JONILSO NOVACOSKI

# UMA INTRODUÇÃO À COMPLEXIDADE COMPUTACIONAL PARAMETRIZADA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Renato Carmo

CURITIBA

2013

N935i Novacoski, Jonilso
Uma introdução à complexidade computacional parametrizada/ Jonilso Novacoski. – Curitiba, 2013.
90 f. : il. color. ; 30 cm.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Informática, 2013.

Orientador: Renato Carmo .
Bibliografia: p. 86-90.

1. Complexidade computacional. 2. Grafos (Sistema de computador). 3. Algoritmos de computador. I. Universidade Federal do Paraná. II.Carmo, Renato. III. Título.

CDD: 511.352

JONILSO NOVACOSKI

# UMA INTRODUÇÃO À COMPLEXIDADE COMPUTACIONAL PARAMETRIZADA

Dissertação aprovada como requisito parcial à obtenção do grau de Mestre no Programa de Pós-Graduação em Informática da Universidade Federal do Paraná, pela Comissão formada pelos professores:

Orientador: Prof. Dr. Renato Carmo
Departamento de Informática, UFPR

Prof. Dr. André Luiz Pires Guedes
Departamento de Informática, UFPR

Prof. Dr. Murilo Vicente Gonçalves da Silva
Departamento de Informática, UTFPR

Curitiba, 18 de dezembro de 2013

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

**PARECER**

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Jonilso Vianei Novacoski, avaliamos o trabalho intitulado, *"Uma Introdução à Complexidade Computacional Parametrizada"*, cuja defesa foi realizada no dia 18 de dezembro de 2013, às 14:00 horas, no Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela:

(X) **aprovação** do candidato. ( ) **reprovação** do candidato.

Curitiba, 18 de dezembro de 2013.

Prof. Dr. Renato José da Silva Carmo
**DINF/UFPR** – Orientador

Prof. Dr. Murilo Vicente Gonçalves da Silva
**UTFPR** – Membro Externo

Prof. Dr. André Luiz Pires Guedes
**DINF/UFPR** – Membro Interno

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

# AGRADECIMENTOS

Ao meu orientador, Prof. Dr. Renato Carmo, pela imensa orientação, paciência, ajuda, acompanhamento, amizade e conhecimentos.

Ao Prof. Dr. André Luiz Pires Guedes, que considero meu co-orientador, pelas aulas, pela amizade, conhecimento e ajuda.

Ao Prof. Dr. Murilo Vicente Gonçalves da Silva pelas contribuições no trabalho.

Ao Programa de Pós-Graduação em Informática, do Setor de Ciências Exatas, da Universidade Federal do Paraná, na pessoa do seu coordenador 2011-2012 Prof. Dr. Luiz Eduardo Soares de Oliveira e na pessoa do seu coordenador 2013-2014 Prof. Dr. Marcos Didonet Del Fabro, pelo tremendo apoio recebido.

Ao Colegiado do Curso de Pós-Graduação em Informática, pelo apoio e compreensão nos tantos momentos difíceis.

À CAPES, pelo financiamento total deste trabalho.

À Secretaria de Estado da Educação do Estado do Paraná - SEED/PR, pela licença especial sem vencimentos.

Aos colegas de laboratório, em particular ao doutorando Alexandre Prusch Züge pelo esforço aplicado em seu trabalho de mestrado e pelo fornecimento cordial do mesmo.

À Secretária do Programa de Pós-Graduação em Informática, Jucélia Miecznikowski, por estar sempre disposta a cooperar.

Aos funcionários do Departamento de Informática, pela prontidão nos atendimentos.

Aos familiares, em particular ao irmão Prof. Dr. Josnei Antonio Novacoski, pela excelente dedicação, compreensão e ajuda.

Em especial, agradeço à minha esposa Ester de Arruda Campos, pela maravilhosa companhia em todos os momentos que compartilhamos desde 2005.

# SUMÁRIO

# LISTA DE FIGURAS

# RESUMO

A Complexidade Parametrizada é uma maneira de analisar a complexidade computacional de um problema computacional. Nesta dissertação damos uma Introdução à Complexidade Computacional Parametrizada com atenção aos problemas computacionais em grafos, concluindo com uma aplicação ao Problema da Clique Máxima.

Palavras-chave: Complexidade Parametrizada, Complexidade Computacional, Problema da Clique Máxima.

# ABSTRACT

The Parameterized Complexity is a form of analizing the computational complexity of a computacional problem. In this dissertation we give a Introduction to Computational Parameterized Complexity with atention to computational problems in graphs, concluding with an aplication to Maximum Clique Problem.

Key-words: Parameterized Complexity, Computational Complexity, Maximum Clique Problem.

# CAPÍTULO 1

# INTRODUÇÃO

O objetivo deste texto é apresentar uma introdução à Complexidade Computacional Parametrizada em português contendo uma aplicação ao problema computacional da Clique Máxima. Para a leitura do texto assumimos alguma familiaridade em Complexidade Computacional como descrita em [Garey and Johnson(1979)] e assumimos também alguma familiaridade nos conceitos de Teoria de Grafos.

A Complexidade Parametrizada é uma maneira de analisar um problema computacional com relação à entrada e com relação a um pedaço destacado da entrada. Este pedaço destacado é chamado de parâmetro do problema computacional.

Um dos objetivos ao destacar um parâmetro na análise de um problema computacional é identificar quanto o parâmetro é responsável pela dificuldade em resolver o problema computacional.

Quando um problema computacional é resolvido por um algoritmo, sua resolução pode ser usada em outro problema computacional ligeiramente diferente. Daí é bastante útil conhecer quais problemas computacionais são ligeiramente diferentes de quais outros problemas computacionais. De modo informal o termo redutível pode ser empregado quando um problema computacional é ligeiramente diferente de um outro problema computacional, dizendo-se que um problema é redutível ao outro problema.

É de grande importância conhecer a complexidade computacional dos problemas computacionais e também é de grande importância conhecer quais problemas computacionais são redutíveis uns aos outros.

Ao tomar dois problemas computacionais redutíveis entre si não há garantia de que os parâmetros de ambos sejam ligeiramente diferentes entre si. Por isso adicionar nas reduções entre problemas computacionais a exigência dos parâmetros de ambos serem

ligeiramente diferentes é um refinamento com relação a sequer considerar os parâmetros envolvidos.

As reduções entre problemas computacionais que consideram os parâmetros envolvidos também geram coleções de problemas computacionais, onde os problemas computacionais de uma coleção são redutíveis entre si. Informalmente, essas coleções são chamadas de classes de complexidade computacional.

As reduções de maior interesse neste texto são aquelas que preservam os parâmetros dos respectivos problemas computacionais. Tudo isto será formalmente definido e claramente exemplificado nos demais capítulos.

A principal referência é o livro [Downey and Fellows(1999)].

Os teoremas deste texto contendo uma prova completa são aqueles cuja demonstração seja esclarecedora para uma introdução à Complexidade Computacional Parametrizada. Por esse mesmo motivo alguns teoremas contem apenas a estrutura da prova. Ainda, alguns teoremas tem apenas seu enunciado pois sua demonstração ou estrutura estão além de uma introdução à Complexidade Computacional Parametrizada.

O texto está estruturado da seguinte maneira.

No Capítulo 2 damos as definições necessárias para grafos, circuitos, problemas computacionais, linguagens, parâmetros e problemas parametrizados. As definições são ilustradas com exemplos e figuras para facilitar seu entendimento.

No Capítulo 3 definimos as reduções entre problemas computacionais e definimos as reduções que levam os parâmetros em consideração, fazendo comparações com algumas reduções há muito conhecidas.

No Capítulo 4 definimos os problemas computacionais relacionados as fórmulas da Teoria da Lógica Proposicional, estudando as reduções entre eles e para outros problemas computacionais. Também aqui fazemos comparações com algumas reduções há muito conhecidas.

No Capítulo 5 definimos circuitos e alguns problemas computacionais envolvendo circuitos, estudando algumas reduções entre eles e para outros problemas já definidos.

No Capítulo 6 definimos a classe dos problemas tratáveis por parâmetro fixo. Apresentamos análises de problemas computacionais levando em consideração seus parâmetros e também mostramos alguns métodos para classificar problemas computacionais.

No Capítulo 7 definimos quasi-ordem, quasi-boa-ordem e o Teorema de Robertson e Seymour. Também vemos algumas relações do Projeto "Graph Minors" de Robertson e Seymour com a classe dos problemas tratáveis por parâmetro fixo.

No Capítulo 8 definimos as classes de complexidade $W[t]$ e vemos em detalhes a classe $W[1]$, que é tida como análoga à classe de problemas $\mathcal{NP}$-Completos. Ainda neste capítulo apresentamos algumas relações entre as classes $W[t]$.

No Capítulo 9 fazemos uma aplicação de Complexidade Parametriza ao problema da clique máxima, mostrando a análise de alguns algoritmos recentes com relação aos parâmetros envolvidos.

# CAPÍTULO 2

# DEFINIÇÕES

Apresentamos neste capítulo as definições necessárias em relação aos problemas computacionais, às linguagens e aos parâmetros. Em relação à Teoria de Grafos acompanhamos o livro [Bondy and Murty(2008)], copiando aqui algumas definições essenciais.

## 2.1 Grafos

Um grafo $G$ é um par ordenado $(V(G), E(G))$, onde $V(G)$ é um conjunto finito de vértices e $E(G)$ é um conjunto finito de arestas. Um grafo $G = (V(G), E(G))$ é um grafo simples se não contém arestas paralelas nem laços.

Uma cobertura por vértices em um grafo simples $G$ é um conjunto de vértices $V' \subseteq V(G)$ que intercepta todas as arestas. Um conjunto independente em $G$ é um conjunto de vértices $V' \subseteq V(G)$ que não contém arestas, isto é, onde para todo $u, v \in V'$ tem-se $\{u, v\} \notin E(G)$.

Uma clique em um grafo simples $G$ é um conjunto de vértices $V' \subseteq V(G)$ tal que $G[V']$ é um grafo completo.

Um conjunto dominante em um grafo simples $G$ é um conjunto de vértices $V' \subseteq V(G)$ onde cada vértice $v$ de $V(G)$ ou $v$ pertence a $V'$ ou $v$ é vizinho de algum vértice $V'$.

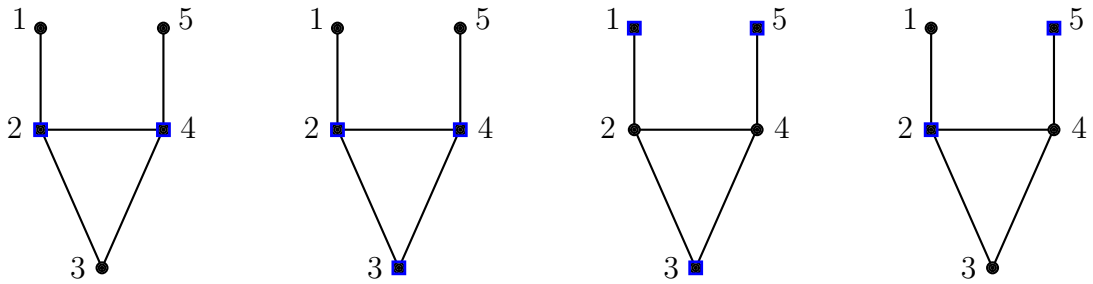


Figura 2.1: Cobertura por vértices, clique, conjunto independente e conjunto dominante

Na Figura 2.1 o grafo da esquerda tem os vértices 2 e 4 destacados onde o conjunto $V' = \{2, 4\}$ é uma cobertura por vértices. No segundo grafo da Figura 2.1 o conjunto $V' = \{2, 3, 4\}$ é a clique máxima. O conjunto $V' = \{1, 3, 5\}$ do terceiro grafo da Figura 2.1 é um conjunto independente e o conjunto $V' = \{2, 5\}$ no grafo da direita é um conjunto dominante.

No interesse de simplificar a leitura escrevemos $G = (V, E)$ para significar o grafo $G = (V(G), E(G))$. Também simplificaremos para $V$ e $E$ para respectivamente significar $V(G)$ e $E(G)$ sempre que ficar claro no contexto.

## 2.2 Problemas Computacionais

Um problema computacional $\Pi$ é uma tripla $\Pi = (D, R, \psi)$ onde $D$ é o conjunto chamado de conjunto de instâncias de $\Pi$, $R$ é o conjunto chamado de conjunto de respostas de $\Pi$ e $\psi \subseteq D \times R$ é uma relação que associa cada instância $I \in D$ a um conjunto $R_I \subseteq R$ de repostas da instância $I$.

Um problema de decisão é um problema computacional onde cada uma de suas instâncias admite exatamente uma dentre duas possíveis respostas. Nos problemas de decisão $\Pi = (D, R, \psi)$ deste texto o conjunto $R$ de respostas é $R = \{1, 0\}$.

COB é o problema de decisão COB $= (D, R, \psi)$ da cobertura por vértices, onde $D$, $R$ e $\psi$ seguem conforme a descrição abaixo.

- As instâncias em $D$ são pares $(G, k)$ onde $G$ é um grafo simples e $k$ um inteiro,
- O conjunto $R$ de respostas é $R = \{1, 0\}$ e
- $\psi(G, k) = 1$ se e somente se $G$ tem uma cobertura por vértices de tamanho $k$.

IND é o problema de decisão IND $= (D, R, \psi)$ do conjunto independente, onde as instâncias em $D$ são pares $(G, k)$, $G$ sendo um grafo simples e $k$ um inteiro, o conjunto $R$ de respostas é $R = \{1, 0\}$ e $\psi(G, k) = 1$ se e somente se o grafo $G$ tem um conjunto independente de tamanho $k$.

CLIQUE é o problema de decisão da clique semelhantemente à COB e IND, com $\psi(G, k) = 1$ se e somente se o grafo simples $G$ tem uma clique de tamanho $k$.

Consideremos o problema de decisão $\Pi = (D, R, \psi)$. O conjunto $Y_\Pi$ é o subconjunto $Y_\Pi \subseteq D$ das instâncias de $\Pi$ tal que $\psi(I) = 1$ se e somente se $I \in Y_\Pi$.

As instâncias positivas (do inglês *yes-instances*) de um problema de decisão $\Pi = (D, R, \psi)$ são os elementos do conjunto $Y_\Pi$.

Assim o conjunto $Y_\Pi$ é o conjunto das instâncias positivas do problema $\Pi$.

Para exemplo de instâncias positivas consideremos o grafo $G = (V(G), E(G))$ da Figura 2.1, onde com $V = \{1, 2, 3, 4, 5\}$ e $E = \{\{1, 2\}, \{2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}$. O grafo $G$ tem cobertura por vértices de tamanho 2 e tem conjunto independente de tamanho 3. Assim, $(G, 2)$ é uma instância positiva de COB e $(G, 3)$ é uma instância positiva de IND. Entretanto $(G, 4)$ não é uma instância positiva de IND, pois nenhum conjunto de 4 vértices é conjunto independente em $G$.

Daqui em diante consideraremos os problemas sempre sendo problemas computacionais de decisão, exceto quando explícito de outra forma.

## 2.3 Linguagem

As definições de esquema de representação, codificação, modelo de computação, $\mathcal{P}$ e $\mathcal{NP}$-Completo são as mesmas do texto de [Garey and Johnson(1979)].

Um alfabeto $\Sigma$ é um conjunto finito não vazio. Uma palavra sobre o alfabeto $\Sigma$ é uma sequência finita de elementos desse alfabeto. O conjunto de todas as palavras sobre $\Sigma$ é $\Sigma^*$. Uma linguagem $L$ é um subconjunto de $\Sigma^*$.

Existe uma correspondência natural entre problemas de decisão e linguagens sobre um alfabeto $\Sigma$, a saber, ao problema de decisão $\Pi$ corresponde a linguagem $L$ formada pelas codificações das instâncias positivas $Y_\Pi$.

## 2.4 Parâmetro e problema parametrizado

Um parâmetro para um problema computacional $\Pi = (D, R, \psi)$ é qualquer coisa calculada a partir das instâncias $I \in D$ em tempo polinomial. Este limite para calcular o parâmetro será relaxado no Capítulo 6, entretanto um dos objetivos ao destacar um parâmetro é identificar quanto o parâmetro é responsável pela dificuldade em resolver o problema computacional, daí calcular o parâmetro precisa ser inferior em dificuldade do que resolver o problema a ele associado.

Como um parâmetro é qualquer coisa calculada a partir $I \in D$, então dado um problema computacional podemos adotar um parâmetro em meio a uma variedade de possibilidades.

Consideremos o problema de decisão COB da cobertura por vértices anteriormente definido. Cada instância $I \in D$ está na forma $(G, k)$. Um parâmetro para COB com instância $(G, k)$ pode ser $k$, chamado de "parâmetro natural" do problema.

Outro parâmetro para COB pode ser o número de arestas do grafo $G$, pois podemos calcular o número de arestas de $G$ em tempo polinomial. Do mesmo modo parâmetros para COB podem ser o número de vértices de $G$, o grau máximo de $G$, o grau mínimo de $G$ ou outra coisa calculada a partir da instância em tempo polinomial.

O parâmetro para COB também poderia ser um subgrafo ou uma estrutura algébrica.

Neste texto os parâmetros estarão restritos aos números inteiros não negativos, ou seja, daqui em diante o parâmetro adotado sempre será um número inteiro não negativo, exceto quando explícito de outra forma.

**Definição 2.1.** *Um problema parametrizado é o par composto de problema* $\Pi$ *e parâmetro.*

Para cada instância $x \in D$ do problema $\Pi$, seja $y$ o parâmetro calculado a partir de $x$. Seja $P = \{\cup_{x \in D} (x, y)\}$. Daí $P$ é um subconjunto de $\Pi \times \mathbb{N}$. Pela Definição 2.1 temos que $P$ é um problema parametrizado. O par ordenado $(x, y) \in P$ é chamado de instância do problema parametrizado $P$, a instância $x \in D$ (instância de $\Pi$) é chamada de parte principal do problema parametrizado $P$ e o inteiro $y$ é chamado de parâmetro.

Representamos $P$ do seguinte modo.

| Problema parametrizado $P$ a partir de $\Pi$ | |
|---|---|
| Instância: | Um elemento $x$ de $D$ e um inteiro positivo $y$. |
| Parâmetro: | O valor $y$ calculado a partir de $x$ em tempo polinomial. |
| Pergunta: | A instância $x$ é uma instância positiva de $\Pi$? |

A Complexidade Parametrizada é uma maneira de analisar e classificar os problemas parametrizados. Para cada possível parâmetro de um problema computacional temos um problema parametrizado e para cada problema parametrizado temos uma análise de sua complexidade computacional.

Abaixo damos alguns exemplos de problemas parametrizados relacionados ao problema de cobertura por vértices.

| Cobertura por Vértices | |
|---|---|
| Instância: | Um grafo $G$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | $G$ tem uma cobertura por vértices de tamanho $k$? |

| Cobertura por Vértices I | |
|---|---|
| Instância: | Um grafo $G$, um inteiro positivo $k$ e o número $n$ de vértices. |
| Parâmetro: | O número $n$ de vértices de $G$. |
| Pergunta: | $G$ tem uma cobertura por vértices de tamanho $k$? |

| Cobertura por Vértices II | |
|---|---|
| Instância: | Um grafo $G$, um inteiro positivo $k$ e o grau mínimo $\delta$. |
| Parâmetro: | O grau mínimo $\delta$ de $G$. |
| Pergunta: | $G$ tem uma cobertura por vértices de tamanho $k$? |

Na introdução deste texto colocamos como um dos objetivos ao usar parâmetros na análise o de identificar e expressar quanto o parâmetro é responsável pela complexidade computacional do problema. Este objetivo é perdido nos casos onde o parâmetro torna-se tão grande quanto a parte principal. Com isso em mente consideremos o problema abaixo.

Conjunto Independente Planar

| | |
|---|---|
| Instância: | Um grafo planar $G$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | $G$ tem um conjunto independente de tamanho $k$? |

Uma análise em palavras simplistas seria: neste caso $k$ é tão grande que mais atrapalha na análise do que ajuda.

Vejamos informalmente uma análise deste problema parametrizado.

Seja $G$ um grafo planar e $k$ um inteiro positivo para uma instância de Conjunto Independente Planar. Como $G$ é um grafo planar, então podemos usar o famoso teorema das quatro cores, que garante que um grafo planar pode ter os vértices coloridos com quatro cores de maneira que vértices de mesma cor formam um conjunto independente no grafo. Se $k \leq \lceil n/4 \rceil$ basta colorir o grafo em quatro cores e verificar o tamanho do conjunto de vértices de cada cor. Se $k > \lceil n/4 \rceil$ então $n < 4k$. Neste caso expressar uma análise em relação a $n$ e a $k$ ao invés de em relação a $n$ não apresenta vantagens pois $n < 4k$.

Este modo de escolher o parâmetro também ocorre com o seguinte problema.

Cobertura por Vértices III

| | |
|---|---|
| Instância: | Um grafo $G$ e um inteiro positivo $k$. |
| Parâmetro: | A quantidade $m$ de arestas de $G$. |
| Pergunta: | $G$ tem uma cobertura por vértices de tamanho $k$? |

Encontrar a cobertura por vértices é encontrar o conjunto onde "vértices cobrem arestas", ou seja, já está relacionado com as arestas. Daí tomar o parâmetro como a quantidade $m$ de arestas não ajuda a identificar quanto o parâmetro é responsável pela complexidade computacional do problema.

## 2.5 Instância positiva em um problema parametrizado

A instância $(x, y) \in P$ é uma instância positiva do problema parametrizado $P \subseteq \Pi \times \mathbb{N}$ se e somente se a parte principal $x$ é uma instância positiva para o problema de decisão $\Pi$ e o parâmetro $y$ tem o mesmo valor do que o calculado a partir da parte principal $x$.

Denotamos por $Y_P$ o conjunto das instâncias positivas do problema parametrizado $P$.

Se $P \subseteq \Pi \times \mathbb{N}$, então $Y_P \subseteq Y_\Pi \times \mathbb{N}$.

Vejamos alguns exemplos nos quais temos algumas instâncias positivas para os problemas parametrizados Cobertura por Vértices, Cobertura por Vértices I e Cobertura por Vértices II definidos anteriormente. Seja $G$ o grafo representado na Figura 2.2: $G$ tem 6 vértices, grau mínimo 2 e com coberturas por vértices de tamanhos 4, 5 e 6.

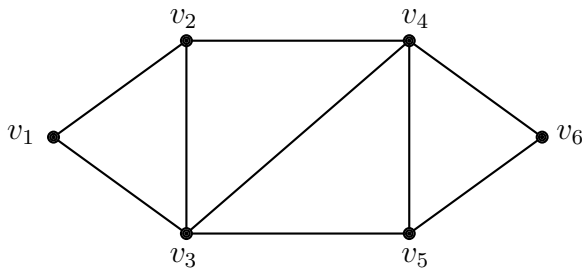


Figura 2.2: Um grafo simples

- $((G, 4), 4)$ é uma instância positiva para a Cobertura por Vértices.
- $((G, 4), 6)$ é uma instância positiva para a Cobertura por Vértices I.
- $((G, 4), 2)$ é uma instância positiva para a Cobertura por Vértices II.
- $((G, 5), 5)$ e $((G, 6), 6)$ são instâncias positivas para a Cobertura por Vértices.
- $((G, 4), 6)$ não é uma instância positiva para a Cobertura por Vértices, pois embora $(G, 4)$ seja instância positiva para COB o valor 6 em $((G, 4), 6)$ não é o valor do parâmetro para Cobertura por Vértices.
- $((G, 4), 2)$ não é uma instância positiva para a Cobertura por Vértices I porque

o parâmetro na Cobertura por Vértices I é o número de vértices do grafo enquanto $G$ tem 6 vértices e não 2.

