**Circuitos e Caminhos Pares/Ímpares
em Grafos e Digrafos**

Algoritmos e Complexidade Computacional

Ronaldo Fumio Hashimoto

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO GRAU DE MESTRE
EM
MATEMÁTICA APLICADA

Área de Concentração: **Ciência da Computação**
Orientador: **Profa. Dra. Yoshiko Wakabayashi**

Durante a elaboração deste trabalho o autor recebeu apoio financeiro da CAPES e CNPq

—São Paulo, março de 1994—

Circuitos e Caminhos Pares/Ímpares em Grafos e Digrafos

Algoritmos e Complexidade Computacional

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Ronaldo Fumio Hashimoto
e aprovada pela comissão julgadora.

São Paulo, 25 de maio de 1994.

Banca examinadora:

- Profa. Dra. Yoshiko Wakabayashi (orientadora) - IME-USP
- Prof. Dr. José Augusto Ramos Soares - IME-USP
- Profa. Dra. Célia Picinin de Mello - IMECC-UNICAMP

Aos meus pais

Agradecimentos

Tenho certeza de que há muitas pessoas que contribuíram para a realização desta dissertação, e que não é possível enumerá-las todas aqui. Mas, gostaria de agradecer de uma forma muito especial:

À Yoshiko, pela sugestão do problema, pela excelente orientação, pelo apoio nos momentos difíceis, pela paciência e compreensão durante todo o mestrado.

Ao Álvaro, ao Flávio, ao Haroldo, à Maria Ângela, à Yoshiko (olha ela aí novamente), pelas discussões em nossos seminários, que foram valiosas para a elaboração deste trabalho.

Ao Alair, por ser uma companhia importante.

Aos professores do Departamento de Ciência da Computação, por terem contribuído para a minha formação acadêmica.

À minha família, pela compreensão, pelo apoio e pela presença constante.

Ao José, que me mostrou que todo trabalho é digno de ser bem feito.

À Mariana, pela demonstração de carinho e pela companhia durante este mestrado.

Aos meus amigos, que souberam me apoiar durante a realização deste trabalho.

E finalmente, aos membros da banca examinadora, pelas sugestões e contribuições que enriqueceram bastante esta dissertação.

A todos, muito obrigado!

Resumo

Neste trabalho estudamos vários problemas sobre circuitos e caminhos em grafos e em digrafos. Consideramos aqui três classes de problemas: de *existência*, de *busca* e de *busca de um mínimo*. Para cada uma dessas classes investigamos os casos em que o objeto em questão é um circuito par/ímpar ou um caminho par/ímpar. Discutimos questões referentes à complexidade computacional desses problemas e apresentamos algoritmos polinomiais para resolver vários deles. A maioria dos resultados que apresentamos foram coletados da literatura, incluindo uma resenha atualizada sobre o problema da existência de circuitos pares em digrafos (um problema cuja complexidade computacional continua desconhecida há vinte anos).

Nossa principal contribuição a este estudo é o desenvolvimento de um algoritmo linear para encontrar circuitos ímpares em digrafos. Descrevemos também algoritmos alternativos (não necessariamente de melhor complexidade) para alguns dos problemas.

Abstract

In this work we study problems on graphs and digraphs concerning cycles and paths of prescribed parity. Let H be either an even/odd cycle or an even/odd path. We consider three classes of problems on the presence of H : existence, search and search for one with the smallest possible length. These problems are investigated from the algorithmic point of view. We discuss their computational complexity and present polynomial algorithms to solve a number of them. Most of the results were collected from the literature, including an updated survey on the problem of existence of even cycles in digraphs (a problem whose computational complexity status has been open for twenty years).

Our main contribution to this study is a linear algorithm to find an odd cycle in a digraph. We also describe alternative (not necessarily of better complexity) algorithms to solve some of the problems.

Introdução

O conceito de circuito tem um papel fundamental na Teoria dos Grafos, e são vários os artigos e monografias que tratam desse assunto. A literatura a respeito de circuitos em grafos é bem mais extensa do que para o caso de digrafos. Há alguns motivos de ordem prática e também teórica que poderiam justificar este fato, mas talvez a principal razão é que o estudo de circuitos em digrafos é consideravelmente mais difícil.

O conceito de caminho também é fundamental na Teoria dos Grafos e em alguns aspectos está intimamente relacionado com o conceito de circuito. É imediato que algoritmos polinomiais para resolver certos problemas sobre caminhos implicam algoritmos polinomiais para os correspondentes problemas sobre circuitos.

Motivados pelos aspectos algorítmicos de problemas sobre circuitos e caminhos, em especial quanto à existência e busca de tais estruturas, o ponto de partida deste trabalho foi a leitura de uma resenha de BERMOND E THOMASSEN [1981] onde são mencionados vários problemas e conjecturas sobre circuitos em digrafos. Um desses problemas é decidir a existência de circuitos pares em digrafos, que aparentemente foi proposto por D. H. YOUNGER, por volta de 1975. À primeira vista causou-nos surpresa que, apesar de ser fácil o problema de decidir se um digrafo contém um circuito ímpar, não se descobriu até hoje se é fácil (polinomial) ou difícil ($\mathcal{NP}$ -completo), decidir se um digrafo contém um circuito par. A literatura revela que grandes esforços têm sido dispensados na tentativa de se resolver esse problema. Vale aqui observar que o problema da existência de circuitos pares em digrafos tem aplicações em Álgebra Linear, e em particular na solução de certos problemas (de matrizes) de interesse em economia matemática.

A leitura desse resenha e de outros artigos trouxe à tona uma certa “assimetria” dos problemas em relação ao caso *grafos* $\times$ *digrafos* e ao caso *par* $\times$ *ímpar* (por exemplo, no caso de digrafos, o problema de decidir a existência de um caminho par entre dois vértices distintos é $\mathcal{NP}$ -completo; já no caso de grafos existem algoritmos polinomiais que resolvem esse mesmo problema). Esta observação contribuiu para que decidíssemos pesquisar os tópicos aqui abordados, principalmente por não termos encontrado na literatura uma abordagem completa e atualizada que respondesse satisfatoriamente todas as questões referentes a esses problemas.

Neste trabalho estudamos três categorias de problemas sobre caminhos e circuitos pares/ímpares em grafos e em digrafos: existência, busca e busca de um mínimo. Com relação a esses problemas, procuramos responder questões referentes à sua complexidade computacional, exibindo algoritmos polinomiais para alguns casos e provando a $\mathcal{NP}$ -completude de outros. A definição desses problemas, e também da notação utilizada nesta dissertação, encontra-se no **primeiro capítulo**.

No **segundo capítulo** estudamos os problemas sobre circuitos ímpares em grafos e em digrafos, descrevendo algoritmos lineares para os problemas de existência e de busca. O algoritmo linear para encontrar um circuito ímpar em digrafos (descrito neste capítulo) não foi encontrado na literatura. Uma implementação desse algoritmo (em linguagem C) está no **Apêndice A**. Para encontrar um circuito ímpar mínimo num grafo G , apresentamos algoritmos de complexidade $O(|V_G| |A_G|)$.

Estudamos, no **terceiro capítulo**, os problemas sobre circuitos pares em grafos. Descrevemos algoritmos lineares para o problema de existência e de busca, e um algoritmo de complexidade $O(|A_G|^2)$ para encontrar um circuito par mínimo num grafo G .

Os problemas envolvendo circuitos pares em digrafos são discutidos no **quarto capítulo**. Apresentamos uma resenha sobre o problema da existência de circuitos pares em digrafos, onde mencionamos vários problemas equivalentes e alguns resultados correlatos. Apresentamos também um algoritmo polinomial, que encontra um circuito par em digrafos planares, obtido recentemente por THOMASSEN [1993].

No **quinto capítulo**, estudamos problemas sobre caminhos pares/ímpares em grafos e em digrafos. Apresentamos vários resultados devidos a LAPAUGH E PAPADIMITRIOU [1984] para os problemas de existência e de busca de um mínimo. Em especial destacamos a descrição de um algoritmo linear para decidir a existência de um caminho par em grafos (entre dois vértices distintos) e a prova de que o respectivo problema em digrafos é $\mathcal{NP}$ -completo.

Finalmente, no **Apêndice B** apresentamos um resumo, indicando para cada um dos problemas, os resultados que foram mencionados neste trabalho.

Índice

1	Conceitos básicos e definição dos problemas	1
1.1	Introdução	1
1.2	Conceitos e notações básicas	1
1.2.1	Grafos e digrafos	2
1.3	Definição dos problemas	6
1.3.1	Problemas de existência	7
1.3.2	Problemas de busca	8
1.3.3	Problemas de busca de um mínimo	9
2	Circuitos ímpares	11
2.1	Introdução	11
2.2	Existência de um circuito ímpar	11
2.2.1	O PCirImpar para grafos	12
2.2.2	O PCirImpar para digrafos	13
2.3	Busca de um circuito ímpar	15
2.3.1	O PEncCirImpar para grafos	15
2.3.2	O PEncCirImpar para digrafos	16
2.4	Busca de um circuito ímpar mínimo	26
2.5	Conclusão	29
3	Circuitos pares em grafos	31
3.1	Introdução	31
3.2	Existência de circuitos pares	31
3.3	Busca de um circuito par	32
3.4	Busca de um circuito par mínimo	33
3.5	Conclusão	34
4	Circuitos pares em digrafos	35
4.1	Introdução	35
4.2	Problemas equivalentes	35
4.2.1	O problema do circuito de peso par	36
4.2.2	O problema do digrafo par	36

4.2.3	O problema do circuito duplo-fraco ímpar	40
4.2.4	O problema do esquema de Polya	54
4.2.5	O problema do permanente-determinante	60
4.2.6	O problema da L -Matriz	62
4.2.7	O problema da Orientação Pfaffiana	63
4.3	Alguns resultados correlatos	70
4.3.1	Coloração de hipergrafos	70
4.3.2	Resultados referentes ao grau dos vértices do digrafo	72
4.3.3	Resultados referentes à conectividade do digrafo	72
4.4	Aspectos algorítmicos	73
4.4.1	Circuitos pares em digrafos planares	76
5	Caminhos pares e caminhos ímpares	89
5.1	Introdução	89
5.2	Existência de caminhos pares em grafos	89
5.2.1	Preliminares	90
5.2.2	Algoritmo para PCamPar via construção de Edmonds	93
5.2.3	O algoritmo de LaPaugh e Papadimitriou	95
5.3	Existência de caminhos pares em digrafos	99
5.4	Busca de caminhos pares em grafos	100
5.4.1	Redução de PEncCamPar para PFaseCompleta	100
5.4.2	Redução de PEncCamPar para o problema de encontrar dois cami- nhos disjuntos	102
5.4.3	Redução de PEncCamPar para o problema do triângulo	111
5.5	Busca de caminhos pares mínimos	112
5.5.1	Redução de PCamParMin para o problema do emparelhamento per- feito de peso máximo	112
5.5.2	Redução de PCamParMin para PFaseCompleta	113
5.6	Conclusão	115
	Conclusão	117
A	Implementação do algoritmo Encontra_Circuito_Ímpar_Digrafo	119
B	Tabelas dos problemas	125
B.1	Circuitos ímpares em grafos	125
B.2	Circuitos ímpares em digrafos	125
B.3	Circuitos pares em grafos	126
B.4	Circuitos pares em digrafos	126
B.5	Caminhos pares/ímpares em grafos	126
B.6	Caminhos pares/ímpares em digrafos	126

Lista de Figuras

2.1	Um digrafo D e o respectivo digrafo D' .	19
2.2	Exemplo de uma execução de Busca_Profundidade_Diferente no digrafo D da figura 2.1.	21
4.1	Digrafo simétrico D obtido a partir de um grafo G .	41
4.2	O 5-circuito duplo (S_5^*) e um 5-circuito duplo-fraco.	42
4.3	Um bloco B e seus componentes adjacentes $B_1, B_2, \dots, B_k$.	43
4.4	Construção dos digrafos H_1 e H_2 a partir do digrafo D .	45
4.5	H_1 ou H_2 é um digrafo par.	46
4.6	Caso 1 do Fato 2.	47
4.7	Caso 2 do Fato 2.	48
4.8	Figuras dos casos 1, 2 e 3 do Fato 6.	50
4.9	Um circuito C de comprimento 2 do digrafo D , um bloco B do grafo subjacente a $D - A_C$ e seus componentes adjacentes B_1, B_2 .	52
4.10	Um exemplo de uma atribuição de pesos 0/1 aos arcos do circuito duplo S_4^* , tal que todos os seus circuitos tenham peso ímpar.	53
4.11	Redução para os problemas do circuito duplo (ou do circuito duplo ímpar, ou do 3-circuito duplo).	54
4.12	Trocando v_i e v_j no circuito C da cobertura singular C_B , obteremos uma nova cobertura singular C'_B que contém um novo circuito C' e um novo laço (v_i, v_i) .	57
4.13	Um hipergrafo H e uma 3-coloração (S_1, S_2, S_3) de H .	71
4.14	Um digrafo 2-conexo sem circuitos pares.	72
4.15	Um exemplo de um digrafo fortemente conexo sem circuitos pares e com um número exponencial de circuitos ímpares.	74
4.16	Ilustração para o Lema 4.4.2.	78
4.17	Ilustração para o Lema 4.4.4.	79
4.18	Um α -desvio estritamente minimal B e um caminho minimal com respeito a α .	80
4.19	Construção do caminho P' .	81
4.20	Um caminho de substituição $L[x, y]$.	82
4.21	Um caminho crucial.	83

5.1	Um exemplo da redução de Edmonds.	93
5.2	Um grafo G e o correspondente grafo T_G	96
5.3	Exemplo da construção de G' (Lema 5.2.8).	97
5.4	Um digrafo D e o correspondente digrafo D' (Teorema 5.3.1).	99
5.5	Figura para o Lema 5.4.1.	104
5.6	O vértice z se repete em T' e não está em P'	105

Lista de Algoritmos

Verifica_Circuito_Ímpar_Grafo(G)	12
Verifica_Circuito_Ímpar_Digrafo(D)	14
Encontra_Circuito_Ímpar_Grafo(G)	15
Passeio_para_Circuito(P)	17
Busca_Profundidade_Diferente(D)	20
Encontra_Circuito_Ímpar_Digrafo(D)	23
Circuito_Ímpar(D)	25
Circuito_Ímpar_Mínimo_Digrafo(D)	28
Circuito_Ímpar_Mínimo_Grafo(G)	28
Circuito_Ímpar_Mínimo_Grafo_Seidel(G)	28
Verifica_Circuito_Par_Grafo(G)	32
Encontra_Circuito_Par_Grafo(G)	32
Circuito_Par_Mínimo_Grafo(G)	33
Karmakar(D, v_i, l)	74
Circ_Peso_Par_Con(D)	77
PCircPPar_Planar(D)	87
Encontra_Emparelhamento_Máximo(G)	92
Verifica_Caminho_Par_Via_Fase_Completa(G, s, t)	94
Verifica_Caminho_Par_Via_Biconexos(G, s, t)	98
Caminho_Par_Via_Alternante($\overline{G}, P$)	100
Encontra_Caminho_Par_Via_Pape_Conrad(G, s, t)	101
Encontra_Caminho_Par_Via_Fase_Completa(G, s, t)	101
Extrai_Dois_Caminhos(G, P, u, v)	103
Busca_Trilha(G, P, s, t)	107
Dois_Caminhos_Trilha(G, P, T, s, t)	109
Encontra_Caminho_Par_Via_Dois_Caminhos(G, s, t)	110
Encontra_Caminho_Par_Via_Triângulo(G, s, t)	111
Caminho_Par_Mínimo_Via_Emparelhamento(G, s, t)	113
Caminho_Par_Mínimo_Via_Fase_Completa(G, s, t)	114

Capítulo 1

Conceitos básicos e definição dos problemas

1.1 Introdução

Neste capítulo apresentaremos alguns conceitos básicos da Teoria dos Grafos e estabeleceremos a notação a ser utilizada. Definiremos também os vários tipos de problemas que serão objeto de estudo desta dissertação.

Assumimos aqui que o leitor tenha alguma familiaridade com conceitos básicos da Teoria da Complexidade Computacional, razão pela qual não incluímos nenhuma seção sobre tais conceitos. Ademais, a terminologia utilizada segue exatamente a de GAREY E JOHNSON [1979].

No que se refere à medida da complexidade computacional dos algoritmos, ainda que algumas vezes não seja mencionada explicitamente, são todas relativas ao tempo.

1.2 Conceitos e notações básicas

Nesta dissertação, na maioria das vezes utilizaremos letras maiúsculas do alfabeto para representar conjuntos e letras minúsculas para representar os seus elementos.

Sejam A e B dois conjuntos quaisquer. Denotaremos por $A \cup B$, $A - B$, $A \cap B$ e $A \oplus B$, a **união**, a **diferença**, a **interseção** e a **diferença simétrica** de A e B , respectivamente. Quando um dos conjuntos for unitário, este será denotado sem as chaves; além disso, no caso da união, usaremos o símbolo $+$ em vez de $\cup$. Assim $A + b$ simplifica a notação $A \cup \{b\}$.

1.2.1 Grafos e digrafos

Um **grafo** é um par ordenado $G = (V_G, A_G)$, onde V_G é um conjunto finito, não vazio, de elementos chamados **vértices** e A_G é um multi-conjunto de elementos chamados **arestas**. Cada aresta é um par não-ordenado $\{u, v\}$ de vértices. Cabe aqui observar que, por ser um multi-conjunto, A_G permite elementos repetidos.

Um **digrafo** $D = (V_D, A_D)$ é definido da mesma forma que um grafo, exceto que o multi-conjunto A_D é constituído de pares ordenados de vértices. Um par ordenado de vértices u, v é denotado por (u, v) . Geralmente, chamaremos os elementos de A_D de **arcos**.

As arestas repetidas (arcos repetidos) de um grafo (digrafo) serão chamadas de **arestas múltiplas** (arcos múltiplos). Uma aresta $\{x, y\}$ (arco (x, y)) poderá ser indicada simplesmente por xy , quando não houver perigo de confusão. Um **laço** é uma aresta (arco) da forma vv . Chamaremos de **grafo simples** (**digrafo simples**) um grafo (digrafo) sem laços e sem arestas múltiplas (arcos múltiplos). Às vezes é conveniente dar um nome a uma aresta (arco) uv , que representaremos por letras gregas minúsculas; nesse caso escrevemos $\alpha = uv$ para indicar que α é o nome da aresta (do arco) uv . Para denotar grafos (digrafos), utilizaremos, na maioria dos casos, as letras maiúsculas G ou H (G, H ou D). Se H é um grafo (digrafo), denotaremos por V_H o seu conjunto de vértices e por A_H o seu multi-conjunto de arestas (arcos). Na análise da complexidade de um algoritmo, cuja entrada é um grafo G , assumiremos que $|V_G| = n$ e $|A_G| = m$.

Se $\alpha = uv$ é uma aresta (arco), então os vértices u e v são chamados **extremos**, ou **pontas**, de α . Dizemos também que α é uma aresta (arco) **incidente** nos vértices u e v . Nesse caso, dizemos que um dos vértices é **adjacente** ao outro e que ambos são **vértices adjacentes** (ou **vizinhos**). O conjunto dos vértices adjacentes a v é denotado por $Adj(v)$. O **grau** de um vértice v em um grafo G é a quantidade de arestas incidentes em v , contando os laços duas vezes e é denotado por $g_G(v)$ ou simplesmente $g(v)$.

Seja D um digrafo. Dizemos que um arco (u, v) em D é um **arco orientado** de u para v e que esse arco sai de u e **entra** em v . O **grau de saída** (**grau de entrada**) de um vértice v em D , é o número de arcos em D que saem de v (entram em v) e é denotado por $g_D^+(v)$ ($g_D^-(v)$), ou simplesmente $g^+(v)$ ($g^-(v)$). Dizemos que um digrafo é **k -regular**, se todos os seus vértices têm grau de entrada e grau de saída exatamente igual a k . Se V_1 e V_2 é uma partição de V_D , então dizemos que um arco $(u, v) \in A_D$ tal que $u \in V_1$ e $v \in V_2$ é um **arco de V_1 para V_2** .

Se D é um digrafo, então o grafo G obtido a partir de D , retirando as orientações de seus arcos (ou seja, substituindo cada par ordenado de A_D por um par não ordenado) é chamado **grafo subjacente a D** .

Definimos como **matriz de adjacência** $A = (a_{u,v})$, de um grafo (digrafo) G , a matriz cujas linhas e colunas são indexadas pelos vértices de G e $a_{u,v}$ é a quantidade de arestas $\{u, v\}$ (arcos (u, v)) em A_G .

Um grafo (digrafo) H é um **subgrafo** (**subdigrafo**) de um grafo (digrafo) G se $V_H \subseteq V_G$ e $A_H \subseteq A_G$. Seja V um subconjunto não vazio de V_G . Denotamos por $G - V$ o **subgrafo** (**subdigrafo**) de G , obtido retirando de G os vértices que estão em V , juntamente com as arestas (os arcos) incidentes nesses vértices. O **grafo** (**digrafo**) **induzido por** V , denotado por $G[V]$, é o subgrafo (subdigrafo) $G - (V_G - V)$. Analogamente, se A é um subconjunto não vazio de A_G , o **grafo** (**digrafo**) $G - A$ é o subgrafo (subdigrafo) de G , obtido retirando as arestas (os arcos) que estão em A . O **grafo** (**digrafo**) **induzido por** A , denotado por $G[A]$, é o subgrafo (subdigrafo) $G - (A_G - A)$.

A **união** de dois grafos (digrafos) G_1 e G_2 , denotado por $G_1 \cup G_2$, é o grafo (digrafo) cujo conjunto de vértices é $V_{G_1} \cup V_{G_2}$ e cujo conjunto de arestas (arcos) é $A_{G_1} \cup A_{G_2}$. Analogamente, a **interseção** de dois grafos (digrafos) G_1 e G_2 , denotado por $G_1 \cap G_2$, é o grafo (digrafo) cujo conjunto de vértices é $V_{G_1} \cap V_{G_2}$ e cujo conjunto de arestas (arcos) é $A_{G_1} \cap A_{G_2}$. A operação de **subdivisão** ou **bipartição** de uma aresta (um arco) $\alpha = uv$ de um grafo (digrafo) G consiste em substituir essa aresta (esse arco) por duas novas arestas (dois novos arcos) ux e xv , sendo x um novo vértice. Mais formalmente, o grafo (digrafo) H que resulta da subdivisão de uma aresta (arco) $\alpha = uv$ de G é tal que $V_H = V_G + x$ e $A_H = (A_G - uv) \cup \{ux, xv\}$, sendo $x \notin V_G$. Partindo-se de G e repetindo-se a operação de subdivisão de uma aresta (arco) nos sucessivos grafos (digrafos) que são obtidos, obtém-se um grafo (digrafo) H que é dito ser obtido de G através de **sucessivas subdivisões** de arestas (arcos). A **subdivisão de um vértice** v de um digrafo D consiste em obter um novo digrafo H substituindo esse vértice por dois novos vértices v_1 e v_2 e um arco v_1v_2 . Além disso, todos os arcos que entravam em v (respectivamente, saíam de v), entram em v_1 (respectivamente, saem de v_2) no novo digrafo H . O grafo (digrafo) obtido pela **contração** da aresta (arco) uv é o grafo (digrafo) H tal que $V_H = V_G - u$ e A_H consiste de todas as arestas (arcos) de G contidas (contidos) em $V_G - u$, juntamente com todas as arestas (arcos) xv , onde xu é uma aresta (arco) em G diferente de α .

Um **passeio** em um grafo (digrafo) G é uma seqüência $P = [v_0, \alpha_1, v_1, \alpha_2, \dots, v_{k-1}, \alpha_k, v_k]$, onde $v_i \in V_G$ ($0 \leq i \leq k$) e $\alpha_j = v_{j-1}v_j \in A_G$ ($1 \leq j \leq k$). Geralmente, indicaremos P apenas pela seqüência de seus vértices, isto é, $P = [v_0, v_1, \dots, v_k]$. Denotaremos por P^{-1} o passeio $[v_k, v_{k-1}, \dots, v_1, v_0]$. Indicaremos por V_P e A_P o conjunto dos vértices e o conjunto das arestas (dos arcos) de P , respectivamente. Quando conveniente, também consideraremos P como sendo o subgrafo (subdigrafo) $G[A_P]$ de G . Dizemos que P é um **passeio de** v_0 **a** v_k ; que v_0 é o **início do passeio** e v_k é o **fim do passeio**. Além disso, os vértices v_0 e v_k são os **extremos do passeio** P . Os vértices $v_1, \dots, v_{k-1}$ são os **vértices internos** de P . Dizemos que o passeio P **passa pelo vértice** v_i , $0 \leq i \leq k$, e **pela aresta** (pelo arco) α_j , $1 \leq j \leq k$. O **comprimento** de um passeio P , denotado por $|P|$, é o

número de arestas (arcos) em P , contando repetições. Um passeio é dito **par** (**ímpar**) se $|P|$ é par (ímpar). Sempre que nos referirmos à paridade de um passeio estaremos nos referindo à paridade de seu comprimento. Um **passeio mínimo** de u a v é um passeio de menor comprimento possível de u a v .

Uma **trilha** é um passeio P tal que se $\alpha, \beta \in A_P$, então $\alpha \neq \beta$. Um **caminho** é um passeio cujos vértices são dois a dois distintos. Um passeio é dito **fechado** se ele tem o início e o fim iguais. Um **circuito** é uma trilha fechada, de comprimento não nulo, cujo início e vértices internos são todos distintos.

Em um digrafo D , um **semi-passeio** é uma seqüência da forma $P = [v_0, \alpha_1, v_1, \alpha_2, \dots, v_{k-1}, \alpha_k, v_k]$, onde $v_i \in V_D$ ($0 \leq i \leq k$) e α_j ($1 \leq j \leq k$) é um arco de D , cujos extremos são v_j e v_{j-1} . Observe que podemos ter $\alpha_j = v_j v_{j-1}$ ou $\alpha_j = v_{j-1} v_j$. Dizemos que P passa por α_j no **sentido correto** se $\alpha_j = v_{j-1} v_j$. Analogamente ao caso de passeio, define-se **semi-caminho** e **semi-circuito**. Cabe aqui ressaltar que todo passeio, caminho ou circuito é um semi-passeio, um semi-caminho ou um semi-circuito, respectivamente.

Sejam V_1 e V_2 dois subconjuntos de V_G . Um passeio (semi-passeio) de V_1 a V_2 é um passeio (semi-passeio) que tem o seu início em V_1 e o seu término em V_2 e nenhum dos seus vértices internos estão em $V_1 \cup V_2$.

Uma **seção** de um passeio $P = [v_0, \alpha_1, v_1, \alpha_2, \dots, v_{k-1}, \alpha_k, v_k]$ é uma subsequência $P' = [v_i, \alpha_{i+1}, v_{i+1}, \dots, \alpha_j, v_j]$, com $0 \leq i \leq j \leq k$. Dizemos também que P' é um **subpasseio** do passeio P , de v_i a v_k . Se $P_1 = [u_0, \alpha_1, u_1, \alpha_2, \dots, u_{k-1}, \alpha_k, u_k]$ e $P_2 = [v_0, \beta_1, v_1, \beta_2, \dots, v_{l-1}, \beta_l, v_l]$ são dois passeios em um grafo (digrafo) G tais que $u_k = v_0$, então chamamos **concatenação** de P_1 e P_2 o passeio $Q = [u_0, \alpha_1, u_1, \alpha_2, \dots, u_{k-1}, \alpha_k, u_k = v_0, \beta_1, v_1, \dots, \beta_l, v_l]$ e denotamos por $P_1 + P_2$. Tais conceitos são definidos analogamente para semi-passeios.

Dois passeios P_1 e P_2 são **disjuntos nos vértices** ou, simplesmente **disjuntos**, se os vértices internos de P_1 e P_2 são dois a dois distintos. Ou seja, P_1 e P_2 podem eventualmente ter extremos em comum. Dizemos que P_1 e P_2 são **totalmente disjuntos** se os vértices de P_1 e P_2 são dois a dois distintos. Os passeios P_1 e P_2 são **disjuntos nas arestas** (**nos arcos**) se $A_{P_1} \cap A_{P_2} = \emptyset$. Uma **coleção de passeios** disjuntos nos vértices, ou nas arestas (nos arcos), é uma coleção de passeios dois a dois disjuntos nos vértices, ou nas arestas (nos arcos).

Um digrafo sem circuitos é chamado **acíclico**. Cabe aqui observar que um tal digrafo pode ter semi-circuitos.

Dois vértices u e v em um grafo (digrafo) estão **ligados** se existe um passeio (semi-passeio) de u a v . A relação de ligação é de equivalência sobre os vértices do grafo (digrafo) e os subgrafos (subdigrafos) gerados pelas classes de equivalência são chamados

componentes do grafo (digrafo). Um grafo (digrafo) é **conexo** se possui um único componente; é **desconexo** se possui mais de um componente.

Dizemos que um vértice v em um grafo (digrafo) G é um **vértice de corte**, ou **ponto de articulação**, se existe uma partição de A_G em dois conjuntos A_1 e A_2 tal que v é o único vértice comum a $G[A_1]$ e $G[A_2]$. Um grafo G é dito **não-separável** se não tem vértices de corte. Um subgrafo não-separável maximal de G é chamado de **componente biconexo**, ou simplesmente **bloco**. Uma aresta (arco) α em um grafo (digrafo) G é uma **aresta (arco) de corte** ou **ponte** se o número de componentes de $G - \alpha$ é maior que o número de componentes de G .

Um digrafo D é **fortemente conexo** se para quaisquer dois vértices u e v em D , existe um caminho de u a v e outro caminho de v a u . Um **componente fortemente conexo** de D é um subdigrafo fortemente conexo maximal de D .

Dizemos que um grafo (digrafo) G é **k -conexo**, se para qualquer $S \subset V_G$, $|S| < k$, $G - S$ é conexo (fortemente conexo). Um grafo (digrafo) 2-conexo também é chamado **biconexo**.

Uma **floresta** é um grafo (digrafo) sem circuitos (semi-circuitos). Uma **árvore** é uma floresta conexa. Dada uma árvore T , podemos fixar um vértice r como sendo a **raiz** de T . Neste caso, dizemos que T é uma **árvore com raiz r** . Se v é um vértice de uma árvore T com raiz r , então os vértices de T que estão no caminho (semi-caminho) de r a v , exceto v , são chamados **ancestrais** de v . Os vértices que têm grau um em uma floresta são chamados **folhas**.

Um grafo (digrafo) G é **bipartido** se V_G pode ser particionado em dois conjuntos V_1 e V_2 tal que toda aresta de G tem um extremo em V_1 e outro em V_2 . Denotaremos tal grafo por $G(V_1, V_2)$.

Um **emparelhamento** em um grafo G é um subconjunto E de arestas de G , distintas de laços, tal que em cada vértice de G incide no máximo uma aresta de E . Seja E um emparelhamento em G . Um vértice v em G é **livre** em relação a E se v não é extremo de nenhuma aresta de E . Dizemos que E é **perfeito** se em G não existem vértices livres em relação a E . Um **caminho alternante** em relação a E é um caminho em G cujas arestas estão alternadamente em E e em $A_G - E$. Um tal caminho é chamado de **caminho de aumento** se tem ambos os extremos livres em relação a E . Sempre que o emparelhamento E estiver subentendido omitiremos a especificação “em relação a E ”.

Um grafo (digrafo) G é **ponderado** se existe um função peso p que associa a cada aresta (arco) de G um número inteiro, ou seja, $p: A_G \rightarrow \mathbb{Z}$. Dizemos que um grafo (digrafo) é $\{0,1\}$ -ponderado se os pesos das arestas (dos arcos) são 0 ou 1.

Um grafo (digrafo) G é dito **planar** se G pode ser desenhado no plano (representando vértices por pontos e arestas (arcos) por curvas ligando seus extremos) de modo que suas arestas (arcos) interceptem somente em seus extremos. Quando um tal desenho é dado, dizemos que o grafo (digrafo) G está **imerso** no plano. Uma **curva de Jordan** é uma curva fechada contínua que não se intercepta. Dado um grafo (digrafo) G imerso no plano, dois pontos do plano, digamos x e y , são ditos **conectados** se não estão em G e podem ser ligados por uma curva de Jordan, cujos pontos não estão em G . Para um tal grafo (digrafo) G , a relação de conexão é de equivalência sobre os pontos no plano não pertencentes a G e as suas classes de equivalência são as **faces** de G . Cabe aqui observar que sempre uma das faces é ilimitada. Essa face é chamada **face externa**. O passeio fechado que “contorna” uma face F em G é chamado **fronteira** ou **contorno** de F .

1.3 Definição dos problemas

São vários os problemas na Teoria dos Grafos que envolvem caminhos e circuitos. Estes têm sido largamente investigados, sendo vasta a literatura a respeito. Nesta seção vamos definir alguns desses problemas que serão tratados nesta dissertação. Veremos basicamente três tipos de problemas, todos eles relativos a circuitos (ou caminhos) com restrição de paridade. Em cada um deles, o dado de entrada consiste de um grafo (ou digrafo) G . Nos problemas sobre caminhos, estes são relativos a caminhos de um dado vértice a outro.

1. **Existência:** decidir se em G existe um circuito (caminho) par/ímpar;
2. **Busca:** encontrar em G , se existir, um circuito (caminho) par/ímpar.
3. **Busca de um mínimo:** encontrar em G , se existir, um circuito (caminho) par/ímpar, mínimo.

Nas Subseções 1.3.1, 1.3.2 e 1.3.3 definimos formalmente os problemas de *existência*, *busca* e *busca de um mínimo*, respectivamente. Cada um desses problemas serão referenciados por nomes mnemônicos (algumas vezes os argumentos serão omitidos), como indicados nas definições a seguir.

1.3.1 Problemas de existência

Problema do caminho par: $\text{PCamPar}(G, s, t)$

Dados: Um grafo (digrafo) G e dois vértices distintos s e t .

Problema: Decidir se existe em G um caminho par de s a t .

Problema do caminho ímpar: $\text{PCamImpar}(G, s, t)$

Dados: Um grafo (digrafo) G e dois vértices distintos s e t .

Problema: Decidir se existe em G um caminho ímpar de s a t .

Problema do circuito par: $\text{PCirPar}(G)$

Dado: Um grafo (digrafo) G .

Problema: Decidir se existe um circuito par em G .

Problema do circuito ímpar: $\text{PCirImpar}(G)$

Dado: Um grafo (digrafo) G .

Problema: Decidir se existe um circuito ímpar em G .

1.3.2 Problemas de busca

Problema de encontrar um caminho par: **PEncCamPar** (G, s, t)

Dados: Um grafo (digrafo) G e dois vértices distintos s e t .

Problema: Encontrar em G , se existir, um caminho par de s a t .

Problema de encontrar um caminho ímpar: **PEncCamÍmpar** (G, s, t)

Dados: Um grafo (digrafo) G e dois vértices distintos s e t .

Problema: Encontrar em G , se existir, um caminho ímpar de s a t .

Problema de encontrar um circuito par: **PEncCirPar** (G)

Dado: Um grafo (digrafo) G .

Problema: Encontrar, se existir, um circuito par em G .

Problema de encontrar um circuito ímpar: **PEncCirÍmpar** (G)

Dado: Um grafo (digrafo) G .

Problema: Encontrar, se existir, um circuito ímpar em G .

1.3.3 Problemas de busca de um mínimo

Problema do caminho par mínimo: PCamParMin (G, s, t)

Dados: Um grafo (digrafo) G e dois vértices distintos s e t .

Problema: Encontrar em G , se existir, um caminho par mínimo de s a t .

Problema do caminho ímpar mínimo: PCamImparMin (G, s, t)

Dados: Um grafo (digrafo) G e dois vértices distintos s e t .

Problema: Encontrar em G , se existir, um caminho ímpar mínimo de s a t .

Problema do circuito par mínimo: PCirParMin (G)

Dado: Um grafo (digrafo) G .

Problema: Encontrar, se existir, um circuito par mínimo em G .

Problema do circuito ímpar mínimo: PCirImparMin (G)

Dado: Um grafo (digrafo) G .

Problema: Encontrar, se existir, um circuito ímpar mínimo em G .

Capítulo 2

Circuitos ímpares

2.1 Introdução

O objetivo principal deste capítulo é estudar problemas relativos a existência, busca e busca de um mínimo para o caso em que o objeto em questão é um circuito ímpar. Mais precisamente, estudaremos os problemas $\text{PCirImpar}(G)$, $\text{PEncCirImpar}(G)$ e $\text{PCirImparMin}(G)$, onde a entrada G é um grafo ou um digrafo. Vamos assumir, neste capítulo, que o grafo (digrafo) é simples e conexo. Lembramos também que estamos assumindo $|V_G| = n$ e $|A_G| = m$.

Veremos que esses problemas, tanto para grafos como para digrafos, podem ser resolvidos por algoritmos polinomiais. Apresentaremos algoritmos lineares para resolver PCirImpar e PEncCirImpar e um algoritmo de complexidade $O(nm)$ para resolver PCirImparMin .

Começaremos estudando o PCirImpar ; a seguir veremos o PEncCirImpar ; e depois o PCirImparMin . Na última seção condensamos os resultados apresentados nas seções anteriores.

