31/10/18

Módulo 04: Problemas NP Completos

Sebastian

20 pontos de prova
05 pontos de trabalho } 25 pontos

Visão Geral

- problemas fáceis x difíceis
  complexidade polinomial x complexidade exponencial
- objetivo é identificar problemas difíceis.
  - algoritmos exponenciais;
  - heurísticas;
  - algoritmos aproximados.

- Diferenças sutis:
  Dado um grafo $G(V,A)$ e um inteiro K, existe uma coloração de G com no máximo K cores?

a) $K=2$
   Grafo bipartido (fácil)

b) $K>2$
   Problema da coloração de grafos (difícil)

c) caminho mais curto em um grafo (fácil)

d) Caminho mais longo em um grafo (difícil)

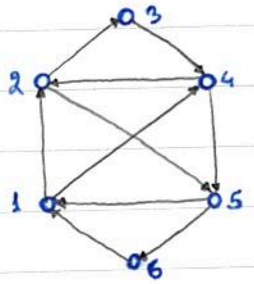
e) Caminho Hamiltoniano (difícil) - procurar um ciclo simples que passa por todos os vértices de um grafo.
   Logo:

f) grau máximo dos vértices igual a 2 (fácil)

g) grau máximo >2 (ciclo hamiltoniano = difícil)

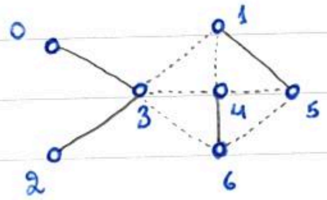
h) Existe ciclo de Euler em G? (passa por todas as arestas)

É possível ter ciclo euleriano se todos os vértices do grafo tiverem grau par. (fácil)



1, 2, 3, 4, 5, 6, 1, 4, 2, 5, 1

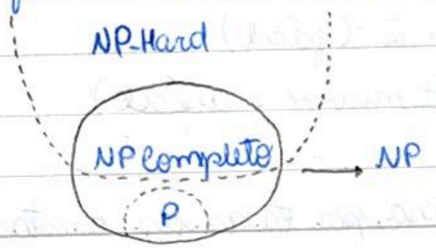
* Emparelhamento Máximo



cobertura por vértices: {3, 4, 5}
cobertura por arestas: {0, 3}, {0, 2}, {1, 5}, {4, 6}

* classe de problemas NP (não determinístico polinomial)
- Problemas de decisão sim ou não.
- Mas muitos problemas são de otimização.
- São problemas de decisão cuja solução pode ser verificada em tempo polinomial com algoritmo determinístico ou problemas de decisão cuja solução pode ser determinada em tempo polinomial com algoritmo não determinístico. (são equivalentes).

Para provarmos que um problema é NP completo, antes devemos provar que ele é da classe NP.



$P \subset NP$

12/11/18

Transformação Polinomial:

Instância P1 → Instância P2 → Solução P2 → Solução P1

↑ transformação Polinomial ↑ Algoritmo A2 ↑ Transformação Polinomial (Nada ou not)

Se resolvo P2 eu consigo resolver P1 também! { usamos a mesma resposta = nada / ou negamos a resposta = not

* Problemas NP completo e NP Difícil
- um problema P1 é NP completo se:
  1. $P1 \in NP$
  2. $P' \propto P1$ para todo $P' \in NP$
     ↳ se reduz (transitividade) $P1 \propto P2 \propto P3$ logo: $P1 \propto P3$

Simplificando 2: 2' existe $P' \in$ NP-Completo tal que $P' \propto P1$.
- Um problema é NP difícil se satisfaz a pelo menos ao item 2.

* Exemplo do clique:
G possui clique de tamanho $>= K$ se e somente se $\bar{G}$ tem conjunto independente de tamanho $\geq K$.

Diz-se que clique $\underset{\text{se reduz}}{\propto}$ Conjunto independente

* Teoremas:
1. Se qualquer problema NP-completo puder ser resolvido em Tempo polinomial, então P = NP.

