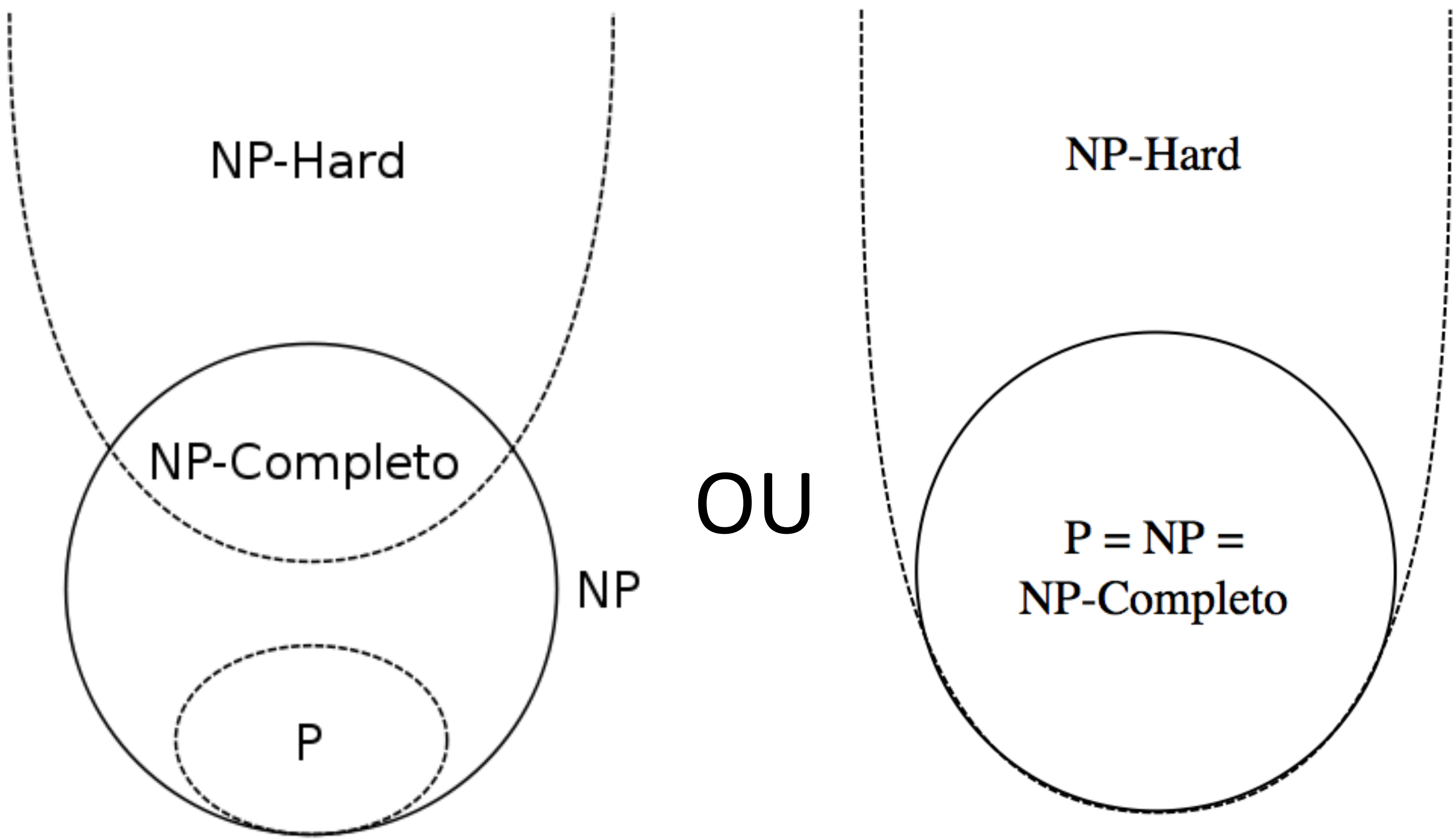
Problemas NP Completos

Visão Geral

- Problemas fáceis X problemas difíceis
  - Complexidade polinomial X complexidade exponencial
- Objetivo: identificar problemas difíceis
  - Opção por heurísticas
  - Opção por algoritmos aproximados (próximo módulo)
- Classe NP-Completo: problemas difíceis ??
  - Questão em aberto
  - Vamos assumir que sim

Visão Geral



Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

1a) Dado um grafo G(V,A) e um inteiro k, existe um caminho entre vértices u,v de tamanho no máximo k?

1b) Dado um grafo G(V,A) e um inteiro k, existe um caminho entre vértices u,v de tamanho no mínimo k?

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

1a) Dado um grafo G(V,A) e um inteiro k, existe um caminho entre vértices u,v de tamanho no máximo k?

Fácil (caminho mais curto)

1b) Dado um grafo G(V,A) e um inteiro k, existe um caminho entre vértices u,v de tamanho no mínimo k?

Difícil (caminho mais longo)

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

2) Dado um grafo G(V,A) e um inteiro k, existe uma coloração de G com no máximo k cores?

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

2) Dado um grafo G(V,A) e um inteiro k, existe uma coloração de G com no máximo k cores?

2a) $k=2$

Fácil (grafo bipartido)

2b) $k > 2$

Difícil (problema da coloração de grafos)

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

3) Dado um grafo G(V,A), existe um ciclo simples que passa todos os vértices sem repetir nenhum?

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

3) Dado um grafo G(V,A), existe um ciclo simples que passa todos os vértices sem repetir nenhum?

3a) grau máximo igual a 2

Fácil

3b) grau máximo > 2

Difícil (problema do ciclo hamiltoniano)

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

4a) Dado um grafo G(V,A), existe um ciclo de hamilton em G?

4b) Dado um grafo G(V,A), existe um ciclo de Euler em G?

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

4a) Dado um grafo G(V,A), existe um ciclo de hamilton em G?

Difícil

4b) Dado um grafo G(V,A), existe um ciclo de Euler em G?