2.2 Existência de um circuito ímpar

O leitor que tenha alguma familiaridade com conceitos básicos em grafos, certamente não teria nenhuma dificuldade em conceber um algoritmo linear para resolver $\text{PCirImpar}(G)$ no caso em que G é um grafo. Já no caso em que G é um digrafo, ainda que não seja imediato, um pouco de reflexão conduziria o leitor a um algoritmo linear. Tais algoritmos são apresentados a seguir.

2.2.1 O PCirÍmpar para grafos

O resultado que mencionaremos a seguir, bastante conhecido na Teoria dos Grafos, nos conduz imediatamente a um algoritmo para resolver PCirÍmpar (veja, por exemplo, BONDY E MURTY [1976]).

Teorema 2.2.1 *Um grafo contém um circuito ímpar se, e somente se, não é bipartido.*

Dessa forma, podemos resolver PCirÍmpar(G) simplesmente verificando se G é bipartido. Para fazer isso, podemos usar o seguinte *algoritmo de busca em largura*, tradicionalmente mencionado em textos sobre algoritmos em grafos (BONDY E MURTY [1976], MCHUGH [1990], AHO, HOPCROFT E ULLMAN [1974]).

Algoritmo Verifica_Circuito_Ímpar_Grafo(G)

Entrada: Um grafo conexo $G = (V_G, A_G)$.

Saída: “Sim”, se G contém um circuito ímpar;
“Não”, caso contrário.

começo

para todo $v \in V_G$ **faça**

 nível(v) := 0;

 seja v um vértice qualquer de G ;

 Fila := v ;

enquanto Fila $\neq \emptyset$ **faça**

começo

 seja $x \in$ Fila;

 Fila := Fila $- x$;

para todo $\{x, y\} \in A_G$ **faça**

 se nível(y) = 0 **então**

começo

 Fila := Fila $+ y$;

 nível(y) := nível(x) + 1;

fim

senão (* nível(y) $\neq 0$ *)

 se nível(x) = nível(y) **então**

 responda “Sim” e pare;

fim

 responda “Não” e pare;

fim.

◇

Não é difícil verificar que se o grafo de entrada G não for bipartido, então esta busca em largura encontrará dois vértices distintos, digamos x e y , tais que $\text{nível}(x) = \text{nível}(y) \neq 0$. É fácil ver que no caso em que G é bipartido, o algoritmo anterior constrói uma bipartição de G . Podemos então enunciar o seguinte resultado.

Teorema 2.2.2 *O algoritmo Verifica_Circuito_Ímpar_Grafo(G) resolve PCirÍmpar(G) em tempo linear.*

2.2.2 O PCirÍmpar para digrafos

Para decidir a existência de circuitos ímpares em um digrafo D é suficiente restringir o teste aos componentes fortemente conexos de D . Considerando que é possível detectar os componentes fortemente conexos em tempo linear (veja, por exemplo, MCHUGH [1990]), basta então analisarmos o caso em que o digrafo é fortemente conexo. Afirmamos que em tal caso é suficiente verificar se o grafo é bipartido ou não. Para demonstrarmos essa afirmação, vamos estabelecer, primeiramente, o seguinte fato (que também vale para grafos):

Lema 2.2.3 *Todo passeio fechado ímpar contém um circuito ímpar.*

Prova: Seja $P = [x_1, x_2, \dots, x_k, x_1]$ um passeio fechado ímpar. A demonstração será feita por indução em k , ou seja, no comprimento de P . Se $k \leq 3$, então o resultado é imediato. Suponha, por hipótese de indução, que todo passeio fechado ímpar, de comprimento menor que k ($k > 3$), contenha um circuito ímpar.

Se os vértices x_i ($1 \leq i \leq k$) do passeio P são dois a dois distintos, então P é um circuito ímpar. Suponha então que P tenha dois vértices, digamos x_i e x_j ($1 \leq i < j \leq k$), tais que $x_i = x_j$. Dessa forma, P pode ser dividido em duas seções: $P_1 = [x_i, x_{i+1}, \dots, x_j]$ e $P_2 = [x_1, x_2, \dots, x_i, x_{j+1}, \dots, x_k, x_1]$. Se P_1 é ímpar, então por hipótese de indução, P_1 contém um circuito ímpar. Suponha então que P_1 seja par. Então P_2 é ímpar, e portanto, por hipótese de indução, P_2 contém um circuito ímpar.

□

Mostraremos a seguir que (em relação a existência de circuitos ímpares) para digrafos fortemente conexos, vale um resultado análogo ao que vimos para grafos. Mais precisamente, vale o seguinte resultado.

Teorema 2.2.4 *Um digrafo fortemente conexo contém um circuito ímpar se, e somente se, não é bipartido.*

Prova: Seja D um digrafo fortemente conexo. Se D contém um circuito ímpar, claramente D não é bipartido. Por outro lado, suponha que D não seja bipartido. Então D contém um semi-circuito ímpar, digamos $C = [x_1, \alpha_1, x_2, \dots, \alpha_{k-1}, x_k, \alpha_k, x_1]$. Seja A_{C+} o subconjunto dos arcos de C que estão orientados de x_i para x_{i+1} ($1 \leq i \leq k-1$). Se $\alpha_k = (x_k, x_1)$, então considere $A_{C+} := A_{C+} + \alpha_k$. Se $A_{C+} = \emptyset$ ou $A_{C+} = A_C$, então C é um circuito ímpar. Suponha, então, que não ocorra nenhum desses casos. Seja $\alpha_i = (x_i, x_{i+1})$ um arco de A_{C+} . Seja P um caminho, em D , de x_{i+1} a x_i (este caminho existe, pois, por hipótese, D é um digrafo fortemente conexo). Se P é par, então $P + (x_i, x_{i+1})$ é um circuito ímpar. Suponha, então, que P seja ímpar. Substitua em C o arco (x_i, x_{i+1}) pelo caminho P , obtendo dessa forma, um semi-passeio fechado ímpar. Podemos repetir esse processo para cada um dos demais arcos de A_{C+} , enquanto não for encontrado um circuito ímpar.

Assim, ou encontraremos um circuito ímpar ou produziremos um passeio fechado ímpar. Pelo Lema 2.2.3, segue que D contém um circuito ímpar. □

Assim, baseados no Teorema 2.2.4, podemos apresentar o seguinte algoritmo linear para decidir a existência de um circuito ímpar.

Algoritmo Verifica_Circuito_Ímpar_Digrafo(D)

Entrada: Um digrafo fortemente conexo D .

Saída: “Sim”, se D contém um circuito ímpar;
“Não”, caso contrário.

começo

seja G o grafo subjacente a D ;

responda Verifica_Circuito_Ímpar_Grafo(G);

fim.

◇

Aplicando-se o algoritmo Verifica_Circuito_Ímpar_Digrafo para cada componente fortemente conexo de D , obtém-se um algoritmo de complexidade $O(m)$ para decidir se D contém um circuito ímpar. Obtemos assim o seguinte resultado.

Teorema 2.2.5 *O $\text{PCirImpar}(D)$, onde D é um digrafo, pode ser resolvido em tempo linear.*

2.3 Busca de um circuito ímpar

Para resolver o PCirÍmpar não foi necessário encontrar um circuito ímpar: simplesmente verificamos se o grafo (ou cada componente fortemente conexo do digrafo) é bipartido ou não. Poderíamos, então, perguntar se encontrar um circuito ímpar (desde que o grafo, ou digrafo, contenha um) continua sendo um problema fácil (linear). A resposta a essa pergunta é afirmativa.

No caso de grafos, veremos que um algoritmo para resolver o PEncCirÍmpar é imediato. Já no caso de digrafos, um algoritmo linear não é tão óbvio. Apresentaremos um algoritmo linear que desenvolvemos. Cabe aqui mencionar que, para o caso de digrafos, não encontramos na literatura algoritmos com essa complexidade.

2.3.1 O PEncCirÍmpar para grafos

Seja G um grafo conexo não bipartido. Neste caso, como vimos na seção anterior, se aplicarmos o algoritmo *Verifica_Circuito_Ímpar*(G), encontraremos dois vértices distintos, digamos x e y , tais que $\text{nível}(x) \neq \text{nível}(y) \neq 0$.

Se $\{u, v\}$ é uma aresta do grafo G , para o qual o algoritmo atualizou $\text{nível}(v) = 0$ para $\text{nível}(v) := \text{nível}(u) + 1$, então dizemos que u é o **pai** do vértice v e escrevemos $\text{pai}(v) = u$. Sejam u e v dois vértices de G tais que $\text{nível}(u) < \text{nível}(v)$. Suponha que exista um caminho $P = [z_1, z_2, \dots, z_k]$, de u a v , tal que $\text{pai}(z_i) = z_{i-1}$, para $i = 2, \dots, k$. Então, dizemos que u é um **ancestral** de v .

Desta forma, para encontrar um circuito ímpar, basta procurar o ancestral comum a x e a y , que tem o maior nível possível. Podemos, assim, obter o algoritmo *Encontra_Circuito_Ímpar_Grafo*, fazendo algumas pequenas modificações no algoritmo *Verifica_Circuito_Ímpar_Grafo* da seção anterior.

Algoritmo Encontra_Circuito_Ímpar_Grafo(G)

Entrada: um grafo conexo G .

Saída: Um circuito ímpar C , se G contém um tal circuito.

começo

```

 $V_C := \emptyset; A_C := \emptyset;$ 
para todo  $v \in V_G$  faça
     $\text{nível}(v) := 0;$ 
seja  $v$  um vértice qualquer de  $G$ ;
 $\text{Fila} := v;$ 
enquanto  $\text{Fila} \neq \emptyset$  faça
```



```

começo
  seja  $x \in \text{Fila}$ ;
   $\text{Fila} := \text{Fila} - x$ ;
  para todo  $\{x, y\} \in A_G$  faça
    se  $\text{nível}(y) = 0$  então
      começo
         $\text{Fila} := \text{Fila} + y$ ;
         $\text{nível}(y) := \text{nível}(x) + 1$ ;
         $\text{pai}(y) := x$ ;
      fim
    senão
      se  $\text{nível}(x) = \text{nível}(y)$  então
        começo
           $V_C := V_C \cup \{x, y\}$ ;  $A_C := \{\{x, y\}\}$ ;
          enquanto  $\text{pai}(x) \neq \text{pai}(y)$  faça
            começo
               $V_C := V_C \cup \{\text{pai}(x), \text{pai}(y)\}$ ;
               $A_C := A_C \cup \{\{x, \text{pai}(x)\}, \{y, \text{pai}(y)\}\}$ ;
               $x := \text{pai}(x)$ ;  $y := \text{pai}(y)$ ;
            fim
           $V_C := V_C + \text{pai}(x)$ ;
           $A_C := A_C \cup \{\{x, \text{pai}(x)\}, \{y, \text{pai}(x)\}\}$ ;
        fim
      fim
    fim
  fim.

```

◇

É fácil ver que o algoritmo `Encontra_Circuito_Ímpar_Grafo` é correto e tem complexidade linear. Dessa forma, podemos enunciar o seguinte resultado.

Teorema 2.3.1 *O algoritmo `Encontra_Circuito_Ímpar_Grafo`(G) resolve `PEncCirImpar`(G) em tempo linear.*

2.3.2 O PEncCirImpar para digrafos

Tal como no caso de `PCirImpar` para digrafos, aqui também podemos restringir o estudo ao caso de digrafos fortemente conexos. Apresentaremos inicialmente dois algoritmos de complexidade $O(nm)$ para o `PEncCirImpar`, o primeiro deles baseado nas idéias discutidas na seção 2.2.2. Posteriormente, descreveremos dois algoritmos lineares para resolver `PEncCirImpar`.

2.3.2.1 Algoritmos ingênuos para o problema

Seja D um digrafo fortemente conexo não bipartido e G o grafo subjacente a D . Se aplicarmos o algoritmo `Encontra_Circuito_Ímpar_Grafo( $G$ )`, encontraremos, em G , um circuito ímpar C . No digrafo D , a estrutura correspondente a C é um semi-circuito.

Se esse semi-circuito é um circuito, não há mais nada a fazer. Caso contrário, a partir desse semi-circuito podemos encontrar um circuito ímpar, ou um passeio fechado ímpar. A demonstração do Teorema 2.2.4 nos indica uma forma de fazer isso: dado um arco $(x_i, x_{i+1}) \in A_{C^+}$, podemos encontrar um caminho (mínimo) Q , de x_{i+1} a x_i , fazendo uma busca em largura, no digrafo D , começando pelo vértice x_{i+1} , até que o vértice x_i seja alcançado (x_i é alcançado, pois D é um digrafo fortemente conexo). Como o algoritmo para encontrar um caminho tem complexidade $O(m)$, um circuito ímpar ou passeio fechado ímpar pode ser encontrado em tempo $O(nm)$. Cabe aqui observar que o comprimento máximo desse passeio fechado ímpar é $O(nm)$.

Descreveremos a seguir um algoritmo que encontra um circuito ímpar, a partir de um passeio fechado ímpar, baseado na demonstração do Lema 2.2.3.

Algoritmo Passeio_para_Circuito(P)

Entrada: Um passeio fechado ímpar P .

Saída: Um circuito ímpar.

Passo 1: Suponha que $P = [v_1, v_2, \dots, v_k, v_1]$. Se os vértices v_i ($1 \leq i \leq k$) de P são dois a dois distintos, então devolva P e pare.

Passo 2: Caso contrário, sejam v_i e v_j dois vértices de P tais que $v_i = v_j$ ($1 \leq i < j \leq k$).

Passo 3: Sejam $P' = [v_i, v_{i+1}, \dots, v_j]$ e $P'' = [v_1, v_2, \dots, v_i, v_{j+1}, \dots, v_k, v_1]$ duas seções de P . Se P' for ímpar, então $P := P'$. Caso contrário, $P := P''$.

Passo 4: Vá para o Passo 1.

◇

O algoritmo `Passeio_para_Circuito` pode ser implementado de forma a ter complexidade $O(|P|)$. De fato, para verificar se os vértices v_i ($1 \leq i \leq k$) de P são dois a dois distintos, podemos utilizar a seguinte estrutura de dados: uma pilha S (com topo T) e um vetor B , ambos de comprimento n . Para inicializar essa estrutura, simplesmente fazemos a atribuição $T := 0$. A inserção de um elemento v nessa estrutura é feita através de três atribuições:

$$\begin{aligned} T &:= T + 1; \\ S[T] &:= v; \\ B[v] &:= T; \end{aligned}$$

É fácil ver que um elemento u está nessa estrutura se, e somente se, $B[u] \leq T$ e $S[B[u]] = u$. Observe que o segundo teste só deve ser feito quando a primeira condição estiver satisfeita.

Devemos percorrer o passeio P de v_1 até v_k . Se um vértice v_j não pertence à estrutura, então inserimos v_j . Caso contrário, essa busca encontra um vértice v_i tal que $v_i = v_j$ ($i < j$). Assim, P' (descrito no algoritmo `Passeio_para_Circuito`) é um circuito de comprimento $T - B(v_i) + 1$. Através dessas observações, não é difícil ver que podemos implementar o algoritmo `Passeio_para_Circuito` de forma que sua complexidade seja $O(|P|)$.

Portanto, a partir de um passeio fechado ímpar (de comprimento máximo $O(nm)$) encontrado em tempo $O(nm)$, encontramos um circuito ímpar em tempo $O(nm)$, utilizando o algoritmo `Passeio_para_Circuito`.

KLEE, LADNER E MANBER [1984] apresentaram um outro algoritmo, cuja complexidade é $O(nm)$, para encontrar um passeio fechado ímpar. Para cada vértice v de D , faz-se uma busca em largura (com vértices repetidos) com raiz v . No k -ésimo estágio da busca, um vértice z recebe um sinal $+$ (ou $-$), se existe um passeio par (ou ímpar, respectivamente) de v ao vértice z . Observe que um vértice qualquer pode receber as duas marcas. Se em algum momento o vértice v receber a marca $-$, então um passeio fechado ímpar foi encontrado. Caso contrário, não existe um circuito ímpar passando por v . Esta busca em largura (com raiz v) pode ser feita em tempo $O(m)$, e, portanto, um passeio fechado ímpar (quando existente) pode ser encontrado em tempo $O(nm)$.

A partir destas considerações, a pergunta natural que surge é se existe algum algoritmo linear para encontrar um circuito ímpar em um digrafo. Respondemos essa pergunta apresentando, na próxima subseção, um tal algoritmo que desenvolvemos para o $\text{PEncCirImpar}(D)$.

2.3.2.2 Um algoritmo linear para o problema

O algoritmo que descreveremos nesta subseção resolve $\text{PEncCirImpar}(D)$ em tempo $O(m)$. Este algoritmo faz uso de um digrafo D' , construído a partir de D , da seguinte forma. A cada vértice $v \in V_D$, fazemos corresponder dois vértices v e v' em D' . Para cada arco $(u, v) \in A_D$, tomamos (u, v') e (u', v) como sendo arcos de D' . Mais formalmente, temos que

$$\begin{aligned} V_{D'} &= V_D \cup \{v' \mid v \in V_D\} \text{ e} \\ A_{D'} &= \{(u, v'), (u', v) \mid (u, v) \in A_D\}. \end{aligned}$$

Figura 2.1: Um digrafo D e o respectivo digrafo D' .

A Figura 2.1 mostra um digrafo D e o respectivo digrafo D' . Note, neste exemplo, que ao circuito par $[1, 5, 3, 2, 1]$ em D , correspondem (de maneira natural) dois circuitos em D' , de mesmo comprimento: $[1', 5, 3', 2, 1']$ e $[1, 5', 3, 2', 1]$. Observe, também, que ao circuito ímpar $C = [5, 3, 4, 5]$ em D , corresponde um único circuito em D' , $[5, 3', 4, 5', 3, 4', 5]$, cujo comprimento é o dobro de $|C|$.

Se $C = [x_1, x_2, \dots, x_k, x_1]$ é um circuito ímpar de D , então definimos o **circuito fraco** correspondente a C , em D' , como sendo o circuito $[x_1, x'_2, x_3, \dots, x_k, x'_1, x_2, \dots, x'_k, x_1]$. Além disso, dizemos que um caminho P , no digrafo D' , é um **caminho bom** se no máximo um dos vértices v ou v' ocorre em P , qualquer que seja v em D .

Notação: Para simplificar a notação dos vértices de D' , vamos representar genericamente um vértice de $V_{D'}$ com letras minúsculas com um til. Dessa forma, se $\tilde{v} = v'$ ($\tilde{v} = v$) então $\tilde{v}' = v$ ($\tilde{v}' = v'$).

Faremos, no digrafo D' , uma busca em profundidade um pouco diferente do conhecido na literatura, começando por um vértice qualquer (que chamaremos de **raiz**). Esta busca, como veremos adiante, constrói dois conjuntos de arcos, que denotaremos por **FORTE** e **FRACO**.

Algoritmo: Busca_Profundidade_Diferente(D)

Entrada: Um digrafo D fortemente conexo.

Saída: Os conjuntos de arcos *FORTE* e *FRACO*.

começo

construa o digrafo D' ; (* anteriormente definido *)

Pilha := $\emptyset$;

seja $\tilde{v} \in V_{D'}$; empilhe($\tilde{v}$);

marque o vértice $\tilde{v}$;

FORTE := $\emptyset$; *FRACO* := $\emptyset$;

enquanto Pilha $\neq \emptyset$ faça

começo

seja $\tilde{v}$ o vértice do topo da Pilha;

se existe um arco $(\tilde{v}, \tilde{y}') \in A_{D'}$ que não foi marcado então

começo

marque o arco $(\tilde{v}, \tilde{y}')$;

se $\tilde{y}' \notin \text{Pilha}$ então

se $\tilde{y}'$ não foi marcado então

se $\tilde{y} \in \text{Pilha}$ então

FRACO := *FRACO* + $(\tilde{v}, \tilde{y}')$;

senão (* $\tilde{y} \notin \text{Pilha}$ *)

começo

FORTE := *FORTE* + $(\tilde{v}, \tilde{y}')$;

empilhe($\tilde{y}'$);

marque o vértice $\tilde{y}'$;

fim

fim

senão (* todos os arcos saindo de $\tilde{v}$ já foram marcados *)

desempilhe o vértice do topo da Pilha;

fim

fim.

◇

O algoritmo *Busca_Profundidade_Diferente* constrói dois conjuntos de arcos. Dizemos que um arco é um **arco fraco** (respectivamente, **arco forte**), se ele pertence ao conjunto *FRACO* (respectivamente, *FORTE*). Podemos adiantar que o conjunto de arcos *FRACO* terá um papel fundamental para encontrar um circuito ímpar. É fácil ver que se um vértice $\tilde{v}$ foi marcado, então existe um caminho bom P , da raiz a $\tilde{v}$, tal que $A_P \subseteq \text{FORTE}$. Por outro lado, se um vértice $\tilde{u}$ não foi marcado e existe um caminho bom Q da raiz a $\tilde{u}$, então é fácil ver que algum arco de Q deve ser fraco.

Figura 2.2: Exemplo de uma execução de Busca_Profundidade_Diferente no digrafo D da figura 2.1.

Lema 2.3.2 *Seja D um digrafo fortemente conexo. Então para todo vértice $v \in V_D$, tem-se que ou $\tilde{v} \in V_{D'}$ ou $\tilde{v}' \in V_{D'}$ são marcados pelo algoritmo Busca_Profundidade_Diferente (independente da escolha da raiz).*

Prova: Suponha por absurdo que existam em D' dois vértices, $\tilde{v}$ e $\tilde{v}'$, que não foram marcados pelo algoritmo. Como D é fortemente conexo, existe um caminho P , em D , da raiz até v . Assim, existe um caminho bom P' , em D' , da raiz até $\tilde{v}$, ou até $\tilde{v}'$, dependendo da paridade de P . Vamos supor, sem perda de generalidade, que P' é um caminho bom da raiz até $\tilde{v}$.

Como o vértice $\tilde{v}$ não foi marcado, então pelo menos um arco de P' é fraco. Seja $(\tilde{x}', \tilde{y})$ o arco fraco tal que o subcaminho Q de P' , de $\tilde{y}$ até $\tilde{v}$, não contenha arcos fracos. Ora, se $(\tilde{x}', \tilde{y})$ é um arco fraco, então isto significa que $\tilde{y}'$ foi marcado pelo algoritmo. Seja Q' o caminho bom de $\tilde{y}'$ até $\tilde{v}'$, obtido a partir de Q , trocando os vértices $\tilde{z} \in V_Q$ por $\tilde{z}'$.

Como o vértice $\tilde{v}'$ não foi marcado e existe um caminho bom da raiz até $\tilde{v}'$ (já que existe um caminho bom da raiz até $\tilde{y}'$ e deste até $\tilde{v}'$) e $\tilde{y}'$ foi marcado, então pelo menos um arco de Q' é fraco. Seja $(\tilde{t}', \tilde{u})$ o último arco fraco desse caminho. Dessa forma, temos que $\tilde{u}' \in V_Q$ é um vértice que foi marcado. Como não existem arcos fracos de $\tilde{u}'$ a $\tilde{v}$ no caminho Q , temos que $\tilde{v}$ entrará na pilha e, portanto, será marcado. Isto é um absurdo, uma vez que assumimos que $\tilde{v}$ não foi marcado.

□

Lema 2.3.3 *Seja D um digrafo fortemente conexo e C' um circuito fraco de D' . Então o conjunto $FRACO$, construído pelo algoritmo `Busca_Profundidade_Diferente`, contém pelo menos um arco de C' .*

Prova: Seja $C' = [\hat{x}_1, \hat{x}'_2, \dots, \hat{x}_k, \hat{x}'_1, \hat{x}_2, \dots, \hat{x}'_k, \hat{x}_1]$ um circuito fraco de D' . Pelo Lema 2.3.2, $\hat{x}_i$ ou $\hat{x}'_i$, $1 \leq i \leq k$, são marcados. Suponha, sem perda de generalidade, que quando $\hat{x}_1$ entrou na pilha, nenhum dos vértices $\hat{x}_i$ e $\hat{x}'_i$ ($1 \leq i \leq k$) estavam na pilha, ou seja, $\hat{x}_1$ é o primeiro vértice do circuito fraco C' a entrar na pilha. Seja $P = [\hat{v}_1, \hat{v}_2, \dots, \hat{v}_k]$ o subcaminho de C' , de $\hat{x}_1$ até $\hat{x}_k$.

Suponha inicialmente que, depois de terminada a busca, exista algum vértice $\hat{v}_i$ não marcado, $2 \leq i \leq k$. Seja j o menor inteiro, $2 \leq j \leq k$, tal que $\hat{v}_j$ não foi marcado. Então, quando o arco $(\hat{v}_{j-1}, \hat{v}_j)$ foi considerado pela busca, o vértice $\hat{v}_j$ não entrou na pilha porque $\hat{v}'_j$ estava na pilha. Portanto, $(\hat{v}_{j-1}, \hat{v}_j)$ é um arco que pertence ao conjunto $FRACO$.

Finalmente, se todos os vértices $\hat{v}_i$, $2 \leq i \leq k$, foram marcados pela busca, então temos dois casos a considerar.

CASO 1: existe um vértice $\hat{v}_i$, $2 \leq i \leq k$, tal que quando ele entrou na pilha, $\hat{v}_1$ já havia saído da pilha.

Dentre todos esses vértices, seja $\hat{v}_j$ o vértice que tem o menor índice. Dessa forma, quando a busca considerou o arco $(\hat{v}_{j-1}, \hat{v}_j)$, o vértice $\hat{v}_1$ ainda estava na pilha. Assim, neste momento o vértice $\hat{v}_j$ não entrou na pilha porque $\hat{v}'_j$ ainda se encontrava na pilha. Portanto, $(\hat{v}_{j-1}, \hat{v}_j)$ é um arco que pertence ao conjunto $FRACO$.

CASO 2: para $i = 2, \dots, k$, quando $\hat{v}_i$ entrou na pilha, o vértice $\hat{v}_1$ ainda estava na pilha.

Neste caso, quando a busca considerou o arco $(\hat{x}_k, \hat{x}'_1)$, o vértice $\hat{x}_1$ ainda se encontrava na pilha. Portanto $(\hat{x}_k, \hat{x}'_1)$ é um arco que pertence ao conjunto $FRACO$.

□

Teorema 2.3.4 *Se D é um digrafo fortemente conexo, então D contém um circuito ímpar se, e somente se, $FRACO \neq \emptyset$, onde $FRACO$ é o conjunto construído pelo algoritmo `Busca_Profundidade_Diferente`.*

Prova: Suponha que $FRACO \neq \emptyset$, e considere um arco fraco $(\hat{x}, \hat{y}')$. Dessa forma, quando a busca considerou o arco $(\hat{x}, \hat{y}')$, o vértice $\hat{y}$ ainda se encontrava na pilha. Seja $\hat{v}_1, \hat{v}_2, \dots, \hat{v}_k$ a sequência de vértices, de $\hat{y}$ a $\hat{x}$, que estavam na pilha no momento em que o arco

$(\hat{x}, \hat{y}')$ foi considerado pela busca. Seja $P = [\hat{v}_1, \alpha_1, \hat{v}_2, \dots, \alpha_{k-1}, \hat{v}_k]$ o caminho bom de $\hat{y}$ a $\hat{x}$. É fácil ver que $A_P \subseteq \text{FORTE}$.

Seja P' o caminho bom, obtido a partir de P , substituindo os vértices $\hat{v}_i$ por $\hat{v}'_i$, $1 \leq i \leq k$, e os arcos $\alpha_i = (\hat{v}_i, \hat{v}_{i+1})$ por $\alpha'_i = (\hat{v}'_i, \hat{v}'_{i+1})$, $1 \leq i \leq k-1$. Então $C = P + (\hat{x}, \hat{y}') + P' + (\hat{x}', \hat{y})$ é um circuito fraco em D' que, em D , corresponde a um circuito ímpar.

Por outro lado, seja C um circuito ímpar de D . Seja C' o circuito fraco correspondente a C . Como D é fortemente conexo, concluímos, pelo Lema 2.3.3, que algum arco de C' será fraco.

□

Os resultados que apresentamos, juntamente com a prova do Teorema 2.3.4, nos conduzem ao algoritmo *Encontra_Circuito_Ímpar_Digrafo* para encontrar um circuito ímpar, em um digrafo fortemente conexo D . Este algoritmo baseia-se no algoritmo *Busca_Profundidade_Diferente*, só que não constrói os conjuntos *FORTE* e *FRACO*. Assim que um arco fraco é detectado, podemos construir um circuito ímpar, como sugere a prova do Teorema 2.3.4.

Algoritmo Encontra_Circuito_Ímpar_Digrafo(D)

Entrada: Um digrafo D fortemente conexo.

Saída: Um circuito ímpar C , se D não for bipartido.

começo

construa o digrafo D' ;

Pilha := $\emptyset$;

seja $\hat{v} \in V_{D'}$; empilhe($\hat{v}$);

marque o vértice $\hat{v}$;

enquanto Pilha $\neq \emptyset$ **faça**

começo

seja $\hat{v}$ o vértice do topo da Pilha;

se existe um arco $(\hat{v}, \hat{y}') \in A_{D'}$ que não foi marcado **então**

começo

marque o arco $(\hat{v}, \hat{y}')$;

se $\hat{y}' \notin \text{Pilha}$ **então**

se $\hat{y}'$ não foi marcado **então**

se $\hat{y} \in \text{Pilha}$ **então**


```

começo
  (* foi encontrado um arco fraco  $(\tilde{x}, \tilde{y}')$  *)

  seja  $\hat{v}_1, \hat{v}_2, \dots, \hat{v}_k$  a seqüência de vértices, de  $\hat{y}$ 
  a  $\tilde{x}$ , que estão na pilha;

   $C = [v_1, \alpha_1, v_2, \dots, v_k, v_1]$  é um circuito ímpar
  em  $D$ ;

  retorne  $C$  e pare;
fim
senão (*  $\hat{y} \notin \text{Pilha}$  *)
  começo
    empilhe( $\hat{y}'$ );
    marque o vértice  $\hat{y}'$ ;
  fim
fim
senão (* todos os arcos saindo de  $\hat{v}$  já foram marcados *)
  desempilhe o vértice do topo da Pilha;
fim
fim.

```

◇

Pelo Teorema 2.3.4, e pelo fato de que o algoritmo `Encontra_Circuito_Ímpar_Digrafo( $D$ )` é uma busca em profundidade, obtemos o seguinte teorema:

Teorema 2.3.5 *O algoritmo `Encontra_Circuito_Ímpar_Digrafo` é correto e tem complexidade $O(m)$.*

Observe que podemos construir o grafo G' , a partir de um grafo G , da mesma maneira que construímos o digrafo D' , a partir de um digrafo D . É fácil ver que os Lemas 2.3.2 e 2.3.3, e os Teoremas 2.2.4 e 2.3.4, também valem para o caso de grafos, substituindo a expressão “digrafo fortemente conexo” por “grafo conexo”. Portanto, o algoritmo `Encontra_Circuito_Ímpar_Digrafo`, com pequenas modificações, *também pode ser utilizado para encontrar um circuito ímpar em grafos*. Claramente, nesse caso, o fato de trabalharmos com um grafo com o dobro de vértices e arestas (em relação ao grafo original) não torna esse algoritmo muito interessante (em relação ao algoritmo `Encontra_Circuito_Ímpar_Grafo` visto na Subseção 2.3.1).

SOARES [1994], obteve um outro algoritmo linear que busca um circuito ímpar em um digrafo fortemente conexo. A descrição desse algoritmo é dada a seguir.

Algoritmo Circuito_Ímpar(D)

Entrada: Um digrafo fortemente conexo D .

Saída: Um circuito ímpar, se D não for bipartido;
“Não” caso contrário.

Passo 1: Seja v um vértice de D .

Passo 2: Faça uma busca em largura em D começando pelo vértice v , marcando cada vértice u de D com sinal $+$ ($-$) se um menor caminho de v a u tem paridade par (ímpar).

Passo 3: Seja H o digrafo obtido a partir de D trocando a orientação de todos os arcos de D ;

Passo 4: Faça uma busca em largura em H começando pelo vértice v , marcando cada vértice u de H com sinal $+$ ($-$) se um menor caminho de v a u tem paridade par (ímpar).

Passo 5: Se existe um vértice w que tem marca $+$ ($-$) em D e marca $-$ ($+$) em H , então encontramos um passeio fechado ímpar P em D passando por v e w . Retorne *Passeio_para_Circuito*(P) e pare. Caso contrário, vá para o Passo 6;

Passo 6: Se existe um arco (x, y) em D tal que x e y foram marcados com mesmo sinal em D , então encontramos um passeio fechado ímpar P que passa por v e pelo arco (x, y) . Retorne *Passeio_para_Circuito*(P) e pare. Caso contrário, vá para o Passo 7;

Passo 7: Neste caso, o digrafo D é um digrafo bipartido. Responda “Não”.

◇

Claramente o algoritmo *Circuito_Ímpar* é correto e todos os seus passos podem ser feitos em tempo $O(m)$.

Fizemos uma implementação dos algoritmos *Encontra_Circuito_Ímpar_Digrafo* e *Circuito_Ímpar* e verificamos que o primeiro algoritmo teve um desempenho melhor.

2.3.2.3 Uma implementação do algoritmo Encontra_Circuito_Ímpar_Digrafo

Apresentamos aqui uma descrição sucinta das estruturas de dados utilizadas para fazer uma implementação do algoritmo Encontra_Circuito_Ímpar_Digrafo. Uma implementação desse algoritmo, em linguagem C, encontra-se no Apêndice A.

Utilizamos uma lista de adjacência para armazenar um digrafo. Além disso, fizemos uso de dois vetores booleanos: um para verificar se um determinado vértice foi marcado, e outro para verificar se um dado vértice está na pilha. Foi utilizado, ainda, um vetor de apontadores, cujo nome é “arco_nao_marcado”, para verificar se existe um arco $(\hat{x}, \hat{y}')$ que não foi marcado. Assim, se “arco_nao_marcado $[\hat{x}] = \text{NULL}$ ”, então todos os arcos que saem de $\hat{x}$ já foram marcados. Caso contrário, “arco_nao_marcado $[\hat{x}]$ ” indica um vértice, digamos $\hat{y}'$, tal que $(\hat{x}, \hat{y}')$ ainda não foi marcado.

2.4 Busca de um circuito ímpar mínimo

Vimos, nas seções anteriores, que para resolver o problema da existência, ou o problema de encontrar um circuito ímpar, existem algoritmos lineares. Vamos apresentar nesta seção um algoritmo, de complexidade $O(nm)$, para encontrar um circuito ímpar mínimo que desenvolvemos através das considerações feitas na seção anterior. Recentemente, CAPRARA E FISCHETTI [1993] obtiveram um algoritmo de complexidade $O(n^3)$ para encontrar um circuito ímpar de peso mínimo em um grafo (digrafo) ponderado utilizando programação dinâmica.

Dizemos um circuito C em um grafo (digrafo) G , é um **circuito induzido** se o seu comprimento é maior que três e o subgrafo (subdigrafo) induzido por V_C ($G[V_C]$) é o próprio circuito C . Observe que o problema de procurar um circuito ímpar mínimo maior que três em um grafo (digrafo) é polinomialmente equivalente ao problema de procurar um circuito induzido ímpar mínimo. Cabe aqui mencionar que o problema de decidir se um grafo contém um circuito induzido ímpar é um problema em aberto; já no caso de digrafos, esse problema é $\mathcal{NP}$ -completo (veja LUBIW [1988/89]).

O teorema a seguir, válido para grafos ou digrafos, será fundamental para encontrar circuitos ímpares mínimos.

Teorema 2.4.1 *Um passeio fechado ímpar mínimo é um circuito ímpar mínimo.*

Prova: Como todo circuito (ímpar) é um passeio fechado (ímpar), é suficiente provarmos que um passeio fechado ímpar mínimo é um circuito (ímpar). Seja $P = [v_1, v_2, \dots, v_k, v_1]$ um passeio fechado ímpar mínimo. Suponha por absurdo que P não seja um circuito. Dessa forma, existem dois vértices v_i e v_j ($i < j$) tais que $v_i = v_j$. Seja $P' = [v_i, v_{i+1}, \dots, v_j]$ a seção de P de v_i até v_j e $P'' = [v_1, \dots, v_i, v_{j+1}, \dots, v_k, v_1]$ a outra seção de P . Note

que P' e P'' são passeios fechados que têm paridades diferentes, uma vez que P é ímpar. Mas isto é um absurdo, pois P por hipótese é um passeio fechado ímpar mínimo.

□

O teorema acima nos sugere que para encontrar um circuito ímpar mínimo basta encontrar um passeio fechado ímpar mínimo.

Para o caso de digrafos (e também para o caso de grafos), o algoritmo de KLEE, LADNER E MANBER [1984], apresentado na Seção 2.3, encontra, para cada busca com raiz em v , um passeio fechado ímpar mínimo que passa pelo vértice v , se tal vértice receber marca $-$. Caso v não receba marca $-$, então não existe um circuito ímpar passando por v . Se chamarmos de $\text{nível}(v)$ o nível em que o vértice v recebeu a marca $-$ (caso v não receba a marca $-$, considere $\text{nível}(v) = \infty$), então um circuito ímpar mínimo passa por um vértice, digamos u , cujo valor de $\text{nível}(u)$ é o menor entre todos os outros vértices do digrafo. Como vimos anteriormente, o algoritmo de KLEE, LADNER E MANBER [1984] tem complexidade $O(nm)$.

Tendo em vista os resultados da seção anterior, apresentamos a seguir, um algoritmo que desenvolvemos para encontrar um circuito ímpar mínimo. Este algoritmo pode ser aplicado tanto para grafos, quanto para digrafos e tem complexidade $O(nm)$.

