PCC104 - Projeto e Análise de Algoritmos

Marco Antonio M. Carvalho

Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

Fonte

Este material é baseado nos livros

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.
- S. Halim. *Competitive Programming*. 3rd Edition, 2013.
- Ian Parberry and William Gasarch. *Problems on Algorithms*. Second Edition, 2002.
- Ian Parberry *Lecture Notes on Algorithm Analysis and Complexity Theory*. Fourth Edition, 2001.

Licença

Classe NP-Completo

Definição Informal

Informalmente, se um problema está na classe NP-Completo, ele está em NP e é "tão difícil" quanto qualquer problema em NP.

Os problemas da classe NPC possuem uma estreita relação entre si, de modo que se um deles for resolvido em tempo polinomial, todos o serão, e consequentemente, P=NP.

Classe NP-Completo

Complexidade

Existe uma forte corrente de pensamento que acredita que os problemas NP-Completos são intratáveis, uma vez que não existe avanço significativo na direção contrária.

Desta forma, P e NP seriam diferentes, mas não é possível concluir nada.

Em certo sentido, os problemas NP-Completos são os mais difíceis em NP.

Uma forma de comparar a dificuldade relativa entre problemas é a redutibilidade em tempo polinomial.

Classe NP-Completo

Definição Formal

Um problema de decisão Q é NP-Completo se:

1. $Q \in$ NP;
2. Q' é polinomialmente redutível a Q para todo $Q' \in$ NP.

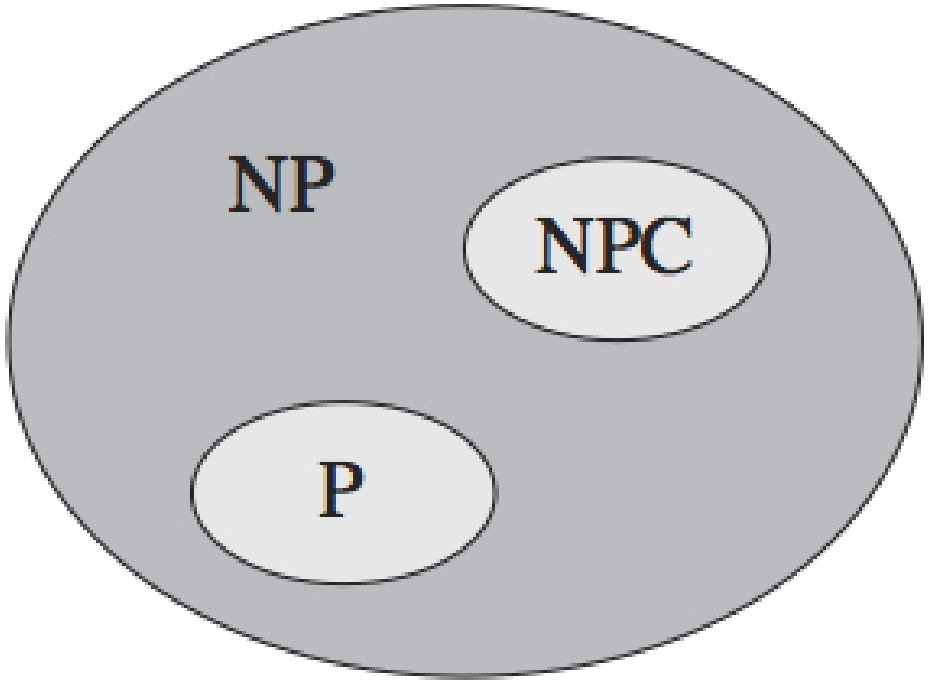
Se é descoberto um algoritmo determinístico polinomial para um problema NP-Completo, ele se torna um problema P, e de acordo com as propriedades acima, todos os outros problemas em NPC também o serão.

As pesquisas sobre P vs. NP se concentram nos problemas NP-Completos por esse motivo.

NP-Difícil

Se um problema satisfaz a propriedade 2, mas não necessariamente a propriedade 1, este pertence à classe NP-Difícil.

Classe NP-Completo



Uma possível relação entre P, NP e NP-Completo. P e NPC seriam contidas em NP, e P$\wedge$NPC=$\emptyset$.

Redutibilidade em Tempo Polinomial

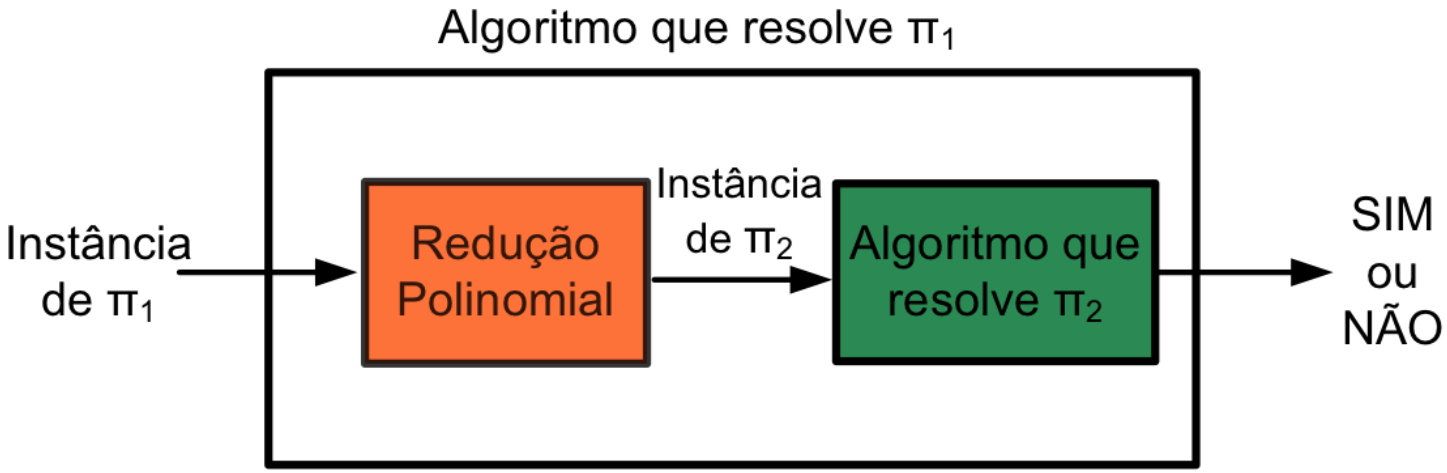
Um problema π_1 pode ser reduzido a um problema π_2 se qualquer instância de π_1 puder ser "facilmente reformulada" como uma instância de π_2: a resposta obtida por π_2 deve ser idêntica à que seria obtida por π_1.

O algoritmo de redução deve ser polinomial.

Se um problema π_1 é redutível a um problema π_2, então π_1 não é mais difícil que π_2.

Com efeito, π_2 é considerado pelo menos tão difícil quanto o π_1, dentro de um fator polinomial.

Redutibilidade em Tempo Polinomial



Redutibilidade em Tempo Polinomial

Por exemplo, o problema de resolver uma equação linear se reduz ao problema de resolver uma equação quadrática:

As instâncias do tipo $ax + b = 0$ são transformadas em $0x^2 + ax + b = 0$.

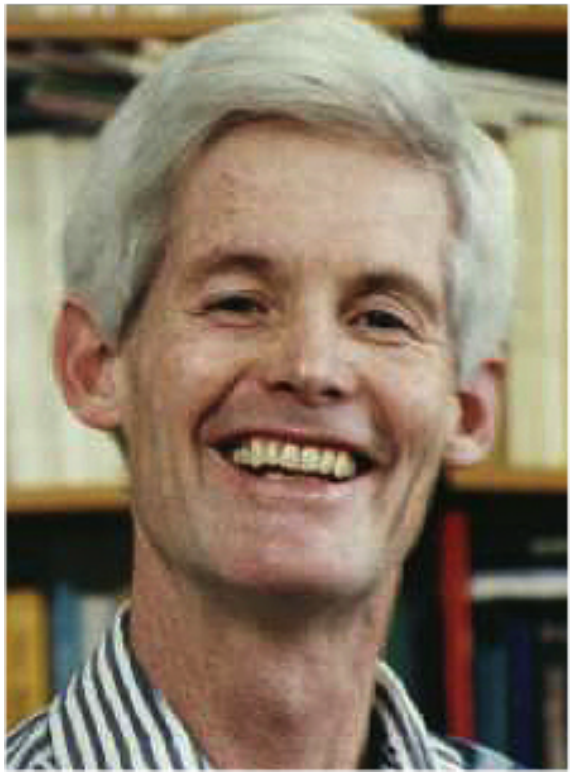
Quando um problema π_1 é polinomialmente reduzível a um problema π_2 denotamos por $\pi_1 \leq_p \pi_2$.

Redutibilidade e Linguagens

Os mesmos conceitos se aplicam à linguagens:

- Uma linguagem L_1 pode ser transformada em uma linguagem L_2;
- A redução deve ser computada por uma máquina de Turing polinomial;
- Cadeias de símbolos só pertencem a L_1 se pertencerem a L_2;
- Consequentemente L_1 e L_2 pertencem à mesma classe de complexidade.

Teorema de Cook



Stephen Arthur Cook

- Matemático;
- Cientista da Computação;
- Professor da Universidade de Toronto;
- **Pai da Teoria da Complexidade Computacional**
  - Formalizou a noção de redução em tempo polinomial;
  - Formalizou o conceito de NP-Completo;
  - Identificou o primeiro problema NP-Completo;
  - Autor do **Teorema de Cook**.
- Pelo teorema, recebeu o Prêmio Turing em 1982.

Teorema de Cook

Definição

A definição de problemas NP-Completos é de certa forma "recursiva". Então qual é o caso base? Qual é o problema NP-Completo original?

SAT

O Problema de Satisfabilidade Booleana (SAT) foi o primeiro problema caracterizado como NP-Completo.

Dada uma expressão lógica com n variáveis booleanas e m conectivos lógicos NOT ($\neg$), AND ($\wedge$) e OR ($\vee$), é necessário determinar se há uma atribuição satisfatória de valores às variáveis, ou seja, que resulte em valor 1 (ou verdadeiro).

O **Teorema de Cook** nos diz que este problema de decisão é NP-Completo.

Teorema de Cook

Histórico

O Teorema de Cook (1971) também é conhecido como Teorema de Cook-Levin, por também ser atribuído independentemente ao russo/americano Leonid Levin.

Levin publicou em 1973 um artigo que considerava problemas de busca, provando haver 6 **problemas universais** (equivalentes aos NP-Completos), embora existam menções em anos anteriores a este trabalho.

O Teorema de Cook, descrito em um artigo de pouco mais do que 7 páginas, define o primeiro problema NP-Completo: o problema de satisfabilidade booleana.

Teorema de Cook

Definição

Resumidamente, Cook definiu uma linguagem L_{SAT} e também uma linguagem geral de problemas NP, reconhecida por uma Máquina de Turing não determinística polinomial genérica.

Posteriormente, foi provado que todas as linguagens L de problemas NP se reduzem a L_{SAT}, logo, a Máquina de Turing não determinística polinomial reconhece L_{SAT}.

Satisfazendo-se as propriedades exigidas, provou que SAT é o primeiro problema NP-Completo.