Fácil

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

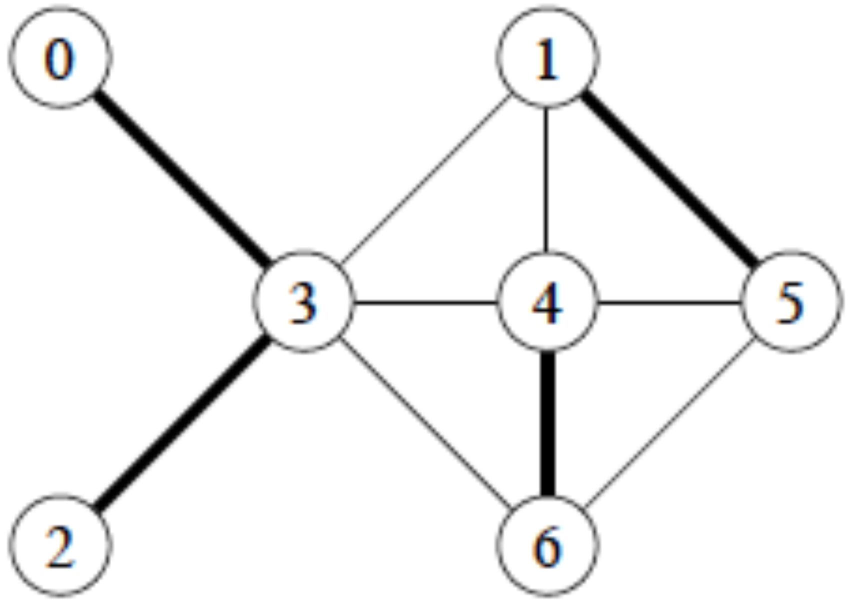
5a) Dado um grafo G(V,A) e um inteiro k, existe uma cobertura de arestas de tamanho no máximo k?

Uma cobertura de arestas de um grafo G = (V,A) é um subconjunto A’ de A tal que todo vértice de V é parte de pelo menos uma aresta de A’

5b) Dado um grafo G(V,A) e um inteiro k, existe uma cobertura de vértices de tamanho no máximo k?

Uma cobertura de vértices de um grafo G = (V,A) é um subconjunto V’ de V tal que toda aresta de A é incidente em pelo menos um vértice de V’

Problemas difíceis vs. Problemas fáceis



Cobertura de Arestas: A' = { (0,3), (0,2), (4,6), (1,5) }
Cobertura de Vértices: V' = {3, 4, 5}

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

5a) Dado um grafo G(V,A) e um inteiro k, existe uma cobertura de arestas de tamanho no máximo k?

5b) Dado um grafo G(V,A) e um inteiro k, existe uma cobertura de vértices de tamanho no máximo k?

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

5a) Dado um grafo G(V,A) e um inteiro k, existe uma cobertura de arestas de tamanho no máximo k?

Fácil

5b) Dado um grafo G(V,A) e um inteiro k, existe uma cobertura de vértices de tamanho no máximo k?

Difícil

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

6a) Dada uma fórmula booleana S na forma normal conjuntiva (CNF), S é satisfatível?

Ex: S = (x1 + x2) (x2 + x3 + x4) (~x1 + x3 + x5)(~x2 + x3 + x4 + x5)

6b) E se S contiver somente cláusulas com 2 literais (S está na forma normal 2-conjuntiva)?

Ex: S = (x1 + x2) (x2 + x4) (~x1 + x3)(~x2 + x3)

Problemas difíceis vs. Problemas fáceis

- Às vezes, a diferença é sutil. Por exemplo:

6a) Dada uma fórmula booleana S na forma normal conjuntiva (CNF), S é satisfatível?

Difícil (PROBLEMA DA SATISFABILIDADE ou SAT)

6b) E se S contiver somente cláusulas com 2 literais (S está na forma normal 2-conjuntiva)?

Fácil (PROBLEMA DO 2-CNF SAT)

Classe NP

- Problemas de Decisão (Sim ou Não)
  - Arcabouço teórico
- Mas muitos problemas são de otimização...
  - problemas de otimização vs. problemas de decisão
    - Determine o circuito simples que passa por todos os vértices com custo total mínimo (otimização)
    - Existe um circuito simples que passa por todos os vértices com custo menor ou igual a k? (decisão)
  - Problema de decisão não é mais difícil que de otimização
    - Se temos evidência que problema de decisão é difícil, provavelmente o problema de otimização relacionado também será

Classe NP

- Problemas de decisão cuja solução pode ser **verificada** em tempo polinomial com **algoritmo determinista**

ou

- Problemas de decisão cuja solução pode ser **determinada** em tempo polinomial com **algoritmo não determinista**

Algoritmo Não Determinista

- Função **escolhe (C)** onde C é um conjunto de alternativas
  - Escolhe a alternativa que leva à solução ótima, caso uma exista
  - Caso não exista solução ótima, escolhe pode retornar qualquer resposta
  - Complexidade: O(1)

  - **Arcabouço teórico: não se esqueça disto!!!**

Algoritmos não deterministas
Exemplos

- Pesquisar um elemento x em um vetor A de n elementos

```
void PesquisaND(A, 1 , n)
{ j ← escolhe(A, 1 , n)
  if (A[j] == x) sucesso; else insucesso;
}
```

- Algoritmo não determinista : O(1)
- Pesquisa sequencial é Ω (n), deterministicamente