2- Se qualquer problema em NP não puder ser resolvido em tempo polinomial, então nenhum problema NP-completo pode ser resolvido em tempo polinomial.

não conhecemos nenhum dos casos. estão em aberto.

* Problema da parada

Dado um algoritmo determinístico qualquer A com entrada e, A termina ou entra em loop infinito?

Exemplo de problema que é NP difícil, mas não é NP completo

SAT $\propto$ Parada

- Entrada do SAT uma expressão booleana S qualquer
- Construir instância do problema da parada consistindo de um algoritmo A e entrada E.
  - Faça E=S
  - A- algoritmo:

    Avalie todas as 2^n possibilidades de atribuições de valores booleanas para as n variáveis que compõem S.

    Se S for satisfazível (alguma possibilidade resulta em S=1), o algoritmo termina. return.

    Senão ele entra em loop infinito. while.
  - S é satisfazível se e somente se A para.

* Teorema de Cook (71):

SAT está em P se e somente se P=NP.

NP-completo aparece pela primeira vez em Karp (72).

21 problemas NP-completos fazendo reduções a partir de SAT

- Mostrar que caixeiro viajante (PCV) é NP-completo

1- Mostrar que PCV $\in$ NP

2. Mostrar que existe P $\in$ NP-completo tal que $P \propto PCV$

PCV por definição é um grafo completo

P = ciclo de Hamilton (CH) → mostrar que $CH \propto PCV$

Grafo G(V,A)

Fazer um ciclo que passe por todos os vértices.

Fazendo o grafo completo:

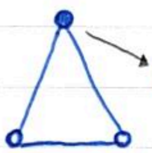
{ Arestas que existem coloco peso 1;
  Arestas que não existem coloco peso 2, por exemplo.

seja n, número de arestas, verificamos se existe caminho de tamanho n.

Ao contrário, também provamos:

Pegamos todos os pesos 1 do grafo completo;

se tiverem tamanho n, teremos o grafo G(V,A).

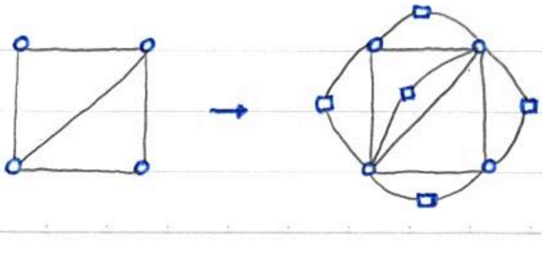


conjunto dominante

cobertura por vértices.

todos estão a distância um de um vértice.

cobre todas as arestas.



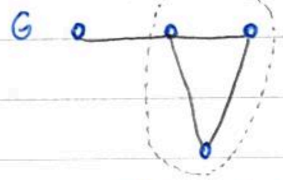
14 / 11 / 18

Se houver vértices isolados (grau zero) ignoramos na hora da transformação.

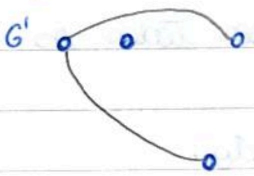
Ou:

A redução pode ser modificada para considerar estes casos fazendo $n = K + X$ onde X é o número de vértices com grau zero.

* Provar que cobertura por vértices é NP-completo



clique $K=3$



G' é complementar de G

1) Se G tem clique de tamanho $\geq K$, então G' tem cobertura por vértice de tamanho $\leq K' = |V| - K$.

2) Se G' tem cobertura por vértices de tamanho $\leq K' = |V| - K$, então G tem clique de tamanho $\geq K$.

* Dados dois Grafos $G1(V1, E1)$ e $G2(V2, E2)$ não direcionados, G1 é subgrafo de G2? Há função que mapeia os vértices de G1 nos vértices de G2, tal que há uma aresta (u, v) em G1 se e somente se há aresta $(f(u), f(v))$ em G2?

Prove que o problema de Isomorfismo de Subgrafo (IS) é NP-completo, partindo da premissa que o clique (CQ) é NP-completo.

$CQ \propto IS$

Seja uma instância genérica do Cq, dada por um grafo $G(V,A)$ e um valor K.