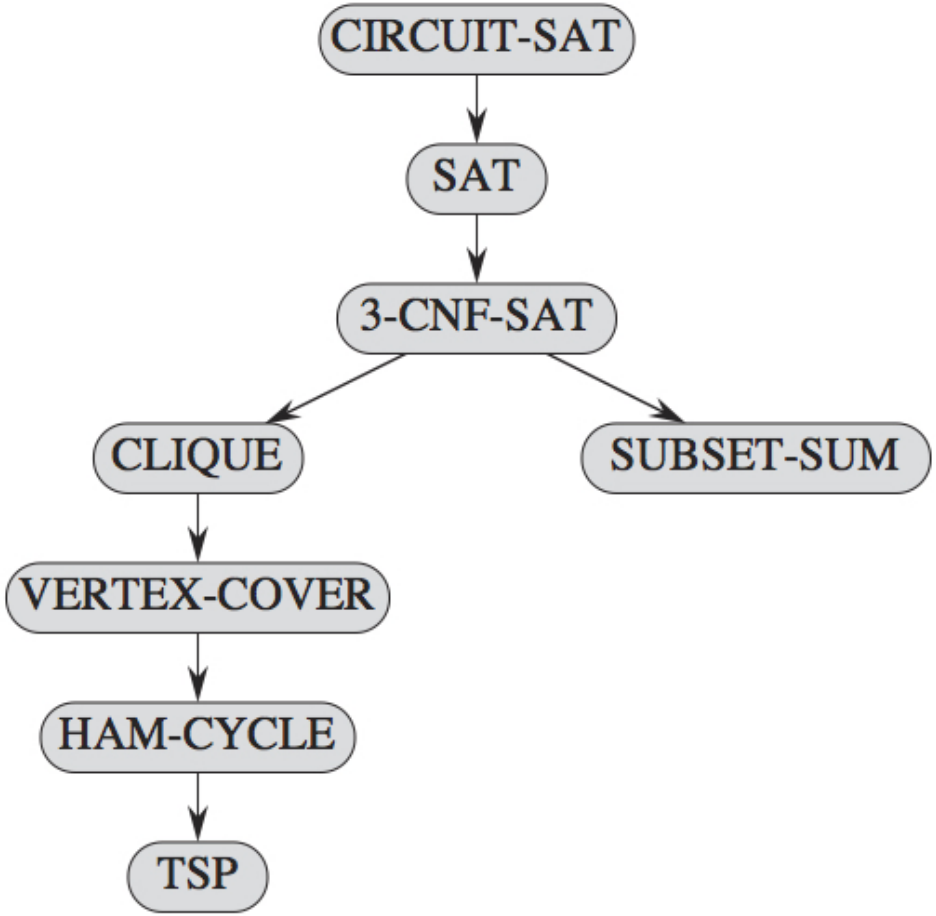
Problemas Intratáveis

NP-Completo

Em 1971, Richard Karp identificou os primeiros 21 problemas da classe e contribuiu para o desenvolvimento da teoria da NP-Completude.

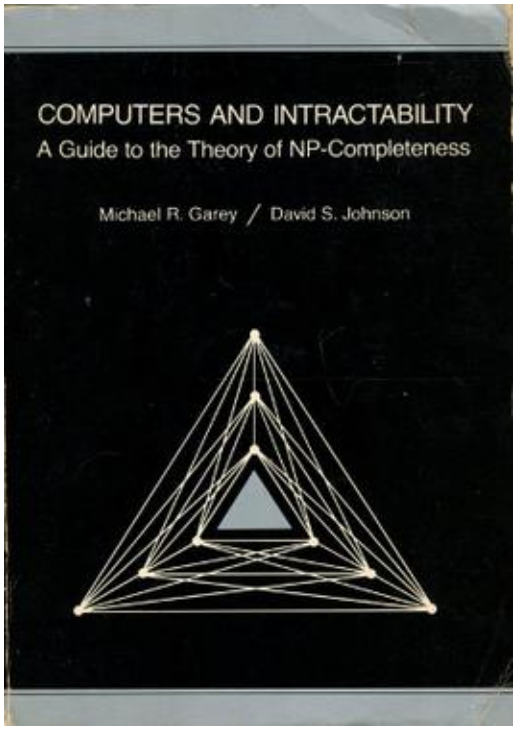
Posteriormente, centenas de outros problemas foram identificados por outros pesquisadores.

Teorema de Cook



Esquema da prova dos problemas NP-Completos.

Problemas Intratáveis



O livro da capa preta

Michael R. Garey and David S. Johnson. 1979. Computers and Intractability: A Guide to the Theory of Np-Completeness. W. H. Freeman & Co., New York, NY, USA.

Foi o primeiro livro a tratar exclusivamente de NP-completude e intratabilidade computacional. Apresenta um apêndice fornecendo um compêndio exaustivo dos problemas NP-completos.

Considerado como um clássico: em um estudo de 2006, o CiteSeer listou o livro como a referência mais citada na literatura de ciência da computação.

3-SAT$\leq_p$Clique

3-SAT

O 3-SAT é um caso especial do SAT, em que cada cláusula contém exatamente 3 literais:

Adicionalmente, a expressão está na Forma Normal Conjuntiva (CNF): grupamentos AND de cláusulas, cada uma sendo um OR de um ou mais literais.

$$E = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

O 3-SAT é um problema NP-Completo, uma vez que SAT$\leq_p$3-SAT.

3-SAT$\leq_p$Clique

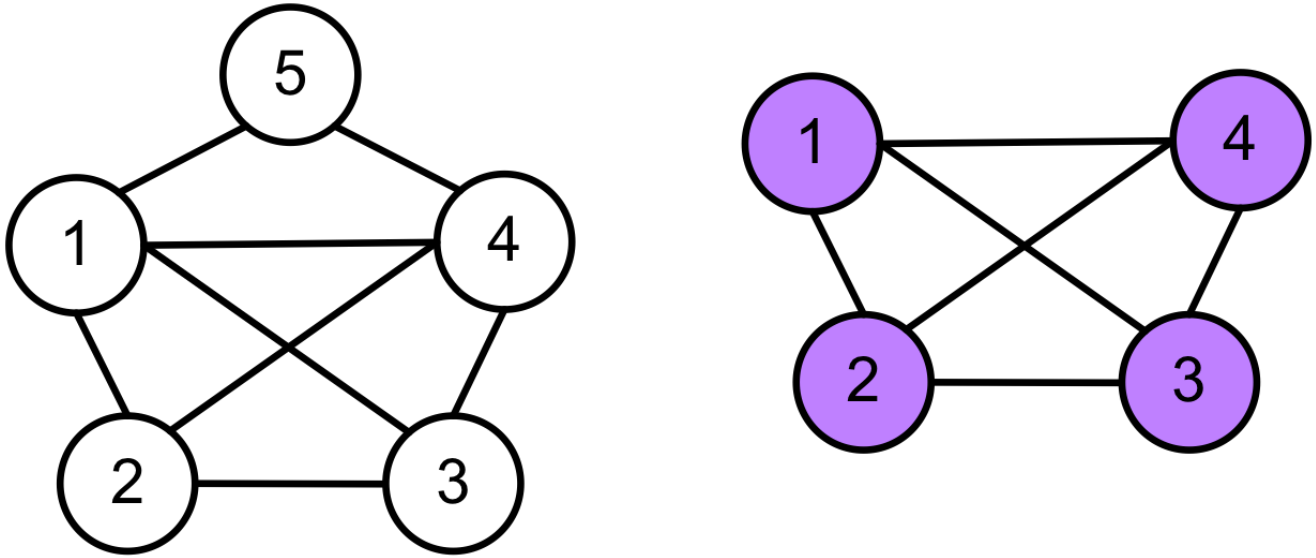
Clique

Um Clique em um grafo não orientado G=(V, E) é um subgrafo completo, ou seja, cada vértice está conectado a todos os demais.

O problema de Clique Máximo consiste em detectar um clique de tamanho máximo em G.

A versão de decisão deste problema pergunta se existe um clique de tamanho k no grafo G.

3-SAT$\leq_p$Clique



Grafo G e um clique de tamanho $k = 4$ no mesmo grafo.

Verificação da Propriedade 1

Dado um grafo G=(V, E), e um conjunto V' de vértices da solução, verificamos se cada par de vértices u, v de V' também pertence a V, e se a aresta $\{u, v\}$ pertence a E.

Esta verificação pode ser feita em tempo polinomial, e se de fato todos os vértices e arestas existirem, a solução é válida.

Portanto, Clique $\in$ NP.

Verificação da Propriedade 2

Como vimos anteriormente, uma instância 3-SAT é uma expressão booleana ϕ em que cada cláusula possui exatamente 3 literais distintos.

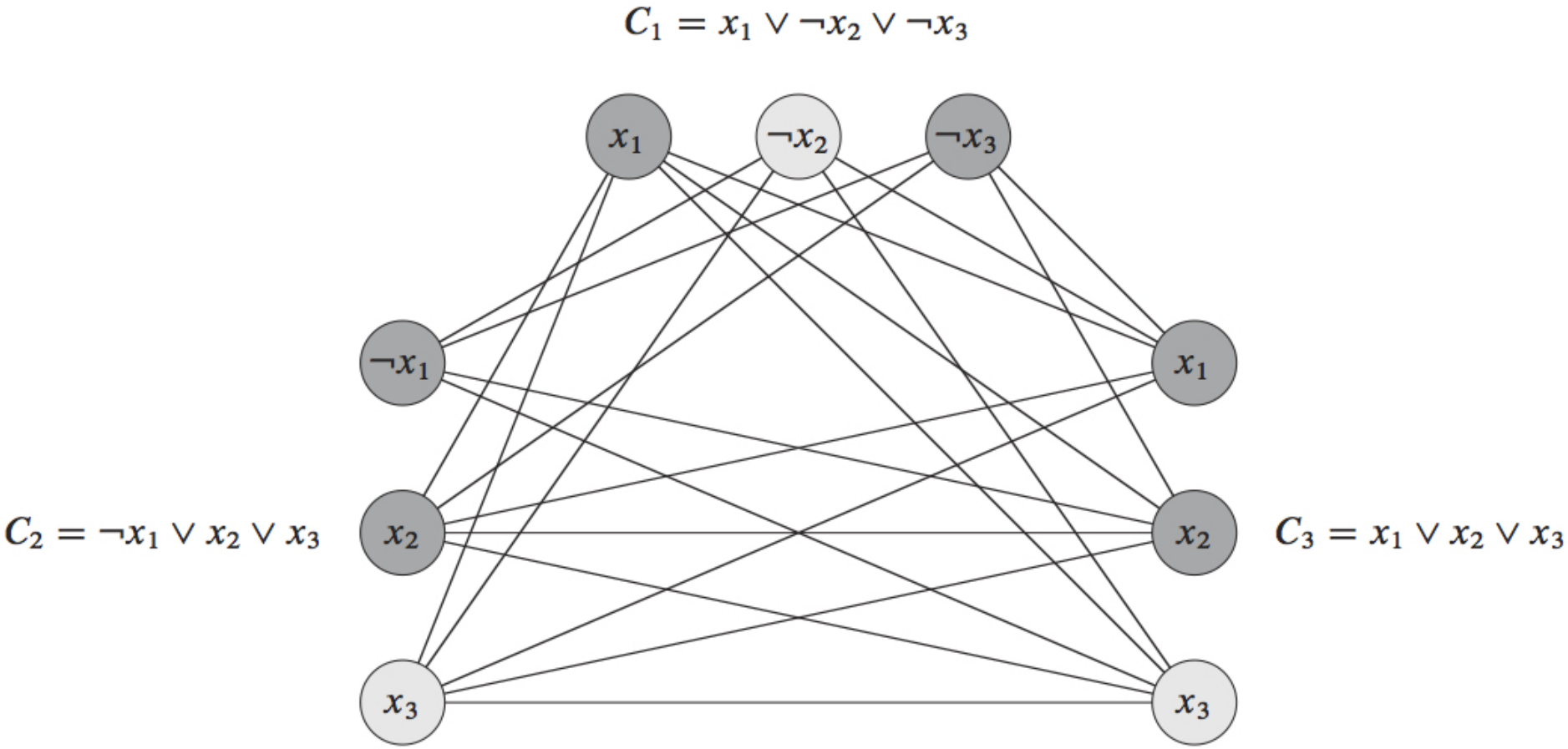
Para a redução, construímos um grafo G=(V, E), tal que para cada cláusula C_r =$\{l_1^r \cup l_2^r \cup l_3^r\}$ em ϕ inserimos uma tripla de vértices v_1^r, v_2^r e v_3^r em V.

Arestas são inseridas entre dois vértices v_i^r e v_j^s de acordo com os seguintes critérios:

- v_i^r e v_j^s estão em diferentes triplas (r$\neq$s);
- Seus literais correspondentes são coerentes ($i.\ e.$, l_i^r não é a negação de l_j^s).

O grafo pode ser construído em tempo polinomial.

3-SAT$\leq_p$Clique



Exemplo de construção do grafo.