O algoritmo *Encontra_Circuito_Ímpar_Digrafo* constrói, a partir de um digrafo de entrada D , o digrafo D' . Analogamente, podemos construir, a partir de um grafo G , o grafo G' , assim definido:

$$V_{G'} = V_G \cup \{v' \mid v \in V_G\} \text{ e} \\ A_{G'} = \{\{u, v'\}, \{u', v\} \mid \{u, v\} \in A_G\}.$$

O seguinte resultado segue imediatamente da construção do digrafo D' (grafo G').

Lema 2.4.2 *A cada caminho mínimo P no digrafo D' (grafo G') de v a v' corresponde um passeio fechado ímpar mínimo no digrafo D (grafo G) que passa pelo vértice v , de mesmo comprimento que P . Reciprocamente, a cada passeio fechado ímpar mínimo Q no digrafo D (grafo G), que passa por um vértice v , corresponde um caminho mínimo no digrafo D' (grafo G'), de v a v' , de mesmo comprimento que Q .*

Dessa forma, procurar um passeio fechado ímpar no digrafo (grafo) G , que passa por um vértice v , equivale a procurar um caminho, de v a v' , em G' .

Portanto, se encontrarmos, para cada vértice $v \in D'$ (G') um caminho mínimo de v a v' , no digrafo D' (no grafo G'), teremos encontrado um passeio fechado ímpar mínimo no digrafo D (no grafo G), que passa pelo vértice v . Dentre todos os passeios fechados ímpares

mínimos encontrados, seja C aquele que tem um menor comprimento. Pelo Teorema 2.4.1, C é um menor circuito ímpar do digrafo D (grafo G).

Para encontrar um caminho mínimo entre dois vértices, digamos s e t , podemos utilizar um algoritmo de busca em largura, que chamaremos de $\text{Caminho_Mínimo}(s, t)$. O algoritmo para encontrar um circuito ímpar mínimo é dado a seguir.

Algoritmo Circuito_Ímpar_Mínimo_Digrafo(D)
(Circuito_Ímpar_Mínimo_Grafo(G))

Entrada: Um digrafo fortemente conexo D (um grafo conexo G).

Saída: Um circuito ímpar mínimo, se D (G) contém um.

Passo 1: Construa o digrafo D' (grafo G');

Passo 2: Para cada vértice $v \in V_{D'}$ ($v \in V_{G'}$), aplique o algoritmo $\text{Caminho_Mínimo}(v, v')$. Seja P_v o caminho devolvido pelo algoritmo e $\text{Comp}(v) = |P_v|$. Caso não exista um caminho de v a v' , considere $\text{Comp}(v) = \infty$;

Passo 3: Se para todos os vértices $v \in V_{D'}$ ($v \in V_{G'}$), $\text{Comp}(v) = \infty$, então retorne $C = \emptyset$ e pare.

Passo 4: Caso contrário, seja $u \in V_{D'}$ ($u \in V_{G'}$) tal que $\text{Comp}(u)$ tenha o menor valor possível. Considere $P_u = [u_1, u'_2, \dots, u_k, u'_1]$. Então $C = [u_1, u_2, \dots, u_k, u_1]$ é um circuito ímpar em D (G). Devolva C e pare.

◇

Pelo Lema 2.4.2, e pelo fato de que cada aplicação do algoritmo Caminho_Mínimo tem complexidade $O(m)$, podemos enunciar o seguinte teorema:

Teorema 2.4.3 *O algoritmo Circuito_Ímpar_Mínimo_Grafo (Circuito_Ímpar_Mínimo_Digrafo) encontra um circuito ímpar mínimo em tempo $O(nm)$.*

Observamos que SEIDEL [1992] obteve um algoritmo para encontrar as distâncias entre todos os vértices de um grafo de complexidade é $O(M(n) \log n)$, onde $M(n)$ é o tempo necessário para multiplicar duas matrizes quadradas de ordem n (que pode ser feito em $o(n^{2.376})$). Dessa forma, para o caso de grafos, podemos encontrar um menor circuito ímpar utilizando o seguinte algoritmo.

Algoritmo Circuito_Ímpar_Mínimo_Grafo_Seidel(G)

Entrada: Um um grafo conexo G .

Saída: Um circuito ímpar mínimo, se G contém um.

Passo 1: Construa o grafo G' ;

Passo 2: Utilizando o algoritmo de SEIDEL [1992], encontre a distância entre todos os pares de vértices de G' . Caso não exista um caminho entre dois vértices, considere a distância entre eles igual a ∞ ;

Passo 3: Se para todos os vértices $w \in V_{G'}$, a distância entre w e w' é igual a ∞ , então retorne $C = \emptyset$ e pare. Caso contrário, seja v e v' um par de vértices que tenha a menor distância possível;

Passo 4: Aplique o algoritmo Caminho_Mínimo(v, v'). Seja P o caminho devolvido por Caminho_Mínimo;

Passo 5: Considere $P = [v_1, v'_2, \dots, v_k, v'_1]$. Então $C = [v_1, v_2, \dots, v_k, v_1]$ é um circuito ímpar mínimo em G . Devolva C e pare;

◇

2.5 Conclusão

Indicamos no quadro abaixo os algoritmos (com as melhores complexidades) discutidos neste capítulo

Problema	Algoritmo	Complexidade
PCirImpar	Verifica_Circuito_Ímpar_Grafo(G)	$O(m)$
	Verifica_Circuito_Ímpar_Digrafo(D)	$O(m)$
PEncCirImpar	Encontra_Circuito_Ímpar_Grafo(G)	$O(m)$
	Circuito_Ímpar(D)	$O(m)$
	Encontra_Circuito_Ímpar_Digrafo(D)	$O(m)$
PCirImparMin	KLEE, LADNER E MANBER [1984]	$O(nm)$
	Circuito_Ímpar_Mínimo_Grafo(G)	$O(nm)$
	Circuito_Ímpar_Mínimo_Grafo_Seidel(G)	$O(M(n) \log n)$
	Circuito_Ímpar_Mínimo_Digrafo(D)	$O(nm)$

Os algoritmos Verifica_Circuito_Ímpar_Grafo, Verifica_Circuito_Ímpar_Digrafo e Encontra_Circuito_Ímpar_Grafo são imediatos. Os algoritmos Encontra_Circuito_Ímpar_Digrafo, Circuito_Ímpar_Mínimo_Grafo e Circuito_Ímpar_Mínimo_Digrafo não foram encontrados na literatura.

O algoritmo de KLEE, LADNER E MANBER [1984] que encontra, em um dado digrafo, um circuito ímpar mínimo tem a mesma complexidade do algoritmo Circuito_Ímpar_Mínimo_Digrafo. Esses algoritmos são fáceis de implementar e basicamente procuram passeios

fechados ímpares mínimos. Talvez a desvantagem do segundo algoritmo é o fato do mesmo trabalhar com um digrafo que tem o dobro de vértices e arcos do digrafo de entrada.

Capítulo 3

Circuitos pares em grafos

3.1 Introdução

Estudaremos neste capítulo os seguintes problemas sobre circuitos pares em grafos: PCirPar, PEncCirPar e PCirParMin. Veremos que todos esses problemas podem ser resolvidos por algoritmos polinomiais. Apresentaremos algoritmos lineares para resolver PCirPar e PEncCirPar e um algoritmo de complexidade $O(m^2)$ para resolver PCirParMin.

3.2 Existência de circuitos pares

O algoritmo linear para resolver o PCirPar que veremos nesta seção é baseado no seguinte resultado bastante conhecido e citado em vários artigos (como por exemplo THOMASSEN [1985a] e ARKIN, PAPADIMITRIOU E YANNAKAKIS [1991]).

Lema 3.2.1 *Um grafo G contém um circuito par se, e somente se, existe em G um bloco que não é uma aresta e nem um circuito ímpar.*

Prova: Seja G um grafo. Se G contém um circuito par, então claramente tal circuito está em um bloco de G que não é uma aresta e nem um circuito ímpar. Para provar a recíproca, suponha que G tenha um bloco B que não seja um circuito ímpar, nem uma aresta. Seja C um menor circuito de B . Se B é bipartido, então C é um circuito par. Suponha, então, que B não seja bipartido e C seja um circuito ímpar. É fácil ver que B contém uma aresta $\{u, v\}$ tal que $u \in V_C$ e $v \notin V_C$. Como B é 2-conexo, então $B - u$ contém um caminho P de v a $V_C - u$. Sem perda de generalidade, suponha que $V_P \cap V_C = \{x\}$. Sejam C_{xu} e C_{ux} duas seções C de x a u e u a x , respectivamente. Então os circuitos $[u, v] + P + C_{xu}$ e $[u, v] + P + C_{ux}^{-1}$ têm paridades diferentes.

□

O fato acima nos conduz imediatamente ao seguinte algoritmo.

Algoritmo Verifica_Circuito_Par_Grafo(G)

Entrada: Um grafo G .

Saída: “Sim”, se G contém um circuito par;
“Não”, caso contrário.

Passo 1: Encontre todos os blocos de G ;

Passo 2: Se existe um bloco que não é uma aresta e nem um circuito ímpar, então responda “Sim”. Caso contrário, responda “Não”.

◇

É bem conhecido que o fato de que é possível encontrar todos os blocos de um grafo em tempo linear (veja AHO, HOPCROFT E ULLMAN [1974]). Dessa forma, temos o seguinte resultado.

Teorema 3.2.2 *O algoritmo Verifica_Circuito_Par_Grafo resolve PCirPar em tempo linear.*

3.3 Busca de um circuito par

O algoritmo linear visto na seção anterior (baseado no Lema 3.2.1) resolve o PCirPar sem fazer a busca explícita de um circuito par. O leitor atento talvez já tenha notado porém que um algoritmo linear para encontrar um tal circuito par (quando este existe) está implícito na própria demonstração do Lema 3.2.1. Essas idéias estão presentes no algoritmo descrito a seguir.

Algoritmo Encontra_Circuito_Par_Grafo(G)

Entrada: Um grafo G .

Saída: Um circuito par em G , se existente;
“Não”, caso contrário.

Passo 1: Encontre todos os blocos de G ;

Passo 2: Se todos esses blocos ou são uma aresta, ou um circuito ímpar, então responda “Não” e pare. Caso contrário, seja B um bloco de G que não é um circuito ímpar e nem uma aresta;

Passo 3: Seja (a, b) uma aresta de B . Encontre em B um menor caminho Q de b a a . Seja C o circuito $Q + [a, b]$. Se C é par, então retorne C e pare. Caso contrário vá para o Passo 4;

Passo 4: Seja $\{u, v\}$ uma aresta de B tal que $u \in V_C$ e $v \notin V_C$;

Passo 5: Encontre um caminho P , em $B - u$, de v a $V_C - u$. Seja x tal que $V_P \cap V_C = \{x\}$;

Passo 6: Sejam C_{xu} e C_{ux} as seções de C de x a u e de u a x , respectivamente;

Passo 7: Retorne o circuito $[u, v] + P + C_{xu}$, se ele é par. Caso contrário, retorne o circuito $[u, v] + P + C_{ux}^{-1}$.

◇

É fácil ver que cada passo do algoritmo *Encontra_Circuito_Par_Grafo* pode ser executado em tempo linear. Essa observação nos permite concluir o seguinte teorema.

Teorema 3.3.1 *O algoritmo Encontra_Circuito_Par_Grafo resolve PEncCirPar em tempo linear.*

3.4 Busca de um circuito par mínimo

Vimos, nas seções anteriores, que para resolver os problemas de existência e de busca existem algoritmos lineares. No entanto, para encontrar um circuito par mínimo, o único algoritmo que conhecemos tem complexidade $O(m^2)$. Esse algoritmo utiliza um outro algoritmo que encontra um caminho ímpar mínimo entre dois vértices distintos.

Algoritmo Circuito_Par_Mínimo_Grafo(G)

Entrada: Um grafo G .

Saída: Um circuito par mínimo, se G contém um;
“Não”, caso contrário.

Passo 1: Para cada aresta $\alpha = \{x, y\}$ de G , procure em $G - \alpha$ um menor caminho ímpar P_α de x a y . Caso não exista um tal caminho, considere $P_\alpha = \emptyset$;

Passo 2: Se todos esses caminhos $P_\alpha = \emptyset$, então responda “Não” e pare. Caso contrário, vá para o Passo 3.

Passo 3: Dentre os caminhos P_α , seja P um de menor comprimento (se $P_\alpha = \emptyset$, considere $|P_\alpha| = \infty$) e sejam x e y o início e o fim de P , respectivamente. Retorne o circuito $P + [y, x]$.

◇

Para encontrar um caminho ímpar mínimo, podemos usar o algoritmo Caminho_Par_Mínimo_Grafo_Via_Fase_Completa de complexidade $O(m)$, descrito no capítulo 5. Assim, obtemos o seguinte resultado.

Teorema 3.4.1 *O algoritmo Circuito_Par_Mínimo_Grafo resolve PCirParMin em $O(m^2)$.*

3.5 Conclusão

As complexidades dos algoritmos estudados neste capítulo estão condensados na tabela a seguir.

Problema	Algoritmo	Complexidade
PCirPar	Verifica_Circuito_Par_Grafo	$O(m)$
PEncCirPar	Encontra_Circuito_Par_Grafo	$O(m)$
PCirParMin	Circuito_Par_Mínimo_Grafo	$O(m^2)$

Os algoritmos Verifica_Circuito_Par_Grafo e Encontra_Circuito_Par_Grafo são obtidos a partir da demonstração do Lema 3.2.1, enquanto que o algoritmo Circuito_Par_Mínimo_Grafo é baseado na busca de caminhos ímpares mínimos. Seria interessante encontrar um outro algoritmo para resolver PCirParMin que não fosse baseado nessa mesma idéia.

Capítulo 4

Circuitos pares em digrafos

4.1 Introdução

Neste capítulo estudaremos o problema do circuito par (PCirPar) para digrafos. A complexidade computacional desse problema, que consiste em decidir se um digrafo contém um circuito par está em aberto. À primeira vista esse fato é um tanto surpreendente já que existem algoritmos lineares para verificar se um digrafo é acíclico, ou mesmo se ele contém um circuito ímpar.

Aparentemente, D.H. YOUNGER (cf. BERMOND E THOMASSEN [1981]) foi o primeiro a reconhecer que encontrar circuitos pares em digrafos não é tão simples. Ele, por volta de 1975, formulou a seguinte pergunta:

Existe um algoritmo polinomial para encontrar um
circuito par em um dado digrafo?

Desde então vários pesquisadores têm investigado esse problema, numa tentativa até o momento infrutífera no sentido de estabelecer qual a sua complexidade computacional. Ainda que a questão central não tenha sido estabelecida, os resultados que encontramos na literatura são interessantes e merecem ser estudados. Faremos aqui uma resenha (que pretendemos estar bem próxima de ser completa) a respeito desse problema.

4.2 Problemas equivalentes

Mencionaremos nesta seção alguns problemas coletados da literatura que são polinomialmente equivalentes ao PCirPar(D).

4.2.1 O problema do circuito de peso par

Seja D um digrafo $\{0,1\}$ -ponderado. Dizemos que o **peso de um circuito** C em D é a soma dos pesos de seus arcos. Veremos a seguir que o seguinte problema é equivalente ao PCirPar.

Problema do circuito de peso par: PCirPPar(D)

Dado: Um digrafo D $\{0,1\}$ -ponderado.

Problema: Decidir se D contém um circuito de peso par.

Teorema 4.2.1 *PCirPar é polinomialmente equivalente ao PCirPPar.*

Prova: Suponha que exista um algoritmo polinomial que resolve PCirPPar. Seja D um digrafo, sobre o qual queremos decidir a existência de um circuito par. Atribua peso 1 aos arcos de D . É fácil ver que PCirPar(D) tem uma resposta afirmativa se, e somente se, PCirPPar(D) tem uma resposta afirmativa.

Por outro lado, suponha que exista um algoritmo polinomial que resolve PCirPar. Seja D um digrafo $\{0,1\}$ -ponderado. Seja D' o digrafo obtido a partir de D subdividindo (bipartindo) uma ou mais vezes todos os arcos cujo peso é 0. É fácil ver que PCirPPar(D) tem uma resposta afirmativa se, e somente se, PCirPar(D') tem uma resposta afirmativa.

□

4.2.2 O problema do digrafo par

Dizemos que um digrafo D é **par** se para qualquer atribuição de pesos 0/1 aos seus arcos, D contém um circuito de peso par. Ou equivalentemente, se após a subdivisão de um número qualquer de arcos, D contém um circuito de comprimento par. Provaremos aqui que o seguinte problema é equivalente ao PCirPPar.

Problema do digrafo par: PDigrafoPar(D)

Dado: Um digrafo D .

Problema: Decidir se D é um digrafo par.

Antes de provarmos que esse problema é polinomialmente equivalente ao PCirPPar, vejamos alguns conceitos preliminares.

Podemos representar qualquer subconjunto, de um dado conjunto E , como um vetor de um espaço vetorial sobre o corpo $GF[2]$, onde a soma de dois vetores é o vetor correspondente à diferença simétrica dos dois subconjuntos representados por eles. Dessa forma,

define-se o **espaço dos ciclos de um grafo** G , como sendo o espaço vetorial sobre o corpo $GF[2]$, gerado pelos circuitos de G . O **espaço dos ciclos de um digrafo** refere-se ao espaço dos ciclos de seu grafo subjacente.

A dimensão do espaço dos ciclos de um grafo (ou digrafo) conexo é $m - n + 1$ (cf. BERGE [1973]). Um conjunto de circuitos de um digrafo D , que é uma base do espaço dos ciclos de D , é chamado de **base de D** .

Nesta subseção não faremos distinção entre circuitos e conjuntos de seus arcos. Dessa forma, se C_1 e C_2 são circuitos, $C_1 \oplus C_2$ denota tanto a diferença simétrica do conjunto de arcos de C_1 e C_2 ou o circuito cujos arcos é a tal diferença.

Teorema 4.2.2 *Todo digrafo fortemente conexo tem uma base. Além disso, se D é um digrafo fortemente conexo e E é um subconjunto dos seus arcos tal que $D - E$ é fortemente conexo, então podemos escolher uma base onde cada arco de E está exatamente em um circuito dessa base.*

Prova: Seja $H_0, H_1, \dots, H_k$ uma seqüência de digrafos tal que H_0 é um circuito e $H_k = D$; e, para cada $i = 1, 2, \dots, k$, H_i é obtido a partir de H_{i-1} adicionando um caminho (ou um circuito) P_i de algum vértice de H_{i-1} a outro (ou ao mesmo) vértice de H_{i-1} .

Uma vez que H_{i-1} é fortemente conexo, H_{i-1} contém um caminho P'_i tal que $P'_i + P_i = S_i$ é um circuito. É fácil ver (por indução em i) que $H_0, S_1, \dots, S_i$ é uma base de H_i . Dessa forma, $H_0, S_1, \dots, S_k$ é uma base de $H_k = D$.

Se $D - E$ é fortemente conexo, então escolha uma seqüência $H_0, H_1, \dots, H_l, \dots, H_k$ tal que $H_l = D - E$, e para $l < i \leq k$, P_i consiste de um arco (x_i, y_i) de E e P'_i de um caminho de y_i a x_i em $D - E$.

□

Note que a prova do Teorema 4.2.2, nos sugere como encontrar uma base de um digrafo fortemente conexo em tempo polinomial. No próximo teorema veremos que é possível atribuir pesos 0/1 aos arcos de um digrafo fortemente conexo, com uma base $\mathcal{C}$, de forma que cada circuito de $\mathcal{C}$ tenha peso ímpar. Para isso, precisamos de dois lemas, que serão apresentados a seguir.

Lema 4.2.3 *Seja $\mathcal{F}$ uma família não vazia de subconjuntos de um conjunto finito, que é fechado sob a diferença simétrica no seguinte sentido: a diferença simétrica de um número ímpar de membros de $\mathcal{F}$ também está em $\mathcal{F}$. Então todo conjunto minimal, que intercepta cada membro de $\mathcal{F}$, tem interseção ímpar com cada membro de $\mathcal{F}$.*

Prova: Suponha por absurdo que exista um conjunto minimal T e um membro E_0 de $\mathcal{F}$ tal que $E_0 \cap T = \{e_1, e_2, \dots, e_k\}$, com k par. Pela minimalidade de T , existe um

conjunto E_i em $\mathcal{F}$ tal que $E_i \cap T = \{e_i\}$, para $i = 1, 2, \dots, k$. Por hipótese, $E' = E_0 \oplus E_1 \oplus \dots \oplus E_k$ está em $\mathcal{F}$. No entanto, $E' \cap T = \emptyset$. Isso é uma contradição, uma vez que T tem interseção não vazia com cada membro de $\mathcal{F}$.

□

Lema 4.2.4 *Se $\mathcal{C}$ é uma família de subconjuntos de um conjunto finito, então exatamente uma das seguintes afirmações é verdadeira:*

- (1) *Existe um subconjunto que tem interseção ímpar com cada membro de $\mathcal{C}$.*
- (2) *Existem subconjuntos $E_1, E_2, \dots, E_k$ em $\mathcal{C}$, tal que k é ímpar e $E_1 \oplus E_2 \oplus \dots \oplus E_k = \emptyset$.*

Prova: Seja $\mathcal{F}$ a família de subconjuntos que podem ser obtidos a partir de $\mathcal{C}$, fazendo a diferença simétrica de um número ímpar de membros de $\mathcal{C}$. Se $\emptyset \in \mathcal{F}$, então vale (2). Caso contrário, $\mathcal{F}$ satisfaz a hipótese do Lema 4.2.3 e portanto qualquer conjunto minimal, que intercepta cada membro de $\mathcal{F}$, satisfaz (1). É fácil ver que (1) e (2) não podem valer ao mesmo tempo.

□

Teorema 4.2.5 *Existe uma atribuição de pesos 0/1 aos arcos de um digrafo fortemente conexo, de maneira que cada circuito de uma base tenha peso ímpar.*

Prova: Seja D um digrafo fortemente conexo e $\mathcal{C}$ uma base de D . Uma vez que $\mathcal{C}$ é uma base, o item (2) do Lema 4.2.4 não é satisfeito. Portanto existe um conjunto minimal T que satisfaz o item (1). Atribua peso 1 aos arcos que pertencem a T e peso 0 aos arcos que estão em $A_D - T$. Como T tem interseção ímpar com cada membro de $\mathcal{C}$, então cada circuito de $\mathcal{C}$ tem peso ímpar.

□

Uma pergunta natural que surge após uma observação atenta da prova do Teorema 4.2.5 é se é fácil encontrar um tal conjunto minimal que satisfaz o item (1) do Lema 4.2.4 (isto é, que tem interseção ímpar com cada membro de uma família $\mathcal{F}$). É interessante notar que para isto, basta simplesmente encontrar um conjunto minimal T que intercepta todos os circuitos de $\mathcal{C}$. De fato, suponha o contrário, ou seja, que exista um outro conjunto minimal $T' \subset T$ tal que T' intercepta todos os membros de $\mathcal{F}$. Uma vez que $\mathcal{C} \subseteq \mathcal{F}$, então T' intercepta todos os circuitos de $\mathcal{C}$. Mas isso é uma contradição, pois T é um conjunto minimal que intercepta todos os circuitos de $\mathcal{C}$. Uma vez que $\mathcal{C}$ é uma base,

é fácil verificar que T tem interseção ímpar com todos os membros de $\mathcal{F}$. Portanto, basta encontrarmos um conjunto minimal T que intercepta todos os circuitos de $\mathcal{C}$. Não é difícil ver que podemos encontrar um tal conjunto em tempo polinomial. Note que procuramos um conjunto minimal (não necessariamente mínimo).

Teorema 4.2.6 *Se D é um digrafo fortemente conexo $\{0,1\}$ -ponderado e $\mathcal{C}$ uma base de D consistindo de circuitos de peso ímpar, então as seguintes afirmações são equivalentes:*

- (1) D é par;
- (2) D tem um circuito de peso par;
- (3) D tem um circuito que é a diferença simétrica de um número par de circuitos de $\mathcal{C}$;
- (4) D contém circuitos $C_1, C_2, \dots, C_k$, tal que k é ímpar e $C_1 \oplus C_2 \oplus \dots \oplus C_k = \emptyset$.

Prova: (1) $\Rightarrow$ (2): Trivial.

(2) $\Rightarrow$ (3): Seja C um circuito de peso par. Então temos que $C = C_1 \oplus C_2 \oplus \dots \oplus C_j$, onde cada C_i é um membro de $\mathcal{C}$. Uma vez que cada circuito C_i tem peso ímpar, temos que j deve ser par.

(3) $\Rightarrow$ (4): Trivial.

(4) $\Rightarrow$ (1): Seja $p: A_D \rightarrow \{0,1\}$ uma atribuição de pesos aos arcos de D . Sejam $p_1, p_2, \dots, p_k$ os pesos dos circuitos $C_1, C_2, \dots, C_k$, respectivamente. Por hipótese temos que $C_1 \oplus C_2 \oplus \dots \oplus C_k = \emptyset$, onde k é ímpar. Então cada arco de $A_{C_1} \cup \dots \cup A_{C_k}$ é coberto por um número par de circuitos. Assim,

$$p_1 + p_2 + \dots + p_k \equiv 0 \pmod{2}.$$

Uma vez que k é ímpar, pelo menos um dos p_i ($i = 1, 2, \dots, k$) deve ser par.

□

Estamos agora em condições de provar o resultado que prometemos no início desta subseção.

Teorema 4.2.7 *O PCirPPar é polinomialmente equivalente ao PDigrafoPar.*

Prova: Suponha que para resolver PCirPPar(D) exista um algoritmo polinomial, onde D é um digrafo fortemente conexo. Selecione uma base $\mathcal{C}$ de D . Isso pode ser feito em tempo polinomial usando o Teorema 4.2.3. Pelo Teorema 4.2.5, podemos atribuir pesos 0/1 aos arcos de D de tal forma que os circuitos de $\mathcal{C}$ tenham peso ímpar. Pelo Teorema 4.2.6, D tem um circuito de peso par se, e somente se, D é um digrafo par.

Por outro lado, suponha que exista um algoritmo que resolve o PDigrafoPar(D). Como descrito acima, selecione uma base $\mathcal{C}$ de D . Se algum circuito de $\mathcal{C}$ tiver peso par, então o problema está resolvido. Suponha então que todos os circuitos de $\mathcal{C}$ tenham peso ímpar. Use o algoritmo que resolve PDigrafoPar(D) e aplique o Teorema 4.2.6.

□

Tendo em vista a equivalência dos problemas PCirPar, PCirPPar e PDigrafoPar, seria interessante encontrar uma caracterização de digrafos pares. Apresentaremos na próxima subseção, uma tal caracterização, que foi obtida por SEYMOUR E THOMASSEN [1987].

4.2.3 O problema do circuito duplo-fraco ímpar

Nesta subseção definiremos o problema do circuito duplo-fraco ímpar e mostraremos que esse problema é polinomialmente equivalente ao PDigrafoPar. Veremos ainda uma caracterização de digrafos pares obtida por SEYMOUR E THOMASSEN [1987]: um digrafo é par se, e somente se, ele contém um circuito duplo-fraco ímpar. Definiremos também os seguintes problemas de decisão: verificar se um digrafo contém uma subdivisão de um circuito duplo, uma subdivisão de um circuito duplo ímpar, ou uma subdivisão de um 3-circuito duplo. Veremos que esses três problemas são $\mathcal{NP}$ -completos.

Dado um grafo G , o **digrafo simétrico** D , associado a G , é o digrafo obtido a partir de G , substituindo cada aresta de G por um circuito de comprimento dois (veja um exemplo na Figura 4.1).

Seja S_k um grafo que é um circuito de comprimento k . O digrafo simétrico associado a S_k é chamado **k -circuito duplo** e será denotado por S_k^* . Um **k -circuito duplo-fraco** é um digrafo obtido pela subdivisão sucessiva de alguns (eventualmente todos) vértices e pela subdivisão sucessiva de alguns (eventualmente todos) arcos de S_k^* (veja um exemplo na Figura 4.2).

Vamos chamar de **circuito duplo-fraco ímpar** um k -circuito duplo-fraco, para um k ímpar. A seguir damos a definição do problema do circuito duplo-fraco ímpar.

Figura 4.1: Digrafo simétrico D obtido a partir de um grafo G .

Problema do circuito duplo-fraco ímpar: $\text{PCirDFImp}(D)$

Dado: Um digrafo D .

Problema: Decidir se D contém um circuito-duplo fraco ímpar.

Para demonstrarmos a equivalência entre PDigrafoPar e PCirDFImp , precisaremos do seguinte resultado obtido por THOMASSEN [1985b].

Teorema 4.2.8 (Thomassen) *Se D é um digrafo tal que para cada vértice $v \in V_D$, $D - v$ tem um circuito, então, ou D tem dois circuitos disjuntos nos vértices, ou D contém um 3-circuito duplo-fraco.*

Se G é um grafo conexo e B um bloco de G , então dizemos que cada componente de $G - A_B$ é um **componente adjacente** a B (veja um exemplo na Figura 4.3).

Dizemos que um grafo G é **2-aresta conexo minimal** se para cada aresta $\alpha \in A_G$, $G - \alpha$ é conexo e contém uma aresta de corte.

Teorema 4.2.9 *Se G é um grafo 2-aresta conexo minimal, então*

- (1) *cada bloco de G é 2-aresta conexo minimal;*
- (2) *se existem arestas múltiplas, estas ocorrem aos pares e constituem blocos de G ;*
- (3) *se G é 2-conexo e todos os vértices de G têm grau par, então G tem pelo menos $n/2 + 1$ vértices de grau 2.*

Figura 4.2: O 5-circuito duplo (S_5^*) e um 5-circuito duplo-fraco.

Prova: As afirmações (1) e (2) são triviais. Vamos demonstrar a afirmação (3) por indução em n .

Se $n < 6$, então G é um circuito. Considere, então, que $n \geq 6$. Suponha, por hipótese de indução, que para qualquer grafo 2-aresta-conexo minimal com número de vértices menor que n , vale (3).

Seja G um grafo 2-aresta conexo minimal com n vértices e seja α_1 uma aresta de G . Então $G - \alpha_1$ contém uma aresta de corte α_2 . Logo, $G - \{\alpha_1, \alpha_2\}$ tem exatamente dois componentes, digamos G_1 e G_2 . Considere $n_1 = |V_{G_1}|$ e $n_2 = |V_{G_2}|$. Temos dois casos para analisar.

CASO 1: α_1 e α_2 podem ser escolhidos de forma que n_1 e n_2 sejam maiores ou iguais a 2.

Seja G'_1 (G'_2 , respectivamente) o grafo obtido a partir de G pela contração do grafo G_2 (G_1 , respectivamente). É fácil ver que, para $i = 1, 2$, G'_i é 2-conexo e todos os seus vértices têm grau par. Claramente, para $i = 1, 2$, G'_i é 2-aresta conexo minimal. Assim, por hipótese de indução, cada G'_i tem pelo menos $\frac{1}{2}(n_i + 1) + 1$ vértices de grau 2. Dessa forma, G tem pelo menos $(\frac{1}{2}(n_1 + 1) + 1) + (\frac{1}{2}(n_2 + 1) + 1) = n/2 + 1$ vértices de grau 2.

Figura 4.3: Um bloco B e seus componentes adjacentes $B_1, B_2, \dots, B_k$.

CASO 2: α_1 e α_2 não podem ser escolhidos de forma que n_1 e n_2 sejam ambos maiores ou iguais a 2.

Neste caso, cada aresta de G incide em um vértice de grau 2, e assim, dois vértices, de grau maior ou igual a 4, não podem ser adjacentes. Se G tem k vértices de grau 2, então G contém $n - k$ vértices de grau pelo menos 4. Dessa forma, G contém pelo menos $4(n - k)$ arestas. Como cada aresta de G incide em um vértice de grau 2, então G contém no máximo $2k$ arestas. Portanto $4(n - k) \leq 2k$, ou seja,

$$k \geq \frac{2}{3}n \geq \frac{n}{2} + 1, \text{ pois } n \geq 6.$$

□

Um digrafo conexo é um **digrafo euleriano** se cada um de seus vértices tem grau de entrada igual ao grau de saída. Um grafo conexo é um **grafo euleriano** se todos os seus vértices têm grau par.

Proposição 4.2.10 *Um digrafo (respectivamente, um grafo) é euleriano se, e somente se, ele é a união de uma coleção de circuitos, dois a dois, disjuntos nos arcos (respectivamente, arestas).*

Proposição 4.2.11 *Se D é um digrafo fortemente conexo e euleriano então D contém um circuito C , tal que $D - A_C$ é a união de um componente fortemente conexo, juntamente com um conjunto de vértices isolados.*

Prova: Dentre todos os circuitos de D , escolha um circuito, digamos C , tal que um componente fortemente conexo, digamos D' , de $D - A_C$ tenha o maior número possível de vértices.

Suponha por absurdo que $D - A_C$ contenha um outro componente fortemente conexo, digamos D'' , cujo conjunto de arcos seja não vazio. Como D'' é euleriano, ele contém um circuito C' .

Como $A_C \cap A_{C'} = \emptyset$, temos que o digrafo obtido pela união de D' com o circuito C é um componente fortemente conexo de $D - A_{C'}$. Mas isso é um absurdo, pois contraria a escolha do circuito C .

□

SEYMOUR E THOMASSEN [1987] obtiveram a seguinte caracterização de digrafos pares.

Teorema 4.2.12 *Um digrafo é par se, e somente se, contém um circuito duplo-fraco ímpar.*

Prova: ($\Leftarrow$) Seja C um k -circuito duplo-fraco, com k ímpar. Sejam $C_1, C_2, \dots, C_{k+2}$ todos os circuitos de C . Faça uma atribuição de pesos 0/1 aos arcos de C . Sejam $p_1, p_2, \dots, p_{k+2}$ os pesos dos circuitos $C_1, C_2, \dots, C_{k+2}$, respectivamente.

Uma vez que todo arco de C é coberto por um número par de circuitos, temos que $p_1 + p_2 + \dots + p_{k+2}$ é um número par. Como $k + 2$ é ímpar, temos que pelo menos um dos p_i ($i = 1, 2, \dots, k + 2$) deve ser par. Dessa forma, C é um digrafo par.

($\Rightarrow$) Suponha por absurdo que o teorema seja falso, e seja D um contra-exemplo com o menor número possível de vértices. Uma vez que D é par se, e somente se, cada um de seus componentes fortes é par, podemos assumir que D é fortemente conexo. Provaremos a seguir vários resultados a respeito de D .

Fato 1: D tem pelo menos quatro vértices.

Demonstração: Uma vez que S_3^* é o menor digrafo par, D deve ter pelo menos quatro vértices.

Fato 2: D é 2-conexo.

Demonstração: Suponha por absurdo que D não seja 2-conexo. Então existe um vértice v , tal que $D' = D - v$ não é fortemente conexo. Dessa forma, $V_{D'}$ tem uma partição em 2 conjuntos, digamos A_1 e A_2 , tais que não existem arcos de A_2 para A_1 .

Sejam $D_1 = D[A_1 + v]$ e $D_2 = D[A_2 + v]$. Para $i = 1, 2$, construa o digrafo H_i tal que:

Figura 4.4: Construção dos digrafos H_1 e H_2 a partir do digrafo D .

$$V_{H_i} = V_{D_i},$$

$$A_{H_1} = A_{D_1} \cup \{ (x, v) \mid (x, y) \in A_D, x \in A_1 \text{ e } y \in A_2 \} \text{ e}$$

$$A_{H_2} = A_{D_2} \cup \{ (v, y) \mid (x, y) \in A_D, x \in A_1 \text{ e } y \in A_2 \}.$$

Afirmamos que H_1 ou H_2 é um digrafo par (veja Figura 4.4). De fato, suponha por absurdo que ambos os digrafos não sejam pares. Faça uma atribuição de pesos 0/1 aos arcos dos digrafos H_1 e H_2 , de tal forma que H_1 e H_2 não tenham circuitos de peso par. Note que para cada arco (x, y) , de A_1 para A_2 , D contém um circuito $C_{x,y}$ que passa por (x, y) . Observe que $C_{x,y}$ contém o vértice v . É fácil ver que podemos atribuir pesos 0/1 a cada um desses arcos (x, y) , de modo que o correspondente circuito $C_{x,y}$ tenha peso ímpar.

Uma vez que D é um digrafo par, ele contém um circuito C de peso par que não está inteiramente em D_1 e em D_2 . Logo C deve conter um arco (x, y) , de A_1 para A_2 , e o vértice v (veja Figura 4.5). Assim, esse circuito contém um subcaminho P_1 de v a x e um outro subcaminho Q_1 de y a v . Analogamente o circuito $C_{x,y}$ contém um subcaminho P_2 de v a x e um outro subcaminho Q_2 de y a v . Como C e $C_{x,y}$ têm paridades diferentes, P_1 e P_2 , ou Q_1 e Q_2 , devem ter paridades diferentes. Suponha, sem perda de generalidade,

Figura 4.5: H_1 ou H_2 é um digrafo par.

que Q_1 e Q_2 tenham paridades diferentes. Então H_2 tem dois circuitos, que passam por (v, y) , cujas paridades são diferentes. Mas isso é um absurdo, uma vez que assumimos que H_2 não é um digrafo par.

