

Projeto Automático de Operadores: Explorando Conhecimentos a Priori

Nina S. Tomita Hirata

TESE APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO GRAU DE DOUTOR
EM
CIÊNCIA DA COMPUTAÇÃO

Área de Concentração : **Ciência da Computação**

Orientador : **Prof. Dr. Junior Barrera**

A autora recebeu auxílio financeiro da OLIVETTI do Brasil,
da CAPES, da Texas A&M University, do CNPq e da FAPESP.

- São Paulo, Outubro de 2000 -

Projeto Automático de Operadores: Explorando Conhecimentos a Priori

Este exemplar corresponde à redação
final da tese devidamente corrigida
e apresentada por Nina S. Tomita Hirata
e aprovada pela Comissão Julgadora.

São Paulo, 27 de outubro de 2000

Banca Examinadora :

- Prof. Dr. Junior Barrera (orientador) - IME/USP
- Prof. Dr. Julio M. Stern - IME/USP
- Prof. Dr. Gerald Jean Francis Banon - DPI/INPE
- Prof. Dr. Luciano da Fontoura Costa - IFSC/USP
- Prof. Dr. Max Henrique Machado Costa - FEEC/UNICAMP

ao Roberto
e aos meus pais

Agradecimentos

Ao meu orientador, Prof. Junior Barrera, pela orientação, amizade e incentivo não só durante o período de doutorado, mas desde o início do mestrado. Igualmente ao Prof. Edward R. Dougherty, com quem tive a oportunidade de trabalhar durante o período de doutorado-sanduíche, na Texas A&M University. Suas maneiras entusiasmadas de realizarem pesquisa foram, muitas vezes, contagiantes.

A todos com quem pude trocar idéias durante o andamento deste trabalho: Prof. Nelson Mascarenhas, Prof. Roberto A. Lotufo, Prof. Eduardo Jordão Neves, Profa. Yoshiko Wakabayashi, Prof. Júlio Stern, Prof. Routo Terada, Prof. Flávio S. C. Silva, colegas do Laboratório de Processamento de Imagens deste instituto, colegas de pós-graduação, pesquisadores que atenderam solicitações por e-mail, e tantas outras pessoas que certamente não estou mencionando.

Aos colegas do CAMDI-Lab e do TCAT, em especial a Barbara Fike e ao Sinan Batman, pela ajuda recebida durante o período na Texas A&M University. Também aos membros da Japanese Graduate Student Association daquela universidade, pelas reuniões gastronômicas que curti bastante.

Aos que tornaram esta jornada possível (sob o ponto de vista financeiro): Olivetti do Brasil nos primeiros meses do doutorado, CAPES (processo BEX 0319/97-2) e Texas A&M University durante o programa de doutorado-sanduíche, CNPq (processo 145158/1998-5) por um curto período após o retorno e, finalmente, à FAPESP (processo 98/14328-6).

Aos membros da banca examinadora, pela revisão do texto preliminar e pelas sugestões de melhoramentos.

A compreensão e incentivo de amigos e familiares, especialmente dos meus pais e do Roberto, fizeram valer a pena o esforço para concluir este trabalho.

A todos, o meu sincero agradecimento !

Resumo

A morfologia matemática vem sendo largamente utilizada para processamento e análise de imagens digitais. O projeto de operadores morfológicos é em geral realizado de forma heurística. Devido a dificuldade inerente a este procedimento, técnicas de projeto automático são de grande importância e interesse. Várias abordagens neste sentido vem sendo propostas, dentre elas técnicas que projetam operadores a partir de exemplos de treinamento (obtidos de amostras de imagens observadas-ideais) que representam de forma simples a transformação desejada pelo usuário. Tomando uma técnica de projeto de operadores baseada no modelo de aprendizado PAC (do inglês, “Probably Approximately Correct”) como ponto de partida, investigamos de forma geral algumas das limitações dessas abordagens. Com base nessa investigação, estudamos o projeto de W -operadores, colocando ênfase sobre questões relacionadas com a precisão de operadores projetados a partir de uma quantidade limitada de exemplos de treinamento. Os frutos deste estudo, apresentados neste trabalho, são técnicas que exploram conhecimentos sobre o problema que desejamos resolver para projetar operadores mais precisos e algoritmos eficientes para implementar as mesmas. Soluções para problemas reais de processamento de imagens ilustram a aplicação das técnicas propostas.

Abstract

Mathematical morphology is being widely used in image processing and analysis. Designing morphological operators is usually done by heuristic methods. However, due to the inherent difficulty of such procedures, automatic design techniques are of increasing interest. In recent years, several approaches for the automatic design of morphological operators have been proposed. Some of them are based on learning from training examples (sampled from observed-ideal pairs of images representing the desired image processing mapping). Starting from a technique based on PAC (Probably Approximately Correct) learning model, we investigate some limitations of those approaches. From this investigation, we study the design of W -operators emphasizing questions related with precision of operators designed from a limited number of training examples. The results of this study, presented in this work, are techniques which exploit knowledge (about the image processing problem being solved) in order to design more accurate operators and efficient algorithms for implementing them. Solutions for some real image processing problems are given to illustrate the application of the proposed techniques.

Índice

1	Introdução	1
1.1	Projeto Automático de Operadores Morfológicos	2
1.2	Limitações das Técnicas Existentes	3
1.3	Propostas e Organização da Tese	5
2	Operadores Morfológicos de Imagens Binárias	9
2.1	Notações e Definições Básicas	9
2.1.1	Imagens Binárias e Conjuntos	10
2.1.2	Operadores de Imagens	10
2.2	W-operadores e Sua Representação	12
2.2.1	Algumas Famílias de W-operadores	14
2.2.2	Núcleo e Base de W-operadores	14
2.2.3	Representações em Termos de Núcleo e de Base	15
2.3	Funções Booleanas e sua Representação	17
2.3.1	Representação Tabular	17
2.3.2	Expressões Booleanas	18
2.4	Equivalência entre Representações Morfológicas e Booleanas	21
2.4.1	Conversão Intervalo-Expressão	21
2.4.2	Conversão Expressão-Intervalo	22
2.4.3	Representações Associadas às Partições de $\mathcal{P}(W)$	22
2.4.4	Resumo	24
2.5	Comentários	24
3	Projeto de W-Operadores	25
3.1	Descrição do Problema	25

3.2	O Problema Visto no Contexto de Aprendizado Computacional	26
3.3	Avaliação de W-operadores	27
3.3.1	Comparações Simples	28
3.3.2	W-operadores Estatisticamente Ótimos	28
3.3.3	Subotimalidade	32
3.4	Projeto baseado no modelo de aprendizado PAC	32
3.5	Limitações	34
3.6	Como Contornar as Limitações	36
3.7	Comentários	38
4	O Algoritmo ISI	39
4.1	Minimização de Funções Booleanas	39
4.1.1	Algoritmos Clássicos	41
4.1.2	O Algoritmo ISI	41
4.2	Minimização na Existência de “Don’t Cares”	48
4.2.1	Variantes do Algoritmo ISI	49
4.2.2	ISI modificado para manipular “don’t cares”	50
4.3	Algoritmo ISI como um Algoritmo de Aprendizado	53
4.4	Paralelização do ISI	54
4.5	Comentários	59
5	Projeto Multi-estágio de W-operadores	61
5.1	Considerações Iniciais	62
5.1.1	Detalhes do Projeto	63
5.1.2	Detalhes dos Experimentos	63
5.2	Número Crescente de Iterações	66
5.3	Número Crescente de Dados de Treinamento	67
5.4	Iterações sobre Diferentes Janelas	69
5.4.1	Exemplos	75
5.5	Diferentes Formas de Treinamento	78
5.5.1	Subdividindo os Dados de Treinamento	78
5.5.2	Variando as Janelas de um Estágio para Outro	82
5.6	Análise e Comentários	84

6	O Uso de Operadores Projetados a Priori	85
6.1	Descrição da Técnica Proposta	85
6.2	Confiança sobre um Operador a Priori	86
6.3	Procedimento Prático	87
6.3.1	Escolha de uma Janela	87
6.3.2	Algoritmo	88
6.4	Resultados Experimentais	89
6.5	Comentários	91
7	Projeto de Operadores Crescentes Ótimos	93
7.1	Descrição do Problema e Algumas Técnicas de Solução	94
7.2	O Método de Chaveamentos	96
7.3	Um Novo Algoritmo Baseado no Método de Chaveamentos	98
7.3.1	Reestruturação do Problema como um Problema de Partição	98
7.3.2	O Problema da Partição Ótima	102
7.3.3	Conjuntos Viáveis	107
7.3.4	O Algoritmo	109
7.3.5	Como Encontrar Conjuntos Viáveis	114
7.3.6	Busca Relaxada	116
7.3.7	Limitantes para o Aumento de Erro	119
7.3.8	Resultados Experimentais	120
7.4	Distribuições a Priori para as Probabilidades Condicionais	125
7.4.1	Custos de Chaveamento a Partir de Distribuições a Priori	126
7.4.2	Resultados Experimentais	127
7.5	Comentários	130
8	Filtros “Stack”	133
8.1	Definições e Propriedades	133
8.2	Filtros W-Stack	136
8.3	Representação de Filtros W-Stack	140
8.4	Filtros W-Stack Ótimos	142
8.5	Projeto de Filtros W-Stack	144
8.6	Alguns Resultados Experimentais	146

8.6.1	Filtragem de Ruído do Tipo Impulso	146
8.6.2	Filtragem de Ruído <i>Speckle</i>	146
8.7	Comentários	149
9	Exemplos de Aplicações	151
9.1	Filtragem de Ruído Tipo Impulso	151
9.2	Reconhecimento de Padrões em Diagramas	155
9.3	Reconhecimento de Textura	158
9.4	Reconhecimento de Caracteres	162
9.5	Segmentação de Texto	164
10	Conclusões	169
10.1	Resumo das Contribuições	169
10.2	Comentários Finais	171
A	Resultados (Imagens) Adicionais das Aplicações	173
A.1	Reconhecimento de Padrões em Diagramas	173
A.2	Reconhecimento de Caracteres	174
A.3	Segmentação de Texto	176
B	Publicações Decorrentes da Pesquisa	185

Lista de Figuras

1.1	Sistema para geração automática de operadores de imagens.	2
1.2	Exemplos de descrição de processamento de imagens por meio de pares de imagens observadas-ideais : (a) Reconhecimento de textura. (b) Filtragem de ruído.	3
1.3	Erro de operadores projetados em espaços de tamanho diferentes.	4
1.4	Complexidade de treinamento para operadores projetados em espaços de tamanho diferentes.	5
2.1	Representação gráfica da função $\psi(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$, onde $\psi\langle 1 \rangle = \{011, 101, 110, 111\}$ e $\psi\langle 0 \rangle = \{000, 001, 010, 100\}$	18
2.2	Representação de ψ através do diagrama de Hasse.	19
2.3	Os intervalos maximais de $On(\psi)$ (esquerda) e os intervalos da expressão minimal (direita).	20
2.4	Esquema da expansão de Shannon de uma função Booleana.	21
2.5	Decomposição do operador $\Psi : \Psi = \Psi_1 \cup \Psi_2 = \Psi_3 \cap \Psi_4$ ($\Psi_2 = \Psi_5 \cap \Psi_3$ e $\Psi_4 = \Psi_5 \cup \Psi_1$), onde Ψ_1 é anti-extensivo, Ψ_2 é anti-extensivo complementar, Ψ_3 é extensivo, Ψ_4 é extensivo complementar, e Ψ_5 é o complemento.	23
3.1	Procedimento computacional baseado no modelo PAC para o aprendizado de W-operadores.	33
3.2	Erro de operadores projetados em espaço de tamanhos diferentes, para diferentes números de exemplos de treinamento.	36
3.3	Os candidatos são todos os operadores.	36
3.4	Os candidatos são os operadores no subespaço.	37
3.5	Os candidatos são os operadores consistentes com os exemplos de treinamento.	37
3.6	Os candidatos são os operadores no subespaço que são consistentes com os exemplos de treinamento.	38
4.1	Implicantes primos e cobertura mínima.	40

4.2	Minimização pelo algoritmo de Quine-McCluskey. (a) Os 0-cubos 000, 001, 100, 101, 110. (b) Os 1-cubos obtidos a partir dos 0-cubos em (a). (c) O 2-cubo obtido a partir dos 1-cubos em (b). (d) Os implicantes primos $X0X$ e $1X0$	42
4.3	Extração do elemento 011 do 3-cubo resulta nos 2-cubos $X00$, $1XX$, $X0X$	44
4.4	Cálculo de implicantes primos pelo algoritmo ISI. $\psi\langle 0 \rangle = \{010, 011, 111\}$ e $\psi\langle 1 \rangle = \{000, 001, 100, 101, 110\}$	46
4.5	ISI com diferentes períodos ($n = 15$, $ \psi\langle 1 \rangle = 9110$ e $ \psi\langle 0 \rangle = 14538$).	51
4.6	ISI com diferentes períodos ($n = 17$, $ \psi\langle 1 \rangle = 1609$ e $ \psi\langle 0 \rangle = 11410$).	51
4.7	ISI com diferentes períodos ($n = 17$, $ \psi\langle 1 \rangle = 8517$ e $ \psi\langle 0 \rangle = 12251$).	52
4.8	ISI com diferentes períodos ($n = 25$, $ \psi\langle 1 \rangle = 1340$ e $ \psi\langle 0 \rangle = 8126$).	52
4.9	Erro MAE para ISI com diferentes períodos ($n = 15$, $ \psi\langle 1 \rangle = 9110$ e $ \psi\langle 0 \rangle = 14538$).	54
4.10	Erro MAE para ISI com diferentes períodos ($n = 17$, $ \psi\langle 1 \rangle = 1609$ e $ \psi\langle 0 \rangle = 11410$).	54
4.11	Erro MAE para ISI com diferentes períodos ($n = 17$, $ \psi\langle 1 \rangle = 8517$ e $ \psi\langle 0 \rangle = 12251$).	55
4.12	Erro MAE para ISI com diferentes períodos ($n = 25$, $ \psi\langle 1 \rangle = 1340$ e $ \psi\langle 0 \rangle = 8126$).	55
4.13	ISI \times ISI paralelo ($n = 17$, $ \psi\langle 1 \rangle = 10351$ e $ \psi\langle 0 \rangle = 16107$).	57
4.14	ISI \times ISI paralelo ($n = 49$, $ \psi\langle 1 \rangle = 9924$ e $ \psi\langle 0 \rangle = 122284$).	58
4.15	ISI \times ISI paralelo : tempo de processamento variando $ \psi\langle 1 \rangle \cup \psi\langle 0 \rangle $	59
5.1	Composição de operadores.	62
5.2	Duas possibilidades para projeto multi-estágio.	64
5.3	Imagens com ruído do tipo sal-e-pimenta.	65
5.4	Modelo Booleano.	66
5.5	Reconhecimento de letra s (livro 1).	67
5.6	Reconhecimento de letra a (livro 2).	67
5.7	Evolução das curvas de erro ao longo das iterações. Janela de iteração e quantidade de imagens de treinamento fixas ao longo das iterações.	68
5.8	Evolução das curvas de erro ao longo das iterações, para diferentes quantidades de exemplos de treinamento (modelo Booleano – $D_a = 0.02$, $D_s = 0.02$).	69
5.9	Evolução das curvas de erro ao longo das iterações, para diferentes quantidades de exemplos de treinamento (reconhecimento de letra s).	70
5.10	Filtragem de ruído sal-e-pimenta : iterações sobre $W_5 \times$ iterações sobre W_{13}	71
5.11	Resultados de 1 a 4-iteraões (janela W_5).	72
5.12	Resultados de 1 e 2-iteraões (janela W_{13}).	72
5.13	Filtragem de ruído Booleano: iterações sobre $W_{3 \times 3} \times$ iterações sobre $W_{5 \times 5}$	73
5.14	Reconhecimento de caractere: iterações sobre $W_{3 \times 3} \times$ iterações sobre $W_{5 \times 5}$	74