Precisamos transformá-la, em tempo polinomial em uma instância do IS definida por dois grafos $G1$ e $G2$.

Faça $G2 = G$

Construa $G1$ como um grafo completo com K vértices

$O(|V| + |E| + K^2)$

Provar que:

1) Se G tem clique de tamanho $\geq K$, então $G1$ é subgrafo de $G2$

2) Se $G1$ é subgrafo de $G2$, G tem clique de tamanho $\geq K$

Note que o problema origem deve ser genérica, mas a transformação não precisa ser genérica, pode ser específica.

Prova 1: Se G tem clique de tamanho $\geq K$, G tem um subgrafo completo de tamanho $\geq K$. Como $G2 = G$, $G2$ também tem um subgrafo completo de tamanho $\geq K$. Como $G1$, por construção, é um grafo completo de K vértices, ele é isomorfo a este subgrafo de $G2$.

Prova 2: Como $G1$, por construção, é um grafo completo de K vértices, ele forma um clique. Como $G1$ é subgrafo de $G2$, $G2$ tem um clique de tamanho $\geq K$. Como $G = G2$, G também tem um clique de tamanho $\geq K$.

* Prove que o problema clique é NP-completo sabendo que:

NP-completo

SAT

3-CNF SAT

Existe um clique de tamanho no mínimo K?

Algoritmo não determinista

```
Verifica ND-clique (V, A, S, k) {
    if ( |S| < K ) return false;
    for i = 1 ate |S| {
        for j = (i+1) ate |S| {
            if (V[i], V[j]) ∈ A return false; }
    }
    return true;
}
```

ou:

Algoritmo determinista

```
Resolve ND-clique (V, A, K) {
    S = ∅;
    for i = 1 to |V| {
        inclui = escolhe (V[i], True, False);
        if (inclui = True);
            S = S + V[i];
    }
    if verificaND-clique (V, A, S, K) return sucesso;
    else return insucesso;
```

tilibra

3 CNF SAT $\propto$ Clique

Seja uma instância genérica do 3 CNF SAT dada por:

$S = C1 * C2 * \dots C_m$

onde $C_i = (l_1^i + l_2^i + l_3^i)$

Uma instância do clique, definida por um grafo $G(V,A)$ e um inteiro K, é construída da seguinte maneira:

- Faça $K = m$
- Construa um grafo $G(V,A)$ da seguinte maneira:
  - Para cada cláusula $C_i = (l_1^i + l_2^i + l_3^i)$, adicione 3 vértices correspondentes em V: $v_1^i\ v_2^i\ v_3^i$
  - Uma aresta é criada entre vértices v_r^i e v_s^j se:
    - + v_r^i e v_s^j correspondem a literais de cláusulas diferentes em S, ou seja $i \neq j$
    - + O literal correspondente a v_r^i não é a negação do literal correspondente a v_s^j, ou seja, $l_r^i \neq \overline{l_s^j}$

Mostrar que:

- Se $S = 1$, então G tem clique de tamanho $\geq K = m$.
- Se G tem clique de tamanho $\geq K = m$, então $S = 1$.

Se $S = 1$:

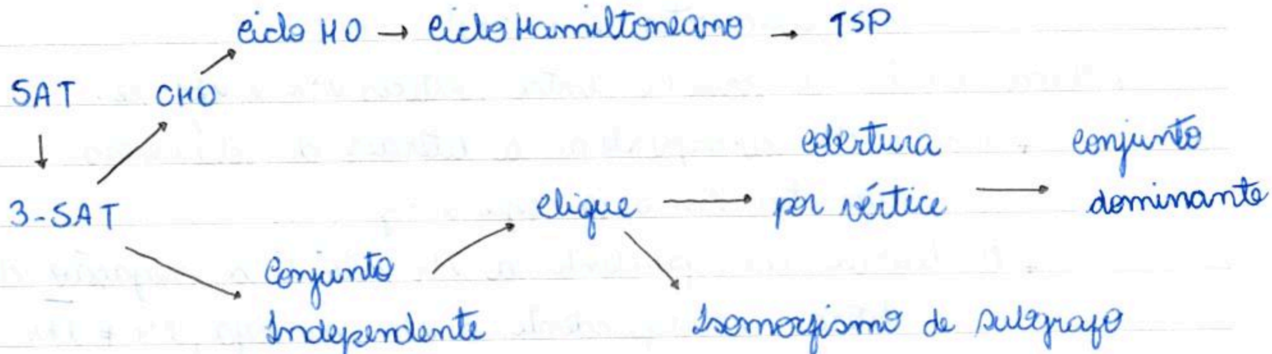
- Para cada C_i, $i = 1 \dots m$, existe pelo menos um l_r^i, $r = 1, 2, 3$ tal que $l_r^i = 1$
- Seja L o conjunto desses literais l_r^i
- Seja V' o conjunto dos vértices v_r^i correspondentes a $l_r^i \in L$
- V' forma um clique de tamanho $K = m$ no Grafo G, pois:
  - Se $l_r^i = 1$ e $l_s^j = 1$ para l_r^i e $l_s^j \in L$, então, por construção:

    $i \neq j$ e $l_r^i \neq \overline{l_s^j}$
  - Logo $(v_r^i, v_s^j) \in A$

- Como isto vale para todas as n cláusulas de S
  Existe clique de tamanho no mínimo $K = n$ em G

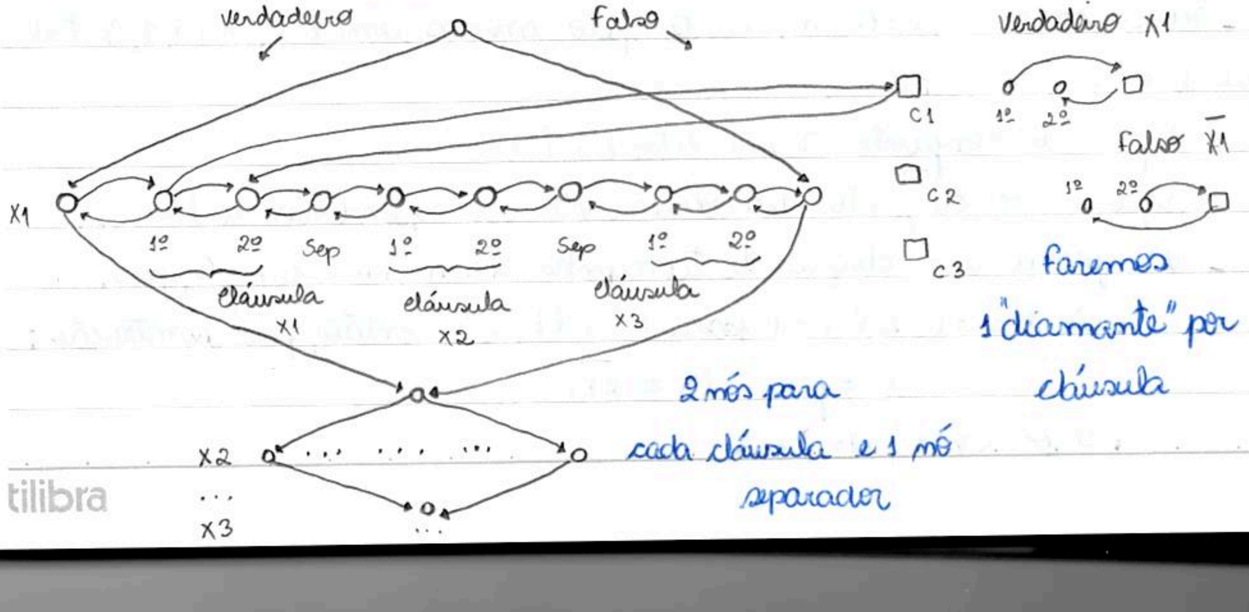
- Se G tem clique V' de tamanho $\geq K = n$
  - V' contém 1 vértice para cada cláusula de S, por construção.
  - Faça $l^i_r = 1$ para todo $v^i_r \in V'$
    logo $C_i = 1$ para todo $i = 1 \dots n$
    $S = 1$

* Resumo:



E pelo teorema de Cook, todos se resumem/reduzem à SAT!