Portanto H_1 ou H_2 é um digrafo par. Suponha, sem perda de generalidade, que H_1 seja um digrafo par. Assim, pela minimalidade de D , o digrafo H_1 contém um circuito duplo-fraco ímpar C_F . Observe que C_F não está inteiramente contido em D_1 , pois caso contrário, ele também estaria em D . Logo, C_F contém um arco ou dois arcos da forma (x, v) que não estão em D_1 .

Seja então (x, y) um arco de A_1 para A_2 , tal que o circuito duplo-fraco ímpar C_F contém o arco (x, v) , que não está em D_1 . Seja $P_{y,v}$ um caminho, em H_2 , de y a v . Temos dois casos para analisar:

CASO 1 : C_F não contém um arco (x', v) , onde $x' \neq x$, que não está em D_1 (Figura 4.6).

Nesse caso, $((C_F - (x, v)) + (x, y)) \cup P_{y,v}$ é um circuito duplo-fraco ímpar em D .

CASO 2 : C_F contém um arco (x', v) , onde $x' \neq x$, que não está em D_1 (Figura 4.7).

Nesse caso, D contém um arco (x', y') , tal que $y' \in A_2$. Seja $P_{y',v}$ um caminho de y' a v em H_2 . Seja w o primeiro vértice de $P_{y',v}$ que pertence a $P_{y',v}$. Seja Q o subcaminho de $P_{y',v}$, de y' a w . É fácil ver que $((C_F - \{(x, v), (x', v)\}) \cup Q) \cup P_{y',v} \cup \{(x, y), (x', y')\}$ é um circuito duplo-fraco ímpar em D .

Figura 4.6: Caso 1 do Fato 2.

Fato 3: Se $E \subseteq A_D$ é tal que $D - E$ é fortemente conexo, então D contém um circuito contendo todos os arcos de E .

Demonstração: Pelo Lema 4.2.2, temos que existe uma base $\mathcal{C}$ tal que cada arco de E está exatamente em um circuito de $\mathcal{C}$. Pelo Lema 4.2.5, podemos atribuir pesos 0/1 aos arcos de D de forma que cada circuito de $\mathcal{C}$ tenha peso ímpar. Como D é um digrafo par, então existe um circuito C_0 de peso par. Assim $C_0 = C_1 \oplus C_2 \oplus \dots \oplus C_k$, onde cada C_i é um circuito em $\mathcal{C}$. Uma vez que C_0 tem peso par e cada C_i , para $i = 1, 2, \dots, k$ tem peso ímpar, temos que k deve ser par.

Seja $D' = C_1 \cup C_2 \cup \dots \cup C_k$. Pelo Teorema 4.2.6, D' é um digrafo par. Afirmamos que $D' = D$. De fato, suponha que D' seja um subdigrafo próprio de D . Então, pela minimalidade de D , o digrafo D' contém um circuito duplo-fraco ímpar C . Como D' é um subdigrafo próprio de D , então o digrafo D contém C . Mas isso é um absurdo. Logo, $D' = D$.

Dessa forma, como cada arco α de E está exatamente em um dos circuitos $C_1, C_2, \dots, C_k$, então todos os arcos de E estão em C_0 .

Fato 4: Todos os vértices de D têm grau de entrada e grau de saída exatamente igual a 2. Conseqüentemente D é euleriano.

Demonstração: Como D é 2-conexo, todos os vértices de D têm grau de entrada e grau de saída pelo menos 2. Suponha por absurdo que D contenha um vértice v que tenha, ou

Figura 4.7: Caso 2 do Fato 2.

grau de saída, ou grau de entrada, pelo menos 3. Sejam $\alpha_1, \alpha_2, \alpha_3$ arcos de D , que saem ou entram em v . Como $D - v$ é fortemente conexo, então $D - \{\alpha_1, \alpha_2\}$ é fortemente conexo. Pelo Fato 3, existe um circuito C_0 que contém os arcos α_1 e α_2 . Mas isso é um absurdo, uma vez que α_1 e α_2 , ou entram, ou saem de um mesmo vértice v .

Fato 5: D contém um circuito C , tal que $D' = D - A_C$ é fortemente conexo. Além disso, cada bloco do grafo subjacente de D' é 2-aresta-conexo minimal.

Demonstração: Como D é euleriano e fortemente conexo então, pela Proposição 4.2.11 e pelo Fato 4, temos que D contém um circuito C tal que $D' = D - A_C$ é fortemente conexo.

Afirmamos que não existe um arco α em D' , tal que $D' - \alpha$ seja fortemente conexo. De fato, suponha que exista um tal arco α . Então, pelo Fato 3, D contém um circuito C' que contém $A_C + \alpha$. Logo, o circuito C é um subdigrafo próprio de C' . Mas isto é um absurdo, pois nenhum circuito contém um outro circuito como subdigrafo próprio. Portanto, para qualquer arco α em D' , o digrafo $D' - \alpha$ não é fortemente conexo.

Seja β um arco de D' e $H = D' - \beta$. Como H não é fortemente conexo, então V_H tem uma partição em 2 conjuntos, digamos A_1 e A_2 , tal que não existem arcos de A_2 para A_1 .

Uma vez que D é euleriano, então D' é euleriano e portanto, em D' , o número de arcos, de A_1 para A_2 , é igual ao número de arcos, de A_2 para A_1 . Além disso, como D'

é fortemente conexo, o número de arcos, de A_1 para A_2 , é maior que zero. Logo, β é um arco de A_2 para A_1 . Assim, H tem exatamente um arco β' , de A_1 para A_2 , e portanto tal arco é uma ponte no grafo subjacente de H . Dessa forma, o grafo subjacente de D' é 2-aresta-conexo minimal, e pelo Teorema 4.2.9, cada bloco desse grafo é 2-aresta-conexo minimal.

Fato 6: Se D não contém circuitos de comprimento 2, então D contém um circuito C de comprimento máximo $(n+1)/2$, tal que o grafo subjacente de $D - A_C$ é 2-conexo.

Demonstração: Como D é 2-conexo, pelo Teorema 4.2.8, temos que D tem 2 circuitos disjuntos nos vértices. Assim D tem um circuito de comprimento máximo $(n+1)/2$. Dentre todos os circuitos de comprimento máximo $(n+1)/2$ escolha um circuito C tal que:

- (1) Maximize o número de vértices de um bloco B , do grafo subjacente G de $D' = D - A_C$;
- (2) Sujeito a (1), maximize o número de vértices no componente de G que contém B .

Observe que, pelo Fato 4, todos os vértices de G têm grau 2 ou 4. Além disso, como D é 2-conexo e euleriano, $D - A_C$ é um digrafo conexo, e conseqüentemente G é conexo. Se $G = B$, então G é 2-conexo e não há mais o que provar. Suponha que G contém um componente adjacente, digamos B' , ao bloco B . Note que $D[A_{B'}]$ é um digrafo euleriano, e portanto contém um circuito, digamos C' . Devido à maximalidade de B , o circuito C' tem comprimento máximo $(n+1)/2$.

Uma vez que D' é euleriano, podemos concluir que ele é um digrafo fortemente conexo. Pelo Fato 5, cada bloco de G é 2-aresta-conexo minimal, ou seja, B é 2-aresta-conexo minimal. Como B não tem circuito de comprimento 2, temos que $|V_B| \geq 3$ e, pelo Teorema 4.2.9.(3), B tem pelo menos 3 vértices de grau 2. Além disso, qualquer vértice de grau 2, em B , está no circuito C , ou em outro bloco distinto de B (isto é, esse vértice é um ponto de articulação). Uma vez que D é 2-conexo, o circuito C contém pelo menos 1 vértice de cada componente adjacente a B . Temos então 3 casos para analisar:

CASO 1 : O bloco B tem somente o componente adjacente B' .

Como B tem somente 1 ponto de articulação e pelo menos 3 vértices de grau 2, então B contém pelo menos 2 vértices do circuito C , digamos x e y (veja Figura 4.8.a) Além disso, o circuito C contém pelo menos 1 vértice de B' (digamos z). Portanto, $B \cup C$ é 2-conexo e está contido em um bloco do grafo subjacente de $D - A_{C'}$. Mas isso é um absurdo, pois contradiz a maximalidade de B .

Figura 4.8: Figuras dos casos 1, 2 e 3 do Fato 6.

CASO 2 : O bloco B tem exatamente 2 componentes adjacentes.

Como B tem exatamente 2 pontos de articulação e pelo menos 3 vértices de grau 2, então B contém pelo menos 1 vértice, digamos z , que pertence ao circuito C . Seja B'' o componente adjacente a B , distinto de B' (veja Figura 4.8.b). Seja v um vértice de B'' que pertence ao circuito C . Seja P um caminho de v a z em G . Observe que P contém dois vértices de C (v e z), passa por dois vértices de B (z e o ponto de articulação y entre os blocos B e B'') e não passa pelo componente adjacente B' . Dessa forma, $B \cup C \cup P$ é 2-conexo e está contido em um bloco do grafo subjacente de $D - A_{C'}$. Mas isso é um absurdo, pois contradiz a maximalidade de B .

CASO 3 : O bloco B tem mais que 3 componentes adjacentes.

Sejam B'' e B''' dois componentes adjacentes a B , distintos de B' . Sejam $u \in B''$ e $v \in B'''$ dois vértices que estão no circuito C (veja Figura 4.8.c). Seja P um caminho de u a v em G . Observe que P contém 2 pontos de articulação distintos de B (o ponto de

articulação entre B e B'' e outro ponto de articulação entre B e B'''), passa por 2 vértices do circuito C , u e v , e não passa pelo componente adjacente B' . Dessa forma, $B \cup C \cup P$ é 2-conexo e está contido em um bloco do grafo subjacente de $D - A_C$. Mas isso é um absurdo, pois contradiz a maximalidade de B .

Fato 7: D tem um circuito de comprimento 2.

Demonstração: Suponha por absurdo que D não tenha circuitos de comprimento 2. Então, seja C um circuito satisfazendo o Fato 6. Pelo Fato 4, $D' = D - A_C$ é euleriano, e portanto D' é fortemente conexo. Seja G o grafo subjacente de D' . Pelo Fato 5, cada bloco de G é 2-aresta-conexo minimal, e, pelo Fato 6, G é 2-conexo. Logo, pelo Lema 4.2.9.(3), G tem pelo menos $(n/2 + 1)$ vértices de grau 2.

Como todo vértice de grau 2 deve estar no circuito C (uma vez que todo vértice de D tem grau 4), o circuito C tem pelo menos $(n/2 + 1)$ vértices. Mas isto é um absurdo, pois, pelo Fato 6, C tem no máximo $(n + 1)/2$ vértices.

Fato 8: D é um circuito duplo.

Demonstração: Pelo Fato 7, o digrafo D tem um circuito, digamos C , de comprimento 2. Uma vez que D é 2-conexo e euleriano, temos que $D' = D - A_C$ é fortemente conexo. Seja G o grafo subjacente de D' . Pelo Fato 5, cada bloco de G é 2-aresta-conexo minimal, ou equivalentemente, G é 2-aresta-conexo minimal.

Seja B um bloco qualquer de G . Podemos afirmar que o número de blocos adjacentes a B é no máximo 2 (veja Figura 4.9), uma vez que o circuito C (de comprimento 2) deve conter pelo menos 1 vértice de cada bloco adjacente a B (isto devido ao fato de D ser 2-conexo). Dessa forma, cada bloco de G contém no máximo 2 pontos de articulação e, pelo Fato 4, concluímos que cada bloco tem no máximo 2 vértices de grau 2 (restringindo-se somente ao bloco).

Sejam $B_1, B_2, \dots, B_k$ os blocos de G . Pelo Teorema 4.2.9.(3), cada bloco B_i ($1 \leq i \leq k$) tem pelo menos $|V_{B_i}|/2 + 1$ vértices de grau 2. Logo,

$$\frac{|V_{B_i}|}{2} + 1 \leq 2, \quad \text{para } i = 1, 2, \dots, k$$

ou,

$$|V_{B_i}| \leq 2, \quad \text{para } i = 1, 2, \dots, k.$$

Uma vez que $|V_{B_i}| \geq 2$ ($i = 1, 2, \dots, k$), temos que cada bloco de G é um circuito de comprimento 2 e conseqüentemente, D é um circuito duplo.

Figura 4.9: Um circuito C de comprimento 2 do digrafo D , um bloco B do grafo subjacente a $D - A_C$ e seus componentes adjacentes B_1, B_2 .

Assumimos no início que D é um digrafo par e não contém um circuito duplo-fraco ímpar. Pelo Fato 8, D é um circuito duplo, e portanto tem um número par de vértices. Mas então é fácil encontrar uma atribuição de pesos 0/1 aos arcos de D , de forma que D não tenha circuito de peso par (veja um exemplo na Figura 4.10). Mas isso é um absurdo, uma vez que D é um digrafo par.

□

Como consequência deste último teorema e do Teorema 4.2.6, temos o seguinte corolário.

Corolário 4.2.13 *Se D é um digrafo fortemente conexo contendo uma base de circuitos de pesos ímpares, então D tem um circuito de peso par se, e somente se, D contém um circuito duplo-fraco ímpar.*

SEYMOUR E THOMASSEN [1987] demonstraram que os seguintes problemas são $\mathcal{NP}$ -completos:

1. **problema do circuito duplo:** decidir se um digrafo contém uma subdivisão de um circuito duplo;
2. **problema do circuito duplo ímpar:** decidir se um digrafo contém uma subdivisão de um circuito duplo ímpar;

D

Figura 4.10: Um exemplo de uma atribuição de pesos 0/1 aos arcos do circuito duplo S_4^* , tal que todos os seus circuitos tenham peso ímpar.

3. **problema do 3-circuito duplo:** decidir se um digrafo contém uma subdivisão de um 3-circuito duplo;

Teorema 4.2.14 *Os problemas do circuito duplo, do circuito duplo ímpar e do 3-circuito duplo são $\mathcal{NP}$ -completos.*

Prova: Considere o seguinte problema: dados um digrafo D e dois vértices distintos x e y , D contém um circuito que passa por x e y ? FORTUNE, HOPCROFT E WYLLIE [1980] mostraram que esse problema é $\mathcal{NP}$ -completo (esse resultado e outros correlatos podem ser encontrados em BENATTI [1993]).

Seja D' um digrafo e x e y dois vértices distintos de D' . Seja D o digrafo obtido de D' pela subdivisão de todos os vértices que são distintos de x e y . Adicione um novo vértice z ao digrafo D e os arcos (x, z) , (z, x) , (y, z) e (z, y) (veja um exemplo na Figura 4.11).

Uma vez que todos os vértices de D' foram subdivididos, com exceção de x e y , qualquer subdivisão de um circuito duplo em D , usa necessariamente os vértices x , y e z . Dessa forma, é fácil ver que D' contém um circuito que passa por x e y se, e somente se, existe uma subdivisão de um circuito duplo (ou de um circuito duplo ímpar, ou de um 3-circuito duplo) em D .

□

Figura 4.11: Redução para os problemas do circuito duplo (ou do circuito duplo ímpar, ou do 3-circuito duplo).

Enquanto o problema de decidir a existência de uma subdivisão de circuito duplo ímpar é $\mathcal{NP}$ -completo, a seguinte questão está em aberto: existe um algoritmo polinomial para encontrar um circuito duplo-fraco ímpar? Ou, equivalentemente, decidir se um digrafo fortemente conexo pode ser ponderado com pesos 0/1 de tal forma que D não tenha nenhum circuito de peso par.

O resultado do Teorema 4.2.14 e a caracterização de digrafos pares, obtida por SEYMOUR E THOMASSEN [1987], parece indicar que o problema do digrafo par é “difícil”.

4.2.4 O problema do esquema de Polya

Definimos uma **permutação** de grau n como sendo uma função bijetora $\sigma: \{1, \dots, n\} \rightarrow \{1, \dots, n\}$. Tal função corresponde a uma n -upla $(\sigma_1, \sigma_2, \dots, \sigma_n)$ e chamamos de **permutação identidade** a n -upla $(1, 2, \dots, n)$. Além disso, denotamos por S_n o conjunto de todas as permutações de grau n .

O **permanente** de uma matriz quadrada $A = (a_{i,j})$, de ordem n , é definido como:

$$\text{perm}(A) = \sum_{\sigma \in S_n} \text{valor}_A(\sigma),$$

onde

$$\text{valor}_A(\sigma) = \prod_{i=1}^n a_{i,\sigma(i)}.$$

Observe que uma permutação $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_n)$ pode ser obtida a partir da permutação identidade fazendo um número finito de trocas de pares de elementos. Por

exemplo, se $\sigma_1 \neq 1$, podemos trocar 1 e σ_1 , obtendo $(\sigma_1, 2, \dots, 1, \dots, n)$ e repetindo esse processo enquanto necessário, chegaremos à permutação σ . Na verdade, há várias maneiras de se fazer trocas de pares de elementos para obtermos uma permutação σ a partir da permutação identidade. No entanto, é bem conhecido o fato de que o número de trocas é sempre par ou ímpar. Dessa forma, dizemos que σ é uma **permutação par** (**permutação ímpar**, respectivamente) se o número de trocas para atingir a permutação σ , partindo da permutação identidade é par (ímpar, respectivamente).

O sinal de uma permutação σ é definido da seguinte forma:

$$\text{signal}(\sigma) = \begin{cases} 1 & \text{se } \sigma \text{ é uma permutação par;} \\ -1 & \text{se } \sigma \text{ é uma permutação ímpar.} \end{cases}$$

Definimos o **determinante** de uma matriz quadrada A , de ordem n , como:

$$\det(A) = \sum_{\sigma \in S_n} \text{signal}(\sigma) \cdot \text{valor}(\sigma).$$

Seja C um subconjunto finito de números inteiros. Uma matriz A é uma **C -matriz**, se A é uma matriz quadrada cujos elementos pertencem ao conjunto C . Assim, por exemplo, uma $\{0,1\}$ -matriz A é uma matriz cujos elementos são 0 ou 1.

POLYA [1913] sugeriu uma idéia para calcular o permanente de uma $\{0,1\}$ -matriz A trocando algumas entradas $+1$ por -1 , resultando uma matriz B , de tal forma que $\det(B) = \text{perm}(A)$. No entanto, ele mostrou que tal transformação nem sempre é possível. Definimos a seguir o problema do esquema de Polya.

Problema do esquema de Polya: $\text{PEsqPolya}(A)$

Dado: Uma $\{0,1\}$ -matriz A .

Problema: Decidir se existe uma $\{-1,0,1\}$ -matriz quadrada B , da mesma ordem que A , tal que $\det(B) = \text{perm}(A)$.

Mostraremos nesta subseção que o problema do esquema de Polya é polinomialmente equivalente ao problema do digrafo par. Para isso, primeiramente, vamos analisar o permanente de uma $\{0,1\}$ -matriz A . A partir dessa matriz, podemos construir um grafo bipartido G , com bipartição $X = \{x_1, x_2, \dots, x_n\}$ e $Y = \{y_1, y_2, \dots, y_n\}$, cuja matriz de adjacência é $\begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$.

Observe que uma permutação σ é um emparelhamento perfeito em G se, e somente se, para cada $i = 1, 2, \dots, n$, temos que $\{x_i, y_{\sigma(i)}\} \in A_G$. Vamos chamar tal permutação

de **permutação perfeita**. Dessa forma, podemos relacionar uma permutação perfeita com um emparelhamento perfeito da seguinte maneira:

$$\text{valor}(\sigma) = \begin{cases} 1 & \text{se } \sigma \text{ é um emparelhamento perfeito} \\ 0 & \text{caso contrário.} \end{cases}$$

E conseqüentemente, $\text{perm}(A)$ é exatamente o número de emparelhamentos perfeitos de G .

Note que se A é uma $\{0,1\}$ -matriz e B é uma $\{-1,0,1\}$ -matriz, então $\text{perm}(A) = \det(B)$ se, e somente se, para qualquer permutação perfeita (ou emparelhamento perfeito) σ , temos que $\text{sign}(\sigma) = \text{valor}_B(\sigma)$.

Seja D um digrafo, com $V_D = \{v_1, v_2, \dots, v_n\}$. Um subdigrafo C_B de D é uma **cobertura de circuitos** de ordem n , se $V_{C_B} = V_D$ e cada vértice de C_B tem grau de entrada e grau de saída exatamente igual a 1. Ou seja, C_B é um subdigrafo gerador de D que consiste de circuitos dois a dois disjuntos nos vértices. Observe que uma cobertura de circuitos de um digrafo D , de ordem n , define uma permutação σ de grau n , e vice-versa, da seguinte forma:

$$\sigma(i) = j \text{ se, e somente se, } (v_i, v_j) \in A_{C_B}.$$

Assim a permutação identidade define uma cobertura de circuitos tal que todos os circuitos são laços. Chamamos tal cobertura de **cobertura identidade**.

É fácil ver que, se D contém um laço em cada vértice, podemos construir uma cobertura de circuitos C_B , a partir de um circuito C , da seguinte forma:

$$V_{C_B} = V_D; A_{C_B} = \{ (v, v) \mid v \notin V_C \} \cup A_C.$$

Vamos chamar tal cobertura de **cobertura singular** e o circuito C de **circuito singular** de C_B . Se σ é a permutação definida por essa cobertura acima, então é fácil determinar quantas trocas, entre pares de elementos, podemos fazer para chegar, partindo de σ , à permutação identidade. Basta simplesmente sabermos o comprimento do circuito C . Essa afirmação pode ser verificada através da seguinte proposição:

Proposição 4.2.15 *Se σ é a permutação definida por uma cobertura singular C_B e C é o circuito singular dessa cobertura, então partindo da permutação σ podemos chegar à permutação identidade com $|V_C| - 1$ trocas de pares de elementos.*

Prova: Vamos fazer indução em $|V_C|$. Para $|V_C| = 1$, temos que σ é a própria permutação identidade. Suponha por hipótese de indução que para $1 < |V_C| < r$, vale a tese.

Figura 4.12: Trocando v_i e v_j no circuito C da cobertura singular C_B , obteremos uma nova cobertura singular C'_B que contém um novo circuito C' e um novo laço (v_i, v_i)

Suponha que $|V_C| = r$. Seja v_i um vértice de C e sejam v_k e v_j dois vértices de C tais que $(v_k, v_i), (v_i, v_j) \in A_C$. Dessa forma, $\sigma(k) = i$ e $\sigma(i) = j$. Trocando os pares i e j em σ obteremos uma nova permutação σ' tal que $\sigma'(i) = i$, $\sigma'(k) = j$ e $\sigma'(l) = \sigma(l)$ para $l \neq i$ e $l \neq k$. Seja C'_B a cobertura de circuitos correspondente à permutação σ' . Esta cobertura é singular, pois contém um circuito $C' = (C - v_i) + (v_k, v_j)$ de comprimento $r - 1$, e todos os outros circuitos são laços (veja a Figura 4.12). Pela hipótese de indução, partindo da permutação σ' podemos atingir a permutação identidade por meio de $r - 2$ trocas de pares de elementos. Portanto, partindo da permutação σ podemos chegar à permutação identidade por meio de $r - 1$ trocas.

□

A próxima proposição mostra que podemos determinar facilmente o sinal (ou a paridade) de uma permutação, definida por uma cobertura de circuitos, simplesmente sabendo o número de circuitos pares contidos nessa cobertura.

Proposição 4.2.16 *Se k é o número de circuitos pares de uma cobertura de circuitos C_B , então $\text{ sinal}(\sigma) = (-1)^k$, onde σ é a permutação correspondente a C_B .*

Prova: Uma permutação é par se o número de trocas para se chegar, partindo da permutação σ , à permutação identidade é par. Pela Proposição 4.2.15, temos que para um circuito C , da cobertura de circuitos C_B , podemos fazer $|V_C| - 1$ trocas de pares de elementos para que cada vértice de C passe a ser um laço. Sejam $C_1, C_2, \dots, C_k$ os circuitos

de comprimento par e $C'_1, C'_2, \dots, C'_l$ os circuitos de comprimento ímpar de C_B . Então podemos fazer

$$\sum_{i=1}^k (|V_{C_i}| - 1) + \sum_{i=1}^l (|V_{C'_i}| - 1)$$

trocas para obtermos a permutação identidade, a partir da permutação σ . Dessa forma o número de trocas é par se, e somente se, k é par.

□

Podemos relacionar, através do próximo teorema, o permanente de $\{0,1\}$ -matrizes, através do esquema de Polyá, com digrafos que não são pares.

Teorema 4.2.17 *Se A é uma $\{0,1\}$ -matriz de ordem n , com todos os elementos da diagonal principal iguais a 1 e D o digrafo cuja matriz de adjacência é A , então, existe uma matriz B satisfazendo o esquema de Polyá se, e somente se, D não é um digrafo par.*

Prova: ($\Leftarrow$) Suponha que D não seja um digrafo par e que $V_D = \{v_1, v_2, \dots, v_n\}$. Faça uma atribuição de pesos 0/1 aos arcos de D , de tal forma que D não tenha circuito de peso par. Construa a matriz B da seguinte forma:

$$b_{i,j} = \begin{cases} -1 & \text{se } a_{i,j} = 1 \text{ e } (v_i, v_j) \text{ tem peso } 0; \\ a_{i,j} & \text{caso contrário.} \end{cases}$$

Vamos mostrar que para toda permutação σ associada a uma cobertura de circuitos de D , temos $\text{senal}(\sigma) = \text{valor}_B(\sigma)$.

Note que circuitos de comprimento par (ímpar) tem um número ímpar (par) de arcos de peso zero.

Seja C_B uma cobertura de circuitos de D e σ a permutação associada a essa cobertura. Suponha que σ seja uma permutação par. Então, pela Proposição 4.2.16, C_B tem um número par de circuitos de comprimento par, e portanto tem um número par de arcos de peso 0. Dessa forma, $\text{valor}_B(\sigma) = 1$. Analogamente, se σ é uma permutação ímpar, temos um número ímpar de circuitos de comprimento par, e portanto $\text{valor}_B(\sigma) = -1$.

($\Rightarrow$) Suponha que exista uma matriz B satisfazendo o esquema de Polyá, ou seja, que para qualquer permutação σ associada a uma cobertura de circuitos de D , temos que $\text{senal}(\sigma) = \text{valor}_B(\sigma)$.

Assim, concluímos que para a permutação identidade σ' , temos $\text{senal}(\sigma') = \text{valor}_B(\sigma') = 1$. Dessa forma, $b_{i,i} = \pm 1$, para $i = 1, 2, \dots, n$. Logo, podemos supor, sem perda de generalidade, que a matriz B possui 1 na sua diagonal.

Faça a seguinte atribuição de peso para os arcos de D :

$$p(\alpha) = \begin{cases} 0 & \text{se } b_{i,j} = -1; \\ 1 & \text{se } b_{i,j} = 1. \end{cases}$$

Seja C um circuito de D . Seja C_B uma cobertura singular tal que o circuito singular seja C . Seja σ a permutação associada a C_B . Se σ é uma permutação par (ímpar, respectivamente), pela Proposição 4.2.15, temos que o circuito C tem comprimento ímpar (par, respectivamente). Como $\text{signal}(\sigma) = \text{valor}_B(\sigma) = 1$ (-1 , respectivamente), temos que existe um número par (ímpar, respectivamente) de arcos de peso zero em C_B . Além disso, observe que todos os laços têm peso 1, uma vez que $b_{i,i} = 1$, para $i = 1, 2, \dots, n$. Dessa forma, o circuito C tem um número par (ímpar, respectivamente) de arcos de peso zero. E, como C tem comprimento ímpar (par, respectivamente), concluímos que C tem peso total ímpar.

□

Finalmente podemos apresentar o resultado principal dessa subseção, que é a prova da equivalência polinomial entre o problema do esquema de Polya e do problema do digrafo par.

Teorema 4.2.18 *O PEsqPolya é polinomialmente equivalente ao PDigrafoPar.*

Prova: Suponha que exista um algoritmo polinomial que resolve PEsqPolya(A). Seja D um digrafo com n vértices e sem laços. Seja A a sua matriz de adjacência. Pelo Teorema 4.2.17, D é um digrafo par se, e somente se, PEsqPolya($A + I$) tem uma resposta negativa, onde I é a matriz identidade de ordem n .

Por outro lado, suponha que exista um algoritmo polinomial que resolve PDigrafoPar(D). Seja A uma $\{0,1\}$ -matriz de ordem n . Seja G um grafo bipartido cuja matriz de adjacência é a matriz $\begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$, com bipartição $X = \{x_1, x_2, \dots, x_n\}$ e $Y = \{y_1, y_2, \dots, y_n\}$. Encontre, em tempo polinomial, um emparelhamento perfeito em G (se G não tem um emparelhamento perfeito, então $\text{perm}(A) = 0$ e responda trivialmente que PEsqPolya(A) tem uma resposta afirmativa). Suponha, sem perda de generalidade, que esse emparelhamento perfeito corresponde à permutação identidade, ou seja, todos os elementos da diagonal principal da matriz A são iguais a 1.

Seja D o digrafo cuja matriz de adjacência é a matriz A . Observe que em cada vértice de D incide um laço. Pelo Teorema 4.2.17, A satisfaz o esquema de Polya se, e somente se, PDigrafoPar(D) tem uma resposta negativa.

□

4.2.5 O problema do permanente–determinante

Veremos nesta seção um outro problema a respeito de permanentes e determinantes, quando restrita a $\{0,1\}$ -matrizes, é polinomialmente equivalente ao problema do circuito par. Trata-se do seguinte problema.

Problema do permanente–determinante: $\text{PPermDet}(A)$

Dado: Uma matriz quadrada A .

Problema: Decidir se $\text{perm}(A) = \det(A)$.

Os resultados mencionados a seguir foram obtidos por VAZIRANI E YANNAKAKIS [1989].

Teorema 4.2.19 *O PPermDet é $\mathcal{NP}$ -completo.*

Prova: É conhecido o fato de que o problema de decidir se o permanente de uma matriz B é igual a zero é $\mathcal{NP}$ -completo (ver VALIANT [1979]). Vamos então fazer uma redução desse problema para $\text{PPermDet}(A)$. Suponha que B seja uma matriz de ordem n . Construa a matriz A , de ordem $(n+2)$, tal que $a_{1,1} = a_{1,2} = a_{2,1} = a_{2,2} = 1$ e as demais entradas das 2 primeiras linhas e colunas sejam iguais a zero. O restante da matriz A é a matriz B . Claramente $\det(A) = 0$ e $\text{perm}(A) = 2 \times \text{perm}(B)$. Portanto, $\det(A) = \text{perm}(A)$ se, e somente se, $\text{perm}(B) = 0$.

□

Provaremos a seguir que $\text{PPermDet}(A)$, quando A é uma $\{0,1\}$ -matriz, é equivalente ao $\text{PCirPar}(D)$. O próximo teorema nos dá imediatamente uma importante relação entre esses dois problemas.

Teorema 4.2.20 *Se D é um digrafo com exatamente um laço incidente em cada um de seus vértices e A a sua matriz de adjacência, então $\det(A) = \text{perm}(A)$ se, e somente se, D não contém circuitos pares.*

Prova: Claramente $\det(A) \leq \text{perm}(A)$. É fácil ver que $\det(A) = \text{perm}(A)$ se, e somente se, toda cobertura de circuitos de D tem sinal positivo.

De fato, se D não tem circuito par, então qualquer cobertura de circuitos de D tem sinal positivo e portanto vale a igualdade. Por outro lado, se D tem um circuito par, digamos C' , construa uma cobertura de circuitos C_B formada por C' e dos laços incidentes nos vértices que não pertencem a C' . Uma vez que C_B tem sinal negativo, a igualdade não vale.

□

Teorema 4.2.21 *PPermDet(A), restrito a entradas inteiras não negativas, é polinomialmente equivalente a PCirPar. Esta afirmação continua válida mesmo quando A é uma $\{0,1\}$ -matriz.*

Prova: Seja D um digrafo sem laços. Seja A a sua matriz de adjacência. Pelo Teorema 4.2.20, D não contém circuitos pares se, e somente se, $\text{perm}(A + I) = \det(A + I)$, onde I é a matriz identidade. Dessa forma, se existe um algoritmo polinomial que resolve PPermDet(A), então é fácil resolver o problema PCirPar(D) em tempo polinomial.

Por outro lado, suponha que exista um algoritmo polinomial para resolver PCirPar(D). Seja A uma matriz quadrada inteira não negativa de ordem n . Observando que $\text{perm}(A) = \det(A)$ se, e somente se, $\text{valor}_A(\sigma) = 0$, para toda permutação ímpar σ , podemos concluir que é suficiente considerar o problema do permanente igual ao determinante para $\{0,1\}$ -matrizes. Assim, substitua todas as entradas de A que são diferentes de 0 por 1.

Primeiramente, verificaremos a existência de uma permutação π tal que $\text{valor}_A(\pi) \neq 0$, ou seja, se $a_{i,\pi(i)} = 1$, para $i = 1, 2, \dots, n$. Isto pode ser feito em tempo polinomial simplesmente verificando se o grafo bipartido G , com bipartição $X = \{x_1, x_2, \dots, x_n\}$ e $Y = \{y_1, y_2, \dots, y_n\}$, e cuja matriz de adjacência é $\begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$, tem um emparelhamento perfeito. Se G não tem um emparelhamento perfeito, então $\text{perm}(A) = 0 = \det(A)$, e portanto PPermDet(A) tem uma resposta afirmativa. Suponha então que G tenha um emparelhamento perfeito, e seja π a permutação correspondente a esse emparelhamento. Se π é uma permutação ímpar, então $\text{sinal}(\pi) \neq \text{valor}_A(\pi)$, e portanto, PPermDet(A) tem uma resposta negativa. Se π é uma permutação par, então construa uma nova matriz B permutando as colunas de A da seguinte forma: coloque a coluna $\pi(i)$ de A na coluna i da matriz B . Obtemos, dessa forma, uma nova matriz B que contém 1 na sua diagonal principal. Uma vez que π é uma permutação par, concluímos que foram feitas um número par de trocas das colunas de A para obter a matriz B , e assim, $\det(A) = \det(B)$ e $\text{perm}(A) = \text{perm}(B)$. Construa o digrafo D , cuja matriz de adjacência seja B . Pelo Teorema 4.2.20, $\text{perm}(B) = \det(B)$ se, e somente se, D não contém circuitos de comprimento par.

□

4.2.6 O problema da L -Matriz

Problemas em Álgebra Linear têm motivado o conceito de matrizes sinalizadas não singulares, ou simplesmente L -matrizes. Duas matrizes reais $m \times n$ $A = (a_{i,j})$ e $B = (b_{i,j})$, têm o mesmo padrão de sinal se

$$\begin{aligned} a_{i,j} &= 0 \text{ se, e somente se, } b_{i,j} = 0 \text{ e} \\ a_{i,j} \cdot b_{i,j} &> 0, \text{ se } a_{i,j} \neq 0. \end{aligned}$$

Um sistema linear $Ax = b$ é dito **sinal-solucionável** se as seguintes condições são satisfeitas: o sistema tem solução; e para cada matriz A' , com o mesmo padrão de sinal que A , e para cada b' , com o mesmo padrão de sinal que b , as soluções dos sistemas $Ax = b$ e $A'x = b'$ têm o mesmo padrão de sinal. Decidir se um sistema é sinal-solucionável é um problema que ocorre em certos modelos de economia (cf. KLEE, LADNER E MANBER [1984]).

Uma matriz real A $m \times n$ é uma **L -matriz** (ou **matriz sinalizada não singular**) se as colunas de qualquer matriz real A' $m \times n$, com o mesmo padrão de sinal que A , são linearmente independentes. Decidir se um sistema é sinal-solucionável é reconhecer uma L -matriz associada ao sistema (cf. [24]). KLEE, LADNER E MANBER [1984] mostraram que o problema de decidir se uma matriz real $m \times n$, é uma L -matriz é $\mathcal{NP}$ -completo, se $m = n + \lfloor n^{1/k} \rfloor$, onde k é um número natural fixo. No entanto, o caso $m = n$ ainda permanece em aberto.

É fácil ver que se em uma L -matriz real substituirmos todas as entradas negativas (respectivamente, positivas) por -1 (respectivamente, 1), a matriz resultante continua sendo uma L -matriz.

Problema da L -matriz: $\text{PLMatriz}(A)$

Dado: Uma $\{-1,0,1\}$ -matriz quadrada A .

Problema: Decidir se A é uma L -matriz.

Teorema 4.2.22 *O PLMatriz é polinomialmente equivalente ao PCirPPar .*

Prova: Suponha que exista um algoritmo polinomial que resolve PCirPPar . Seja B uma $\{-1,0,1\}$ -matriz quadrada. Uma condição necessária para que B seja uma L -matriz é que suas linhas e colunas possam ser permutadas de forma que a diagonal principal tenha todos os seus valores diferentes de zero. Essas operações preservam a propriedade de B ser uma L -matriz e fazer tais permutações é exatamente encontrar um emparelhamento perfeito em um grafo bipartido (ver demonstração do Teorema 4.2.18). Dessa forma, podemos

supor que todos os elementos da diagonal principal de B são iguais a um (uma vez que a multiplicação de uma linha por uma constante diferente de zero também resulta em uma L -matriz). Seja D um digrafo tal que $V_D = \{v_1, v_2, \dots, v_n\}$ e $A_D = \{(v_i, v_j) \mid b_{i,j} \neq 0\}$. Seja A a matriz de adjacência de D . Claramente B é uma L -matriz se, e somente se, a expressão de $\det(B)$ tem somente termos positivos; ou equivalentemente, se, e somente se, $\text{perm}(A) = \det(B)$. Logo, B é uma L -matriz se, e somente se, $\text{PEsqPolya}(A)$ tem uma resposta afirmativa (B é a matriz que satisfaz o esquema de Polya). Assim, se associarmos peso 1 (respectivamente, 0) ao arco (v_i, v_j) tal que $b_{i,j} = 1$ (respectivamente, $b_{i,j} = -1$), da demonstração do Teorema 4.2.17, segue-se que B é uma L -matriz se, e somente se, $\text{PCirPPar}(D)$ tem uma resposta negativa.