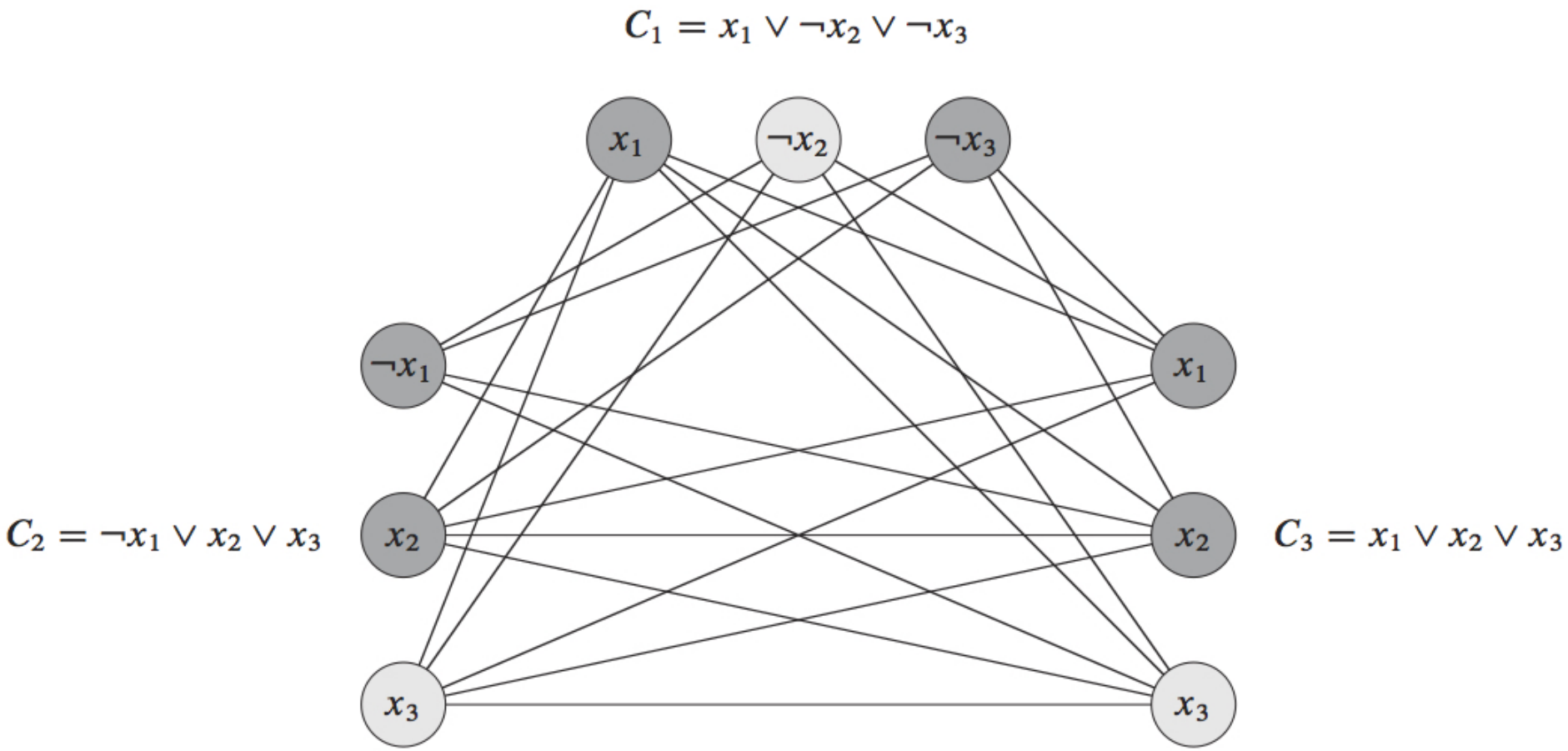
Verificação da Propriedade 2

Suponha que ϕ possui uma atribuição satisfatória: toda cláusula deve ter pelo menos um literal com valor 1 para que a atribuição seja satisfatória.

Podemos escolher o vértice "verdadeiro" de cada cláusula/tripla, pois eles serão adjacentes entre si – um literal e sua negação não são adjacentes.

Formarão um Clique, portanto.

3-SAT$\leq_p$Clique



Uma atribuição que satisfaz a fórmula é $x_2 = 0$, $x_3 = 1$ e x_1 pode ser 0 ou 1. O clique de tamanho 3 corresponde às variáveis principais.

Verificação da Propriedade 2

3-SAT e Clique obtêm as mesmas respostas para a mesma instância: se ϕ não possuir uma atribuição satisfatória, não haverá clique em G.

O caso do grafo apresentado é restrito, dada a sua estrutura em triplas, mas é suficiente.

Se houver um algoritmo de tempo polinomial que resolva Clique em grafos gerais, ele também resolverá em grafos restritos.

Como o algoritmo é polinomial, Clique $\in$ NP-Completo.

Clique$\leq_p$Cobertura de Vértices

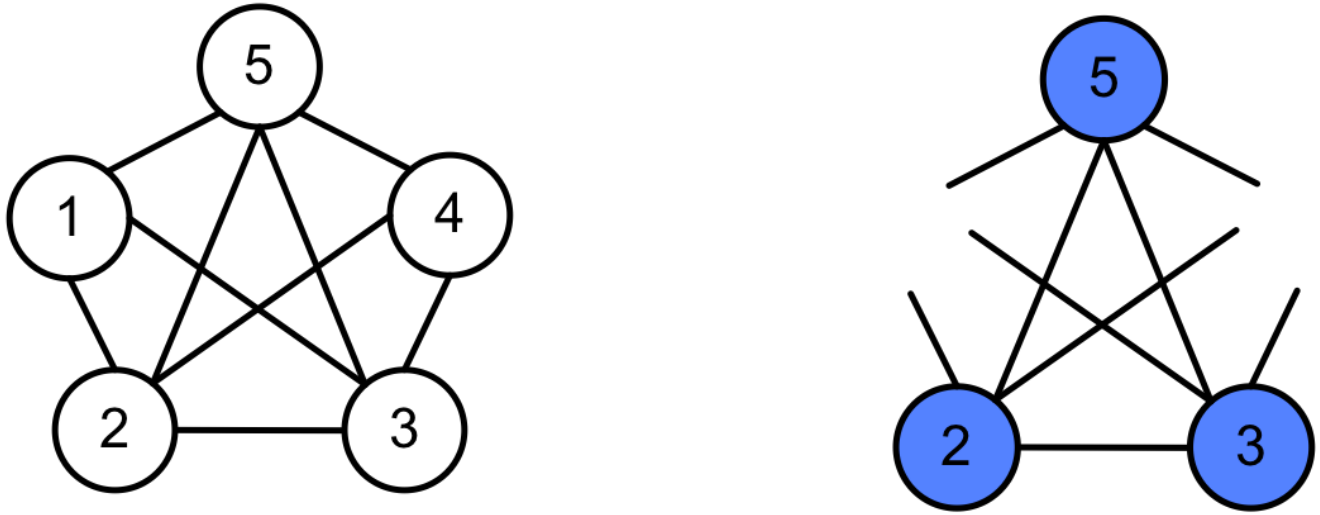
Cobertura de Vértices

Uma Cobertura de Vértices (ou Conjunto Dominante) em um grafo não orientado G=(V, E) é um subconjunto V' de V tal que para cada aresta $\{u, v\}$ do grafo G, u ou v pertencem a V'.

O problema de Cobertura de Vértices Mínima (ou Conjunto Dominante Mínimo) consiste em encontrar a cobertura de vértices de um grafo que requer o menor número de vértices.

A versão de decisão do problema de cobertura de vértices pergunta se existe em um grafo G uma cobertura de vértices (ou conjunto dominante) de tamanho k.

Clique$\leq_p$Cobertura de Vértices



Grafo de exemplo e uma possível cobertura de vértices (não mínima).

Verificação da Propriedade 1

Para um grafo G=(V, E), uma solução consiste em um conjunto V' de vértices e um inteiro k.

Inicialmente, conferimos se $|V'| = k$.

Posteriormente, conferimos para cada aresta $\{u, v\}$ de G, se u ou v pertencem a V'.

Caso todos os testes sejam satisfeitos, temos uma solução válida.

A verificação pode ser realizada em tempo polinomial.

Portanto, Cobertura de Vértices $\in$ NP.

Verificação da Propriedade 2

A redução se baseia na noção de grafo complementar.

Para um grafo simples G=(V, E), o seu grafo complementar é $\overline{G}$=(V, $\overline{E}$), em que as arestas de $\overline{E}$ são todas as que não pertencem a E.

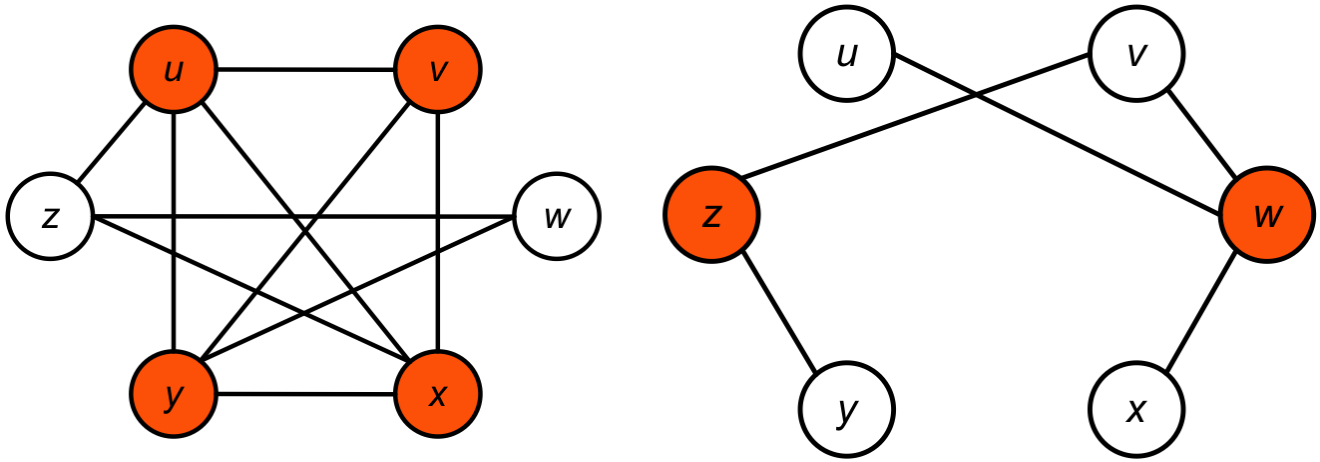
Tomamos como entrada o grafo G e a cardinalidade do clique k.

Calculamos o complemento $\overline{G}$.

O resultado é o grafo $\overline{G}$ e a cardinalidade da cobertura $|V| - k$.

O grafo G tem um clique de tamanho k se e somente se o grafo $\overline{G}$ tem uma cobertura de vértices de tamanho $|V| - k$.

Clique$\leq_p$Cobertura de Vértices



Clique V' em G e cobertura de vértices $V - V'$ em $\overline{G}$.

Verificação da Propriedade 2

- Todos os pares de vértices em V' são adjacentes, pois trata-se de um clique;
- Uma aresta qualquer $\{u, v\}$ pertencente a $\overline{E}$ não pertence a E, logo, u ou v não pertencem a V'.
- Equivalentemente, pelo menos um entre u e v está em $V - V'$
  - Portanto, a aresta $\{u, v\}$ está coberta por V-V'.
- Todas as arestas de E são cobertas por $V - V'$, implicando em uma cobertura de tamanho $|V| - k$ em $\overline{G}$.

Logo, Clique e Cobertura de Vértices obtêm as mesmas respostas para a mesma instância, e desta forma, Cobertura de Vértices $\in$ NP-Completo.