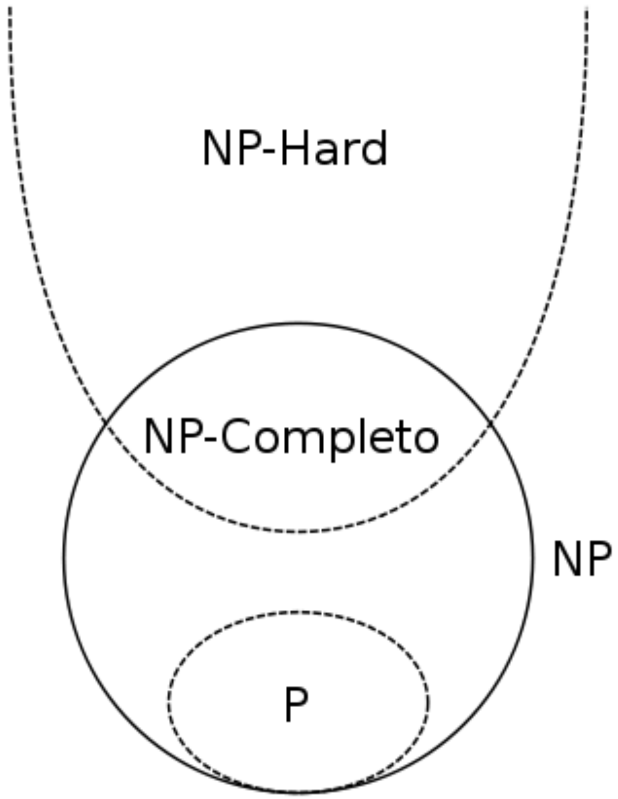
Algoritmos não deterministas
Exemplos

- Problema da Satisfabilidade

```
void AvalND(E, n);
{ for (i = 1; i <= n; i++)
    { x_i ← escolhe (true, false);
      if (E(x_1, x_2, ···, x_n) == true) sucesso; else insucesso;
    }
}
```

- Algoritmo não determinista : O(n)
- Solução determinística: O(2^n)

Visão Geral



Classe NP inclui problemas de decisão que:

• podem ser verificados em tempo polinomial deterministicamente, ou

•podem ser resolvidos em tempo polinomial não deterministicamente (usando a escolhe)

Classe P inclui problemas de decisão que: podem ser resolvidos em tempo polinomial deterministicamente

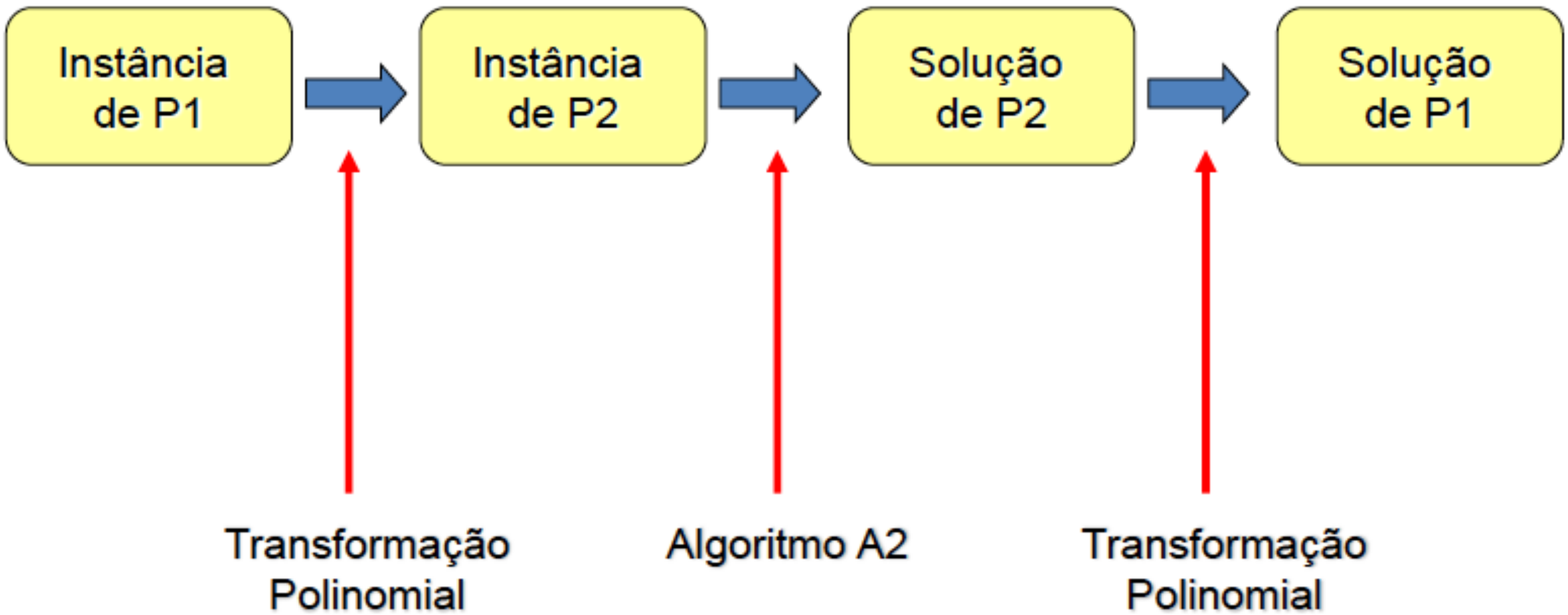
$P \subset NP$ ou $P = NP$??

Transformação Polinomial

- Informalmente, um problema está em NP-Completo se ele está em NP e é “tão difícil” quanto qualquer outro problema em NP

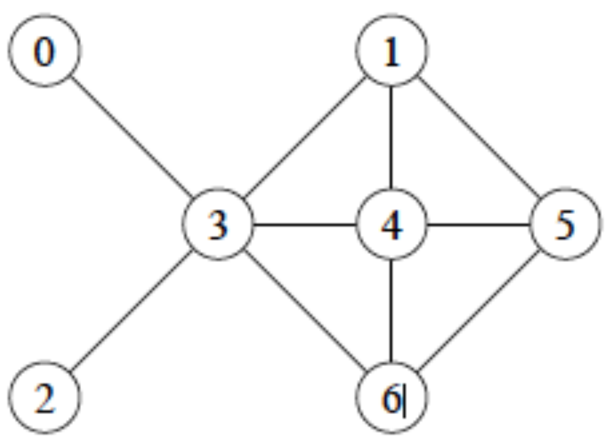
- Esta noção de ser “tão difícil quanto” está ligada ao conceito de transformação/redução polinomial

Transformação Polinomial



Conjunto Independente de Vértices de um Grafo

- Dado um grafo G(V,A), um subconjunto de vértices V' forma um conjunto independente se todo par de vértices em V' é não adjacente, ou seja para todo u, v $\in$ V temos que (u,v) $\notin$ A.

