Aula 07 – Problemas NP-Complete

Prof. Eduardo Zambon

Departamento de Informática (DI)
Centro Tecnológico (CT)
Universidade Federal do Espírito Santo (Ufes)

Algoritmos e Fundamentos da Teoria de Computação (ToCE)
Engenharia de Computação

Introdução

- Classe $\mathcal{P}$: problemas tratáveis.
- Classe $\mathcal{NP}$: problemas intratáveis.
- Classe $\mathcal{NPC}$: problemas "fundamentais" da classe $\mathcal{NP}$.
- **Estes *slides***: Quais são os principais problemas NP-complete?
- **Objetivos**: Apresentar reduções de SAT para outros problemas, caracterizando-os como NP-complete.

Referências

Chapter 16 – NP-Complete Problems
T. Sudkamp
Section 7.5 – Additional NP-Complete Problems
M. Sipser
Section 6.5 – NP-Complete Languages
A. Maheshwari

Introdução

- Prova de que SAT é NP-complete (Teorema de Cook): associação entre computações de TM com fórmulas CNF.
- Prova é extremamente elaborada e trabalhosa.
- Construir raciocínio similar para cada problema que se quer provar ser NP-complete é inviável.
- $\Rightarrow$ Redução de problemas provê um método alternativo mais simples.

Redução e Problemas NP-Complete

- Condições para uma linguagem L ser NP-complete:
  1. Linguagem deve estar em $\mathcal{NP}$.
  2. Linguagem deve ser NP-hard.
- Solução usual para condição 1: projetar uma NTM que decide a linguagem em tempo polinomial (não-determinístico).
- Condição 2 é mais difícil: requer mostrar que toda linguagem em $\mathcal{NP}$ é redutível a L em tempo polinomial (determinístico).
- Ao invés de produzir reduções diretas para L, um problema NP-complete pode ser usado como passo intermediário.

Redução e Problemas NP-Complete

Teorema abaixo permite diminuir o número de reduções necessárias para apenas uma.

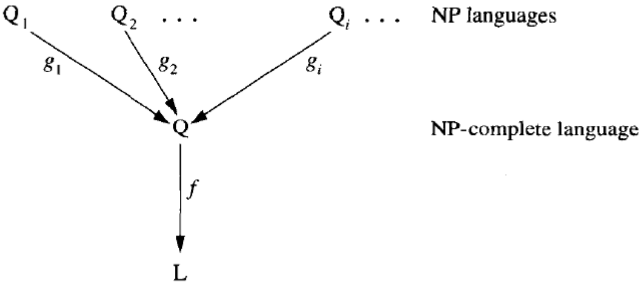
Teorema 16.1.1 (Sudkamp)

Seja Q uma linguagem NP-complete. Se Q é redutível a L em tempo polinomial, então L é NP-hard.

- Seja f uma função computável que reduz Q para L em tempo polinomial.
- Seja Q_i qualquer linguagem em $\mathcal{NP}$.
- Como Q é NP-complete, existe uma redução g_i de Q_i para Q.
- A função composta $f \circ g_i$ é uma redução de Q_i para L.

Redução e Problemas NP-Complete

Processo de dois passos do teorema anterior pode ser ilustrado como abaixo.



Vamos usar reduções de SAT para mostrar que outros problemas são NP-complete.

3-SAT

- O problema de 3-Satisfatibilidade (3-SAT) é um subproblema de SAT que também é NP-complete.
- Uma fórmula é dita 3-CNF se ela está em CNF e cada cláusula contém exatamente 3 literais.
- Objetivo de 3-SAT é determinar se uma fórmula 3-CNF é satisfatível.
- A condição necessária para mostrar que 3-SAT é NP-hard é descrita pela redução abaixo.

Redução	**Instâncias**	**Condição**
SAT	fórmula CNF u	u é satisfatível
para	$\downarrow$	se e somente se
3-SAT	fórmula 3-CNF u'	u' é satisfatível

3-SAT

Teorema 16.2.1(Karp, 1971)

O problema 3-SAT é NP-complete.

- 3-SAT está em $\mathcal{NP}$: uma NTM que resolve SAT (para fórmulas CNF arbitrárias) também resolve o subproblema 3-SAT (para o caso especial de fórmulas 3-CNF).
- Resta mostrar que 3-SAT é NP-hard.
- Redução do slide anterior requer conversão de u em u' preservando a satisfatibilidade das fórmulas.
- Para essa redução, basta converter cada cláusula separadamente.

