais

Definição: Se $A \propto_{f(n)} B$ e $f(n) \in O(n^k)$ para algum valor k real, então a redução de A para B é **polinomial**.

Observações:

- No caso de obtenção de uma cota superior para A, a importância das reduções polinomiais é óbvia pois, havendo um algoritmo polinomial para B, a redução leva imediatamente a um algoritmo *eficiente* para A.
- Todas reduções vistas anteriormente são polinomiais.
- A existência de uma redução polinomial do problema A para o problema B é denotada por $A \propto_{\text{poli}} B$.

Programação Linear e reduções em Fluxos em Redes

- Perceba que uma *formulação como PL* de um problema Π qualquer, nada mais é do que uma *redução de* Π *à PL!*
- Como um PL pode ser resolvido em tempo polinomial através de um algoritmo de pontos interiores, se a formulação PL de Π tiver um número de restrições e variáveis polinomial no tamanho da instância de Π, *teremos um algoritmo polinomial para* Π*!* Daí a importância de sabermos fazer modelos de PL.
- A exemplo da PL, uma classe de problemas que também desempenha um papel de "coringa" nas reduções são os problemas de *fluxos em redes*.
- O mais geral dentre esses problemas é o chamado *Problema do Fluxo de Custo Mínimo* (FCM), inclusive porque vários outros problemas importantes de fluxos em redes são casos especiais do FCM.

O Problema do Fluxo de Custo Mínimo (FCM)

- **Dados de entrada**: a rede $G = (N, A)$ (grafo orientado). $|N| = n$ (número de vértices ou nós) e $|A| = m$ (número de arcos); custo por unidade de fluxo através do arco $(i, j) \in A$ dado por c_{ij} (supõe-se que o custo do fluxo varia linearmente com a quantidade de fluxo); a cada arco (i, j) está associado um fluxo mínimo l_{ij} e um fluxo máximo (ou capacidade) u_{ij} que pode atravessá-lo; a cada vértice i de N está associado um número $b(i)$ que representa fornecimento ($b(i) > 0$) ou demanda ($b(i) < 0$) de fluxo. Os vértices com $b(i) = 0$ são ditos ser vértices de passagem.
- **Problema**: encontrar o fluxo que deve passar em cada arco da rede de modo a atender a demanda de cada vértice, respeitando as capacidades dos arcos e que minimize o custo total.

FCM formulado como um PL

- Variáveis de decisão: x_{ij} é o fluxo no arco (i, j) de A.
- Hipótese: vamos supor que $\sum_{i \in N} b(i) = 0$.

$$\begin{array}{lll} \min & \sum_{(i,j)\in A} c_{ij} x_{ij} & \\ \text{s.a.} & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ij} = b(i) & \forall\ i \in N \qquad (\dagger) \\ & l_{ij} \leq x_{ij} \leq u_{ij} & \forall\ (i,j) \in A \end{array}$$

Sendo $\mathcal{N}$ a matriz de incidência vértice–arco de G, este programa linear pode ser escrito como:

$$\begin{array}{ll} \min & cx \\ \text{s.a.} & \mathcal{N}x = b \\ & l \leq x \leq u \end{array}$$

Nota: ($\dagger$) são chamadas de restrições de conservação de fluxo.

Redução de Problemas de Caminhos Mínimos ao FCM

- Na versão básica do Problema de Caminho Mínimo (CM), são dados dois vértices s e t de N e uma *distância* c_{ij} é associada a cada arco (i,j) de A. *Deseja-se encontrar um caminho de menor distância total de s para t.*
- Este problema pode ser modelado como um FCM bastando, para isso, fazer: (1) $b(s) = 1$, $b(t) = -1$ e $b(i) = 0$ para todo $i \in N \setminus \{s,t\}$, e (2) $l_{ij} = 0$ e $u_{ij} = 1$ para todo $(i,j) \in A$.
- No Problema de Todos os Caminhos Mínimos (TCM) desejamos computar o caminho mais curto de s para todos os demais vértices de N.
- A redução de TCM para FCM poder ser feita tomando-se: (1) $b(i) = -1$ para todo $i \in N \setminus \{s\}$, (2) $b(s) = (n-1)$ e (3) $l_{ij} = 0$ e $u_{ij} = (n-1)$.