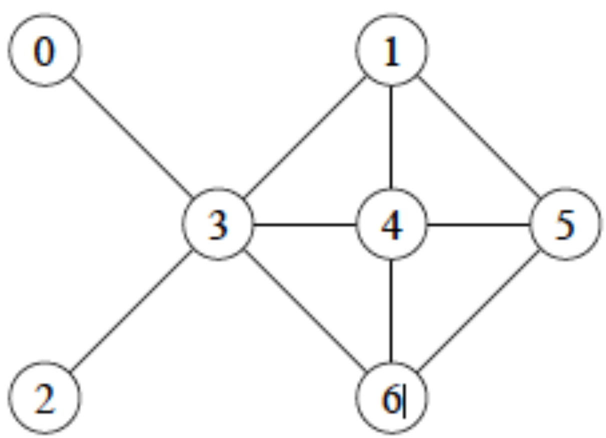


V' = {0, 2, 1, 6) forma um conjunto independente

- Problema de decisão: dados um grafo G(V,A) e um inteiro k, existe um conjunto independente com no mínimo k vértices?

Clique de um grafo

- Dado um grafo G(V,A), um subconjunto de vértices V' forma uma clique se todo par de vértices em V' é adjacente, ou seja para todo u, v $\in$ V temos que (u,v) $\in$ A



V' = {1, 4, 5) forma uma clique

- Problema de decisão: dados um grafo G(V,A) e um inteiro k, existe uma clique em G com no mínimo k vértices?

Transformação Polinomial

- Sejam P1 o problema da clique e P2 o problema do conjunto independente. Apresente uma transformação polinomial de P1 para P2 (P1 $\propto$ P2).

- Seja uma instância genérica da clique I_{cl} definida por um grafo G(V,A) e um inteiro k > 0.
- Criamos uma instância do conjunto independente I_{ci} considerando o grafo complementar $\overline{G}$ de G e o mesmo inteiro k

- Essa transformação é polinomial pois:
  - O grafo complementar $\overline{G}$ pode ser obtido em tempo polinomial (O(V^2))
  - G possui clique de tamanho >= k **se e somente se** $\overline{G}$ tem conjunto independente de tamanho ≥ k

Transformação Polinomial

- Se existe um algoritmo que resolve o Conjunto Independente em tempo polinomial, ele pode ser usado para resolver clique também em tempo polinomial

- Diz-se que Clique $\propto$ Conjunto Independente

- Note que $\propto$ é transitiva:
  - $P1 \propto P2$ e $P2 \propto P3 \Rightarrow P1 \propto P3$

Classes 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-Completo

  Podemos simplificar propriedade 2 para:

  2' existe $P' \in$ NP-Completo tal que $P' \propto P1$

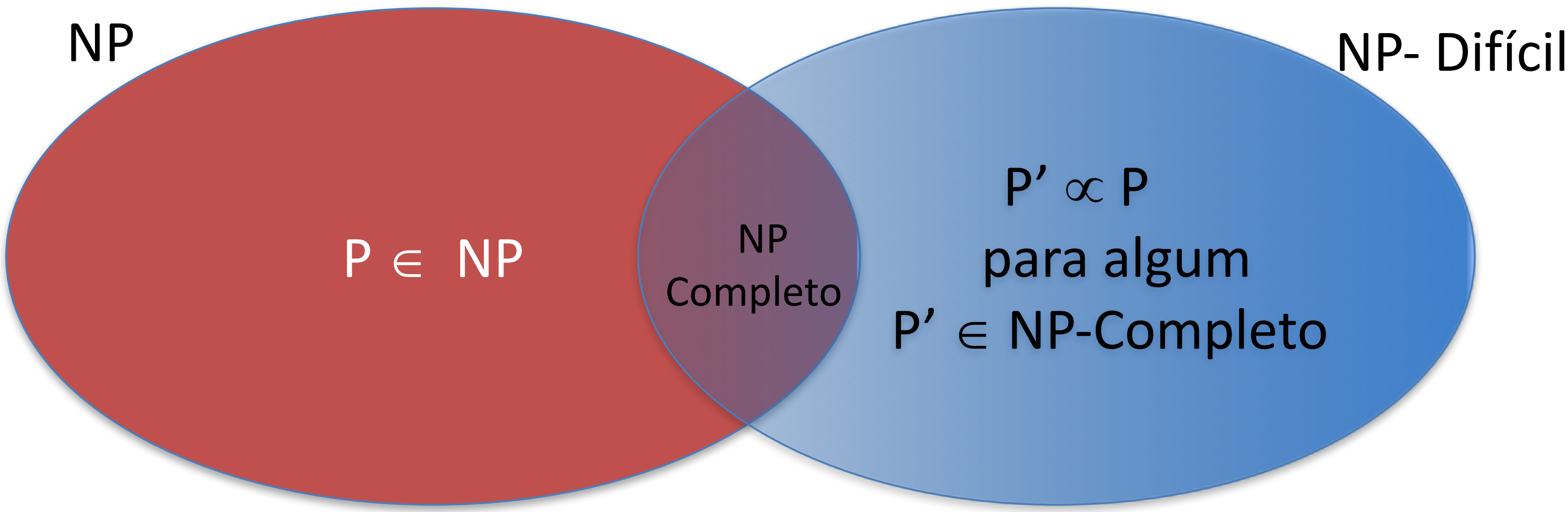
  (Veja prova do Lema 34.8 no livro do Cormem)

- **Um problema P1 é NP-Difícil se somente a propriedade 2 (ou 2') for válida**

Teoremas

- Se qualquer problema NP-Completo puder ser resolvido em tempo polinomial, então P = NP.
- Se qualquer problema em NP não puder ser resolvido em tempo polinomial, então nenhum problema NP-Completo pode ser resolvido em tempo polinomial
- (Teorema 34.4, Cormem)

Classes NP-Completo e NP Difícil



Problema da Parada

- Dado um algoritmo determinista qualquer A com entrada E, A termina ou entra em loop infinito?