Ciclo Hamiltoniano$\leq_p$Caixeiro Viajante

Ciclo Hamiltoniano

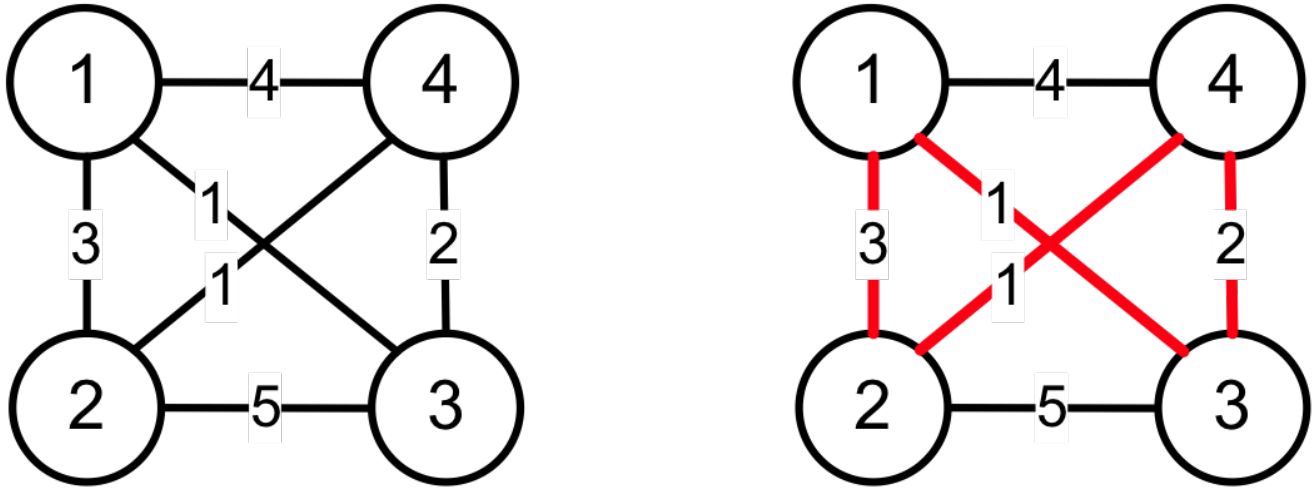
Um ciclo hamiltoniano em um grafo não ponderado G=(V, E) consiste em decidir se existe um caminho fechado, ou seja, uma sequência de visitação de vértices e arestas tal que todos os vértices são visitados uma única vez e que ao final retorne ao vértice inicial.

Problema do Caixeiro Viajante

O problema do Caixeiro Viajante (PCV) em um grafo ponderado completo G=(V, E) consiste em encontrar um ciclo Hamiltoniano em G tal que a soma dos pesos das arestas utilizadas seja minimizada.

A versão de decisão deste problema pergunta se existe um ciclo Hamiltoniano em G cuja soma dos pesos das arestas utilizadas seja k.

Ciclo Hamiltoniano$\leq_p$Caixeiro Viajante



Exemplo de instância do Problema do Caixeiro Viajante e solução correspondente.

Ciclo Hamiltoniano$\leq_p$Caixeiro Viajante

Verificação da Propriedade 1

A solução para o problema é uma sequência de $|V|$ vértices.

A verificação determina se cada vértice foi visitado exatamente um vez e se a totalização dos pesos das arestas utilizadas é no máximo k.

A verificação pode ser realizada em tempo polinomial, portanto, Caixeiro Viajante $\in$ NP.

Verificação da Propriedade 2

Um grafo G=(V, E) é uma instância para o ciclo Hamiltoniano.

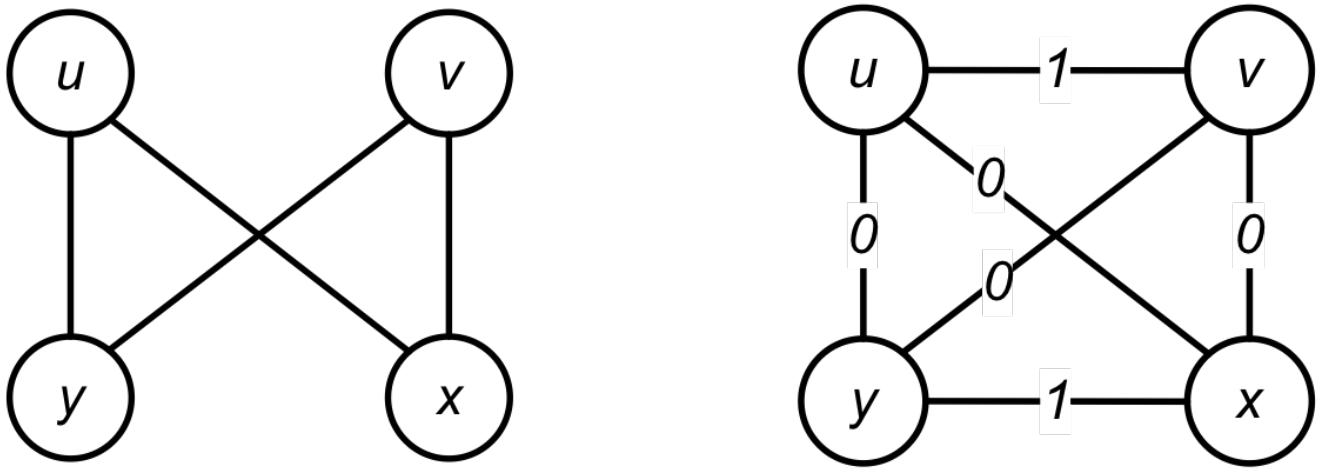
A partir dele construímos uma instância para o Caixeiro Viajante como a seguir.

Seja G'=(V', E') um grafo completo, definimos a função de custo como:

- $c(i,j)$=0, se $\{i, j\} \in E$;
- $c(i,j)$=1, se $\{i, j\} \ni E$.

Como G' não é orientado e não possui laços, $c(i, i) = 1$.

Ciclo Hamiltoniano$\leq_p$Caixeiro Viajante



Grafo G original e grafo ponderado G' construído a partir de G.

Verificação da Propriedade 2

A construção do grafo G' pode ser feita em tempo polinomial.

O grafo G possui um ciclo Hamiltoniano se e somente se G' possui uma viagem do caixeiro viajante com custo 0.

Suponha que G tem um ciclo Hamiltoniano, então todas as arestas do ciclo existem em G' e possuem custo 0, formando uma viagem em G' com custo 0.

Reciprocamente, se G' possui uma viagem com custo zero, as respectivas arestas existem em G, formando um ciclo.

Ciclo Hamiltoniano e Caixeiro Viajante obtêm as mesmas respostas para a mesma instância.

Logo, o Problema do Caixeiro Viajante $\in$ NP-Completo.

Dúvidas?

3-SAT

Definição

O 3-SAT é um caso especial do SAT, em que cada cláusula contém exatamente 3 literais.

Adicionalmente, a expressão está na Forma Normal Conjuntiva (CNF): grupamentos AND de cláusulas, cada uma sendo um OR de um ou mais literais.

$$E = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_4)$$

O 3-SAT é um problema NP-Completo, uma vez que SAT$\leq_p$3-SAT.

SAT$\leq_p$3-SAT

Verificação da Propriedade 1

A solução consiste no conjunto de valores a serem atribuídos às variáveis.

Para verificar uma solução para o 3-SAT, simplesmente substituímos as variáveis pelos valores dados na solução e avaliamos o resultado, o que pode ser feito em tempo polinomial.

Se a atribuição é satisfatória, a solução é válida, portanto, 3-SAT$\in$NP.

Verificação da Propriedade 2

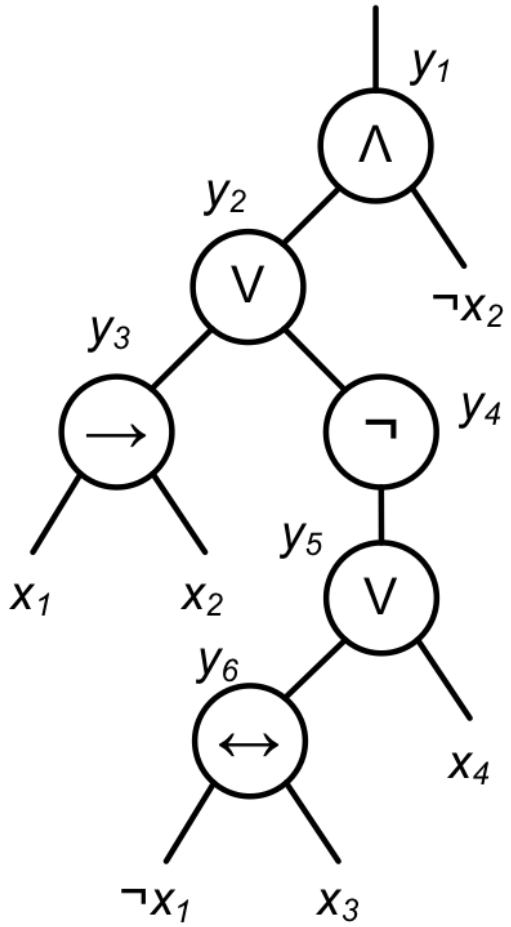
A redução é feita em três etapas: progressivamente, cada passo transforma a entrada ϕ, deixando-a próxima da 3-CNF.

O **primeiro passo** consiste em construir a árvore binária de análise para a expressão ϕ (a associatividade pode ser utilizada para garantir que cada nó terá apenas dois filhos).

Os nós internos são os conectivos lógicos e as folhas são os literais.

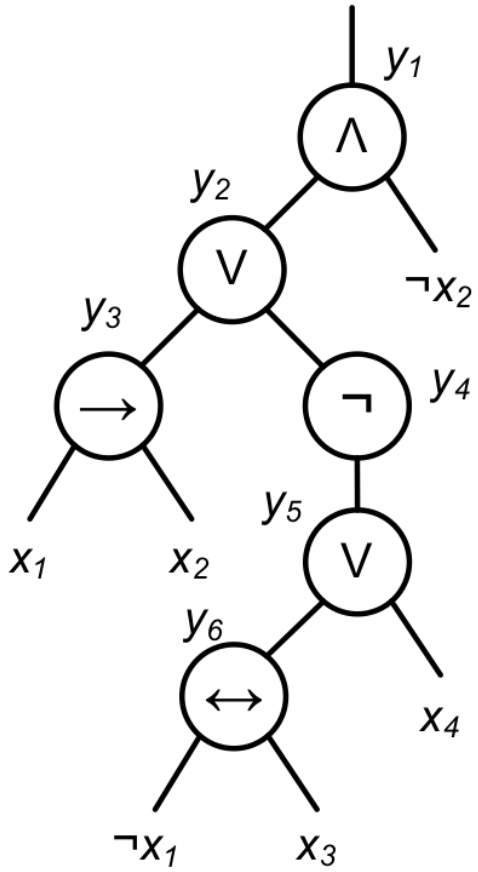
Adicionamos uma variável y_i para a saída de cada nó interno.

SAT$\leq_p$3-SAT



$$\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

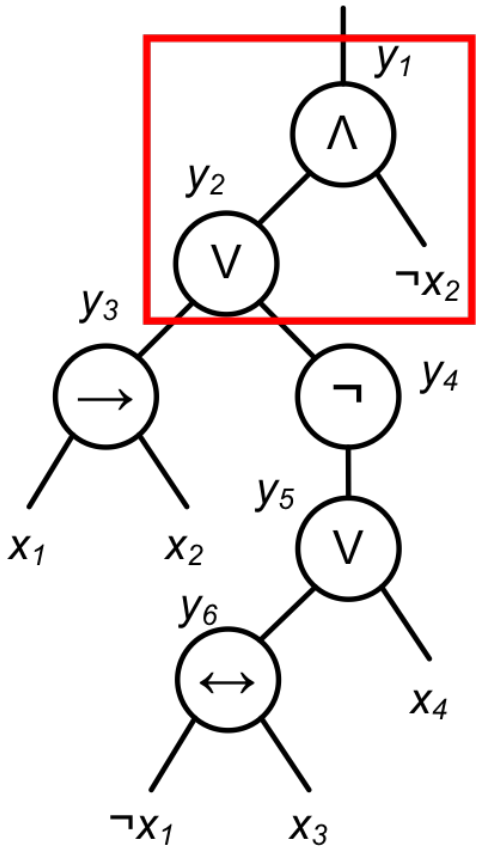
SAT$\leq_p$3-SAT



Descrição

Reescrevemos a fórmula original ϕ como um AND da variável de raiz e uma conjunção de cláusulas descrevendo a operação de cada nó.