- $((G, 4), 4)$ não é uma instância positiva para a Cobertura por Vértices II pois o parâmetro na Cobertura por Vértices II é o grau mínimo, mas $G$ tem $\delta = 2$.

Para simplificar a leitura e sem perda de generalidade denotaremos por $(G, k)$ uma instância $((G, k), k)$ sempre que ficar claro no contexto.

## 2.6 Linguagem parametrizada

Uma linguagem parametrizada $L$ é um subconjunto de $\Sigma^* \times \mathbb{N}$.

Do mesmo modo que na correspondência natural entre problemas de decisão $\Pi$ e linguagens $L$ temos a correspondência natural entre problemas parametrizados e linguagens parametrizadas, a saber, ao problema parametrizado $P \subseteq \Pi \times \mathbb{N}$ corresponde a linguagem parametrizada $L$ formada pelas codificações das instâncias positivas $Y_P$ do problema parametrizado $P$.

# CAPÍTULO 3

# REDUÇÕES ENTRE PROBLEMAS COMPUTACIONAIS

Neste capítulo descrevemos as reduções parametrizadas entre problemas parametrizados. Continuamos usando letras com fontes em maiúsculas para representar problemas computacionais de decisão, como COB visto anteriormente, e usando letras com fonte diferenciada como Cobertura por Vértices para representar problemas parametrizados.

A noção de redução entre problemas computacionais é uma ideia comum detrás dos resultados de completude. Para um modelo apropriado de computação, uma redução de um problema $P$ para um problema $P'$ é um método que computa $P$ a partir de $P'$ neste modelo. Assim, as reduções permitem que os problemas possam ser parcialmente ordenados em termos de sua complexidade computacional.

Uma função $f$ é limitada polinomialmente se e somente se $f$ é uma transformação polinomial como em [Garey and Johnson(1979)].

Uma redução de $P$ para $P'$ é uma função $f\colon \Sigma^* \to \Sigma^*$ limitada polinomialmente tal que $x \in P$ se e somente se $f(x) \in P'$, para todo $x \in \Sigma^*$. Estas reduções tem grande importância principalmente pelo fato a seguir relacionado a classe de problemas $\mathcal{P}$, classe definida em [Garey and Johnson(1979)].

**Teorema 3.1.** *Sejam $P_1$ e $P_2$ dois problemas tal que existe uma redução de $P_1$ para $P_2$. Se $P_2 \in \mathcal{P}$ então $P_1 \in \mathcal{P}$, ou seja, se $P_1 \notin \mathcal{P}$ então $P_2 \notin \mathcal{P}$.*

Dois problemas, $P$ e $P'$, são tidos de mesma complexidade computacional do ponto de vista de uma dada redução se e somente se existe uma redução de $P$ para $P'$ e vice-versa.

Sejam $\Pi_1 = (D_1, R_1, \psi_1)$ e $\Pi_2 = (D_2, R_2, \psi_2)$ problemas de decisão. Uma redução de $\Pi_1$ para $\Pi_2$ é uma função $f\colon D_1 \to D_2$ limitada polinomialmente tal que para todo $I \in D_1$, $I \in Y_{\Pi_1}$ se e somente se $f(I) \in Y_{\Pi_2}$.

Se existe uma função $f$ nessas condições para a redução de $\Pi_1$ para $\Pi_2$ então dizemos que $\Pi_1$ é redutível a $\Pi_2$ e escrevemos

$$\Pi_1 \propto \Pi_2.$$

Consideremos os problemas de decisão COB e IND definidos anteriormente, os quais são relacionados respectivamente à cobertura por vértices e ao conjunto independente.

**Teorema 3.2.** *COB é redutível a IND, ou seja, COB* $\propto$ *IND.*

*Demonstração.* Seja $(G, k)$ uma instância de COB.

É necessário e suficiente uma função $f$ com $f(G, k) = (G', k')$ tal que $G$ tem uma cobertura por vértices de tamanho $k$ se e somente se $G'$ tem um conjunto independente de tamanho $k'$, com a função $f$ sendo limitada polinomialmente.

Vale lembrar da teoria de grafos onde um grafo $G$ com $n$ vértices tem uma cobertura por vértices de tamanho $k$ se e somente se o grafo $G$ tem um conjunto independente de tamanho $n - k$.

Daí precisamos saber o valor $n$ de vértices em $G$, o que calculamos em tempo polinomial. Fazemos $G' = G$ e $k' = n - k$ também em tempo polinomial.

Portanto construímos $(G', k')$, tal que $(G, k) \in Y_{COB}$ se e somente se $(G', k') \in Y_{IND}$.

□

Como o raciocínio da redução dada na demonstração é reversível, temos

**Corolário 3.3.** *IND é redutível a COB, ou seja, IND* $\propto$ *COB.*

Assim COB e IND tem a mesma complexidade computacional, já que IND é redutível a COB e COB é redutível a IND.

## 3.1 Reduções entre problemas parametrizados

A diferença da redução entre problemas computacionais para a redução entre problemas parametrizados é a adicional exigência de que o parâmetro do segundo problema parametrizado ser limitado por uma transformação do parâmetro do primeiro. Mais formalmente temos,

**Definição 3.4.** *Sejam $P$ e $P'$ problemas parametrizados. $P$ é redutível a $P'$ por uma redução parametrizada se existe uma função $g$ limitada polinomialmente e se existe uma função $f\colon P \to P'$ com $f(x,k) = (g(x,k), h(k)) = (x', k')$ tal que para todo $(x,k) \in P$, $(x,k) \in Y_P$ se e somente se $f(x,k) \in Y_{P'}$.*

Vale comentar algumas das características da Definição 3.4 acima quando existe $f$ e $g$.

- A função $f$ mapeia instâncias positivas de $P$ em instâncias positivas de $P'$. E se $f(x,k)$ é uma instância positiva em $P'$ então $(x,k)$ é instância positiva em $P$.
- A função $g$, que leva $(x,k)$ em $x'$ com $g(x,k) = x'$, é limitada polinomialmente. Pela definição usual para a notação $O$ temos que $g$ é $O(|x|^c)$, para $c \geq 0$.
- Para computar $x'$ a função $g$ pode usar a instância $(x,k)$, pois o domínio da função $g$ é o mesmo que o da função $f$.
- Para computar $k'$ a função $h$ não pode usar o termo $x$ da instância $(x,k)$, pois na definição consta $f(x,k) = (g(x,k), h(k))$. Daí a função $h$ depende apenas de $k$ e é independente de $x$, ou seja, o domínio de $h$ é o conjunto dos parâmetros de $P$.
- A função $h$ não é necessariamente limitada polinomialmente.
- A complexidade da redução de $P$ a $P'$ por uma redução parametrizada é de tempo $O(g(x,k)h(k))$, com $g$ sendo limitada polinomialmente. Ou seja, a complexidade da redução de $P$ a $P'$ é de tempo $O(|x|^c h(k))$ para $c > 0$.

Se o problema parametrizado $P$ é redutível ao problema parametrizado $P'$ por uma redução parametrizada, então escrevemos

$$P \leq_{rp} P'.$$

Se $P \leq_{rp} P'$ e $P' \leq_{rp} P$ dizemos que $P$ e $P'$ tem a mesma complexidade parametrizada e então escrevemos

$$P \cong P'.$$

## 3.2 Exemplos de reduções entre problemas parametrizados

Vejamos uma redução parametrizada entre os seguintes problemas parametrizados.

| Clique | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | $G$ tem uma clique de tamanho $k$? |

| Conjunto Independente | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | $G$ tem um conjunto independente de tamanho $k$? |

**Teorema 3.5.** Clique $\leq_{rp}$ Conjunto Independente *e* Conjunto Independente $\leq_{rp}$ Clique*, ou seja,* Clique $\cong$ Conjunto Independente.

Prova-se o Teorema 3.5 observando que uma instância $(x, k)$ de um dos problemas parametrizados é redutível a uma instância $(x', k')$ do outro problema pelos seguintes fatos.

- Um grafo $G$ tem clique de tamanho $k$ se e somente se o grafo complementar de $G$

tem conjunto independente de tamanho $k$. O parâmetro $k'$ depende apenas de $k$ (neste caso é idêntico a $k$).

- Computamos $x'$ em tempo $O(|x|^2)$ pois $x'$ é o grafo complementar computado a partir de $(x, k)$. Portanto obtemos $(x', k')$ em tempo $h(k).O(|x|^2)$ e $h$ depende apenas de $k$.

A redução acima é bastante conhecida tanto quanto a redução do problema da cobertura por vértices para o problema do conjunto independente exibida na demonstração do Teorema 3.2. Na redução COB $\propto$ IND do Teorema 3.2 foi tomada uma instância $(G, k)$ para COB e transformada na instância $(G, k' = n - k)$ de IND.

A conhecida redução CLIQUE $\propto$ IND foi usada na redução Clique $\leq_{rp}$ Conjunto Independente do Teorema 3.5. Mas isto não ocorre com a redução COB $\propto$ IND do Teorema 3.2, pois neste caso $k'$ depende do número de vértices sendo $k' = n - k$ e portanto $k'$ não depende apenas de $k$.

Vejamos isto de modo mais claro. A redução parametrizada da Definição 3.4 apresenta uma instância $(x, k)$ sendo transformada na instância $(x', k')$ tal que é necessário o valor $k'$ ser computado apenas a partir de $k$. Já no Teorema 3.2, o valor $k'$ não foi computado apenas a partir de $k$, pois foi necessário saber o número de vértices $n$ calculado a partir do grafo $G$ para computar $k' = n - k$ e daí $k'$ ficou dependente também de $x$ da instância $(x, k)$.

Portanto a redução do Teorema 3.2 não é uma redução parametrizada.

São relativamente poucas as reduções bastante conhecidas e que também são reduções parametrizadas.

Para outro exemplo de redução parametrizada consideremos o seguinte problema.

Clique por arestas

| | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Existe $M \subseteq E$, tal que $\|M\| = k$ e $G[M]$ é um grafo completo? |

**Teorema 3.6.** CLIQUE $\leq_{rp}$ CLIQUE POR ARESTAS *e* CLIQUE POR ARESTAS $\leq_{rp}$ CLIQUE*, ou seja,* CLIQUE $\cong$ CLIQUE POR ARESTAS.

*Demonstração.* Se $G = (V, E)$ é um grafo e $k$ é um inteiro positivo de modo que $(G, k)$ é uma instância positiva de CLIQUE, então $(G, k(k-1)/2)$ é uma instância positiva de CLIQUE POR ARESTAS.

Reciprocamente, consideremos uma instância $(G, k)$ de CLIQUE POR ARESTAS. Se $(G, k)$ é uma instância positiva de CLIQUE POR ARESTAS então existe $M \subseteq E$, com $|M| = k$, tal que $G[M]$ é um grafo completo. Existem também $k'$ e $k''$ inteiros tais que

$$(k'-1)(k'-2)/2 < k \leq k'(k'-1)/2 = k''.$$

Como $G[M]$ é um grafo completo então $k = k''$. Daí o conjunto $S$ de vértices de $G[M]$ tem tamanho $k'$ e $G[S]$ é um grafo completo. Portanto $(G, k')$ é uma instância positiva de CLIQUE.

□

Nos demais capítulos apresentamos outras reduções parametrizadas e faremos comparações entre elas e reduções bastante conhecidas.

## 3.3 Outras formas de reduções entre problemas parametrizados

A redução parametrizada da Definição 3.4 é uma das formas de redução dada em [Downey and Fellows(1999)], onde são apresentadas outras três formas de reduções buscando análises e reduções ainda mais refinadas. As outras formas de redução demandam definições que não cabem neste texto introdutório.

A redução parametrizada da Definição 3.4 é também chamada de redução parametrizada padrão.

# CAPÍTULO 4

# FÓRMULAS

Neste capítulo definimos fórmulas da Teoria da Lógica Proposicional e alguns problemas computacionais relacionados a elas. Iniciamos com as definições necessárias para fórmulas e valorações de fórmulas. Em seguida definimos alguns problemas computacionais relacionados a fórmulas e examinamos algumas reduções entre eles.

Dado um conjunto finito $V$ de variáveis, o conjunto dos literais de $V$ é o conjunto $L(V) = \cup_{v \in V}\{v, \neg v\}$. Aos literais $v \in V$ chamados de literais positivos e aos literais $\{\neg v \colon v \in V\}$ chamamos de literais negativos. As fórmulas sobre $V$ são definidas como segue.

- Cada literal é uma fórmula,
- Se $F_1, F_2, \ldots, F_n$ são fórmulas, então $F_1 \wedge F_2 \wedge \ldots \wedge F_n$ é uma fórmula que chamamos de conjunção e $F_1 \vee F_2 \vee \ldots \vee F_n$ é uma fórmula que chamamos de disjunção.

Seja $V$ um conjunto de variáveis. Uma valoração de $V$ é uma função $\tau \colon V \to \{0, 1\}$. O peso de uma valoração $\tau$ é a quantidade de 1's atribuídos às variáveis, ou seja, é a quantidade de elementos $v \in V$ tais que $\tau(v) = 1$.

Sejam $V$ um conjunto de variáveis, $F$ uma fórmula e $\tau$ uma valoração de $V$. Definimos a satisfatibilidade de $F$ a partir da valoração $\tau$ e estendendo $\tau$ do seguinte modo.

- $\tau(F) = min(\tau(F_1), \tau(F_2), \ldots, \tau(F_s))$ se $F = F_1 \wedge F_2 \wedge \ldots \wedge F_s$,
- $\tau(F) = max(\tau(F_1), \tau(F_2), \ldots, \tau(F_{s'}))$ se $F = F_1 \vee F_2 \vee \ldots \vee F_{s'}$ e
- $\tau(\neg F) = 1 - \tau(F)$.

Uma valoração $\tau$ satisfaz uma fórmula $F$ se e somente se $\tau(F) = 1$. Neste caso dizemos que $F$ é satisfeita por $\tau$ ou dizemos que $F$ é satisfatível.

Uma fórmula é monótona se ela não contém negações. Uma fórmula é antimonótona se ela não contém negações exceto para os literais que são todos negativos.

Consideremos a conjunção $F = F_1 \wedge F_2 \wedge \ldots \wedge F_s$ e a disjunção $F' = F_1 \vee F_2 \vee \ldots \vee F_{s'}$, onde cada $F_i$ é uma fórmula ou um literal. Usaremos os símbolos $\bigwedge$ e $\bigvee$ respectivamente para escrever $F$ na forma $F = \bigwedge_{i=1,\ldots,s} F_i$ e $F'$ na forma $F' = \bigvee_{i=1,\ldots,s'} F_i$.

Para exemplificar o uso da notação $\bigwedge$ consideremos as fórmulas $F' = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_1)$ e $F'' = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_4 \vee x_1)$. Fazemos $F_1 = x_1 \vee x_2$, $F_2 = x_2 \vee x_3$, $F_3 = x_3 \vee x_1$ e assim $F' = F_1 \wedge F_2 \wedge F_3$, o que escrevemos como sendo $F' = \bigwedge_{i=1,2,3} F_i$. Fazendo de modo similar para $F''$ e a escrevemos como sendo $F'' = \bigwedge_{i=1,\ldots,4} F_i$.

Informalmente, uma fórmula é $t$-normalizada se ela está na forma de conjunção-de-disjunções-de-conjunções- ... de-literais com $t$ alternâncias.

Seja $n$ um inteiro positivo e seja $F$ uma fórmula. $F$ está em $n$CNF se $F$ está em forma normal conjuntiva $F = F_1 \wedge F_2 \wedge \ldots \wedge F_s$ e cada $F_i$ é uma disjunção de no máximo $n$ literais, para $i = 1, \ldots, s$. $F$ está em $n$DNF se $F$ está em forma normal disjuntiva $F = F_1 \vee F_2 \vee \ldots \vee F_s$ e cada $F_i$ é uma conjunção de no máximo $n$ literais, para $i = 1, \ldots, s$.

Uma fórmula $F$ é 1-normalizada se $F$ está em $n$CNF ou em $n$DNF.

Uma fórmula $F$ é $t$-normalizada se $F$ é uma conjunção de fórmulas $(t-1)$-normalizadas ou se $F$ é uma disjunção de fórmulas $(t-1)$-normalizadas.

Um exemplo de fórmula 2-normalizada é uma fórmula em CNF e um exemplo de 3-normalizada é uma conjunção de disjunções de conjunções de literais.

## 4.1 Os problemas SAT e 3SAT

Consideremos os problemas computacionais **SAT** e **3SAT** como definidos abaixo.

| **SAT** |
|---|
| Instância: Uma fórmula $F$ em CNF. |
| Resposta: Há uma valoração que satisfaz $F$? |

| **3SAT** |
| --- |
| Instância: Uma fórmula $F$ em 3CNF. |
| Resposta: Há uma valoração que satisfaz $F$? |

O teorema abaixo consta em [Garey and Johnson(1979)] e é bastante conhecido.

**Teorema 4.1.** **SAT** *é redutível a* **3SAT**, *ou seja,* **SAT** $\propto$ **3SAT**.

A demonstração completa consta em [Garey and Johnson(1979)]. Uma parte importante para provar o Teorema 4.1 acima é tomar uma fórmula $C$ como instância de **SAT** e construir uma fórmula $C'$ para instância de **3SAT**, tal que $C$ é satisfatível se e somente se $C'$ é satisfatível.

Nessa construção de $C'$ a partir de $C$, para cada cláusula $c_j = z_1 \vee z_2 \vee \ldots \vee z_m$ de $C$ com $m > 3$ literais faz-se uma coleção de $m-2$ cláusulas com 3 literais cada para compor $C'$, introduzindo-se $U'_j = \left\{y_j^i \colon 1 \leq i \leq m-3\right\}$ com $m-3$ variáveis novas.

Consideremos os seguintes problemas parametrizados Sat e 3Sat, os quais tomamos parâmetros nos problemas computacionais **SAT** e **3SAT** do seguinte modo.

| Sat | |
| --- | --- |
| Instância: | Uma fórmula $F$ em CNF e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $F$? |

| 3Sat | |
| --- | --- |
| Instância: | Uma fórmula $F$ em 3CNF e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $F$? |

Usando a redução do Teorema 4.1 para tentar provar se Sat $\leq_{rp}$ 3Sat, percebe-se que os parâmetros não são redutíveis entre si. Suponha que a fórmula $C$ da demonstração do Teorema 4.1 tenha uma valoração de peso $k$ que a satisfaz, tal que exatamente uma variável $u_p$ na cláusula $C_j$ seja verdadeira. Daí, para $C'$ ser satisfatível é necessário fazer com que $y_j^1, \ldots, y_j^{p-2}$ sejam todas verdadeiras. Portanto, o peso da valoração que satisfaz

$C'$ não depende apenas do parâmetro $k$ da valoração de $C$, mas depende de $p$ e daí depende do tamanho das cláusulas de $C$.

Como vimos na Definição 3.4, o parâmetro de 3Sat deve depender apenas do parâmetro de Sat para que Sat $\leq_{rp}$ 3Sat. Disto e dos argumentos acima não podemos usar a redução do Teorema 4.1 para fazer com que Sat $\leq_{rp}$ 3Sat.

Não é conhecido se Sat $\leq_{rp}$ 3Sat por alguma redução parametrizada e conjectura-se em [Downey and Fellows(1999)] que não existe uma tal redução parametrizada.

## 4.2 Reduções parametrizadas

Vejamos alguns exemplos de reduções parametrizadas com relação a alguns problemas computacionais envolvendo fórmulas.

Para $n \geq 2$ inteiro consideremos o problema parametrizado $n$Sat como a restrição do problema Sat com fórmulas em $n$CNF. Para $n = 2$ temos o problema 2Sat.

2Sat

| | |
|---|---|
| Instância: | Uma fórmula $F$ em 2CNF e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $F$? |

Sejam os problemas parametrizados Sat-M e Sat-AntiM as restrições do problema Sat ao considerar, respectivamente, apenas fórmulas em CNF monótonas e antimonótonas. Sejam 2Sat-M e 2Sat-AntiM as restrições respectivamente de Sat-M e Sat-AntiM ao considerar apenas fórmulas em 2CNF.

No teorema abaixo não apresentamos a recíproca pois o raciocínio da redução é reversível, o que também ocorre em outros teoremas nos quais é necessário a recíproca e ela não for apresentada.

**Teorema 4.2.** Cobertura por Vértices $\cong$ 2Sat-M*, via redução parametrizada.*

*Demonstração.* Para demonstrar o teorema precisamos apresentar uma redução parametrizada de Cobertura por Vértices para o problema 2Sat-M e também uma

redução parametrizada do problema 2Sat-M para a Cobertura por Vértices, ou seja, é necessário provar que Cobertura por Vértices $\leq_{rp}$ 2Sat-M e também que 2Sat-M $\leq_{rp}$ Cobertura por Vértices.

Seja $(G, k)$ uma instância de Cobertura por Vértices. Vamos construir uma instância $(F, k')$ de 2Sat-M tal que o grafo $G$ tem cobertura por vértices de tamanho $k$ se e somente se a fórmula $F$ tem uma valoração de peso $k'$.

Para cada vértice do grafo $G$ criamos uma variável para a fórmula $F$. Para cada aresta $\{u, v\}$ do grafo criamos uma cláusula $(u \vee v)$. Daí, das cláusulas criadas a partir das arestas do grafo temos $F = (u \vee v) \wedge \ldots \wedge (u' \vee v')$. Para uma aresta $\{u, v\}$ o vértice $u$ ou o vértice $v$ devem fazer parte de uma cobertura por vértices, equivalentemente na cláusula $(u \vee v)$ a variável $u$ ou a variável $v$ precisa ser verdadeira para a cláusula ser satisfeita. Temos assim $k' = k$ e construímos $F$ em tempo polinomial, portanto é uma redução parametrizada. Como o raciocínio na redução acima é reversível então temos o teorema.

□

Consideremos o problema parametrizado de programação inteira binária PIB do seguinte modo.

PIB

| | |
|---|---|
| Instância: | Uma matriz binária $A$, um vetor binário $b$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há solução binária $x$ de peso $k$ para $A.x \geq b$? (sendo o peso de $x$ a quantidade de 1's em $x$.) |

No caso do exemplo abaixo a redução é uma redução parametrizada e vem de uma observação que consta na redução em [Karp(1972)].

**Teorema 4.3.** Sat-M $\cong$ PIB*, via redução parametrizada.*

*Demonstração.* Sejam $F = C_1 \wedge \ldots \wedge C_p$ uma fórmula monótona em CNF, $x_1, \ldots, x_m$ as variáveis de $F$ e $k$ um inteiro positivo. Seja $A$ a matriz $\{a_{i,j} \colon i = 1, \ldots, p, j = 1, \ldots, m\}$

com $a_{i,j} = 1$ se $x_j$ está presente na cláusula $C_i$, e $a_{i,j} = 0$ caso contrário. Seja $b$ o vetor com 1 na $j$-ésima posição para $j = 1, \ldots, p$.

Afirmamos que $A.x \geq b$ tem uma solução de peso $k$ se e somente se $F$ tem uma valoração de peso $k$ que a satisfaça.

Seja $x = (a_1, \ldots, a_m)$ uma solução de $A.x \geq b$ com peso $k$. Seja $S = \{i\colon\ a_i$ é o $i$-ésimo índice de $x$ e $a_i = 1\}$. Daí, $|S| = k$ já que $x$ tem peso $k$. Como $x$ é uma solução, então qualquer linha da matriz $A$ ao multiplicar por $x$ tem resultado maior ou igual a 1. Seja $l_s = (a_{s,1}, a_{s,2}, \ldots, a_{s,m})$ a linha $s$ da matriz $A$. Assim, existe $p \in S$ tal que o $p$-ésimo índice de $x$ é 1 e $a_{s,p}$ também é 1. Daí, pela redução, tem-se que a variável $x_p$ está na cláusula $C_s$. Portanto, atribuindo-se verdadeiro a $x_p$ satisfaz a cláusula $C_s$. Como toda linha $s$ de $A$ tem um índice $p$ com valor 1 tal que o $p$-ésimo índice de $x$ é 1, então tem-se uma valoração de peso $k$ que satisfaz $F$.