- Exemplo de problema que é NP-Difícil mas não é NP-Completo
  – Problema indecidível: não há algoritmo de qualquer complexidade que o resolva

  - Para provar que problema da Parada é NP-Difícil vamos mostrar que SAT $\propto$ Parada:

SAT $\propto$ Parada

- Seja a entrada do SAT um expressão booleana S qualquer.
- Construa uma instância do problema da parada consistindo de um algoritmo A e entrada E como segue:
  - Faça E = S
  - Seja A o algoritmo:
    - Avalie todas as 2^n possibilidades de atribuições de valores booleanas para as n variáveis que compõem S
    - Se S for satisfatível (alguma das possibilidades resulta em S=1), o algoritmo termina (*return*)
    - Senão, ele entra em loop infinito (*while (1);*)

- S é satisfatível se e somente se A pára.

O Primeiro Problema NP-Completo

- CIRCUIT – SAT
  - CIRCUIT - SAT $\in$ NP
  - P $\propto$ CIRCUIT - SAT para todo P $\in$ NP
    - Prova: mostra como obter de um algoritmo polinomial A com entrada E que verifica uma candidata a solução S' um circuito combinatorial C tal que S' é solução se e somente C é satisfatível
    - "simula" execução de A com C
    - Transformação em tempo polinomial

- Teorema de Cook: SAT está em P se e somente se P = NP
- SAT $\in$ NP
- Mostra como obter de qualquer algoritmo polinomial não determinista de decisão A com entrada E uma fórmula Q(A,E) tal que A termina com sucesso para E se e somente se Q é satisfatível

PCV é NP-Completo

1 Mostrar que PCV $\in$ NP

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

PCV é NP-Completo

1 Mostrar que PCV $\in$ NP

a) Mostrar algoritmo polinomial determinista que verifica candidata a solução

```
verificaPCV(V,A,k, S)
{ char visita[|V|]; int sum = 0;
  for (i=0; i < |V| ; i++)  visita[i] = FALSE;
  if (|S| <> |V|) return FALSE;
  visita[S[0]]   = TRUE;
  for (i = 0; i < |S|; i++) {
        if ((A[S[i],S[i+1]] == 0) ||(visita[S[i+1]] == TRUE) )   return FALSE;
        visita[S[i+1]] = TRUE;
        sum += A[S[i],S[i+1]];
  }
  sum += A[S[|V|-1], S[0]];
  if (sum <= k) return TRUE;
  return FALSE;
```

Complexidade: O(|V|)

PCV é NP-Completo

1 Mostrar que PCV $\in$ NP

b) Mostrar algoritmo polinomial não determinista que produz solução

```
resolvePCV(V,A,k)
{
    int i = 0;
    S = {V[i]};
    for (i = 0; i < |V|; i++)
   {
      j = escolhe (V[i], lista-adj(V[i]));
      S  = S ∪  V[j]
    }
    return verificaPCV (V,A,k,S);
}
```

ALTERNATIVA!!!!

Complexidade: O(|V|)

PCV é NP-Completo

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

QUE PROBLEMA ESCOLHER????

PCV é NP-Completo

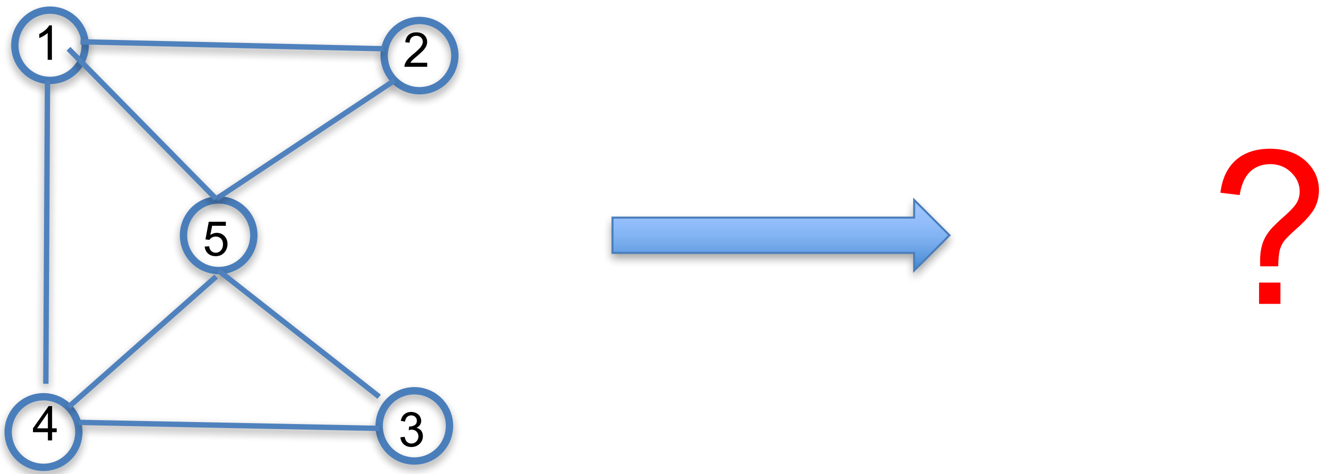
2 Mostrar que existe P $\in$ NP-Completo tal que $P \propto PCV$

P = Ciclo de Hamilton (CH) -> Mostrar que CH $\propto$ PCV

Seja uma instância genérica do CH dada por um grafo G(V,A),

precisamos construir uma instância do PCV dada por um grafo G'(V',A') ponderado completo e um inteiro k.

CH $\propto$ PCV



Exemplo de Instância de CH
G(V,A)

Instância de PCV
G'(V',A')
k

CH $\propto$ PCV

2 Mostrar que CH $\propto$PCV

Seja a instância G(V,A) de CH, construimos uma instância do PCV como segue:

1 Faça V' = V e A' = A

2 Adicione todas as arestas faltantes de forma que G' seja completo.

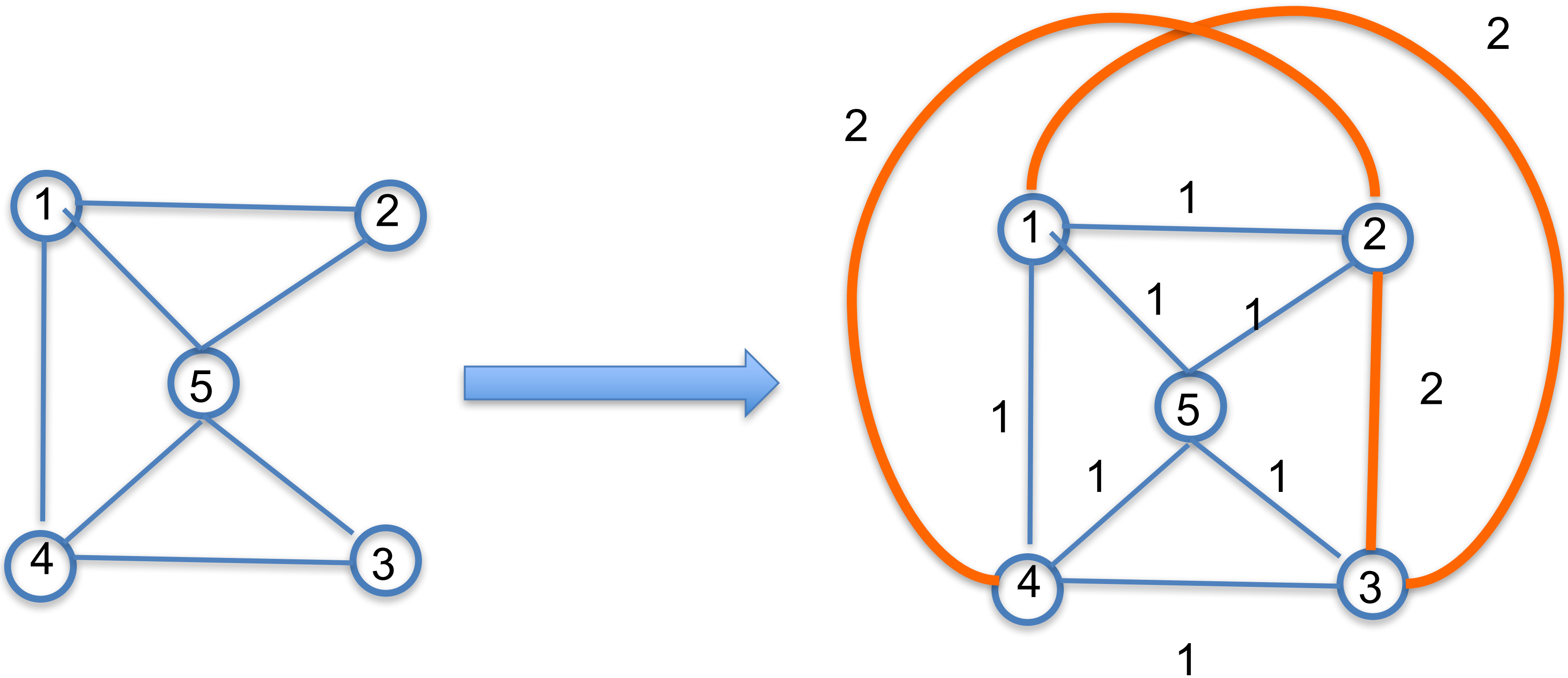
3 Para cada aresta que existia em A, atribua peso 1

4 Para cada nova aresta, atribua peso 2

5 Faça k = |V|

TRANSFORMAÇÃO POLINOMIAL O(V^2)

CH∝ PCV



Exemplo de Instância de CH
G(V,A)

Instância de PCV
G'(V',A')
k = |V| = 5

TRANSFORMAÇÃO POLINOMIAL O(V^2)

PCV é NP-Completo

3 Agora prove que:

a) Se existe solução para CH em $G(V,A)$, existe solução para PCV em $G'(V',A')$ e $k = |V|$

- Seja S a solução para CH em $G(V,A)$
- S contém um ciclo simples com todos os vértices.
- Por construção de G', S existe em G', sendo que todas as arestas do ciclo têm peso 1.
- Ciclo tem $|V|$ vértices, logo tem custo total $|V|$
- Portanto, G' tem ciclo simples com custo total no máximo $k=|V|$

PCV é NP-Completo

3 Agora prove que:

b) Se existe solução para PCV em G’(V’,A’) com k= |V|, existe solução para CH em G(V,A)

- Seja S a solução para PCV em G’(V’,A’) e k
- S contém ciclo simples com |V| vértices e custo total no máximo |V|. Logo o ciclo só pode usar arestas de peso 1.
- As arestas de peso 1 já existiam em G(V,A)
- Logo S existe em G(V,A) e é solução para o CH naquele grafo.

Você tem um mapa de uma região com m vilas e estradas conectando estas vilas e deseja determinar se existe um conjunto de $n < m$ vilas que dominam a região. Em outras palavras, se uma delegacia de polícia fosse implantada em cada um destas n vilas, então todos as outras $m - n$ vilas estariam ligados diretamente a uma das vilas com delegacias.

Este problema é conhecido como o problema do Conjunto Dominante. Prove que ele é NP-Completo

Dica: O Problema da Cobertura de Vértices é NP-Completo. Note que os Problemas da Cobertura de Vértices e do Conjunto Dominante são problemas diferentes

- Provar que Conjunto Dominante (CD) é NP-Completo:
  - $CD \in NP$
  - Existe $\pi \in$ NP-Completo tal que $\pi \propto CD$

- $CD \in NP$

1. Apresentar algoritmo não-determinista polinomial que resolve CD

   OU

2 Apresentar algoritmo determinista polinomial que verifica CD

1. Apresentar algoritmo não-determinista polinomial que resolve CD

```
ResolveND_CD(V,A) {
        S = ∅;
        for i = 1 to |V| do {
          inclui <-   escolhe (V[i], True,False,n)
               if (inclui == TRUE) S = S + V[i]
         }
         if verificaD_CD(V,A,S,n) return sucesso
         else return insucesso
}
```

Este algoritmo é polinomial (explicar)

2 Apresentar algoritmo determinista polinomial que verifica CD

```
verificaD_CD(V,A,S,n) {
    if (|S| > n ) return FALSE
     for i = 1 to |V| do {
          if (V[i] ∉ S) {
                  adj = FALSE
                  for  each vertice j in S   if (V[i], j) ∈ A ) adj = TRUE
              if (adj = FALSE) return FALSE
          }
    }
     return TRUE
}
```

Este algoritmo é polinomial (explicar)