5.15	Diferentes janelas de iteração para o reconhecimento da letra <i>s</i>	76
5.16	Resultados para reconhecimento de caractere “s”. Duas iterações sobre a janela $W_{7 \times 7}$, usando 6 pares de imagens de treinamento.	76
5.17	Diferentes janelas de iteração – modelo Booleano ($D_a = D_s = 0.02$).	77
5.18	Modelo Booleano ($D_a = 0.06, D_s = 0.06$). Imagem de teste e resultados de 1 a 6 iteraões.	77
5.19	Erro sobre imagens de teste e de treinamento.	78
5.20	Desempenho sobre as imagens de teste ($N = 4$).	79
5.21	Desempenho sobre as imagens de teste ($N = 6$).	79
5.22	Desempenho sobre as imagens de teste ($N = 11$).	80
5.23	Desempenho sobre as imagens de teste ($N = 15$).	80
5.24	Desempenho sobre as imagens de teste ($N = 20$).	81
5.25	Diferentes janelas de uma iteração para outra.	83
6.1	Imagem com ruídos pontuais e respectiva imagem ideal.	89
6.2	Extração de borda de uma imagem ruidosa.	90
6.3	Reconhecimento de segmento de linhas a certo grau de inclinação.	92
7.1	Grafo representando o espaço de operadores crescentes.	95
7.2	O método de chaveamentos.	97
7.3	Um operador crescente produzido por chaveamentos.	99
7.4	Chaveamentos que resultam em um operador não-crescente.	100
7.5	Conjuntos de Chaveamento e suas respectivas partições.	101
7.6	Ilustração para a proposição 7.2.	103
7.7	Custo de chaveamento e peso.	105
7.8	Ilustração para a proposição 7.7.	105
7.9	Dinâmica do particionamento.	107
7.10	Um conjunto violador.	107
7.11	Ilustração para as definições 7.8 e 7.9.	108
7.12	Aplicação do algoritmo OVP - exemplo 1	111
7.13	Aplicação do algoritmo OVP - exemplo 2.	112
7.13	Aplicação do algoritmo OVP - exemplo 2 (continuação).	113
7.14	Resultado do algoritmo relaxado não é ótimo.	117
7.15	Resultado do algoritmo relaxado é ótimo.	117

7.16	Exemplo de uma porção irreduzível	118
7.17	Porção irreduzível é processada corretamente.	119
7.18	Porção irreduzível é processado incorretamente.	119
7.19	Grupo A : 15% de ruído sal-e-pimenta	120
7.20	Grupo B: Ruído de borda, diferentes densidades.	120
7.21	Grupo C: Modelo Booleano com grãos quadrados e subconjuntos do quadrado 3×3 como ruído.	121
7.22	Grupo D : Retângulos com ruído de borda, com diferentes densidades.	121
7.23	Distribuição de δ (parâmetro da intensidade de ruído).	128
7.24	Exemplos de ruído de borda.	128
7.25	Curva de erro para $\delta = 45$	129
7.26	Curva de erro para $\delta = 35$	130
7.27	Curva de erro para $\delta = 25$	131
7.28	Curva de erro para $\delta = 21$	131
8.1	Decomposição threshold.	134
8.2	Propriedade stack.	138
8.3	Isomorfismo entre funções Booleanas positivas e filtros stack.	139
8.4	Imagens de treinamento.	147
8.5	Filtragem de ruído impulso.	147
8.6	Filtragem de ruído impulso - robustez.	148
8.7	Filtragem de ruído speckle (intensidade com 1 visada).	148
8.8	Filtragem de ruído speckle (amplitude com 4 visadas).	150
9.1	Filtragem de ruído: exemplo de imagens de treinamento.	151
9.2	Imagem de teste, resultado, e diferença simétrica (erro 335 pixels).	152
9.3	Resultado do filtro de ordem e diferença simétrica (erro 324 pixels).	153
9.4	Resultado da combinação, diferença simétrica (erro 199 pixels).	153
9.5	Imagem de teste, resultado do operador projetado (erro 303 pixels), resultado do filtro de ordem (erro 258 pixels), e união dos dois resultados (erro 175 pixels), respectivamente.	154
9.6	Reconhecimento de padrões: exemplo de imagem de treinamento.	155
9.7	Reconhecimento de padrão em diagramas: imagem (teste) observada, resultado da primeira iteração e resultado da segunda iteração, respectivamente.	156
9.8	Reconhecimento de padrão em diagramas: resultado do operador e filtro de ordem 12 por uma janela circular de raio 9, respectivamente.	157

9.9	Reconhecimento de padrão em diagramas: resultado da reconstrução (resultado final) e o mesmo sobreposto à imagem original, respectivamente.	157
9.10	Imagem de treinamento para reconhecimento de textura.	158
9.11	Outra imagem de treinamento para reconhecimento de textura.	159
9.12	Reconhecimento de textura: imagem teste 1, resultado da primeira e segunda iterações, respectivamente.	160
9.13	Reconhecimento de textura: imagem teste 2, resultado da primeira e segunda iterações, respectivamente.	161
9.14	Reconhecimento do caractere “a”: resultado da primeira e segunda iterações, respectivamente.	163
9.15	Reconhecimento do caractere “a”: resultado do operador projetado.	163
9.16	Um par de imagem de treinamento para segmentação de texto.	164
9.17	Segmentação de texto : (a) imagem (teste) observada e (b) resultado do operador projetado.	165
9.18	Efeitos do filtro de pós-processamento para segmentação de texto: resultado do (a) operador, (b) filtro de ordem, (c) fechamento vertical, (d) preenche buracos, (e) abertura por área e (f) resultado final sobreposto à imagem observada.	166
9.19	Efeitos do filtro para geração de marcadores para a segmentação de texto.	167
9.20	Segmentação de texto : imagem observada, resultado do operador e resultado final, respectivamente.	168

Lista de Tabelas

3.1	Probabilidades e três operadores.	31
4.1	Comparação entre os algoritmos QM e ISI.	48
4.2	ISI com e sem agrupamento de elementos.	53
5.1	Iterações equivalentes	75
6.1	Comparação entre uso de priori e não uso.	89
6.2	Comparação entre uso de priori e não uso.	91
7.1	Desempenho do algoritmo OVP: janela $W_{3 \times 3}$	122
7.2	Desempenho do Algoritmo OVP: janela W_{13}	122
7.3	Aumento de Erro: fator de relaxamento $k = 1$ e $k = 3$	123
7.4	Tempo de processamento do algoritmo OVP.	124
7.5	Algoritmo OVP contra algoritmo baseado na representação MAE.	125
9.1	Filtragem de ruído sal-e-pimenta: detalhes do treinamento.	152
9.2	Reconhecimento de somadores/multiplicadores: detalhes do treinamento.	155
9.3	Reconhecimento de textura: detalhes do treinamento.	159
9.4	Reconhecimento de caracteres “a”: detalhes do treinamento.	162
9.5	Segmentação de Texto: detalhes do treinamento.	164

Lista de Janelas



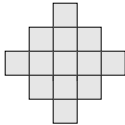
$W_{1 \times 3}$



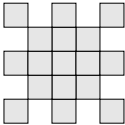
W_5



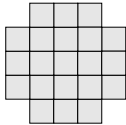
$W_{3 \times 3}$



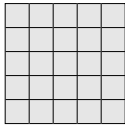
W_{13}



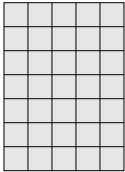
W_{17}



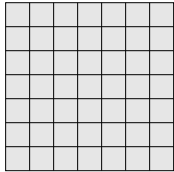
W_{21}



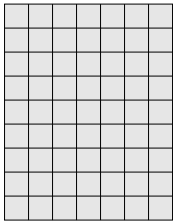
$W_{5 \times 5}$



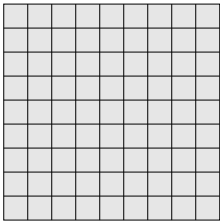
$W_{7 \times 5}$



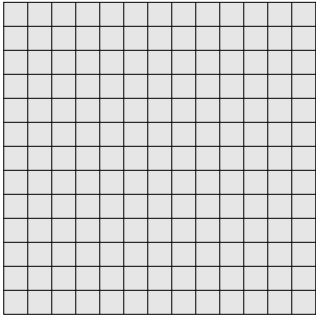
$W_{7 \times 7}$



$W_{9 \times 7}$



$W_{9 \times 9}$



$W_{13 \times 13}$

Capítulo 1

Introdução

Com o avanço da tecnologia, que vem facilitando a aquisição de imagens digitais e tornando possível a manipulação de grande quantidade de dados, procedimentos computacionais que visam auxiliar o processamento e análise de imagens digitais [64, 134, 20, 117] vem sendo cada vez mais utilizados. Estes são importantes para auxiliar diagnósticos médicos, na automação industrial, na análise de safras agrícolas, na previsão de tempo, no processamento de documentos e outras aplicações.

Uma técnica abrangente e bastante utilizada para processamento e análise de imagens é a *morfologia matemática* [118, 147]. Sob o ponto de vista formal [148, 10], ela estuda mapeamentos (ou operadores) entre *reticulados completos* [23]. Em particular, ela pode ser utilizada para modelar mapeamentos entre imagens [147, 75] pois estas podem ser entendidas como elementos de um reticulado completo. No contexto de processamento de imagens, esses mapeamentos são geralmente denominados *operadores morfológicos* (de imagens). Algumas discussões sobre a adequação destes operadores para processamento e análise de imagens podem ser encontradas, por exemplo, em [147, 138].

A morfologia matemática, originada na “École Nationale Supérieure des Mines de Paris” em meados da década de sessenta, ficou conhecida a partir da publicação dos livros de Matheron [118] e de Serra [147]. Desde então, vários trabalhos foram publicados sobre este assunto, entre eles os livros de Serra [148], de Heijmans [75] e outros tais como [41, 11, 45, 151]. Estas e outras publicações comprovam o potencial da morfologia matemática como ferramenta para resolver problemas de processamento de imagens, tais como segmentação [121, 21, 22, 163, 35, 85], processamento de imagens médicas [66, 162], processamento de imagens de documentos [17, 19, 140, 112], filtragem de ruído [141], processamento de imagens coloridas [29, 89], entre outros. Existem também vários trabalhos sobre implementações eficientes dos operadores morfológicos em “hardware” ou “software”, tais como [12, 25, 2, 94].

A tarefa de projetar (descrever) procedimentos computacionais que realizam transformações adequadas das imagens para que as informações de interesse possam ser extraídas não é trivial. Ela requer do projetista conhecimentos específicos sobre as técnicas de processamento de imagens, além de muita experiência e criatividade. Esta dificuldade tem motivado esforços que buscam automatizar o projeto (ou parte do projeto) de operadores de imagens. No contexto de morfologia matemática, estes esforços começaram a ser observados no final da década de oitenta, à medida que sua utilização começou a ser difundida.

1.1 Projeto Automático de Operadores Morfológicos

Por projeto automático de operadores de imagens entendemos qualquer procedimento computacional que, a partir da descrição de uma transformação de imagens, gera uma especificação de um operador que realiza a transformação descrita (veja esquema na figura 1.1). Geralmente, tais procedimentos trabalham com um espaço fixo de operadores e são baseados em algoritmos que buscam, no espaço de operadores considerado, aquele que melhor se adequa à descrição dada.

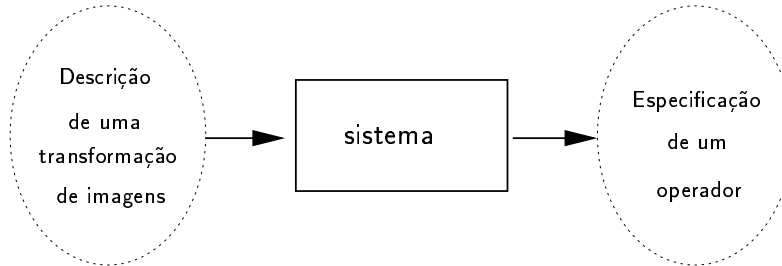


Figura 1.1: Sistema para geração automática de operadores de imagens.

As várias tentativas para automatizar o projeto de operadores morfológicos podem ser divididas em três grupos de abordagens: as baseadas em técnicas de inteligência artificial, as baseadas em modelagem de imagens e as baseadas em técnicas de indução a partir de exemplos.

A aplicação de técnicas de inteligência artificial, tais como sistemas especialistas e prova automática de teoremas, para automatizar a geração de procedimentos eficientes para análise de imagens é abordada, por exemplo, em [145, 164, 93, 92].

Alguns tipos de imagens (por exemplo, certos tipos de ruído, ou de textura) podem ser modeladas e simuladas. Abordagens baseadas em modelagem de imagens estudam a caracterização formal das imagens e a sua manipulação analítica. O problema de projetar operadores é reduzido então ao problema de estimar os parâmetros corretos do modelo [149, 56, 44, 146].

Nas abordagens baseadas em indução, uma transformação de imagens é expressa através de pares de imagens, onde cada par é formado por uma imagem anterior e outra posterior ao processamento desejado. Dois pares de imagens que descrevem processamentos diferentes são mostrados na figura 1.2. O primeiro par (fig. 1.2a) ilustra um processamento que consiste em extrair regiões da imagem que correspondem a um padrão ou textura específica. O segundo par (fig. 1.2b) expressa um processamento cujo efeito é a filtragem de ruído. A imagem a ser processada é denominada *imagem observada*, enquanto a imagem desejada como o resultado do processamento é denominada *imagem ideal*. O operador é projetado a partir de exemplos obtidos dos pares de imagens observadas-ideais.

Várias técnicas para projeto de operadores a partir de amostras de imagens observadas-ideais vem sendo propostas. Técnicas no contexto de estimação estatística consideram as imagens como realizações de processos aleatórios [43] e os operadores a serem projetados como estimadores estatísticos. Mais especificamente, se denotamos por \mathbf{S} o processo associado às imagens observadas e por \mathbf{I} o processo associado às imagens ideais, projetar um operador consiste em escolhermos um operador Ψ tal que $\Psi(\mathbf{S})$ seja próximo de \mathbf{I} segundo alguma medida estatística de proximidade. Um operador é considerado estatisticamente ótimo em um espaço de operadores, com relação a uma medida de erro, se possui o menor erro entre todos os demais operadores no mesmo espaço. Em geral, a medida estatística utilizada é o erro absoluto médio (MAE) ou o erro quadrático médio (MSE).

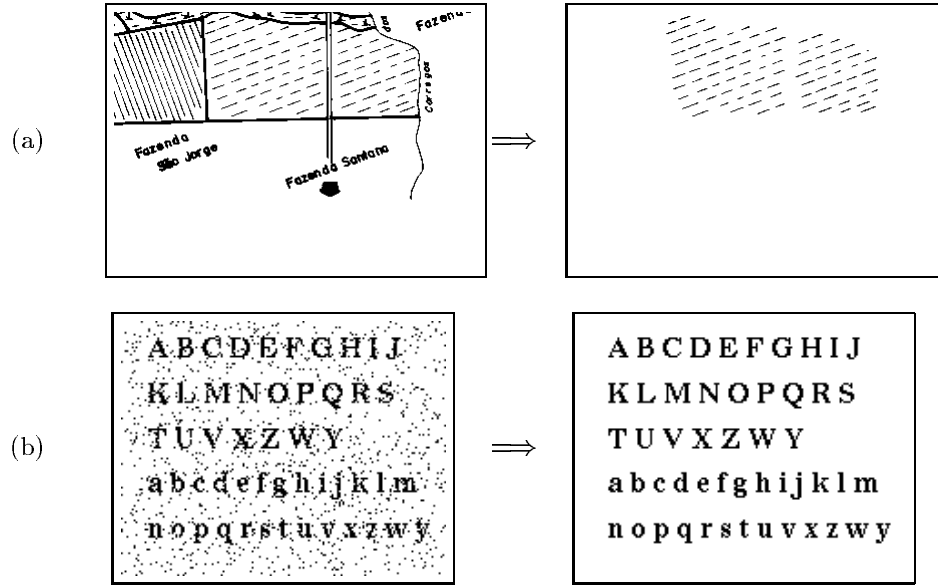


Figura 1.2: Exemplos de descrição de processamento de imagens por meio de pares de imagens observadas-ideais : (a) Reconhecimento de textura. (b) Filtragem de ruído.

Os primeiros trabalhos sobre projeto de operadores morfológicos estatisticamente ótimos são creditados a E. R. Dougherty [39, 40]. A formulação inicial é restrita à classe de operadores crescentes, porém é generalizada para a classe de operadores não necessariamente crescentes logo em seguida [52, 13]. Vários trabalhos continuaram a ser publicados na mesma linha de pesquisa, abordando temas como projeto de operadores crescentes [119, 55, 69], precisão dos operadores projetados [53], projeto de operadores não necessariamente crescentes [52], modelagem do problema como um problema de aprendizado computacional [15], projeto multi-estágio de operadores [55, 144], técnicas de multi-resolução [49, 87, 95], o uso de conhecimentos a priori e o estudo de outros aspectos relacionados ao projeto de operadores [47, 14, 42, 14, 47].

Outras técnicas de projeto de operadores a partir de imagens observadas-ideais utilizam técnicas adaptativas tais como algoritmos genéticos [70, 169, 125] ou redes neurais [143, 144, 166, 152, 153].

Ainda dentro desta abordagem, observam-se trabalhos sobre projeto de filtros de mediana [60], filtros de ordem (“rank-order filters”) e “stack filters” [165, 115, 33, 34, 107, 155], que são casos especiais de operadores morfológicos. Estes filtros foram introduzidos inicialmente independente do contexto de morfologia matemática e são citados na literatura como filtros não-lineares [1] em contraposição aos filtros clássicos lineares. Eles foram inseridos no contexto de morfologia matemática por Maragos e Schafer [114, 115].