De outro modo, tendo-se uma valoração de peso $k$ que satisfaz $F$, temos que exatamente $k$ variáveis de $x_1, \ldots, x_m$ são satisfeitas. Fazendo-se um vetor $x = (a_1, \ldots, a_m)$ tendo valor 1 em $a_i$ se a variável $x_i$ foi satisfeita e valor 0 caso contrário, tem-se que o peso de $x$ é $k$. Daí, pela redução, a linha $p$ de $A$ multiplicada por $x$ tem resultado maior ou igual a 1, uma vez que na linha $p$ há valores 1 que representam a pertinência de variáveis na cláusula $C_p$ e como $F$ foi teve uma valoração de peso $k$ que a satisfez então a cláusula $C_p$ também teve. Daí, há $j$ na linha $p$ de $A$ com valor 1 que refere-se a variável $x_j$ satisfeita de $C_p$ e no vetor $x$ o índice $j$ tem valor 1 pela construção de $x$. O que vale para toda linha de $A$. Portanto há solução de peso $k$ para $A.x \geq b$

Vale notar que $k' = k$ e que a transformação da fórmula para a matriz é feita em tempo polinomial, portanto a redução é uma redução parametrizada. Como o raciocínio na redução acima é reversível então temos o teorema. □

# CAPÍTULO 5

# CIRCUITOS

Neste capítulo definimos os circuitos progredindo para os problemas parametrizados a partir de circuitos. Usaremos circuitos para definir vários problemas parametrizados como também para reduções entre problemas parametrizados envolvendo circuitos, grafos e fórmulas.

Seja $G$ um grafo direcionado $G = (V, E)$. Sejam $u, v$ vértices de $G$ e seja $(u, v)$ uma aresta de $G$.

Dizemos que a aresta $(u, v)$ sai do vértice $u$ chega no vértice $v$. O grau de entrada (de saída) do vértice $v$ de $V(G)$ é a quantidade de vértices que chegam em (que saem de) $v$. Denotamos por $d^+_G(v)$ e $d^-_G(v)$, respectivamente o grau de entrada e grau de saída do vértice $v$.

Chamamos de vértices fonte aos vértices com grau de entrada nulo e chamamos de vértices sorvedouro aos vértices com grau de saída nulo. Um vértice $v$ é não-fonte (não-sorvedouro) se $v$ não é um vértice fonte (sorvedouro).

Na Figura 5.1 representamos o grafo direcionado $G = (V, E)$ com vértices fonte $x_1, x_2, x_3$ e o sorvedouro sendo vértice $x_8$. Os rótulos atribuídos aos vértices fontes $x_1, x_2, x_3$ foram respectivamente 1, 1 e 0. O rótulo nas arestas $(u, v)$ tem o mesmo valor do rótulo do vértice $u$, exceto na aresta $(x_3, x_5)$ onde o rótulo é 1.

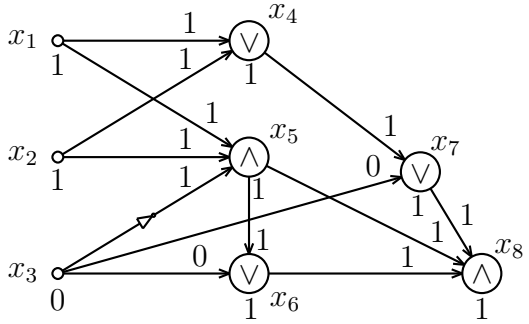


Figura 5.1: Um grafo direcionado com rótulos nas arestas e vértices

## 5.1 Circuitos

Um circuito é um grafo direcionado acíclico sem laços e sem arestas paralelas, com rótulos nos vértices e nas arestas, com um único vértice sorvedouro seguindo a nomenclatura abaixo.

Os vértices fonte são chamados de entradas do circuito e o vértice sorvedouro é chamado de saída do circuito. Os vértices não-fonte e não-sorvedouro são chamados de portas do circuito.

O rótulo nas entradas do circuito é o valor 1 ou 0 atribuído a elas. O rótulo na aresta $(u, v)$ é o valor do rótulo do vértice $u$ (aresta positiva) ou é o valor $1 - r$ (aresta negativa), onde $r$ é o rótulo do vértice $u$.

As portas do circuito são portas `e` ou portas `ou`, para as quais usamos respectivamente os símbolos $\wedge$ e $\vee$.

O rótulo de uma porta `e` é o menor valor dos rótulos das arestas que chegam nesta porta. O rótulo de uma porta `ou` é o maior valor dos rótulos das arestas que chegam nesta porta. O rótulo na saída do circuito é o mesmo do rótulo da aresta que chega ali.

Na Figura 5.2 representamos um circuito com rótulo 1 na saída $x_{12}$.

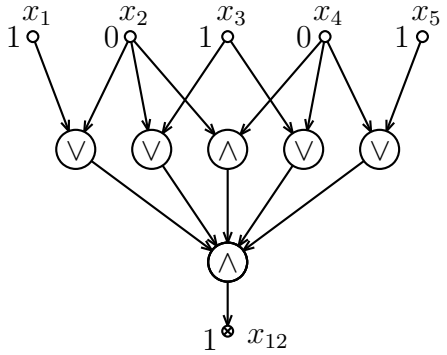


Figura 5.2: Um circuito com rótulo 1 na saída

Seja $C$ um circuito e seja $E$ o conjunto das entradas de $C$. Uma valoração de $C$ é uma função $\nu\colon E \to \{0, 1\}$ atribuindo rótulos aos vértices fonte. O peso de uma valoração $\nu$ é a quantidade de 1's atribuídos às entradas, ou seja, é a quantidade de elementos $v \in E$ tais que $\nu(v) = 1$. Uma valoração $\nu$ satisfaz um circuito $C$ quando o rótulo da saída do

circuito é 1, neste caso dizemos que $\nu(C) = 1$.

Se $\nu(C) = 1$ dizemos que $\nu$ é uma valoração que satisfaz $C$ ou que $C$ é satisfeito por $\nu$ ou dizemos que $C$ é satisfatível.

O circuito representado na Figura 5.2 é satisfatível pois com a valoração $\nu(x_1) = 1$, $\nu(x_2) = 0$, $\nu(x_3) = 1$, $\nu(x_4) = 0$, $\nu(x_5) = 1$ temos 1 no rótulo da saída do circuito.

Na Figura 5.3 representamos o circuito de entradas $x_1, x_2, x_3, x_4, x_5$ com a valoração $\nu(x_1) = 1$, $\nu(x_2) = 0$, $\nu(x_3) = 1$, $\nu(x_4) = 0$ e $\nu(x_5) = 1$. Assim, o peso da valoração $\nu$ é 3. Como o rótulo na saída do circuito é 1 então o circuito da Figura 5.3 é satisfeito pela valoração $\nu$.

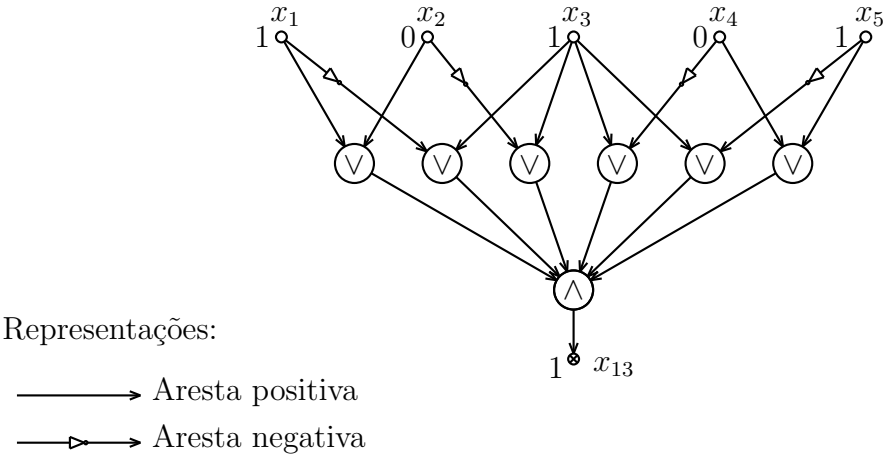


Figura 5.3: Um circuito satisfeito por uma valoração de peso 3

Consideremos os circuitos representados na figura abaixo.

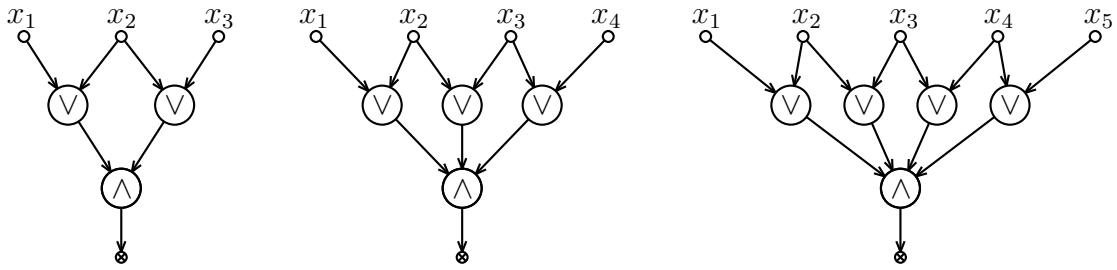


Figura 5.4: Circuitos com uma regra aparente na sua construção

Os circuitos representados na Figura 5.4 seguem uma regra: quaisquer duas entradas $x_i$ e $x_{i+1}$ de um circuito tem sua saída para uma porta *ou* e todas as saídas das portas *ou* vão para a entrada de uma porta *e*.

Seguindo essa regra podemos construir os circuitos para todo número $n$ de entradas, $n > 1$ inteiro positivo.

Seja $C$ um circuito com entradas $x_1, x_2, \ldots, x_n$ e que siga a regra da construção dos circuitos representados na Figura 5.4. Seja $x_j$ uma porta ou uma entrada de $C$.

Chamamos de *fan-in* (*fan-out*) de $x_j$ o grau de entrada (grau saída) de $x_j$.

Dessa forma, todas as portas `ou` em $C$ tem *fan-in* 2 e a porta `e` tem *fan-in* $n - 1$. Portanto o *fan-in* da porta `e` depende da quantidade de entradas do circuito.

Seja $\mathcal{F}$ o conjunto dos circuitos que sigam a regra representada na Figura 5.4 e seja $C$ um circuito qualquer em $\mathcal{F}$ com $n$ entradas $x_1, x_2, \ldots, x_n$.

Classificamos as portas de $C$ em duas denominações: chamamos de portas pequenas as portas que tem *fan-in* 2 e portas amplas as portas que não tem uma constante como limite no *fan-in*. Chamamos o conjunto $\mathcal{F}$ de família de circuitos.

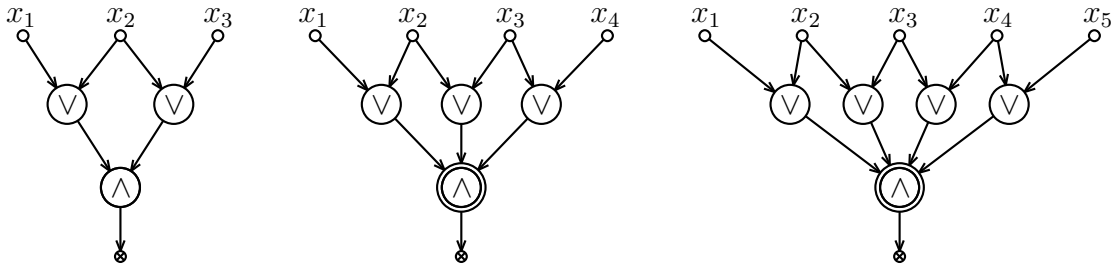


Figura 5.5: Circuitos de uma família, com portas pequenas e portas amplas

Os circuitos representados na Figura 5.5 pertencem a família de circuitos $\mathcal{F}$ onde temos um circuito $C_n$ para cada $n > 1$ inteiro positivo, ou seja, $\mathcal{F} = \{C_2, C_3, \ldots, C_n, \ldots\}$.

Usando outras regras temos outras famílias $\mathcal{F}$.

Seja $\mathcal{F} = \{C_1, \ldots, C_n, \ldots\}$ uma família qualquer de circuitos tal que $C_n$ tem $n$ entradas, para $n$ inteiro positivo. Sejam $C_i$ e $C_j$ dois circuitos quaisquer de $\mathcal{F}$. Uma porta dos circuitos $C_i$ e $C_j$ é uma porta pequena se o *fan-in* é limitado por alguma constante, por exemplo 2. Uma porta é ampla nos circuitos $C_i$ e $C_j$ se o *fan-in* não é limitado por uma constante.

No restante do texto chamamos uma porta de qualquer $C_j \in \mathcal{F}$ de porta pequena se existe uma constante como limite para seu *fan-in* e que não depende do número de

entradas do circuito.

Na Figura 5.6 representamos o circuito $C_7$ de uma família $\mathcal{F}$ com entradas $x_1, \ldots, x_7$, com uma porta ampla `e` e com seis portas pequenas `ou` de *fan-in* 3.

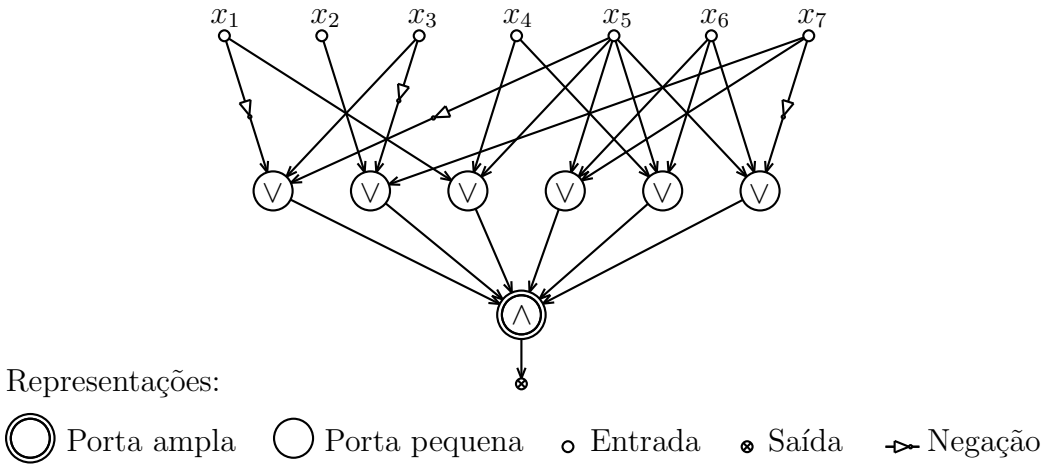


Figura 5.6: Um circuito com seis portas pequenas

O *fan-out* das portas e entradas de um circuito pode ser limitado ou ilimitado. No que segue um circuito $C$ é sempre um circuito $C_i$ qualquer de uma família $\mathcal{F} = \{C_1, \ldots, C_n, \ldots\}$ qualquer de circuitos. O *fan-in* de um circuito $C$ é o número máximo do *fan-in* das portas pequenas de $C$.

A profundidade de um circuito $C$ é o número máximo de portas no caminho de uma entrada à saída de $C$. A trama (do inglês *weft*) do circuito $C$ é a "profundidade de portas amplas". Mais precisamente temos a definição a seguir.

**Definição 5.1.** *Seja $C$ um circuito. A trama de $C$ é o número máximo de portas amplas no caminho de uma entrada à saída de $C$.*

O circuito representado na Figura 5.6 tem trama 1, profundidade 2 e com portas pequenas tendo *fan-in* limitado em 3. Na Figura 5.7 temos a representação de um circuito de trama 2, profundidade 4 e com portas pequenas tendo *fan-in* limitado em 2.

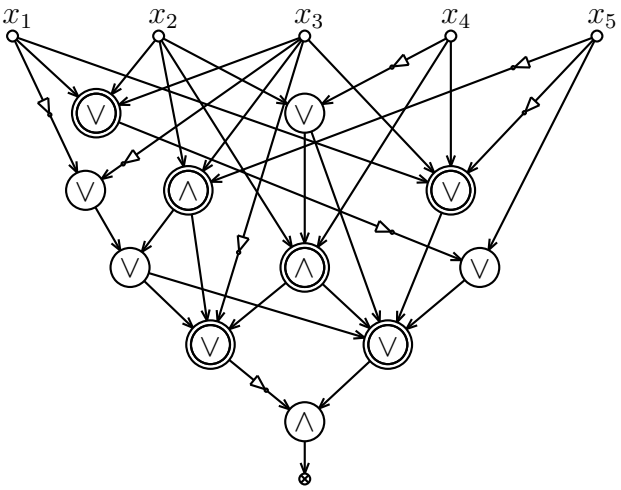


Figura 5.7: Um circuito de trama 2 e profundidade 4

Criamos os circuitos representados na Figura 5.4 a partir de uma regra para uma família de circuitos $\mathcal{F}$. A seguir mostramos como criar uma família de circuitos $\mathcal{F}$ para representar o problema de Cobertura por Vértices.

Consideremos o grafo $G$ com 5 vértices representado à esquerda na Figura 5.8. Vamos construir um circuito $C_5$ a partir de $G$ tal que $C_5$ é satisfeito por uma valoração de peso $k$ se e somente se o grafo $G$ tem uma cobertura por vértices de tamanho $k$.

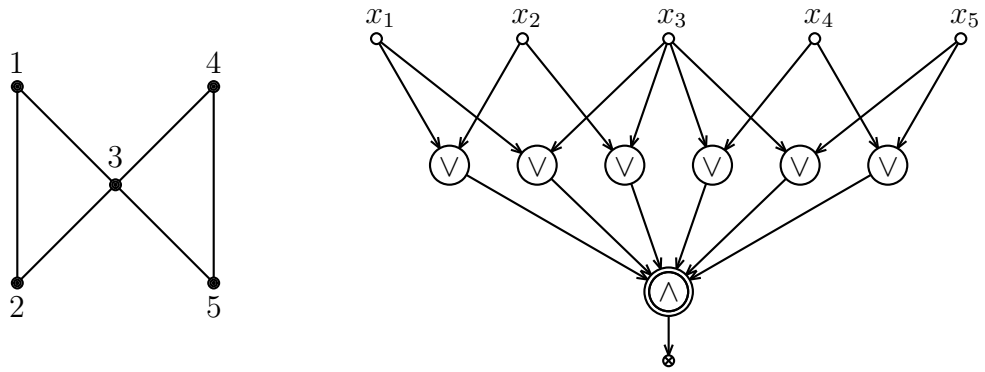


Figura 5.8: Um circuito a partir de um grafo para a cobertura por vértices

Para cada vértice $i$ do grafo $G$ criamos uma entrada $x_i$ no circuito $C_5$. Para cada aresta $\{i, j\}$ do grafo criamos uma porta pequena `ou` recebendo a saída das entradas $x_i$ e $x_j$ do circuito. As saídas de todas as portas pequenas vão para uma única porta `e` ampla e esta vai ao sorvedouro. Exemplificamos este circuito $C_5$ à direita da Figura 5.8, onde o circuito $C_5$ tem trama 1 profundidade 2.

Vejamos como $C_5$ é satisfeito por uma valoração $\nu$ de peso $k$ se e somente se o grafo $G = (V, E)$ tem uma cobertura por vértices de tamanho $k$. Para uma aresta $\{i, j\}$ o vértice $i$ ou o vértice $j$ deve fazer parte de uma cobertura por vértices, equivalentemente a valoração em $x_i$ e $x_j$ precisa ser $\nu(x_i) = 1$ ou de $\nu(x_j) = 1$ para a porta pequena $\vee$ correspondente ter saída 1. Portanto, $C_5$ é satisfeito por uma valoração de peso $k$ se e somente se o grafo $G = (V, E)$ tem uma cobertura por vértices de tamanho $k$.

## 5.2 O problema parametrizado Circuit-Sat

O problema computacional de satisfazer circuitos damos na forma parametrizada como a seguir, com o parâmetro no peso da valoração.

| Circuit-Sat | |
|---|---|
| Instância: | Um circuito $C$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $C$? |

Restringindo os circuitos para o problema Circuit-Sat em relação à profundidade e à trama definimos os problemas parametrizados da satisfatibilidade de circuitos de trama $t$ e profundidade $h$ denominado de WCS$(t, h)$ (do inglês *Weighted weft t depth h Circuit Satisfatibility*).

| WCS$(t, h)$ | |
|---|---|
| Instância: | Um circuito $C$ de trama $t$ e profundidade $h$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $C$? |

Pelo Teorema 4.2 mostramos que Cobertura por Vértices $\cong$ 2Sat-M, via redução parametrizada. Vejamos uma relação entre Cobertura por Vértices e WCS$(1, 2)$.

**Teorema 5.2.** Cobertura por Vértices $\leq_{rp}$ WCS$(1, 2)$, *via redução parametrizada.*

*Demonstração.* Usando a redução exemplificada na Figura 5.8 tomamos uma instância $(G, k)$ de Cobertura por Vértices e criamos um circuito a partir de $G$ tal que o

circuito é satisfeito por uma valoração de peso $k$ se e somente se o grafo $G$ tem uma cobertura por vértices de tamanho $k$. Com isso $k' = k$

A redução é feita em tempo polinomial, $k'$ depende apenas de $k$ e portanto é uma redução parametrizada.

□

De modo bastante direto temos o seguinte fato.

**Teorema 5.3.** SAT $\leq_{rp}$ WCS$(2, 2)$, *via redução parametrizada.*

*Demonstração.* Seja $F$ uma instância de SAT. Vamos criar uma instância $C$ de WCS$(2, 2)$ a partir de $F$. De cada variável de $F$ fazemos uma entrada para $C$. Para cada cláusula de $F$ criamos uma porta ampla `ou` recebendo as saídas (com ou sem negação) das respectivas entradas. Todas as saídas dessas portas `ou` fazemos ser entradas de uma porta ampla `e`, que vai à saída de $C$. Daí, $F$ tem valoração de peso $k$ se e somente se o circuito $C$ é satisfeito por uma valoração de peso $k$. Essa redução é polinomial, $k' = k$ e portanto é uma redução parametrizada. □

**Corolário 5.4.** 2SAT $\leq_{rp}$ WCS$(1, 2)$, *via redução parametrizada.*

Consideremos a Figura 5.9 a seguir. Partimos do grafo à esquerda para construir o circuito à direita, tal que o grafo tem conjunto dominante de tamanho $k$ se e somente se o circuito é satisfeito por uma valoração de peso $k$. Para cada vértice do grafo criamos uma entrada $x_i$ no circuito e criamos uma porta ampla $\vee$ para $x_i$. Para cada porta ampla $\vee$ de $x_i$ existe uma aresta saindo da entrada $x_i$ e das entradas cujos vértices correspondentes sejam vizinhos do vértice $i$. Todas as portas amplas $\vee$ tem uma saída para a entrada de uma única porta ampla $\wedge$ que vai ao sorvedouro.

Daí se o grafo tem um conjunto dominante $V'$ de tamanho $k$ significa que qualquer vértice do grafo tem um vizinho em $V'$ ou está em $V'$. Então o circuito tem valoração de peso $k$. Se o circuito tem valoração de peso $k$ então todas as portas amplas $\vee$ tem saída 1, ou seja, em cada porta ampla $\vee$ ao menos uma das entradas tem valor 1. Portanto o grafo tem um conjunto dominante de tamanho $k$.