SAT$\leq_p$3-SAT

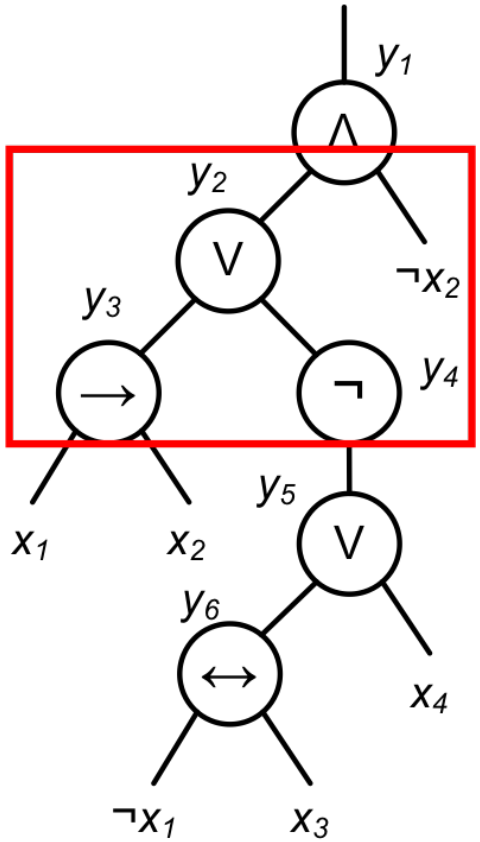


Descrição

Reescrevemos a fórmula original ϕ como um AND da variável de raiz e uma conjunção de cláusulas descrevendo a operação de cada nó.

$$\phi^1 = y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2))$$

SAT$\leq_p$3-SAT

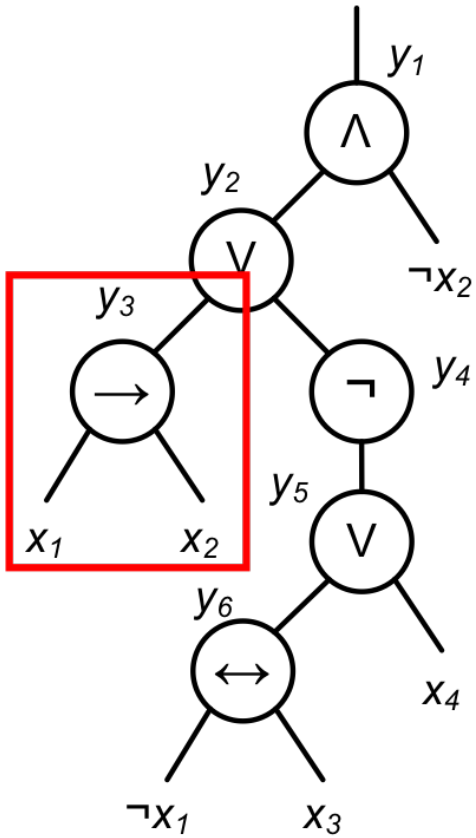


Descrição

Reescrevemos a fórmula original ϕ como um AND da variável de raiz e uma conjunção de cláusulas descrevendo a operação de cada nó.

$$\phi^1 = y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \wedge (y_2 \leftrightarrow y_3 \vee y_4)$$

SAT$\leq_p$3-SAT

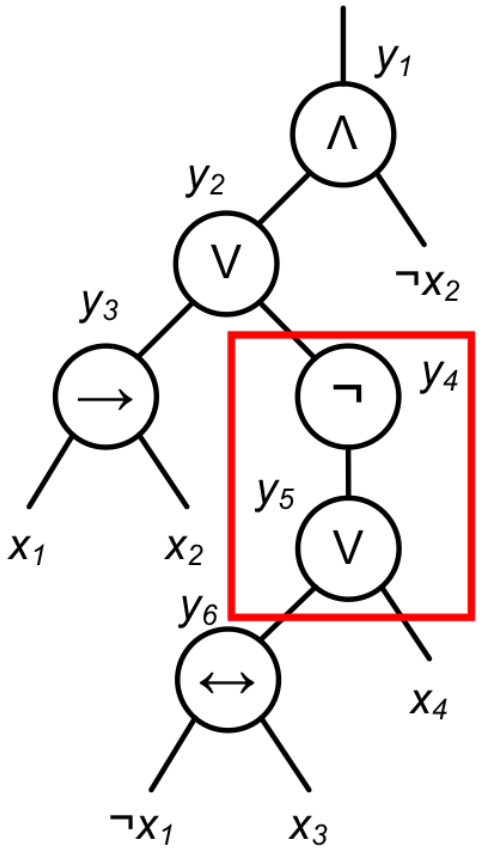


Descrição

Reescrevemos a fórmula original ϕ como um AND da variável de raiz e uma conjunção de cláusulas descrevendo a operação de cada nó.

$$\begin{aligned}\phi^1 = y_1 &\wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ &\wedge (y_2 \leftrightarrow y_3 \vee y_4) \\ &\wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2))\end{aligned}$$

SAT$\leq_p$3-SAT

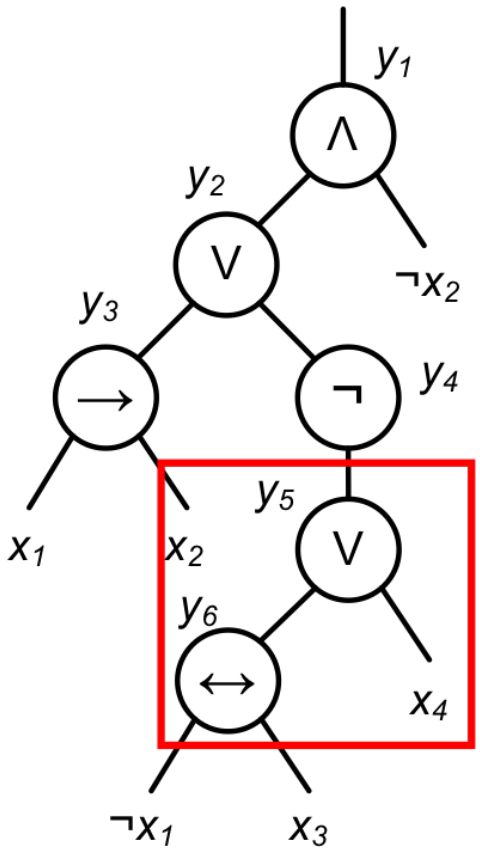


Descrição

Reescrevemos a fórmula original ϕ como um AND da variável de raiz e uma conjunção de cláusulas descrevendo a operação de cada nó.

$$\begin{aligned}\phi^1 = y_1 &\wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ &\wedge (y_2 \leftrightarrow y_3 \vee y_4) \\ &\wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\ &\wedge (y_4 \leftrightarrow \neg y_5)\end{aligned}$$

SAT$\leq_p$3-SAT

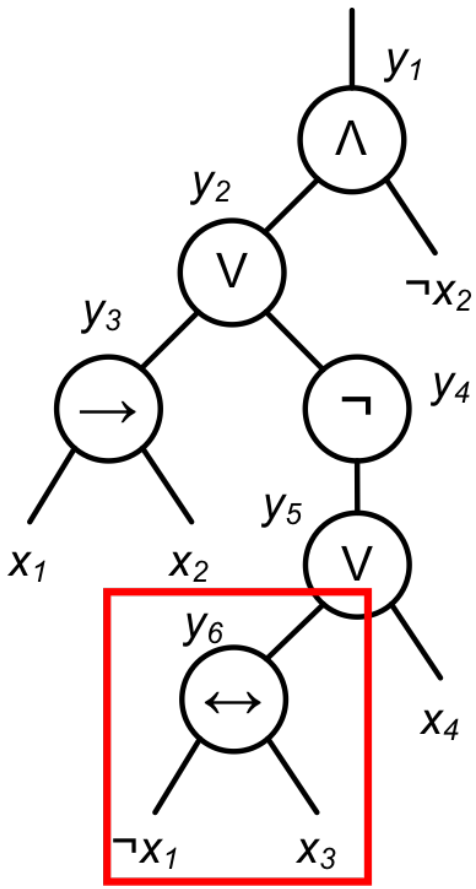


Descrição

Reescrevemos a fórmula original ϕ como um AND da variável de raiz e uma conjunção de cláusulas descrevendo a operação de cada nó.

$$\begin{aligned}\phi^1 = y_1 &\wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ &\wedge (y_2 \leftrightarrow y_3 \vee y_4) \\ &\wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\ &\wedge (y_4 \leftrightarrow \neg y_5) \\ &\wedge (y_5 \leftrightarrow (y_6 \vee x_4))\end{aligned}$$

SAT$\leq_p$3-SAT



Descrição

Reescrevemos a fórmula original ϕ como um AND da variável de raiz e uma conjunção de cláusulas descrevendo a operação de cada nó.

$$\begin{aligned}\phi^1 = y_1 &\wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ &\wedge (y_2 \leftrightarrow y_3 \vee y_4) \\ &\wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\ &\wedge (y_4 \leftrightarrow \neg y_5) \\ &\wedge (y_5 \leftrightarrow (y_6 \vee x_4)) \\ &\wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3))\end{aligned}$$

Verificação da Propriedade 2

A expressão ϕ^1 obtida é uma conjunção de cláusulas ϕ_i^1 cada qual com 3 literais.

O único requisito adicional é que cada cláusula seja um OR de literais.

Verificação da Propriedade 2

O **segundo passo** consiste em converter ϕ^1 para a forma normal conjuntiva.

Constrói-se a tabela verdade para cada cláusula, avaliando-se cada combinação de atribuições possível.

Usando as entradas da tabela avaliadas como 0, construímos uma expressão na **forma normal disjuntiva** – um OR de ANDs, equivalente a $\neg\phi_i^1$.

SAT$\leq_p$3-SAT

y_1	y_2	x_2	$(y_1 \leftrightarrow (y_2 \wedge \neg x_2)$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

$$(y_1 \wedge y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge \neg x_2) \vee (\neg y_1 \wedge y_2 \wedge \neg x_2)$$

Tabela verdade para uma das cláusulas do exemplo.

Verificação da Propriedade 2

Aplicando as leis de DeMorgan, obtemos a expressão na fórmula normal conjuntiva, que é equivalente a ϕ_i^1 original:

$$(\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee \neg y_2 \vee x_2)$$

Convertemos cada cláusula ϕ_i^1 de ϕ^1 para a CNF, gerando ϕ^2.

Verificação da Propriedade 2

O **terceiro passo** certifica que cada cláusula possui exatamente 3 literais distintos.

A partir de ϕ^2 criamos ϕ^3 utilizando variáveis auxiliares p e q.

Para cada cláusula C_i de ϕ^2, incluímos as seguintes cláusulas em ϕ^3:

- Se C_i tem 3 literais distintos, inclua como está;
- Se C_i tem dois literais $(l_1 \vee l_2)$, inclua $(l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$;
- Se C_i tem apenas um literal (l_1), inclua $(l_1 \vee p \vee q) \wedge (l_1 \vee p \vee \neg q) \wedge (l_1 \vee \neg p \vee q) \wedge (l_1 \vee \neg p \vee \neg q)$.

Verificação da Propriedade 2

O número de variáveis e cláusulas adicionadas é polinomial, o tamanho de ϕ^3 é polinomial no tamanho de ϕ e cada um dos passos pode ser realizado em tempo polinomial.

As expressões ϕ^1, ϕ^2 e ϕ^3 são equivalentes entre si.

SAT e 3-SAT obtêm as mesmas respostas para uma instância.

Logo, 3-SAT é um problema NP-Completo.

Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Cobertura de Vértices

Uma Cobertura de Vértices (ou Conjunto Dominante) em um grafo não orientado G=(V, E) é um subconjunto V' de V tal que para cada aresta $\{u, v\}$ do grafo G, u ou v pertencem a V'.

A versão de decisão do problema de cobertura de vértices pergunta se existe em um grafo G uma cobertura de vértices (ou conjunto dominante) de tamanho k.

Ciclo Hamiltoniano

Um ciclo hamiltoniano em um grafo não ponderado G=(V, E) consiste em decidir se existe um caminho fechado, ou seja, uma sequência de visitação de vértices e arestas tal que todos os vértices são visitados uma única vez e que ao final retorne ao vértice inicial.

Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Verificação da Propriedade 1

A solução para o problema é a sequência de $|V|$ vértices.

A verificação determina se de fato existe uma aresta entre cada par de vértices consecutivos e entre o vértice final e o inicial.

Caso todas as adjacências existam, a solução é válida.

A verificação pode ser realizada em tempo polinomial, portanto, Ciclo Hamiltoniano $\in$ NP.

Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Verificação da Propriedade 2

A partir de um grafo G=(V, E) e um inteiro k, construiremos um grafo G'=(V', E') que contem um ciclo Hamiltoniano somente se G possuir uma cobertura de vértices de tamanho k.

Para isso, usaremos um dispositivo que é um fragmento de grafo que impõe algumas propriedades.