Por outro lado, suponha que exista um algoritmo polinomial que resolve PLMatriz . Seja D um digrafo simples $\{0,1\}$ -ponderado. Seja D' o digrafo obtido a partir de D subdividindo cada arco de peso 0. Pelo Teorema 4.2.1, D tem um circuito de peso par se, e somente se, D' tem um circuito par.

Seja A a matriz de adjacência de D' . Como foi mencionado anteriormente, $A + I$ é uma L -matriz se, e somente se, a expressão de $\det(A + I)$ tem somente termos positivos. Portanto, A é uma L -matriz se, e somente se, $\det(A + I) = \text{perm}(A + I)$ (observe que $A + I$ tem somente elementos positivos). Logo, pelos Teoremas 4.2.20 e 4.2.1, segue-se que $\text{PCirPPar}(D)$ tem uma resposta positiva se, e somente se, $\text{PLMatriz}(A + I)$ tem uma resposta negativa.

□

4.2.7 O problema da Orientação Pfaffiana

O problema de decidir se uma orientação de um grafo bipartido é uma orientação Pfaffiana, é polinomialmente equivalente ao problema do esquema de Polya, que por sua vez é polinomialmente equivalente ao problema do digrafo par. A importância de uma orientação Pfaffiana, de um grafo G , reside no fato de que, encontrada uma tal orientação, podemos determinar o número de emparelhamentos perfeitos de G em tempo polinomial. Esse problema tem aplicações em Física, especialmente quando o grafo é planar (cf. KASTELEYN [1967]).

Seja $C = [v_1, \alpha_1, v_2, \dots, v_k, \alpha_k, v_1]$ um semi-circuito de um digrafo. Dizemos que C é um **semi-circuito alternável**, se existe um emparelhamento perfeito tal que C é um semi-circuito alternante. Além disso, dizemos que um semi-circuito é **alternável com um emparelhamento** se esse semi-circuito é alternante para esse emparelhamento. É óbvio que um semi-circuito alternável é um semi-circuito de comprimento par. Seja A_{C+} o subconjunto dos arcos de C que estão orientados de v_i para $v_{(i+1) \bmod k}$ ($1 \leq i \leq k$). Seja $A_{C-} := A_C - A_{C+}$.

Dizemos que uma família de semi-circuitos $\mathcal{F}$ de um digrafo é **bem orientada**, se para todo semi-circuito $C \in \mathcal{F}$, temos que $|A_{C+}|$ é ímpar.

Um digrafo $\vec{G}$, obtido pela orientação de cada aresta de um grafo G , é chamado de uma **orientação** de G e tal orientação é chamada de uma **orientação Pfaffiana**, se a família dos semi-circuitos alternáveis de $\vec{G}$ é bem orientada.

Problema da orientação Pfaffiana: **POrPf**(G)

Dado: Um grafo G .

Problema: Decidir se G tem uma orientação Pfaffiana.

Veremos nesta subseção que esse problema, quando restrito a grafos bipartidos, é polinomialmente equivalente ao problema do esquema de Polya. Para isto, demonstraremos a seguir que a partir de uma orientação Pfaffiana de um grafo G , podemos obter o número de emparelhamentos perfeitos de G .

Dizemos que uma matriz quadrada B é uma matriz **anti-simétrica**, se $B^T = -B$, onde B^T é a matriz transposta de B . O **Pfaffiano** de uma matriz anti-simétrica B , de ordem $2n$, é definido como

$$Pf(B) = \sum_{\pi} \text{sign}(\pi) \cdot b_{i_1, i_2} \cdot b_{i_3, i_4} \cdot \dots \cdot b_{i_{2n-1}, i_{2n}},$$

onde $\pi = (i_1, i_2, i_3, i_4, \dots, i_{2n-1}, i_{2n})$ é uma permutação de grau $2n$ tal que $\pi(i) = j \Rightarrow \pi(j) = i$, e

$$i_1 < i_3 < \dots < i_{2n-1}$$

$$i_1 < i_2$$

$$i_3 < i_4$$

$$\vdots$$

$$i_{2n-1} < i_{2n}.$$

Podemos relacionar o Pfaffiano de uma matriz anti-simétrica B com o seu determinante através da seguinte proposição (cf. **BERGE** [1973]):

Proposição 4.2.23 *Se B é uma matriz anti-simétrica de ordem n , então*

$$\det(B) = \begin{cases} (Pf(B))^2 & \text{se } n \text{ é par;} \\ 0 & \text{se } n \text{ é ímpar.} \end{cases}$$

Seja D um digrafo com conjunto de vértices $V_D = \{v_1, v_2, \dots, v_n\}$. Podemos construir uma matriz anti-simétrica $B = (b_{i,j})$, de ordem n , cujos elementos são $-1, 0$ e 1 , da seguinte forma:

$$b_{i,j} = \begin{cases} 1 & \text{se } (v_i, v_j) \in A_D; \\ -1 & \text{se } (v_j, v_i) \in A_D; \\ 0 & \text{caso contrário.} \end{cases}$$

Chamamos essa matriz B de **matriz adjunta** de D .

A próxima proposição mostra a importância do Pfaffiano de uma matriz adjunta de um digrafo D .

Proposição 4.2.24 *Se D é um digrafo de ordem $2n$ e B a sua matriz adjunta, então $|Pf(B)|$ é menor ou igual ao número de emparelhamentos perfeitos de D . Além disso, o número de emparelhamentos perfeitos de D é igual a $|Pf(B)|$ se, e somente se, cada termo não nulo da expansão do $Pf(B)$ tem o mesmo sinal.*

Prova: Para demonstrar essa proposição, basta observar que cada termo não nulo da expansão do $Pf(B)$ corresponde a um emparelhamento perfeito de D e vice-versa. Além disso, se dois termos não nulos têm sinais opostos, então eles se cancelam.

□

KASTELEYN [1961] demonstrou o seguinte teorema, que relaciona o Pfaffiano de uma matriz adjunta de um digrafo com uma orientação Pfaffiana.

Teorema 4.2.25 *Se D é um digrafo de ordem $2n$, B a sua matriz adjunta e E_p um emparelhamento perfeito de D , então as seguintes afirmações são equivalentes:*

- (1) $|Pf(B)|$ é igual ao número de emparelhamentos perfeitos de D ;
- (2) D é uma orientação Pfaffiana;
- (3) A família dos semi-circuitos alternáveis com E_p é bem orientada.

Prova: Se E_p é o único emparelhamento perfeito de D , então o resultado vale, uma vez que D não tem semi-circuitos alternáveis. Assim, vamos assumir que D tem mais que um emparelhamento perfeito.

(1 $\Rightarrow$ 2) Seja $C = [v_1, v_2, \dots, v_{2p}, v_1]$ um semi-circuito alternável e sejam E e E' dois emparelhamentos perfeitos de D tais que:

$$E \oplus E' = A_C.$$

Suponha que o conjunto dos arcos que são comuns a E e E' seja:

$$E \cap E' = \{(v_{2p+1}, v_{2p+2}), \dots, (v_{2n-1}, v_{2n})\}.$$

Os termos correspondentes aos emparelhamentos E e E' na expansão do $Pf(B)$ são:

$$\begin{aligned} t &= \text{signal}(\sigma) \cdot b_{1,2} \cdot b_{3,4} \cdots b_{2p-1,2p} \cdot b_{2p+1,2p+2} \cdots b_{2n-1,2n} \text{ e} \\ t' &= \text{signal}(\sigma') \cdot b_{1,2p} \cdot b_{2,3} \cdots b_{2p-2,2p-1} \cdot b_{2p+1,2p+2} \cdots b_{2n-1,2n}, \end{aligned}$$

onde,

$$\begin{aligned} \sigma &= (1, 2, 3, 4, \dots, 2p-1, 2p, 2p+1, 2p+2, \dots, 2n-1, 2n) \text{ e} \\ \sigma' &= (1, 2p, 2, 3, \dots, 2p-2, 2p-1, 2p+1, 2p+2, \dots, 2n-1, 2n). \end{aligned}$$

Observe que $\text{signal}(\sigma) \cdot \text{signal}(\sigma') = 1$, pois com $2p-2$ trocas de pares de elementos, podemos passar da permutação σ para a permutação σ' .

Por hipótese, e pela Proposição 4.2.24 temos que $t \cdot t' = 1$. Logo,

$$t \cdot t' = \text{signal}(\sigma) \cdot \text{signal}(\sigma') \cdot b_{1,2} \cdot b_{3,4} \cdot b_{4,5} \cdots b_{2p-1,2p} \cdot b_{1,2p}.$$

Como $b_{1,2p} = -b_{2p,1}$, reescrevendo a igualdade acima temos que:

$$-1 = -t \cdot t' = b_{1,2} \cdot b_{3,4} \cdot b_{5,6} \cdots b_{2p-1,2p} \cdot b_{2p,1}.$$

Dessa forma, a quantidade de -1 no produto acima é ímpar e, como temos um número par de fatores, a quantidade de 1 também é ímpar. Assim, temos que $|A_{C^+}|$ e $|A_{C^-}|$ são ímpares. Portanto a família de semi-circuitos alternáveis de D é bem orientada, ou seja, D é uma orientação Pfaffiana.

(2 $\Rightarrow$ 3) Trivial, uma vez que cada semi-circuito alternável com E_p pertence à família dos semi-circuitos alternáveis e, por hipótese, D é uma orientação Pfaffiana.

(3 $\Rightarrow$ 1) Seja E um emparelhamento perfeito de D distinto de E_p . Sejam $C_1, C_2, \dots, C_k$ os semi-circuitos de $E \oplus E_p$. Note que todos esses semi-circuitos são pares e, além disso, pertencem à família dos semi-circuitos alternáveis com E_p . Seja p_i o comprimento do semi-circuito C_i . Seja $r_1 = 0$ e $r_i = \sum(p_1 + p_2 + \dots + p_{i-1})$, para $i = 2, 3, \dots, k$. Então podemos supor que

$$\begin{aligned} C_1 &= [v_{1+r_1}, v_{2+r_1}, \dots, v_{p_1+r_1}], \\ C_2 &= [v_{1+r_2}, v_{2+r_2}, \dots, v_{p_2+r_2}], \\ &\vdots \\ C_i &= [v_{1+r_i}, v_{2+r_i}, \dots, v_{p_i+r_i}], \\ &\vdots \\ C_k &= [v_{1+r_k}, v_{2+r_k}, \dots, v_{p_k+r_k}]. \end{aligned}$$

Suponha, sem perda de generalidade, que o arco (v_{1+r_i}, v_{2+r_i}) , para $i = 1, \dots, k$, pertence ao emparelhamento E_p . Vamos supor também que o conjunto dos arcos comuns aos emparelhamentos E e E_p é

$$E \cap E_p = \{(v_{p_k+r_k+1}, v_{p_k+r_k+2}), \dots, (v_{2n-1}, v_{2n})\}.$$

Para simplificar a notação, vamos considerar, para $i = 1, \dots, k$, os seguintes fatores:

$$t_{C_i}^p = b_{1+r_i, 2+r_i} \cdot b_{3+r_i, 4+r_i} \cdots b_{p_i+r_i-1, p_i+r_i},$$

$$t_{C_i} = b_{1+r_i, p_i+r_i} \cdot b_{2+r_i, 3+r_i} \cdots b_{p_i+r_i-2, p_i+r_i-1} \text{ e}$$

$$t_C = b_{p_k+r_k+1, p_k+r_k+2} \cdots b_{2n-1, 2n}.$$

Observe que $t_{C_i}^p$, t_{C_i} e t_C estão relacionados com arcos de E_p , E e $E_p \cap E$, respectivamente. A partir desses fatores, para $i = 1, 2, \dots, k$, vamos considerar as seguintes permutações:

$$\sigma_{C_i}^p = (1 + r_i, 2 + r_i, 3 + r_i, 4 + r_i, \dots, p_i + r_i - 1, p_i + r_i),$$

$$\sigma_{C_i} = (1 + r_i, p_i + r_i, 2 + r_i, 3 + r_i, \dots, p_i + r_i - 2, p_i + r_i - 1) \text{ e}$$

$$\sigma_C = (p_k + r_k + 1, p_k + r_k + 2, \dots, 2n - 1, 2n).$$

Utilizando os fatores acima, podemos escrever os termos correspondentes aos emparelhamentos E_p e E na expansão de $Pf(B)$ da seguinte forma:

$$t_p = \text{sign}(\sigma_p) \cdot t_{C_1}^p \cdot t_{C_2}^p \cdots t_{C_k}^p \cdot t_C \text{ e}$$

$$t = \text{sign}(\sigma) \cdot t_{C_1} \cdot t_{C_2} \cdots t_{C_k} \cdot t_C,$$

onde,

$$\sigma_p = (\sigma_{C_1}^p, \sigma_{C_2}^p, \dots, \sigma_{C_k}^p, \sigma_C) = (1, 2, \dots, 2n - 1, 2n) \text{ e}$$

$$\sigma = (\sigma_{C_1}, \sigma_{C_2}, \dots, \sigma_{C_k}, \sigma_C).$$

Observe que, para $i = 1, 2, \dots, k$, podemos passar da permutação σ_{C_i} para a permutação $\sigma_{C_i}^p$ com $p_i - 2$ trocas, logo podemos passar da permutação σ_p para a permutação σ com $(p_1 - 2) + (p_2 - 2) + \dots + (p_k - 2)$ trocas entre pares de elementos. Como cada p_i é par ($i = 1, 2, \dots, k$), temos que as permutações σ_p e σ têm o mesmo sinal, e portanto, $\text{signal}(\sigma_p) \cdot \text{signal}(\sigma) = 1$.

Por hipótese, $|A_{C_i^+}|$ é ímpar ($1 \leq i \leq k$). Uma vez que cada semi-circuito C_i tem comprimento par, temos que $|A_{C_i^-}|$ ($1 \leq i \leq k$) também é ímpar. Dessa forma, para $i = 1, 2, \dots, k$, o seguinte produto tem sinal negativo:

$$\Pi_i = b_{1+r_i, 2+r_i} \cdot b_{2+r_i, 3+r_i} \cdot b_{3+r_i, 4+r_i} \cdots b_{p_i+r_i-1, p_i+r_i} \cdot b_{p_i+r_i, 1+r_i}.$$

Como $b_{p_i+r_i, 1+r_i} = -b_{1+r_i, p_i+r_i}$ ($i = 1, 2, \dots, k$), temos que

$$-\Pi_i = t_{C_i}^p \cdot t_{C_i} > 0.$$

Logo,

$$t_p \cdot t = t_{C_1}^p \cdot t_{C_1} \cdots t_{C_k}^p \cdot t_{C_k} \cdot t_C \cdot t_C.$$

assim, $t_p \cdot t > 0$. Portanto os termos não nulos da expansão de $Pf(B)$ têm o mesmo sinal, e pela Proposição 4.2.24, $|Pf(B)|$ é exatamente o número de emparelhamentos perfeitos de D .

□

Como consequência do teorema acima, temos que se conseguirmos encontrar uma orientação Pfaffiana de um grafo, então utilizando a Proposição 4.2.23, poderemos determinar facilmente o número de emparelhamentos perfeitos de G . Surge então a seguinte pergunta: existe um algoritmo polinomial para encontrar uma orientação Pfaffiana? Não se conhece até o momento a resposta a essa pergunta. Além disso, a complexidade do seguinte problema está em aberto.

Problema da verificação Pfaffiana $\text{PVerPf}(\vec{G})$

Dado: Uma orientação $\vec{G}$.

Problema: Decidir se $\vec{G}$ é uma orientação Pfaffiana de G .

Observe que uma orientação $\vec{G}$ contém um número exponencial de semi-circuitos alternáveis, portanto a solução de $\text{PVerPf}(\vec{G})$ não é trivial.

VAZIRANI E YANNAKAKIS [1989] demonstraram que esse problema é polinomialmente equivalente a $\text{POrPf}(G)$. Além disso, eles mostraram que $\text{POrPf}(G)$ está em Co-NP .

KASTELEYN [1961] demonstrou que todo grafo planar tem uma orientação Pfaffiana e que podemos encontrar uma tal orientação em tempo polinomial.

Para finalizar esta subseção vamos demonstrar que $\text{POrPf}(G)$ é polinomialmente equivalente a PEsqPolya , quando G é um grafo bipartido. Consideraremos até o final desta subseção que G é um grafo bipartido e que cada conjunto da bipartição de G tem n vértices.

Lema 4.2.26 *Seja $\vec{G}$ uma orientação de um grafo bipartido G , e B a sua matriz adjunta. Se $\begin{pmatrix} 0 & C^T \\ C & 0 \end{pmatrix}$ é a matriz de adjacência de G , então $\vec{G}$ é uma orientação Pfaffiana se e somente se $\text{perm}(C) = Pf(B)$.*

Prova: Como já vimos na Subseção 4.2.4, $\text{perm}(C)$ é o número de emparelhamentos perfeitos de G . Assim pelo teorema de 4.2.25, temos que $\vec{G}$ é uma orientação Pfaffiana se, e somente se, $|Pf(B)|$ é igual ao número de emparelhamentos perfeitos de G .

□

Lema 4.2.27 *Seja G um grafo bipartido e $\begin{pmatrix} 0 & C^T \\ C & 0 \end{pmatrix}$ a sua matriz de adjacência. Então G tem uma orientação Pfaffiana se, e somente se, existe uma $\{-1, 0, 1\}$ -matriz M , tal que $\text{perm}(C) = \det(M)$.*

Prova: ($\Rightarrow$) Seja $\vec{G}$ uma orientação Pfaffiana de G e $B = \begin{pmatrix} 0 & -M \\ M & 0 \end{pmatrix}$ a sua matriz adjunta. Como $\vec{G}$ é uma orientação Pfaffiana, pelo Lema 4.2.26, temos que $\text{perm}(C) = Pf(B)$. Pela Proposição 4.2.23, temos que $(Pf(B))^2 = \det(B)$. Assim

$$\text{perm}(C) = \sqrt{\det(B)} = |\det(M)|.$$

Portanto, se $\det(M) \geq 0$, então M é a matriz procurada. Caso contrário, multiplique qualquer linha da matriz M por -1 , obtendo uma nova matriz M' .

($\Leftarrow$) Suponha que $\text{perm}(C) = \det(M) \geq 0$. Seja $B = \begin{pmatrix} 0 & M^T \\ M & 0 \end{pmatrix}$.

Pela Proposição 4.2.23, temos a igualdade:

$$(\det(M))^2 = \det(B) = (Pf(B))^2.$$

Assim,

$$|Pf(D)| = \det(M) = \text{perm}(C).$$

Como $\text{perm}(C)$ é igual ao número de emparelhamentos perfeitos de G , pelo Teorema 4.2.25, concluímos que uma orientação $\vec{G}$, cuja matriz adjunta é B , é uma orientação Pfaffiana de G .

□

Teorema 4.2.28 *Se G é um grafo bipartido, então $\text{POrPf}(G)$ é polinomialmente equivalente a $\text{PEsqPolya}(A)$.*

Prova: Suponha que exista um algoritmo polinomial que resolve $\text{POrPf}(G)$. Seja A uma $\{0, 1\}$ -matriz de ordem n . Seja G um grafo bipartido cuja matriz de adjacência é $\begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$. Pelo Lema 4.2.27, $\text{PEsqPolya}(A)$ tem uma resposta afirmativa se, e somente se, $\text{POrPf}(G)$ tem uma resposta afirmativa.

Por outro lado, suponha que exista um algoritmo polinomial que resolve PEsqPolya . Seja $\begin{pmatrix} 0 & A^T \\ A & 0 \end{pmatrix}$ a matriz de adjacência de G . Pelo Lema 4.2.27, $\text{POrPf}(G)$ tem uma resposta afirmativa se, e somente se, $\text{PEsqPolya}(A)$ tem uma resposta afirmativa.

□

4.3 Alguns resultados correlatos

4.3.1 Coloração de hipergrafos

Um **hipergrafo** é um par ordenado $H = (V_H, \mathcal{A}_H)$, onde V_H é um conjunto finito, não vazio, de elementos chamados **hiper-vértices**, ou simplesmente **vértices**, e $\mathcal{A}_H$ é uma coleção de subconjuntos, de V_H , chamados **hiper-arestas**. Um hipergrafo H é dito **quadrado** se $|V_H| = |\mathcal{A}_H|$.

Uma **r -coloração** de um hipergrafo H é uma família de conjuntos disjuntos $S_1, S_2, \dots, S_r$, tais que $S_1 \cup S_2 \cup \dots \cup S_r = V_H$ e para cada $S \in \mathcal{A}_H$, $S \not\subseteq S_i$, para $1 \leq i \leq r$. Um hipergrafo é dito **r -colorível** se admite uma r -coloração.

Um hipergrafo H é um **subhipergrafo** de um hipergrafo G se $V_H \subseteq V_G$ e $\mathcal{A}_H \subseteq \mathcal{A}_G$. Dizemos que H é um **hipergrafo condensado** se ele não é 2-colorível, mas qualquer subhipergrafo próprio de H é 2-colorível, ou seja, H é um hipergrafo 3-colorível minimal.

Figura 4.13: Um hipergrafo H e uma 3-coloração (S_1, S_2, S_3) de H .

Se $\mathcal{F} \subseteq \mathcal{A}_H$ e $B \subseteq V_H$, dizemos que B pode ser emparelhado com $\mathcal{F}$ se existe uma função bijetora $f: B \rightarrow \mathcal{F}$ tal que $b \in f(b)$ para cada $b \in B$. Uma tal função é um emparelhamento do hipergrafo H .

O seguinte problema sobre coloração de hipergrafos motivou o estudo de circuitos pares em digrafos.

Problema do hipergrafo condensado: $\text{PHiperCond}(H)$

Dado: Um hipergrafo quadrado H .

Problema: Decidir se H é um hipergrafo condensado.

A seguir apresentamos uma caracterização de hipergrafos condensados, obtida por SEYMOUR [1974], em termos de digrafos sem circuitos pares.

Teorema 4.3.1 (Seymour) *Seja H um hipergrafo quadrado condensado e $f: V_H \rightarrow \mathcal{A}_H$ um emparelhamento. Se D é um digrafo com $V_D = V_H$ e $A_D = \{(x, y) \mid x \in V_H \text{ e } y \in f(x), y \neq x\}$, então D é fortemente conexo e não contém circuitos pares. Por outro lado, se D é um digrafo fortemente conexo sem circuitos pares, então o hipergrafo H , tal que $V_H = V_D$ e $\mathcal{A}_H = \{\{x\} \cup \{y \mid (x, y) \in A_D\} \mid x \in V_D\}$, é um hipergrafo quadrado condensado.*

LOVÁSZ [1973] provou que o problema de decidir se um hipergrafo é 2-colorível é $\mathcal{NP}$ -completo. Esse resultado, junto com o Teorema 4.3.1, parece indicar que PCirPar é $\mathcal{NP}$ -completo.

Figura 4.14: Um digrafo 2-conexo sem circuitos pares.

4.3.2 Resultados referentes ao grau dos vértices do digrafo

Em 1975, LOVÁSZ, fez a seguinte pergunta:

Existe um número natural k , tal que todo digrafo, cujos vértices tenham grau de saída mínimo maior ou igual a k , tem um circuito par?

THOMASSEN [1985a] mostrou que essa pergunta tem uma resposta negativa, apresentando uma família de digrafos, que satisfaz a propriedade acima e não tem circuitos pares.

FRIEDLAND [1989] provou que todo digrafo k -regular, com $k \geq 7$, tem um circuito par. KOH [1976] mostrou uma família infinita de digrafos 2-regulares sem circuitos pares. Esses resultados levaram FRIEDLAND [1989] a conjecturar que todo digrafo k -regular, com $k \geq 3$, tem um circuito par. Recentemente, THOMASSEN [1992] mostrou que essa conjectura é verdadeira.

No entanto, esses resultados não ajudam a encontrar um algoritmo polinomial para o PCirPar, uma vez que o problema de decidir se um digrafo contém um subdigrafo k -regular, com $k \geq 2$, é $\mathcal{NP}$ -completo (cf. FRIEDLAND [1989]).

4.3.3 Resultados referentes à conectividade do digrafo

Até o presente momento, o único digrafo 2-conexo sem circuitos pares que se conhece, foi obtido por SYLVIA BOYD (cf. THOMASSEN [1986]). Esse digrafo está indicado na Figura 4.14.

Em 1986, THOMASSEN levantou a questão quanto à existência de outros digrafos 2-conexos sem circuitos pares (possivelmente de uma família infinita de tais digrafos). Essa é uma questão que ainda está em aberto.

Uma outra pergunta que foi formulada por LOVÁSZ [1975] (cf. THOMASSEN [1986]) é a seguinte:

Existe um número natural k , tal que todo digrafo k -conexo tem um circuito par?

THOMASSEN [1992] deu uma resposta afirmativa para essa pergunta e mostrou que o melhor valor de k é 3 (note que essa questão ficou em aberto durante 17 anos).

VAZIRANI E YANNAKAKIS [1989] formularam a seguinte pergunta:

Existe um número natural r tal que para todo $k \geq r$, todo digrafo k -conexo contém um subdigrafo 3-regular?

THOMASSEN [1992] mostrou que tal r não existe.

4.4 Aspectos algorítmicos

Claramente, a busca de circuitos pares em um dado digrafo pode ser reduzido à busca nos seus componentes fortemente conexos. No entanto, um digrafo fortemente conexo, sem circuitos pares, pode ter um número exponencial de circuitos ímpares. A título de ilustração, considere um digrafo D , com $V_D = \{u_1, u_2, \dots, u_k\} \cup \{v_1, v_2, \dots, v_k\} \cup \{z\}$ ($k \geq 2$) e $A_D = A_1 \cup A_2 \cup A_3 \cup A_4$, onde

$$\begin{aligned} A_1 &= \{ (v_i, z) \mid 1 \leq i \leq k \}; \\ A_2 &= \{ (z, u_i) \mid 1 \leq i \leq k \}; \\ A_3 &= \{ (u_i, v_j) \mid 1 \leq i \leq j \leq k \}; \\ A_4 &= \{ (v_i, u_j) \mid 1 \leq i < j \leq k \}. \end{aligned}$$

Esse digrafo, que é fortemente conexo, sem circuitos pares, contém $O(2^k)$ circuitos ímpares (veja a Figura 4.15). Este fato indica que para testar a existência de um circuito par a estratégia de examinar cada um dos circuitos do digrafo poderia levar a um algoritmo de desempenho pessimista exponencial.

Figura 4.15: Um exemplo de um digrafo fortemente conexo sem circuitos pares e com um número exponencial de circuitos ímpares.

KARMAKAR [1984] obteve um algoritmo exponencial para encontrar um circuito par baseado na seguinte idéia: se um vértice u está em um circuito de comprimento $2l$, então existe um vértice v tal que:

1. existe um caminho P , de u a v , de comprimento l ;
2. existe um caminho Q , de v a u , de comprimento l ;
3. P e Q são disjuntos nos vértices (com exceção, é claro, de u e v).

O algoritmo apresentado a seguir encontra, se existir, um circuito de comprimento $2l$, que passa pelo vértice v_i . Considere, neste algoritmo, que a multiplicação entre duas matrizes é booleana.

Algoritmo: Karmakar(D, v_i, l)

Entrada: Um digrafo D , um vértice $v_i \in V_D$ e um inteiro l ($2l < n$).

Saída: Um circuito de comprimento $2l$, que passa pelo vértice v_i , caso exista.

Passo 1: Seja $A = (a_{i,j})$ a matriz de adjacência do digrafo D ;

Passo 2: Obtenha a matriz $B = A^l$;

Passo 3: Se $b_{i,j} = b_{j,i} = 1$, então vá para o Passo 4. Caso contrário, pare;

Passo 4: Faça uma busca em largura, com raiz em v_i (v_j , respectivamente) e determine todos os caminhos distintos, de comprimento l , de v_i a v_j (de v_j a v_i , respectivamente). Seja $\mathcal{P}_i$ ($\mathcal{P}_j$, respectivamente) a coleção formada por esses caminhos. Se $\mathcal{P}_i$ ou $\mathcal{P}_j$ é vazio, então pare. Caso contrário, vá para o Passo 5;

Passo 5: Se existirem dois caminhos $P_i \in \mathcal{P}_i$ e $P_j \in \mathcal{P}_j$, que sejam disjuntos nos vértices, com exceção de v_i e v_j , então vá para o Passo 6. Caso contrário, pare;

Passo 6: $P_i + P_j$ é um circuito de comprimento $2l$ que passa pelo vértice v_i .

◇

A condição $b_{i,j} = b_{j,i} = 1$ é necessária, mas não suficiente para existência de um circuito de comprimento $2l$ passando pelo vértice v_i . Se um tal circuito existe, o método encontra através de uma busca exaustiva (ele determina todos os caminhos de comprimento l , entre v_i e v_j). Dessa forma, a complexidade do algoritmo $\text{Karmakar}(D, v_i, l)$, no pior caso, é $O(n^l)$.

Para decidir se um digrafo contém um circuito par, uma idéia natural seria verificar, para cada arco (ou vértice), se existe um circuito par passando por ele. Reaímos assim no seguinte problema: dados um digrafo D e um arco $\alpha \in A_D$, existe um circuito par passando por α ? Denotaremos esse problema por $\text{PCirParArco}(D, \alpha)$. O seguinte resultado, obtido por THOMASSEN [1985a], indica que uma tal idéia não vale a pena ser perseguida.

Teorema 4.4.1 *O PCirParArco é $\mathcal{NP}$ -completo.*

Prova: Claramente, o PCirParArco é um problema que está em $\mathcal{NP}$. Considere agora o seguinte problema $\mathcal{NP}$ -completo (cf. FORTUNE, HOPCROFT E WYLLIE [1980]): dados um digrafo $\tilde{D}$ e quatro vértices x_1, x_2, y_1 e y_2 , existem em $\tilde{D}$ dois caminhos disjuntos nos vértices, P_1 e P_2 , tais que P_i começa em x_i e termina em y_i ($i = 1, 2$)?

Não é difícil ver que esse problema pode ser reduzido ao PCirParArco. De fato, a partir de $\tilde{D}$, podemos construir um digrafo D subdividindo cada arco de $\tilde{D}$ e adicionado os arcos (y_2, x_1) e (y_1, x_2) . É fácil ver que D tem um circuito par passando pelo arco (y_2, x_1) se, e somente se, $\tilde{D}$ tem dois caminhos P_1 e P_2 como descritos acima.

□

Obviamente, o problema de verificar se existe um circuito ímpar passando por um arco é equivalente ao PCirParArco. KLEE, LADNER E MANBER [1984], independentemente, mostraram que decidir a existência de um circuito par (ou ímpar), passando por um vértice, também é $\mathcal{NP}$ -completo. Note que a $\mathcal{NP}$ -completude para o caso de arcos implica a $\mathcal{NP}$ -completude para o caso de vértices, e vice-versa.

4.4.1 Circuitos pares em digrafos planares

Recentemente, THOMASSEN [1993] desenvolveu um algoritmo polinomial para encontrar um circuito par em um digrafo planar. Vamos, nesta seção, descrever esse algoritmo detalhadamente. Na subseção 4.4.1.1 introduziremos algumas definições preliminares. Na subseção 4.4.1.2 apresentaremos um algoritmo polinomial $\text{Circ_Peso_Par_Planar_Con}(D)$, onde D é um digrafo $\{0,1\}$ -ponderado planar 2-conexo, que encontra um circuito de peso par, ou um arco α satisfazendo a seguinte propriedade: se D contém um circuito par, então $D - \alpha$ contém um circuito de peso par. E finalmente, na subseção 4.4.1.3, mostraremos um algoritmo, de nome $\text{CircPPar_Planar}(D)$, que resolve o $\text{PCirPPar}(D)$, em $O(n^6)$, onde D é um digrafo $\{0,1\}$ -ponderado planar. Consideraremos nesta seção que D é um digrafo simples. Note que isso não constitui perda de generalidade.

4.4.1.1 Definições preliminares

Dizemos que um circuito C de um grafo planar G é um **circuito facial** se C é o contorno de uma face. Denominaremos de **circuito externo** o circuito facial da face externa. É fácil ver que se G é 2-conexo, então o contorno de cada face é um circuito. Note que qualquer circuito facial pode ser considerado como circuito externo.

Se C é um circuito de um grafo planar, então $\text{int}(C)$ consiste de todos os vértices e arestas no interior de C . Denotaremos por $\overline{\text{int}}(C) = C \cup \text{int}(C)$. Definimos $\text{ext}(C)$ e $\overline{\text{ext}}(C)$ de maneira análoga. Se H é um subgrafo 2-conexo de um grafo planar, e S é o circuito externo de H , então vamos definir $\text{int}(H) = \text{int}(S)$. Analogamente, definimos $\overline{\text{int}}(H)$, $\text{ext}(H)$ e $\overline{\text{ext}}(H)$.

A imersão no plano de um grafo planar pode ser representada por uma ordenação de suas arestas em sentido horário em torno de cada vértice. O algoritmo linear de HOPCROFT E TARJAN [1974] para teste de planaridade de um grafo também encontra uma tal ordenação horária em tempo linear. É fácil ver que, conhecendo essa ordenação, podemos encontrar os circuitos faciais de um grafo planar em $O(n)$, simplesmente fazendo uma busca em profundidade. Além disso, se C é um circuito qualquer, podemos encontrar $\text{int}(C)$ e $\text{ext}(C)$ em tempo linear. Por exemplo, o seguinte algoritmo encontra $\overline{\text{int}}(C)$.

1. Escolha um sentido para percorrer C (horário ou anti-horário);
2. Seja P um caminho que contém exatamente um vértice de C e uma aresta do circuito externo;
3. Se a primeira aresta de P está à esquerda (direita) de C , então $\overline{\text{int}}(C)$ consiste de C e de todos os caminhos com o primeiro vértice em C e cuja primeira aresta está à direita (esquerda) de C , encontrados através de uma busca em largura especializada.

Descreveremos o algoritmo $\text{Circ_Peso_Par_Con}(D)$ em um digrafo planar D considerando que é dada essa ordenação das arestas do seu grafo subjacente. Para a análise da complexidade computacional do algoritmo, às vezes consideraremos o digrafo desenhado sobre o plano. Entretanto, isso não é assumido no algoritmo e não altera a sua complexidade.

Seja C um semi-circuito de um digrafo planar D . Quando escrevermos $\text{int}(C)$, estaremos referindo ao circuito subjacente de C . De maneira análoga, escreveremos $\overline{\text{int}}(C)$, $\text{ext}(C)$ e $\overline{\text{ext}}(C)$. Seja H um subdigrafo de D e G o grafo subjacente de H . Se G é 2-conexo, então quando escrevermos $\text{int}(H)$, estaremos nos referindo a $\text{int}(G)$. Analogamente, escreveremos $\overline{\text{int}}(H)$, $\text{ext}(H)$ e $\overline{\text{ext}}(H)$.

4.4.1.2 Restringindo o problema para digrafos planares 2-conexos

Nesta subseção apresentaremos um algoritmo de complexidade $O(n^4)$, desenvolvido por THOMASSEN [1993] que resolve o seguinte problema: dado um digrafo $\{0,1\}$ -ponderado planar 2-conexo D , encontrar em D um circuito de peso par, ou um arco α , tal que se D contém um circuito de peso par, então $D - \alpha$ contém um circuito de peso par. Vamos chamar esse algoritmo de Circ_Peso_Par_Con .

Se $\alpha = (u, v)$ é um arco de um digrafo, então um caminho P de u a v , que não passa por α , é chamado de α -desvio. Também, dizemos que o arco α contém um desvio P ou que P é um desvio de α .

Se x e y são dois vértices distintos de um digrafo D e $\beta = (x, y)$, ou $\beta = (y, x)$, então dizemos que um caminho Q , de x a y , distinto do arco (x, y) se $\beta = (x, y)$, é **minimal com respeito a β** , se Q é o único caminho, de x a y , em $\overline{\text{int}}(Q + \beta)$. Observe que não excluimos a possibilidade de existir um caminho de y a x em $\overline{\text{int}}(Q + \beta)$. Podemos verificar se um dado caminho P é minimal com respeito a um arco α em tempo linear (fazendo uma busca em profundidade em $\overline{\text{int}}(P + \alpha)$). Dessa forma, podemos encontrar um caminho minimal com respeito a α em tempo $O(n^2)$.

Um desvio B de um arco α é um **desvio estritamente minimal** em D , se em $\text{int}(B + \alpha)$ não há nenhum outro arco distinto de α que tenha um desvio em $\overline{\text{int}}(B + \alpha)$.

Lema 4.4.2 *Se C é um circuito em um digrafo planar 2-conexo D , então C é facial ou $\overline{\text{int}}(C)$ contém um desvio.*

Prova: Se C não é um circuito facial, então existe um arco (u, v) tal que $v \in V_C$ e $u \in \text{int}(C)$. Como D é 2-conexo, então $D - v$ tem um caminho, em $\overline{\text{int}}(C)$, de u a $V_C - v$. Nesse caso, o arco (u, v) contém um desvio que está em $\overline{\text{int}}(C)$ (veja Figura 4.4.2).