- Existe $\pi \in$ NP-Completo tal que $\pi \propto$ CD

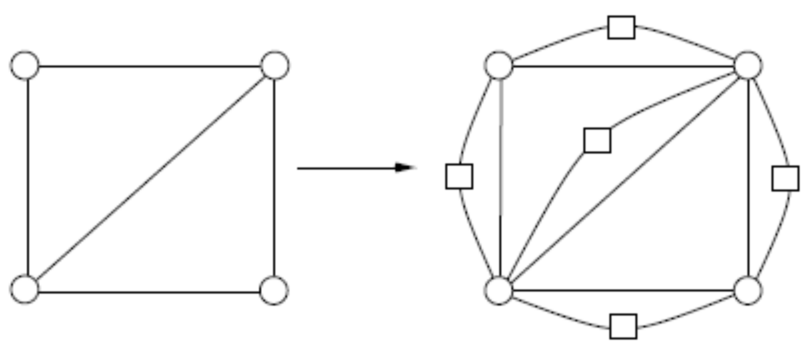
Seja π = Cobertura de Vértices (CV), provar que

$$CV \propto CD$$

- Seja uma instância genérica do CV, dada por um grafo G (V,A) e um valor k
- Precisamos transformá-la, em tempo polinomial em uma instância do CD definida por um grafo G´(V´, A´) e um valor n

  note que a instância criada para o CD pode ser específica (por quê?)

- Faça n = k
- Construa G´ da seguinte maneira:
  - Todo vértice de G também é vértice de G´
  - Toda aresta de G também é aresta de G´
  - Para cada aresta $(i,j) \in A$:
    - cria-se um vértice z_{ij} em G´
    - adicionam-se duas novas arestas (i, z_{ij}) e (z_{ij}, j)
- Esta operação pode ser feita polinomialmente
- EX:



- Agora prove que:
  - Se existe solução do CV em (G,k), existe solução do CD em (G´,n)
  - Se existe solução do CD em (G´,n), existe solução do CV em (G,k)

- Se G tem CV de tamanho ≤ k , G´ tem CD de tamanho ≤ n=k
- Seja S(CV) a solução do CV em G
- Para cada aresta (i,j) $\in$ A qualquer: ou i ou j ou ambos estão em S(CV), dado que a aresta precisa ser coberta por pelo menos 1 vértice
- Suponha que i $\in$ S(CV) (o mesmo vale para j):
  - em G´, o vértice i domina a sub-região (triângulo) definida por (i, j e z_{ij}), dado que j e z_{ij} são vizinhos de i
  - como isto vale para qualquer aresta de A, ela vale para todas também.
- Logo, S(CV) é também solução do CD em G´, n

- Se G´ tem CD de tamanho ≤ n, G tem CV de tamanho ≤ k=n
- Seja S(CD) a solução do CD em G´
- Para cada triângulo <i, j, z_{ij}> $\in$ G´:
  - Ou i, ou j , ou z_{ij} (ou qq combinação deles) $\in$ S(CD)
  - Se i ou j $\in$ S(CD), basta considerá-lo como parte da solução S*
  - Se z_{ij} $\in$ S(CD), então basta substituí-lo por um dos outros dois (i ou j), mantendo assim o número de elementos de S* ≤ n
- Por construção de G´, todas as arestas estão cobertas por S*, que portanto é solução do CV

- E se G tiver nos com grau 0?

A redução anterior assume que G não tem nenhum vértice desconexo. Isto porque:

  – Estes vértices têm que fazer parte da solução do CD mas não do CV

A redução pode ser modificada para considerar estes casos apenas fazendo $n = k+X$ onde X é o número de vértices com grau 0

- Prove que o problema da Cobertura de Vértices (CV) é NP-Completo, partindo da premissa que o problema da Clique (Cq) é NP-Completo.

- CV $\in$ NP

1. Apresentar algoritmo não-determinista polinomial que resolve CV

OU

2 Apresentar algoritmo determinista polinomial que verifica CV

1) Apresentar algoritmo não-determinista polinomial que resolve CV

```
algoritmo resolveCV(V,E) {
 S = conjunto vazio;   //  inicializa conjunto solucao
 for i=1 to |V| do  {
     flag = escolhe(V[i], TRUE, FALSE);  //escolhe se vertice V[i] estará em S
     if (flag == TRUE) S = S +  V[i];
            }
            if |S| > k return insucesso;
      for i=1 to |E| do {
             Seja (u,v) a aresta sendo processada
             Se (u nao pertence a S) e (v nao pertence a S) return false;
       }
       return true;
      }
```

O(|A|) -> polinomial

2) Apresentar algoritmo determinista polinomial que verifica CV

```
algoritmo verificaCV(V,E,S)
{
    if (|S| > k) return false; //solucao tem no maximo k elementos.
      for i=1 to |E| do {
           Seja (u,v) a aresta sendo processada
           Se (u nao pertence a S) e (v nao pertence a S) return false;
     }
     return true;
}
```

O(|A|) -> polinomial

- CV $\in$ NP-Completo:

$$Cq \propto CV$$

- 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 CV definida por um grafo G´(V´, A´) e um valor k’

- Faça G’ = $\overline{G}$ (grafo complementar) e k' = |V| - k.
- Agora prove que:

1) Se G tem Cq de tamanho ≥ k, então G’ tem CV de tamanho ≤ k'= |V| - k.

2) Se $\overline{G}$ tem CV de tamanho ≤ k’ = |V| - k, então G tem Cq de tamanho ≥ k

1) Se G tem Cq de tamanho ≥ k, então G’ tem CV de tamanho ≤ k'= |V| - k.

De fato, seja a solução de Cq dada por S_{Cq}. Quaisquer dois vértices em S_{Cq} têm que ser adjacentes em G. Consequentemente, eles **não** podem ser adjacentes em G’ = $\overline{G}$.

Assim, para toda e qualquer aresta (u,v) em $\overline{G}$, ou u ou v ou ambos estão no subconjunto V - S_{Cq}, já que a aresta (u,v) não existe em G, e logo u e v não podem ambos pertencer à solução da clique.