3-SAT

- Seja w uma cláusula de u e w' a conversão.
- A conversão deve preservar a satisfatibilidade de w e w'.
- Existem 4 casos possíveis para a transformação, conforme o tamanho de w.
- Caso tamanho 3: Trivial. $w = v_1 \vee v_2 \vee v_3 = w'$.
- Caso tamanho 2: A cláusula $w = v_1 \vee v_2$ é convertida em

$$w' = (v_1 \vee v_2 \vee y) \wedge (v_1 \vee v_2 \vee \neg y).$$

- Caso tamanho 1: A cláusula $w = v_1$ é convertida em

$$w' = (v_1 \vee y \vee z) \wedge (v_1 \vee \neg y \vee z) \wedge (v_1 \vee y \vee \neg z) \wedge (v_1 \vee \neg y \vee \neg z).$$

- Caso tamanho $n > 3$: requer a inclusão de $n - 3$ novas variáveis. (Detalhes no livro do Sudkamp.)

3-SAT

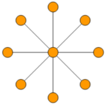
- A transformação de uma cláusula em uma fórmula 3-CNF é polinomial no número de literais da cláusula.
- Trabalho total da redução é a soma do trabalho da transformação de cada cláusula.
- $\Rightarrow$ Redução é polinomial no número de cláusulas da fórmula original.
- Não é o caso que qualquer subproblema de um problema NP-complete também é automaticamente NP-complete.
- Por exemplo, 2-SAT tem uma solução em tempo polinomial determinístico.

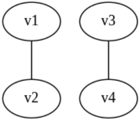
Vertex Cover Problem (VCP)

Input: Grafo não-direcionado $G = (\mathrm{N}, \mathrm{A})$, natural k.

Output: sim; se existe uma cobertura por vértices de G contendo k vértices.
não; caso contrário.

- Uma cobertura por vértices é um conjunto $\mathrm{VC} \subseteq \mathrm{N}$ aonde para toda aresta $[u, v] \in \mathrm{A}$, pelo menos um dos vértices u ou v está em VC.
- O tamanho da cobertura por vértices de um grafo não tem relação com o número de vértices ou arestas no grafo.





Reduções de 3-SAT

Teorema 16.3.1(Karp, 1971)

O VCP é NP-complete.

- VCP está em $\mathcal{NP}$: uma NTM escolhe um subconjunto de N com tamanho k e testa se esse subconjunto cobre as arestas do grafo.
- A condição necessária para mostrar que VCP é NP-hard é descrita pela redução abaixo.

Redução	**Instâncias**	**Condição**
3-SAT	fórmula 3-CNF u	u é satisfatível
para	$\downarrow$	se e somente se
VCP	$G = (\mathrm{N}, \mathrm{A})$, natural k	G tem uma cobertura
		de tamanho k

Reduções de 3-SAT

- Para qualquer fórmula 3-CNF u, devemos construir um grafo G que possui uma cobertura por vértices sss u é satisfatível.
- Vamos representar uma fórmula 3-CNF u como abaixo

$$u = (u_{1,1} \vee u_{1,2} \vee u_{1,3}) \wedge \cdots \wedge (u_{m,1} \vee u_{m,2} \vee u_{m,3}).$$

- Cada $u_{i,j}$, $1 \leq i \leq m$ e $1 \leq j \leq 3$ é um literal sobre o conjunto $\mathrm{V} = \{x_1, \ldots, x_n\}$ de variáveis Booleanas.
- Índice i indica a cláusula e índice j indica o literal na cláusula.
- Redução consiste em construir um grafo G a partir da fórmula 3-CNF u aonde a satisfatibilidade de u é equivalente à existência de uma cobertura por vértices em G contendo $n + 2m$ vértices.