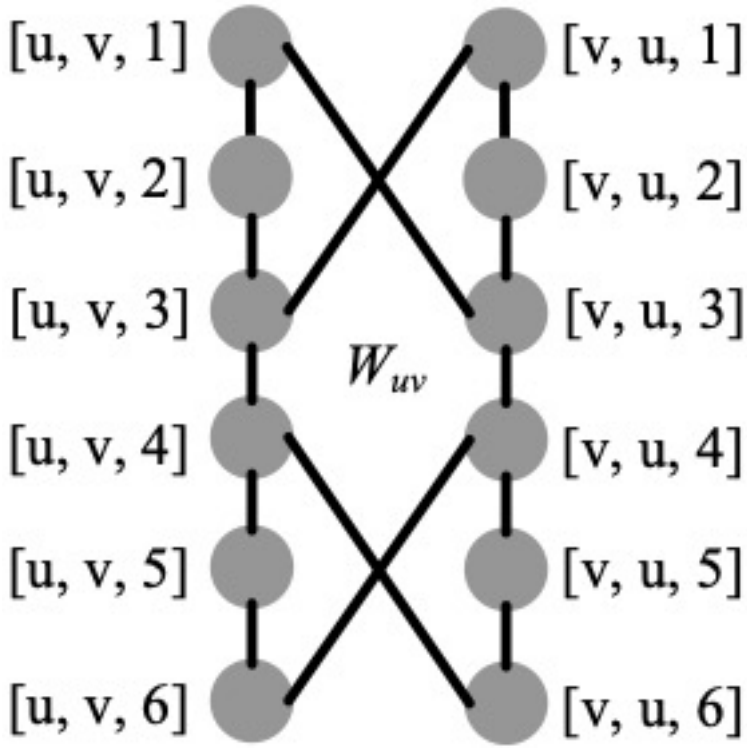
Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Descrição

Para cada aresta $\{u, v\}$ do grafo G, o grafo G' conterá uma cópia deste dispositivo, denotada por w_{uv}.

Cada aresta em w_{uv} é denotada por $[u, v, i]$ ou $[v, u, i]$ em que $1 \leq i \leq 6$

Cada dispositivo w_{uv} contém 12 vértices e 14 arestas.



Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

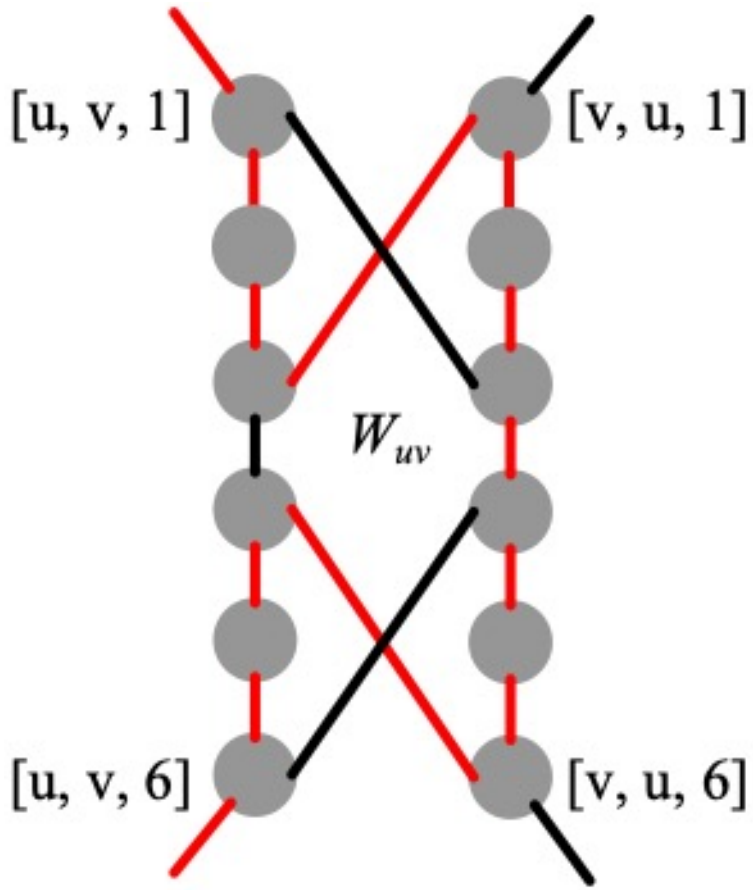
Verificação da Propriedade 2

A relação entre o dispositivo e o grafo G' é pré-determinada

- Apenas os vértices $[u, v, 1]$, $[u, v, 6]$, $[v, u, 1]$, $[v, u, 6]$ possuem arestas externas ao dispositivo;
- Qualquer ciclo Hamiltoniano em G' percorrerá as arestas do dispositivo;
- Só existem 3 casos.

Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

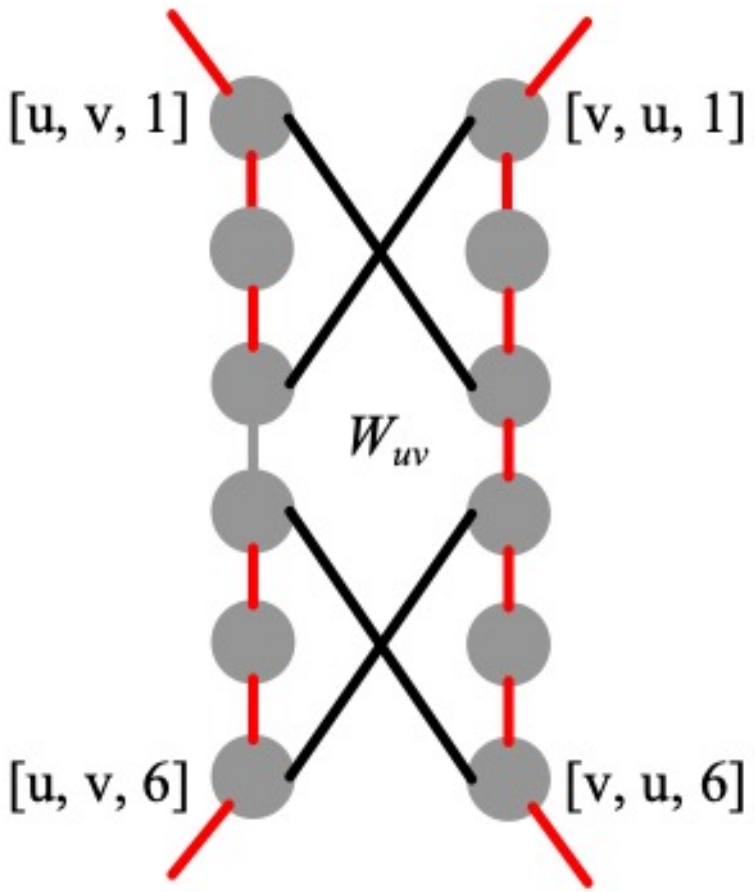
Se o ciclo passar pelo vértice [u, v, 1], ele deve sair pelo vértice [u, v, 6] e visitar todos os vértices do dispositivo.



Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Ou visitar apenas os 6 vértices de [u, v, 1] a [u, v, 6].

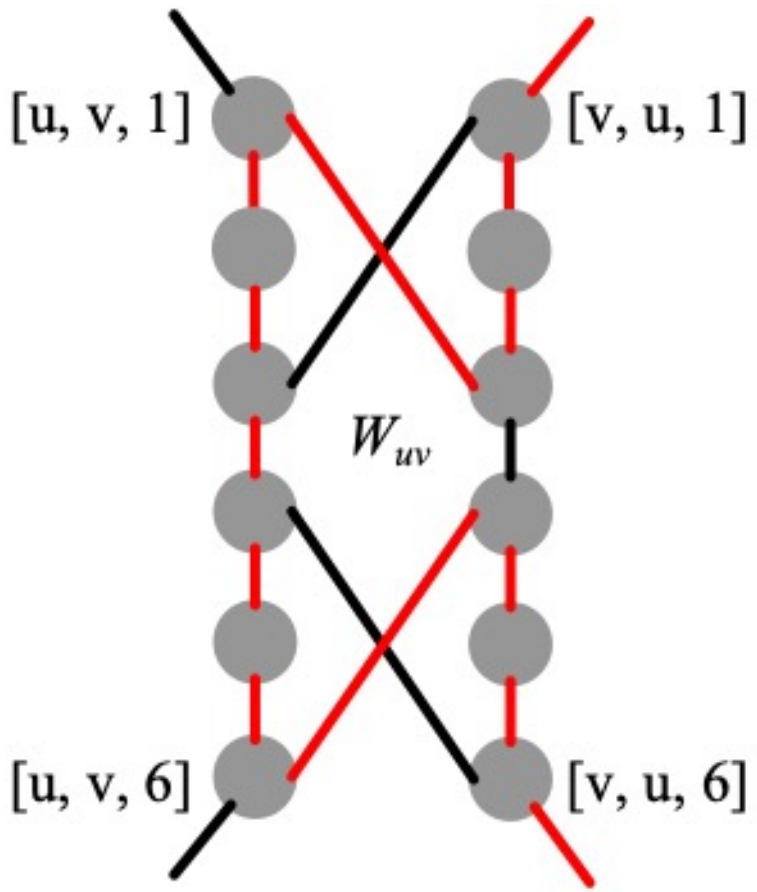
Neste caso, o ciclo deverá entrar novamente no dispositivo para visitar os demais vértices.



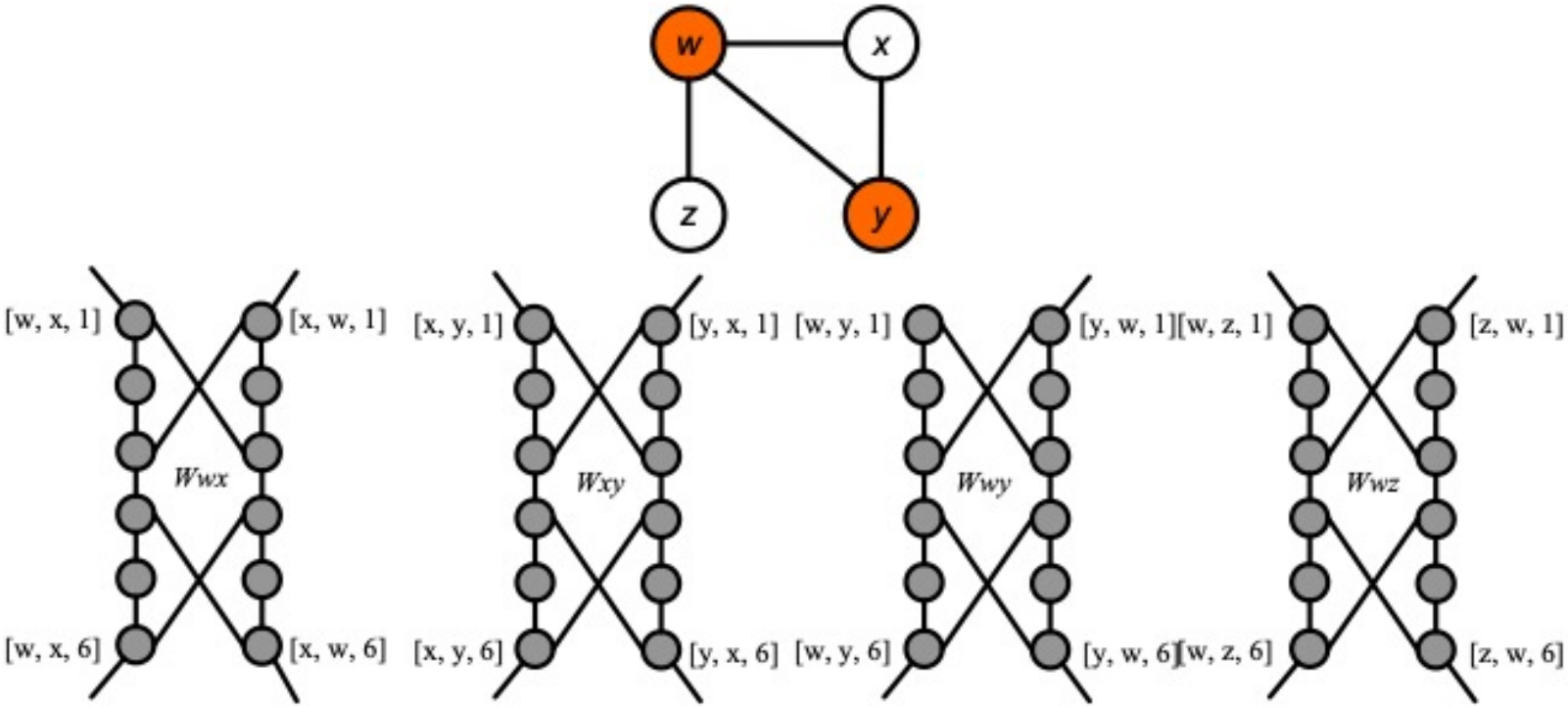
Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

De forma semelhante, se o ciclo entrar pelo vértice $[v, u, 1]$, deverá sair pelo vértice $[v, u, 6]$ e visitar todos os 12 vértices do dispositivo.