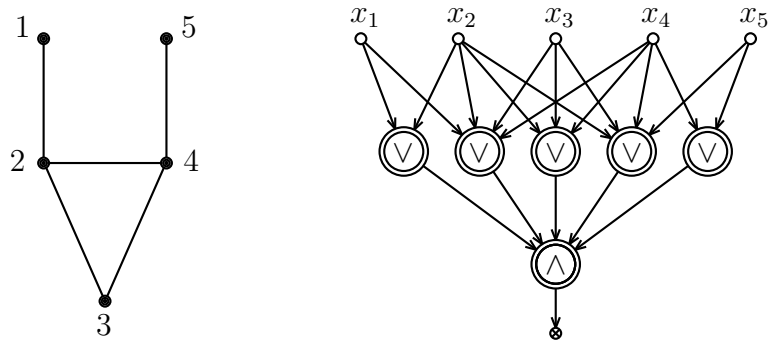


Figura 5.9: Um circuito a partir de um grafo para o conjunto dominante

A redução exemplificada na Figura 5.9 é feita em tempo polinomial com $k' = k$ e é reversível, a generalização da construção prova o seguinte teorema.

**Teorema 5.5.** Conjunto Dominante $\cong$ WCS$(2, 2)$, *via redução parametrizada.*

Ainda neste texto apresentamos outros problemas parametrizados redutíveis a WCS$(t, h)$.

# CAPÍTULO 6

# A CLASSE $FPT$ DOS PROBLEMAS COMPUTACIONAIS TRATÁVEIS POR PARÂMETRO FIXO

Apresentamos neste capítulo as definições para a classe $FPT$ dos problemas computacionais tratáveis por parâmetro fixo. Também apresentamos alguns métodos para que um problema parametrizado possa ser classificado como $FPT$. Ainda, damos vários exemplos de problemas parametrizados pertencentes a classe $FPT$ com reduções parametrizadas para outros problemas.

Ao analisar um problema parametrizado tentamos descrever o tempo de execução de uma solução para o problema computacional em relação ao tamanho da instância e ao parâmetro. Usar um parâmetro para um problema computacional é também localizar qual ou quais são os aspectos da instância do problema computacional que realmente causam dificuldade para resolver esse problema.

A redução parametrizada dada na Definição 3.4 não demanda um limite superior em relação a redução do parâmetro de um problema ao parâmetro do outro. Podemos expressar o tempo de execução de uma solução para um problema parametrizado sem impor um limite superior na relação com o parâmetro do seguinte modo.

**Definição 6.1.** *Um problema parametrizado $P$ é tratável por parâmetro fixo se pode ser resolvido em tempo $f(k)|x|^{\alpha}$, para algum $\alpha > 0$, onde $(x, k)$ é uma instância de $P$ e a função $f$ depende apenas do parâmetro $k$.*

A Definição 6.1 é uma das formas de definir um problema tratável por parâmetro fixo, em [Downey and Fellows(1999)] são apresentadas outras três formas buscando análises ainda mais refinadas. Essas outras três formas demandam definições que não cabem neste texto introdutório.

Denotamos por $FPT$ (do inglês *fixed parameter tractable*) a classe dos problemas

tratáveis por parâmetro fixo.

O problema parametrizado mais famoso da classe $FPT$ é Cobertura por Vértices. Este problema também é exemplo do quanto podemos apresentar melhores algoritmos. Como foi observado em [Fellows and Langston(1987)] a Cobertura por Vértices podia ser resolvida com tempo de execução $O(f(k)n^3)$ e atualmente temos o tempo de execução $O(kn + 1,2738^k)$ por [Chen et al.(2006)Chen, Kanj, and Xia]. Várias melhorias ocorreram nesse intervalo. Algumas são as seguintes.

- Em 1987 foi resolvido em tempo $O(f(k)n^3)$ por [Fellows and Langston(1987)], com $f(k)$ ainda não explícito.
- Em 1987 foi resolvido em tempo $O(f(k)n^2)$ por [Johnson(1987)], com $f(k)$ ainda não explícito.
- Em 1988 foi resolvido em tempo $O(2^k n)$ por [Fellows(1988)].
- Em 1993 foi resolvido em tempo $O(kn + 2^k k^{2k+2})$ por [Buss and Goldsmith(1993)].
- Em 1994 foi resolvido em tempo $O(kn + 2^k k^2)$ por [Downey and Fellows(1994)].
- Em 1998 foi resolvido em tempo $O(kn + (4/3)^k k^2)$.
- Em 2001 foi resolvido em tempo $O(kn + 1,286^k k)$.
- Em 2005 foi resolvido em tempo $O(kn + 1,2745^k k^4)$.
- Em 2006 foi resolvido em tempo $O(kn + 1,2738^k)$.

Omitimos algumas referências acima para facilitar a leitura. As referências dos resultados do ano 1998 é [Balasubramanian et al.(1998)Balasubramanian, Fellows, and Raman], do ano de 2001 é [Chen et al.(2001)Chen, Kanj, and Jia], a referência para o ano de 2005 é [Chandran and Grandoni(2005)] e de 2006 é [Chen et al.(2006)Chen, Kanj, and Xia].

Em muitos casos temos algoritmos com tempo $O(f(k) + n^c)$, isto é, a parte exponencial está em um termo aditivo, ao invés de multiplicativo como na Definição 6.1 para

$FPT$. Vejamos como isto não define uma classe diferente de $FPT$: seja $P$ um problema parametrizado resolvido por um algoritmo $\mathcal{A}$ em $f(k)n^c$ passos. Se $f(k) \leq n$ o algoritmo demanda no máximo $n^{c+1}$ passos, e se $f(k) > n$ o algoritmo executa em menos do que $(f(k))^{c+1}$ passos. Em ambos estes os casos, $\mathcal{A}$ resolve $P$ em no máximo $g(k) + n^{c'}$ passos, onde $g(k) = (f(k))^{c+1}$ e $c' = c + 1$. Portanto o tempo de execução $O(f(k) + n^c)$ é compatível com $FPT$.

Entretanto, nas aplicações é mais bem-vindo um algoritmo com tempo de execução $O(f(k) + n^c)$ do que um algoritmo com tempo de execução $O(f(k)n^c)$.

A seguir apresentamos alguns métodos para classificar um problema parametrizado como $FPT$ e também apresentamos alguns problemas parametrizados fazendo uso dos métodos.

## 6.1 As árvores de busca limitadas

Existem várias técnicas para classificar um problema parametrizado como tratável por parâmetro fixo, apresentamos algumas delas iniciando pelo método das árvores de busca limitadas (do inglês *bounded search trees*).

Para classificar o problema Cobertura por Vértices como $FPT$ podemos fazer da seguinte maneira.

1. Tomamos uma instância $(G, k)$, onde $k > 0$ e o grafo $G$ tem $n$ vértices (se $k$ fosse nulo seria suficiente verificar se existe uma aresta no grafo).

2. Escolhemos uma aresta $\{v, w\}$ desse grafo.

3. Como todas as arestas precisam ser cobertas então $v$ ou $w$ deve pertencer a uma cobertura por vértices.

4. Criamos uma árvore binária tal que cada filho tem a escolha de $v$ ou $w$, assim cada filho tem um conjunto de vértices $S$ para formar uma cobertura com $k$ vértices.

5. Em cada filho escolhemos uma aresta $\{v', w'\}$ tal que nem $v'$ nem $w'$ pertença $S$ criando dois filhos e repetindo o processo até formar uma cobertura por vértices de tamanho $k$,

6. Não é necessário descer mais do que $k$ níveis nesta árvore e podemos fazer isto em tempo $O(2^k n)$.

Em poucas palavras, construímos uma árvore binária de altura no máximo $k$. Em seguida fazemos um busca nessa árvore, por exemplo em profundidade. Isso prova o teorema a seguir.

**Teorema 6.2.** *O problema* Cobertura por Vértices *pode ser resolvido em tempo* $O(2^k|V(G)|)$.

A argumentação usada acima para demonstrar o Teorema 6.2 tem sua essência no chamado método das árvores de busca limitadas. Esse método aparece nas soluções de muitos problemas computacionais combinatórios com algoritmos em basicamente duas partes, dadas abaixo.

1. Computamos dentro do algoritmo algum espaço de busca, que comumente é uma árvore de busca de tamanho exponencial.

2. Em cada ramificação da árvore executamos algum algoritmo relativamente eficiente, frequentemente baseado numa forma simples como uma busca em profundidade.

A complexidade exponencial do pior caso desses algoritmos vem das instâncias onde é necessário visitar toda a árvore. Em Complexidade Parametrizada foram observados muitos problemas parametrizados onde o tamanho da árvore depende unicamente do parâmetro. Daí, fixado o parâmetro, o espaço de busca fica constante e o algoritmo é eficiente.

Consideremos o seguinte problema parametrizado.

| Log-Cobertura | |
|---|---|
| Instância: | Um grafo $G$ e um inteiro positivo $k = \log(\|G\|)$. |
| Parâmetro: | O inteiro positivo $k = \log(\|G\|)$. |
| Pergunta: | $G$ tem uma cobertura por vértices de tamanho $k$? |

**Corolário 6.3.** *O problema* Log-Cobertura $\in \mathcal{P}$ *pois pode ser resolvido em tempo* $O(|G|^2)$.

O problema Log-Cobertura tem de antemão a característica do parâmetro ser $k = \log(|G|)$. Em algumas aplicações podemos ter uma ou outra característica prevista para usar como parâmetro, por exemplo ter apenas grafos planares nas instâncias como no caso do teorema a seguir, onde usamos o método das árvores de busca limitadas.

**Teorema 6.4.** *O problema* Conjunto Independente *restrito a grafos planares pode ser resolvido em tempo* $O(6^k n^2)$.

*Demonstração.* Seja $(G, k)$ uma instância tal que $G$ é um grafo planar. Como $G$ é planar então existe um vértice $w$ de grau no máximo 5, que pode ser encontrado em tempo $O(n^2)$. Sejam $S \subseteq V(G)$ um conjunto independente maximal em $G$ e $v$ um vértice qualquer de $G$. Assim $N[v] \cap S \neq \emptyset$, pois de outro modo $S \cup \{v\}$ seria conjunto independente em $G$, o que contradiz a maximalidade de $S$. Portanto ao menos um dos no máximo 6 vértices de $N[w]$ pertence a um conjunto independente maximal de $G$. Assim construímos uma árvore de raiz $\{G, X = \emptyset\}$ e com no máximo 6 filhos, um para cada $u \in N[w]$. Em cada filho $\{G', S'\}$ temos $S' = X + u$ para formar um conjunto independente maximal $S$ contendo $S'$. $G'$ é o subgrafo $G - N[u]$ que continua sendo planar, então o raciocínio usado continua válido em $G'$ e prosseguimos a busca nos filhos. Não é necessário descer nessa árvore além da profundidade $k$. □

No problema Cobertura por Vértices questionamos o tamanho do conjunto onde "vértices cobrem arestas." Um problema relacionado e bem conhecido é o problema Conjunto Dominante, onde "vértices cobrem vértices." O problema do conjunto dominante

pode ser parametrizado tomando o tamanho do conjunto solução como parâmetro e daí temos o seguinte problema parametrizado.

| Conjunto Dominante | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Existe em $G$ um conjunto dominante de tamanho $k$? |

Consideremos o problema Conjunto Dominante Planar como sendo Conjunto Dominante restrito aos grafos planares. Daí, usando o método das árvores de busca limitadas e o lema a seguir temos Conjunto Dominante Planar como $FPT$.

A definição de face também acompanhamos do livro [Bondy and Murty(2008)].

**Lema 6.5.** *Seja $G = (V, E)$ um grafo simples planar com uma partição de vértices em dois conjuntos $V = V_1 \cup V_2$ satisfazendo (i) o grau mínimo dos vértices em $V_1$ é ao menos 3 e (ii) se $V_1$ é um conjunto independente em $G$. Existe um vértice $u \in V_2$ de grau no máximo 10 em $G$.*

*Demonstração.* Seja $G$ com o número mínimo possível de vértices e o máximo possível de arestas nas condições do enunciado. Seja $H$ o subgrafo de $G$ induzido por $V_2$, daí $H$ também é planar. Em qualquer face de $H$ pode haver no máximo um vértice de $V_1$, do contrário uma aresta poderia ser adicionada entre dois vértices de $V_2$ nas bordas da face e portanto $G$ não teria o máximo de arestas como suposto. Seja $u$ um vértice de grau no máximo 5 em $H$. Então $u$ tem grau no máximo 10 em $G$. □

A prova do teorema a seguir é feita usando o Lema 6.5 para construir uma árvore de busca onde, para cada nó, temos no máximo 11 filhos.

**Teorema 6.6** ([Downey and Fellows(1994)])**.** Conjunto Dominante Planar *pode ser resolvido em tempo $O(11^k|G|)$.*

No início deste capítulo apresentamos algumas das várias melhorias do tempo de execução para Cobertura por Vértices. Para Conjunto Dominante Planar pelo Teorema 6.6 temos o tempo de execução $O(11^k|G|)$, uma melhoria possível é a seguinte.

**Teorema 6.7.** Conjunto Dominante Planar *pode ser resolvido em tempo* $O(8^k|G|)$ *de [Alber et al.(2003)Alber, Fernau, Niedermeier, Fan, Fellows, Rosamond, and Stege].*

Outro problema bastante conhecido é o problema do conjunto de vértices de realimentação.

Conjunto de Vértices de Realimentação

| | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Existe um conjunto de vértices $V' \subseteq V$ de cardinalidade no máximo $k$ tal que $G - V'$ seja acíclico? |

Usando o método das árvores de busca em [Downey and Fellows(1994)] temos o resultado a seguir.

**Teorema 6.8.** Conjunto de Vértices de Realimentação *pode ser resolvido em tempo* $O((2k+1)^k.n^2)$.

Como no caso de outros problemas $FPT$, também ocorreram muitas melhorias para o tempo execução do problema Conjunto de Vértices de Realimentação chegando ao tempo $O(3,83^k kn^2)$ em [Cao et al.(2010)Cao, Chen, and Liu], onde também consta uma tabela comparativa entre várias melhorias.

Para provar o teorema a seguir criamos uma árvore binária como no início do capítulo.

**Teorema 6.9.** 2Sat-M *é* $FPT$.

*Demonstração.* Seja $(F, k)$ uma instância de 2Sat-M. Assim $F$ é uma fórmula monótona, isto é, com literais todos positivos. Para cada cláusula $C = (u \vee v)$ de $F$ criamos dois filhos, um representando a escolha da variável $u$ para ser verdadeira e outro filho para a variável $v$ ser verdadeira. Ao escolher uma variável pra ser verdadeira podemos eliminar outras cláusulas da fórmula. Se ainda houver mais cláusulas na fórmula então ramificamos tomando outra cláusula e repetindo o processo criando dois filhos. Não é necessário descer à profundidade maior que $k$ nessa árvore. Fazemos isto em tempo $O(2^k n)$. Portanto 2Sat-M é $FPT$. □

## 6.2 A "Kernelização" - redução ao núcleo do problema

A Kernelização é outra técnica para classificar um problema parametrizado como $FPT$ e explicar alguns bons resultados de pré-processamento nas aplicações.

Nas aplicações é bastante comum haver um pré-processamento ou uma "redução" dos dados da instância. Em vários casos o impacto do pré-processamento impressiona, como é o caso do trabalho em [Weihe(1998)] com trens e estações usando um grafo bipartido em vértices para as estações e vértices para os trens tendo uma aresta quando o trem faz uma parada na estação. O problema em questão é o de encontrar o mínimo de estações tal que todo trem faz uma parada em ao menos uma das estações. O problema de decisão associado é $\mathcal{NP}$-Completo. Os grafos do trabalho de [Weihe(1998)] são da ordem de 10.000 vértices. Ao aplicar um pré-processamento foi o suficiente para reduzir os grafos do trabalho em componentes com no máximo 29 vértices e daí podendo ser resolvido rapidamente.

Informalmente o método da kernelização (do inglês *kernelization*) ou redução ao núcleo consiste em diminuir uma instância $I$ de um problema parametrizado para uma instância "equivalente" $I'$, onde o tamanho de $I'$ é limitado em função do parâmetro e a diminuição é feita em tempo polinomial. Em seguida analisamos a instância $I'$, exaustivamente se necessário. A solução de $I'$ é então usada na solução de $I$. Ao fazer tudo isso o tempo de execução é o tempo da diminuição mais o tempo da solução de $I'$, sendo que este é limitado em função do parâmetro. Usualmente esta técnica nos leva um fator aditivo a $f(k)$ ao invés de multiplicativo como na Definição 6.1.

Vejamos como "encolher" uma instância de Cobertura por Vértices. Seja $(G, k)$ uma instância de Cobertura por Vértices, daí qualquer vértice de grau maior que $k$ deve pertencer a toda cobertura por vértices de $G$ que tenha tamanho $k$. Então removemos de $G$ todos os vértices de grau maior que $k$, guardando-os em um conjunto $P$. O grafo resultante deve ter no máximo $(k - |P|)(k + 1)$ vértices para que o grafo original tenha uma cobertura por vértices de tamanho $k$, neste caso podemos exaustivamente procurar uma cobertura por vértices de tamanho $k - |P|$ no grafo resultante. O tempo de execução

desta busca ficará em função do parâmetro $k$ nesse núcleo de no máximo $(k - |P|)(k + 1)$ vértices. Este algoritmo de redução removendo vértices é de [Buss and Goldsmith(1993)].

Removemos do grafo $G$ todos os $p$ vértices de grau maior do que $k$. Se o grafo resultante tiver no máximo $(k-p)(k+1)$ vértices então procuraremos nele os demais $k-p$ vértices. A observação é que no grafo resultante o número máximo de vértices depende do parâmetro $k$.

O limite $(k-p)(k+1)$ citado ocorre pois removemos todos os $p$ vértices de grau maior do que $k$. O grafo resultante tem o grau de seus vértices limitado em $k$. Se este grafo tiver uma cobertura de vértices com tamanho $k - p$, cada vértice tendo no máximo $k$ vizinhos, então temos no máximo $(k - p).k$ vértices vizinhos neste grafo resultante. Somando os $(k - p).k$ vértices vizinhos com os $k - p$ vértices da cobertura temos $(k - p).k + (k - p)$, que é o mesmo que $(k - p)(k + 1)$.

**Teorema 6.10.** Cobertura por Vértices *pode ser resolvido em tempo* $O(kn + k^{4k})$.

*Demonstração.* Seja $(G, k)$ uma instância de Cobertura por Vértices. Fazemos a redução acima no grafo $G$ em tempo $O(kn)$. O grafo resultante tem no máximo $k(k + 1)$ vértices como vimos e daí uma busca completa poderia requerer $\binom{k(k+1)}{k}$ passos, o que pode ser feito em tempo $O(k^{4k})$.

□

Podemos definir a técnica de kernelização da seguinte forma.

**Definição 6.11.** *Seja $P$ um problema parametrizado. Kernelização é uma transformação polinomial de uma instância $(x, k)$ em uma instância $(x', k')$ tal que*

- $(x, k) \in Y_P$ *se e somente se* $(x', k') \in Y_P$,
- $k' \leq k$ *e*
- $|x'| \leq \eta(k)$ *para alguma função* $\eta$ *dependendo apenas de* $k$.

Da Definição 6.11 chamamos $x'$ de núcleo do problema (do inglês *problem kernel*) e chamamos $\eta(k)$ de tamanho do núcleo do problema. Dizemos que o tamanho do núcleo é polinomial se a função $\eta(k)$ for limitada polinomialmente.

O problema a seguir é $FPT$ cuja prova pode ser feita usando kernelização.

Folhas na Árvore Geradora

| | |
|---|---|
| Instância: | Um grafo $G$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Existe uma árvore geradora de $G$ com $k$ folhas? |

O problema de decisão associado ao problema Folhas na Árvore Geradora é $\mathcal{NP}$-Completo.

A demonstração do Teorema abaixo é por kernelização ocorrendo um pré-processamento de $(G, k)$ para chegar ao núcleo do problema. No grafo $G$ pode haver vértices $v$ de grau 2 e seus vizinhos $u$ e $w$ também de grau 2, representado no lado esquerdo da Figura 6.1. O início do algoritmo ocorre a verificação se o grafo $G$ é conexo, pois apenas grafos conexos podem ter uma árvore geradora.

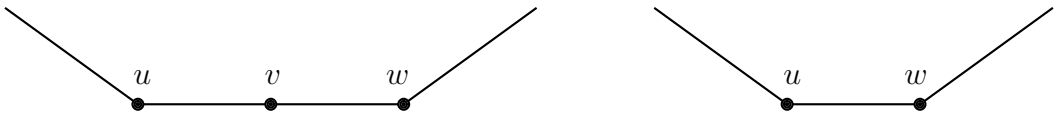


Figura 6.1: Remoção de vértices descartáveis

Chamamos os vértices $v$ de vértices descartáveis.

**Teorema 6.12** ([Downey and Fellows(1999)])**.** *O problema parametrizado* Folhas na Árvore Geradora *pode ser resolvido em tempo* $O(n + (2k)^{4k})$.

O argumento da prova usa o pré-processamento a seguir e o algoritmo abaixo. No pré-processamento de $G$ são removidos os vértices descartáveis $v$ adicionando uma aresta entre seus dois vizinhos. O resultado do pré-processamento é o grafo $G'$ sem vértices descartáveis.

O algoritmo faz o seguinte.

**Passo 1.** Se $G$ não é conexo então a instância $(G, k)$ não é positiva.

**Passo 2.** Se $G$ tem um vértice de grau ao menos $k$ então a instância $(G, k)$ é positiva.

**Passo 3.** Se $G$ é conexo e o grau máximo é menor do que $k$ então aplique o pré-processamento obtendo $G'$. Se $|V(G')| < 4(k+2)(k+1)$ então $G'$ é o núcleo do problema de tamanho, senão a instância $(G, k)$ é positiva.

Tem-se o Teorema 6.12 acima pois mesmo que seja necessário buscar exaustivamente por uma árvore geradora em $G'$ isto pode ser feito em tempo $O((2k)^{4k})$. A prova completa consta em [Downey and Fellows(1999)].

O método da kernelização pode ser usado em conjunto com o das árvores de busca limitadas para melhores resultados e análises mais refinadas.

## 6.3 Um problema é $FPT$ se e somente se ele tem kernelização

O resultado a seguir é bastante importante, com $P$ sendo um problema parametrizado.

**Teorema 6.13.** *$P \in FPT$ se e somente se $P$ tem kernelização.*

*Demonstração.* Seja $P$ um problema parametrizado e seja $(x, k)$ uma instância de $P$. Se $P$ tem kernelização então por definição existe $(x', k')$ computado em tempo polinomial a partir de $(x, k)$ e por definição $|x'| \leq g(k)$. Assim, resolver $(x', k')$ por força bruta é feito em tempo $f(k)$ para alguma função $f$ dependendo apenas de $k$. Portanto $(x, k)$ foi resolvido em tempo $f(k)|x|^{\alpha}$ para alguma constante $\alpha > 0$. Logo $P \in FPT$.

Se $P$ é um problema parametrizado tratável por parâmetro fixo, isto é, se $P \in FPT$, então existem uma constante $\alpha$ e um algoritmo que determinam se $(x, k)$ pertence a $Y_P$ em tempo $f(k)|x|^{\alpha}$. Se $f(k) \leq |x|$ então a instância é resolvida em tempo $f(k)|x|^{\alpha} \leq |x|^{\alpha+1}$. Se $f(k) > |x|$ então $f(k)$ é o núcleo do problema e portanto $P$ tem kernelização.

□

O resultado do Teorema 6.13 pode ser visto como uma forma de caracterizar um

problema parametrizado, pois a kernelização não é necessariamente a melhor ferramenta para o desenvolvimento de algoritmos. Exemplo disto é o tempo de execução $O(kn + 1,2738^k)$ por [Chen et al.(2006)Chen, Kanj, and Xia].