Reduções de 3-SAT

- Transformar a pergunta de satisfatibilidade na pergunta de cobertura de vértices requer a representação de atribuições Booleanas e fórmulas como grafos.
- Os vértices de G consistem dos seguintes conjuntos:
  1. $\{x_i, \neg x_i \mid 1 \leq i \leq n\}$;
  2. $\{u_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$.
- As arestas de G são dadas pela união de:
  1. $\mathrm{T} = \{[x_i, \neg x_i] \mid 1 \leq i \leq n\}$;
  2. $\mathrm{C}_k = \{[u_{k,1}, u_{k,2}], [u_{k,2}, u_{k,3}], [u_{k,3}, u_{k,1}]\}$, para $1 \leq k \leq m$;
  3. $\mathrm{L}_k = \{[u_{k,1}, v_{k,1}], [u_{k,2}, v_{k,2}], [u_{k,3}, v_{k,3}]\}$, para $1 \leq k \leq m$;

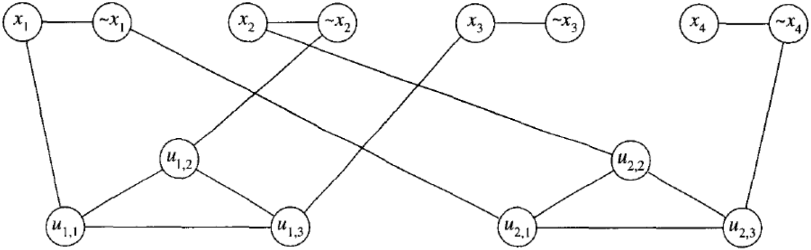
  aonde $v_{k,j}$ é o literal x_i que ocorre na posição $u_{k,j}$ da fórmula.

Reduções de 3-SAT

- Uma aresta em T conecta as duas possibilidades de um literal x_i.
- Uma cobertura de vértices deve incluir pelo menos um vértice do par x_i, $\neg x_i$.
- Pelo menos n vértices são necessários para cobrir os arcos em T.
- Uma cobertura de T com n vértices seleciona exatamente um dos x_i ou $\neg x_i$.
- Isso pode ser visto como uma atribuição de valores para as variáveis em V.

Reduções de 3-SAT

- Cada cláusula de u gera um subgrafo triangular em C_k.
- Um conjunto de vértices que cobre C_k deve conter ao menos dois vértices.
- Portanto, a cobertura dos arcos de G requer pelo menos $n + 2m$ vértices.
- As arestas L_k fazem a conexão entre cobertura e satisfatibilidade.
- Exemplo: o grafo abaixo representa a redução da fórmula $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_4)$.



Reduções de 3-SAT

- Seja uma fórmula 3-CNF u com m cláusulas e n variáveis.
- O grafo G construído pela redução possui $3m + n$ vértices e $6m + n$ arestas.
- A fórmula u é satisfatível sss G possui uma cobertura de vértices com $n + 2m$ vértices.
- A redução é polinomial em relação ao tamanho da fórmula.
- $\Rightarrow$ VCP é NP-hard.

Reduções de 3-SAT

Usando uma redução a partir de 3-SAT é possível chegar nos resultados abaixo.

Teorema 16.3.2 (Karp, 1971)

O Problema do Circuito Hamiltoniano é NP-complete.

Teorema 16.3.3

O Problema da Soma de Subconjunto é NP-complete.

- As reduções dos problemas acima requerem uma troca de domínio.
- Porém, muitas vezes é possível ficar no mesmo domínio.
- Regra de ouro: procurar um problema similar para tornar a redução mais simples.

Problemas de Otimização

- Problema de Decisão: determinar se uma solução existe.
- No entanto, existem vários problemas em que se quer determinar uma solução ótima.
- O conceito de ótimo é dependente do critério de otimização: minimizar custo, maximizar lucro, etc.
- Problema de Otimização $\neq$ Problema de Decisão.
- No entanto, questões de complexidade são as mesmas para ambos os tipos de problemas.

Problemas de Otimização

- TSP (*Traveling Salesman Problem*): versão de otimização do HCP (*Hamiltonian Circuit Problem*).
- Corresponde à busca pelo tour de menor custo em um grafo.
- Nome do problema descreve um vendedor que busca visitar todas as cidades exatamente uma vez, percorrendo a menor distância possível.
- TSP pode ser convertido em um problema de decisão colocando-se um limite de distância nas instâncias.

TSP de Decisão

Input: Grafo direcionado com pesos $G = (N, A, w)$, natural k.

Output: sim; se G tem um tour de custo $\leq k$
não; caso contrário.