Ou os 6 vértices, de $[v, u, 1]$ a $[v, u, 6]$, conforme a figura anterior.



Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

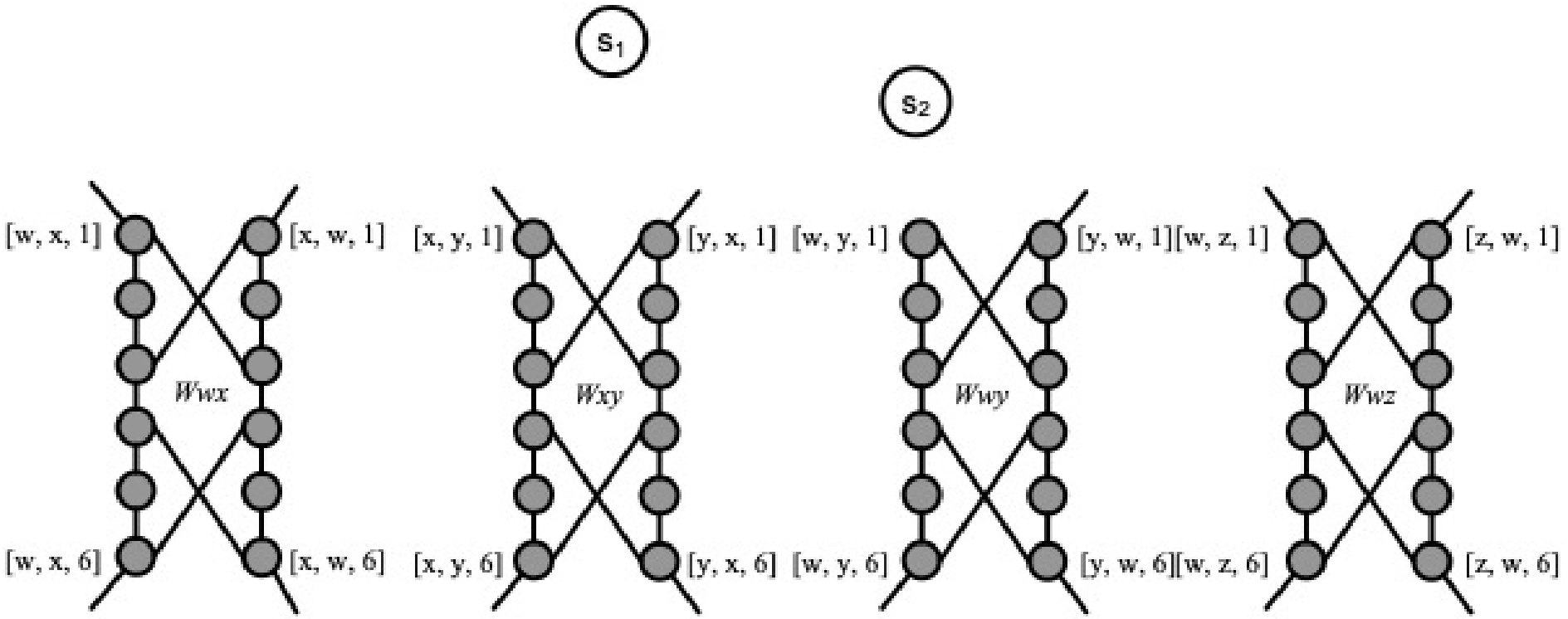


Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Verificação da Propriedade 2

Além dos vértices dos dispositivos, adicionamos a V' vértices seletores $s_1, s_2, s_3, \ldots, s_k$.

As arestas incidentes a vértices seletores serão utilizadas para selecionar os k vértices da cobertura.



Verificação da Propriedade 2

Existem ainda dois outros tipos de arestas em E'.

Para cada vértice u pertencente a V, são arestas que unem pares de dispositivos.

Formando um caminho que possui todos os dispositivos correspondentes a arestas incidentes sobre u.

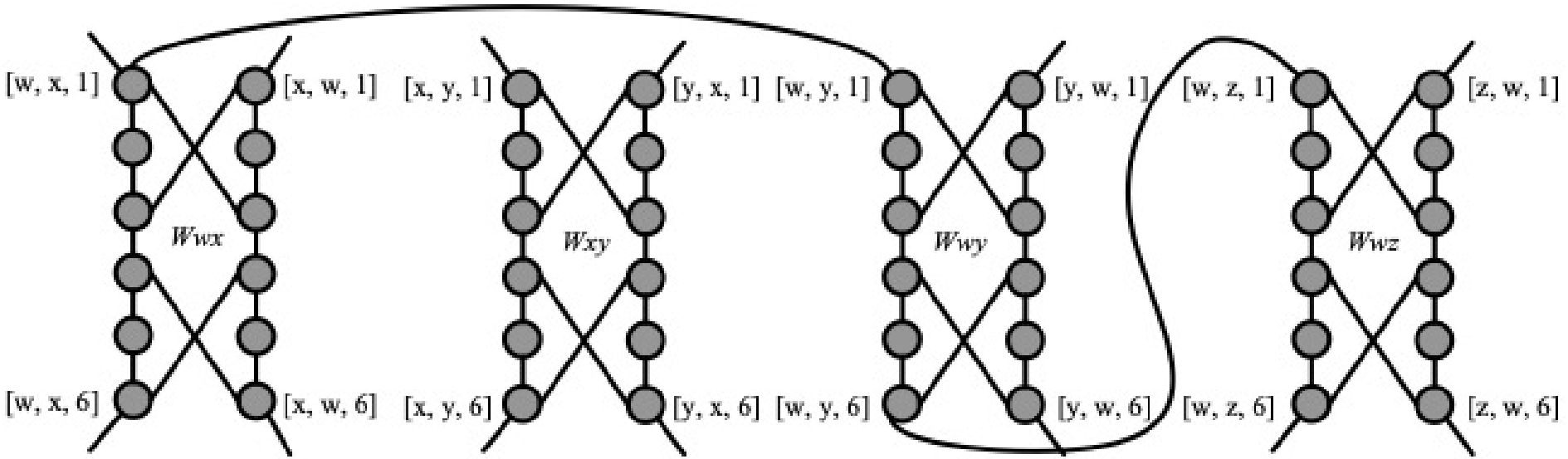
Os vértices incidentes a u são arbitrariamente ordenadas e então criamos um caminho passando por todos os dispositivos relacionados, adicionando os vértices a E'.

Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Verificação da Propriedade 2

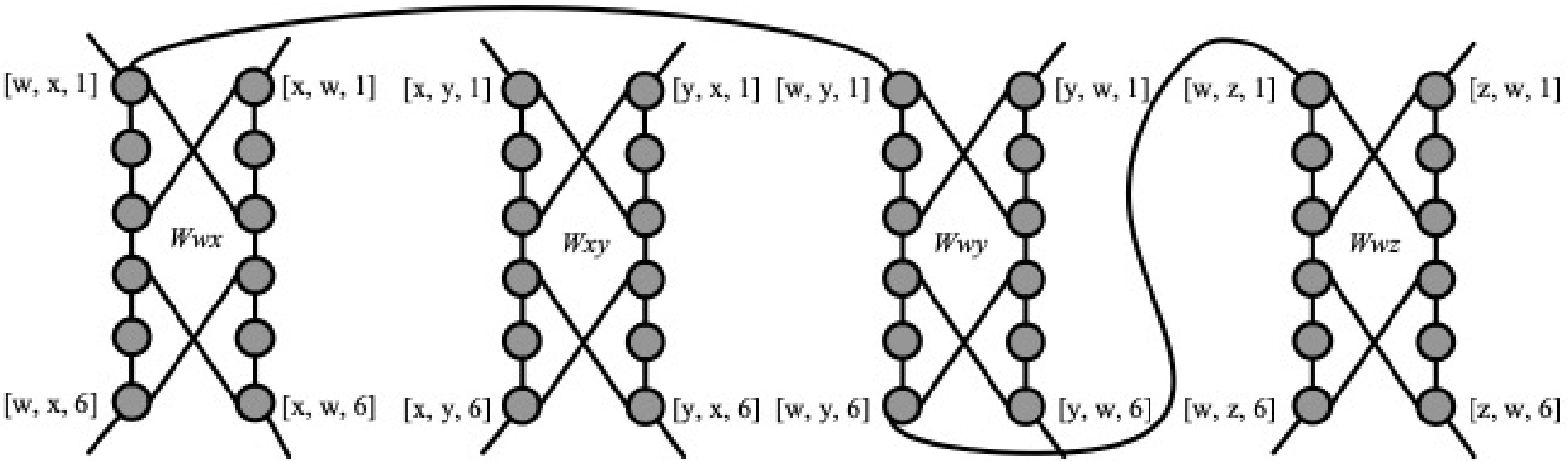
Suponha que ordenamos os vizinhos de w como x, y e z.

Adicionamos as arestas $\{[w, x, 6], [w, y, 1])$ e $([w, y, 6], [w, z, 1]\}$.



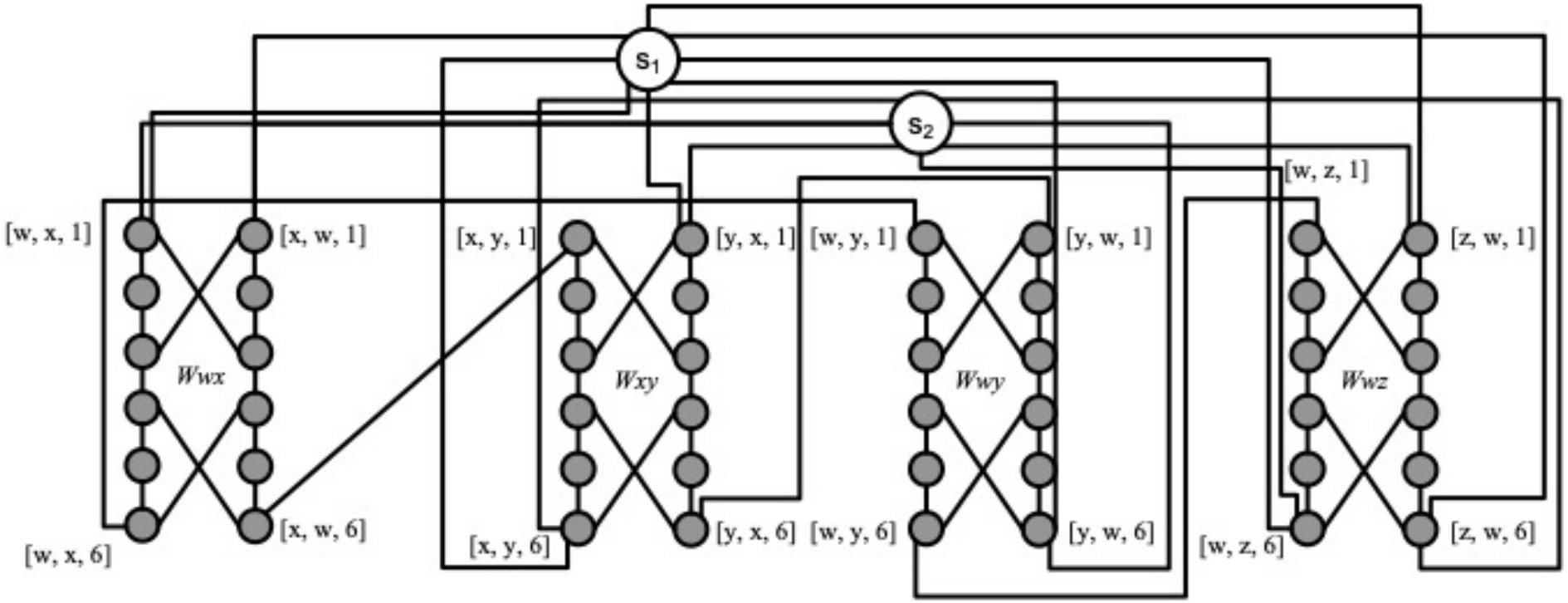
Verificação da Propriedade 2

A idéia é: se selecionarmos um vértice u de G na cobertura de vértices, podemos "cobrir" todos os dispositivos relacionados às arestas incidentes a u através de um caminho em G'.