Redução do Problema do Fluxo Máximo (FM) ao FCM

- No FM, é dado um vértice fonte/origem s, um vértice sorvedouro/destino t e capacidades nos arcos da rede. Deseja-se computar o fluxo máximo que pode ser enviado de s para t, respeitadas as capacidades dos arcos da rede.
- Para formular o FM como um FCM faz-se o seguinte: (1) adiciona-se um arco (t,s) à rede com capacidade $u_{ts} = \infty$ e custo $c_{ts} = -1$; (2) faz-se $b(i) = 0$ para todo vértice $i \in N$ e $c_{ij} = 0$ para todo arco (i,j) em A. Neste caso, o fluxo mínimo para todo arco (i,j) é nulo ($l_{ij} = 0$).
- Por construção, ao minimizar o custo do fluxo no novo arco (t,s), estamos maximizando o fluxo neste arco.
- Como todo $b(i)$ é nulo, a quantidade de fluxo em (t,s) deve ser a mesma que vai de s para t através dos arcos originalmente em A.

Problema de Alocação de Recursos (AR) reduzido ao FCM

- No Problema de Alocação de Recursos (AR) são dados dois conjuntos N_1 e N_2 de mesmo tamanho e uma coleção de pares $A \subseteq N_1 \times N_2$, representando possíveis alocações de recursos de N_1 em elementos de N_2. Além disso, dado um par (i, j) de A, c_{ij} denota o custo desta associação.
- Deseja-se encontrar em A um emparelhamento 1-para-1 entre os elementos de N_1 e N_2 cujo custo total (soma dos custos dos pares do emparelhamento) seja mínima.
- Para formular este problema como um FCM, basta construir a rede $G = (N_1 \cup N_2, A)$ e fazer: (1) $b(i) = 1$ para todo $i \in N_1$; (2) $b(i) = -1$ para todo $i \in N_2$, $l_{ij} = 0$ e $u_{ij} = 1$ para todo arco $(i, j) \in A$.
- Vimos anteriormente uma variante deste problema: o problema do transporte (caso das fazendas de cana e das usinas de álcool).

Observações sobre problemas de Fluxos em Rede

- Pela redução mostrada aqui, sabemos que o FCM pode ser resolvido usando qualquer algoritmo disponível para resolver PLs, em particular, o SIMPLEX.
- Porém, é possível especializar o algoritmo do SIMPLEX exclusivamente para tratar problemas de fluxos em redes. Esse algoritmo é chamado de *network simplex*.
- Nessa implementação o SIMPLEX é muito mais eficiente do que no caso geral, valendo-se da relação existente entre *soluções básicas* e *árvores* na rede.
- Existe uma implementação para o *network simplex* com complexidade $O(\min\{nm \log n(\log n + \log C), nm^2 \log^2 n\})$, onde $\log C$ é o espaço necessário (em número de bits) para armazenar o valor do maior custo unitário de transporte de fluxo dentre todos arcos da rede (Orlin, 1997 + Tarjan, 1997).

Observações sobre problemas de Fluxos em Rede

- O Problema de Fluxo de Custo Mínimo (FCM) possui uma característica muito importante: se todas as restrições envolverem apenas valores inteiros, então todos os vértices do poliedro que define a região viável do problema possuem coordenadas inteiras.
- Sendo assim, se as restrições forem inteiras, o SIMPLEX (e sua versão especialida *network simplex*) sempre encontrará soluções inteiras.

Observações sobre problemas de Fluxos em Rede

- O resultado anterior juntamente com as reduções que mostramos dos problemas de *caminhos mínimos*, *fluxo máximo* e *alocação de recursos* ao FCM, mostram que esses problemas admitem algoritmos polinomiais.
- Mas, nunca é demais lembrar que o uso de reduções *não* necessariamente conduz ao algoritmo mais eficiente para resolver um problema!
- Por exemplo:
  - O Problema do Fluxo Máximo (FM) pode ser resolvido em $O(n^2\sqrt{m})$ (Cheriyan e Maheshwari, 1989).
  - O Problema de Todos Caminhos Mínimos (TCM) pode ser resolvido em $O(n^3)$ (Floyd-Warshall, 1962).

Maiores informações sobre Fluxos em Redes:

Consulte o livro *"Network Flows: Theory, algorithms, and applications", R.K. Ahuja, T.L. Magnanti e J.B. Orlin, Prentice-Hall, 1993.*