Problemas de Otimização

- Uma solução para o problema de decisão pode ser utilizado de forma iterativa para produzir uma solução do problema de otimização.
- Seja n o número de nós de G, l a soma dos n arcos de menor custo e u a soma dos n arcos de maior custo.
- O custo de qualquer tour de G deve estar entre l e u.
- O tour mínimo pode ser encontrado iterando-se k de l a u e parando quando a primeira solução for obtida.

Problemas de Otimização

Teorema 16.5.1 (Sudkamp)

O TSP é NP-complete.

- O HCP pode ser visto como um subproblema do TSP.
- Uma redução de HCP para TSP é trivial.
- Basta associar um peso 1 a todos os arcos e tomar $k = card(\mathrm{N})$.

Algoritmos de Aproximação

- A classe $\mathcal{NP}$ tem significado tanto teórico quanto prático.
- Problemas NP-complete surgem naturalmente em inúmeras áreas: reconhecimento de padrões, escalonamento, projeto de redes, teoria dos grafos, etc.
- Considere o TSP. Sabendo que esse problema é NP-complete, como proceder?
- Pergunta 1: o fato do problema ser NP-complete é relevante para a situação em particular?
  - Se a rota contém bem poucas cidades, a complexidade assintótica da solução é irrelevante.
- Pergunta 2: é possível melhorar o desempenho da solução exaustiva?
  - Uma busca exaustiva por todas as sequências de nós requer examinar n^{n-1} caminhos em potencial.
  - Um algoritmo de programação dinâmica consegue resolver o problema em tempo $O(n2^n)$. Continua exponencial mas é mais eficiente.

Algoritmos de Aproximação

- Pergunta 3: é possível reformular o problema como um problema similar que pode ser resolvido em tempo polinomial?
  - As soluções para o novo problema podem não ser tours ótimos mas ainda podem ser úteis em alguns casos.
- Nessa reformulação é necessário responder duas perguntas:
  - Existe um algoritmo polinomial que resolve o problema aproximado sem o tour ficar arbitrariamente maior que o tour ótimo?
  - Se a resposta da pergunta anterior é não, que condições poderiam ser adicionadas ao problema para se obter uma solução aproximada em tempo polinomial? (Isto é, é possível simplificar o problema para torná-lo mais fácil?)

Algoritmos de Aproximação

- Solução de um problema de otimização: custo da solução.
- $c(p_i)$: custo da solução do algoritmo de aproximação para instância p_i.
- $c^*(p_i)$: custo da solução ótima para instância p_i.
- A qualidade de um algoritmo de aproximação é medida pela comparação dos custos das soluções.

Definição 16.6.1 (Sudkamp)

Um algoritmo que produz soluções aproximadas para um problema de otimização é um algoritmo de α-aproximação se

1. o problema é de minimização e $c(p_i) \leq \alpha \cdot c^*(p_i)$, ou
2. o problema é de maximização e $c^*(p_i) \leq \alpha \cdot c(p_i)$;

para uma constante $\alpha \geq 1$ e todas as instâncias p_i do problema.

Algoritmos de Aproximação

- Um algoritmo de 2-aproximação para um problema de minimização produz soluções com custo no máximo 2 vezes a solução ótima.
- Se o problema for de maximização, o custo é no mínimo metade da solução ótima.
- Com a Definição 16.1.1, podemos perguntar sobre o TSP:
  - Existe um algoritmo de α-aproximação de tempo polinomial para o TSP?
  - Se não existir, quais as mudanças necessárias ao problema para se obter tal algoritmo?

Algoritmos de Aproximação

Teorema 16.6.2 (Sudkamp)

Se $\mathcal{P} \neq \mathcal{NP}$, não existe um algoritmo de α-aproximação de tempo polinomial para o TSP.

- O algoritmo de aproximação não existe porque ele poderia ser usado para resolver o HCP em tempo polinomial.
- Isso é impossível sob a hipótese que $\mathcal{P} \neq \mathcal{NP}$.
- No entanto, é possível construir um algoritmo de 2-aproximação se os pontos do TSP estiverem em um plano 2D e a distância entre eles for a Euclidiana.
- Esse tipo de problema é chamado de *Euclidean TSP* (ETSP).

Aula 07 – Problemas NP-Complete

Prof. Eduardo Zambon

Departamento de Informática (DI)
Centro Tecnológico (CT)
Universidade Federal do Espírito Santo (Ufes)

Algoritmos e Fundamentos da Teoria de Computação (ToCE)
Engenharia de Computação