1.2 Limitações das Técnicas Existentes

Ao analisarmos os trabalhos mencionados, podemos observar que algumas técnicas utilizadas são *dependentes de contexto* enquanto outras são *independentes de contexto*. As técnicas dependentes de contexto exploram, ou levam em consideração na sua formulação, características específicas do problema que desejamos resolver, enquanto as independentes não. Além disso, essas técnicas apresentam diferentes graus de restrição em função do grau de dependência de contexto e do tipo de restrição

imposta sobre o espaço de operadores.

As técnicas dependentes de contexto são naturalmente restritivas pois tem aplicação restrita por construção, além de serem difíceis de modelar. Para entender as limitações das técnicas independentes de contexto, que são mais genéricas, consideramos uma técnica baseada no modelo de aprendizado PAC (Probably Approximately Correct), proposta originalmente em [157]. Devemos salientar que essa técnica constitui o ponto de partida de nosso estudo.

O modelo PAC, introduzido por Valiant [160] e aprofundado e estendido por outros [97, 73, 98], é um assunto bastante estudado na área de aprendizado computacional [122, 105]. Este modelo estuda vários aspectos relacionados ao aprendizado de conceitos (modelados por funções Booleanas) a partir de exemplos. A técnica baseada neste modelo e mencionada acima considera o problema de projeto de operadores invariantes por translação e localmente definidos por uma janela W , denominados W -operadores. A consideração desta classe de operadores é bastante natural uma vez que os W -operadores são equivalentes às funções Booleanas de $|W|$ variáveis (onde $|W|$ é o tamanho da janela W).

De acordo com o modelo PAC, se os operadores forem gerados de forma a serem consistentes com os exemplos de treinamento, então operadores com qualquer precisão especificada podem ser obtidos, desde que uma quantidade suficiente de dados sejam utilizados. No entanto, na prática, a quantidade de exemplos de treinamento é limitada. Quais são as consequências desse fato ?

Na figura 1.3 comparamos o desempenho de operadores projetados pela técnica baseada no modelo

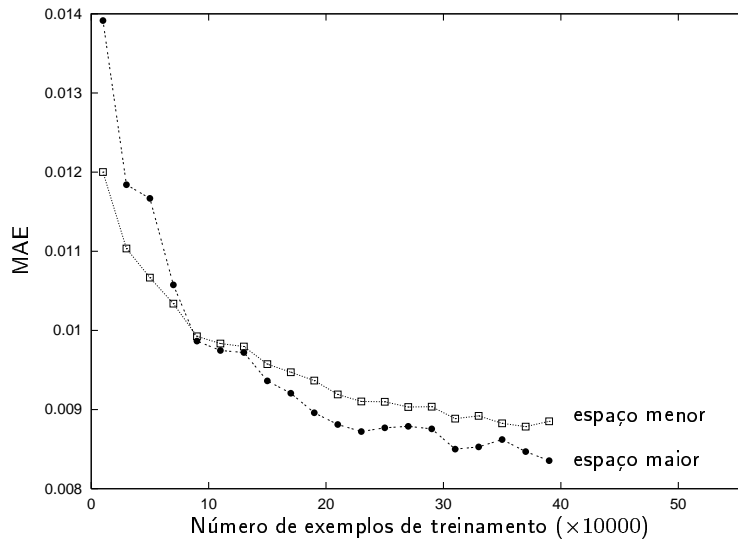


Figura 1.3: Erro de operadores projetados em espaços de tamanho diferentes.

PAC em dois espaços diferentes para diferentes quantidades de exemplos de treinamento. Os espaços considerados são os espaços definidos por duas janelas de tamanhos diferentes, uma menor contida em uma outra maior. A janela menor define o espaço menor e a maior o espaço maior. O eixo das abscissas indica a quantidade de exemplos utilizada para projetar os operadores e o eixo das ordenadas indica o erro MAE do operador projetado. O erro MAE foi medido sobre um conjunto de imagens disjunto das utilizadas para projetar o operador. Uma explicação mais detalhada do gráfico é apresentada no capítulo 3.

Podemos observar que, para uma pequena quantidade de exemplos, o operador projetado no

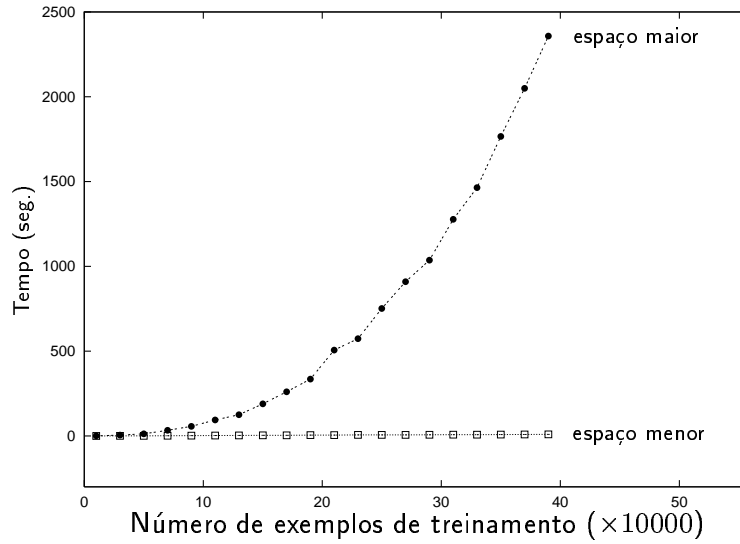


Figura 1.4: Complexidade de treinamento para operadores projetados em espaços de tamanho diferentes.

espaço menor possui erro MAE menor do que o do operador projetado no espaço maior. À medida que a quantidade de exemplos aumenta, a diferença entre ambos diminui, até que a situação se inverte.

Um outro aspecto do projeto é o tempo de processamento. Na figura 1.4 mostramos o tempo gasto para projetar operadores nos dois espaços. O tempo de projeto no espaço maior é consideravelmente maior que no espaço menor, principalmente quando a quantidade de exemplos de treinamento é grande.

Esta análise mostra que o projeto de operadores em espaços grandes pode esbarrar em problemas de estimação estatística (imprecisão dos operadores projetados) e complexidade de tempo. Este problema, que decorre da razão entre o “tamanho de amostra” e o “tamanho do espaço de operadores”, não depende da particular técnica utilizada. De fato, este é um problema comum a todas as técnicas que buscam extrair informações a partir de dados, imagens ou não [38, 62, 132].

1.3 Propostas e Organização da Tese

Neste trabalho procuramos focalizar os esforços para entender as limitações da técnica baseada no modelo PAC e, principalmente, para encontrar formas de contorná-las. Em seu conjunto, procuramos fornecer uma visão geral sobre o problema de projeto de operadores morfológicos, abordando vários aspectos relacionados principalmente com a questão da precisão dos operadores projetados. O trabalho é restrito ao projeto de operadores morfológicos binários.

Uma solução imediata para melhorar a precisão dos operadores projetados seria o aumento da quantidade de dados de treinamento. No entanto, conforme já mencionamos, esta quantidade é limitada em geral e, por outro lado, o aumento de dados implica no aumento do tempo de projeto. Outra solução seria o controle do tamanho do espaço de operadores através da redução do tamanho da janela W . Neste caso, o operador projetado pode não ser satisfatório pelo simples fato de que o operador ótimo nesse espaço não é satisfatório. Ou seja, muitas vezes existe a necessidade de

projetarmos operadores em espaços grandes a partir de uma quantidade fixa de dados.

Uma forma natural para melhorar a precisão dos operadores projetados consiste em incorporarmos conhecimentos sobre o problema que desejamos resolver no processo de projeto de operadores. Os conhecimentos sobre um problema podem refletir em conhecimentos sobre o operador desejado bem como sobre as imagens envolvidas. Por exemplo, problemas que consistem em obter marcadores para os objetos de interesse (i.e., casos nos quais o resultado ideal é um subconjunto da imagem observada) podem ser resolvidos por operadores anti-extensivos. Da mesma forma, conhecimentos sobre as imagens podem permitir sua modelagem e, conseqüentemente, a manipulação analítica [150] ou estimativas mais precisas [47].

Esta tese é organizada da seguinte forma: os três primeiros capítulos (incluindo esta introdução) fornecem uma visão geral do problema como um todo, juntamente com um embasamento teórico e referências bibliográficas; cada um dos capítulos, do quarto ao oitavo, aborda um tema específico, que são contribuições no sentido de contornar o problema da precisão e da complexidade de tempo.

- No capítulo 2 (**Operadores Morfológicos de Imagens Binárias**) apresentamos notações, noções e conceitos básicos a serem utilizados no restante do texto. Definimos e revemos as noções de imagens, operadores morfológicos de imagens, representação canônica dos operadores, algumas propriedades algébricas, composição de W-operadores, diversas formas de representação de W-operadores e a equivalência entre W-operadores e funções Booleanas. A representação de W-operadores é um tópico importante para o desenvolvimento dos capítulos subseqüentes. Introduzimos duas novas famílias de operadores, os anti-extensivo complementares e os extensivo complementares.
- No capítulo 3 (**Projeto de W-Operadores**) apresentamos uma visão geral sobre projeto de operadores morfológicos no contexto de aprendizado a partir de exemplos. Revemos o projeto baseado em PAC-learning e analisamos suas limitações formalmente. Por último, apresentamos considerações sobre quais devem ser as preocupações das técnicas que objetivam contornar as limitações existentes. Estas são discutidas sob uma perspectiva bastante ampla.

As técnicas de projeto de operadores devem estar baseadas em algoritmos eficientes. Uma das contribuições desta tese é uma série de melhorias propostas para o algoritmo ISI (do inglês “Incremental Splitting of Intervals”), proposto recentemente [157] para o aprendizado de funções que caracterizam W-operadores.

- No capítulo 4 (**O Algoritmo ISI**) apresentamos uma generalização do algoritmo (extração de intervalos em vez de pontos isolados), uma prova formal da corretude do algoritmo e heurísticas para torná-lo computacionalmente eficiente, com possibilidade para paralelização. Os algoritmos modificados conforme as heurísticas propostas são avaliados e comparados quanto ao tempo de processamento, tamanho de representação, e precisão da função gerada (este último no contexto de processamento de imagens).

O problema de projetar um operador pode ser substituído pelo problema de projetar vários operadores mais simples e que, compostos de forma adequada, efetuam o mesmo processamento. Este tipo de projeto é o que denominamos *projeto multi-estágio*.

- No capítulo 5 (**Projeto Multi-estágio de W-operadores**) estudamos um particular caso desta abordagem: o projeto de operadores sobre janelas grandes através da composição de operadores

projetados sobre janelas menores. A idéia básica consiste em projetar um operador a partir do resultado de outro, previamente projetado, repetindo-se este processo até que o decréscimo de erro entre um estágio e outro torne-se desprezível. Neste capítulo investigamos vários aspectos práticos do método, inclusive a melhor forma de utilização das imagens de treinamento. Este estudo é uma extensão e aprofundamento do trabalho em [55].

- No capítulo 6 (**Uso de Operadores Projetados a Priori**) investigamos a possibilidade de aproveitar um operador projetado previamente, seja heurística ou automaticamente, para projetar um outro de melhor precisão. Propomos uma técnica baseada em chaveamentos (i.e., alteração do valor do operador para algumas entradas), e que leva em consideração um fator de confiança definido pelo usuário. A técnica proposta pode ser vista como uma generalização do trabalho em [54].

O espaço de operadores pode ser adequadamente restrito a um subespaço a partir do conhecimento sobre características ou propriedades do operador desejado, sem reduzir a janela W . Neste caso, para que operadores possam ser projetados em um subespaço específico, tornam-se necessários algoritmos específicos. Por exemplo, em [119] um método baseado em *chaveamentos* é proposto para projetar operadores ótimos no subespaço dos operadores crescentes. A idéia básica deste método consiste em fazer modificações sobre o operador ótimo no espaço original (supondo que este é conhecido) de forma que o operador resultante satisfaça as restrições impostas, ao mesmo tempo que o erro devido às modificações seja o menor possível. Na prática o operador ótimo no espaço original não é conhecido, mas este método pode ser aplicado considerando-se apenas os exemplos observados no treinamento. Embora a idéia esteja presente no artigo original, nenhum algoritmo eficiente para calcular os chaveamentos é dado. Os custos de chaveamento, i.e., o aumento de erro devido ao chaveamento, dependem de probabilidades que, na prática, são estimadas a partir de exemplos de treinamento. Portanto, o resultado do algoritmo de chaveamento pode não ser satisfatório.

- No capítulo 7 (**Projeto de Operadores Crescentes ótimos**) revemos, inicialmente, os principais algoritmos conhecidos para projeto de operadores crescentes binários. Uma das principais contribuições desta tese é um novo algoritmo eficiente, baseado no método de chaveamentos, para projetar operadores crescentes ótimos. O algoritmo proposto é baseado em uma formulação totalmente nova e explora fortemente a estrutura de representação decorrente da propriedade crescente. Denominamos este algoritmo de OVP (“Optimal Valid Partition”) pois o problema é formulado como um problema de partição.

Consideramos também o uso de distribuições a priori para as probabilidades que definem o custo de chaveamentos, como forma para melhorar a precisão de operadores projetados pelo método de chaveamentos. Um estudo sobre distribuições a priori para essas probabilidades são introduzidas em [47].

Existe uma classe de filtros sobre imagens em níveis de cinza que podem ser implementados através de operadores sobre imagens binárias e cujo erro MAE pode ser relacionado ao erro MAE do operador binário que o implementa. Estes são conhecidos como “*stack filters*”.

- No capítulo 8 (**Filtros Stack**) revemos os “stack filters” [165] no contexto de morfologia matemática, restrito ao domínio das imagens em níveis de cinza com número finito de níveis, e relacionamos a definição utilizada no contexto de morfologia matemática com a utilizada em outros

contextos. Propriedades e representação destes operadores são revistos, sempre com a preocupação de inserí-los no contexto de projeto de operadores. Relembramos e analisamos o conceito de filtros stack estatisticamente ótimos. Mostramos como o algoritmo OVP pode ser utilizado para o projetar filtros stack ótimos em relação ao erro MAE. Mostramos também aplicações destes filtros para filtragem de ruídos “speckle” e do tipo impulso em imagens níveis de cinza.

No capítulo 9 apresentamos exemplos de aplicação das técnicas propostas para resolver problemas reais de processamento de imagens. No capítulo 10 apresentamos um resumo das principais contribuições e as conclusões deste trabalho. O apêndice apresenta uma seção com imagens que ilustram os resultados obtidos para os exemplos apresentados no capítulo 9 e uma lista de artigos que decorreram do estudo relacionado com a elaboração desta tese.

Capítulo 2

Operadores Morfológicos de Imagens Binárias

Este capítulo fornece um embasamento conceitual necessário para o desenvolvimento dos capítulos subsequentes. Começando por algumas definições e notações básicas, revemos inicialmente o conceito de W -operadores, que constituem a classe de operadores estudada nesta tese. Mostramos também alguns exemplos destes operadores, bem como algumas de suas propriedades. Em seguida, concentramo-nos no tópico de maior relevância para os capítulos subsequentes: as representações dos W -operadores, uma vez que para projetá-los necessitamos escolher uma representação para os mesmos.

A maior parte deste capítulo consiste de revisões de resultados e propriedades conhecidas [10, 11, 157, 16, 48], organizadas de forma adequada para as necessidades deste texto. Esta organização fornece uma visão global e bem estruturada sobre os W -operadores, principalmente sob a perspectiva de projeto de operadores.

2.1 Notações e Definições Básicas

Seja $E = Z^2$. Os elementos de E são denotados por letras minúsculas tais como x, y e z . A *origem* de E é denotada por o e a operação usual de adição em E por $+$. Subconjuntos¹ de E são denotados por letras maiúsculas tais como X e S . A cardinalidade de um conjunto $X \subseteq E$ é denotada por $|X|$. O *translado* de um conjunto $S \subseteq E$ por $z \in E$ é denotado S_z e definido por $S_z = \{x + z : x \in S\}$. O *complemento* de S é denotado S^c e definido por $S^c = \{x \in E : x \notin S\}$. O *transposto* de S é denotado \check{S} e definido por $\check{S} = \{x \in E : -x \in S\}$. A união e a intersecção de dois subconjuntos $A, B \subseteq E$ são denotados, respectivamente, $A \cup B$ e $A \cap B$, e definidos por $A \cup B = \{x \in E : x \in A \text{ ou } x \in B\}$ e $A \cap B = \{x \in E : x \in A \text{ e } x \in B\}$.

Seja $W \subseteq E$, um subconjunto especial a ser denominado de *janela*. A coleção de todos os subconjuntos de W é denotada por $\mathcal{P}(W)$. Assim, $\mathcal{P}(E)$ denota a coleção de todos os subconjuntos de E . Seja $K = \{0, 1, \dots, k\}$, k inteiro e $k > 0$. Uma *imagem digital* definida sobre W é uma função $f : W \rightarrow K$. Se $k = 1$, então a imagem é *binária*; caso contrário, ela é uma imagem em *níveis* de

¹Conhecimentos sobre noções básicas da teoria de conjuntos [59] e matemática discreta [139] podem ser úteis para a leitura deste trabalho.

cinza. A coleção de todas as imagens definidas sobre W , com valores em K , é denotada por K^W . Da mesma forma, K^E denota todas as imagens definidas sobre E com valores em K . Nesta tese a atenção será voltada a imagens binárias, com algumas exceções.