□

Figura 4.16: Ilustração para o Lema 4.4.2.

Lema 4.4.3 *Podemos encontrar um desvio estritamente minimal em um digrafo planar 2-conexo em $O(n^2)$.*

Prova: Seja D um digrafo planar 2 conexo. Para um arco $\alpha = (u, v)$, podemos encontrar um α -desvio B e $D_1 = \overline{int}(B + \alpha)$ em $O(n)$. No digrafo D_1 , podemos selecionar um arco α_1 e decidir em tempo $O(n)$ se D_1 contém um α_1 -desvio B_1 . Se tal desvio existir, então faça $B := B_1$ e $\alpha := \alpha_1$, e repita novamente o processo anterior. Continuando este processo, encontraremos, em $O(n^2)$, um arco α e um α -desvio B , tal que nenhum arco, com exceção de α , em $\overline{int}(B + \alpha)$ contém um desvio. Dessa forma, B é um desvio estritamente minimal.

□

O primeiro passo do algoritmo Circ_Peso_Par_Con é encontrar em $O(n^2)$ um desvio estritamente minimal $B = [u_1, u_2, \dots, u_l]$. Considere $\alpha = (u_1, u_l) = (u, v)$. Podemos supor que o circuito externo de D está em $\overline{ext}(B + \alpha)$ e que ele contém α . No que segue, nos referiremos a B e α como especificados.

Lema 4.4.4 *Para B e α como acima definidos, $int(B + \alpha)$ é vazio ou consiste de arcos da forma (u_j, u_i) , $i < j$. Além disso, $int(B + \alpha)$ não contém dois arcos (u_j, u_i) e (u_t, u_s) , tais que $s \leq i < j \leq t$.*

Prova: A última afirmação segue como consequência do fato de que (u_j, u_i) não pode ter um desvio em $\overline{int}(B + \alpha)$.

Suponha por absurdo que a primeira afirmação seja falsa. Então $int(B + \alpha)$ contém pelo menos um vértice. Como D é 2-conexo, temos que D contém um arco (x, y) , onde $x \in B$ e $y \in int(B + \alpha)$. Suponha que $x = u_i$ (veja Figura 4.17). Uma vez que $D - u_i$

Figura 4.17: Ilustração para o Lema 4.4.4

é fortemente conexo, existe um caminho Q , em $D - u_i$, de y a V_B . Note que Q está em $\overline{int}(B + \alpha)$. Seja $u_j \in B$ o último vértice de Q . Observe que $u_j \neq u_i$. Além disso, pela minimalidade de B , temos que $j < i$. Seja B' o subcaminho de B que começa em u_j e termina em u_i . Assim, pelo Lema 4.4.2, o circuito $C = Q + B' + (u_i, y)$ é facial. Como $D - u_j$ é fortemente conexo, existe um caminho P , em $D - u_j$, de $V_Q - u_j$ a V_B . Note que P está em $\overline{ext}(C)$. Seja Q' o subcaminho de Q que começa em y e termina no primeiro vértice de P e B'' o subcaminho de B do último vértice de P (digamos $u_k, k < j$) a u_j . Dessa forma, $C' = Q' + P + B'' + B' + (u_i, y)$ é um circuito em $\overline{int}(B + \alpha)$ que não é facial e pelo Lema 4.4.2 $\overline{int}(C')$ contém um desvio. Isto é um absurdo, uma vez que assumimos que B é estritamente minimal.

□

O segundo passo do algoritmo `Circ_Peso_Par_Con` é encontrar, em $O(n^2)$, um caminho P , de v a u , minimal com respeito a α . Seja $H = \overline{int}(B \cup P + \alpha)$ e S o circuito externo de H . Claramente α está em S . É possível que $S = B + \alpha$. Se $S \neq B + \alpha$, então definimos um **P -segmento** de S como um caminho, cujos vértices estão contidos em $V_S \cap V_P$. Um **P -segmento especial** de S é um P -segmento de S que tem seus extremos em B e os demais vértices em $ext(B + \alpha)$. A Figura 4.18 ilustra, em linhas fortes, um exemplo de um caminho minimal com respeito ao arco α . Nessa figura temos três P -segmentos especiais de S : um de a a b , outro de b a c , e o último de f a g .

Lema 4.4.5 Se $R = [x_1, x_2, \dots, x_r]$ é um P -segmento especial de S , tal que $x_1 = u_j$ e $x_r = u_i$, então $i < j$.

Prova: Podemos desenhar H no plano de modo que S seja o circuito externo (como foi mencionado anteriormente, isto não faz parte do algoritmo `Circ_Peso_Par_Con`). Suponha

Figura 4.18: Um α -desvio estritamente minimal B e um caminho minimal com respeito a α .

por absurdo que $j < i$. Desenhe dois arcos $\beta_1 = (u, u_j)$ e $\beta_2 = (v, u_i)$ na face externa de H . Se $R' = \beta_1 + R + \beta_2 + \alpha$, então $\text{int}(R')$ contém dois arcos (um arco do subcaminho de P de v até u_j e outro do subcaminho de P de u_i até u) que se cruzam. Mas isto é uma contradição, pois H é um digrafo planar.

□

Chamamos de **P -segmento interior** um subcaminho de P , cujos arcos estão em $\text{int}(B + \alpha)$. Sejam $P_1, P_2, \dots, P_r$, P -segmentos maximais de S , ordenados da seguinte forma: o subcaminho de B de u_1 a V_{P_i} é menor que o subcaminho de B de u_1 a V_{P_j} , se $j < i$. A maximalidade de um P -segmento implica que ele intercepta B . É fácil verificar que $P_1, P_2, \dots, P_r$ ocorrem nesta ordem sobre P e que cada P_i é a união de P -segmentos especiais de S ($1 \leq i \leq r$). Na Figura 4.18 existem dois P -segmentos maximais de S : P_1 de a a c , e P_2 de f a g . Para $i = 1, 2, \dots, r$, sejam P'_i e P''_i o primeiro e o último destes P -segmentos especiais de P_i , respectivamente. Na Figura 4.18, P'_1 vai de a a b , enquanto que P''_1 vai de b a c e $P'_2 = P''_2 = P_2$. Sejam C'_i e C''_i os circuitos obtidos pela união de P'_i com um subcaminho de B e P''_i com outro subcaminho de B , respectivamente. O subcaminho de P do último vértice de P_i até o primeiro vértice de P_{i+1} contém um subcaminho Q_i , que é um P -segmento interior, tal que o primeiro vértice está em C''_i e o último vértice está em C'_{i+1} e todos os outros vértices estão em B , mas não em $C''_i \cup C'_{i+1}$. Note que os vértices do subcaminho de P do último vértice de P_i até o primeiro vértice de Q_i estão em $\overline{\text{int}}(C''_i)$.

Figura 4.19: Construção do caminho P' .

O *terceiro passo* do algoritmo *Circ_Peso_Par_Con* é verificar se o circuito $P + \alpha$ tem peso par. Nesse caso, o algoritmo pára. Suponha, então, que $P + \alpha$ tenha peso ímpar.

O *quarto passo* do algoritmo é verificar, para $i = 1, 2, \dots, r - 1$, a paridade do peso dos circuitos obtidos a partir de $P + \alpha$ pela substituição do subcaminho de P , do último vértice de P_i até o primeiro vértice de Q_i , por um subcaminho de B . Assuma que todos os circuitos tenham peso ímpar, pois caso contrário o algoritmo pára.

O *quinto passo* do algoritmo é verificar, para $i = 1, 2, \dots, r - 1$, a paridade do peso dos circuitos obtidos a partir de $P + \alpha$ pela substituição do subcaminho de P , do último vértice de Q_i até o primeiro vértice de P_{i+1} , por um subcaminho de B . Assuma que todos os circuitos tenham peso ímpar, pois caso contrário o algoritmo pára.

Seja P' o caminho, obtido a partir de P , substituindo o subcaminho de P do último vértice de P_i até o primeiro vértice de Q_i , e também do último vértice de Q_i até o primeiro vértice de P'_{i+1} , por um subcaminho de B , para $i = 1, 2, \dots, r - 1$. Então, $P' + \alpha$ tem peso ímpar. As modificações para construir P' não resultam em vértices repetidos. De fato, o primeiro vértice de Q_i está em C''_i e os outros vértices, a partir deste, estão em $\text{ext}(C''_i)$, enquanto que o subcaminho de P , do último vértice de P_i ao primeiro vértice de Q_i consiste de vértices e arcos em $\overline{\text{int}}(C''_i)$ e possivelmente de arcos em $\text{int}(B + \alpha)$ ligando dois vértices de C''_i . Na Figura 4.19, temos uma ilustração da construção do caminho P' em linhas fortes, no caso em que P é o caminho indicado na Figura 4.18.

Sejam x e y dois vértices de P' , tal que x precede y . Um caminho L em $\text{ext}(P' + \alpha)$, de x a y , é um **caminho de substituição**, se nenhum vértice de $V_L - \{x, y\}$ pertence

Figura 4.20: Um caminho de substituição $L[x, y]$.

a P' . Dizemos que L é um **caminho de substituição minimal**, que denotamos por $L[x, y]$, se não existe outro caminho de substituição de x a y em $\overline{int}(P' \cup L + \alpha)$.

Para cada par de tais vértices x e y de P' , Circ_Peso_Par_Con decide em $O(n)$, se existe um caminho de substituição de x a y . Se tal caminho existir, então o algoritmo encontra um minimal de x a y ($L[x, y]$) em $O(n^2)$. Dessa forma, Circ_Peso_Par_Con encontra todos os caminhos de substituição minimais em $O(n^4)$.

Fixe um caminho de substituição minimal $L[x, y]$. Seja S' o circuito externo de $S \cup L[x, y]$. S' é obtido a partir de S , simplesmente, substituindo alguns subcaminhos de S por subcaminhos de $L[x, y]$. Seja L' um subcaminho comum a S' e $L[x, y]$ e x' e y' o primeiro e o último vértice de L' , respectivamente. Na Figura 4.20 ilustra como poderia ser $L[x, y]$ e L' , no caso em que o caminho P' é o indicado na Figura 4.19: $L[x, y]$ é o caminho, de x a y , indicado pelas linhas fortes e L' é o caminho, de x a y' , também indicado pelas linhas fortes.

Afirmamos que não existe outro subcaminho comum a S' e $L[x, y]$. De fato, suponha por absurdo que L'' seja outro subcaminho comum a S' e $L[x, y]$. Suponha que L'' suceda L' em $L[x, y]$. Se W é o subcaminho de $L[x, y]$, de y' a $V_{L''}$, ele deve usar um arco $\alpha' \in \overline{int}(B + \alpha)$. Pelo Lema 4.4.4, temos que $\alpha' \in P'$. Mas isso é um absurdo, pois apenas

Figura 4.21: Um caminho crucial.

dois vértices de P' estão em $L[x, y]$ (que são x e y). Dessa forma, S' é obtido, a partir de S , pela substituição de um subcaminho de S por L' .

Os vértices em $L' \cap S$ pertencem a B (com possível exceção de x' e y' , que podem pertencer a P -segmentos especiais). Assim, $L' \cap S$ particiona L' em subcaminhos $L_1, L_2, \dots, L_s$, e para $i = 1, 2, \dots, s-1$, L_i e L_{i+1} têm um vértice em comum.

O próximo passo do algoritmo é testar se o peso do circuito, obtido a partir de $P' + \alpha$, pela substituição do subcaminho de P , de x a y , pelo caminho $L[x, y]$ é par. Suponha que esse circuito tenha peso ímpar, pois caso contrário o algoritmo pára.

O passo seguinte do algoritmo `Circ_Peso_Par_Con` é testar se o peso dos circuitos obtido pela união de L_i ($1 \leq i \leq s$) (e possivelmente um ou dois subcaminhos de P) com um subcaminho de B é par. Assumiremos que todos esses circuitos tenham peso ímpar. Na Figura 4.20 existem dois desses circuitos: $[b, \dots, x, \dots, h, \dots, g, \dots, e, \dots, b]$ e $[h, \dots, y', \dots, y, \dots, h]$.

Dizemos que um caminho M em $\overline{ext}(S')$ é um **caminho crucial**, se o primeiro vértice w e o último vértice z estão em L' e nenhum outro vértice está em S' . Além disso, M satisfaz a seguinte propriedade: se $w \in V_{L_i} - V_{L_{i+1}}$ e $z \in V_{L_j} - V_{L_{j-1}}$, então $j > i$ e $j - i$

é ímpar. Um exemplo de um caminho crucial está indicado, em linhas fortes, na Figura 4.21.

O algoritmo *Circ_Peso_Par_Con* decide, em $O(n^2)$, se existe um caminho crucial M . Como existem $O(n^2)$ pares w e z , então o algoritmo encontra um caminho crucial em $O(n^4)$. Um caminho crucial é importante devido ao seguinte fato.

Lema 4.4.6 *Se D tem um caminho crucial M , então D tem um circuito de peso par.*

Prova: Existem 4 casos a considerar:

1. $x' \in B$ e $y' \in B$;
2. $x' \in B$ e $y' \in S - V_B$;
3. $x' \in S - V_B$ e $y' \in B$;
4. $x' \in S - V_B$ e $y' \in S - V_B$.

Vamos considerar o caso 3, ou seja, que $x' \in S - V_B$ e $y' \in B$ (os outros casos são similares). Dessa forma, x' pertence a um P -segmento T . Observe que $x' = x$. Para cada k , $i \leq k \leq j$, seja S_k o circuito que é obtido pela união de L_k com um segmento de B . Note que se $i = 1$, então S_i contém um subcaminho de T (na Figura 4.21, o circuito S_1 é $[b, \dots, x, \dots, w, \dots, h, \dots, g, \dots, e, \dots, b]$ e S_2 é $[h, \dots, z, \dots, y', \dots, y, \dots, h]$).

Seja R_1 o circuito em $M \cup S_i \cup S_{i+1} \cup \dots \cup S_j$ contendo M (na Figura 4.21, o caminho R_1 é $[b, \dots, x, \dots, w, \dots, z, \dots, y', \dots, y, \dots, h, \dots, b]$).

Seja R_2 o circuito obtido a partir de $P' + \alpha$, pela substituição de um segmento de P' pelo caminho de substituição $L[x, y]$. Seja R_3 o circuito obtido a partir de R_2 , pela substituição do subcaminho de L' de w a z pelo caminho crucial M . Então $R_1, R_2, R_3, S_i, S_{i+1}, \dots, S_j$ é uma coleção de $j - i + 4$ circuitos, onde cada arco é coberto por um número par de circuitos. Portanto a soma dos pesos de todos os circuitos é par. Uma vez que $j - i + 4$ é ímpar, pelo menos um dos circuitos tem peso par.

□

Dessa forma, se o algoritmo tiver encontrado um caminho crucial, então o *último passo* é encontrar os circuitos $R_1, R_2, R_3, S_i, S_{i+1}, \dots, S_j$, e dentre estes, determinar um de peso par. Isso pode ser feito em $O(n)$. Se não existe nenhum caminho crucial, então o algoritmo pára. Portanto, o algoritmo *Circ_Peso_Par_Con* tem complexidade $O(n^4)$. Vamos considerar o que acontece se nenhum caminho crucial foi encontrado e todos os circuitos testados pelo algoritmo têm peso ímpar.

Lema 4.4.7 *Suponha que o algoritmo não encontrou um caminho crucial, e todos os circuitos encontrados por ele tem peso ímpar. Se D tem um circuito de peso par, então $D - \alpha$ tem um circuito de peso par.*

Prova: Dentre todos os circuitos de peso par em D , escolha um circuito C que tenha o menor número possível de arcos que não estão em B . Vamos supor que C contém α , pois ~~caso contrário C é um circuito de peso par em $D - \alpha$. Como o caminho P é minimal com respeito a α , então o circuito C deve estar em $\overline{ext}(P + \alpha)$.~~

Primeiramente, suponha que C não está em $\overline{ext}(P' + \alpha)$. Então, pela definição de P' , todo arco de C que está em $int(P' + \alpha)$ deve estar em $\overline{int}(C'_{i+1})$ (ou $\overline{int}(C''_i)$), ou ele é um arco em $int(B + \alpha)$, cujas pontas estão em $B \cap C'_{i+1}$ (ou $B \cap C''_i$). Vamos considerar o circuito C''_i , pois o outro caso (circuito C'_{i+1}) é análogo.

Seja u_c o último vértice de P''_i e u_d o primeiro vértice de Q_i . Então, P' contém um subcaminho $W' = [u_c, u_{c+1}, \dots, u_{d-1}, u_d, u_e]$, onde (u_d, u_e) , $e < d$, é o primeiro arco de Q_i . Note que o circuito C contém um subcaminho, digamos W , de u_c a u_e .

No subcaminho de C , de v a u , o vértice u_c precede qualquer arco em $int(C''_i)$ (devido à minimalidade de P e à minimalidade de B). O subcaminho de C , a partir do vértice u_c , contém vértices de $int(C''_i)$. O subcaminho de C , a partir de um último vértice de C''_i , usa um arco de $int(B + \alpha)$. Pelo Lema 4.4.4, esse arco é (u_d, u_e) . Dessa forma, se $Z = [u_e, u_{e+1}, \dots, u_c]$ é um subcaminho de B , de u_e a u_c , então $W + Z$ e $W' + Z$ são dois circuitos. Se os seus pesos têm paridades diferentes, então encontramos um circuito de peso par que não contém α . Por outro lado, se os seus pesos têm a mesma paridade, então substitua, no circuito C , o subcaminho W pelo subcaminho W' e obteremos um outro circuito que contradiz a minimalidade de C . Portanto, podemos supor que C está em $\overline{ext}(P' + \alpha)$.

Nesse caso, $V_C \cap V_{P'}$ particiona C em subcaminhos. Uma vez que o peso de $P' + \alpha$ é ímpar e o peso de C é par, um destes subcaminhos é um caminho de substituição T , digamos de x a y , cujo peso tem paridade diferente do subcaminho de P' de x a y . Considere, então o caminho minimal de substituição $L[x, y]$ e o seu subcaminho $L' = L_1 \cup L_2 \cup \dots \cup L_k$, de x' a y' , definido anteriormente. Se substituirmos o subcaminho de P' , de x a y , por $L[x, y]$, então por hipótese, obteremos um circuito de peso ímpar. Assim os pesos de $L[x, y]$ e T têm paridades diferentes. Portanto T tem um subcaminho T' , de $t_1 \in L[x, y]$ a $t_2 \in L[x, y]$, tal que os pesos de T' e do subcaminho de $L[x, y]$, de t_1 a t_2 , tenham paridades diferentes e T' tem somente os vértices t_1 e t_2 em comum com $L[x, y]$.

Vamos considerar o caso que t_1 e t_2 não estão em L' . Podemos supor que, no subcaminho de $L[x, y]$, o vértice t_1 , ou t_2 , sucede y' (outro caso é análogo). Dessa forma, pelo fato do digrafo ser planar e pela minimalidade de B , no subcaminho de $L[x, y]$, ambos os vértices t_1 e t_2 sucedem y' . Esse subcaminho de $L[x, y]$ tem uma estrutura complicada, no

entanto podemos afirmar que todos os seus vértices e arcos estão em $\overline{\text{int}}(C')$, onde C' é o circuito obtido pela união de L_k e um subcaminho de B . Precisamos mostrar que existe um circuito de peso par obtido a partir do subcaminho de $L[x, y]$, de t_1 a t_2 , em $D - \alpha$.

Seja N o caminho obtido pela concatenação do subcaminho de $L[x, y]$, de t_2 a y , e do subcaminho de P' de y a t_3 , onde t_3 é um primeiro vértice de B que precede y' . Seja N' o caminho obtido pela concatenação do subcaminho de B , de t_3 a y' , e do subcaminho de $L[x, y]$, de y' a t_1 . Observe que $N + N'$ é um caminho de t_2 a t_1 . É fácil ver que $N + N'$ concatenado com o subcaminho de $L[x, y]$, de t_1 a t_2 , e $N + N' + T'$ são dois circuitos. Esses dois circuitos têm pesos de paridades diferentes e eles estão em $D - \alpha$.

Por outro lado, suponha que $t_1 \in V_{L_i} - V_{L_{i+1}}$ e $t_2 \in V_{L_j} - V_{L_{j-1}}$, onde $j \geq i$. Uma vez que D não tem caminhos cruciais, então $j - i$ é par. Analogamente à demonstração do Lema 4.4.6, vamos definir os circuitos $R_1, R_2, R_3, S_i, S_{i+1}, \dots, S_j$ (exceto que $j - i$ é par e considerando T' ao invés de M). Como vimos, cada arco em $R_1 \cup R_2 \cup R_3 \cup S_i \cup S_{i+1} \cup \dots \cup S_j$ é coberto por um número par destes $j - i + 4$ circuitos. Além disso, os pesos dos circuitos R_2 e R_3 , que contêm o arco α , têm paridades diferentes. Portanto um dos circuitos $R_1, S_i, S_{i+1}, \dots, S_j$ deve ter peso par.

□

Teorema 4.4.8 *Se D é um digrafo $\{0,1\}$ -ponderado planar 2-conexo, então o algoritmo $\text{Circ_Peso_Par_Con}(D)$ encontra, em $O(n^4)$, um circuito de peso par, ou um arco α tal que, se D tem um circuito de peso par, então $D - \alpha$ tem um circuito de peso par.*

4.4.1.3 Um algoritmo para digrafos planares

THOMASSEN [1986] demonstrou que podemos reduzir o problema do circuito de peso par (PCirPPar) para o mesmo problema com entrada restrita a digrafos 2-conexos. Vamos nesta subseção apresentar tal redução e posteriormente descrever um algoritmo (que chamaremos de PCirPPar_Planar) para encontrar um circuito de peso par em um digrafo planar.

Seja D um digrafo ponderado fortemente conexo, com $n \geq 3$, contendo um vértice z tal que $D' = D - z$ não é fortemente conexo. Dessa forma, $V_{D'}$ tem uma partição em 2 conjuntos, digamos A_1 e A_2 , tais que não existem arcos de A_2 para A_1 . Considere os digrafos $D_1 = D[A_1 + z]$ e $D_2 = D[A_2 + z]$. Para cada arco (x, y) , de A_1 a A_2 , seja $C_{x,y}$ um circuito de D que contém (x, y) . Se $C_{x,y}$ tem peso ímpar, então adicione o arco (x, z) em D_1 e o arco (z, y) em D_2 e atribua pesos a esses arcos de modo que $C_{x,y} \cup \{(x, z), (z, y)\}$ não tenha circuito de peso par. Vamos denominar esta redução de **redução dois componentes**. É fácil ver que esta redução tem complexidade $O(n^2)$.

4.4. ASPECTOS ALGORÍTMICOS

Lema 4.4.9 *D tem um circuito de peso par se, e somente se, um dos circuitos $C_{x,y}$ tem peso par ou algum dos digrafos D_1 ou D_2 tem um circuito de peso par.*

Prova: Se algum circuito $C_{x,y}$ tem peso par, então não há o que provar. Dessa forma, suponha que cada circuito $C_{x,y}$ tenha peso ímpar. Se um dos digrafos D_1 ou D_2 tem um circuito de peso par, então é fácil encontrar um circuito de peso par em D , uma vez que a atribuição de pesos foram feitas somente nos arcos novos de modo conveniente. Por outro lado, suponha que D contenha um circuito C de peso par. Suponha que C não está em D_1 e nem em D_2 . Então C contém um arco (x, y) de A_1 para A_2 . Uma vez que $C_{x,y}$ tem peso ímpar, então podemos assumir que os pesos dos subcaminhos de $C_{x,y}$ e C , de z a x , têm a mesma paridade e que os pesos dos subcaminhos de $C_{x,y}$ e C , de y a z , são de paridades diferentes. Dessa forma, o circuito em D_2 consistindo do arco (z, y) e do subcaminho de C , de y a z , tem peso par.

□

O Lema 4.4.9 garante que podemos reduzir o problema de encontrar um circuito de peso par em um digrafo D para o mesmo problema com entrada restrita a digrafos 2-conexos. Note ainda que esse lema vale para digrafos não planares.

Considere uma outra redução descrita a seguir: seja D um digrafo $\{0,1\}$ -ponderado fortemente conexo, com pelo menos três vértices. Se D contém um arco $\alpha = (x, y)$ tal que $g^+(x) = 1$ ou $g^-(y) = 1$, então contraia o arco α . Se o peso de α é um e $g^+(x) = 1$ (respectivamente, $g^-(y) = 1$), então devemos mudar os pesos dos arcos que originalmente entravam em x (respectivamente, saíam de y). É fácil ver que o digrafo resultante tem um circuito de peso par se, e somente se, D tem um circuito de peso par. Vamos chamar essa redução de **redução aumento de grau**. Também não é difícil ver que esta redução tem complexidade $O(n)$.

Teorema 4.4.10 *Existe um algoritmo de complexidade $O(n^6)$ que resolve $\text{PCirPPar}(D)$, onde D é um digrafo $\{0,1\}$ -ponderado planar.*

Prova: Para provar este teorema, apresentaremos um algoritmo, de nome $\text{PCirPPar_Planar}(D)$, onde D é um digrafo $\{0,1\}$ -ponderado planar, que encontra um circuito de peso par, se existir. Vamos considerar que um digrafo com exatamente um vértice é fortemente conexo. Assim, um digrafo com exatamente dois vértices será considerado 2-conexo.

Primeiramente o algoritmo PCirPPar_Planar encontra os componentes fortemente conexos de D . Sejam $D_1, D_2, \dots, D_k$ os componentes fortemente conexos, que têm pelo menos 2 vértices. Para $i = 1, 2, \dots, k$, seja $n_i = |V_{D_i}|$ e $n = n_1 + \dots + n_k$.

O algoritmo aplica sucessivamente as reduções **aumento de grau** e **dois componentes** até que todos os componentes sejam 2-conexos. Se um deles, digamos D_1 , tem

ϕ_1

ϕ_2

então ele aplica o algoritmo `Circ_Peso_Par_Con`. A partir daí, ele aplica `_Peso_Par_Con` até encontrar um circuito de peso par ou até que D_1 seja um digrafo que não seja 2-conexo. Se este é o caso, então ele aplica `_Con` a outro componente com pelo menos 3 vértices, e assim por diante.

O número de vezes que o algoritmo aplica `Circ_Peso_Par_Con` é menor que $6n - 12$, uma vez que D tem no máximo $6n - 12$ arcos. Dessa forma, o tempo entre duas aplicações de aumento de grau, ou dois componentes, é $O(n^5)$.

Se aplicarmos a redução aumento de grau em um digrafo, digamos D_1 , o novo digrafo terá um vértice a menos. Se aplicarmos a redução dois componentes em um digrafo, digamos D_1 , teremos dois novos digrafos D'_1 e D''_1 , tais que $n_1 = n'_1 + n''_1 - 1$. Dessa forma, quando a redução aumento de grau, ou dois componentes, é aplicada, o número $n - 2k$ (que é não-negativo, uma vez que todos os digrafos $D_1, D_2, \dots, D_k$ têm pelo menos dois vértices) decresce de uma unidade. Assim, essas reduções são aplicadas no máximo $n - 2$ vezes. Segue-se, portanto, que a complexidade de `PCircPPar_Planar` é $O(n^6)$.

□

Capítulo 5

Caminhos pares e caminhos ímpares

5.1 Introdução

Nesse capítulo estudaremos os seguintes problemas: $\text{PCamPar}(G, s, t)$, para grafos e digrafos; $\text{PEncCamPar}(G, s, t)$ e $\text{PCamParMin}(G, s, t)$ para grafos. Lembramos que como os problemas sobre caminhos pares são computacionalmente equivalentes aos seus correspondentes problemas sobre caminhos ímpares (eles estão relacionados por uma simples redução que consiste em adicionar um novo vértice adjacente a s), é suficiente estudar somente o caso par.

Os correspondentes problemas sobre caminhos sem restrição de paridade são todos fáceis (tanto para grafos como digrafos) e foram amplamente estudados. O leitor interessado pode consultar os textos MCHUGH [1990]; BONDY E MURTY [1976]; e AHO, HOPCROFT E ULLMAN [1974].

Na Seção 5.2 estudaremos o problema PCamPar para grafos; em seguida, na Seção 5.3 estudaremos esse mesmo problema para digrafos. Veremos que existem algoritmos polinomiais para o primeiro caso e que, surpreendentemente, no segundo caso o problema é $\mathcal{NP}$ -completo. Nas Seções 5.4 e 5.5, estudaremos os problemas PEncCamPar e PCamParMin para grafos. Na última seção apresentaremos um resumo dos resultados das seções anteriores.

5.2 Existência de caminhos pares em grafos

Nesta seção apresentaremos dois algoritmos polinomiais que resolvem o problema da existência de caminhos pares em grafos. O primeiro algoritmo, de complexidade $O(\sqrt{nm})$, utiliza uma construção devido a EDMONDS, que reduz o PCamPar para o problema de

decidir se um grafo tem um emparelhamento perfeito (veja LAPAUGH E PAPADIMITRIOU [1984], THOMASSEN [1985a]). Veremos que é possível melhorar a complexidade, de $O(\sqrt{nm})$ para $O(m)$, fazendo uma redução (através de uma escolha apropriada de um emparelhamento inicial) para o chamado problema da fase completa (veja Subseção 5.2.2). Outro algoritmo, de complexidade $O(m)$, obtido por LAPAUGH E PAPADIMITRIOU [1984], é baseado na decomposição do grafo em componentes biconexos.

5.2.1 Preliminares

Faremos nesta subseção alguns comentários sobre o problema do emparelhamento máximo, em preparação ao resultados que serão apresentados nas próximas subseções.

Problema do emparelhamento máximo: PEmpMax(G)

Dado: Um grafo G .

Problema: Encontrar um emparelhamento máximo em G .

BERGE, em 1957, obteve um resultado importante que caracteriza quando um emparelhamento em um grafo é máximo.

Teorema 5.2.1 (Berge) *Um emparelhamento em um grafo G é máximo se, e somente se, G não contém caminhos de aumento.*

EDMONDS [1965], utilizando o resultado de BERGE, apresentou um algoritmo de complexidade $O(n^4)$ que resolve PEmpMax(G). A partir de então muitos outros pesquisadores estudaram esse problema, e progressivamente algoritmos mais eficientes foram desenvolvidos. Na tabela abaixo indicamos a complexidade de alguns desses algoritmos.

Algoritmo de complexidade	Obtido por
$O(n^3)$	GABOW [1976]
$O(n^3)$	PAPE E CONRADT [1980]
$O(nm)$	KAMEDA E MUNRO [1974]
$O(n^{2.5})$	EVEN E KARIV [1975]
$O(\sqrt{n} \min\{m \log n, m + n^{3/2}\})$	BLUM [1993]
$O(\sqrt{nm})$	MICALI E VAZIRANI [1980]

GABOW [1976] refinou o algoritmo de EDMONDS [1965] e obteve um algoritmo de complexidade $O(n^3)$. Embora o algoritmo de PAPE E CONRADT [1980] tenha complexidade igual a do algoritmo de GABOW [1976], estudos empíricos mostraram que na prática

o primeiro algoritmo é mais rápido (cf. SYSLO, DEO E KOWLIK [1983]). KAMEDA E MUNRO [1974] desenvolveram um algoritmo melhor para grafos esparsos.

O algoritmo desenvolvido por EVEN E KARIV é relativamente complicado e utiliza uma estrutura de dados bem complexa. Estudos empíricos mostraram que esse algoritmo não é muito bom do ponto de vista prático e tem somente interesse teórico (cf. SYSLO, DEO E KOWALIK [1983]).

Recentemente, BLUM [1993] fez um novo tratamento para encontrar um emparelhamento máximo e obteve um algoritmo de complexidade $O(\sqrt{n} \cdot \min\{m \log n, m + n^{3/2}\})$. Teoricamente o algoritmo mais rápido para resolver PEMPMax foi desenvolvido por MICALI E VAZIRANI [1980].

Vale aqui observar que para o caso de grafos bipartidos, HOPCROFT E KARP [1973] obtiveram um algoritmo de complexidade $O(\sqrt{nm})$ e ALT, BLUM, MEHLHORN E PAUL [1991] desenvolveram um outro algoritmo de complexidade $O(n^{1.5} \sqrt{m/\log n})$.

O Teorema de BERGE [1957] conduz a uma estratégia (apresentada a seguir) utilizada na maioria dos algoritmos para encontrar um emparelhamento máximo em um grafo G .

Passo 1: Seja E um emparelhamento em G ;

Passo 2: Procure um caminho de aumento P em relação a E .

Se um tal caminho não existir, então pare (E é um emparelhamento máximo).
Caso contrário, construa outro emparelhamento $E' := E \oplus P$ (claramente E' tem uma aresta a mais que E);

Passo 3: $E := E'$;

Passo 4: Volte para o Passo 2;

Vamos chamar o Passo 2 de **passo de aumento**. Por exemplo, um passo de aumento do algoritmo de PAPE E CONRADT [1980] tem complexidade $O(n^2)$.

HOPCROFT E KARP [1973] desenvolveram um algoritmo bastante engenhoso para determinar um emparelhamento máximo em um grafo. Tal algoritmo tem como passo de aumento a busca de um conjunto maximal de menores caminhos de aumento (de mesmo comprimento), totalmente disjuntos nos vértices, em relação a um emparelhamento E , que chamaremos de **uma fase completa**. Esses autores mostraram que, para encontrar um emparelhamento máximo E_G em G , é suficiente executar no máximo $2 \times \lfloor \sqrt{|E_G|} \rfloor$ fases completas. Descrevemos a seguir o algoritmo devido a HOPCROFT E KARP.

Algoritmo Encontra_Emparelhamento_Máximo(G)

Entrada: Um grafo G .

Saída: Um emparelhamento máximo em G .

Passo 1: $E_G := \emptyset$;

Passo 2: Seja $l(E_G)$ o comprimento de um menor caminho de aumento em G (relativo a E_G). Encontre um conjunto maximal $\{P_1, P_2, \dots, P_k\}$ de caminhos de aumento, totalmente disjuntos nos vértices, tal que $|P_i| = l(E_G)$, para $i = 1, 2, \dots, k$.

Se tal conjunto não existe, então pare.

Passo 3: $E_G := E_G \oplus A_{P_1} \oplus \dots \oplus A_{P_k}$;

Volte para o Passo 2.

◇

Teorema 5.2.2 (Hopcroft e Karp) *Se em um grafo G a cardinalidade de um emparelhamento máximo é s , então o algoritmo Encontra_Emparelhamento_Máximo(G) constrói um emparelhamento máximo em $2\lfloor\sqrt{s}\rfloor+2$ execuções do Passo 2.*

HOPCROFT E KARP [1973] mostraram que se G é um grafo bipartido, então uma fase completa pode ser feita em $O(m)$, e portanto, neste caso, o algoritmo Encontra_Emparelhamento_Máximo, para grafos bipartidos, tem complexidade $O(\sqrt{nm})$. Poderíamos, então, perguntar se a execução de uma fase completa continua sendo linear mesmo no caso de grafos não bipartidos. Para responder essa pergunta, vamos enunciar formalmente o problema:

Problema da fase completa: **PFaseCompleta**(G, E_G)

Dados: Um grafo G e um emparelhamento E_G .

Problema: Encontrar um conjunto maximal $\{P_1, \dots, P_k\}$ de caminhos de aumento totalmente disjuntos nos vértices, tal que $|P_i| = l(E_G)$, para $i = 1, \dots, k$; sendo $l(E_G)$ o comprimento de um menor caminho de aumento em G relativo a E_G .

Figura 5.1: Um exemplo da redução de Edmonds.

MICALI E VAZIRANI [1980] apresentaram um algoritmo (extremamente complicado) que resolve $\text{PFaseCompleta}(G, E_G)$ em tempo $O(m)$. Infelizmente, no artigo que contém tal resultado, os autores não apresentaram uma prova satisfatória de que o algoritmo resolve corretamente o problema da fase completa. PETERSON E LOUI [1988] expuseram o algoritmo de MICALI E VAZIRANI [1980] de uma forma mais clara e apresentaram uma prova informal. Em 1989, VAZIRANI apresentou em um relatório técnico uma prova para uma versão modificada do algoritmo.

Assumiremos aqui que o algoritmo obtido por MICALI E VAZIRANI [1980] está correto e que o problema $\text{PFaseCompleta}(G, E_G)$ pode ser resolvido em $O(m)$. Portanto, o algoritmo $\text{EncontraEmparelhamentoMáximo}(G)$ resolve $\text{PEmpMax}(G)$ em $O(\sqrt{nm})$.

Nas próximas subseções estudaremos os problemas referentes a caminhos pares utilizando alguns resultados dessa subseção.

5.2.2 Algoritmo para PCamPar via construção de Edmonds

Descrevemos aqui a redução, devido a EDMONDS, do $\text{PCamPar}(G, s, t)$ para o problema do emparelhamento máximo.

Seja H_1 uma cópia de $G - t$ e H_2 uma cópia de $G - s$ disjunta de H_1 . Para cada vértice $v \in V_G - \{s, t\}$ chame de $\bar{v}$ e $\tilde{v}$ as cópias de v em H_1 e H_2 , respectivamente. Considere $\bar{v}$ e $\tilde{v}$ **vértices correspondentes**. Ligue por arestas os vértices correspondentes. Denote esse novo grafo por $\bar{G}$ (veja um exemplo na Figura 5.1).