## 6.4 Reduções entre problemas parametrizados da classe $FPT$

Vimos no Teorema 3.1 que se $P_1$ e $P_2$ são problemas tal que existe uma redução de $P_1$ para $P_2$, se $P_2 \in \mathcal{P}$ então $P_1 \in \mathcal{P}$. De modo semelhante temos alguns resultados fazendo o uso de reduções parametrizadas com problemas $FPT$.

**Teorema 6.14** ([Downey and Fellows(1992)])**.** *Se o problema parametrizado $P$ é redutível ao problema parametrizado $P' \in FPT$ por uma redução parametrizada então $P \in FPT$, ou seja,*

$$\textit{se } P \leq_{rp} P' \textit{ e } P' \in FPT, \textit{ então } P \in FPT.$$

*Demonstração.* Seja $g(k)|x|^c$ o tempo de execução da redução da instância $(x, k)$ de $P$ para a instância $(x', k')$ de $P'$ por uma redução parametrizada tal que $(x, k) \in P$ se e somente se $(x', k') \in P'$. Como $P' \in FPT$ então $P'$ pode ser resolvido em tempo $f(k')|x'|^\alpha$, para algum $\alpha \in \mathbb{N}$ e com $f$ dependendo apenas do parâmetro $k'$. Portanto $P \in FPT$. □

Como Cobertura por Vértices $\leq_{rp}$ 2Sat-M pelo Teorema 4.2 então

2Sat-M é FPT

como consequência imediata do Teorema 6.14.

Existe casos onde os algoritmos tem tempo de execução $O(f(k) + n^c)$, onde a parte exponencial está em um termo aditivo ao invés de multiplicativo como na definição de $FPT$. Como vimos após a Definição 6.1, isto não define uma classe diferente da $FPT$.

# CAPÍTULO 7

# O TEOREMA DE ROBERTSON E SEYMOUR E A CLASSE $FPT$

Neste capítulo apresentamos a relação do projeto Graph Minors com a classe $FPT$ e com o método da redução ao núcleo do problema. Também neste capítulo vemos como a classe $FPT$ pode ser caracterizada por conjuntos de obstrução.

Os trabalhos e publicações de Robertson e Seymour chamamos de "projeto Graph Minors".

Alguns dos primeiros problemas parametrizados da classe $FPT$ vieram das várias publicações do projeto Graph Minors, o que apresentamos a seguir.

Iniciamos com algumas nomenclaturas necessárias .

**Definição 7.1.** *Uma quasi-ordem é uma relação binária $R$ sobre um conjunto $S$, tal que $R$ seja reflexiva e transitiva.*

Seja $S$ um conjunto e seja $R$ uma quasi-ordem sobre $S$.

Para facilitar a leitura usaremos $\preceq$ para representar a relação binária $R$, daí escrevemos $x \preceq y$ para representar $xRy$. Para $x, y \in S$ escrevemos $y \succ x$ se $x \preceq y$ e se $y \npreceq x$.

Seja $S' \subseteq S$ e $S'' \subseteq S$.

Dizemos que $S'$ é fechado superiormente quando para todo $x \in S'$, se $x \preceq y$ então $y \in S'$. Dizemos que $S''$ é fechado inferiormente quando para todo $y \in S''$, se $x \preceq y$ então $x \in S''$.

**Definição 7.2.** *$S'$ é um filtro em $S$ se e somente se $S'$ é fechado superiormente. $S''$ é um ideal em $S$ se e somente se $S''$ é fechado inferiormente.*

O *filtro gerado* por $S'$ é o conjunto $F(S') = \{y \in S\colon \text{ existe } x \in S' \text{ e } x \preceq y\}$. O *ideal gerado* por $S''$ é o conjunto $I(S'') = \{x \in S\colon \text{ existe } y \in S'' \text{ e } x \preceq y\}$.

Dizemos que $S'$ é *finitamente gerado* se $S'$ é um filtro ou $S'$ é um ideal que pode ser gerado por um subconjunto finito de $S'$, ou seja, dizemos que o filtro $S'$ é finitamente gerado se $F(S')$ é finito e dizemos que o ideal $S'$ é finitamente gerado se $I(S')$ é finito.

Escrevemos $(S, \preceq)$ para representar uma quasi-ordem $\preceq$ sobre $S$, o que simplificamos dizendo que $(S, \preceq)$ é uma quasi-ordem.

Sejam $(S, \preceq)$ uma quasi-ordem e $A = \{a_0, a_1, \ldots\}$ uma sequência de elementos de $S$.

- $A$ é uma sequência boa se existe $a_i \preceq a_j$ com $i < j$.
- $A$ é uma sequência ruim se não é boa.
- $A$ é uma cadeia se para todo $i < j$, $a_i \preceq a_j$.
- $A$ é uma anticadeia se para todo $i \neq j$ não existe $a_i \preceq a_j$ nem existe $a_j \preceq a_i$.
- $(S, \preceq)$ é Noetheriana se $S$ não contém uma sequência infinita decrescente, isto é, se não existe sequência $a_0 \succ a_1 \succ a_2 \succ \ldots$.
- $(S, \preceq)$ tem base finita se para todo $S' \subseteq S$, $S'$ é finitamente gerado.

Com isso podemos apresentar o seguinte resultado.

**Teorema 7.3.** *Seja $(S, \preceq)$ uma quasi-ordem. As seguintes afirmações são equivalentes.*

1. *$(S, \preceq)$ não tem sequência ruim.*
2. *Toda sequência infinita de $S$ contém uma cadeia infinita.*
3. *$(S, \preceq)$ é Noetheriana e $S$ não contém anticadeia infinita.*
4. *$(S, \preceq)$ tem base finita.*

O Teorema 7.3 usamos para definir uma quasi-boa-ordem.

## 7.1 Quasi-Boa-Ordem - WQO

Agora apresentamos uma quasi-boa-ordem − WQO (do inglês *Well-Quasi-Ordering*).

**Definição 7.4.** *Seja $(S, \preceq)$ uma quasi-ordem. Se $(S, \preceq)$ satisfaz qualquer caracterização do Teorema 7.3, então dizemos que $(S, \preceq)$ é uma quasi-boa-ordem − WQO.*

Seja $(S, \preceq)$ uma quasi-boa-ordem (uma WQO).

Consideremos o seguinte problema parametrizado.

| AcimaDe$(x, y)$ | |
|---|---|
| Instância: | $y \in S$ e $x \in S$. |
| Parâmetro: | $x \in S$. |
| Pergunta: | $x \preceq y$ ? |

Supondo que o problema parametrizado AcimaDe$(x, y)$ possa ser resolvido em tempo polinomial para qualquer $x$, então temos o seguinte princípio.

**Princípio da Quasi-Boa-Ordem** − o problema de decisão "$y \in F$?" é resolvido em tempo polinomial para qualquer filtro $F$ de $(S, \preceq)$.

A prova do Princípio da Quasi-Boa-Ordem baseia-se no seguinte: se $F$ é um filtro então $F$ tem uma base finita $\{b_1, b_2, \ldots, b_n\}$, uma vez que $(S, \preceq)$ é uma WQO. Então, para decidir se $y \in F$, somente precisamos perguntar "$\exists i \leq n$ tal que $b_i \preceq y$?" E como, para cada $b_i$ parâmetro fixo, nós estamos supondo que AcimaDe$(b_i, y)$ está em $\mathcal{P}$, então a questão "$\exists i \leq n$ tal que $b_i \preceq y$?" também está em $\mathcal{P}$.

O Princípio da Quasi-Boa-Ordem também poder ser usado na forma dual para ideais

se $F$ é um filtro de $(S, \preceq)$ então $S - F$ é um ideal, e vice-versa.

**Definição 7.5.** *Seja $(S, \preceq)$ uma quasi-ordem e seja $I$ um ideal de $(S, \preceq)$. Dizemos que o conjunto $O \subseteq S$ forma um conjunto de obstrução para $I$ se*

$x \in I$ *se e somente se* para todo $y \in O$, *não existe* $y \preceq x$.

Pela Definição 7.5 o conjunto $O$ é um conjunto de obstrução para $I$ se $I$ é o complemento de $F(O)$.

**Princípio da Obstrução** – Se $(S, \preceq)$ é uma WQO, então todo ideal $I$ tem um conjunto de obstrução finito e ainda, se o problema parametrizado AcimaDe$(x, y)$ está em $\mathcal{P}$ para todo $x \in S$, então para todo ideal $I$ temos "$y \in I$?" resolvido em tempo polinomial.

A prova do Princípio da Obstrução baseia-se no seguinte: como $(S, \preceq)$ é uma WQO então $I$ tem (uma base finita) um conjunto de obstrução finito e supondo que o problema AcimaDe$(x, y)$ está em $\mathcal{P}$ para todo $x \in S$, então o problema de decisão "$y \in I$?" também está em $\mathcal{P}$.

Disso temos um novo método para demonstrar que um problema está em $\mathcal{P}$: provamos que o problema é caracterizado por um filtro ou ideal finitamente gerado com um conjunto de obstrução finito em uma quasi-ordem com AcimaDe$(x, y)$ em $\mathcal{P}$.

Um exemplo de um conjunto de obstrução é daremos na ordenação topológica após sua definição no que segue.

Um homeomorfismo do grafo $G_1 = (V_1, E_1)$ no grafo $G_2 = (V_2, E_2)$ é uma função injetora dos vértices de $V_1$ para $V_2$ com a propriedade que as arestas de $E_1$ são mapeadas para caminhos disjuntos de $G_2$, onde estes caminhos disjuntos em $G_2$ representam possíveis subdivisões das arestas de $G_1$.

**Definição 7.6.** *O conjunto de grafos homeomorfos gera uma ordem parcial que chamamos de ordenação topológica.*

Embora a ordenação topológica não seja uma WQO, existe nela um número importante de resultados com base finita. Por exemplo, um resultado famoso com a base finita $O = \{K_{3,3}, K_5\}$ é o Teorema de Kuratowski.

**Teorema 7.7** (Teorema de Kuratowski)**.** *Os grafos $K_{3,3}$ e $K_5$ representados na Figura 7.1 formam um conjunto de obstrução para o ideal dos grafos planares na ordenação topológica.*

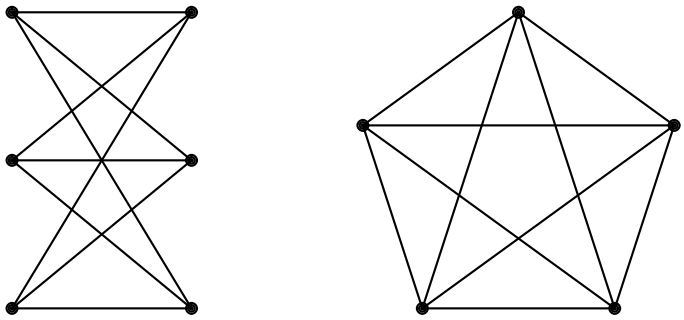

Figura 7.1: Grafos $K_{3,3}$ e $K_5$

Ainda em relação a homeomorfismos, consideremos o seguinte problema parametrizado.

Escrevemos $G_1 \leq_{top} G_2$ quando o grafo $G_1$ é homeomorfo ao grafo $G_2$.

HOMEOMORFISMO

| | |
|---|---|
| Instância: | Um grafo $G$ e um grafo $H$. |
| Parâmetro: | O grafo $H$. |
| Pergunta: | O grafo $H$ é homeomorfo ao grafo $G$, isto é, $H \leq_{top} G$? |

O problema parametrizado HOMEOMORFISMO foi conjecturado como sendo $FPT$ em 1989 por Michael Fellows. Essa conjectura foi provada usando análises provindas do projeto Graph Minors.

**Teorema 7.8.** *O problema* HOMEOMORFISMO *é* $FPT$ *e foi resolvido em tempo* $O(|G|^3)$.

O Teorema 7.8 foi provado por Martin Grohe, Kenichi Kawarabayashi, Dániel Marx e Paul Wollan em 2011.

Como vimos a ordenação topológica não é uma WQO, vejamos uma ordenação que é WQO no que segue.

## 7.2 Graph Minor Theorem

Nesta seção vemos o Graph Minor Theorem e suas relações com a classe $FPT$.

Iniciamos com a nomenclatura necessária.

Uma contração de uma aresta $\{u, v\}$ é a operação da remoção dessa aresta do grafo, os vértices $u$ e $v$ são unidos formando um novo vértice $w$ e as arestas anteriormente incidentes a $u$ e a $v$ tornam-se incidentes a $w$.

Na Figura 7.2 representamos essa operação da contração de uma aresta.

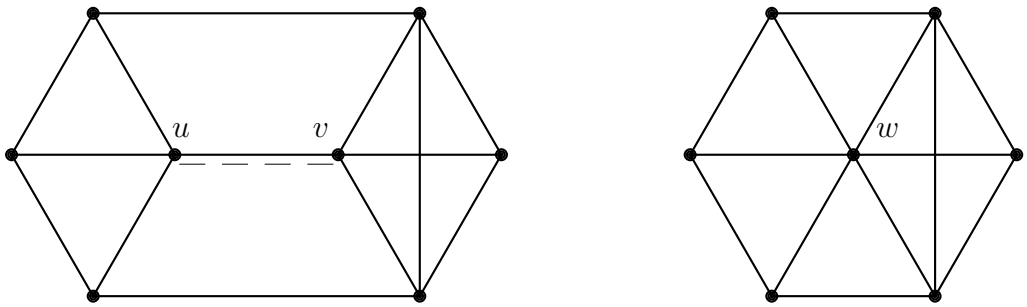


Figura 7.2: Contração de arestas

**Definição 7.9.** *Sejam $G$ e $H$ grafos. Dizemos que $G$ é um menor (do inglês minor) de $H$ e escrevemos $G \leq_{minor} H$ se o grafo $G$ pode ser obtido a partir de $H$ por uma sequência de contrações de arestas e remoções de vértices e arestas .*

O Teorema de Kuratowski exibido no Teorema 7.7 pode ser apresentado em termos de menores de grafos do seguinte modo.

**Teorema 7.10.** *Os grafos $K_{3,3}$ e $K_5$ da Figura 7.1 formam um conjunto de obstrução para o ideal dos grafos planares sob $\leq_{minor}$.*

Formulamos a Conjectura de Wagner do seguinte modo.

**Definição 7.11** (Conjectura de Wagner)**.** *Os grafos finitos formam uma quasi-boa-ordem WQO pela relação de menores $\leq_{minor}$.*

Agora que definimos a Conjectura de Wagner, renomeamos o projeto Graph Minors como sendo os trabalhos e publicações de Robertson e Seymour para provar a Conjectura de Wagner.

O seguinte teorema é considerado um dos triunfos do século 20.

**Teorema 7.12** (Graph Minor Theorem)**.** *Os grafos finitos formam uma quasi-boa-ordem WQO pela relação de menores $\leq_{minor}$.*

A prova do Teorema 7.12 espalha-se por mais de 23 artigos do projeto Graph Minors.

Para mais resultados de Robertson e Seymour consideremos o seguinte problema parametrizado.

MINOR TESTING

| | |
|---|---|
| Instância: | Um par de grafos $G$ e $H$. |
| Parâmetro: | O grafo $H$. |
| Pergunta: | $G$ é um menor de $H$, isto é, $G \leq_{minor} H$? |

Em [Robertson and Seymour(1995)] foi apresentado um algoritmo para menores em grafos tal que, para cada grafo $H$ fixo e dado um grafo $G$, o algoritmo decide se $H$ é um menor de $G$ em tempo $O(n^3)$.

**Teorema 7.13** ([Robertson and Seymour(1995)])**.** *O problema parametrizado* MINOR TESTING *pode ser resolvido em tempo* $f(k)n^3$.

**Corolário 7.14.** MINOR TESTING *é* $FPT$.

Vale lembrar que MINOR TESTING é um problema $\mathcal{NP}$-Completo em suas versão como problema computacional de decisão.

Outro problema $FPT$ provindo do projeto Graph Minors damos a seguir.

De modo informal consideremos que o genus de um grafo é o menor número de "alças" necessárias a serem colocadas em uma esfera para obtenção de uma superfície onde o grafo seja imersível sem que nenhuma aresta se cruze com outra. Consideremos o seguinte problema.

GRAPH GENUS

| | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | O grafo $G$ tem genus $k$? |

Usando o Teorema 7.13 tem-se que

**Corolário 7.15.** GRAPH GENUS *pode ser resolvido em tempo* $f(k)n^3$ *e portanto* GRAPH GENUS *é* $FPT$.

Vale lembrar que Graph Genus é um problema $\mathcal{NP}$-Completo em suas versão como problema computacional de decisão.

Pelo Teorema 7.12 - Minor Graph Theorem e pelo Teorema 7.13 muitos outros problemas parametrizados foram classificados como $FPT$.

Entretanto não é claro como os distintos $O(n^3)$ algoritmos para o Teorema 7.13 podem ser combinados em um único algoritmo, onde cada um dos algoritmos do Teorema 7.13 refere-se aos diferentes conjuntos de obstrução finitos.

## 7.3 A classe $FPT$ caracterizada por conjuntos de obstrução

Vejamos uma recente relação entre a classe $FPT$ e o Minor Graph Theorem.

Seja $(S, \preceq)$ uma quasi-ordem.

Dizemos que $\preceq$ é uma quasi-ordem de tempo polinomial se existe um algoritmo que decide se $x \preceq y$ em tempo $O((|x| + |y|)^{O(1)})$, para $x, y \in S$.

No Teorema 6.13 foi provado que um problema é $FPT$ se e somente se o problema tem kernelização.

Foi provado recentemente [Fellows and Jansen(2013)] o seguinte resultado.

**Teorema 7.16.** *Para qualquer problema parametrizado $\mathcal{Q} \in \Sigma^* \times \mathbb{N}$ as seguintes afirmações são equivalentes.*

1. *O problema $\mathcal{Q} \in FPT$ e admite um núcleo de tamanho computável.*
2. *Existe uma quasi-ordem $\preceq$ de tempo polinomial em $\Sigma^* \times \mathbb{N}$ e existe uma função $f : \mathbb{N} \to \mathbb{N}$ computável tal que para todo $(x, k) \notin \mathcal{Q}$ existe uma obstrução $(x', k') \notin \mathcal{Q}$ de tamanho no máximo $f(k)$ com $(x', k') \preceq (x, k)$ e com $\mathcal{Q}$ sendo decidível.*

Na prova do Teorema 7.16 em [Fellows and Jansen(2013)] é mostrado que os problemas parametrizados com $f$ sendo o tamanho do núcleo são caracterizados por conjuntos de obstrução de tamanho $f$ com uma quasi-ordem de tamanho polinomial.

Assim a classe $FPT$ é caracterizada por conjuntos de obstrução.

**Corolário 7.17** ([Fellows and Jansen(2013)]). *Se $\mathcal{Q} \in \Sigma^* \times \mathbb{N}$ é um problema parametrizado e o tamanho do núcleo de $\mathcal{Q}$ é $O(k^c)$, então existe uma quasi-ordem $\preceq$ de tempo polinomial em $\Sigma^* \times \mathbb{N}$ e uma função $f$ conforme o Teorema 7.16 com $f(k) \in O(k^c)$.*

Portanto, os problemas parametrizados com núcleos de tamanho polinomial podem ser caracterizados por conjuntos de obstrução de tamanho polinomial.

Esta relação quantitativa entre o tamanho do núcleo e o tamanho do conjunto de obstrução está direcionando pesquisas para esclarecer a relação entre a kernelização e os conjuntos de obstrução de modo mais concreto.

# CAPÍTULO 8

# CLASSES DE COMPLEXIDADE

Neste capítulo apresentamos as definições e exemplos para as classes de complexidade da chamada Hierarquia $W$ e suas relações com circuitos através do problema parametrizado Circuit-Sat, o qual foi visto anteriormente. Também apresentamos definições e resultados de completude.

Ainda neste capítulo vemos versões parametrizadas de problemas $\mathcal{NP}$-Completos e como estes problemas estão espalhados pelas classes da Hierarquia $W$. Apresentamos um refinamento da classe $\mathcal{NP}$-Completo ao classificar as versões dos problemas dessa classe como problemas parametrizados pertencentes às diversas classes da Hierarquia $W$.

Iniciamos definindo a Hierarquia $W$ e posicionando a classe $FPT$ nessa Hierarquia. Em seguida comparamos com a classe $\mathcal{NP}$-Completo e apresentamos algumas implicações.

## 8.1 A Hierarquia $W$

Reproduzimos a definição dada para WCS$(t, h)$, problema parametrizado da satisfatibilidade de circuitos de trama $t$ e profundidade $h$, que é a restrição do problema Circuit-Sat para os circuitos de trama $t$ e profundidade $h$.

| WCS$(t, h)$ | |
|---|---|
| Instância: | Um circuito $C$ de trama $t$ e profundidade $h$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $C$? |

**Definição 8.1.** *Seja $P$ um problema parametrizado e $t$ um inteiro positivo. $P$ está na classe $W[t]$ se e somente se $P$ é redutível a* WCS$(t, h)$*, via redução parametrizada e para algum $h > 0$.*

Uma conjectura fundamental em Complexidade Parametrizada é que $FPT \neq W[1]$, o que é análoga a conjectura $\mathcal{P} \neq \mathcal{NP}$ porém mais "fraca".

No Teorema 5.3 vimos que SAT $\leq_{rp}$ WCS$(2, 2)$, via redução parametrizada. Portanto

$$\text{SAT} \in W[2].$$

No Corolário 5.4 temos 2SAT $\leq_{rp}$ WCS$(1, 2)$ e daí

$$2\text{SAT} \in W[1].$$

No Teorema 5.5 temos CONJUNTO DOMINANTE $\leq_{rp}$ WCS$(2, 2)$, via redução parametrizada. Portanto

$$\text{CONJUNTO DOMINANTE} \in W[2].$$

Pelo Teorema 5.2 temos

$$\text{COBERTURA POR VÉRTICES} \in W[1].$$

**Definição 8.2.** *A classe $W[P]$ é a classe onde não há limite na profundidade e na trama dos circuitos.*

Pela Definição 8.2 temos

$$\text{CIRCUIT-SAT} \in W[P].$$

Dessas definições temos

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq W[P].$$

Conjectura-se em [Downey and Fellows(1999)] que essas inclusões são próprias.

Denominamos *Hierarquia W* para a união dessas classes, sendo a letra $W$ provinda da trama (do inglês *Weft*) dos circuitos de cada classe.

**Definição 8.3.** *Um problema parametrizado $P$ é $W[t]$-Difícil se todo problema parametrizado de $W[t]$ é redutível a $P$ via redução parametrizada.*

**Definição 8.4.** *Um problema parametrizado $P$ é $W[t]$-Completo se e somente se $P$ está na classe $W[t]$ e se todo problema parametrizado de $W[t]$ é redutível a $P$.*

No que segue apresentamos vários problemas parametrizados $W[t]$-Difíceis e $W[t]$-Completos, começando com a classe $W[1]$ e estudando os problemas $W[1]$-Difíceis e $W[1]$-Completos.

Também apresentamos vários problemas $\mathcal{NP}$-Completos cujas versões parametrizadas são $W[t]$-Completos para distintos $t$, ocorrendo assim um refinamento da classe $\mathcal{NP}$-Completo.

## 8.2 A Classe $W[1]$

A classe $W[1]$ da Hierarquia $W$ é tida como análoga à classe $\mathcal{NP}$-Completo, o que vemos em detalhes no que segue.

Foi definido o problema parametrizado $\mathrm{WCS}(t, h)$ da satisfatibilidade de circuitos com trama $t$ e profundidade $h$ pelo qual $W[t]$ foi construída. Reproduzimos a definição de $\mathrm{WCS}(t, h)$ para $t = 1$ no seguinte problema parametrizado.

| $\mathrm{WCS}(1, h)$ | |
|---|---|
| Instância: | Um circuito $C$ de trama 1 e profundidade $h$ e um inteiro $k \geq 0$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $C$? |

Reproduzimos a Definição 8.1 restringindo $t = 1$ para a classe $W[1]$.