2.1.1 Imagens Binárias e Conjuntos

Recordamos aqui a equivalência entre imagens binárias e conjuntos.

Definição 2.1 (Limiarização e corte) O mapeamento $T_t : K^W \rightarrow \mathcal{P}(W)$ definido por

$$x \in T_t[f] \iff f(x) \geq t \quad (2.1)$$

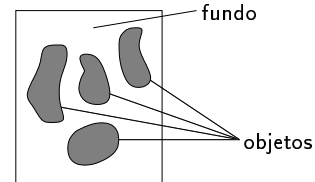
para todo $f \in K^W$, $t \in K$ e $x \in W$, é a limiarização de f no nível t . O conjunto $T_t[f]$ é o corte de f no nível t .

Seja f uma imagem binária e seja $T_1[f]$ o seu correspondente conjunto; seja S um subconjunto de W e 1_S , a função indicadora de S , definida por, $\forall x \in W$,

$$1_S(x) = 1 \iff x \in S, \quad (2.2)$$

a sua correspondente função binária. Qualquer imagem binária $f \in \{0, 1\}^W$ define um único conjunto em $\mathcal{P}(W)$ e vice-versa, pois para todo $S \in \mathcal{P}(W)$ e $f \in \{0, 1\}^W$, $T_1[1_S] = S$ e $1_{T_1[f]} = f$. O conjunto de todas as imagens binárias definidas sobre W pode ser, portanto, equivalentemente denotado por $\mathcal{P}(W)$.

O conjunto S define os *objetos* (“foreground”) da imagem 1_S , enquanto S^c define o *fundo* (“background”) de 1_S . Nas figuras ao longo deste texto, os objetos aparecem como os elementos escuros enquanto o fundo aparece como regiões brancas (ver figura ao lado). Os pontos de E são muitas vezes referidos como “pixel”; os pontos que correspondem aos objetos são os “pixels acesos” e aqueles que correspondem ao fundo são os “pixels apagados”. Por conveniência, ao longo deste texto, um conjunto S será denominado imagem, mas deve ficar claro que a imagem em questão é 1_S .



A coleção $\mathcal{P}(W)$, com a relação usual de inclusão de conjuntos, forma um *reticulado Booleano completo* [23]. As operações de união e interseção são, respectivamente, os operadores \cup e \cap definidos acima. O menor e o maior elementos deste reticulado são, respectivamente, \emptyset e W .

2.1.2 Operadores de Imagens

Um *operador de imagens* é uma função $\Psi : K^E \rightarrow K^E$. Um *operador* de imagens binárias pode ser entendido também como um *operador de conjuntos*, i.e., $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$. Na maior parte deste texto trataremos apenas de operadores binários, os quais serão tratados como operadores de conjuntos, salvo menção em contrário.

Alguns operadores básicos

Definição 2.2 (Constante I e O) Os operadores $I : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ e $O : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$, definidos por $I(S) = E$ e $O(S) = \emptyset$ para todo $S \in \mathcal{P}(E)$, são respectivamente os operadores constante E e constante \emptyset .

Definição 2.3 (Identidade) O operador $\iota : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ definido por $\iota(S) = S$ para todo $S \in \mathcal{P}(E)$ é denominado o operador identidade.

Definição 2.4 (Complemento) O operador $\nu : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ definido por $\nu(S) = S^c$ para todo $S \in \mathcal{P}(E)$ é denominado o operador complemento.

Definição 2.5 (Translação) Dado $z \in E$, o operador $\tau_z : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ definido por $\tau_z(S) = S_z$ para todo $S \in \mathcal{P}(E)$ é denominado o operador translação por z .

Definição 2.6 (Erosão e dilatação) Seja $B \in \mathcal{P}(E)$. Os operadores $E_B : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ e $D_B : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ definidos, respectivamente, por

$$E_B(S) = \{z \in E : B_z \subseteq S\}$$

e

$$D_B(S) = \{z \in E : B_z \cap S \neq \emptyset\}$$

para todo $S \in \mathcal{P}(E)$, são denominados a erosão e a dilatação por B . O conjunto B é denominado elemento estruturante.²

Definição 2.7 (Operador sup-gerador) Sejam A, B tais que $A \subseteq B \subseteq E$. O operador definido por

$$\Lambda_{(A,B)}(S) = \{z \in E : A_z \subseteq S \subseteq B_z\} \quad (2.3)$$

$\forall S \in \mathcal{P}(E)$, é o operador sup-gerador com parâmetros A e B .

Os operadores sup-geradores são equivalentes aos operadores “hit-or-miss” [147]. Estes são denotados $\mathcal{H}_{(U,V)}$, $U, V \in \mathcal{P}(E)$, e definidos por $\mathcal{H}_{(U,V)}(S) = \{x \in E : U_x \subseteq S \text{ e } V_x \subseteq S^c\}$, para qualquer $S \in \mathcal{P}(E)$. Note que $\Lambda_{(U,V)} = \mathcal{H}_{(U,V^c)}$.

Definição 2.8 (Composição de operadores) Sejam Ψ_1 e Ψ_2 dois operadores de $\mathcal{P}(E)$ em $\mathcal{P}(E)$. A composição de Ψ_1 com Ψ_2 é denotado $\Psi_2 \Psi_1$ e definido por $\Psi_2 \Psi_1(S) = \Psi_2(\Psi_1(S))$, $\forall S \in \mathcal{P}(E)$.

Dois operadores formados por composição e que são bastante utilizados são a abertura $\gamma_B = D_B E_B$ e o fechamento $\varphi_B = E_B D_B$.

²Na literatura, duas definições para a dilatação são utilizadas. A apresentada neste texto consiste da definição utilizada por Serra [147]. A outra definição, utilizada por Heijmans [75] por exemplo, usa a definição $D_B(S) = \{z \in E : \check{B}_z \cap S \neq \emptyset\}$, onde \check{B} corresponde ao conjunto transposto de B (neste caso, o elemento estruturante é o conjunto \check{B}).

Proposição 2.9 *Um operador sup-gerador $\Lambda_{(A,B)}$ pode ser expresso em termos de uma erosão e uma dilatação, da seguinte forma:*

$$\Lambda_{(A,B)} = E_A \wedge \nu D_{B^c}$$

Dem.: *Veja demonstração em [9].* ■

Definição 2.10 (Operador Invariante por Translação) *Um operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é dito ser invariante por translação (i.t.) se e somente se (sse) $\Psi(S_z) = [\Psi(S)]_z$, para todo $S \in \mathcal{P}(E)$ e $z \in E$.*

A invariância por translação implica que o resultado de um operador transladado por z é exatamente o resultado obtido transladando-se primeiro o objeto por z e depois aplicando o operador sobre este objeto transladado. Sob outra perspectiva, significa que a localização dos objetos não é relevante: objetos iguais a menos de translação são mapeados para objetos iguais, a menos de translação. Todos os operadores apresentados acima são invariantes por translação.

Definição 2.11 (Operadores localmente definidos) *Um operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é localmente definido (l.d.) em W sse $x \in \Psi(S) \iff x \in \Psi(S \cap W_x)$, para todo $x \in E$ e $S \in \mathcal{P}(E)$.*

2.2 W-operadores e Sua Representação

Definição 2.12 (W-operador) *Um operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é um W-operador sse ele é i.t. e l.d. em W .*

Qualquer operador i.t. $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é localmente definido em E , pois $E_x = E$ para todo $x \in E$ e, portanto, $x \in \Psi(S) \iff x \in \Psi(S \cap E) \iff x \in \Psi(S \cap E_x)$ para todo $S \in \mathcal{P}(E)$. Logo, os operadores i.t. podem ser vistos dentro do contexto de W-operadores (usando $W = E$). Mais geralmente, se Ψ é um W-operador, então ele é também um W' -operador para quaisquer W' satisfazendo $W \subseteq W'$.

A coleção de todos os W-operadores, denotado Ψ_W , forma também um reticulado Booleano completo. A relação de ordem (\leq), as operações de ínfimo (\wedge) e supremo (\vee), e o operador complemento são herdados do reticulado dos conjuntos. Isto é, para quaisquer $\Psi_1, \Psi_2 \in \Psi_W$,

$$\begin{aligned} \Psi_1 \leq \Psi_2 &\iff \Psi_1(S) \subseteq \Psi_2(S), \forall S \in \mathcal{P}(E) \\ (\Psi_1 \wedge \Psi_2)(S) &= \Psi_1(S) \cap \Psi_2(S), \forall S \in \mathcal{P}(E) \\ (\Psi_1 \vee \Psi_2)(S) &= \Psi_1(S) \cup \Psi_2(S), \forall S \in \mathcal{P}(E) \\ (\nu \Psi_1)(S) &= [\Psi_1(S)]^c, \forall S \in \mathcal{P}(E). \end{aligned}$$

O menor e o maior elementos deste reticulado são os operadores O e I , respectivamente.

Proposição 2.13 (Caracterização de W-operadores por funções) *Um operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é um W-operador sse existe uma função $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ tal que*

$$\Psi(S) = \{x \in E : \psi(S_{-x} \cap W) = 1\} \tag{2.4}$$

para todo $S \subseteq \mathcal{P}(E)$.

Dem.:

(\Rightarrow) Seja $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ um W-operador. Defina $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ da seguinte forma:

$$\psi(X) = 1 \iff o \in \Psi(X)$$

para todo $X \in \mathcal{P}(W)$. Então,

$$\begin{aligned} x \in \Psi(S) &\stackrel{(1)}{\iff} x \in \Psi(S \cap W_x) \\ &\stackrel{(2)}{\iff} o \in [\Psi(S \cap W_x)]_{-x} \\ &\stackrel{(3)}{\iff} o \in \Psi(S_{-x} \cap W) \\ &\stackrel{(4)}{\iff} \psi(S_{-x} \cap W) = 1 \end{aligned}$$

para todo $S \in \mathcal{P}(E)$. A equivalência (1) vale pois Ψ é l.d., (3) pois Ψ é i.t., e (4) pois $S_{-x} \cap W \subseteq W$.

(\Leftarrow) Para todo $x, z \in E$ e $S \in \mathcal{P}(E)$,

$$\begin{aligned} x \in \Psi(S_z) &\stackrel{(1)}{\iff} \psi(S_{z-x} \cap W) = 1 \\ &\stackrel{(2)}{\iff} -z + x \in \Psi(S) \\ &\stackrel{(3)}{\iff} x \in [\Psi(S)]_z \end{aligned}$$

As equivalências (1) e (2) decorrem da definição, e (3) pela propriedade de translação. Portanto Ψ é i.t. Além disso,

$$\begin{aligned} x \in \Psi(S) &\stackrel{(1)}{\iff} \psi(S_{-x} \cap W) = 1 \\ &\stackrel{(2)}{\iff} \psi((S \cap W_x)_{-x}) = 1 \\ &\stackrel{(3)}{\iff} \psi((S \cap W_x)_{-x} \cap W) = 1 \\ &\stackrel{(4)}{\iff} x \in \Psi(S \cap W_x) \end{aligned}$$

As equivalências (1) e (4) decorrem da definição, (2) pois $S_{-x} \cap W = (S \cap W_x)_{-x}$ e (3) $(S \cap W_x)_{-x} \subseteq W$. Portanto Ψ é l.d. ■

Heijmans [75] (seção 4.5) mostra que qualquer função $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ caracteriza um W-operador. Juntando isso com a proposição anterior, temos que todo W-operador Ψ é caracterizado por um mapeamento $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ e vice-versa. O mapeamento ψ é denominado a *função característica* de Ψ .

Exemplo 2.14 A erosão E_A e a dilatação D_A são W-operadores com $W = A$, enquanto $\Lambda_{(A,B)}$ é um W-operador com $W = B$. O operador que extrai as bordas internas de uma imagem binária, definido por $\Psi = i \wedge \nu E_B$, onde B é o elemento estruturante 3×3 , é um $W_{3 \times 3}$ -operador. A função característica da erosão E_A é a função ε_A , definida por $\varepsilon_A(X) = 1 \iff A \subseteq X, \forall X \in \mathcal{P}(W)$. A função característica da dilatação D_A é a função δ_A , definida por $\delta_A(X) = 1 \iff A \cap X \neq \emptyset, \forall X \in \mathcal{P}(W)$.

2.2.1 Algumas Famílias de W-operadores

Os W-operadores que satisfazem certas propriedades definem subclasses de operadores. Apresentamos nesta seção algumas dessas propriedades que serão retomadas mais adiante. Seja Ψ um W-operador. Então, Ψ é

- *anti-extensivo* sse $\Psi(S) \subseteq S$, $\forall S \in \mathcal{P}(E)$.
- *anti-extensivo complementar* sse $\Psi(S) \subseteq S^c$, $\forall S \in \mathcal{P}(E)$.
- *extensivo* sse $S \subseteq \Psi(S)$, $\forall S \in \mathcal{P}(E)$.
- *extensivo complementar* sse $S^c \subseteq \Psi(S)$, $\forall S \in \mathcal{P}(E)$.
- *crescente* sse $S_1 \subseteq S_2 \implies \Psi(S_1) \subseteq \Psi(S_2)$, para quaisquer $S_1, S_2 \in \mathcal{P}(E)$.
- *decrescente* sse $S_1 \subseteq S_2 \implies \Psi(S_2) \subseteq \Psi(S_1)$, para quaisquer $S_1, S_2 \in \mathcal{P}(E)$.
- *auto-dual* sse $\Psi^*(S) = \Psi(S)$, para todo $S \in \mathcal{P}(E)$, onde o dual de Ψ , Ψ^* , é definido por $\Psi^*(S) = [\Psi(S^c)]^c$.

A noção de operadores extensivo-complementares e anti-extensivo complementares, embora simples, não são conhecidas na literatura consultada.

2.2.2 Núcleo e Base de W-operadores

O núcleo de um W-operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é definido por:

$$\mathcal{K}_W(\Psi) = \{X \in \mathcal{P}(W) : o \in \Psi(X)\} \quad (2.5)$$

Note que se $W = E$, então temos $\mathcal{K}_W(\Psi) = \mathcal{K}(\Psi) = \{X \in \mathcal{P}(E) : o \in \Psi(X)\}$ (a definição clássica de núcleo para operadores i.t.).

Uma vez que $o \in \Psi(X) \iff \psi(X) = 1$, para todo $X \in \mathcal{P}(W)$, podemos reescrever

$$\mathcal{K}_W(\Psi) = \{X \in \mathcal{P}(W) : \psi(X) = 1\}.$$

As seguintes relações entre $\mathcal{K}_W(\Psi)$ e $\mathcal{K}(\Psi)$ são verificadas:

1. $\mathcal{K}_W(\Psi) \subseteq \mathcal{K}(\Psi)$. De fato, $\forall X \in \mathcal{P}(W)$,

$$\begin{aligned} X \in \mathcal{K}_W(\Psi) &\iff \psi(X) = 1 \quad (\text{definição de } \mathcal{K}_W(\Psi)) \\ &\implies o \in \Psi(X_o \cap W) \\ &\iff o \in \Psi(X) \\ &\iff X \in \mathcal{K}(\Psi) \end{aligned}$$

2. $S \in \mathcal{K}(\Psi) \iff S \cap W \in \mathcal{K}_W(\Psi)$. De fato,

$$S \in \mathcal{K}(\Psi) \iff o \in \Psi(S) \iff \psi(S_{-o} \cap W) = 1 \iff \psi(S \cap W) = 1 \iff S \cap W \in \mathcal{K}_W(\Psi).$$

Definição 2.15 (Intervalo) O conjunto $[A, B] = \{X \in \mathcal{P}(W) : A \subseteq X \subseteq B\}$ é denominado intervalo com extremidades A e B. Se $A = B$ então $[A, B]$ é um intervalo trivial (i.e., contém um único elemento) e se $A \not\subseteq B$ então $[A, B]$ é um intervalo degenerado (i.e., $[A, B] = \emptyset$).

Definição 2.16 (Base) A base de um operador i.t. Ψ , denotado $\mathcal{B}_W(\Psi)$, é o conjunto de todos os intervalos maximais de $\mathcal{K}_W(\Psi)$. Isto é, $[A, B] \in \mathcal{B}_W(\Psi)$ implica que $\forall [A', B'] \subseteq \mathcal{K}_W(\Psi)$, se $[A, B] \subseteq [A', B']$ então $[A, B] = [A', B']$.