Este subconjunto é portanto solução do CV em $\overline{G}$.

Como S_{Cq} tem no mínimo k vértices, V - S_{Cq} tem no máximo |V| - k vértices.

2) Se $\overline{G}$ tem CV de tamanho $\leq k' = |V| - k$, então G tem Cq de tamanho $\geq k$

Seja a solução do CV dada por S_{CV}.

Então temos que, para cada par de vértices u, v:

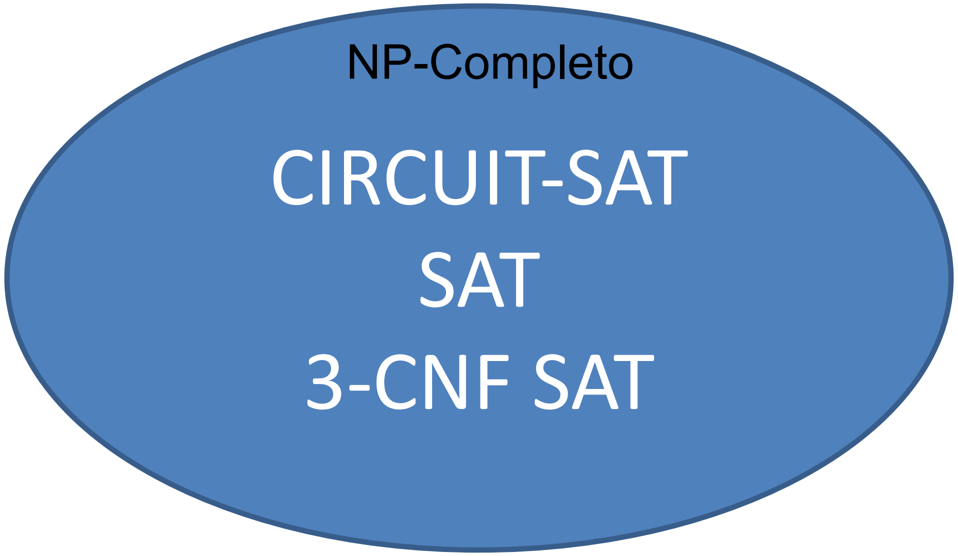
Se a aresta (u,v) existe em $\overline{G}$, então ou u ou v ou ambos pertencem a S_{CV} (a aresta tem que ser coberta)

Se nem u nem v pertencerem a S_{CV}, então (u,v) não existe em $\overline{G}$, e logo existe em G.

Logo, todos vértices que estão em $V - S_{CV}$ são vizinhos entre si em G (formam uma clique)

Como S_{CV} tem no máximo k' vértices, $V - S_{CV}$ tem no mínimo $|V| - k' = k$ vértices

- Prove que o problema da CLIQUE é NP-Completo sabendo que:



- Clique: existe uma clique de tamanho no mínimo k?

Prove que Clique $\in$ NP:

1. Apresentar algoritmo não-determinista polinomial que resolve Clique

OU

2. Apresentar algoritmo determinista polinomial que verifica Clique

```
VerificaD_Clique(V,A, S, k){
    if  (|S| < k) return  FALSE
   for  i=1 to |S|  {
              for j = (i+1) to |S| {
               if (V[i], V[j])∉A  return FALSE;
               }
    } return TRUE
```

Prove que Clique $\in$ NP:

1. Apresentar algoritmo não-determinista polinomial que resolve Clique

OU

2. Apresentar algoritmo determinista polinomial que verifica Clique

```
resolveND_Clique(V,A, k){
  S=∅;
  for  i=1 to |V|  {
          inclui = escolhe(V[i], TRUE, FALSE);
          if (inclui = TRUE) S = S + V[i];
  }
  if verificaD_Clique(V,A, S,k)  return sucesso; else return INSUCESSO;
}
```

- Existe $\pi \in$ NP-Completo tal que $\pi \propto$ Clique

  Seja π = 3CNF SAT, provar que :

  3CNF SAT $\propto$ Clique

- Transformação polinomial ????

- Seja uma instância genérica do 3CNF SAT dada por:

$$S = C_1 * C_2 * \ldots C_n$$

onde $C_i = (l^i_1 + l^i_2 + l^i_3)$

- Uma instância da clique, definida por um grafo G(V,A) e um inteiro k, é construída da seguinte maneira:
  – Faça k = n
  – Construa um grafo G(V,A) da seguinte maneira:
    - Para <u>cada</u> cláusula $C_i = (l^i_1 + l^i_2 + l^i_3)$, adicione 3 vértices correspondentes em V: v^i_1, v^i_2, v^i_3
    - Uma aresta é criada entre vértices v^i_r e v^j_s se :
      – v^i_r e v^j_s correspondem a literais de cláusulas diferentes em S, ou seja $i \neq j$
      – O literal correspondente a v^i_r <u>não</u> é a negação do literal correspondente a v^j_s , ou seja

$$l^i_r \neq \overline{l^j_s}$$

- Mostrar que:
  - Se S = 1, então G tem clique de tamanho $\geq$ k =n
  - Se G tem clique de tamanho $\geq$ k =n, então S = 1
- Se S = 1:
  - Para cada C_i, i=1...n, existe pelo menos um l^i_r , r=1,2,3 tal que l^i_r= 1
  - Seja L o conjunto destes literais l^i_r
  - Seja V´ o conjunto dos vértices v^i_r correspondentes a $l^i_r \in$ L
  - V´ forma uma clique de tamanho k=n no grafo G pois:
    - Se l^i_r = 1 e l^j_s =1 para l^i_r e $l^j_s \in$ L, então, por construção:

      $i \neq j$ e $l^i_r \neq \overline{l}^j_s$
    - Logo $(v^i_r , v^j_s) \in A$
    - Como isto vale para todas as n cláusulas de S

    $\Rightarrow$ Existe Clique de tamanho no mínimo k=n em G

- Se G tem clique V’ de tamanho $\geq k = n$
  - V’ contem 1 vertice para cada clausula de S (por construcao)
  - Faca $l^i_r = 1$ para todo $v^i_r \in V'$
    - Logo $C_i = 1$ para todo $i = 1...n$

    $\Rightarrow S = 1$