**Definição 8.5.** *Seja $P$ um problema parametrizado. $P$ está na classe $W[1]$ se e somente se $P$ é redutível a $\mathrm{WCS}(1, h)$ via redução parametrizada, para algum $h$ inteiro positivo.*

Abaixo repetimos a Figura 5.8 contendo um grafo $G$ com 5 vértices e contendo um circuito $C_5$ com profundidade $h = 2$, com trama $t = 1$ que é a "profundidade de portas amplas" deste circuito e com *fan-in* 2 nas portas **ou** pequenas, tal que

$G$ tem cobertura por vértices de tamanho $k$ se e somente se

$C_5$ é satisfeito por valoração de peso $k$.

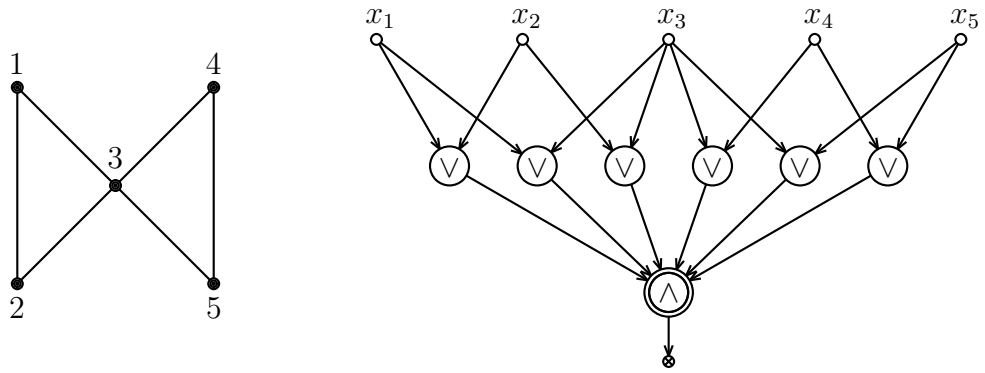


Figura 8.1: Um circuito a partir de um grafo para a cobertura por vértices (repetição)

Esse ideia de construir uma família de circuitos a partir de um problema parametrizado exemplifica como proceder para verificar em qual classe da Hierarquia $W$ um problema parametrizado pertence.

Pelo Teorema 5.2 Cobertura por Vértices $\leq_{rp}$ WCS$(1, 2)$ e daí temos

Cobertura por Vértices $\in W[1]$.

Consideremos as fórmulas $F' = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_1)$ e $F'' = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (x_3 \vee x_4) \wedge (x_4 \vee x_1)$. Destas fórmulas $F'$ e $F''$ construímos os circuitos de trama 1 e profundidade 2, respectivamente representados à esquerda e à direita na Figura 8.2. As variáveis de $F'$ e $F''$ são representadas como entradas no circuito associado. As portas pequenas nos circuitos representam as disjunções $\vee$ das fórmulas $F'$ e $F''$. No Capítulo 4 reescrevemos $F'$ e $F''$ como sendo $F' = \bigwedge_{i=1,2,3} F_i$ e $F'' = \bigwedge_{i=1,\ldots,4} F_i$. O símbolo $\bigwedge$ usado ao reescrever as fórmulas está representado nos circuitos como uma porta **e** ampla. Com

isso é bastante direto representar circuitos a partir de fórmulas, em particular as fórmulas normalizadas.

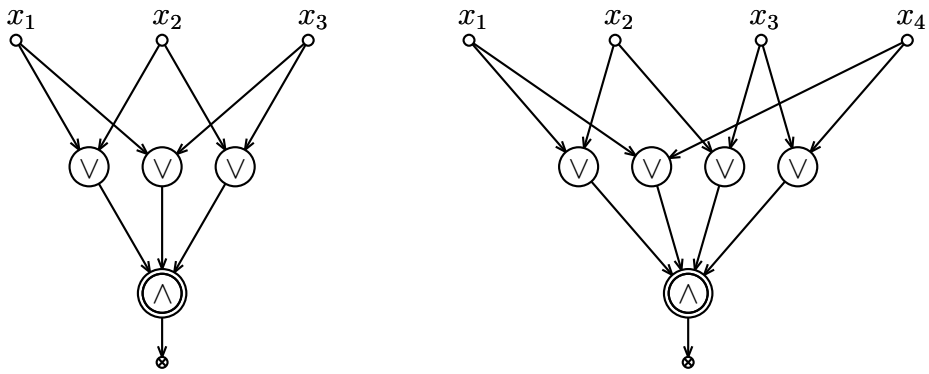


Figura 8.2: Circuitos representando fórmulas com valoração de mesmo peso

Os circuitos da Figura 8.2 acima são representações das fórmulas $F'$ e $F''$. O circuito é satisfeito por uma valoração de peso $k$ se e somente se a fórmula correspondente é satisfeita por uma valoração de peso $k$. Assim, um circuito de trama 1 e profundidade 2 pode ser computado a partir de qualquer fórmula de 2SAT de valoração no mesmo peso $k$.

Portanto 2SAT é redutível a WCS$(1, 2)$ via redução parametrizada, ou seja,

$$\text{2SAT} \leq_{rp} \text{WCS}(1, 2).$$

Daí

$$\text{2SAT} \in W[1].$$

Para o problema CONJUNTO INDEPENDENTE usamos a Figura 8.3 onde exibimos um grafo conexo e um circuito de trama 1 e profundidade 2, tal que o grafo tem conjunto independente de tamanho $k$ se e somente se o circuito é satisfeito por uma valoração de peso $k$. O circuito foi criado a partir do grafo do seguinte modo: criamos uma entrada no circuito para cada vértice do grafo. Para cada aresta $\{u, v\}$ do grafo criamos uma porta `ou` pequena com negação na aresta antes da porta. Utilizamos uma única porta `e` ampla para receber todas as saídas das portas `ou` descritas. A generalização imediata desta

transformação prova que Conjunto Independente restrito aos grafos conexos pertence a classe $W[1]$, uma vez que Conjunto Independente restrito aos grafos conexos é redutível a WCS$(1, 2)$, via redução parametrizada.

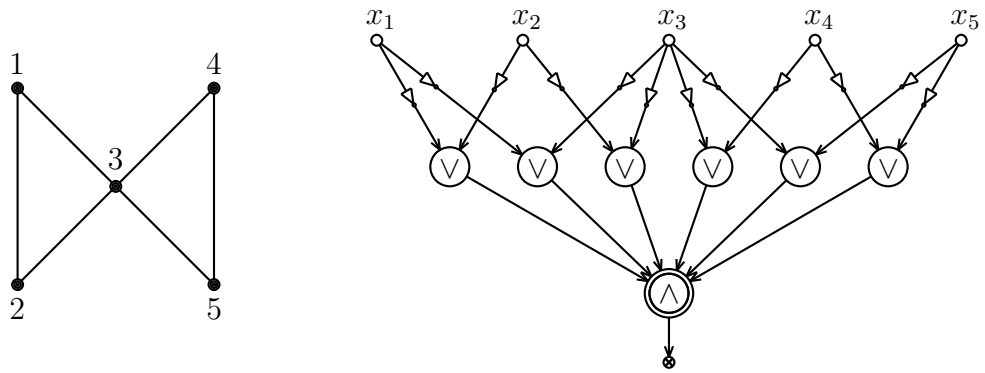


Figura 8.3: Um circuito a partir de um grafo para o conjunto independente

Os próximos três teoremas serão muito úteis para a classificação de problemas parametrizados com relação a classe $W[1]$, mas primeiro vejamos alguns exemplos e definições que são necessários.

Pela Definição 8.5 um problema parametrizado $P$ pertence a classe $W[1]$ se e somente se $P$ é redutível a WCS$(1, h)$, por uma redução parametrizada, onde $h$ é um inteiro positivo. Os circuitos de WCS$(1, h)$ tem trama 1 e profundidade limitada por $h$.

Os circuitos $C_3$, $C_4$ e $C_5$ representados na Figura 5.4 partem de uma regra para formar uma família de circuitos $\mathcal{F}$. Seja $\mathcal{F}$ a família de circuitos tal que o circuito $C_n$ com entradas $x_1, x_2, \ldots, x_n$ tem uma porta ampla $\wedge$ para as entradas $x_i$ com $i$ ímpar e tem uma porta ampla $\wedge$ para as entradas $x_j$ com $j$ par. Consideremos o circuito $C_8$ representado na Figura 8.4 como representante da família de circuitos $\mathcal{F}$.

Consideremos a Figura 8.4 para exemplificar a trama $t$, a profundidade $h$ e o *fan-in* $s$ do circuito $C_8$ representante da família $\mathcal{F}$.

- A profundidade de $C_8$ é $h = 3$ pois o caminho de qualquer entrada à saída passa por no máximo 3 portas do circuito.
- A trama de $C_8$ é $t = 2$ pois o caminho de qualquer entrada à saída passa por no máximo 2 portas amplas.

- O *fan-in* das portas pequenas de $C_8$ é $s = 2$ pois qualquer porta pequena do circuito tem grau de entrada sendo 2.

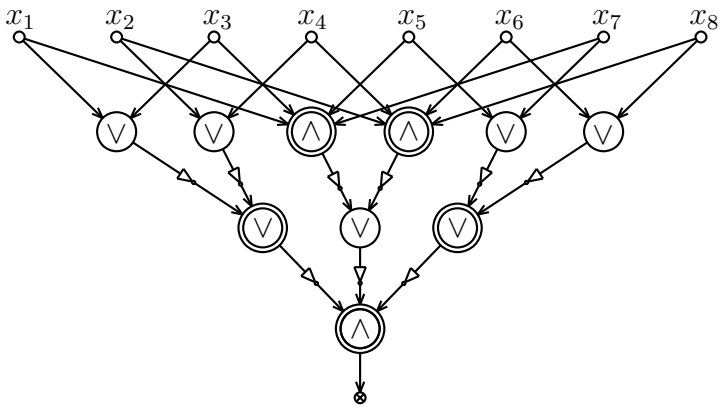


Figura 8.4: Um circuito de uma família de circuitos de trama 2 e profundidade 3

Considerando o circuito $C_8$ da Figura 8.4 acima como representante da família $\mathcal{F}$ em relação à trama, à profundidade e ao *fan-in*, então podemos dizer que qualquer circuito $C \in \mathcal{F}$ tem trama $t = 2$, profundidade $h = 3$ e *fan-in* das portas pequenas limitado em $s = 2$.

Vamos definir o problema parametrizado WCS$(1, h, s)$ como a restrição do problema parametrizado WCS$(1, h)$ aos circuitos com limite $s$ no *fan-in* das portas pequenas.

| WCS$(1, h, s)$ | |
|---|---|
| Instância: | Um circuito $C$ com trama 1, profundidade $h$, *fan-in* $s$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $C$? |

**Teorema 8.6** (Lema 2.1 de [Downey(1995)])**.** *Seja $h$ um inteiro positivo.* WCS$(1, h) \leq_{rp}$ WCS$(1, 2, 2^h + 1)$*, por uma redução parametrizada.*

Para provar o Teorema 8.6 toma-se um circuito $C$ de trama $t = 1$ e profundidade $h$ de WCS$(1, h)$ e constrói-se um circuito $C'$ de trama $t = 1$, profundidade 2 e *fan-in* $s = 2^h + 1$ de WCS$(1, 2, s)$, tal que $C$ é satisfeito por uma valoração de em $k$ se e somente se $C'$ é satisfeito por uma valoração de peso $k'$, onde $k' = f(k)$ ($k'$ depende apenas de $k$).

O circuito $C'$ resultante da demonstração do Teorema 8.6 tem uma forma bastante simples: com uma única porta **e** ampla antes da saída do circuito recebendo a saída de todas as portas pequenas **ou**. Estas portas pequenas **ou** tem seu *fan-in* limitado por $s = 2^h + 1$.

Na Figura 8.5 a seguir representamos um circuito de *fan-in* $s = 6$ com uma única porta **e** ampla antes da saída do circuito recebendo a saída de todas as portas **ou** pequenas.

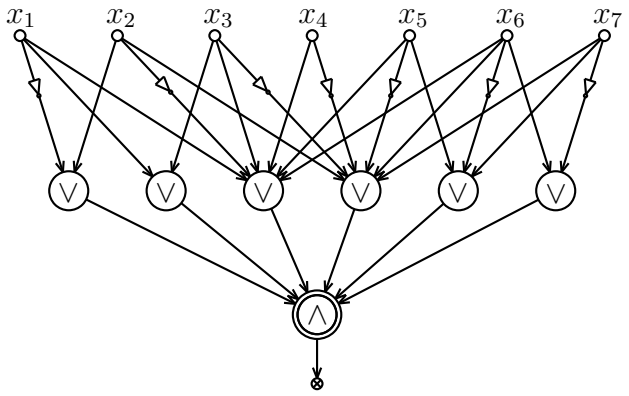


Figura 8.5: Um circuito com uma porta **e** ampla e portas **ou** pequenas de *fan-in* 6

Uma consequência do Teorema 8.6 é estratificação de $W[1]$ como a união de $\mathrm{WCS}(1, 2, s)$ com os circuitos tendo $s$ no limite do *fan-in* das portas pequenas, ou seja,

$$W[1] = \bigcup_{s=1}^{\infty} \mathrm{WCS}(1, 2, s).$$

Cometemos o abuso de afirmar "$W[1]$ igual a união de $\mathrm{WCS}(1, 2, s)$": é abuso uma vez que $W[1]$ é uma classe de complexidade e $\mathrm{WCS}(1, 2, s)$ é um problema parametrizado. Cometemos esse abuso pelo fato que a Definição 8.5 associa a classe $W[1]$ com a redução a um problema específico, mas deve ser levado em conta a redução necessária associada.

Semelhantemente às fórmulas antimonótonas temos os circuitos antimonótonos: um circuito é antimonótono se todas as arestas que saem das entradas do circuito são arestas negativas (arestas com rótulo $\neg$) e não possui nenhuma outra aresta negativa no circuito.

Na Figura 8.6 damos uma representação de um circuito antimonótono.

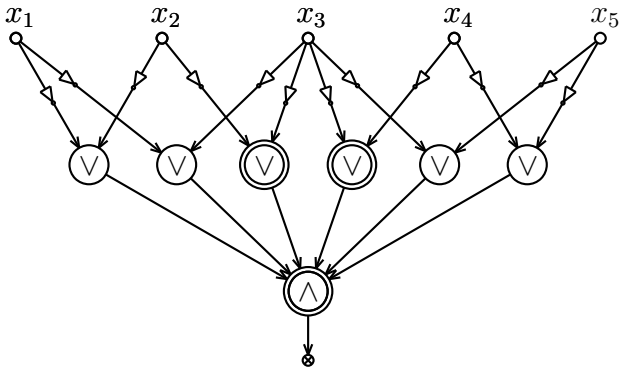


Figura 8.6: Um circuito antimonótono

Dos circuitos antimonótonos temos o problema WCS-AntiM$(1, h, s)$ dado a seguir.

WCS-AntiM$(1, h, s)$

| | |
|---|---|
| Instância: | Um circuito $C$ antimonótono de trama 1, de profundidade $h$ e de *fan-in* $s$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $C$? |

O problema WCS-AntiM$(1, h, s)$ é importante para transformar qualquer circuito de WCS$(1, 2, s)$ em um circuito de WCS-AntiM$(1, 2, s)$, isto devido ao seguinte Teorema.

**Teorema 8.7.** *Seja $s \geq 2$ e $s \in \mathbb{N}$.* WCS$(1, 2, s) \cong$ WCS-AntiM$(1, 2, s)$*, por redução parametrizada.*

Com o próximo resultado concluímos que a classe $W[1]$ é equivalente a 2Sat, ou seja, daí

2Sat é $W[1]$-Completo.

**Teorema 8.8.** *Seja $s \geq 2$ e $s \in \mathbb{N}$.* WCS-AntiM$(1, 2, s) \cong$ WCS$(1, 2, 2)$*, por redução parametrizada.*

**O resumo dos últimos resultados é o seguinte.**

- $W[1] = \bigcup_{h=1}^{\infty}$ WCS$(1, h)$ pela Definição 8.1.

- WCS$(1, h) \cong$ WCS$(1, 2, s = 2^h + 1)$ pelo Teorema 8.6 para $h \geq 2$.
- WCS$(1, 2, s) \cong$ WCS-AntiM$(1, 2, s)$ pelo Teorema 8.7 para $s \geq 2$.
- WCS-AntiM$(1, 2, s) \cong$ WCS$(1, 2, 2)$ pelo Teorema 8.8 para $s \geq 2$.
- Portanto WCS$(1, 2, 2)$ é $W[1]$-Completo.
- Como WCS$(1, 2, 2) \cong$ 2Sat então 2Sat é $W[1]$-Completo.

Em outras palavras, qualquer problema parametrizado $P$ pertencente a classe $W[1]$ é redutível a 2Sat por uma redução parametrizada, isto é,

$$P \leq_{rp} \text{2Sat}, \text{ para todo } P \in W[1].$$

Com os resultados acima, se $P \leq_{rp}$ 2Sat então $P \in W[1]$. Se 2Sat $\leq_{rp} P$ então $P$ é $W[1]$-Difícil, o que mostramos em detalhes a seguir.

## 8.3 As Classes $W[1]$-Difícil e $W[1]$-Completo

Reproduzimos as definições para $W[t]$-Difícil e $W[t]$-Completo dadas anteriormente, restringindo $t$ para $t = 1$.

**Definição 8.9.** *Um problema parametrizado $P$ é $W[1]$-Difícil se todo problema parametrizado de $W[1]$ é redutível a $P$, por uma redução parametrizada. Um problema parametrizado $P$ é $W[1]$-Completo se e somente se $P \in W[1]$ e $P$ é $W[1]$-Difícil.*

Pela definição acima ainda e pelos teoremas anteriores temos

**Corolário 8.10.** 2Sat *é $W[1]$-Completo.*

A prova do Corolário 8.10 vem do Corolário 5.4, onde provamos que 2Sat $\in W[1]$ e vem pelo resumo de vários resultados feito acima, onde qualquer problema parametrizado de $W[1]$ é redutível a WCS$(1, 2, 2)$.

Daí temos os seguintes problemas $W[1]$-Completos.

**Corolário 8.11.** *n*SAT *é $W[1]$-Completo, para $n \geq 2$ inteiro.*

Assim, para provar quem um problema parametrizado $P$ é $W[1]$-Completo precisamos provar que $P$ pertence a $W[1]$ e que todo problema de $W[1]$ é redutível a $P$ por uma redução parametrizada.

## 8.4 Clique, Conjunto Independente e a Classe $W[1]$-Completo

Pelo Teorema 3.5 temos CLIQUE $\cong$ CONJUNTO INDEPENDENTE, isto é, CLIQUE $\leq_{rp}$ CONJUNTO INDEPENDENTE e CONJUNTO INDEPENDENTE $\leq_{rp}$ CLIQUE.

Relacionamos estes problemas com a classe $W[1]$ iniciando pelo seguinte Teorema.

**Teorema 8.12.** CONJUNTO INDEPENDENTE $\in W[1]$.

Seja $(G, k)$ uma instância de CONJUNTO INDEPENDENTE. Se $G$ é um grafo conexo então já vimos anteriormente a redução para circuitos e exemplificamos na Figura 8.3.

Para os grafos não conexos exemplificamos na Figura 8.7 a seguir, tal que o grafo tem conjunto independente de tamanho $k$ se e somente se o circuito é satisfeito por uma valoração de peso $k$.

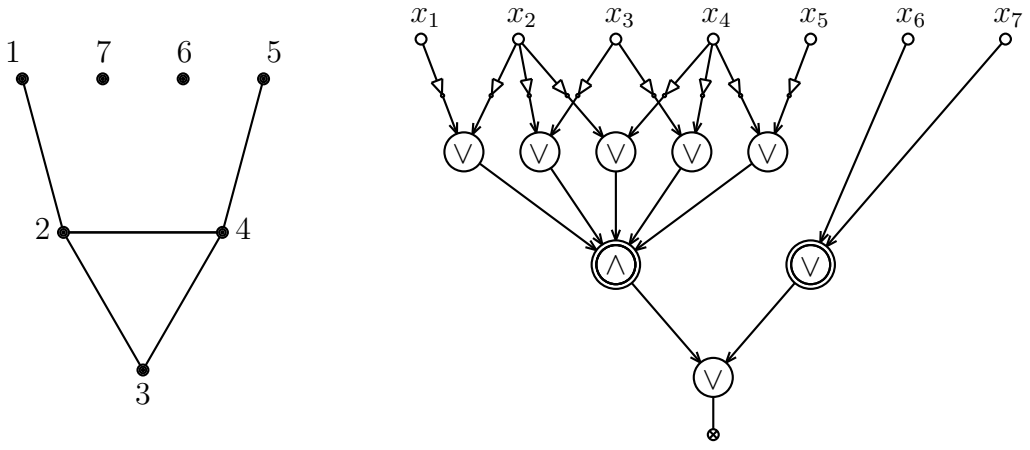


Figura 8.7: Um circuito a partir de um grafo não-conexo para o conjunto independente

Seja $G$ um grafo não conexo. Como exemplificado na Figura 8.7 construímos um circuito $C$ com uma entrada no circuito para cada vértice de $G$. Para cada aresta $\{u, v\}$

do grafo criamos uma porta **ou** pequena com arestas com negação saindo de $u$ e de $v$ para a entrada desta porta **ou**. Usamos uma única porta **e** ampla para todas essas portas **ou** descritas. Todas as entradas associadas a um vértice isolado tem sua saída em uma única porta **ou** ampla. Por fim uma porta **ou** de *fan-in* 2 recebe a saída das duas portas amplas. Desta forma Conjunto Independente restrito a grafos não conexos é redutível a WCS$(1, 3)$.

As reduções do problema Conjunto Independente para os problemas WCS$(1, 2)$ e WCS$(1, 3)$ são reduções parametrizadas com $k' = k$.

Portanto Conjunto Independente $\in W[1]$.

**Teorema 8.13.** Conjunto Independente *é* $W[1]$*-difícil.*

*Demonstração.* Pelo Corolário 8.10 para todo problema $P \in W[1]$ temos

$$P \leq_{rp} \text{2Sat}.$$

Pelo Corolário 5.4 temos

$$\text{2Sat} \leq_{rp} \text{WCS}(1, 2, 2).$$

Do Teorema 8.7 temos

$$\text{WCS}(1, 2, 2) \leq_{rp} \text{WCS-AntiM}(1, 2, 2).$$

A redução do Corolário 5.4 nos dá

$$\text{WCS-AntiM}(1, 2, 2) \leq_{rp} \text{2Sat-AntiM}.$$

Portanto basta provar que 2Sat-AntiM $\leq_{rp}$ Conjunto Independente.

Seja $(F, k)$ uma instância de 2Sat-AntiM. Daí a fórmula $F$ é antimonótona, ou seja, todos os literais de $F$ são negativos e ainda, todas as cláusulas de $F$ tem no máximo dois

literais.

Criamos um grafo $G$ com um vértice para cada variável de $F$. Para cada cláusula $(\neg u \vee \neg v)$ de $F$ criamos uma aresta $\{u, v\}$. Como a cláusula $(\neg u \vee \neg v)$ tem valoração verdadeira se não valorar ambas as variáveis $u$ e $v$ verdadeiras, então não podemos ter ambos os vértices $u$ e $v$ em um mesmo conjunto independente, o que ocorre pois construímos a aresta $\{u, v\}$ incidente a eles. Reciprocamente, como existe a aresta $\{u, v\}$ então ou $u$ ou $v$ pertence ao conjunto independente e daí ou $u$ ou $v$ é respectivamente valorada como verdadeira.