2.2.3 Representações em Termos de Núcleo e de Base

Qualquer W-operador pode ser expresso unicamente em termos de seu núcleo, isto é, como a união de operadores sup-geradores caracterizados pelos intervalos do núcleo (Barrera e Salas [16]):

$$\Psi = \bigvee \left\{ \Lambda_{(A,B)} : [A, B] \subseteq \mathcal{K}_W(\Psi) \right\}. \quad (2.6)$$

Esta representação é uma generalização do resultado prévio para o caso de operadores i.t devido a Banon e Barrera [9],

$$\Psi = \bigvee \left\{ \Lambda_{(A,B)} : [A, B] \subseteq \mathcal{K}(\Psi) \right\}, \quad (2.7)$$

que, por sua vez, generaliza um resultado prévio devido a Matheron [118], para o caso de operadores i.t. crescentes:

$$\Psi = \bigvee \left\{ E_B : B \in \mathcal{K}(\Psi) \right\}. \quad (2.8)$$

Em termos de sua base, Ψ pode ser expresso por :

$$\Psi = \bigvee \left\{ \Lambda_{(A,B)} : [A, B] \in \mathcal{B}(\Psi) \right\} \quad (2.9)$$

e no caso de operadores crescentes, por :

$$\Psi = \bigvee \left\{ E_A : [A, E] \in \mathcal{B}(\Psi) \right\}. \quad (2.10)$$

Esta última representação é facilmente derivada a partir da proposição 2.9 e da equação 2.9, pois se Ψ é crescente, então todos os intervalos maximais contidos em $\mathcal{K}_W(\Psi)$ são da forma $[A, E]$, e $D_{E^c}(S)^c = D_{\emptyset}(S)^c = \emptyset^c = E$, $\forall S \in \mathcal{P}(E)$. Isto implica que $E_A \wedge \nu D_{E^c} = E_A$.

Núcleo de Algumas Famílias de W-Operadores

Os operadores que satisfazem certas propriedades algébricas possuem núcleos com características especiais. A seguir listamos algumas classes de operadores e a característica dos respectivos núcleos.

Proposição 2.17 Seja $W \subseteq \mathcal{P}(E)$ tal que $o \in W$. Um W-operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é anti-extensivo sse todos os elementos $\mathcal{K}_W(\Psi)$ contém a origem.

Dem.:

(\implies) Seja $X \in \mathcal{K}_W(\Psi)$. Então $o \in \Psi(X)$ por definição. Como Ψ é anti-extensivo, então $\Psi(X) \subseteq X$, e portanto $o \in X$.

(\impliedby) Se todos os elementos de $\mathcal{K}_W(\Psi)$ contém a origem, então $x \in \Psi(S) \implies x \in \{x \in E : \psi(S_{-x} \cap W) = 1\} \implies o \in S_{-x} \cap W \implies o \in S_{-x} \implies x \in S$. ■

Proposição 2.18 Um W -operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é anti-extensivo complementar sse todos os elementos de $\mathcal{K}_W(\Psi)$ não contém a origem.

Dem.: Análogo ao caso anti-extensivo. ■

Proposição 2.19 Um W -operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é extensivo sse todos os elementos de $\mathcal{P}(W)$ que contém a origem pertencem a $\mathcal{K}_W(\Psi)$.

Dem.:

(\implies) Se Ψ é extensivo, então $\forall S \in \mathcal{P}(E)$, $S \subseteq \Psi(S)$. Seja $X \subseteq W$ tal que $o \in X$. Como $o \in X \subseteq \Psi(X)$ então $o \in \Psi(X)$, isto é, $X \in \mathcal{K}_W(\Psi)$.

(\impliedby) Seja $S \in \mathcal{P}(E)$. Para todo $x \in S$, $o \in S_{-x}$ e portanto, $S_{-x} \cap W \in \mathcal{K}_W(\Psi)$, isto é, $\psi(S_{-x} \cap W) = 1$. Logo $S \subseteq \Psi(S)$ (ou seja, Ψ é extensivo). ■

Proposição 2.20 Um W -operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é extensivo complementar sse todos os elementos de $\mathcal{P}(W)$ que não contém a origem pertencem a $\mathcal{K}_W(\Psi)$.

Dem.: Análogo ao caso extensivo. ■

Proposição 2.21 Um W -operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é crescente sse $X \in \mathcal{K}_W(\Psi) \implies Y \in \mathcal{K}_W(\Psi)$, para todo $X, Y \in \mathcal{P}(W)$ satisfazendo $X \subseteq Y$.

Dem.:

(\implies) Sejam $X, Y \in \mathcal{P}(W)$ tal que $X \subseteq Y$. Se $X \in \mathcal{K}_W(\Psi)$ então $o \in \Psi(X)$. Como Ψ é crescente, então $\Psi(X) \subseteq \Psi(Y)$ e portanto $o \in \Psi(Y)$. Logo $Y \in \mathcal{K}_W(\Psi)$.

(\impliedby) Sejam $S, S' \in \mathcal{P}(E)$ tal que $S \subseteq S'$. Então,

$$\begin{aligned}
 x \in \Psi(S) &\iff o \in \Psi(S_{-x}) \\
 &\iff S_{-x} \cap W \in \mathcal{K}_W(\Psi) \\
 &\implies S'_{-x} \cap W \in \mathcal{K}_W(\Psi) \quad (\text{pois } S_{-x} \cap W \subseteq S'_{-x} \cap W) \\
 &\iff o \in \Psi(S'_{-x}) \\
 &\iff x \in \Psi(S')
 \end{aligned}$$

■

Proposição 2.22 Um W -operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é decrescente sse $X \notin \mathcal{K}_W(\Psi) \implies Y \notin \mathcal{K}_W(\Psi)$, para todo $X, Y \in \mathcal{P}(W)$ satisfazendo $X \subseteq Y$.

Dem.: Análoga ao caso crescente. ■

Proposição 2.23 Um W -operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é auto-dual sse $X \in \mathcal{K}_W(\Psi) \iff X^c \cap W \notin \mathcal{K}_W(\Psi)$, $\forall X \in \mathcal{P}(W)$.

Dem.:

(\implies) Para qualquer $X \in \mathcal{P}(W)$, $X \in \mathcal{K}_W(\Psi) \iff o \in \Psi(X) = [\Psi(X^c)]^c \iff o \notin \Psi(X^c) \iff X^c \notin \mathcal{K}_W(\Psi)$.

(\impliedby) Para qualquer $x \in E$, $x \in \Psi(S) \iff o \in \Psi(S_{-x}) \iff S_{-x} \cap W \in \mathcal{K}_W(\Psi) \iff S^c \cap W \notin \mathcal{K}_W(\Psi) \iff o \in \Psi(S_{-x}) \iff x \notin \Psi(S^c) \iff x \in [\Psi(S^c)]^c$.

■

2.3 Funções Booleanas e sua Representação

Nesta seção apresentamos uma revisão sobre funções Booleanas [77, 37]. Uma função $f : \{0, 1\}^n \rightarrow \{0, 1\}$, onde n é um inteiro positivo, é denominado *função Booleana*. Como existe uma correspondência 1 para 1 entre elementos de $\mathcal{P}(W)$ e $\{0, 1\}^{|W|}$, os mapeamentos ψ de $\mathcal{P}(W)$ em $\{0, 1\}$ podem também ser vistos como funções Booleanas. Supondo $W = \{w_1, w_2, \dots, w_n\}$, cada elemento $X \in \mathcal{P}(W)$ pode ser visto como um elemento do espaço $\{0, 1\}^{|W|}$ e vice-versa: X equivale a $\underline{X} = (x_1, x_2, \dots, x_{|W|}) \in \{0, 1\}^{|W|}$ onde $x_i = 1 \iff w_i \in X, \forall i = 1, 2, \dots, |W|$. Isto é, se x_1, x_2, \dots, x_n são as variáveis da função Booleana ψ , então o valor de ψ é calculado para um elemento $X \in \mathcal{P}(W)$ fazendo-se $x_i = 1 \iff w_i \in X$. Conseqüentemente, para representar W -operadores podemos estudar representações de funções Booleanas.

Daqui em diante, não faremos distinção entre mapeamentos de $\{0, 1\}^n$ para $\{0, 1\}$ e mapeamentos de $\mathcal{P}(W)$ para $\{0, 1\}$. Um mapeamento $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ será considerada uma função Booleana de $|W|$ variáveis.

2.3.1 Representação Tabular

Uma função Booleana pode ser definida especificando-se o seu valor para cada elemento de $\{0, 1\}^n$. Uma forma simples para representar uma função assim especificada é através de uma tabela de 2 colunas, onde cada linha da primeira coluna contém um elemento de $\{0, 1\}^n$, e a segunda coluna contém o valor da função Booleana para o elemento correspondente na primeira coluna. Se alguma ordenação padronizada é utilizada, então pode-se omitir a primeira coluna. Tabelas deste tipo são conhecidas como *tabelas-verdade*.

Exemplo 2.24 A seguinte tabela-verdade define uma função Booleana de 3 variáveis $\psi : \{0, 1\}^3 \rightarrow \{0, 1\}$. A ordem adotada para os elementos de $\{0, 1\}^3$ é a ordem lexicográfica.

$x_1 x_2 x_3$	$\psi(x_1, x_2, x_3)$
000	1
001	1
010	0
011	1
100	0
101	0
110	0
111	1

Uma função Booleana ψ pode também ser caracterizada em termos do seu conjunto-1, denotado $\psi\langle 1 \rangle$ e definido por $\psi\langle 1 \rangle = \{X \in \{0, 1\}^n : \psi(X) = 1\}$. Alternativamente, pode também ser caracterizada pelo seu conjunto-0, denotado $\psi\langle 0 \rangle$ e definido por $\psi\langle 0 \rangle = \{X \in \{0, 1\}^n : \psi(X) = 0\}$. Se a função é completamente especificada, i.e, se seu valor é definido para todos os elementos de $\{0, 1\}^n$, então $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle = \{0, 1\}^n$. A função que mapeia todos os elementos de $\{0, 1\}^n$ para 0 é denotado por um zero em negrito, **0**; aquele que mapeia todos para 1 é denotado **1**. No exemplo acima, $\psi\langle 1 \rangle = \{000, 001, 011, 111\}$ e $\psi\langle 0 \rangle = \{010, 100, 101, 110\}$. Vale notar que $\psi\langle 1 \rangle$ equivale a $\mathcal{K}_W(\Psi)$.

O conjunto $\{0, 1\}^n$ (ou equivalentemente, $\mathcal{P}(W)$) é um *reticulado Booleano completo* com respeito à relação \leq (respec, \subseteq). A figura 2.1 mostra os elementos de $\{0, 1\}^n$ através de um *diagrama de Hasse*.

Uma função Booleana ψ será freqüentemente ilustrada através desse diagrama, onde elementos de $\psi\langle 1 \rangle$ aparecem como nós escuros (círculos hachurados) e os elementos de $\psi\langle 0 \rangle$ aparecem como nós brancos (círculos não hachurados). A figura 2.1 ilustra a função $\psi(x_1, x_2, x_3) = x_1 x_2 + x_1 x_3 + x_2 x_3$, conhecida como a *função da mediana*.

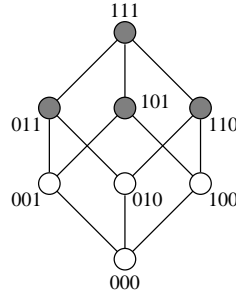


Figura 2.1: Representação gráfica da função $\psi(x_1, x_2, x_3) = x_1 x_2 + x_1 x_3 + x_2 x_3$, onde $\psi\langle 1 \rangle = \{011, 101, 110, 111\}$ e $\psi\langle 0 \rangle = \{000, 001, 010, 100\}$.

Um sério problema com a representação tabular é que ela requer muito espaço para armazenamento. Uma tabela-verdade de uma função com n variáveis contém exatamente 2^n linhas. Isto implica que tanto a tabela-verdade quanto quaisquer outras representações que consistem em enumerar explicitamente os elementos (ou parte dos elementos) de $\{0, 1\}^n$ podem tornar-se inviáveis do ponto de vista prático. No contexto de operadores morfológicos, significa que representar o núcleo de um W-operador explicitamente não é uma solução prática.

2.3.2 Expressões Booleanas

Funções Booleanas podem ser definidas a partir de expressões contendo literais (variáveis ou variáveis complementadas), os operadores binários $+$ e \cdot , e parênteses. Mais precisamente, definimos expressões Booleanas em n -variáveis x_1, x_2, \dots, x_n recursivamente da seguinte forma :

- Os símbolos $0, 1$ e x_1, x_2, \dots, x_n são expressões Booleanas ;
- Se E_1 e E_2 são expressões Booleanas em x_1, x_2, \dots, x_n , então também o são $(E_1) + (E_2)$, $(E_1) \cdot (E_2)$ e $\overline{E_1}$ (o operador \cdot será omitido e, em geral, os parênteses são omitidos).

A partir de uma expressão, podemos construir sua tabela-verdade avaliando seu valor para cada elemento de $\{0, 1\}^n$. Por exemplo, a tabela-verdade para a expressão $f(x_1, x_2, x_3) = \overline{x_1}(\overline{x_2}x_3 + x_2x_3) + x_3(\overline{x_1}\overline{x_2} + x_1x_2)$ é exatamente a mesma mostrada no exemplo 2.24. Duas expressões são ditas *equivalentes* se elas definem uma mesma função Booleana.

Existem duas expressões canônicas, SSOP (soma canônica de produtos, do inglês “standard sum of products”) e SPOS (produto canônico de somas, do inglês “standard product of sums”), que podem ser diretamente obtidas a partir da tabela verdade. A forma SSOP de ψ é a soma dos produtos correspondentes a elementos mapeados para 1, enquanto a forma SPOS é o produto de somas correspondentes aos elementos mapeados para 0. No caso do exemplo 2.24, a forma SSOP e SPOS são, respectivamente, $\psi(x_1, x_2, x_3) = \overline{x_1} \overline{x_2} \overline{x_3} + \overline{x_1} \overline{x_2} x_3 + \overline{x_1} x_2 x_3 + x_1 x_2 x_3$ e $\psi(x_1, x_2, x_3) = (x_1 + \overline{x_2} + x_3)(\overline{x_1} + x_2 + x_3)(\overline{x_1} + x_2 + \overline{x_3})(\overline{x_1} + \overline{x_2} + x_3)$. O termo canônico vem do fato de que uma

função pode ser unicamente representada nesta forma [77]. No diagrama de Hasse, ψ é representada como na figura 2.2.

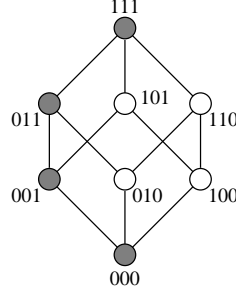


Figura 2.2: Representação de ψ através do diagrama de Hasse.

Conforme vimos nos exemplos citados, uma mesma função Booleana pode ser representada por diferentes expressões. A minimização de funções Booleanas, i.e, a sua representação por expressões equivalentes minimais, é um assunto bastante estudado [77, 32, 120]. Uma das formas minimais mais estudadas é a forma minimal de soma de produtos.

Definição 2.25 (Expressão minimal) *Uma expressão escrita como soma de produtos é uma expressão minimal (ou MSOP) se (1) não existe outra expressão equivalente envolvendo um menor número de produtos, e (2) não existe nenhuma outra expressão equivalente envolvendo o mesmo número de produtos mas um número menor de literais.*

Apresentamos alguns conceitos adicionais que serão úteis para o entendimento de expressões que consistem de soma de produtos. Um produto com $n - k$ literais associa valor 1 a exatamente 2^k elementos de $\mathcal{P}(W)$. Estes 2^k elementos formam um intervalo e são geralmente denominados de k -cubos. Por exemplo, um elemento $\{X\} \in \mathcal{P}(W)$ é um 0-cubo enquanto $\mathcal{P}(W)$ é um n -cubo. A forma SSOP de uma função Booleana pode ser entendida como uma coleção de 0-cubos. Similarmente, qualquer expressão como soma de produtos corresponde a uma coleção de cubos (ou intervalos). Graficamente, um 0-cubo corresponde a um vértice no diagrama de Hasse, enquanto um 1-cubo corresponde a uma aresta, um 2-cubo a uma face, e assim por diante.

Um intervalo $[A, B]$ pode ser denotado sucintamente por uma cadeia de caracteres de comprimento $|W|$, contendo os caracteres 0, 1 ou X : para cada variável x_i , o i -ésimo caractere da cadeia é

$$\begin{aligned} 1 & \text{ se } w_i \in A \\ 0 & \text{ se } w_i \notin B \\ X & \text{ se } w_i \notin A \text{ e } w_i \in B \end{aligned}$$

Por exemplo, o intervalo $[100, 110]$ pode ser representado por $1X0$.

Funções Booleanas herdam a relação de ordem do reticulado $(\mathcal{P}(W), \subseteq)$, i.e., para quaisquer funções Booleanas ψ_1 e ψ_2 , $\psi_1 \leq \psi_2 \iff \psi_1(X) \leq \psi_2(X)$ para todo $X \in \mathcal{P}(W)$.

Definição 2.26 (Implicante primo) *Um implicante primo de uma função Booleana ψ é um produto p tal que $p(x_1, x_2, \dots, x_n) = 1 \implies \psi(x_1, x_2, \dots, x_n) = 1$ e não existe nenhum outro produto p' , $p \leq p'$ tal que $p'(x_1, x_2, \dots, x_n) = 1 \implies \psi(x_1, x_2, \dots, x_n) = 1$.*

Os implicantes primos de uma função Booleana correspondem aos cubos maximais formados por elementos de $\psi\langle 1 \rangle$.