O objetivo desta subseção é demonstrar a equivalência entre a existência de um caminho par de s a t em G e a existência de um emparelhamento perfeito em $\overline{G}$. Para isso, seja $E_{\overline{G}}$ um emparelhamento em $\overline{G}$ que consiste das arestas que têm como extremos os vértices correspondentes. Em relação a $E_{\overline{G}}$, note que os únicos vértices livres em $\overline{G}$ são s e t .

Teorema 5.2.3 *Existe um caminho par de s a t em G se, e somente se, $\overline{G}$ tem um caminho de aumento (relativo a $E_{\overline{G}}$).*

Prova: ($\Rightarrow$) Seja $P = [s, v_1, v_2, v_3, \dots, v_k, t]$ um caminho par de s a t em G . É fácil ver que o caminho $[s, \bar{v}_1, \bar{v}_1, \bar{v}_2, \bar{v}_2, \bar{v}_3, \bar{v}_3, \dots, \bar{v}_k, \bar{v}_k, t]$ é um caminho de aumento relativo a $E_{\overline{G}}$.

($\Leftarrow$) Seja $\overline{P} = [s, \bar{v}_1, \bar{v}_1, \bar{v}_2, \bar{v}_2, \bar{v}_3, \bar{v}_3, \dots, \bar{v}_k, \bar{v}_k, t]$ um caminho de aumento em $\overline{G}$ (em relação a $E_{\overline{G}}$). É fácil ver que $[s, v_1, v_2, v_3, \dots, v_k, t]$ é um caminho par em G de s a t .

□

Pelo Teorema de BERGE [1957] (ver Teorema 5.2.1), podemos concluir que $\overline{G}$ tem um caminho de aumento de s a t se, e somente se, o emparelhamento $E_{\overline{G}}$ não é máximo. Assim, podemos enunciar o seguinte corolário.

Corolário 5.2.4 *Existe um caminho par de s a t em G se, e somente se, $\overline{G}$ tem um emparelhamento perfeito.*

Pelo Corolário 5.2.4, podemos resolver $\text{PCamPar}(G, s, t)$ resolvendo o problema $\text{PEmpMax}(\overline{G})$. Assim, utilizando o algoritmo de MICALI E VAZIRANI [1980] para o problema do emparelhamento máximo, podemos resolver o $\text{PCamPar}(G, s, t)$ em $O(\sqrt{nm})$.

A partir do emparelhamento $E_{\overline{G}}$, podemos encontrar um emparelhamento perfeito em $\overline{G}$ simplesmente encontrando um caminho de aumento em $\overline{G}$ (ou seja, executando um passo de aumentação). Para isso, podemos utilizar, por exemplo, o algoritmo de PAPE E CONRADT [1980] que faz um passo de aumentação em tempo $O(n^2)$, ou usar o algoritmo de MICALI E VAZIRANI [1980] que resolve $\text{PFaseCompleta}(\overline{G}, E_{\overline{G}})$ em $O(m)$.

Apresentamos a seguir um algoritmo linear para resolver $\text{PCamPar}(G, s, t)$ utilizando uma fase do algoritmo de MICALI E VAZIRANI [1980].

Algoritmo Verifica_Caminho_Par_Via_Fase_Completa(G, s, t)

Entrada: Um grafo G e dois vértices distintos s e t de G .

Saída: “Sim”, se existir um caminho par entre s e t ;

“Não”, caso contrário.

Passo 1: Construa $\overline{G}$ e $E_{\overline{G}}$;

Passo 2: Utilizando o algoritmo de MICALI E VAZIRANI [1980], resolva PFaseCompleta($\overline{G}$, $E_{\overline{G}}$);

Passo 3: Se esse algoritmo devolver um caminho de aumento entre s e t , então responda “Sim”. Caso contrário, responda “Não”.

◇

5.2.3 O algoritmo de LaPaugh e Papadimitriou

Nesta subseção veremos um algoritmo linear para o PCamPar(G , s , t) desenvolvido por LAPAUGH E PAPADIMITRIOU [1984]. A grande vantagem deste algoritmo é a sua simplicidade (tanto em termos das idéias em que se baseia, quanto em termos de sua implementação).

Considere inicialmente os seguintes conjuntos:

$$C_G = \{ C \mid C \text{ é um componente biconexo de } G \} \text{ e}$$

$$Ar_G = \{ v \mid v \text{ é um ponto de articulação de } G \}.$$

Construa, a partir de G , o grafo T_G tomando:

$$V_{T_G} = C_G \cup Ar_G \text{ e}$$

$$A_{T_G} = \{ \{C, v\} \mid C \in C_G, v \in C \text{ e } v \in Ar_G \}.$$

É fácil verificar que T_G é uma árvore (veja um exemplo na Figura 5.2). Considere agora a função $im: V_G \rightarrow V_{T_G}$ assim definida: se v é um ponto de articulação, então $im(v) = v$; caso contrário $im(v)$ é o (único) componente biconexo C tal que $v \in C$.

Utilizaremos a seguir os resultados abaixo, bastante conhecidos na literatura (veja, por exemplo, BONDY E MURTY [1976]).

Lema 5.2.5 *Seja G um grafo 2-conexo e s, t dois vértices distintos de G . Então existem pelo menos dois caminhos de s a t disjuntos nos vértices.*

Lema 5.2.6 *Se G é um grafo 2-conexo, então para quaisquer duas arestas distintas de G existe um circuito que contém essas arestas.*

Figura 5.2: Um grafo G e o correspondente grafo T_G .

A partir das definições e dos lemas anteriores podemos provar os seguintes resultados.

Lema 5.2.7 *Seja G um grafo e s, t dois vértices distintos de G . Sejam $C_1, C_2, \dots, C_k$ os vértices de T_G correspondentes aos componentes biconexos de G que estão no caminho de $im(s)$ a $im(t)$ em T_G . Se P é um caminho de s a t em G , então P só contém arestas dos componentes biconexos $C_1, C_2, \dots, C_k$.*

Prova: Não é difícil ver que existe em G um caminho P' de s a t consistindo de subcaminhos em $C_1, C_2, \dots, C_k$. Considere o grafo $G[A_P \oplus A_{P'}]$. Note que esse grafo consiste de circuitos, onde cada circuito está em um dos componentes biconexos $C_1, C_2, \dots, C_k$. Assim, as arestas de P estão nos componentes biconexos $C_1, C_2, \dots, C_k$.

□

Lema 5.2.8 *Seja G um grafo 2-conexo e s, t dois vértices distintos de G . Existe em G um caminho par e outro ímpar, de s a t , se e somente se, G não é bipartido.*

Prova: $(\Rightarrow)$ Sejam P e Q dois caminhos de s a t de paridades diferentes. Então o grafo $G[A_P \oplus A_Q]$ consiste de circuitos e pelo menos um desses circuitos deve ser ímpar. Logo, G não é um grafo bipartido.

Figura 5.3: Exemplo da construção de G' (Lema 5.2.8).

($\Leftarrow$) Seja $C = [v_1, v_2, \dots, v_l, v_1]$ um circuito ímpar em G . Vamos contruir, a partir de G , um novo grafo G' (veja Figura 5.3), acrescentando dois vértices novos u e v ; e as arestas $\gamma_1 = \{u, s\}$, $\gamma_2 = \{u, t\}$, $\beta_i = \{v, v_i\}$, para $i = 1, 2, \dots, l$. Mais formalmente,

$$\begin{aligned} V_{G'} &= V_G \cup \{u, v\}, \\ A_{G'} &= A_G \cup \{\gamma_1, \gamma_2\} \cup \{\beta_1, \beta_2, \dots, \beta_l\}. \end{aligned}$$

Uma vez que G é 2-conexo, claramente G' é 2-conexo. Assim, segue-se do Lema 5.2.5, que existem dois caminhos de u a v disjuntos nos vértices. Sejam $P = [u, \gamma_1, s, \dots, v_j, \beta_j, v]$ e $Q = [u, \gamma_2, t, \dots, v_k, \beta_k, v]$ tais caminhos. Podemos supor que os vértices v_j e v_k são os únicos vértices do circuito C que estão nesses caminhos. Suponha, sem perda de generalidade, que $j < k$. É fácil ver que os seguintes caminhos de s a t têm paridades diferentes:

$$\begin{aligned} &[s, \dots, v_j, v_{j+1}, \dots, v_k, \dots, t] \text{ e} \\ &[s, \dots, v_j, v_{j-1}, \dots, v_1, v_l, \dots, v_k, \dots, t]. \end{aligned}$$

□

Teorema 5.2.9 *Seja G um grafo e s e t dois vértices distintos de G . Existem em G um caminho par e outro ímpar de s a t se, e somente se, no caminho de $im(s)$ a $im(t)$ em T_G existe um vértice correspondente a um componente biconexo de G que não é bipartido.*

Prova: Sejam $C_1, C_2, \dots, C_k$ os vértices de T_G correspondentes aos componentes biconexos no caminho de $im(s)$ a $im(t)$. Primeiramente suponhamos que exista em G dois caminhos P e P' , de s a t de paridades diferentes. Observe que o grafo $G[A_P \oplus A_{P'}]$ consiste de circuitos, sendo que pelo menos um deles é ímpar. Pelo Lema 5.2.7 podemos concluir que um desses circuitos está em um dos componentes biconexos $C_1, C_2, \dots, C_k$, e portanto, algum componente biconexo não é bipartido.

Suponhamos que exista j ($1 \leq j \leq k$) tal que C_j seja um componente biconexo não bipartido. Seja a_i o ponto de articulação pertencente a C_i e C_{i+1} , para $i = 1, 2, \dots, k-1$. Seja Q um caminho de s a a_{j-1} e Q' um caminho de a_j a t em G . Pelo Lema 5.2.7, Q contém só arestas dos componentes biconexos $C_1, C_2, \dots, C_{j-1}$ e Q' só contém arestas dos componentes biconexos $C_{j+1}, \dots, C_k$. Pelo Lema 5.2.8, em C_j existem dois caminhos R e R' de a_j a a_{j+1} de paridades diferentes, e portanto, em G os caminhos $Q + R + Q'$ e $Q + R' + Q'$, de s a t , têm paridades diferentes.

□

Apresentamos a seguir o algoritmo que LAPAUGH E PAPADIMITRIOU [1984] obtiveram para resolver $PCamPar(G, s, t)$.

Algoritmo Verifica_Caminho_Par_Via_Biconexos(G, s, t)

Entrada: Um grafo G e dois vértices distintos s e t de G .

Saída: “Sim”, se existir um caminho par entre s e t ;
 “Não”, caso contrário.

Passo 1: Encontre todos os componentes biconexos de G ;

Passo 2: Construa T_G e encontre nesse grafo o caminho de $im(s)$ a $im(t)$;

Passo 3: Se um dos componentes biconexos correspondentes aos vértices desse caminho não é bipartido, então responda “Sim”. Caso contrário, encontre qualquer caminho de s a t em G . Se o caminho encontrado é par, então responda “Sim”. Caso contrário, responda “Não”.

◇

O Passo 1 pode ser implementado em $O(m)$ usando um algoritmo baseado numa busca em profundidade (veja AHO, HOPCROFT, ULLMAN [1974]). Não é difícil ver que os passos 2 e 3 podem também ser implementados em $O(m)$. Temos então o seguinte resultado.

Teorema 5.2.10 *O algoritmo Verifica_Caminho_Par_Via_Biconexos resolve $PCamPar(G, s, t)$ em tempo linear.*

Figura 5.4: Um digrafo D e o correspondente digrafo D' (Teorema 5.3.1).

5.3 Existência de caminhos pares em digrafos

Enquanto existem algoritmos lineares que resolvem $\text{PCamPar}(G, s, t)$ para o caso em que G é um grafo, esse problema é $\mathcal{NP}$ -completo para digrafos, conforme veremos no teorema a seguir, devido a LAPAUGH E PAPADIMITRIOU [1984].

Teorema 5.3.1 *Se D é um digrafo e s e t são dois vértices distintos de D , então $\text{PCamPar}(D, s, t)$ é $\mathcal{NP}$ -completo.*

Prova: É imediato que PCamPar é um problema em $\mathcal{NP}$. É suficiente então provarmos que um problema $\mathcal{NP}$ -completo pode ser polinomialmente reduzido ao PCamPar . Para isso, considere o seguinte problema: “dados um digrafo D e três vértices distintos s, t e m , decidir se o digrafo D contém um caminho de s a t que passa pelo vértice m ”. FORTUNE, HOPCROFT E WYLLIE [1980] mostraram que esse problema é $\mathcal{NP}$ -completo.

Seja D um digrafo e s, m e t três vértices distintos de D . Construa um digrafo D' subdividindo cada vértice de D distinto de m (veja um exemplo na Figura 5.4). Mais formalmente temos que

$$\begin{aligned} V_{D'} &= \{v_1, v_2 \mid v \in V_D - m\} + m, \\ A_{D'} &= \{(v_1, v_2) \mid v \in V_D - m\} \cup \{(u_2, v_1) \mid (u, v) \in A_D, v \neq m\} \cup \\ &\quad \{(v_2, m) \mid (v, m) \in A_D\} \cup \{(m, v_1) \mid (m, v) \in A_D\}. \end{aligned}$$

Observe que $D' - m$ é um digrafo bipartido. Dessa forma, em D' qualquer caminho de s_1 a t_2 , que não passa por m , tem comprimento ímpar. Portanto, existe em D um caminho de s a t que passa por m se, e somente se, existe em D' um caminho par de s_1 a t_2 .

□

5.4 Busca de caminhos pares em grafos

Para decidir se um grafo tem um caminho par entre dois vértices distintos não precisamos necessariamente encontrar um tal caminho. De fato, nem todos os algoritmos que vimos na Seção 5.3 exibem explicitamente tais caminhos, quando existem. É claro que em alguns casos, como os algoritmos baseados na construção de Edmonds, fica evidente como encontrar polinomialmente um tal caminho, quando existe. A pergunta natural que surge então é se existe um algoritmo linear para resolver PEncCamPar. Veremos que a resposta a essa pergunta é afirmativa, apresentando, na primeira subseção, um algoritmo linear baseado na redução para o problema da fase completa.

Na terceira subseção, descrevemos um outro algoritmo, também linear, que resolve PEncCamPar reduzindo-o para o chamado problema do triângulo. Cabe aqui ressaltar que tais algoritmos são extremamente complicados. Dessa forma, na segunda subseção, apresentamos um algoritmo simples para o mesmo problema, de complexidade $O(nm)$, reduzindo-o para o problema de encontrar dois caminhos disjuntos, entre dois vértices distintos, em um grafo 2-conexo. Vale aqui lembrar que podemos também utilizar um passo de aumento do algoritmo de PAPE E CONRADT [1980] (que é fácil de ser implementado) e resolver PEncCamPar em $O(n^2)$.

5.4.1 Redução de PEncCamPar para PFaseCompleta

Dado um grafo G qualquer, considere o grafo $\overline{G}$, obtido a partir de G , e o emparelhamento $E_{\overline{G}}$, ambos descritos na Seção 5.2.2. Observe que a demonstração do Teorema 5.2.3, nos sugere o seguinte algoritmo linear para encontrar um caminho par de s a t em G , a partir de um caminho alternante (relativo a $E_{\overline{G}}$), de s a t , em $\overline{G}$.

Algoritmo Caminho_Par_Via_Alternante($\overline{G}$, P)

Entrada: Um caminho alternante P de s a t em $\overline{G}$ (relativo a $E_{\overline{G}}$).

Saída: Um caminho par de s a t em G .

passo 1: Suponha que $P = [s, \bar{v}_1, \bar{v}_1, \bar{v}_2, \bar{v}_2, \bar{v}_3, \bar{v}_3, \dots, \bar{v}_k, \bar{v}_k, t]$;

passo 2: Retorne o caminho $[s, v_1, v_2, v_3, \dots, v_k, t]$.

◇

Portanto, usando qualquer um dos algoritmos polinomiais que resolvem $\text{PEmpMax}(\bar{G})$ (mencionado na Seção 5.2.1) podemos resolver $\text{PEncCamPar}(G, s, t)$. Utilizando um passo de aumento do algoritmo de PAPE E CONRADT [1980] podemos encontrar um caminho par em G de s a t em $O(n^2)$.

Algoritmo Encontra_Caminho_Par_Via_Pape_Conradt(G, s, t)

Entrada: Um grafo G e dois vértices distintos s e t de G .

Saída: Um caminho par de s a t , se existir um;
“Não”, caso contrário.

Passo 1: Construa $\bar{G}$ e $E_{\bar{G}}$;

Passo 2: Utilizando um passo do algoritmo de PAPE E CONRADT [1980], procure um caminho de aumento em $\bar{G}$ (em relação a $E_{\bar{G}}$);

Passo 3: Se esse algoritmo devolver um caminho de aumento P de s a t , então retorne o caminho devolvido por $\text{Caminho_Par_Via_Alternante}(\bar{G}, P)$. Caso contrário, responda “Não”.

◇

Para obtermos um algoritmo linear para resolver $\text{PEncCamPar}(G, s, t)$, basta usarmos o algoritmo de MICALI E VAZIRANI [1980], que resolve $\text{PFaseCompleta}(\bar{G}, E_{\bar{G}})$ em $O(m)$.

Algoritmo Encontra_Caminho_Par_Via_Fase_Completa(G, s, t)

Entrada: Um grafo G e dois vértices distintos s e t de G .

Saída: Um caminho par de s a t , se existir um;
“Não”, caso contrário.

Passo 1: Construa $\bar{G}$ e $E_{\bar{G}}$;

Passo 2: Utilizando o algoritmo de MICALI E VAZIRANI [1980], resolva $\text{PFaseCompleta}(\bar{G}, E_{\bar{G}})$;

Passo 3: Se esse algoritmo devolver um caminho de aumento P de s a t , então retorne o caminho devolvido por $\text{Caminho_Par_Via_Alternante}(\bar{G}, P)$. Caso contrário, responda “Não”.

◇

5.4.2 Redução de PEncCamPar para o problema de encontrar dois caminhos disjuntos

LAPAUGH E PAPADIMITRIOU [1984] mostraram que podemos utilizar a árvore T_G , descrita na Seção 5.2.3, para restringir a busca de um caminho par em grafos 2-conexos que não são bipartidos. Usando T_G , podemos determinar os componentes biconexos e os pontos de articulação de G que estão no caminho de s a t . Se todos os componentes biconexos do caminho em T_G de $im(s)$ a $im(t)$ são bipartidos, então basta encontrar um caminho qualquer de s a t , uma vez que em um componente biconexo bipartido, todos os caminhos entre dois vértices fixos têm a mesma paridade. Suponhamos então que exista um componente biconexo não bipartido, digamos C , correspondente a um vértice no caminho em T_G de $im(s)$ a $im(t)$. Considere a e a' os pontos de articulação de G que são adjacentes a C nesse caminho (eventualmente, $a = s$ e/ou $a' = t$). Seja P um caminho de s a a e Q um caminho de a' a t em G . Se $|A_P| + |A_Q|$ for par (ímpar), então é suficiente buscar um caminho par (ímpar) R no componente biconexo C de a a a' , uma vez que $P + R + Q$ seria um caminho par de s a t .

Portanto, podemos considerar que o grafo de entrada do problema PEncCamPar é 2-conexo não bipartido.

De acordo como o Lema 5.2.8, se G é um grafo 2-conexo não bipartido, então G possui um caminho par (e outro ímpar) entre quaisquer dois vértices distintos. Analisando a demonstração desse lema, ocorreu-nos que o problema de encontrar um caminho par, entre dois vértices distintos s e t , pode ser reduzido ao problema dos dois caminhos disjuntos (em grafos 2-conexos), cuja definição damos a seguir.

Problema dos dois caminhos: **PDoisCam**(G, u, v)

Dados: Um grafo 2-conexo (não bipartido) G e dois vértices distintos u e v .

Problema: Encontrar dois caminhos disjuntos, de u a v , em G .

Descrevemos a seguir um algoritmo recursivo bastante simples que resolve **PDoisCam**(G, u, v). Este algoritmo tem complexidade $O(nm)$. Posteriormente, apresentaremos um algoritmo *linear* que resolve esse mesmo problema utilizando o conceito de trilha entrelaçada.

5.4.2.1 Um algoritmo polinomial para PDoisCam

O algoritmo a seguir resolve **PDoisCam**(G, u, v) e foi baseado na prova do Lema 5.2.5 (cf. BONDY E MURTY [1976])

Algoritmo Extrai_Dois_Caminhos(G, P, u, v)

Entrada: Um menor caminho P entre dois vértices distintos u e v de um grafo 2-conexo G .

Saída: Dois caminhos disjuntos de u a v .

Passo 1: Se $|P| = 1$, então encontre um circuito que passa pela aresta $\{v, u\}$. Seja P_1 a seção desse circuito, de u a v , que é distinta da aresta $\{v, u\}$. Retorne P_1 e $[u, v]$;

Passo 2: Caso contrário, seja x o vértice de V_P tal que $\{x, v\} \in A_P$ e seja $P' = P - v$. Sejam P_1 e P_2 os dois caminhos disjuntos devolvidos pela chamada de Extrai_Dois_Caminhos(G, P', u, x) (note que P' é um menor caminho de u a x , em G);

Passo 3: Como G é 2-conexo, então $G - x$ é conexo. Seja Q um (menor) caminho de u a v em $G - x$. Se Q é disjunto de P_1 (respectivamente P_2), então retorne Q e $P_1 + [x, v]$ (respectivamente $P_2 + [x, v]$). Caso contrário, seja w o último vértice de Q que intercepta P_1 ou P_2 (note que w pode ser o vértice v). Suponha que $w \in P_1$. Seja P_3 a seção de P_1 , de u a w ; e Q_1 a seção de Q , de w a v . Retorne $P_3 + Q_1$ e $P_2 + [x, v]$. (Se $w \in P_2$, obtemos dois caminhos disjuntos de maneira análoga.)

◇

É fácil ver, por indução no comprimento de P , que o algoritmo Extrai_Dois_Caminhos encontra corretamente dois caminhos disjuntos de s a t . Vamos denotar por $T(k, m)$ a complexidade do algoritmo Extrai_Dois_Caminhos quando aplicado a um grafo com m arestas e um caminho P de comprimento k . Então

$$\begin{aligned} T(1, m) &= O(m); \\ T(k, m) &= T(k-1, m) + O(m), \text{ se } k > 1. \end{aligned}$$

Logo, a complexidade de Extrai_Dois_Caminhos é $T(k, m) = O(km) = O(nm)$.

5.4.2.2 Um algoritmo linear para PDoisCam

Sejam s e t dois vértices distintos de G . Seja $P = [v_1, v_2, \dots, v_l]$ um caminho de s a t . Dizemos que um vértice v_i é um **ancestral** de um vértice v_j em P , se $i < j$. Dizemos que v_i é um **predecessor** (sucessor, respectivamente) de v_j em P , se $i = j - 1$ ($i = j + 1$, respectivamente). Uma sequência de vértices $T = [u_1 = s, u_2, \dots, u_k = t]$ é uma **trilha entrelaçada** com P , se T satisfaz as seguintes propriedades:

Figura 5.5: Figura para o Lema 5.4.1.

Propriedade 1: T é uma trilha de s a t ;

Propriedade 2: todos os vértices repetidos em T estão em V_P ;

Propriedade 3: se uma aresta $\{u_i, u_{i+1}\}$ é também uma aresta de P , então u_i é sucessor de u_{i+1} em P ;

Propriedade 4: se $u_i \in V_T \cap V_P$, $1 < i < k$, então $\{u_{i-1}, u_i\}$ ou $\{u_i, u_{i+1}\}$ é aresta de P .

A Propriedade 3 garante que se uma aresta α é usada por P e T , então α aparece em sentidos opostos em P e T . A Propriedade 4 garante que se um vértice interno de P é usado pela trilha, então pelo menos uma das duas arestas de P adjacentes a esse vértice também é usada pela trilha. Isso nos permite afirmar que se P e T são disjuntos nas arestas, também são disjuntos nos vértices.

Lema 5.4.1 *Seja G um grafo e sejam s e t dois vértices distintos de G e suponha que exista pelo menos um caminho P de s a t . Se existir uma trilha T entrelaçada com P , então existem dois caminhos disjuntos de s a t .*

Prova: Se não existem arestas comuns a P e T , decorre da Propriedade 4 que P e T são disjuntos nos vértices.

Suponha que P e T têm pelo menos uma aresta em comum. Vamos definir P' e T'' , onde P' é um caminho de s a t , T'' é uma trilha entrelaçada com P' ; e além disso, P' e T'' têm menos arestas em comum do que P e T .

Seja $u \neq s$ o primeiro vértice interno de T que está em P . Pela Propriedade 4, existe uma aresta $\{u, v\}$ que está em P e em T , com v sucedendo u em T e com u sucedendo v em P . Seja Q_1 a seção de T de s a u ; e seja Q_2 a seção de P de u a t . (Veja Figura 5.5.)

Figura 5.6: O vértice z se repete em T' e não está em P' .

Definimos P' como a concatenação de Q_1 com Q_2 . É imediato concluir que P' é um *passeio* de s a t . Pela definição de trilha entrelaçada, T somente tem vértices repetidos em P , o que implica que Q_1 é um *caminho* de s a u . Pela escolha de $\{u, v\}$, segue que Q_2 é um caminho de u a t internamente disjunto de Q_1 , e portanto P' é um caminho de s a t .

Seja $x \neq s$ o primeiro vértice interno de P que está em T (eventualmente, $x = v$). Pela Propriedade 4, existe uma aresta $\{x, y\}$ que está em P e em T , com x sucedendo y em T e com y sucedendo x em P . Seja Q_3 a seção de P de s a x ; e seja Q_4 a seção de T de x a t . (Veja Figura 5.5.)

Definimos T' como a concatenação de Q_3 com Q_4 . É imediato concluir que T' é um *passeio* de s a t . Pela escolha de x , o único vértice comum a Q_3 e Q_4 é x , e como Q_4 é parte de uma trilha, T' é também uma trilha.

Observe que pela escolha de u e x , o vértice x aparece em P antes de u , e o vértice u aparece em T antes de x . Todos os vértices e arestas entre u e x (entre x e u , respectivamente) em P (em T , respectivamente) não fazem parte de P' (de T' , respectivamente). Portanto, os únicos vértices e arestas comuns a P' e T' são vértices e arestas comuns a Q_2 e Q_4 . Eventualmente, alguns vértices em $V_{T'}$, que estão na seção de P de x a u , podem se repetir em T' . (Veja um exemplo na Figura 5.6.) Dessa forma, T' pode ter vértices repetidos que não estão em P' . Neste caso, para cada vértice $w \in V_{T'} - V_{P'}$, que se repete em T' , retire a seção de T' correspondente entre a primeira e a última ocorrência de w . Denote essa nova trilha por T'' .

Uma vez que Q_2 e Q_4 são respectivamente seções de P e T que são entrelaçados, e como foram retirados os vértices que se repetem na trilha T' que não pertencem ao caminho P' , concluímos que T'' é um trilha entrelaçada com P' .

Como P' e T'' têm menos arestas em comum do que P e T , a indução está completa.

□

Um algoritmo para encontrar uma trilha entrelaçada, se esta existir, segue das seguintes idéias. Começamos fazendo uma busca em profundidade em G , a partir de s , tentando visitar o vértice t . Cada vez que essa busca visita um vértice $u \neq s, t$ de P , forçamos que a busca obedeça as Propriedades 3 e 4 da definição de trilha entrelaçada. Além disso, esse algoritmo constrói um conjunto de vértices *Especial* que é formado por vértices de P que são atingidos pela trilha por um vértice que não é seu sucessor em P . Neste caso, para se garantir a Propriedade 4, precisamos usar em T uma das arestas adjacentes a esse vértice em P . Como em T somente podemos usar arestas de P com sentidos opostos, isso deverá ser feito pelo algoritmo. Depois que um vértice v é marcado como especial, não podemos permitir que esse vértice seja atingido de outra forma que não seja através de seu sucessor em P . Além disso, o algoritmo empilha todos os ancestrais de v em P até encontrar um vértice w em P , que já foi visitado. Isso permitirá que a trilha T construída pelo algoritmo tenha uma propriedade a mais da trilha entrelaçada, que facilitará encontrar dois caminhos disjuntos de s a t a partir de T e P .

Propriedade 5: Seja $u \neq s$ o primeiro vértice de T que está em P . Seja $\{x, y\}$ a primeira aresta em T , a partir de u , que não está em P . (Note que $x \in V_P$ e que x também é o primeiro vértice de P , distinto de s , que está em T ; e que portanto a seção interna de P de s a x não tem vértices em comum com T .) Então os vértices e arestas internos da seção de P de x a u não fazem parte da seção de T de x a t .

Para um caminho $R = [r_1, r_2, \dots, r_k]$ vamos definir as seguintes funções:

$$\text{Pred}(v, R) = \begin{cases} \emptyset & \text{se } v \notin R; \\ \emptyset & \text{se } v = r_1; \\ r_{i-1} & \text{se } v = r_i, 1 < i \leq k. \end{cases}$$

$$\text{Suc}(v, R) = \begin{cases} r_{i+1} & \text{se } v = r_i, 1 \leq i < k; \\ \emptyset & \text{se } v = r_k; \\ \emptyset & \text{se } v \notin R. \end{cases}$$

Abaixo segue o algoritmo em detalhes. Provaremos a seguir que se não for possível visitar t através dessa busca em profundidade feita pelo algoritmo, então não existem dois caminhos disjuntos de s a t em G .

Algoritmo Busca-Trilha(G, P, s, t)

Entrada: Um caminho P entre dois vértices distintos s e t de um grafo 2-conexo G .

Saída: Uma trilha entrelaçada com P de s a t .

começo

crie uma pilha vazia; $Aresta_Visitada := \emptyset$;

$Visitados := \{s\}$; $Especial := \emptyset$; empilhe(s); $T := [s]$;

enquanto pilha não vazia e t não é atingível faça

começo

seja u o vértice do topo da pilha;

se existe $\{u, v\} \in A_G - Arresta_Visitada$ então

começo

se $v \notin Visitados$ e $v \neq \text{Suc}(u, P)$

(* Esta última condição não faria parte de uma busca em profundidade usual. Serve para garantir a Propriedade 3. *)

e $(v \notin Especial \text{ ou } u = \text{Suc}(v, P))$ então

(* A única forma legítima de se atingir um vértice já marcado como especial é através de seu sucessor em P . *)

começo

se $u \in V_P$ e $\text{Pred}(u, P) \notin Visitados$ então

(* O algoritmo dá preferência a um vértice do caminho P . *)

$v := \text{Pred}(u, P)$;

se $v = t$ então

começo

$T := T + [u, t]$; t é atingível;

fim

senão

se $v \in V_P$ e $u \neq \text{Suc}(v, P)$ então

(* Precisamos garantir aqui a Propriedade 4. O vértice v não será nem considerado visitado nem irá para a pilha. *)

começo

$Especial := Especial + v$; $w := \text{Pred}(v, P)$;

empilhe(w); $Visitados := Visitados + w$;

$T := T + [u, v, w]$;

$Aresta_Visitada := Arresta_Visitada + \{v, w\}$;


```

    (* se  $w$  é especial, não será mais considerado especial. *)
    Especial := Especial -  $w$ ;
  fim
senão (* Busca em profundidade usual *)
  começo
     $T := T + [u, v]$ ; empilhe( $v$ ); Visitados := Visitados +  $v$ ;
    (* se  $v$  é especial, não será mais considerado especial. *)
    Especial := Especial -  $v$ ;
  fim
fim
Aresta_Visitada := Aresta_Visitada +  $\{u, v\}$ ;
fim
senão
  começo
    Desempilhe( $u$ );  $T := T - [u]$ ;
    seja  $x$  o vértice do topo da pilha; seja  $y$  o último vértice de  $T$ ;
    se  $x \neq y$  então
      (*  $y$  era um vértice especial que não entrou na pilha *)
       $T := T - [y]$ ;
    fim
  fim
fim
fim

```

◇

Lema 5.4.2 *Se o algoritmo declarar o vértice t atingível, então a sequência de vértices T , devolvida pelo algoritmo Busca-Trilha, é uma trilha entrelaçada com P .*

Prova: Claramente T é uma trilha, pois cada aresta de G é visitada no máximo uma única vez. As Propriedades 3 e 4 da definição de uma trilha entrelaçada são garantidas conforme comentários no Algoritmo.

□

Observe que só ocorre repetições de vértices em T quando a busca em profundidade é alterada para se garantir a Propriedade 4. De resto, como usual numa busca em profundidade, cada vértice é considerado “visitado” e vai para a pilha no máximo uma única vez. Além disso, note que quando o bloco referente à Propriedade 4 é executado, um vértice de P é marcado como “especial” (sem entrar na pilha). Uma nova visita a esse vértice somente será permitida através de um seu sucessor em P , no caso em que o vértice será considerado como “visitado” (entrando na pilha e deixando de ser “especial”). Daí para frente, como é usual numa busca em profundidade, esse vértice não será mais usado pela trilha.

Lema 5.4.3 *Seja G um grafo e sejam s e t dois vértices distintos de G e suponha que exista pelo menos um caminho P de s a t . Então G contém dois caminhos disjuntos de s a t se, e somente se, t é declarado atingível pelo Algoritmo Busca_Trilha.*

Prova: Se o vértice t é atingível, pelos Lemas 5.4.1 e 5.4.2 existem dois caminhos disjuntos entre s e t .

Por outro lado, suponha que t não foi declarado atingível pelo algoritmo Busca_Trilha. Se s é o único vértice em P que foi visitado, então claramente G não contém dois caminhos disjuntos de s a t . Assuma então que exista algum vértice de P , distinto de s , que foi visitado pelo algoritmo. Assim, G contém exatamente um vértice $w \in \text{Especial}$. Seja $X = \text{Visitados} - w$. Seja $\{u, v\}$ uma aresta de G tal que $u \in X$ e $v \in V_G - X$. Suponha que $v \notin V_P$. Neste caso v seria visitado pelo algoritmo e portanto v pertenceria a X . Mas isso é uma contradição, pois estamos assumindo que $v \in V_G - X$. Dessa forma, podemos considerar que $v \in V_P$. Evidentemente v é um vértice que foi visitado pelo algoritmo. Afirmamos que $v = w$. De fato, w é o único vértice visitado em $V_G - X$. Logo, qualquer caminho de s a t em G passa pelo vértice w e portanto G não contém dois caminhos disjuntos de s a t .

□

É fácil ver que utilizando estruturas de dados convenientemente, o algoritmo Busca_Trilha pode ser implementado de modo que sua complexidade seja $O(m)$.

Os resultados dos Lemas 5.4.2 e 5.4.3 e a demonstração do Lema 5.4.1 induz o seguinte algoritmo linear para se transformar P e uma trilha devolvida pelo algoritmo Busca_Trilha(G, P, s, t) em dois caminhos disjuntos de s a t . Cabe aqui lembrar que a trilha devolvida pelo algoritmo Busca_Trilha satisfaz a Propriedade 5.

Algoritmo Dois_Caminhos_Trilha(G, P, T, s, t)

Entrada: Um caminho P entre dois vértices distintos s e t de um grafo 2-conexo G e uma trilha entrelaçada T com P , devolvida pelo algoritmo Busca_Trilha(G, P, s, t).

Saída: Dois caminhos disjuntos de s a t .

Passo 1: $v := s$;

Passo 2: (* Devido à propriedade da trilha devolvida pelo algoritmo Busca_Trilha, os vértices internos de T de s a v não estão em P . *)

Percorra a trilha T , a partir do vértice v , até encontrar um vértice $u \in V_P - \{v\}$;

Passo 3: Se $u = t$, retorne P e T e pare. Caso contrário vá para o Passo 4.

Passo 4: Seja Q_1 a seção de T de s a u . Seja Q_2 a seção de P de u a t . Definimos $P' := Q_1 + Q_2$.

Passo 5: ($\ast \{u, w\} \in A_T$, onde u é sucessor de w em $P \ast$)

Percorra a trilha T , a partir do vértice u , até encontrar uma aresta $\{x, z\} \notin A_P$, com $x \in P$.

Passo 6: Seja Q_3 a seção de P de s a x . Seja Q_4 a seção de T de x a t . Definimos $T' := Q_3 + Q_4$.

Passo 7: ($\ast$ Como vale a Propriedade 5 para T e P , segue que T' é uma trilha entrelaçada com P' . Além disso, a Propriedade 5 continua sendo satisfeita para T' e P' (no lugar de T e P) $\ast$)

$T := T'$; $P := P'$; $v := x$;

Passo 8: Vá para o Passo 2;

◇

Note que, podemos implementar o algoritmo `Dois_Caminhos_Trilha` de modo que ele tenha complexidade $O(m)$, se utilizarmos estruturas de dados convenientes (por exemplo, para concatenação de dois passeios, usarmos Listas Ligadas).

5.4.2.3 Um algoritmo linear para `PEncCamPar`

Utilizando as idéias contidas na demonstração do Lema 5.2.8, podemos resolver `PEncCamPar` com o seguinte algoritmo que também tem complexidade $O(m)$. Lembramos aqui que um circuito ímpar pode ser encontrado em tempo linear, conforme mencionamos no Capítulo 2.

Algoritmo `Encontra_Caminho_Par_Via_Dois_Caminhos( $G, s, t$ )`

Entrada: Um grafo 2-conexo não bipartido G e dois vértices distintos s e t de G .

Saída: Um caminho par de s a t .