Portanto $F$ tem uma valoração de peso $k$ se e somente se $G$ tem um conjunto independente de tamanho $k'$, com $k' = k$. E como $k' = k$ e criamos $G$ em tempo $O(|F|^{O(1)})$, então a redução é uma redução parametrizada, o que prova o teorema. □

**Corolário 8.14.** Conjunto Independente *é* $W[1]$*-Completo.*

Já que pelo Teorema 3.5 temos Conjunto Independente $\cong$ Clique, portanto temos o seguinte resultado.

**Corolário 8.15.** Clique *é* $W[1]$*-Completo.*

## 8.5 Aceitação Curta em Máquina de Turing

O Teorema Cook-Levin sugere que o problema computacional da satisfatibilidade de fórmulas em CNF não possui um algoritmo polinomial, de forma semelhante temos um problema na classe $W[1]$ que é completo: uma versão parametrizada de uma máquina de Turing não-determinística. Daí a classe $W[1]$ é tida como análoga à classe $\mathcal{NP}$-Completo.

Consideremos o seguinte problema parametrizado.

| Aceitação Curta em Máquina de Turing | |
|---|---|
| Instância: | Uma máquina de Turing Não-Determinística $M$, uma palavra $x$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Existe algum caminho na computação de $M$ tal que $x$ é aceito por $M$ em no máximo $k$ transições? |

Provar que Aceitação Curta na Máquina de Turing pertence a $W[1]$ significa que 2Sat, Clique e Conjunto Independente são "mais difíceis" do que Aceitação Curta na Máquina de Turing, pois se 2Sat for $FPT$ então pelo Teorema 6.14 tem-se Aceitação Curta na Máquina de Turing também em $FPT$.

Provar que Aceitação Curta na Máquina de Turing é $W[1]$-Difícil significa que todo problema parametrizado de $W[1]$ é redutível a Aceitação Curta na Máquina de Turing, via redução parametrizada.

**Teorema 8.16.** Aceitação Curta na Máquina de Turing *é* $W[1]$*-Completo.*

A prova do Teorema 8.16 não cabe neste texto introdutório.

Portanto, Aceitação Curta na Máquina de Turing $\in W[1]$ e Aceitação Curta na Máquina de Turing é $W[1]$-Difícil.

Assim, por exemplo temos a consequência de que

$$\text{Aceitação Curta na Máquina de Turing} \leq_{rp} \text{Clique}.$$

Daí a classe $W[1]$ atrai bastante atenção, pois pelo Teorema 8.16 temos $W[1]$ como análoga à classe $\mathcal{NP}$-Completo.

## 8.6 As Classes $W[t]$ e as relações entre elas

No Capítulo 4 foi definido que uma fórmula é *t-normalizada* se ela está na forma de *conjunção-de-disjunções-de-conjunções- ... de-literais* com $t-1$ alternâncias. Considere-

mos o seguinte problema.

| Sat $t$-Normalizada | |
|---|---|
| Instância: | Uma fórmula $F$ $t$-normalizada e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | Há valoração de peso $k$ que satisfaz $F$? |

Desse problema temos o Teorema da Normalização de [Downey and Fellows(1995)].

**Teorema 8.17** (Teorema da Normalização)**.** Sat $t$-Normalizada *é* $W[t]$*-Completo, para todo inteiro* $t \geq 2$.

Por consequência do Teorema Normalização um circuito $C$ de trama $t$ e profundidade $h$ pode ser considerado como uma fórmula $t$-normalizada. Também do Teorema da Normalização temos

**Corolário 8.18.** Sat *é* $W[2]$*-Completo.*

Pelo Teorema da Normalização podemos classificar um problema parametrizado $P$ na classe $W[t]$ da Hierarquia $W$ se o problema $P$ for redutível, via redução parametrizada, a uma fórmula $t$-normalizada, para algum $t > 1$. No caso $t = 1$ vimos na Seção 8.2 que um problema parametrizado $P$ está na classe $W[1]$ se e somente se o problema $P$ for redutível a 2Sat, via redução parametrizada.

Com o Teorema 5.5 onde Conjunto Dominante $\cong$ WCS$(2, 2)$ temos o seguinte resultado

**Corolário 8.19.** Conjunto Dominante *é* $W[2]$*-Completo.*

Portanto,

**Corolário 8.20.** Sat $\cong$ Conjunto Dominante*.*

Com o Teorema da Normalização temos as seguintes relações.

$$FPT = W[1]\text{-Monótona} \subseteq$$

$$\subseteq W[1] = W[1]\text{-}\textsc{Antimonótona} = W[2]\text{-}\textsc{Antimonótona} \subseteq$$

$$\subseteq W[2] = W[2]\text{-}\textsc{Monótona} = W[3]\text{-}\textsc{Monótona} \subseteq$$

$$\ldots \subseteq W[P].$$

Conjectura-se em [Downey and Fellows(1999)] que essas inclusões são próprias.

# CAPÍTULO 9

# APLICAÇÃO AO PROBLEMA DA CLIQUE MÁXIMA

Neste capítulo fazemos uma aplicação das classes de complexidade $W[t]$ ao problema parametrizado Clique, apresentando análise de alguns algoritmos bastante usados para problema computacional da Clique Máxima com vista aos parâmetros envolvidos.

## 9.1 Revisando alguns resultados

Reproduzimos o problema da clique com parâmetro no tamanho do conjunto solução.

Clique

| | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | $G$ tem uma clique de tamanho $k$? |

No Teorema 3.5 vimos que Clique $\cong$ Conjunto Independente e nos Teoremas 8.12 e 8.13 vimos que Conjunto Independente é $W[1]$-Completo. Daí a Clique é $W[1]$-Completo. Comentamos que a classe $W[1]$ é tida como análoga à classe $\mathcal{NP}$-Completo.

Reproduzimos abaixo a Definição 6.1 dada para a classe $FPT$, que é tida como análoga à classe $\mathcal{P}$.

**Definição 9.1.** *Um problema parametrizado $P$ é tratável por parâmetro fixo (é $FPT$) se pode ser resolvido em tempo $f(k)|x|^{\alpha}$, para algum $\alpha \in \mathbb{N}$, onde $(x, k)$ é uma instância de $P$ e a função $f$ depende apenas do parâmetro $k$.*

Dos Teoremas 6.2 e 6.10 temos que Cobertura por Vértices $\in FPT$. O tempo de execução $O(kn + 1,2738^k)$ em [Chen et al.(2006)Chen, Kanj, and Xia] para Cobertura por Vértices é coerente com a classe $FPT$, conforme visto anteriormente. Também

vimos o quanto os tempos de execução para Cobertura por Vértices melhoraram ao longo dos anos.

No Teorema 3.2 vimos que COB $\propto$ IND e no Corolário 3.3 vimos que IND $\propto$ COB, onde os problemas de decisão COB e IND foram definidos relacionados respectivamente à cobertura por vértices e ao conjunto independente.

No Teorema 3.5 vimos que Conjunto Independente $\leq_{rp}$ Clique.

Uma redução parametrizada de Cobertura por Vértices para Clique não é conhecida e em [Downey and Fellows(1999)] conjectura-se não existir.

Como vimos, o parâmetro pode ser tomado de outras formas, além do tamanho do conjunto solução. Consideremos o seguinte problema parametrizado.

Grande Clique

| | |
|---|---|
| Instância: | Um grafo $G = (V, E)$ com $n$ vértices e um inteiro positivo $k$. |
| Parâmetro: | O inteiro positivo $k$. |
| Pergunta: | $G$ tem uma clique de tamanho $n - k$? |

O tempo de execução $O(kn + 1, 2738^k)$ para Cobertura por Vértices é útil para o problema parametrizado Grande Clique pelo seguinte teorema.

**Teorema 9.2.** Grande Clique $\cong$ Cobertura por Vértices*, via redução parametrizada.*

A prova do Teorema 9.2 usa a redução bastante conhecida da cobertura por vértices ao conjunto independente, exibida no Teorema 3.2 e também usa o Teorema 3.5, onde Conjunto Independente $\cong$ Clique.

Informalmente o tempo de execução $O(kn+1, 2738^k)$ para Cobertura por Vértices é útil para Grande Clique nos casos onde procuramos "grandes" cliques. Ou seja, a redução parametrizada de Cobertura por Vértices para Grande Clique significa que procurar uma cobertura por vértices de tamanho $k$ em um grafo $G$ resolve o problema de procurar uma clique de tamanho $n - k$ no grafo complementar de $G$.

Como no Teorema 9.2, Grande Clique é redutível a Cobertura por Vértices

e como Cobertura por Vértices $\in FPT$, então temos a seguinte consequência.

**Corolário 9.3.** Grande Clique *é* $FPT$.

## 9.2 Degeneração de um grafo

A degeneração é conhecida como uma medida de esparsidade dos grafos e a definimos do seguinte modo.

**Definição 9.4.** *A degeneração de um grafo simples $G$ é o menor número $d$, tal que todo subgrafo de $G$ contém um vértice de grau no máximo $d$.*

A degeneração de um grafo é um limite superior para a clique máxima conforme o resultado a seguir.

**Teorema 9.5.** *Se um grafo simples $G$ tem degeneração $d$, então a maior clique nesse grafo tem no máximo $d+1$ vértices, isto é, $\omega(G) \leq d+1$.*

*Demonstração.* Seja $G$ um grafo simples com uma degeneração $d$ e seja $S$ a clique máxima de $G$. O grafo $G[S]$ é um subgrafo de $G$. Pela Definição 9.4, qualquer subgrafo de $G$ tem um vértice de grau no máximo $d$. Daí $G[S]$ tem um vértice de grau no máximo $d$. Portanto a maior clique $S$ no grafo $G$ tem no máximo $d+1$ vértices. □

A relação $\omega(G) \leq d+1$ entre a Clique Máxima e degeneração conforme dada pelo Teorema 9.5 é de grande importância, o que mostraremos nas análises dos algoritmos para Clique Máxima.

Se lermos a Definição 9.4 de modo simplista pode parecer difícil para calcular a degeneração de um grafo. Entretanto, apresentamos o algoritmo degeneração($G$) para calcular

a degeneração $d$ de um grafo $G$ em tempo polinomial.

---

degeneração($G$)

---

$d \leftarrow 0$

Enquanto $G$ *não estiver vazio*

  Se $\delta(G) > d$

    $d \leftarrow \delta(G)$

  remova do grafo $G$ um vértice de grau mínimo (e suas arestas)

Devolva $d$

---

Consideremos o grafo da Figura 9.1 para calcular a degeneração.

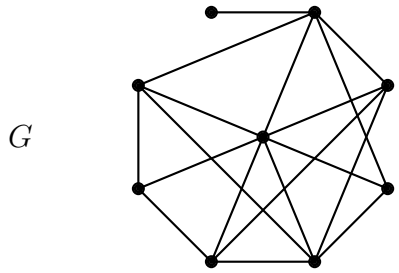


Figura 9.1: Um grafo simples com degeneração 3

Consideremos usar o algoritmo degeneração($G$) com $G$ sendo o grafo da Figura 9.1. A variável $d$ inicia com valor 0. Após ler todo o grafo é encontrado o grau mínimo $\delta(G)$ sendo 1 e daí $d \leftarrow 1$. Depois de remover o vértice de grau 1 (e suas arestas), o grafo $G$ resultante tem grau mínimo sendo 3. A variável $d$ é então atualizada para $d \leftarrow 3$, valor o qual permanece e é o valor devolvido pelo algoritmo como sendo a degeneração do grafo. Daí, o grafo da Figura 9.1 tem degeneração 3. A prova do algoritmo degeneração($G$) mostraremos mais adiante.

Na Figura 9.2 numeramos em ordem crescente os vértices da Figura 9.1 conforme o algoritmo degeneração($G$) foi removendo os vértices do grafo, usando uma escolha qualquer nos casos onde havia mais de um vértice com grau igual ao grau mínimo.

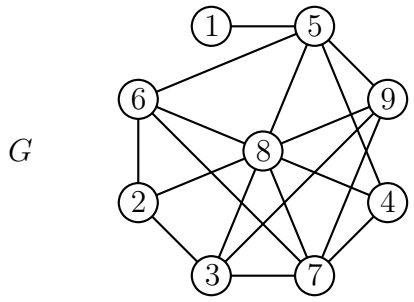


Figura 9.2: Grafo com vértices numerados ao usar o algoritmo degeneração($G$)

## 9.3 Grafos direcionados usando a degeneração

Vamos construir grafos direcionados acíclicos a partir de um grafo e de sua degeneração.

Seja $H = (V, E)$ um grafo direcionado acíclico. Denota-se por $\Delta^+(H)$ o valor

$$\Delta^+(H) = max\{d_H^+(v), v \in V(H)\}$$

e dizemos que $\Delta^+(H)$ é o grau máximo de saída no grafo direcionado $H$.

Seja $G$ um grafo simples com degeneração $d$ e seja $<$ uma ordem total sobre $V(G)$. Define-se o grafo direcionado $G_<$ por

$$\begin{aligned} V(G_<) &= V(G), \\ E(G_<) &= \{(u,v) \in V(G) \times V(G) \colon \{u,v\} \in E(G) \text{ e } u < v\}. \end{aligned}$$

Para criar um grafo $G_<$ a partir do grafo $G$ da Figura 9.2 direcionamos as arestas de $G$ da Figura 9.2 do vértice de menor número para o vértice de maior número como exemplificado na Figura 9.3.

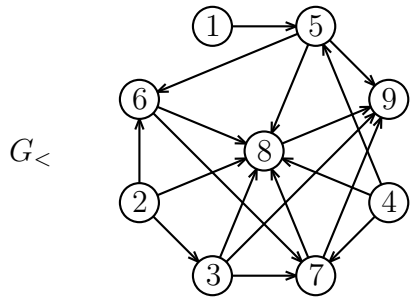


Figura 9.3: Grafo direcionado e vértices numerados pelo algoritmo degeneração($G$)

Outra forma de representar o grafo direcionado da Figura 9.3 damos na Figura 9.4.

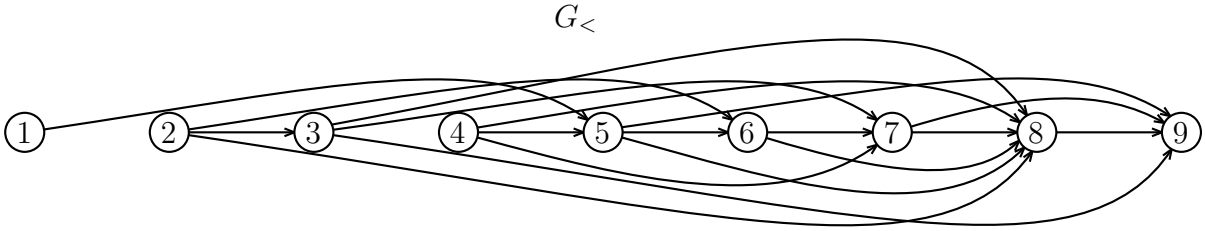


Figura 9.4: Grafo direcionado e vértices alinhados pelos seus números

Ou seja, a Figura 9.3 e a Figura 9.4 são representações do mesmo grafo direcionado $G_<$. Na Figura 9.4 os vértices foram representados alinhados seguindo a ordem total do vértice de menor número para o vértice de maior número da esquerda para a direita.

## 9.4 Degeneração ordenada

A partir daqui vamos restringir a forma de criar um grafo direcionado $G_<$ a partir de $G$ usando a degeneração ordenada, que é definida a seguir.

**Definição 9.6.** *Uma degeneração ordenada em um grafo simples $G = (V, E)$ de degeneração $d$ é uma ordem total $<$ sobre $V(G)$, tal que $\Delta^+(G_<) \leq d$.*

Ou seja, em uma degeneração ordenada cada vértice de $G_<$ precisa ter $d$ ou menos vizinhos de saída. O grafo $G_<$ da Figura 9.3 e da Figura 9.4 tem sua ordem total conforme uma degeneração ordenada, pois cada vértice tem 3 ou menos vizinhos de saída, ou seja, $\Delta^+(G_<) \leq 3$ e o grafo $G$ tem degeneração $d = 3$.

Daqui em diante o grafo $G_<$ é sempre gerado a partir de um grafo $G$ como na definição a seguir.

**Definição 9.7.** *O grafo $G_<$ é um grafo construído a partir do grafo $G$, tal que $\Delta^+(G_<) \leq d$, onde $d$ é degeneração de $G$.*

O vértice $v$ é *posterior* do vértice $w$ se $w < v$ em $V(G_<)$.

Para facilitar a leitura, reproduzimos uma cópia da Figura 9.4.

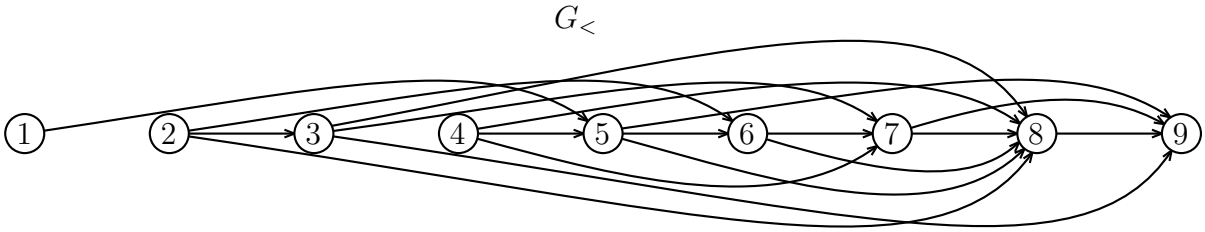


Figura 9.5: Grafo direcionado de acordo com a degeneração ordenada

Na forma de representação usada na Figura 9.5 (cópia da Figura 9.4) qualquer vértice tem seus vértices de saída sempre à sua direita. Daí, pela Figura 9.5 facilmente percebe-se o grau máximo de saída $\Delta^+(G_<)$ do grafo $G_<$ como sendo 3.

A relação entre degeneração $d$ e degeneração ordenada temos no seguinte resultado.

**Teorema 9.8.** *Um grafo $G$ tem degeneração $d$ se e somente se o grafo $G$ tem uma degeneração ordenada com $\Delta^+(G_<) \leq d$.*

*Demonstração.* Seja $G$ um grafo com degeneração $d$, o que significa que todo subgrafo de $G$ contém um vértice $v$ de grau no máximo $d$. Vamos construir $G_<$ a partir de $G$ e $d$: inicialmente fazemos $G_<$ com $V(G_<)$ e $E(G_<)$ sendo vazios. Tomamos o vértice $v$ de grau no máximo $d$ e adicionamos as arestas $(v, w)$ em $E(G_<)$, tal que $\{v, w\} \in E(G)$. Adicionamos $v$ em $V(G_<)$ como o maior dos vértices já existentes em $V(G_<)$. Removemos do grafo $G$ o vértice $v$ e as arestas incidentes em $v$. Como $G$ tem degeneração $d$, então a cada remoção de um vértice $v$ do grafo $G$ existe um vértice no subgrafo resultante com grau no máximo $d$. Repetimos este processo removendo um vértice de grau no máximo $d$ do grafo $G$ até $G$ ficar vazio, concluindo $V(G_<)$ com os mesmos vértices de $V(G)$. Assim temos uma ordem $<$ sobre $V(G)$, tal que $\Delta^+(G_<) \leq d$. Portanto $G$ tem uma degeneração ordenada.

Reciprocamente, seja $G$ o grafo simples com $G_<$ sendo o grafo direcionado com uma degeneração ordenada tendo $\Delta^+(G_<) \leq d$. Seja $H$ um subgrafo qualquer de $G$. Seja $v$ o vértice de $H$ que em $V(G_<)$ é o menor de todos vértices de $H$ em $V(G_<)$ ($v$ é o vértice de $H$ que está mais a esquerda na representação de $G_<$ que mencionamos acima). Como $\Delta^+(G_<) \leq d$ então $d^+_{G_<}(v)$ também é menor ou igual a $d$, isto é, $d^+_{G_<}(v) \leq d$. Daí $v$ tem $d$ ou menos vizinhos em $H$, onde $H$ foi definido com um subgrafo qualquer de $G$. Portanto $G$ tem degeneração $d$. □

O Teorema 9.5 garante que a degeneração é um limite para a clique máxima. Já o resultado a seguir garante um limite na quantidade de arestas do grafo, baseado em sua degeneração.

**Teorema 9.9.** *Um grafo com degeneração $d$ e $n$ vértices tem no máximo $d(n - \frac{d+1}{2})$ arestas.*

*Demonstração.* Seja $G$ um grafo com degeneração $d$ e $n$ vértices. Pelo Teorema 9.8 $G$ tem uma degeneração ordenada $G_<$. A quantidade de arestas em $G$ é a mesma que em $G_<$, pois em $G_<$ há apenas o direcionamento das arestas de $G$. Daí basta provar o máximo de arestas de $G_<$. Seja $S = \{v_1, v_2, \ldots, v_n\}$ a ordenação do vértices de $V(G_<)$. Assim, $v_i$ tem no máximo $d$ arestas de saída para vértices do conjunto $\{v_{i+1}, \ldots, v_n\}$ dos vértices posteriores a $v_i$ em $S$, para $1 \leq i \leq n$. Para que $v_i$ possa ter arestas para $d$ vertices o índice $i$ precisa ser menor ou igual $n-d$, pois para $i > n-d$ não há $d$ vertices posteriores. Daí que, $d$ é o máximo de arestas de saída dos $n-d$ vértices $v_1, \ldots, v_{n-d}$. Os vértices $v_{n-d+j}$ tem no máximo $d-j$ arestas de saída, $0 < j \leq d$. Assim, o máximo de arestas em $G$ é

$$(n-d)\,d + \sum_{i=1}^{d} d - i = (n-d)\,d + \frac{(d-1)\,d}{2} = d\,(n - \frac{d+1}{2}).$$

□

Degeneração e degeneração ordenada podem ser computadas pela estratégia gulosa simples de repetidamente ir removendo um vértice grau mínimo do grafo até o grafo ficar

vazio, como no algoritmo degeneração($G$) visto anteriormente e repetido aqui.

degeneração($G$)

$d \leftarrow 0$

Enquanto $G$ *não estiver vazio*

  Se $\delta(G) > d$

    $d \leftarrow \delta(G)$

  remova do grafo $G$ um vértice de grau mínimo (e suas arestas)

Devolva $d$

Assim, a degeneração é o máximo dos graus dos vértices do momento em que eles são removidos do grafo e a ordem na qual os vértices vão sendo removidos forma uma degeneração ordenada. Essa é a essência do algoritmo de [Batagelj and Zaversnik(2003)].

Adicionando no algoritmo degeneração($G$) uma lista $D$, tal que $D[i]$ contém uma lista dos vértices de grau $i$, podemos computar a degeneração de um grafo em tempo $O(m+n)$, onde $m$ é a quantidade de arestas e $n$ a quantidade de vértices do grafo.

## 9.5 Subgrafos de acordo com a degeneração ordenada

Nesta seção vamos construir subgrafos de $G$ a partir de $G_<$, onde $G$ é um grafo simples com degeneração $d$ e $G_<$ é o grafo direcionado acíclico visto anteriormente.

Seja $v$ um vértice de $V(G_<)$.

$N^+(v)$ é o conjunto dos vizinhos posteriores de $v$, isto é,

$$N^+(v) = \{w, (v, w) \in E(G_<)\}.$$

Sejam $v_1 < v_2 < \ldots < v_n$ os vértices de $V(G_<)$.