Exemplo 2.27 Os implicantes primos da função Booleana $\psi(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3 + x_1 x_2 x_3$ são os produtos $\bar{x}_1 \bar{x}_2$, $\bar{x}_1 x_3$ e $x_2 x_3$, que correspondem respectivamente aos cubos maximais 00X, 0X1 e X11 de $\psi\langle 1 \rangle$. A representação minimal de ψ é dada pela expressão $\psi(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 + x_2 x_3$, que corresponde aos dois implicantes primos necessários para cobrir todos os elementos de $\psi\langle 1 \rangle$. Veja a representação através do diagrama de Hasse na figura 2.3.

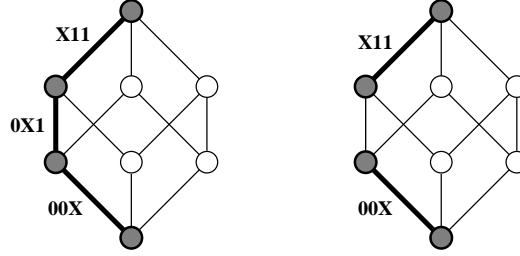


Figura 2.3: Os intervalos maximais de $On(\psi)$ (esquerda) e os intervalos da expressão minimal (direita).

A forma MSOP da função Booleana do exemplo 2.24 é a expressão $\psi(x_1, x_2, x_3) = \bar{x}_1 \bar{x}_2 + x_2 x_3$. Existe um conjunto de regras algébricas que, quando convenientemente aplicadas sobre quaisquer expressões Booleanas, proporcionam um meio para se obter sua forma SSOP [77, 96]. Uma vez que as funções Booleanas estejam na forma SSOP, algoritmos clássicos como o *mapa de Karnaugh* (para minimização de funções de até 6 variáveis) ou o algoritmo de *Quine-McCluskey* (QM) podem ser utilizados para a sua minimização. A minimização de funções Booleanas será discutida em um capítulo à parte, capítulo 4.

Expansão de Shannon

A função que resulta quando algum argumento x_i de ψ é substituído por uma constante b ($b \in \{0, 1\}$) é denominada *restrição de ψ ou co-fator*, e é denotada por $\psi|_{x_i=b}$. Isto é, para qualquer argumento x_i , $\psi|_{x_i=b}(x_1, \dots, x_n) = \psi(x_1, \dots, x_{i-1}, b, x_{i+1}, \dots, x_n)$. Note que as funções $\psi|_{x_i=1}$ e $\psi|_{x_i=0}$ são funções que dependem de $n - 1$ variáveis.

Usando esta notação, a *expansão de Shannon* de ψ em torno de x_i é dada por

$$\psi = x_i \psi|_{x_i=1} + \bar{x}_i \psi|_{x_i=0} \quad (2.11)$$

Por exemplo, a função do exemplo 2.24 expandida em torno de x_2 é expressa como :

$$\begin{aligned} \psi(x_1, x_2, x_3) &= \bar{x}_1 \bar{x}_2 \bar{x}_3 + \bar{x}_1 \bar{x}_2 x_3 + \bar{x}_1 x_2 x_3 + x_1 x_2 x_3 \\ &= x_2 (\bar{x}_1 x_3 + x_1 x_3) + \bar{x}_2 (\bar{x}_1 \bar{x}_3 + \bar{x}_1 x_3) \end{aligned}$$

Ou seja, neste caso, os co-fatores são : $\psi|_{x_i=1} = \bar{x}_1 x_3 + x_1 x_3$ e $\psi|_{x_i=0} = \bar{x}_1 \bar{x}_3 + \bar{x}_1 x_3$. Cada um dos co-fatores pode ser sucessivamente expandido em torno de uma outra variável. A expansão da mesma função por x_2 , seguida pela expansão por x_1 e depois por x_3 pode ser visualizado no esquema da figura 2.4. Esta expansão será explorada mais adiante, quando falaremos sobre representações associadas às partições de $\mathcal{P}(W)$.

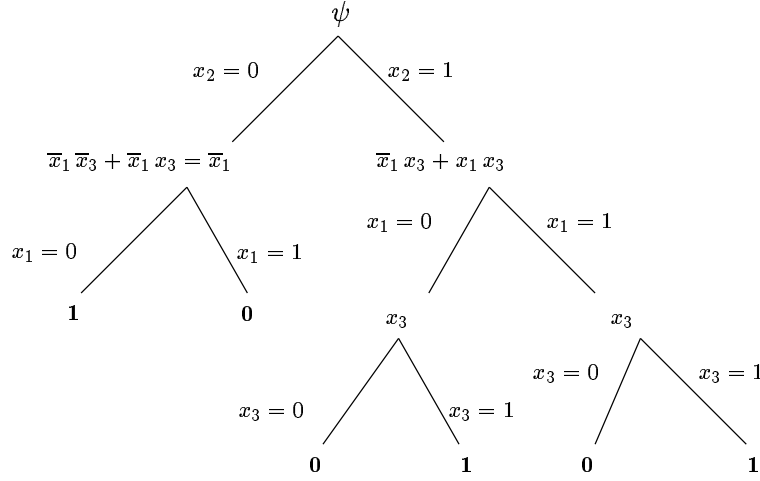


Figura 2.4: Esquema da expansão de Shannon de uma função Booleana.

2.4 Equivalência entre Representações Morfológicas e Booleanas

Nesta seção veremos primeiramente como mapear um operador morfológico para uma função Booleana e vice-versa. Faremos também algumas observações com respeito à equivalência de representações.

2.4.1 Conversão Intervalo-Expressão

Seja $[A, B]$ um intervalo de $\mathcal{P}(W)$. Definimos uma função $p_{[A, B]} : \{0, 1\}^n \rightarrow \{0, 1\}$ pela seguinte expressão :

$$p_{[A, B]}(x_1, x_2, \dots, x_n) = \prod_{i=1}^n \sigma_i(A, B) \quad (2.12)$$

onde

$$\sigma_i(A, B) = \begin{cases} x_i & \text{se } w_i \in A, \\ \bar{x}_i & \text{se } w_i \notin B, \\ ' & \text{se } w_i \notin A \text{ e } w_i \in B. \end{cases}$$

e ' indica o caractere vazio. Por exemplo, o intervalo $[A, B] = [010, 110] \subseteq \mathcal{P}(W)$ corresponde ao produto $p_{[A, B]} = x_2 \bar{x}_3$. Note que $p_{[A, B]}(X) = 1 \iff A \subseteq X \subseteq B$. Portanto, $p_{[A, B]}$ é a função característica do operador sup-gerador caracterizado pelo intervalo $[A, B]$. Os casos particulares são:

- o intervalo trivial $[A, A]$ que corresponde ao produto canônico dado por :

$$\sigma_i(A, A) = \begin{cases} x_i, & \text{se } w_i \in A, \\ \bar{x}_i, & \text{se } w_i \notin A. \end{cases}$$

- o intervalo $[A, E]$ que corresponde à erosão (neste caso nenhuma variável presente no produto é complementada)

$$\sigma_i(A, W) = \begin{cases} x_i, & \text{se } w_i \in A \\ ' , & \text{se } w_i \notin A \end{cases}$$

- e o intervalo $[\emptyset, B]$ que corresponde à anti-dilatação (neste caso todos as variáveis presentes no produto são complementadas)

$$\sigma_i(\emptyset, B) = \begin{cases} \bar{x}_i, & \text{se } w_i \notin B \\ ' ', & \text{se } w_i \in B. \end{cases}$$

Com estas regras, é possível, por exemplo, escrever a expressão Booleana ψ correspondente a um operador Ψ a partir de sua forma canônica. De fato, a expressão 2.6 pode ser simplificada para

$$\Psi = \bigvee \{ \Lambda_{(B, \bar{B})} : [B, \bar{B}] \subseteq \mathcal{K}_W(\Psi) \} \quad (2.13)$$

que corresponde a forma SSOP da função característica. A representação canônica em termos da base (Eq. 2.9) corresponde a soma de todos os implicantes primos.

2.4.2 Conversão Expressão-Intervalo

Para mapear um produto de uma expressão para o correspondente intervalo, o seguinte procedimento deve ser aplicado. Seja $p = \prod_{i=1}^n \sigma_i$, $\sigma_i \in \{x_i, \bar{x}_i, ' '\}$, um produto. Então, o intervalo $[A, B]$ associado a p é dado por $A = \{w_i : \sigma_i = x_i\}$ e $B = \{w_i : \sigma_i = x_i \text{ ou } \sigma_i = ' '\}$.

Com esta regra, qualquer expressão Booleana expressa como soma de produtos pode ser facilmente convertida para uma expressão morfológica expressa como supremo de sup-geradores.

2.4.3 Representações Associadas às Partições de $\mathcal{P}(W)$

Devido a equivalência entre representações Booleanas e representações morfológicas de W -operadores, para cada representação Booleana existe uma representação equivalente na representação morfológica e vice-versa. Aqui estudamos as representações que podem ser associadas às partições de $\mathcal{P}(W)$.

Seja $\{\mathcal{P}_i : i = 1, 2, \dots, k\}$ uma partição de $\mathcal{P}(W)$ (i.e., $\bigcup_{i=1}^k \mathcal{P}_i = \mathcal{P}(W)$ e $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ se $i \neq j$) e seja $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ uma função Booleana. Para cada i em $\{1, 2, \dots, k\}$ seja

$$\psi_i(X) = \begin{cases} \psi(X), & \text{se } X \in \mathcal{P}_i, \\ 0, & \text{se } X \notin \mathcal{P}_i. \end{cases} \quad (2.14)$$

Então,

$$\psi = \sum_{i=1}^k \psi_i.$$

Para obter a equivalente representação morfológica, seja Ψ_i o operador caracterizado pela função ψ_i , $i \in \{1, 2, \dots, k\}$. Então,

$$\Psi = \bigvee_{i=1}^k \Psi_i \quad (2.15)$$

e cada um dos operadores Ψ_i é um W -operador.

A representação acima é válida para o caso particular em que as partes da partição são intervalos de $\mathcal{P}(W)$. No caso da expansão de Shannon, as partes da partição são construídas sucessivamente bipartindo-se cada uma das partes, a partir de $\mathcal{P}(W)$. Portanto, ela equivale a um caso particular

desta representação, onde as variáveis de expansão definem as partições e os co-fatores definem as componentes ψ_i da decomposição. A seguir apresentamos uma nova decomposição de W-operadores, que é um caso particular da representação acima.

Proposição 2.28 *Seja $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ e seja $z \in E$. Se denotamos $\tau_z \Psi$ por Ψ_z , então $\Psi = \tau_{-z}(\iota_z \wedge \Psi_z) \vee \tau_{-z}(\nu_z \wedge \Psi_z)$.*

Dem.: A demonstração segue do fato de que $\Psi = \tau_{-z} \Psi_z$, $I = \iota \vee \nu$, e a distributividade de \vee . Ou seja,

$$\Psi = \tau_{-z} \Psi_z = \tau_{-z} (I \wedge \Psi)_z = \tau_{-z} [(\iota_z \wedge \Psi_z) \vee (\nu_z \wedge \Psi_z)].$$

■

Corolário 2.29 *Qualquer operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ pode ser expresso como $\Psi = (\iota \wedge \Psi) \vee (\nu \wedge \Psi)$.*

Dem.: Este é um caso particular da proposição 2.28, quando $z = o$. ■

Proposição 2.30 *Qualquer W-operador pode ser expresso unicamente como o supremo entre um operador anti-extensivo e um operador anti-extensivo complementar.*

Dem.: Seja $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$. Como $\iota \wedge \Psi \leq \iota$ e $\nu \wedge \Psi \leq \nu$, temos que $\iota \wedge \Psi$ é anti-extensivo e $\nu \wedge \Psi$ é anti-extensivo complementar. Portanto, do corolário 2.29 segue que existe uma decomposição de Ψ como supremo entre um operador anti-extensivo e um operador anti-extensivo complementar. A unicidade desta representação decorre do fato de que $(\iota \wedge \Psi) \wedge (\nu \wedge \Psi) = (\iota \wedge \nu) \wedge \Psi = O$. ■

Em particular, se a partição considerada na equação 2.15 for dada por $\mathcal{P}_1 = \{S \in \mathcal{P}(E) : o \notin S\}$ e $\mathcal{P}_2 = \{S \in \mathcal{P}(E) : o \in S\}$, então $\Psi = \Psi_1 \vee \Psi_2$ e, além disso, Ψ_1 é anti-extensivo complementar e Ψ_2 é anti-extensivo. Observe que Ψ_1 e Ψ_2 podem ser expressos de diversas formas.

Dualmente, qualquer operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ pode ser expresso como o ínfimo entre um operador extensivo e um operador extensivo complementar. Além disso, qualquer operador extensivo complementar pode ser escrito como o supremo entre um operador anti-extensivo e o operador complemento. A figura 2.5 ilustra estes conceitos.

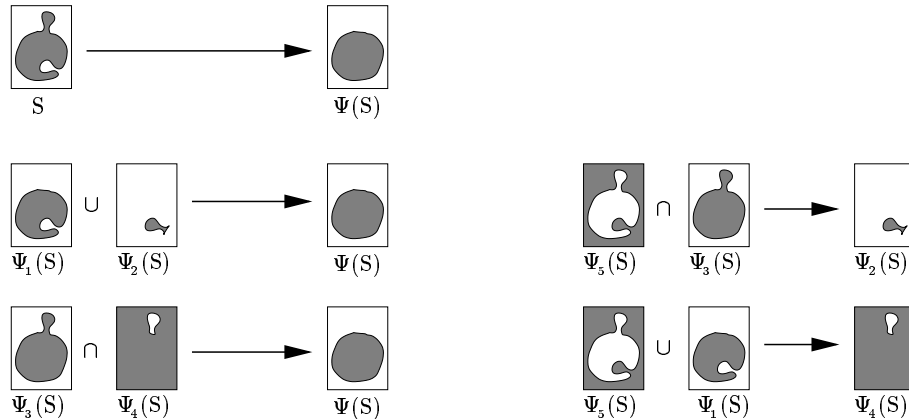


Figura 2.5: Decomposição do operador $\Psi : \Psi = \Psi_1 \cup \Psi_2 = \Psi_3 \cap \Psi_4$ ($\Psi_2 = \Psi_5 \cap \Psi_3$ e $\Psi_4 = \Psi_5 \cup \Psi_1$), onde Ψ_1 é anti-extensivo, Ψ_2 é anti-extensivo complementar, Ψ_3 é extensivo, Ψ_4 é extensivo complementar, e Ψ_5 é o complemento.

2.4.4 Resumo

No quadro a seguir, resumimos a correspondência entre elementos da representação Booleana e os da representação morfológica.

Representação morfológica	Representação Booleana
intervalo	cubo
operador sup-gerador	produto
erosão	produto sem literais complementadas
anti-dilatação	produto somente com literais complementadas
união de sup-geradores	soma de produtos
intervalos maximais do núcleo	implicantes primos
equação 2.13	SSOP
equação 2.9	soma de implicantes primos

Uma estrutura adequada para representar uma função Booleana expressa através de soma de produtos seria, portanto, uma coleção de intervalos onde cada intervalo corresponde a um produto da expressão.

2.5 Comentários

Para entender, dentro de um contexto mais abstrato, os resultados apresentados neste capítulo, convém lembrarmos que a morfologia matemática estuda mapeamentos entre reticulados completos. Como já mencionamos, a coleção $\mathcal{P}(E)$ munida com a relação usual de inclusão de conjuntos, \subseteq , forma um reticulado Booleano completo. Portanto, o conjunto de mapeamentos entre imagens binárias (conjuntos) pode ser entendido como um caso particular da morfologia matemática. Os resultados referentes à representação de operadores morfológicos num contexto mais abstrato podem ser encontrados em [10, 75, 148].

Em relação ao domínio das imagens, nesta tese nos restringimos ao plano digital Z^2 , porém a maior parte dos resultados a serem apresentados valem também para outras grades como as hexagonais. Informações sobre grades digitais podem ser encontradas por exemplo em [103].

Uma forma relativamente compacta para representar um operador é através dos intervalos maximais contidos no núcleo (ou, equivalentemente, os implicantes primos da função Booleana característica). Dado um operador nesta forma (que possui uma estrutura totalmente paralela), um problema bastante interessante e difícil é encontrar uma representação equivalente mais compacta (expressões curtas). Em geral, operadores podem ser representados mais compactamente quando expressos em estruturas sequenciais (composição de operadores mais simples). Para este último, alguns trabalhos relacionados podem ser encontrados, por exemplo, nas referências [71, 173, 136, 168, 172, 128, 63]. Existem também estudos sobre decomposição de funções Booleanas, isto é, sua expressão como uma composição de duas ou mais funções Booleanas definidas sobre um número menor de variáveis. Os trabalhos conhecidos nesta área tratam apenas com decomposições de certas formas bem específicas. Uma das decomposições bastante estudadas é a decomposição de Curtis/Ashenhurst e suas variantes [36, 131]. A representação de funções Booleanas através de estruturas como o BDD (Diagramas de Decisão Binária) [27] é utilizada para avaliar eficientemente o valor da função.