- Caminho Hamiltoneano em grafo orientado - CHO
  CHO está em NP
  CHO é NP-completo. Reduzir 3-SAT a CHO
  $(x_1, x_2, \overline{x_3})\ (\overline{x_1}, \overline{x_2}, x_3)\ (x_1, x_2, x_3)$

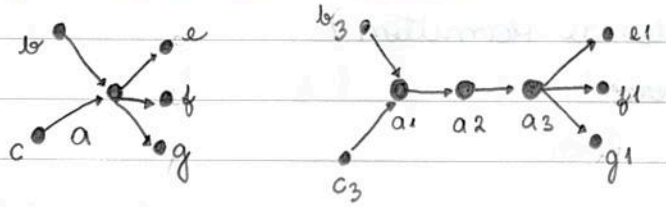


faremos 1 "diamante" por cláusula

2 nós para cada cláusula e 1 nó separador

* ciclo Hamiltoneano
Está em NP
Está em NP completo
Reduzir ciclo Hamiltoneano Orientado a ciclo Hamiltoneano

Triplicaremos cada vértice



* SAT se reduz a 3-SAT

Para cada cláusula $ci = \{z1\}$ com 1 literal criar quatro cláusulas usando duas variáveis artificiais $yi1$ e $yi2$ da seguinte forma:

$$\{z1\ yi1\ yi2\}, \{z1, \overline{yi1}, yi2\}, \{z1, yi1, \overline{yi2}\}, \{z1, \overline{yi1}, \overline{yi2}\}$$

Para cada cláusula $ci = \{z1, z2\}$ com dois literais criar duas cláusulas usando uma variável artificial $yi1$ da seguinte forma:

$$\{z1, z2, yi1\}, \{z1, z2, \overline{yi1}\}$$

Para cada cláusula $ci = \{z1, z2, z3\}$ com três literais criar essa mesma cláusula.

Para toda cláusula $ci = \{z1, z2 \dots zK\}$ com $K > 3$ literais criar $K-2$ cláusulas usando $K-3$ variáveis artificiais $yi1, yi2, \dots yiK-3$ da seguinte forma:

$$\{z1, z2, yi1\}, \{\overline{yi1}, z3, yi2\} \{\overline{yi2}, z4, yi3\}, \dots \{yiK-3, zK-1, zK\}$$

* Algoritmos Aproximativos / Aproximados
- Problemas NP Difíceis. O que fazer?
- tamanho da entrada pequeno
  Algoritmo exponencial

- Concentrar no caso médio
- Algoritmo com podas (ciclo de Hamilton)
  Explora simetria da árvore

- branch-and-bound
  Interromper a procura caso vejamos que não vai dar em nada.

- Heurísticas e algoritmos aproximados:
  fornecemos um limite para quão ruim a solução pode ser.

$$\frac{S}{S^*} \leq R(n)$$ problemas de minimização

S é a solução produzida e S^* é a solução ótima.
Para problemas de maximização, a razão é invertida.

Se $P \neq NP$ é possível provar que não tem algoritmo aproximado.

- Problema da cobertura por vértice:

```
Input G;
S ← {}; E' ← E[G]
While E' is not empty do
    Let (u, v) be an arbitrary edge of E'
    S ← S ∪ {u, v}
    Remove from E' every edge incident on either u or v.
return S
```

tilibra

O algoritmo é polinomial com razão de aproximação $R(n)=2$ para este problema.

Duas arestas de A não compartilham extremidade, logo não há duas arestas em A que podem ser cobertas pelo mesmo vértice em S^*.

Logo:

$|S^*| \geq |A|$ A é limite inferior para o tamanho da cobertura de vértices ótima.