Passo 1: Construa o grafo G' e sejam u e v os dois vértices novos acrescentados a G para se obter o grafo G'' ;

Passo 2: Encontre um caminho P de u a v ;

- Passo 3: Seja T uma trilha entrelaçada com P devolvida pelo algoritmo `Busca_Trilha( $G', P, u, v$ )`;
- Passo 4: Sejam P_1 e P_2 os dois caminhos devolvidos pelo algoritmo `Dois_Caminhos_Trilha( $G', P, T, u, v$ )`;
- Passo 5: Usando as idéias presentes na demonstração do Lema 5.2.8, e os caminhos disjuntos P_1 e P_2 , encontraremos em G um caminho par Q de s a t . Retorne Q e pare.

◇

5.4.3 Redução de PEncCamPar para o problema do triângulo

Vamos ver nesta subseção que o problema de encontrar um caminho par entre dois vértices distintos, em grafos 2-conexos, pode ser reduzido ao problema do triângulo, cuja definição damos a seguir.

Problema do triângulo: $\text{PTriângulo}(G, x, y, z)$

Dados: Um grafo G e três vértices distintos x, y e z .

Problema: Encontrar um circuito em G que passa por x, y e z .

A redução de $\text{PEncCamPar}(G, s, t)$ para o problema do triângulo pode ser feita da seguinte maneira. Considere o grafo G' construído na demonstração do Lema 5.2.8. Seja $\tilde{G} = G' - u$, onde u é o vértice acrescentado a G (adjacente a s e t), para se obter o grafo G' . Seja H o grafo obtido a partir de $\tilde{G}$ acrescentando dois vértices novos s' e t' ; e três novas arestas $\alpha = \{s', s\}$, $\beta = \{t', t\}$ e $\gamma = \{s', t'\}$. Mais formalmente,

$$\begin{aligned} V_H &= V_{\tilde{G}} \cup \{s', t'\} \text{ e} \\ A_H &= A_{\tilde{G}} \cup \{\alpha, \beta, \gamma\}. \end{aligned}$$

Seja v o vértice que foi acrescentado a G (adjacente aos vértices de um circuito ímpar), para se obter o grafo G' . Então, pelo Lema 5.2.8, é fácil verificar que podemos encontrar uma solução de $\text{PDoisCam}(G, u, v)$ resolvendo $\text{PTriângulo}(H, s', v, t')$, e vice-versa.

LAPAUGH E RIVEST [1980] obtiveram um algoritmo linear para resolver PTriângulo , mas tal algoritmo é complicado e baseado em uma análise exaustiva de casos. Utilizando esse algoritmo, podemos resolver o problema do circuito par em tempo linear.

Algoritmo `Encontra_Caminho_Par_Via_Triângulo( $G, s, t$ )`

Entrada: Um grafo 2-conexo não bipartido G e dois vértices distintos s e t de G .

Saída: Um caminho par de s a t .

- Passo 1: Construa o grafo H (como definido acima);
- Passo 2: Seja C o circuito devolvido pelo algoritmo de LAPAUGH E RIVEST [1980] que resolve $\text{PTriângulo}(H, s', v, t')$;
- Passo 3: Utilizando o circuito C e as idéias presentes na demonstração do Lema 5.2.8, encontraremos um caminho par P de s a t . Retorne P e pare.

◇

Finalizamos esta seção lembrando que o PEncCamPar pode ser resolvido em tempo linear por algoritmos relativamente trabalhosos de serem implementados (como é o caso dos algoritmos $\text{Encontra_Caminho_Par_Via_Fase_Completa}$ e $\text{Encontra_Caminho_Par_Via_Triângulo}$) e e que há outras alternativas para resolvê-lo através de um algoritmo mais simples, também linear ($\text{Encontra_Caminho_Par_Via_Dois_Caminhos}$). O algoritmo $\text{Encontra_Caminho_Par_Via_Pape_Conrad}$ é fácil de ser implementado, porém a sua complexidade é $O(n^2)$.

5.5 Busca de caminhos pares mínimos

Nesta seção apresentaremos dois algoritmos polinomiais para resolver o problema do caminho par mínimo. O primeiro algoritmo resolve PCamParMin reduzindo-o para o problema do emparelhamento perfeito de peso máximo. Utilizando o algoritmo devido a EDMONDS (veja LOVÁSZ E PLUMER [1986]) para resolver este problema, PCamParMin pode ser resolvido em $O(n^3)$. O segundo algoritmo, reduz o problema do caminho par mínimo para o PFaseCompleta já mencionado na Seção 5.2.1.

5.5.1 Redução de PCamParMin para o problema do emparelhamento perfeito de peso máximo

Mostraremos a seguir como o problema do caminho par mínimo pode ser reduzido para o seguinte problema:

Problema do emparelhamento perfeito de peso máximo: $\text{PEmpPPMax}(G)$

Dado: Um grafo $\{0,1\}$ -ponderado G .

Problema: Encontrar em G um emparelhamento perfeito de peso máximo.

Sejam G , s , t os dados de entrada de PCamParMin . A partir de G construa o grafo $\overline{G}$ usando a construção de EDMONDS, descrita na Seção 5.2.2. Para cada aresta de $\overline{G}$, cujas pontas estejam em uma mesma cópia, atribua peso zero; para todas as outras arestas, atribua peso um. É fácil ver que se E é um emparelhamento perfeito em $\overline{G}$, então podemos

encontrar em $\overline{G}$ um caminho alternante (relativo a E) $P = [s, \bar{v}_1, \bar{v}_1, \bar{v}_2, \bar{v}_2, \bar{v}_3, \bar{v}_3, \dots, \bar{v}_k, \bar{v}_k, t]$, tal que as arestas $\{\bar{v}_i, \bar{v}_i\} \notin E$, em $O(m)$. Observe que esse caminho P é também um caminho alternante relativo a $E_{\overline{G}}$. Dessa forma, podemos obter um caminho par, de s a t em G , utilizando o algoritmo `Caminho_Par_Via_Alternante`($\overline{G}$, P). Claramente, um emparelhamento perfeito de peso máximo utiliza o menor número possível de arestas cujas pontas estão em uma mesma cópia de $\overline{G}$. Esses comentários justificam o seguinte resultado.

Teorema 5.5.1 *PCamParMin(G, s, t) pode ser polinomialmente reduzido ao problema de encontrar um emparelhamento perfeito de peso máximo em $\overline{G}$.*

Algoritmo `Caminho_Par_Mínimo_Via_Emparelhamento`(G, s, t)

Entrada: Um grafo G e dois vértices distintos s e t de G .

Saída: Um caminho par mínimo de s a t , se existe um tal caminho.
“Não”, caso contrário.

Passo 1: Construa o grafo $\overline{G}$ e atribua pesos 0/1 às suas arestas, conforme mencionado anteriormente;

Passo 2: Resolva o problema de encontrar um emparelhamento perfeito de peso máximo do grafo $\overline{G}$ (para isso utilize o algoritmo de EDMONDS);

Passo 3: Se em $\overline{G}$ não existe um emparelhamento perfeito, então responda “Não” e pare. Caso contrário, a partir de um emparelhamento perfeito de peso máximo encontre em $\overline{G}$ um caminho alternante P de s a t , como descrito acima.

Passo 4: Aplique o algoritmo `Caminho_Par_Via_Alternante`($\overline{G}$, P) e retorne o caminho par, de s a t , devolvido por esse algoritmo.

◇

5.5.2 Redução de PCamParMin para PFaseCompleta

Considere o grafo $\overline{G}$, obtido a partir de um grafo G , e o emparelhamento $E_{\overline{G}}$, ambos descritos na Seção 5.2.2. Note que um menor caminho de aumento (relativo a $E_{\overline{G}}$) utiliza o menor número possível de arestas cujas pontas estão em uma mesma cópia de G . Isso justifica o seguinte resultado.

Teorema 5.5.2 *PCamParMin(G, s, t) pode ser polinomialmente reduzido ao problema de encontrar em $\overline{G}$ um menor caminho de aumento relativo a $E_{\overline{G}}$.*

Logo, utilizando o algoritmo linear de MICALI E VAZIRANI [1980], que resolve $\text{PFaseCompleta}(\overline{G}, E_{\overline{G}})$, podemos resolver $\text{PCamParMin}(G, s, t)$ em tempo linear.

Algoritmo Caminho_Par_Mínimo_Via_Fase_Completa(G, s, t)

Entrada: Um grafo G e dois vértices distintos s e t de G .

Saída: Um caminho par mínimo de s a t , se existe um tal caminho.
“Não”, caso contrário.

Passo 1: Construa o grafo $\overline{G}$ e o emparelhamento $E_{\overline{G}}$;

Passo 2: Utilize o algoritmo linear de MICALI E VAZIRANI [1980], que resolve $\text{PFaseCompleta}(\overline{G}, E_{\overline{G}})$;

Passo 3: Se o algoritmo devolver um caminho de aumento P , então retorne o caminho par devolvido por $\text{Caminho_Par_Via_Alternante}(\overline{G}, P)$. Caso contrário, responda “Não”.

◇

Como observação final, cabe aqui notar que LAPAUGH E PAPADIMITRIOU [1984] mostraram que podemos restringir a aplicação do algoritmo $\text{Caminho_Par_Mínimo_Via_Fase_Completa}(G, s, t)$ para componentes biconexos que não são bipartidos.

De fato, usando a árvore T_G , podemos determinar os pontos de articulação e os componentes biconexos no caminho de s a t . Se todos esses componentes biconexos são bipartidos, então um caminho mínimo, de s a t pode ser encontrado, bastando para isso concatenar os caminhos mínimos entre os respectivos pontos de articulação de cada um desses componentes. Isso resolve PCamParMin . Consideramos agora o caso em que existem componentes biconexos não bipartidos. Suponha que o caminho de $im(s)$ a $im(t)$ em T_G contenha, nesta ordem, os pontos de articulação $a_1, a_2, \dots, a_k \in Ar_G - \{s, t\}$, e considere $a_{k+1} = t$. É imediato que, para $i = 1, 2, \dots, k$,

um caminho par mínimo de s a $a_{i+1} = \min\{[\text{um caminho par mínimo de } s \text{ a } a_i] + [\text{um caminho par mínimo de } a_i \text{ a } a_{i+1}], [\text{um caminho ímpar mínimo de } s \text{ a } a_i] + [\text{um caminho ímpar mínimo de } a_i \text{ a } a_{i+1}]\};$

um caminho ímpar mínimo de s a $a_{i+1} = \min\{[\text{um caminho par mínimo de } s \text{ a } a_i] + [\text{um caminho ímpar mínimo de } a_i \text{ a } a_{i+1}], [\text{um caminho ímpar mínimo de } s \text{ a } a_i] + [\text{um caminho par mínimo de } a_i \text{ a } a_{i+1}]\}.$

Note que alguns caminhos da expressão acima podem não existir (casos de componentes biconexos bipartidos). Neste caso, na determinação do mínimo deve-se considerar somente caminhos existentes.

Portanto, para encontrar um caminho par mínimo de s a t , é suficiente utilizarmos o algoritmo `Caminho_Par_Mínimo_Via_Fase_Completa` para os componentes biconexos não bipartidos que estão no caminho de $im(s)$ a $im(t)$ em T_G .

5.6 Conclusão

É interessante observar que no caso de grafos, para todos os problemas discutidos neste capítulo existem algoritmos polinomiais. Já no caso de digrafos, os problemas são todos difíceis. O fato de `PCamPar` para digrafos ser $\mathcal{NP}$ -completo implica que para digrafos, `PEncCamPar` é $\mathcal{NP}$ -difícil e `PCamParMin` é $\mathcal{NP}$ -difícil.

Resumimos nos quadros abaixo os resultados discutidos neste capítulo.

Problema (Grafos)	Algoritmo	Comple- xidade
PCamPar	Verifica_Caminho_Par_Via_Fase_Completa	$O(m)$
	Verifica_Caminho_Par_Via_Biconexos	$O(m)$
PEncCamPar	Encontra_Caminho_Par_Via_Fase_Completa	$O(m)$
	Encontra_Caminho_Par_Via_Pape_Conradt	$O(n^2)$
	Encontra_Caminho_Par_Via_Dois_Caminhos	$O(m)$
	Encontra_Caminho_Par_Via_Triângulo	$O(m)$
PCamParMin	Caminho_Par_Mínimo_Via_Emparelhamento	$O(n^3)$
	Caminho_Par_Mínimo_Via_Fase_Completa	$O(m)$

Problema (Digrafos)	Complexidade
PCamPar	$\mathcal{NP}$ -completo
PEncCamPar	$\mathcal{NP}$ -difícil
PCamParMin	$\mathcal{NP}$ -difícil

Os algoritmos `Encontra_Caminho_Par_Via_Dois_Caminhos_Disjuntos` e `Encontra_Caminho_Par_Via_Triângulo` foram inspirados na demonstração do Lema 5.2.8, sendo que o primeiro é consideravelmente simples e o segundo utiliza um algoritmo de LAPAUGH E RIVEST [1980] que é razoavelmente complicado.

Cabe aqui mencionar que está em aberto o problema quanto à existência de um algoritmo $O(m)$ para resolver PCamParMin em grafos 2-conexos não bipartidos, sem utilizar a redução para um problema de emparelhamento.

Conclusão

Apesar do conceito de circuitos e de caminhos ser bastante conhecido na Teoria de Grafos, não encontramos na literatura um trabalho que abrangesse o estudo de problemas sobre circuitos e caminhos com restrição de paridade (tanto em grafos, quanto em digrafos) como os que apresentamos nesta dissertação. Acreditamos, dessa forma, que este trabalho possa ser útil às pessoas interessadas em conhecer o “status” dos problemas considerados ou queiram aprofundar seus estudos sobre os assuntos tratados nesta dissertação. Esperamos também que o mesmo tenha permitido ao leitor identificar questões ainda a serem respondidas (ou que não foram satisfatoriamente respondidas) e que poderiam ser melhor exploradas.

Dois problemas que valeriam a pena ser melhor investigados são $\text{PEncCamParMin}(G, s, t)$ e $\text{PCirParMin}(G)$, onde G é um grafo e s e t são dois vértices distintos de G . Para o segundo problema, seria interessante encontrar um algoritmo (de complexidade $O(|A_G|^2)$ ou melhor) sem recair no problema de encontrar caminhos ímpares mínimos. Para o primeiro problema, acreditamos ser possível encontrar um algoritmo linear simples que aproveite a estrutura do grafo $\overline{G}$ construído via redução de EDMONDS. Cabe aqui notar que $\overline{G}$ tem uma estrutura simétrica e que se $E_{\overline{G}}$ é o emparelhamento inicial em $\overline{G}$, os únicos vértices livres são s e t .

Apêndice A

Implementação do algoritmo Encontra_Circuito_Ímpar_Digrafo

```
# include <stdio.h>
# include <stdlib.h>
# include <ctype.h>

# define vmax 200

/* definicao de um vertice de um digrafo */
struct vertice {
    int v;
    struct vertice * prox;
};

/* definicao de um digrafo */
typedef struct vertice * digrafo[2*vmax];

/* Definicao de algumas variaveis globais */
digrafo D, Dlinha;
int N, i;

/* Pilha */
int pilha[2*vmax], topo;

/* vetores booleanos */
char vertice_marcado[2*vmax];
char esta_na_pilha[2*vmax];
```



```
/* vetor de apontadores para verificar se existe arcos */
/* marcados */
struct vertice * arco_nao_marcado[2*vmax];
```

```
/* Prototipos */
void imprime (digrafo D, int N);
void aloca_vertice (struct vertice * * ap);
void le_monta_digrafo (char nome_arquivo[], digrafo D);
void ins_vertice (digrafo D, int u, int v);
void monta_digrafo_linha (digrafo D, digrafo Dlinha);
int Encontra_Circuito_Impar_Digrafo (digrafo D);
```

```
/* Esta funcao imprime um digrafo */
void imprime (digrafo D, int N) {
    struct vertice *ap;
    int i;
```

```
    for (i=1; i<=N; i++) {
        ap = D[i];
        printf("%3d -> ", i);
        while (ap) {
            printf("%3d", ap->v);
            ap = ap->prox;
        }
        printf("\n");
    }
}
```

```
/* Esta função aloca espaço para um vértice */
void aloca_vertice (struct vertice * * ap) {
    * ap = (struct vertice *)
        malloc ( sizeof (struct vertice));
    if ( ! (* ap) ) {
        printf ("Nao foi possivel alocar espaco\n");
        exit (0);
    }
}
```

```
/* Esta funcao le um arquivo e monta um digrafo */
void le_monta_digrafo (char nome_arquivo[], digrafo D) {
    int v1, v2, i;
    FILE * arquivo;

    arquivo = fopen (nome_arquivo, "r");

    /* numero de vertices */
    fscanf (arquivo, "%d", &N);

    if (N > vmax) {
        printf ("Aumentar o tamanho do vetor\n");
        exit (0);
    }

    for (i=1; i<=N; i++)
        D[i] = NULL;

    while (1) {
        struct vertice * ap_v2;

        /* primeiro vertice */
        fscanf (arquivo, "%d", &v1);
        /* vertice adjacente a v1 */
        fscanf (arquivo, "%d", &v2);

        if (feof ( arquivo )) break;

        aloca_vertice ( & ap_v2);
        ap_v2->v = v2;

        ap_v2->prox = D[v1];
        D[v1] = ap_v2;
    }
    fclose (arquivo);
    imprime (D, N);
}

/* insere o arco (u, v) no digrafo D */
void ins_vertice (digrafo D, int u, int v) {
    struct vertice * x;
```

```

aloca_vertice (& x);
x->v = v;
if (D[u] == NULL) {
    D[u] = x;
    x->prox = NULL;
}
else {
    x->prox = D[u];
    D[u] = x;
}
}

void monta_digrafo_linha (digrafo D, digrafo Dlinha) {
int i;
struct vertice * ap;

    for (i=1; i<=N; i++)
        for (ap = D[i]; ap; ap = ap->prox) {
            ins_vertice (Dlinha, i, N + ap->v);
            ins_vertice (Dlinha, i + N, ap->v);
        }
}

int vertice_linha (int v) {
    if ( v <= N )
        return (v + N);
    else
        return (v - N);
}

int Encontra_Circuito_Impar_Digrafo (digrafo D) {
int v, u;

    monta_digrafo_linha (D, Dlinha);

    /* inicializa vertice_marcado, esta_na_pilha */
    /* e arco_marcado */
    for (i=1; i<=2*N; i++) {
        vertice_marcado[i] = esta_na_pilha[i] = 0;
        arco_nao_marcado[i] = Dlinha[i];
    }
}

```

```
/* inicializa a pilha */
topo = 0;

/* seja v um vertice de Dlinha */
v = 1;

/* empilha v */
pilha[++topo] = v;
esta_na_pilha[v] = 1;

/* marque o vertice v */
vertice_marcado[v] = 1;

while (topo) {
    /* seja v o vertice do topo da Pilha */
    v = pilha[topo];

    /* se existe um arco (v, u) que nao foi marcado */
    if (arco_nao_marcado[v]) {
        u = arco_nao_marcado[v]->v;

        /* marque o arco (v, u) */
        arco_nao_marcado[v] = arco_nao_marcado[v]->prox;

        /* se u nao esta na pilha */
        if ( ! esta_na_pilha[u] )

            /* se u nao foi marcado */
            if ( ! vertice_marcado[u] )

                /* se u' esta na Pilha */
                if ( esta_na_pilha[vertice_linha (u)] ) {

                    /* encontramos um arco fraco */
                    printf("Existe um circuito impar\n");
                    i = topo;
                    while ( pilha[i] != vertice_linha (u) )
                        i--;
                    while ( i <= topo ) {
                        if ( pilha[i] > N )
```



```

        pilha[i] = vertice_linha (pilha[i]);
        printf("%3d ", pilha[i++]);
    }
    if (u > N) u = vertice_linha (u);
    printf("%3d\n", u);
    return 1;
}
else { /* u' nao esta na Pilha */
    pilha[++topo] = u;
    vertice_marcado[u] = 1;
    esta_na_pilha[u] = 1;
}
}
else /* todos os arcos ja foram marcados */
    esta_na_pilha[pilha[topo--]] = 0;
}
return 0;
}

void main (int argc, char * argv[]) {

    printf ("Implementacao de Encontra_Circuito_Impar_Digrafo\n");
    printf ("Ronaldo Fumio Hashimoto 10/12/93\n");
    if (argc != 2) {
        printf ("cimpar %carquivo%c\n", '<', '>');
        exit (0);
    }

    if (access (argv[1], 0) != 0) {
        printf ("arquivo \"%s\" nao foi encontrado\n", argv[1]);
        exit (0);
    }

    le_monta_digrafo (argv[1], D);
    if (! Encontra_Circuito_Impar_Digrafo (D))
        printf("Nao existe circuito impar\n");
}

```

Apêndice B

Tabelas dos problemas

Este apêndice contém um resumo dos resultados apresentados nessa dissertação. Os algoritmos marcados com * foram de nossa autoria.

B.1 Circuitos ímpares em grafos

Problema	Algoritmo	Complexidade
PCirImpar	Verifica_Circuito_Ímpar_Grafo(G)	$O(m)$
PEncCirImpar	Encontra_Circuito_Ímpar_Grafo(G)	$O(m)$
PCirImparMin	KLEE, LADNER E MANBER [1984]	$O(nm)$
	Circuito_Ímpar_Mínimo_Grafo_Seidel(G)	$O(M(n) \log n)$
	Circuito_Ímpar_Mínimo_Grafo(G) *	$O(nm)$

B.2 Circuitos ímpares em digrafos

Problema	Algoritmo	Complexidade
PCirImpar	Verifica_Circuito_Ímpar_Digrafo(D)	$O(m)$
PEncCirImpar	Encontra_Circuito_Ímpar_Digrafo(D) *	$O(m)$
	Circuito_Ímpar(D)	$O(m)$
PCirImparMin	KLEE, LADNER E MANBER [1984]	$O(nm)$
	Circuito_Ímpar_Mínimo_Digrafo(D) *	$O(nm)$

B.3 Circuitos pares em grafos

Problema	Algoritmo	Complexidade
PCirPar	Verifica_Circuito_Par_Grafo	$O(m)$
PEncCirPar	Encontra_Circuito_Par_Grafo	$O(m)$
PCirParMin	Circuito_Par_Mínimo_Grafo	$O(m^2)$

B.4 Circuitos pares em digrafos

Problema	Complexidade
PCirPar	em aberto
PEncCirPar	em aberto
PCirParMin	em aberto

B.5 Caminhos pares/ímpares em grafos

Problema	Algoritmo	Comple- xidade
PCamPar	Verifica_Caminho_Par_Via_Fase_Completa	$O(m)$
	Verifica_Caminho_Par_Via_Biconexos	$O(m)$
PEncCamPar	Encontra_Caminho_Par_Via_Fase_Completa	$O(m)$
	Encontra_Caminho_Par_Via_Pape_Conrad	$O(n^2)$
	Encontra_Caminho_Par_Via_Dois_Caminhos	$O(m)$
	Encontra_Caminho_Par_Via_Triângulo *	$O(m)$
PCamParMin	Caminho_Par_Mínimo_Via_Emparelhamento	$O(n^3)$
	Caminho_Par_Mínimo_Via_Fase_Completa	$O(m)$

B.6 Caminhos pares/ímpares em digrafos

Problema	Complexidade
PCamPar	$\mathcal{NP}$ -completo
PEncCamPar	$\mathcal{NP}$ -difícil
PCamParMin	$\mathcal{NP}$ -difícil

Bibliografia

- [1] A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, 1974.
- [2] H. Alt, N. Blum, K. Mehlhorn and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time $O(n^{1.5}\sqrt{m/\log n})$. *Inform. Process. Lett.*, 37(1991), 237-240.
- [3] E.M. Arkin, C.H. Papadimitriou and M. Yannakakis. Modularity of cycles and paths in graphs. *J. of Assoc. for Comp. Mach.*, 38(1991), 255-274.
- [4] C. Berge. Two theorems in graph theory. *Proc. Nat. Acad. Sci. U.S.A.*, 43(1957), 449-844.
- [5] C. Berge. *Graphs and Hypergraphs*. North-Holland, Amsterdam, 1973.
- [6] H.G. Benatti. *Homeomorfismo em grafos: algoritmos e complexidade computacional*. Dissertação de Mestrado, IME-USP, 1993.
- [7] J.C. Bermond and C. Thomassen. Cycles in digraphs - a survey. *J. of Graph Theory*, 5(1981), 1-43.
- [8] N. Blum. A new approach to maximum matching in general graphs (revised version). Report 8546-CS, University of Bonn, 1993, 1-53.
- [9] J.A. Bondy and U.S R. Murty, *Graph Theory with Applications*, Macmillan, London, 1976.
- [10] A. Caprara and M. Fischetti. Odd Cut-Sets, Odd Cycles, and $0-1/2$ Chvátal-Gomory Cuts, Preprint. University of Bologna, Italy. 1993.
- [11] J. Edmonds. Paths, trees and flowers. *Canad. J. Math.*, 17(1965), 449-467.
- [12] S. Even and O. Kariv. An $O(n^{2.5})$ algorithm for maximum matching in general graphs. *Proc. 16th Ann. Symp. on Foundations of Computer Sci.*, IEEE, 100-112.

- [13] S. Fortune, J. Hopcroft and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(1980), 111-121.
- [14] S. Friedland. Every 7-regular digraph contains an even cycle. *J. of Combinatorial Theory*, B 46(1989), 249-252.
- [15] H.N. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *J. Assoc. Comput. Mach.*, 23(1976), 221-234.
- [16] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, New York, 1979.
- [17] J.E. Hopcroft and R.M. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM J. Comput.*, 2(1973), 225-231.
- [18] J.E. Hopcroft and R.E. Tarjan. Efficient planarity testing. *J. Assoc. Comput. Math.*, 21(1974), 549-568.
- [19] A. Itai and M. Rodeh. Finding a minimum circuit in a graph. In *Ninth Annual ACM Symposium on Theory of Computing*, Boulden, Colorado, May 1977, 1-10.
- [20] T. Kameda and I. Munro. An $O(|V| \cdot |E|)$ algorithm for maximum matching of graphs. *Computing*, 12(1974), 91-98.
- [21] S.B. Karmakar. An algorithm for finding a circuit of even length in a directed graphs. *Int. J. Syst. Sci.*, 15, 11(1984), 1197-1201.
- [22] P.W. Kasteleyn. The statistics of dimers on a lattice. I The number of dimer arrangements on a quadratic lattice. *Physica*, 27(1961), 1209-1225.
- [23] P.W. Kasteleyn. Graph theory and crystal physics. In *Graph Theory and Theoretical Physics*, Academic Press, New York, 1967, 43-110.
- [24] V. Klee, R. Ladner and R. Manber. Signsolvability revisited. *Linear Algebra and its Applications*, 59(1984), 131-157.
- [25] K.M. Koh. Even circuits in directed graphs and Lovász's conjecture. *Bull. Malaysian Math. Soc.*, 7(1976), 47-52.
- [26] A.S. Lapaugh and C.H. Papadimitriou. The even-path problem for graphs and digraphs. *Networks*, 14, 4(1984), 597-614.
- [27] A.S. LaPaugh and R.L. Rivest. The subgraph homeomorphism problem. *J. of Comp. and Syst. Sci.*, 20(1980), 133-149.
- [28] L. Lovász. Coverings and colorings in hypergraphs. In *Proc. Forth. S. E. Conf. on Combinatorics, Graph Theory and Computing*, Utilitas Math., Winnipeg, 1973, 3-12.

- [29] L. Lovász. Problem 2. *Recent Advances in Graph Theory* (M. Friedler, ed.), Proceedings Symposium Prague, 1974, Academia Praha, Prague, 1975.
- [30] L. Lovász and M. Plummer. *Matching Theory*. North-Holland, Amsterdam, 1986.
- [31] A. Lubiw. A note on odd/even cycles. *Discrete Applied Math.*, 22(1988/89), 87-92.
- [32] M. Marcus and H. Minc. On the relation between the determinant and the permanent. *Illinois J. Math.*, 5(1961), 376-381.
- [33] J. A. McHugh. *Algorithmic Graph Theory*, Prentice-Hall, 1990.
- [34] S. Micali and V.V. Vazirani. An $O(|V|^{1/2}|E|)$ algorithm for finding maximum matchings in general graphs. *Twenty-first Annual Symposium on Foundations of Computer Science*, IEEE, New York (1980), 17-27.
- [35] P.A. Peterson and M.C. Loui. The general maximum matching algorithm of Micali and Vazirani. *Algorithmica*, 3(1988), 511-533.
- [36] G. Polya. Aufgabe 424. *Arch. Math. Phys.*, (3) 20(1913) 271.
- [37] R. Seidel. On the all-pairs-shortest-path problem. *Twenty-fourth Annual ACM Symposium on the Theory of Computing*, Victoria, Canada, (1992), 745-749.
- [38] P. Seymour. On the two-colouring of hypergraphs. *Quart. J. Math. Oxford*, Series 3, 25(1974), 303-312.
- [39] P. Seymour and C. Thomassen. Characterization of even directed graphs. *J. of Combinatorial Theory*, B42(1987), 36-45.
- [40] J.A.R. Soares. Comunicação pessoal.
- [41] M.M. Syslo, N. Deo and J.S. Kowalik. *Discrete Optimization Algorithms (with Pascal Programs)*. Prentice-Hall, 1983.
- [42] C. Thomassen. Even cycles in directed graphs. *Europ. J. Combinatorics*, 6(1985a), 85-89.
- [43] C. Thomassen. The 2-linkage problem for acyclic digraphs. *Discrete Math.*, 55(1985b), 73-87.
- [44] C. Thomassen. Sign-nonsingular matrices and even cycles in directed graphs. *Linear Algebra and its Applications*, 75(1986), 27-41.
- [45] C. Thomassen. The even cycle problem for planar digraphs. *J. of Algorithms*, 15(1993), 61-75.

- [46] C. Thomassen. The even cycle problem for directed graphs. *J. of the American Math. Society*, 5(1992), 217-229.
- [47] L.G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci*, 8(1979), 189-201.
- [48] V.V. Vazirani and M. Yannakakis. Pfaffian orientations, 0-1 permanents, and even cycles in directed graphs. *Discrete Applied Math.*, 25(1989), 179-190.

Índice Remissivo

A

- adjacência
 - matriz de, 3
- adjacente
 - componente, 41
 - vértice, 2
- ancestral
 - de um vértice, 5
- arco, 2
 - bipartição de um, 3
 - de corte, 5
 - desvio de um, 77
 - forte, 20
 - fraco, 20
 - incidente, 2
 - múltiplo, 2
 - orientado, 2
 - subdivisão de um, 3
- aresta, 2
 - bipartição de uma, 3
 - de corte, 5
 - incidente, 2
 - múltipla, 2
 - subdivisão de uma, 3
- articulação
 - ponto de, 5
- árvore, 5
 - com raiz, 5
- aumentação
 - passo de, 91

B

- base

- do espaço dos ciclos, 37
- biconexo
 - componente, 5
 - grafo, 5
- bipartição
 - de um arco, 3
 - de uma aresta, 3
- bloco, 5
- bom
 - caminho, 19
- busca
 - em largura, 12

C

- caminho, 4
 - alternante, 5
 - bom, 19
 - crucial, 83
 - de aumento, 5
 - de substituição, 81
 - de substituição minimal, 82
- ciclos
 - espaço dos, 37
- circuito, 4,
 - induzido, 26
 - cobertura de, 56
 - duplo fraco-ímpar, 40
 - externo, 76
 - facial, 76
 - fraco, 19
 - peso de um, 36
 - sigular, 56
- cobertura

- de circuitos, 56
- identidade, 56
- singular, 56
- completa
 - fase, 91
- componente
 - adjacente, 41
 - biconexo, 5
 - fortemente conexo, 5
- concatenação
 - de dois passeios, 4
- contorno
 - de uma face, 6
- contração
 - de um arco, 3
 - de uma aresta, 3
- curva de Jordan, 6

D

- desvio
 - de um arco, 77
 - estritamente minimal, 77
 - minimal com respeito a um arco, 77
- determinante, 55
- diferença
 - de dois conjuntos, 1
 - simétrica de dois conjuntos, 1
- digrafo, 2
 - acíclico, 4
 - bipartido, 5
 - euleriano, 43
 - fortemente conexo, 5
 - imerso no plano, 6
 - induzido
 - por um conjunto de arcos, 3
 - por um conjunto de vértices, 3
 - k-conexo, 5
 - par, 36
 - planar, 6
 - ponderado, 5
 - simétrico, 40

- simples, 2

E

- emparelhamento
 - de um grafo, 5
 - perfeito, 5
- Entrelaçada
 - Trilha, 103
- euleriano
 - digrafo, 43
 - grafo, 43
- extremo
 - de uma aresta, 2
 - de um arco, 2
 - de um passeio, 3

F

- face, 6
 - contorno de uma, 6
 - externa, 6
 - fronteira de uma, 6
- fase
 - completa, 91
- floresta, 5
- folhas, 5
- forte
 - arco, 20
- fraco
 - arco, 20
 - circuito, 19
- fronteira
 - de uma face, 6

G

- grafo, 2
 - 2-aresta conexo minimal, 41
 - biconexo, 5
 - bipartido, 5
 - conexo, 5
 - desconexo, 5

- euleriano, 43
- imerso no plano, 6
- induzido
 - por um conjunto de arestas, 3
 - por um conjunto de vértices, 3
- k-conexo, 5
- não-separável, 5
- planar, 6
- ponderado, 5
- simples, 2
- subjacente, 2
- grau
 - de entrada, 2
 - de saída, 2
 - de um vértice, 2
 - de uma permutação, 54

H

- hiper-arestas, 70
- hiper-vértices, 70
- hipergrafo, 70
 - condensado, 70
 - quadrado, 70
 - r-coloração de um, 70

I

- identidade
 - cobertura, 56
 - permutação, 54
- ímpar
 - permutação, 55
- induzido,
 - circuito, 26
- interseção
 - de dois conjuntos, 1
 - de dois digrafos, 3
 - de dois grafos, 3

K

- k-circuito duplo, 40

- k-circuito duplo-fraco, 40
- k-regular, 2

L

- laço, 2

M

- mínimo
 - circuito, 4
- matriz
 - adjunta de um digrafo, 65
 - anti-simétrica, 64
 - Pfaffiano de uma, 64

P

- par
 - digrafo, 36
 - permutação, 55
- passeio, 3
 - ímpar, 4
 - comprimento de um, 3
 - concatenação, 4
 - disjuntos nas arestas, 4
 - disjuntos nos arcos, 4
 - disjuntos nos vértices, 4
 - extremos de um, 3
 - fechado, 4
 - fim de um, 3
 - início de um, 3
 - mínimo, 4
 - par, 4
 - seção de um, 4
 - totalmente disjuntos, 4
 - vértices internos de um, 3
- passo de aumentação, 91
- PCamImpar, 7
- PCamImparMin, 9
- PCamPar, 7
- PCamParMin, 9
- PCirDFImp, 41

PCirImpar, 7
 PCirImparMin, 9
 PCirPar, 7
 PCirParMin, 9
 PCirPPar, 36
 PDigrafoPar, 36
 PDoisCam, 102
 PEmpMax, 90
 PEmpPPMax, 112
 PEncCamImpar, 8
 PEncCamPar, 8
 PEncCirImpar, 8
 PEncCirPar, 8
 perfeita
 permutação, 56
 permanente, 54
 permutação, 54
 ímpar, 55
 identidade, 54
 par, 55
 perfeita, 56
 sinal de uma, 55
 PEsqPolya, 55
 PFaseCompleta, 92
 PHiperCond, 71
 PLMatriz, 62
 ponta
 de um arco, 2
 de uma aresta, 2
 ponte, 5
 POrPf, 64
 PPermDet, 60
 Problema
 da L -matriz, 62
 da fase completa, 92
 da orientação Pfaffiana, 64
 da verificação Pfaffiana, 68
 de encontrar um caminho par, 8
 de encontrar um caminho ímpar, 8
 de encontrar um circuito par, 8
 de encontrar um circuito ímpar, 8

do 3-circuito duplo, 53
 do caminho par, 7
 do caminho par mínimo, 9
 do caminho ímpar, 7
 do caminho ímpar mínimo, 9
 do circuito de peso par, 36
 do circuito duplo, 52
 do circuito duplo ímpar, 52
 do circuito duplo-fraco ímpar, 41
 do circuito par, 7
 do circuito par mínimo, 9
 do circuito ímpar, 7
 do circuito ímpar mínimo, 9
 do digrafo par, 36
 do emparelhamento máximo, 90
 do emparelhamento perfeito de peso
 máximo, 112
 do esquema de Polya, 55
 do hipergrafo condensado, 71
 do permanente-determinante, 60
 do triângulo, 111
 dos dois caminhos, 102

PTriângulo, 111

PVerPf, 68

R

redução

aumento de grau, 87
 dois componentes, 86

S

segmento, 79
 especial, 79
 interior, 80
 semi-caminho, 4
 semi-circuito, 4
 semi-passeio, 4
 simétrica
 diferença, 1
 simétrico
 digrafo, 40

simples

digrafo, 2

grafo, 2

subdigrafo, 3

subdivisão

de um arco, 3

de um vértice, 3

de uma aresta, 3

subgrafo, 3

subhipergrafo, 70

subpasseio, 4

T

Trilha

entrelaçada, 103

trilha, 4

U

união

de dois conjuntos, 1

de dois digrafos, 3

de dois grafos, 3

V

vértice, 2

adjacente, 2

de corte, 5

livre, 5

subdivisão de um, 3

vértices

adjacentes, 2

correspondentes, 93

vizinhos, 2