Capítulo 3

Projeto de W-Operadores

Neste capítulo apresentamos uma visão geral sobre o projeto de W-operadores a partir de exemplos obtidos de amostras de imagens observadas-ideais. Analisamos inicialmente os diversos aspectos do problema no contexto de aprendizado computacional. Em seguida analisamos as limitações das técnicas existentes para projeto de W-operadores. Por último discorremos sobre uma possível interpretação que permite entendermos quais devem ser as preocupações das técnicas que visam contornar as limitações.

3.1 Descrição do Problema

Seja Ψ_W o espaço de todos os W-operadores e seja $\mathcal{S} \subseteq \mathcal{P}(E) \times \mathcal{P}(E)$ uma coleção de pares de imagens binárias tal que, para todo $(S, I) \in \mathcal{S}$, S é uma imagem observada (i.e., uma imagem que desejamos processar) e I é a respectiva imagem ideal (o resultado que gostaríamos de obter após o processamento de S).

As imagens em \mathcal{S} são consideradas realizações de dois conjuntos aleatórios, \mathbf{S} e \mathbf{I} , *estacionários* com respeito à janela de observação W [43, 65]. Informalmente, isto significa que a probabilidade de um dado padrão (subconjunto) ser observado através de W em qualquer parte da imagem é a mesma. Mais ainda, consideramos que os conjuntos aleatórios \mathbf{S} e \mathbf{I} são *conjuntamente estacionários*, isto é, se denotamos por X os padrões observados nas imagens S e por y o valor observado na correspondente posição na respectiva imagem ideal I , a probabilidade de um par (X, y) ser observado é a mesma em qualquer ponto nas imagens (S, I) .

Conseqüentemente, podemos entender que a coleção de imagens \mathcal{S} é associada a dois processos aleatórios \mathbf{X} e \mathbf{y} caracterizados por uma distribuição de probabilidade conjunta $P(\mathbf{X}, \mathbf{y})$. As realizações de \mathbf{X} são subconjuntos $X \in \mathcal{P}(W)$ que obedecem a uma distribuição $P(\mathbf{X})$, e as realizações de \mathbf{y} são valores $y \in \{0, 1\}$. A probabilidade de se observar y em I , dado que um certo padrão X foi observado em S é dada pela probabilidade condicional $P(\mathbf{y}|\mathbf{X})$.

Estamos interessados em encontrar W-operadores que transformam as imagens observadas S no domínio \mathcal{S} para as respectivas imagens ideais I . Mais precisamente, desejamos projetar (escolher) um W-operador Ψ tal que $\Psi(S)$ seja o mais próximo possível de I em termos estatísticos. A noção de “proximidade estatística” será explicada mais adiante.

A seguir apresentamos o problema no contexto de aprendizado computacional, com o intuito de identificar os diversos aspectos que fazem parte do problema, a partir de noções bem estabelecidas naquela área de pesquisa.

3.2 O Problema Visto no Contexto de Aprendizado Computacional

O problema de aprendizado a partir de exemplos é um assunto vastamente estudado, com aplicações em diversas áreas. Este e outros assuntos correlatos são temas de pesquisa de uma área conhecida como *aprendizado computacional* (ou “Machine Learning”, em inglês). Alguns conceitos desta área de pesquisa são informalmente apresentados a seguir. Referências para este assunto podem ser encontrados, por exemplo, em [122, 105, 97, 3, 73, 161, 167]. Reconhecimento de padrões (por poderem, muitas vezes, ser vistos como problema de aprendizado) são também leituras recomendáveis [57, 61, 156, 102, 91]. O objetivo central aqui é inserir o problema de projeto de W-operadores neste contexto.

O aprendizado de interesse é o do tipo *supervisionado*, i.e., aquele no qual, para cada exemplo observado, é fornecida também a classificação do mesmo. Os sistemas de aprendizado analisam esses exemplos, chamados *exemplos de treinamento*, e procuram produzir uma função (classificador) capaz de classificar tanto os exemplos de treinamento quanto outros exemplos obtidos de um mesmo contexto da forma mais correta possível. A parte central de um sistema destes é construído, em geral, por um *algoritmo de aprendizado*, isto é, um algoritmo que analisa os exemplos de treinamento e seleciona uma função (classificador) dentre um grupo de candidatos.

Um problema de aprendizado é geralmente associado a um *espaço de conceitos*. Os conceitos podem possuir uma conotação abstrata e dependem do problema em questão. Por exemplo, no caso de processamento de imagens, um espaço de conceitos seria o espaço formado por todas as possíveis transformações de imagens. Para que estes conceitos possam ser manipulados formal e computacionalmente, eles precisam ser modelados. Os conceitos que podem ser expressos num determinado modelo constituem o *espaço de hipóteses*. Em nosso contexto, o espaço de hipóteses adotado é o espaço de W-operadores. Diferentes janelas W definem espaços de diferentes tamanhos.

Os elementos no espaço de hipóteses, ou simplesmente *hipóteses*, possuem uma representação que se conforma ao modelo formal adotado. No caso de W-operadores, como vimos no capítulo 2, existem diversas representações possíveis. Se as hipóteses admitem uma representação canônica, então a escolha de uma hipótese pode ser entendida como a escolha dos parâmetros desta representação canônica. Por exemplo, no caso de W-operadores, a especificação do núcleo ou da base é suficiente para especificar um W-operador.

Na prática, é comum o espaço de hipóteses ser definido em função da representação adotada, quando o ideal seria a escolha de uma representação capaz de expressar todos ou o maior número possível de conceitos. Um ponto importante que deve ser lembrado é que **um conceito que não pode ser representado não pode ser aprendido** [122].

Os algoritmos de aprendizado são aqueles que recebem como entrada um conjunto de exemplos de treinamento e escolhem uma hipótese no espaço de hipóteses que melhor representa o conceito associado aos dados de treinamento. Esse processamento é denominado *treinamento* ou *aprendizado*. A diversidade de representação das hipóteses pode abrir possibilidades para a utilização de diferentes algoritmos de aprendizado, que podem tirar proveito da particular representação adotada. Uma

importante característica desejável nos algoritmos de aprendizado é a *capacidade de generalização*, i.e., a capacidade de gerar hipóteses que classificam corretamente outros exemplos além daqueles observados durante o treinamento. Algumas das técnicas mais comumente utilizadas por algoritmos de aprendizado são, por exemplo, árvores de decisão [26], redes neurais [24, 74, 72] e algoritmos genéticos [126].

A escolha de uma hipótese que melhor representa o conceito associado aos dados de treinamento envolve a minimização de alguma medida de erro. Em geral, procura-se uma hipótese que seja o mais consistente possível com os dados de treinamento, isto é, hipóteses que classificam os dados de treinamento corretamente. Uma vez que uma hipótese é escolhida, sua avaliação é, em geral, realizada sobre um conjunto de *exemplos de validação*, que consistem de exemplos similares aos exemplos de treinamento, i.e., obtidos de um mesmo contexto. O erro da hipótese sobre os dados de validação fornece uma medida do seu desempenho.

Em geral, o processo de treinamento é um processo computacionalmente caro tanto em termos de tempo como em termos de espaço. A complexidade de espaço está ligado à representação das hipóteses consideradas e à quantidade de exemplos de treinamento. Portanto, podemos dizer que o espaço de hipóteses é limitado pelo espaço de memória disponível na máquina. Mais crítico, porém, é o tempo de processamento dos algoritmos de aprendizado. Em problemas de aprendizado consideram-se dois tipos de tempo: o *tempo de aprendizado* e o *tempo de aplicação*. O primeiro é o tempo necessário para processar os exemplos de treinamento e escolher uma hipótese no espaço de hipóteses; o segundo é o tempo necessário para aplicar a hipótese escolhida sobre uma instância do problema. Por exemplo, no caso de processamento de imagens, o tempo para projetar um operador não precisa necessariamente ser pequeno. No entanto, é desejável que o tempo para processar uma imagem (aplicar o operador projetado) seja pequeno, principalmente quando este está inserido num sistema de tempo real.

Existem vários outros tópicos importantes e interessantes tais como a questão da robustez do algoritmo de aprendizado, formas de se combinar hipóteses obtidas de diferentes algoritmos de aprendizado, complexidade de amostras, “PAC learning”, entre outros, que são estudados nesta área, mas que não serão abordados em nosso estudo.

Para concluir esta seção, notamos que o problema de projeto de W-operadores no contexto de aprendizado computacional pode ser caracterizado em termos de três elementos principais : um **espaço de hipóteses**, um **critério de avaliação** e uma **técnica de busca**. Conforme mencionamos na introdução, existem abordagens que são mais específicas (pois utilizam espaço de hipóteses muito restritivos ou técnicas de busca que dependem de características específicas das imagens) e outras que são mais genéricas.

3.3 Avaliação de W-operadores

A escolha de um operador pode estar associada a vários critérios. Um dos principais critérios é a qualidade do operador, isto é, o quão bem ele processa as imagens. Nesta seção apresentamos vários conceitos úteis para comparar operadores e para estabelecer um critério de escolha de um operador. Apresentamos a noção de operadores estatisticamente ótimos e tecemos alguns comentários sobre critérios que norteiam a escolha de um operador na prática. As noções e conceitos a serem apresentados aqui são um resumo de diversos trabalhos, tais como [39, 53, 46].

3.3.1 Comparações Simples

Primeiramente consideramos a situação em que dois W-operadores, Ψ_1 e Ψ_2 , caracterizados respectivamente pelas funções Booleanas ψ_1 e ψ_2 , precisam ser comparados. Listamos algumas formas de comparação.

Tamanho

O *tamanho* de um operador é a quantidade de dados ou de espaço necessários para o armazenamento ou representação do mesmo. O custo máximo para implementação de operadores em *hardware* é em geral bem definido. Este custo pode limitar, por exemplo, o número de portas lógicas ou o espaço que podem ser utilizados na placa de circuito.

Diferença Lógica

A *diferença lógica* entre dois W-operadores fornece uma medida que indica a diferença entre ambos quando os mesmos são comparados como funções lógicas. Ela é definida por:

$$\varepsilon_{log}[\Psi_1, \Psi_2] = \frac{|\mathcal{S}[\Psi_1, \Psi_2]|}{2^n} \quad (3.1)$$

onde $\mathcal{S}[\Psi_1, \Psi_2]$ é a *diferença simétrica* entre ψ_1 e ψ_2 dado por

$$\mathcal{S}[\Psi_1, \Psi_2] = \{X \in \mathcal{P}(W) : \psi_1(X) \neq \psi_2(X)\}. \quad (3.2)$$

Diferença Probabilística

A *diferença probabilística* entre Ψ_1 e Ψ_2 indica a probabilidade da diferença simétrica, i.e.,

$$\varepsilon_{prob}[\Psi_1, \Psi_2] = P(\mathcal{S}[\Psi_1, \Psi_2]) . \quad (3.3)$$

Com relação a esta medida, se $\psi_1(X) \neq \psi_2(X)$, mas se a probabilidade de observar X é desprezível, então o fato de que $\psi_1(X) \neq \psi_2(X)$ é estatisticamente inconsequente. A diferença probabilística indica se os elementos na diferença simétrica são probabilisticamente relevantes ou não. Ou seja, mesmo que dois operadores pareçam muito diferentes em sua construção algébrica, eles podem ser muito similares como operadores sobre uma classe de imagens de interesse.

3.3.2 W-operadores Estatisticamente Ótimos

O desempenho de um operador pode ser avaliado através de algumas medidas estatísticas que comparam os resultados por ele gerados com os resultados esperados.

Função de Perda

Seja $l : K \times K \rightarrow R$ uma função, denominada aqui de *função de perda*. O *erro médio*, ou o *risco*, de um W-operador Ψ segundo l e com relação ao domínio \mathcal{S} , denotado por $R_l(\Psi)$, é definido como

sendo a esperança da função de perda l calculada sobre \mathcal{S} . Isto é,

$$R_l\langle\Psi\rangle = E[l(\mathbf{y}, \psi(\mathbf{X}))],$$

onde (\mathbf{X}, \mathbf{y}) é o processo aleatório conjunto, com distribuição conjunta $P(\mathbf{X}, \mathbf{y})$, que caracteriza as imagens em \mathcal{S} .

Por exemplo, se a função de perda considerada é a diferença absoluta, $l(a, b) = |a - b|$, o risco é o **erro absoluto médio** (MAE, do inglês “mean absolute error”), dado por $MAE\langle\Psi\rangle = E[|\mathbf{y} - \psi(\mathbf{X})|]$, $\forall \Psi \in \Psi_W$.

A noção de otimalidade

Definição 3.1 (Operador Ótimo) Dizemos que $\Psi_* \in \Psi_W$ é ótimo em Ψ_W , segundo a função de perda l e com relação a \mathcal{S} , sse $R_l\langle\Psi_*\rangle \leq R_l\langle\Psi\rangle$, para todo $\Psi \in \Psi_W$.

Um operador ótimo em Ψ_W será denotado $\Psi_{\text{opt},W}$ enquanto um operador ótimo em um subespaço $\Psi_C \subset \Psi_W$ será denotado por $\Psi_{\text{opt},C}$. As referências à função de perda l e ao espaço \mathcal{S} (ou equivalentemente, ao processo conjunto (\mathbf{X}, \mathbf{y})) serão omitidas propositadamente daqui em diante, exceto nas situações em que a especificação explícita dos mesmos seja necessária.

Exemplo 3.2 O erro MAE de um W-operador Ψ é dado por :

$$\begin{aligned} MAE\langle\Psi\rangle &= E[|\mathbf{y} - \psi(\mathbf{X})|] \\ &= \sum_{(\mathbf{X}, \mathbf{y})} |y - \psi(\mathbf{X})| P(\mathbf{X}, y) \\ &= \sum_{(\mathbf{X}, \mathbf{y})} |y - \psi(\mathbf{X})| P(\mathbf{X}) P(y|\mathbf{X}) \\ &= \sum_{\mathbf{X}} P(\mathbf{X}) \sum_y |y - \psi(\mathbf{X})| P(y|\mathbf{X}) \\ &= \sum_{\mathbf{X}} P(\mathbf{X}) [\psi(\mathbf{X}) P(0|\mathbf{X}) + (1 - \psi(\mathbf{X})) P(1|\mathbf{X})] \end{aligned}$$

Podemos deduzir um W-operador MAE-ótimo em termos das probabilidades condicionais, minimizando cada um dos termos do somatório, i.e., minimizando a expressão

$$e(\mathbf{X}) = \psi(\mathbf{X}) P(0|\mathbf{X}) + (1 - \psi(\mathbf{X})) P(1|\mathbf{X})$$

para cada $\mathbf{X} \in \mathcal{P}(W)$. Como $\psi(\mathbf{X}) \in \{0, 1\}$, se tomarmos $\psi(\mathbf{X}) = 0$, então teremos $e(\mathbf{X}) = P(1|\mathbf{X})$; se tomarmos $\psi(\mathbf{X}) = 1$, então teremos $e(\mathbf{X}) = P(0|\mathbf{X})$. Portanto, a função característica do operador que minimiza o MAE é :

$$\psi_{\text{opt},W}(\mathbf{X}) = \begin{cases} 1, & \text{se } P(1|\mathbf{X}) > P(0|\mathbf{X}), \\ 0, & \text{se } P(1|\mathbf{X}) \leq P(0|\mathbf{X}). \end{cases} \quad (3.4)$$

Mais ainda, como $P(1|\mathbf{X}) + P(0|\mathbf{X}) = 1.0$, então $P(1|\mathbf{X}) > P(0|\mathbf{X})$ sse $P(1|\mathbf{X}) > 0.5$ (ou equivalentemente, $P(1|\mathbf{X}) \leq P(0|\mathbf{X})$ sse $P(1|\mathbf{X}) \leq 0.5$). Logo, $\psi_{\text{opt},W}$ pode ser reescrito como :

$$\psi_{\text{opt},W}(\mathbf{X}) = \begin{cases} 1, & \text{se } P(1|\mathbf{X}) > 0.5, \\ 0, & \text{se } P(1|\mathbf{X}) \leq 0.5. \end{cases} \quad (3.5)$$

O erro MAE de $\Psi_{\text{opt},W}$ é dado, portanto, por

$$MAE\langle \Psi_{\text{opt},W} \rangle = \sum_{\{X \in \mathcal{P}(W) : \psi_{\text{opt},W}(X)=0\}} P(X) P(1|X) + \sum_{\{X \in \mathcal{P}(W) : \psi_{\text{opt},W}(X)=1\}} P(X) P(0|X)$$

Precisão de um W-operador

Definição 3.3 (Aumento de Erro ou Precisão) O aumento de erro de um W-operador Ψ em relação ao operador ótimo $\Psi_{\text{opt},W}$, é dado por:

$$\Delta(\Psi, \Psi_{\text{opt},W}) = R\langle \Psi \rangle - R\langle \Psi_{\text{opt},W} \rangle \quad (3.6)$$

O inverso desta diferença, $1/\Delta(\Psi, \Psi_{\text{opt},W})$, é denominado a precisão de Ψ .