Verificação da Propriedade 2

O segundo tipo de aresta une o primeiro e o último vértices de cada um desses caminhos a cada um dos vértices seletores.



Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

Verificação da Propriedade 2

Pode ser provado que o tamanho de G' é polinomial no tamanho de G.

Consequentemente, G' pode ser construído em tempo polinomial.

São adicionados

- $\leq 12|E|+|V|$ vértices;
- $2|E|$-$|V|$ arestas;
- O número total de arestas em G' é então $\leq 16|E|+(2|V|-1)|V|$.

Verificação da Propriedade 2

Por que funciona?

Suponha que G=(V, E) possui uma cobertura de vértices de tamanho k.

Formamos um ciclo Hamiltoniano em G incluindo as arestas a seguir para cada vértice u_j da cobertura de vértices

- Seja s_j um vértice seletor com $1 \leq j \leq k$.
- Sejam ainda u_j^i os vértices vizinhos de u_j, em uma ordenação arbitrária, com $1 \leq i \leq d(uj)$ e $1 \leq j \leq k$.

$$\{s_j, [u_j, u_j^{(i)}, 1] : 1 \leq j \leq k\}$$
$$\cup\{s_{j+1}, [u_j, u_j^{d(u_j)}, 6] : 11 \leq j \leq k-1\}$$
$$\cup\{s_1, [u_j, u_k^{d(u_k)}, 6]\}.$$

Verificação da Propriedade 2

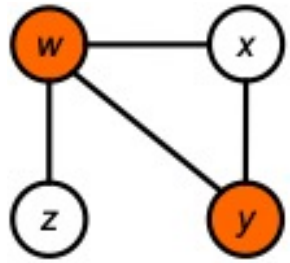
Incluímos as arestas da forma $\{([u_j, u_j^{(i)}, 6], [u_j, u_j^{(i+1)}, 6]): 1 \leq i \leq d(j)\}$ que conectam os dispositivos.

Também incluímos as arestas dos dispositivos, de acordo com um dos 3 casos de visitação e se a aresta é coberta mais de uma vez.

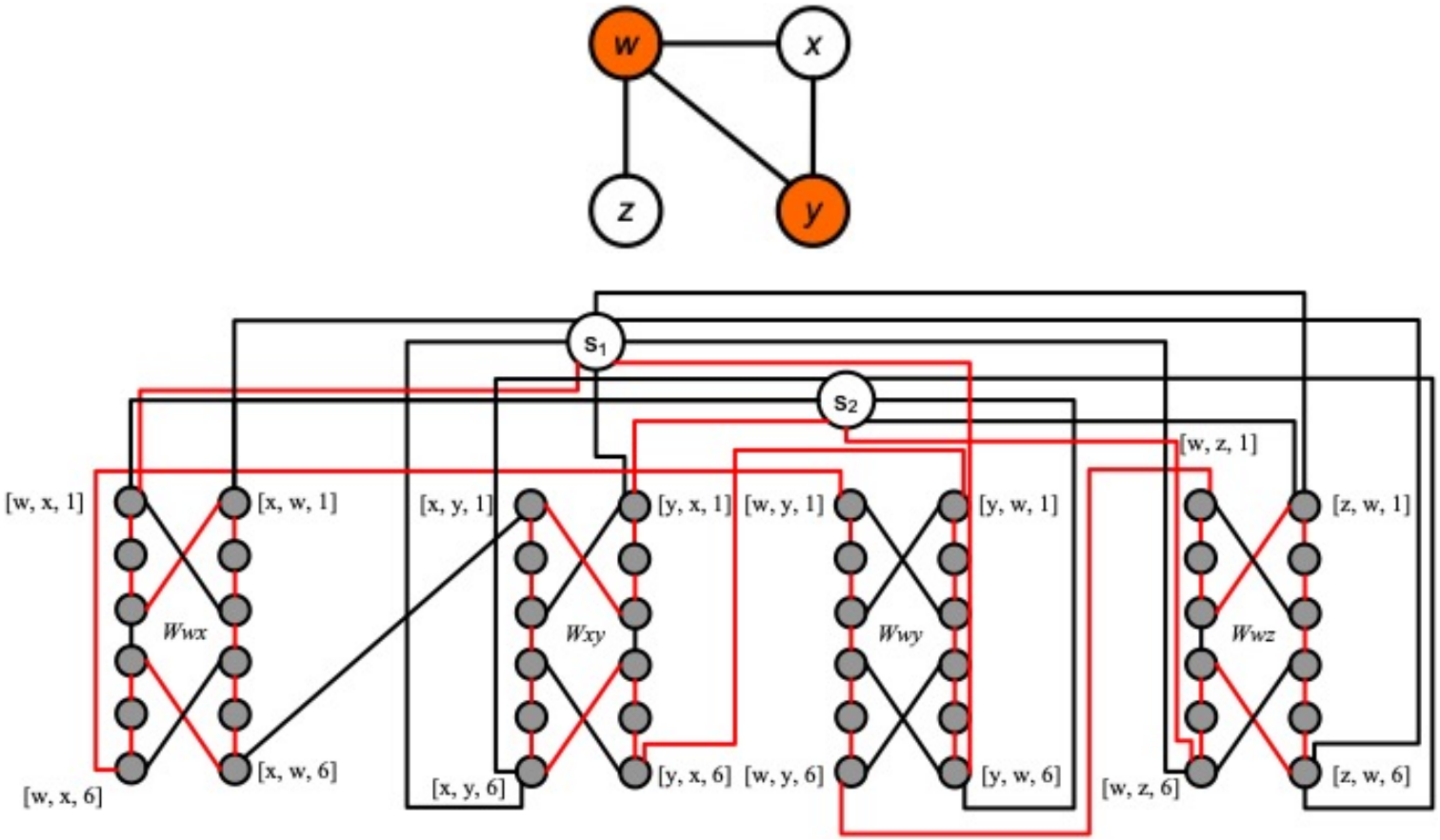
O slide a seguite apresenta o que temos para o grafo de exemplo.

Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

- $(S_1, [w, x, 1])$;
- $(S_1, [y, w, 6])$;
- $(S_2, [y, x, 1])$;
- $(S_2, [w, z, 6])$;
- $([y, x, 6], [y, w, 1])$;
- $([w, y, 6], [w, z, 1])$;
- $([w, x, 6], [w, y, 1])$.



Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano



Cobertura de Vértices$\leq_p$Ciclo Hamiltoniano

3-SAT

O 3-SAT é um caso especial do SAT, em que cada cláusula contém exatamente 3 literais. Adicionalmente, a expressão está na Forma Normal Conjuntiva (CNF): grupamentos AND de cláusulas, cada uma sendo um OR de um ou mais literais.

O 3-SAT é um problema NP-Completo, uma vez que SAT$\leq_p$3-SAT.

Subset Sum

Dado um conjunto (ou multiconjunto) de números inteiros, determinar se há um subconjunto não vazio cuja soma seja exatamente s.

3-SAT$\leq_p$Soma de Subconjuntos

Verificação da Propriedade 1

A solução para o problema é um conjunto S', subconjunto de S.

Caso S' seja de fato um subconjunto de S, e a soma dos elementos de S' seja igual a t, a solução é válida.

A verificação pode ser realizada em tempo polinomial, portanto, Soma de Subconjuntos $\in$ NP.

3-SAT$\leq_p$Soma de Subconjuntos

Verificação da Propriedade 2

Para transformarmos uma instância do 3-SAT, fazemos duas suposições simplificadoras sobre Φ:

- Nenhuma cláusula contém ao mesmo tempo uma variável e sua negação.
- Cada variável aparece em pelo menos uma cláusula. Caso contrário seria inútil uma atribuição para ela.

3-SAT$\leq_p$Soma de Subconjuntos

Verificação da Propriedade 2

Para cada variável x_i, são criados dois números em S.

Para cada cláusula C_j, são criados dois números em S.

Os números são representados na base 10, e cada um contém $n + k$ dígitos.

Cada dígito corresponde a uma variável ou cláusula.

A base 10 impede o transporte de dígitos mais baixos para dígitos mais altos.

3-SAT$\leq_p$Soma de Subconjuntos

Cada dígito corresponde a uma variável ou cláusula.

Os k dígitos menos significativos são representam as cláusulas.

Os k dígitos mais significativos representam as variáveis.

O destino t tem valor 1 em cada dígito referente uma variável e valor 4 em cada dígito referente a uma cláusula.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v'_1	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v'_2	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v'_3	=	0	0	1	1	1	0	0
t	=	1	1	1	4	4	4	4

3-SAT$\leq_p$Soma de Subconjuntos

Para cada variável x_i, existem dois inteiros, v_i e v'_i em S.

Cada um tem valor 1 no dígito referente à variável correspondente e 0 nos dígitos de outras variáveis.

Se o literal x_i aparece na cláusula C_j, então o dígito que identifica C_j em v_i tem valor 1.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v'_1	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v'_2	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v'_3	=	0	0	1	1	1	0	0
t	=	1	1	1	4	4	4	4

3-SAT$\leq_p$Soma de Subconjuntos

Se o literal $\neg x_i$ aparecer na cláusula C_j, então o dígito identificado por C_j em x'_i tem valor 1.

Em todas os outros dígitos identificados por cláusulas, v_i e v'_i têm valor 0.

Todos os valores v_i e v'_i são diferentes.

Bits menos significativos são diferentes.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v'_1	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v'_2	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v'_3	=	0	0	1	1	1	0	0
t	=	1	1	1	4	4	4	4

3-SAT$\leq_p$Soma de Subconjuntos

Para cada cláusula C_j, existem dois inteiros s_j e s'_j em S.

Cada s_j possui valor 1 no dígito identificado por C_j.

Cada s'_j possui valor 2 no dígito identificado por C_j.

Possuem valor 0 em todos os dígitos além do dígito identificado por C_j.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v'_1	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v'_2	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v'_3	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s'_1	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s'_2	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s'_3	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s'_4	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

3-SAT$\leq_p$Soma de Subconjuntos

S={ 1001001,1000110,
0100001, 0101110,
0010011, 0011100,
0001000, 0002000,
0000100, 0000200,
0000010, 0000020,
0000001, 0000002}
$t = 1114444$.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v'_1	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v'_2	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v'_3	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s'_1	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s'_2	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s'_3	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s'_4	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

3-SAT$\leq_p$Soma de Subconjuntos

S={ 1001001,1000110,
0100001, 0101110,
0010011, 0011100,
0001000, 0002000,
0000100, 0000200,
0000010, 0000020,
0000001, 0000002}
t = 1114444.
S'= v'1, v'2, v3;
Adiciona-se o valor das variáveis s_i e s'_i para totalizar o valor 4 em cada coluna.

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v'_1	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v'_2	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v'_3	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s'_1	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s'_2	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s'_3	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s'_4	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Verificação da Propriedade 2

Por que funciona?

Suponha que Φ possua uma atribuição satisfatória

Para cada variável x_i=1, incluímos, v_i em S', caso contrário incluímos v'_i;

Em outras palavras, incluímos em S' os literais com valor 1.

Desta forma, teremos o valor 1 na soma de cada dígito referente às variáveis

Correspondendo ao valor dos dígitos de t.

Como cada cláusula é satisfeita, a soma de cada dígito correspondente é pelo menos 1;

Completamos a soma até quatro usando o valor das variáveis s_i e s'_i correspondentes.

3-SAT$\leq_p$Soma de Subconjuntos

Verificação da Propriedade 2

A transformação pode ser realizada em tempo polinomial.

O conjunto S possui $2n + 2k$ valores, cada qual com $n + k$ dígitos.

Cada valor pode ser construído em tempo polinomial de $n + k$.

O destino t tem $n + k$ dígitos, e é produzido em tempo constante.

3-SAT$\leq_p$Soma de Subconjuntos

Verificação da Propriedade 2

Se G possuir uma atribuição satisfatória, então Soma de Subconjuntos possui um subconjunto cuja soma seja t.

3-SAT e Soma de Subconjuntos obtêm as mesmas respostas para a mesma instância, logo, Soma de Subconjuntos $\in$ NP-Completo.