Pela Definição 9.7 o grafo $G_<$ é construído a partir de $G$ com cada vértice $v_i \in V(G_<)$ tendo no máximo $d$ vizinhos posteriores, para $i = 1, 2, \ldots, n$ e temos

$$|N^+(v_i)| \leq d.$$

Seja $V_i$ o conjunto dos vizinhos posteriores a $v_i$ incluindo $v_i$, ou seja,

$$V_i = \{v_i\} \cup N^+(v_i), \text{ para } i = 1, 2, \ldots, n.$$

Os subgrafos $G[V_1], G[V_2], \ldots, G[V_n]$ chamamos de *subgrafos de G de acordo com a degeneração ordenada.*

Copiamos a seguir as Figuras 9.2 e 9.4 para facilitar a visualização dos subgrafos de acordo com a degeneração ordenada.

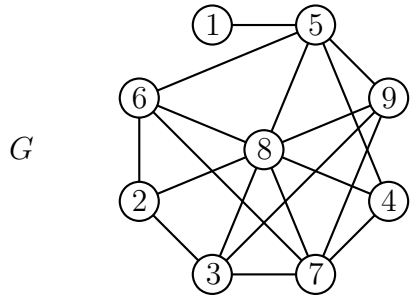


Figura 9.6: Grafo $G$ com vértices numerados pelo algoritmo degeneração($G$)

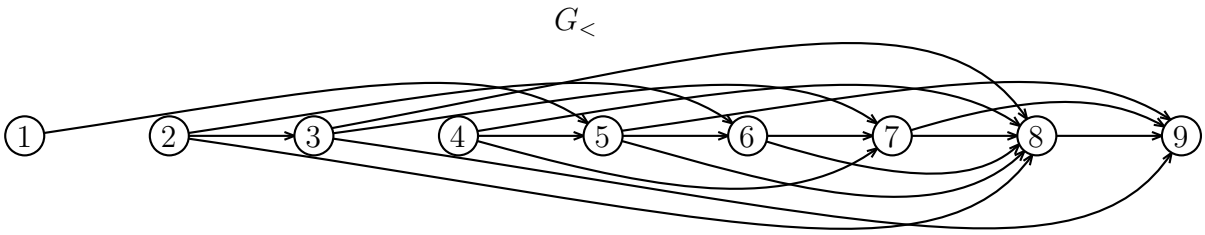


Figura 9.7: Grafo $G_<$ conforme a degeneração ordenada

Na Figura 9.8 representamos os subgrafos de $G$ de acordo com a degeneração ordenada em $G_<$, onde $G$ e $G_<$ são os grafos representados respectivamente nas Figuras 9.6 e 9.7.

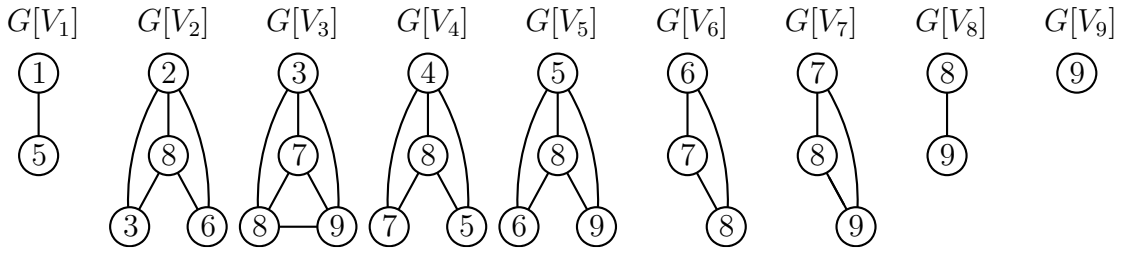


Figura 9.8: Subgrafos de $G$ de acordo com a degeneração ordenada

Usamos os subgrafos de acordo com a degeneração ordenada para encontrar uma Clique Máxima $S$, pois $S$ pertence a um desses subgrafos como vemos no resultado a seguir.

**Teorema 9.10.** *Se $S$ é uma Clique Máxima do grafo $G$, então $S$ pertence a um dos subgrafos $G[V_1], G[V_2], \ldots, G[V_n]$ de $G$, onde $n$ é o numero de vértices de $G$.*

*Demonstração.* Seja $G$ um grafo com $n$ vértices e seja $S$ uma Clique Máxima de $G$. Sejam $v_1 < v_2 < \ldots < v_n$ os vértices de $V(G_<)$ e sejam $G[V_1], G[V_2], \ldots, G[V_n]$ os subgrafos de $G$ de acordo com a degeneração ordenada. Seja $v \in S$ o menor dos vértices de $S$ em $V(G_<)$. Seja esse vértice $v$ o $x$-ésimo vértice de $(v_1, v_2, \ldots, v_n)$, ou seja, $v = v_x$ com $1 \leq x \leq n$. Como $v$ é o menor vértice entre os vértices de $S$, então todos os $S - v$ vizinhos de $v$ são posteriores de $v$ em $G_<$ (estão à direita de $v$). Portanto a Clique Máxima $S$ pertence ao subgrafo $G[V_x]$ e temos o teorema. □

O algoritmo CliqueMaximaPorSugrafos($G$) abaixo vem do Teorema 9.10: o algoritmo tem uma lista de subgrafos de acordo com a degeneração tal que em um desses subgrafos está a Clique Máxima de $G$. O algoritmo CliqueMaxima($H$) é qualquer algoritmo que encontre a clique máxima em $\mathcal{L}[i]$, que é subgrafo de $G$.

CliqueMaximaPorSugrafos($G$)

$\mathcal{L} \leftarrow (G[V_1], G[V_2], \ldots, G[V_n])$
$S \leftarrow \emptyset$
para $i = 1$ até $n$ faça
    $S' \leftarrow CliqueMaxima(\mathcal{L}[i])$
    Se $|S| < |S'|$
        $S \leftarrow S'$
Devolva $S$

## 9.6 O algoritmo Bron–Kerbosch e a degeneração ordenada

Em [Bron and Kerbosch(1973)] é apresentado um algoritmo para a Clique Máxima. Chamamos de *Bron–Kerbosch* ou *BK* o algoritmo de [Bron and Kerbosch(1973)] para a

Clique Máxima sem pivoteamento.

Denotamos por $N(v)$ o conjunto dos vizinhos de $v$, isto é,

$$N(v) = \{w \text{ tal que } \{v, w\} \in E(G)\}.$$

O algoritmo *BK* recebe um grafo $G$ como argumento. Após receber um grafo o algoritmo *BK* escolhe um vértice $v$ e inicia duas chamadas recursivas: em uma das chamadas o grafo $G - v$ é passado como argumento para *BK* e na outra chamada recursiva anota-se o vértice $v$ como parte da clique máxima passando como argumento o grafo $G[N(v)]$. Chamamos de *pivô* o vértice $v$ escolhido.

---

BK($G$)

---

$v \leftarrow$ um vertice de $G$

Se $G = \emptyset$
| Devolva $\emptyset$

$C \leftarrow \text{BK}(G - v)$

$C' \leftarrow \{v\} \cup \text{BK}(G[N(v)])$

Se $|C| > |C'|$
| Devolva $C$

Devolva $C'$

---

Em [Eppstein et al.(2010)Eppstein, Löffler, and Strash] foram relatados experimentos onde compararam-se os tempos de execução do algoritmo *BK* em relação as escolhas do pivô.

O principal algoritmo de [Eppstein et al.(2010)Eppstein, Löffler, and Strash] chamamos de *BK-Degen*.

O algoritmo *BK-Degen* é uma modificação do algoritmo *BK*, onde as escolhas dos vértices pivôs seguem a degeneração ordenada.

Em resumo, o algoritmo *BK-Degen* calcula uma degeneração ordenada do grafo $G$ da instância e coloca os vértices em uma lista $\mathcal{L}$ em ordem crescente. Daí escolhe os pivôs $v$ na sequência dada em $\mathcal{L}$ e faz chamadas do algoritmo *BK* passando como parâmetro o grafo $G - v$ e o grafo $G[N(v)]$.

O algoritmo *BK-Degen* tem tempo de execução $O(nd3^{d/3})$, onde $n$ é o número de vértices e $d$ é a degeneração do grafo.

## 9.7 Clique é $FPT$ quando é parametrizada pela degeneração

Consideremos o problema parametrizado Clique-Degen com o parâmetro sendo a degeneração do grafo, parâmetro que é calculado em tempo polinomial conforme visto anteriormente.

Clique-Degen

| | |
|---|---|
| Instância: | Um grafo $G$ e dois inteiros positivos $k$ e $d$. |
| Parâmetro: | A degeneração $d$ do grafo $G$. |
| Pergunta: | $G$ tem uma clique de tamanho $k$? |

O algoritmo *BK-Degen* com o tempo de execução $O(nd3^{d/3})$ nos dá o seguinte.

**Teorema 9.11.** Clique-Degen *pode ser resolvido em tempo* $f(d)n^2$*, onde* $d$ *é a degeneração e* $n$ *a quantidade de vértices do grafo da instância e* $f(d) = d3^{d/3}$*.*

Do Teorema 9.11 temos o seguinte resultado.

**Corolário 9.12.** Clique-Degen *é* $FPT$*.*

De modo mais preciso o algoritmo *BK-Degen* escolhe os pivôs $v$ na sequência da degeneração ordenada $\mathcal{L}$ até o $(n-d)$-ésimo vértice, pois os últimos $d$ vértices formam no máximo uma clique de tamanho $d$ como exemplificado na Figura 9.9. Mais precisamente o tempo de execução de *BK-Degen* é $(n-d+1).f(d)$, onde $n$ é o número de vértices e $d$ a degeneração do grafo e $f(d) = d3^{d/3}$.

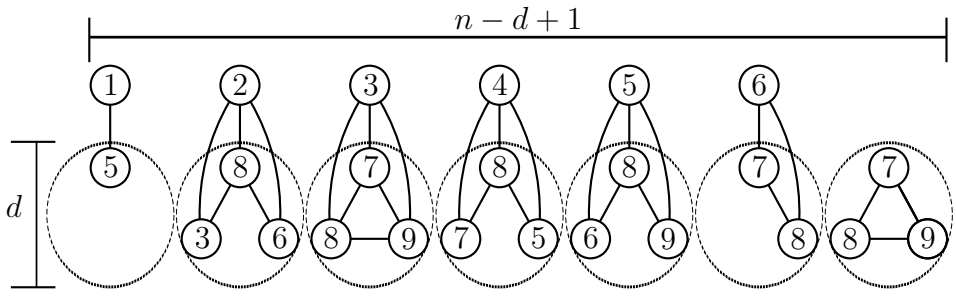


Figura 9.9: Tempo $(n-d+1)f(d)$ para encontrar a clique em um grafo

Os grafos obtidos em muitas aplicações apresentam valores baixos para a degeneração, como exemplo mostramos na Tabela 9.1 a seguir com base no BioGRID da versão 3.2.96.

| Organism (PPI network) | $n$ | $m$ | $\Delta$ | $d$ |
|---|---|---|---|---|
| Rattus norvegicus | 1710 | 2065 | 188 | 4 |
| Mus musculus | 5744 | 10738 | 294 | 9 |
| Caenorhabditis elegans | 3667 | 6927 | 527 | 10 |
| Arabisopsis thaliana | 6021 | 13464 | 438 | 12 |
| Drosophila melanogaster | 8168 | 37503 | 175 | 28 |
| Homo Sapiens | 16504 | 104832 | 9128 | 44 |
| Schizosaccharomyces pombe | 3659 | 50757 | 515 | 45 |
| Saccharomyces cerevisae | 6306 | 206693 | 2583 | 78 |

Tabela 9.1: Índices de oito grafos da rede PPI pela base de dados BioGRID da versão 3.2.96. Para cada rede listamos o número de vértices ($n$), o número de arestas ($m$), o grau máximo ($\Delta$) e a degeneração ($d$).

Vemos na Tabela 9.1 que cada rede PPI listada tem o número de vértices ($n$), o número de arestas ($m$) e o grau máximo ($\Delta$) significativamente maiores do que a degeneração ($d$).

## 9.8 Análise dos algoritmos *mcq*, *mcr*, *mcs* e *mcf*

Em [Carmo and Züge(2012)] encontramos uma exposição de algoritmos baseados em "branch and bound" para a solução exata do problema da clique máxima em uma forma unificada. Dessa exposição temos a semelhança dos algoritmos para a clique máxima no uso de um pré-processamento na ordem dos vértices, o que acontece nos principais algoritmos para a clique máxima.

Grande parte dos algoritmos para a clique máxima são variações do algoritmo *BK* de [Bron and Kerbosch(1973)] ou podem ser expressos como variações, o que é bastante esclarecido em [Carmo and Züge(2012)].

O algoritmo *BK* genérico em [Bron and Kerbosch(1973)] não especifica uma ordem na qual o algoritmo escolhe os vértices pivôs. Em [Carraghan and Pardalos(1990)] vemos uma ordem nos vértices pivôs privilegiando os vértices tenham menor grau.

Aos algoritmos para a clique máxima que são melhoramentos do *BK* e que privilegiam os vértices de menor grau na escolha dos pivôs chamamos de algoritmos em conformidade

com uma degeneração ordenada, pois ao escolher os pivôs em ordem não-descrescente dos graus dos vértices forma-se uma degeneração ordenada.

O algoritmo *mcq* em [Tomita and Seki(2003)] é uma modificação do *BK* e a ordem dos vértices é feita de forma decrescente nos graus dos vértices, porém o algoritmo *mcq* opera tomando da direita para a esquerda nessa ordenação, daí privilegiando os vértices de menor grau. Portanto o algoritmo *mcq* é um algoritmo em conformidade com uma degeneração ordenada.

Sejam *mcr* e *mcs* algoritmos respectivamente de [Tomita and Kameda(2007)] e de [Tomita et al.(2010)Tomita, Sutani, Higashi, Takahashi, and Wakatsuki].

A ordenação dos vértices em ambos os algoritmos *mcr* e *mcs* ocorre de modo parecido com o algoritmo *mcq* privilegiando os vértices de menor grau com a tomada dos pivôs ocorrendo em ordem não-descrescente nos graus dos vértices. Portanto, os algoritmos *mcr* e *mcs* são algoritmos em conformidade com uma degeneração ordenada.

Em [Eblen et al.(2011)Eblen, Phillips, Rogers, and Langston] foi apresentado o algoritmo *mcf* para o problema da clique máxima. Dado um grafo $G$ com $n$ vértices como entrada para o algoritmo *mcf*, um dos pré-processamentos *mcf* é um algoritmo guloso para guardar uma clique máxima como referência para posteriores cortes nas buscas.

O algoritmo *mcf* foi construído em relações diretas com os pré-processamentos e as reduções ao núcleo do problema como nos algoritmos para Cobertura por Vértices dados no Capítulo 6. O algoritmo genérico do *mcf* não está em conformidade com uma degeneração ordenada, pois não apresenta diferença em ordenar os vértices do grau menor para o grau maior ou vice-versa. Entretanto nos experimentos com o algoritmo genérico do *mcf* fica transparente como a ordenar do maior grau para o menor para os vértices pivôs causa perda de performance. Daí, consideramos o algoritmo *mcf* como aquele no qual a tomada dos pivôs esteja em conformidade com uma degeneração ordenada.

Portanto os algoritmos *mcq*, *mcr*, *mcs* e *mcf* aplicados ao problema parametrizado Clique-Degen com parâmetro na degeneração do grafo da instância também resultam que Clique-Degen é $FPT$ tanto como o algoritmo *BK-Degen* com o tempo de execução

$O(nd3^{d/3})$.

Como os algoritmos *mcq*, *mcr*, *mcs* e *mcf* para o problema da clique máxima estão em conformidade com uma degeneração ordenada quando em relação à degeneração $k$ dos grafos da instância, então os algoritmos *mcq*, *mcr*, *mcs* e *mcf* tem tempo de execução $f(k)n^{\alpha}$, com $\alpha > 0$.

Assim, quando a clique máxima é analisada em relação à degeneração temos os algoritmos *BK-Degen*, *mcq*, *mcr*, *mcs* e *mcf* trazendo a tratabilidade por parâmetro fixo para o problema Clique-Degen, ou seja,

$$\text{Clique-Degen} \in FPT.$$

# REFERÊNCIAS

[Alber et al.(2003)Alber, Fernau, Niedermeier, Fan, Fellows, Rosamond, and Stege] Jochen Alber, Henning Fernau, Rolf Niedermeier, Hongbing Fan, Michael R. Fellows, Fran Rosamond, and Ulrike Stege. A refined search tree technique for dominating set on planar graphs, 2003.

[Balasubramanian et al.(1998)Balasubramanian, Fellows, and Raman] R. Balasubramanian, Michael R. Fellows, and Venkatesh Raman. An improved fixed-parameter algorithm for vertex cover. *Information Processing Letters*, 65(3):163–168, February 1998. ISSN 0020-0190 (print), 1872-6119 (electronic).

[Batagelj and Zaversnik(2003)] Vladimir Batagelj and Matjaz Zaversnik. An $o(m)$ algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003. URL `http://arxiv.org/abs/cs.DS/0310049`.

[Bondy and Murty(2008)] J.A. Bondy and U.S.R Murty. *Graph Theory*. Springer Publishing Company, Incorporated, 2008. ISBN 978-3642142789.

[Bron and Kerbosch(1973)] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, September 1973. ISSN 0001-0782. doi: http://dx.doi.org/10.1145/362342.362367. URL `http://dx.doi.org/10.1145/362342.362367`.

[Buss and Goldsmith(1993)] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within p. *SIAM J. Comput.*, 22(3):560–572, June 1993. ISSN 0097-5397. doi: 10.1137/0222038. URL `http://dx.doi.org/10.1137/0222038`.

[Cao et al.(2010)Cao, Chen, and Liu] Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set, new measure and new structures. *CoRR*, abs/1004.1672, 2010. URL `http://arxiv.org/abs/1004.1672`.

[Carmo and Züge(2012)] Renato Carmo and Alexandre Züge. Branch and bound algorithms for the maximum clique problem under a unified framework. *Journal of the Brazilian Computer Society*, 18(2):137–151, December 2012. ISSN 0104-6500. doi: 10.1007/s13173-011-0050-6. URL http://dx.doi.org/10.1007/s13173-011-0050-6.

[Carraghan and Pardalos(1990)] Randy Carraghan and Panos M. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9(6), 1990. doi: http://dx.doi.org/DOI:\%2010.1016/0167-6377(90)90057-C. URL http://dx.doi.org/DOI:%2010.1016/0167-6377(90)90057-C.

[Chandran and Grandoni(2005)] L. Sunil Chandran and Fabrizio Grandoni. Refined memorization for vertex cover. *Inf. Process. Lett.*, 93(3):125–131, 2005. ISSN 0020-0190. doi: 10.1016/j.ipl.2004.10.003. URL http://dx.doi.org/10.1016/j.ipl.2004.10.003.

[Chen et al.(2001)Chen, Kanj, and Jia] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280 – 301, 2001. ISSN 0196-6774. doi: http://dx.doi.org/10.1006/jagm.2001.1186. URL http://www.sciencedirect.com/science/article/pii/S0196677401911861.

[Chen et al.(2006)Chen, Kanj, and Xia] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In *Proceedings of the 31st international conference on Mathematical Foundations of Computer Science*, MFCS'06, pages 238–249, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-37791-3, 978-3-540-37791-7. doi: 10.1007/11821069_21. URL http://dx.doi.org/10.1007/11821069_21.

[Downey(1995)] R. Downey. Using symbolic techniques to find the maximum clique in very large sparse graphs. *Theoretical Computer Science*, 141(1-2):109–131, April 1995. ISSN 03043975. doi: 10.1016/0304-3975(94)00097-3. URL http://portal.

acm.org/citation.cfm?id=787456. title="Fixed-parameter tractability and completeness II: On completeness for W[1]".

[Downey and Fellows(1999)] R.G. Downey and M.R. Fellows. *Parameterized complexity*, volume 5. Springer New York, 1999.

[Downey and Fellows(1992)] Rod Downey and Michael Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:161–178, 1992.

[Downey and Fellows(1995)] Rod G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM J. Comput.*, 24(4):873–921, August 1995. ISSN 0097-5397. doi: 10.1137/S0097539792228228. URL http://dx.doi.org/10.1137/S0097539792228228.

[Downey and Fellows(1994)] Rodney G. Downey and Michael R. Fellows. Parameterized computational feasibility. In *Feasible Mathematics II*, pages 219–244. Birkhauser, 1994.

[Eblen et al.(2011)Eblen, Phillips, Rogers, and Langston] John D. Eblen, Charles A. Phillips, Gary L. Rogers, and Michael A. Langston. The maximum clique enumeration problem: algorithms, applications and implementations. In *Proceedings of the 7th international conference on Bioinformatics research and applications*, ISBRA'11, pages 306–319, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21259-8. URL http://dl.acm.org/citation.cfm?id=2009164.2009194.

[Eppstein et al.(2010)Eppstein, Löffler, and Strash] David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in sparse graphs in near-optimal time, June 2010. URL http://arxiv.org/abs/1006.5440.

[Fellows(1988)] Michael R. Fellows. On the complexity of vertex set problems. Technical report, Computer Science Department, University of New Mexico, 1988.

[Fellows and Jansen(2013)] Michael R. Fellows and Bart M. P. Jansen. FPT is characterized by useful obstruction sets. *CoRR*, abs/1305.3102, 2013. URL `http://arxiv.org/abs/1305.3102`.

[Fellows and Langston(1987)] Michael R. Fellows and Michael A. Langston. Nonconstructive advances in polynomial-time complexity. *Information Processing Letters*, 26(3):155–162, 1987. doi: 10.1016/0020-0190(87)90054-8. URL `http://www.sciencedirect.com/science/article/pii/0020019087900548`.

[Garey and Johnson(1979)] M.R. Garey and D.S. Johnson. *Computers and intractability*. Freeman San Francisco, 1979.

[Johnson(1987)] David S. Johnson. The np-completeness column: An ongoing guide. *J. Algorithms*, 8:285–303, 1987. ISSN 0196-6774. doi: 10.1016/0196-6774(87)90043-5. URL `http://www.sciencedirect.com/science/article/pii/0196677487900435`.

[Karp(1972)] R.M. Karp. Reducibility among combinatorial problems. *Complexity of computer computations: proceedings*, page 85, 1972.

[Robertson and Seymour(1995)] Neil Robertson and P. D. Seymour. Graph minors. XIII. the disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. ISSN 0095-8956. doi: 10.1006/jctb.1995.1006. URL `http://dx.doi.org/10.1006/jctb.1995.1006`.

[Tomita and Kameda(2007)] Etsuji Tomita and Toshikatsu Kameda. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *Journal of Global Optimization*, 37(1):95–111, January 2007. ISSN 0925-5001. doi: http://dx.doi.org/10.1007/s10898-006-9039-7. URL `http://dx.doi.org/10.1007/s10898-006-9039-7`.

[Tomita and Seki(2003)] Etsuji Tomita and Tomokazu Seki. *An Efficient Branch-and-Bound Algorithm for Finding a Maximum Clique*. Springer Berlin/Heidelberg, 2003. URL `http://www.springerlink.com/content/7jbjyglyqc8ca5n9`.

[Tomita et al.(2010)Tomita, Sutani, Higashi, Takahashi, and Wakatsuki] Etsuji Tomita, Yoichi Sutani, Takanori Higashi, Shinya Takahashi, and Mitsuo Wakatsuki. A simple and faster branch-and-bound algorithm for finding a maximum clique. In Md Rahman and Satoshi Fujita, editors, *WALCOM: Algorithms and Computation*, volume 5942, chapter 18, pages 191–203. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-11439-7. doi: 10.1007/978-3-642-11440-3_18. URL `http://dx.doi.org/10.1007/978-3-642-11440-3_18`.

[Weihe(1998)] K. Weihe. Covering trains by stations or the power of data reduction. In R. Battiti and A. A. Bertossi, editors, *Algorithms and Experiments ALEX 98*, pages 1–8. `http://rtm.science.unitn.it/alex98/proceedings.html`, 1998.