Temos que:

$|S| = 2|A|$

O que nos leva a: $|S| = 2|A| \leq 2|S^*|$

$$\frac{S}{S^*} \leq 2$$

* Problema do Caixeiro Viajante

Dado um grafo não direcionado, completo ponderado, cujos pesos das arestas satisfaçam:

$d(i,j) = d(i,j) \quad \forall i,j \in V$

$d(i,j) \geq 0 \quad \forall i,j \in V$

$d(i,j) \leq d(i,k) + d(k,j) \quad \forall i,j,k \in V$ desigualdade triangular

• AGM é um limite inferior para solução ótima do PCV em um grafo GVA

$S^* > AGM$ (pesos positivos)

Algoritmo de Prim para obter AGM: $O(V^2)$

21/11/22

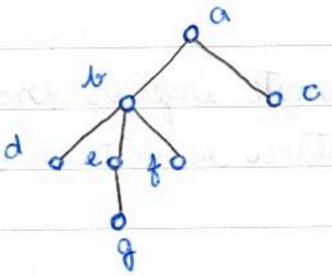
Algoritmo:

Escolha uma raiz r em V[G]

Faça MST-Prim (G,C,r)

Liste os vértices (busca em profundidade) em pré order

retorne ciclo hamiltoneano com vértices ordenados como em L.



Ciclo:

a b d e g f c a

$AGM \leq S^*$

$$S \leq 2 \times AGM \leq 2 \times S^*$$

$$\frac{S}{S^*} \leq 2$$

* Algoritmo de Christophides

- Conceito de grafo euleriano

Algoritmo:

1. Construa AGM
2. Dobre suas arestas para encontrar grafo euleriano
3. Encontre caminho euleriano (busca em profundidade)
4. Converta em caixeiro viajante

- Premissa: não precisa dobrar as arestas da AGM para encontrar grafo euleriano

Trocaremos o passo 2 do algoritmo:

2. Adicione a AGM o casamento mínimo com pesos dos vértices de grau ímpar

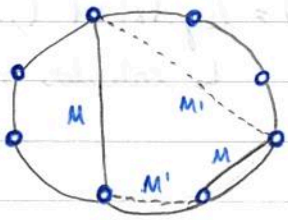
Esse algoritmo tem razão de 1.5 com tempo polinomial

M min : o custo total das arestas adicionadas pelo casamento mínimo

$$S = AGM + M\,min$$

Sejam M e M' dois casamentos dos vértices de grau ímpar da AGM tais que:

$$M + M' \leq S^*$$



$$\frac{S}{S^*} \leq \frac{3}{2} \text{ (contas slide)}$$

* Problema da cobertura de conjuntos

Dados um conjunto finito X e uma família F de subconjuntos de X, tal que todo elemento de X pertence a pelo menos um subconjunto de F.

$$X = \bigcup_{S \in F} S$$

O problema da cobertura de conjuntos consiste em encontrar um subconjunto de tamanho mínimo C de F cujos membros "cubram" todos os elementos de X:

$$X = \bigcup_{S \in C} S$$

Algoritmo:

Greedy-set-cover (X, F)

$U = X$

$C = \emptyset$

while $U \neq \emptyset$ do

select $S \in F$ that maximizes $|S \cap U|$

$U = U - S$

$C = C \cup \{S\}$

return C

complexidade:

$O(|S||X||F|) \min(|X|, |F|)$

Escolhe em cada fase o subconjunto restante que cobre o maior número de elementos ainda descobertos.

A razão de aproximação é $R(n) = \log |X| + 1$ (cresce com a entrada, mas devagar).

↳ cálculos nos slides

Ideia geral:

Atribua custo 1 em cada vez que cobrir elementos em S.

Distribua esse custo entre o número de elementos que estão sendo cobertos pela primeira vez.

logo: $c_x = \dfrac{1}{|S_i - (S_1 \cup S_2 \cup \ldots \cup S_{i-1})|}$

Um esquema de aproximação de tempo polinomial é um esquema que:

para qualquer $\varepsilon > 0$ fixo o esquema executa em tempo polinomial em função do tamanho n da entrada.

Tempo pode variar para diferentes valores de ε.