Note que $\Delta(\Psi, \Psi_{\text{opt},W}) \geq 0$, pois $\Psi_{\text{opt},W}$ possui o menor risco entre todos os W-operadores. O desempenho de um operador é melhor quanto menor é o seu aumento de erro (ou equivalentemente, quanto maior é a sua precisão).

No caso particular do erro MAE, podemos reescrever o aumento de erro de Ψ como :

$$\begin{aligned} \Delta(\Psi, \Psi_{\text{opt},W}) &= MAE\langle \Psi \rangle - MAE\langle \Psi_{\text{opt},W} \rangle \\ &= \sum_X P(X) [\psi(X) P(0|X) + (1 - \psi(X)) P(1|X)] \\ &\quad - \sum_X P(X) [\psi_{\text{opt},W}(X) P(0|X) + (1 - \psi_{\text{opt},W}(X)) P(1|X)] \\ &= \sum_X P(X) [(\psi(X) - \psi_{\text{opt},W}(X)) P(0|X) + (\psi_{\text{opt},W}(X) - \psi(X)) P(1|X)] \end{aligned}$$

Seja $c(X)$ o aumento de erro devido a X , para cada $X \in \mathcal{P}(W)$. Então,

$$\begin{aligned} c(X) &= P(X) [(\psi(X) - \psi_{\text{opt},W}(X)) P(0|X) + (\psi_{\text{opt},W}(X) - \psi(X)) P(1|X)] \\ &= \begin{cases} P(X) [P(0|X) - P(1|X)], & \text{se } \psi_{\text{opt},W}(X) = 0 \text{ e } \psi(X) = 1, \\ P(X) [P(1|X) - P(0|X)], & \text{se } \psi_{\text{opt},W}(X) = 1 \text{ e } \psi(X) = 0, \\ 0, & \text{se } \psi_{\text{opt},W}(X) = \psi(X). \end{cases} \end{aligned}$$

Usando o fato de que $P(0|X) = 1 - P(1|X)$, temos que $P(0|X) - P(1|X) = 1 - P(1|X) - P(1|X) = 1 - 2P(1|X)$. Logo, se $\psi_{\text{opt},W}(X) = 0$, então como $P(0|X) \geq P(1|X)$, temos que $P(0|X) - P(1|X) \geq 0$ e portanto $P(0|X) - P(1|X) = |2P(1|X) - 1|$. Similarmente, se $\psi_{\text{opt},W}(X) = 1$, então $P(1|X) > P(0|X)$ e $P(1|X) - P(0|X) > 0$, e portanto, $P(1|X) - P(0|X) = P(1|X) - 1 + P(1|X) = 2P(1|X) - 1 = |2P(1|X) - 1|$. Portanto, a equação acima pode ser simplificada para

$$\Delta(\Psi, \Psi_{\text{opt},W}) = \sum_{\{X : \psi_{\text{opt},W}(X) \neq \psi(X)\}} |2P(1|X) - 1| P(X) \quad (3.7)$$

Analisando esta equação, podemos concluir que se o valor de $P(1|X)$ é muito próximo de 0.5 ou se o valor de $P(X)$ é muito próximo de 0, então X não contribui muito para $\Delta(\Psi, \Psi_{\text{opt},W})$.

X	$P(1 X)$	$P(X)$	$\psi_{\text{opt,W}}$	ψ_1	ψ_2
000	0.3	0.13	0	0	0
001	0	0.12	0	0	0
010	0.8	0.08	1	1	0
011	0.65	0.10	1	0	1
100	0.12	0.16	0	0	0
101	a	0.10	0	1	0
110	1	0.11	1	1	1
111	b	0.20	1	1	0

Tabela 3.1: Probabilidades e três operadores.

Exemplo 3.4 Para ilustrar alguns dos conceitos apresentados acima, apresentamos um exemplo para uma janela de 3 pontos. Supomos que as probabilidades $P(X)$ e $P(1|X)$ são conhecidas, e portanto também o operador MAE-ótimo, $\psi_{\text{opt,W}}$ (tabela 3.1). Dados dois operadores, ψ_1 e ψ_2 (também na tabela 3.1), vamos fazer algumas comparações.

A diferença lógica entre Ψ_1 e Ψ_2 apenas indica o quão parecidos/diferentes eles são como funções lógicas, portanto não representa um critério de escolha. A título de curiosidade, calculamos a diferença lógica de Ψ_1 e Ψ_2 , com relação a $\Psi_{\text{opt,W}}$.

$$\varepsilon_{\log}[\Psi_1, \Psi_{\text{opt,W}}] = \frac{\#\{101\}}{8} = 0.125$$

$$\varepsilon_{\log}[\Psi_2, \Psi_{\text{opt,W}}] = \frac{\#\{111\}}{8} = 0.125$$

Estas medidas podem indicar qual dos dois operadores mais se assemelha logicamente com o operador ótimo.

Se estivermos interessados em medidas estatísticas, podemos calcular a diferença probabilística. A diferença probabilística entre Ψ_1 e Ψ_2 fornece uma indicação de como eles diferem em termos estatísticos, mas não indica qual deles é o melhor. Para isso, precisamos calcular a diferença em relação a $\Psi_{\text{opt,W}}$.

$$\varepsilon_{\text{prob}}[\Psi_1, \Psi_{\text{opt,W}}] = P(\{101\}) = 0.1$$

$$\varepsilon_{\text{prob}}[\Psi_2, \Psi_{\text{opt,W}}] = P(\{111\}) = 0.2$$

Se levarmos em consideração a diferença probabilística, Ψ_1 é melhor que Ψ_2 . Agora calculemos o aumento de erro de Ψ_1 e Ψ_2 em função das probabilidades $P(y = 1|101) = a$ e $P(y = 1|111) = b$. Note que devemos ter $a \leq 0.5$ e $b > 0.5$, pois $\psi_{\text{opt,W}}$ é o W-operador MAE-ótimo.

$$\Delta(\Psi_1, \Psi_{\text{opt,W}}) = P(101)|2a - 1| = 0.1 - 0.2a$$

$$\Delta(\Psi_2, \Psi_{\text{opt,W}}) = P(111)|2b - 1| = 0.4b - 0.2$$

O aumento de erro $\Delta(\Psi_1, \Psi_{\text{opt,W}})$ envolve a variável a , enquanto $\Delta(\Psi_2, \Psi_{\text{opt,W}})$ envolve a variável b . Portanto, se $0.1 - 0.2a < 0.4b - 0.2$ então podemos concluir que Ψ_1 é um operador mais preciso que Ψ_2 e se $0.1 - 0.2a > 0.4b - 0.2$ então Ψ_2 é mais preciso que Ψ_1 . Se $0.1 - 0.2a = 0.4b - 0.2$ então eles são igualmente precisos.

Neste exemplo, mostramos que dois operadores com mesma diferença lógica podem apresentar diferenças probabilísticas distintas com relação ao operador ótimo. Mais ainda, aquele com diferença probabilística maior não é necessariamente mais preciso; ele pode ser menos preciso ou, então, ambos podem ser igualmente precisos.

3.3.3 Subotimalidade

Conforme vimos, a escolha de um operador pode levar em conta diferentes medidas de desempenho. Mais do que isso, ela pode também considerar outros critérios onde algumas restrições tais como o tamanho máximo do operador, as propriedades que devem ser satisfeitas pelo operador ou tempo e espaço (memória) necessários para o projeto são impostos. Um conjunto de restrições define um novo espaço de hipóteses que pode ser um subconjunto do espaço de hipóteses original. Analisamos qual a relação entre o operador ótimo no espaço original e no subespaço.

Seja $\Psi_C \subset \Psi_W$. Um operador $\Psi_{\text{opt},C} \in \Psi_C$ é sub-ótimo em Ψ_W com relação ao subespaço Ψ_C sse $\Psi_{\text{opt},C}$ é um operador ótimo em Ψ_C . Da mesma forma que a noção de ótimo é associada a um espaço, a noção de subotimalidade é associada a um subespaço.

Para qualquer $\Psi \in \Psi_C$, o risco de Ψ , $R(\Psi)$, pode ser expresso em termos do risco do operador ótimo, da seguinte forma :

$$R(\Psi) = R(\Psi_{\text{opt},W}) + \Delta(\Psi, \Psi_{\text{opt},W}). \quad (3.8)$$

Se denotamos o operador ótimo em Ψ_C por $\Psi_{\text{opt},C}$, então $R(\Psi_{\text{opt},C}) \leq R(\Psi)$, para todo $\Psi \in \Psi_C$. Logo, $\forall \Psi \in \Psi_C$,

$$R(\Psi_{\text{opt},W}) + \Delta(\Psi_{\text{opt},C}, \Psi_{\text{opt},W}) \leq R(\Psi_{\text{opt},W}) + \Delta(\Psi, \Psi_{\text{opt},W}) \quad (3.9)$$

isto é,

$$\Delta(\Psi_{\text{opt},C}, \Psi_{\text{opt},W}) \leq \Delta(\Psi, \Psi_{\text{opt},W}). \quad (3.10)$$

Das equações 3.8 e 3.10 segue que :

Um operador $\Psi_{\text{opt},C} \in \Psi_C$ é sub-ótimo em Ψ_W ($\Psi_C \subset \Psi_W$) se, e somente se, é ótimo em Ψ_C ou, equivalentemente, se possui o menor aumento de erro em relação a $\Psi_{\text{opt},W}$.

Por exemplo, se $W_1 \subseteq W_2$, então um operador ótimo no espaço dos W_1 -operadores é subótimo no espaço dos W_2 -operadores.

3.4 Projeto baseado no modelo de aprendizado PAC

Descrevemos aqui uma técnica independente de contexto originalmente proposta em [158]. Ela é baseada no modelo de aprendizado PAC (Probably Approximately Correct) generalizado conforme mencionamos na introdução desta tese. O que descrevemos aqui é o procedimento computacional desta técnica e não seus fundamentos teóricos (estes podem ser encontrados no trabalho original [158] ou em referências específicas [160, 73, 97, 98]). O procedimento consiste de três etapas principais conforme ilustradas na figura 3.1 e explicadas em seguida.

1. **Estatística** – estimação das probabilidades condicionais

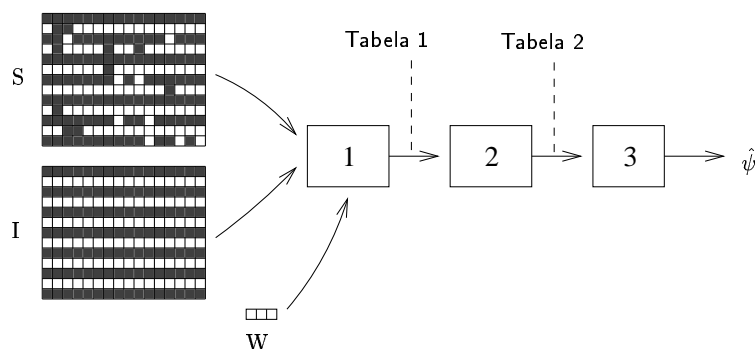










Figura 3.1: Procedimento computacional baseado no modelo PAC para o aprendizado de W-operadores.

Nesta etapa, coletam-se exemplos a partir das imagens de treinamento, percorrendo-se uma janela W sobre as imagens. Para cada pixel da imagem observada, recorta-se o padrão X sob a janela W e anota-se o correspondente valor y na imagem ideal. Este procedimento dá origem a uma tabela, cujas entradas são padrões X e a quantidade de vezes que os mesmos foram observados associados ao valor 1, ou ao valor 0, na respectiva imagem ideal. A partir destes dados, pode-se estimar $P(X)$ e $P(1|X)$ para cada um dos padrões observados. Para o exemplo da figura 3.1 obtemos a seguinte tabela (Tabela 1):

X	freq. de 0	freq. de 1
	61	0
	5	0
	6	2
	2	6
	8	0
	0	8
	2	7
	0	75

2. Decisão - classificação que minimiza o erro

Para cada padrão X observado na etapa 1 atribuímos uma classificação 0 ou 1, dependendo da função de perda em questão. Se consideramos o erro MAE, de acordo com a equação 3.5 devemos fazer $\hat{\psi}(X) = 1$ se $P(1|X) > 0.5$ e $\hat{\psi}(X) = 0$ caso contrário, significando que o operador a ser projetado deverá atribuir classificação $\hat{\psi}(X)$ para o padrão X. Para o exemplo da figura 3.1 obtemos as seguintes classificações:

X	$\hat{\psi}(X)$
	0
	0
	0
	1
	0
	1
	1
	1

3. Especificação do operador – generalização da classificação e minimização de representação.

Se denotarmos o operador a ser projetado por $\hat{\psi}$, podemos dizer que após a etapa de decisão os dados estão organizados como duas listas: uma lista de elementos que pertencem a $\hat{\psi}\langle 1 \rangle$ e outra lista de elementos de $\hat{\psi}\langle 0 \rangle$. Os padrões não observados durante a etapa de estatística não aparecem em nenhuma dessas duas listas.

Dois objetivos precisam ser cumpridos nesta etapa : (1) generalização da classificação (ver seção 3.2) e (2) especificação de um operador. Os algoritmos utilizados nesta etapa são geralmente denominados algoritmos de aprendizado. Em nossas implementações utilizamos um algoritmo denominado ISI, o qual será descrito em detalhes no capítulo 4; este algoritmo devolve uma função Booleana representada por uma coleção de intervalos e que é consistente com as classificações da segunda etapa (e por isto é um algoritmo PAC, no sentido generalizado). A vantagem de representar os operadores como funções Booleanas é a facilidade de mapeá-los para a representação morfológica e vice-versa, conforme vimos no capítulo 2.

No exemplo da figura 3.1, todos os padrões são observados e portanto não ocorre generalização. Uma coleção de intervalos que caracteriza $\hat{\psi}\langle 1 \rangle$ é a coleção $\{0XX, X0X, XX0\}$.

O procedimento descrito acima é independente de contexto pois não está restrito a um particular tipo de problema (filtragem, segmentação, reconhecimento de padrões geométricos, etc) ou imagens. Além disso, ele é bastante genérico pois pode ser utilizado para projetar W-operadores para janelas arbitrariamente grandes. Este é o procedimento utilizado ao longo deste texto para projetar W-operadores a partir de exemplos quando nenhum outro conhecimento é considerado. Será referenciado como *treinamento padrão*.

3.5 Limitações

Vimos na introdução que as técnicas dependentes de contexto são naturalmente restritivas pois só se aplicam a uma determinada subclasse de problemas, enquanto uma técnica genérica (quando considera um espaço grande de operadores) pode esbarrar em problemas de imprecisão dos operadores projetados e de complexidade de tempo dos algoritmos de aprendizado. A seguir analisamos formalmente a questão da precisão dos operadores projetados.

Quando projetamos um operador ótimo em Ψ_W a partir de probabilidades estimadas, projetamos na verdade um estimador do operador ótimo, o qual denotamos aqui por $\hat{\Psi}_{\text{opt},W}$. Isto significa que existe um aumento de erro $\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})$ associado a $\hat{\Psi}_{\text{opt},W}$, ou seja, que

$$R\langle \hat{\Psi}_{\text{opt},W} \rangle = R\langle \Psi_{\text{opt},W} \rangle + \Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W}). \quad (3.11)$$

Se $\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})$ é pequeno, então podemos dizer que o operador projetado é quase ótimo. O que acontece em geral, quando consideramos janelas grandes, é que o aumento de erro $\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})$ é significativo.

Consideremos agora um subespaço $\Psi_C \subset \Psi_W$ e o problema de projetar um operador ótimo $\hat{\Psi}_{\text{opt},C}$ em Ψ_C . Como anteriormente, podemos escrever

$$R\langle \hat{\Psi}_{\text{opt},C} \rangle = R\langle \Psi_{\text{opt},C} \rangle + \Delta(\hat{\Psi}_{\text{opt},C}, \Psi_{\text{opt},C}). \quad (3.12)$$

Tanto $\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})$, como $\Delta(\hat{\Psi}_{\text{opt},C}, \Psi_{\text{opt},C})$, são variáveis aleatórias pois dependem de uma amostra de treinamento. Consequentemente, podemos falar em *aumento de erro médio*, $E[\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})]$ e $E[\Delta(\hat{\Psi}_{\text{opt},C}, \Psi_{\text{opt},C})]$. Combinando as equações 3.11 e 3.12, podemos ver que projetar um operador ótimo em um subespaço Ψ_C pode ser mais vantajoso do que projetar um ótimo em Ψ_W se

$$\Delta(\Psi_{\text{opt},C}, \Psi_{\text{opt},W}) + E[\Delta(\hat{\Psi}_{\text{opt},C}, \Psi_{\text{opt},C})] \leq E[\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})], \quad (3.13)$$

apesar de $\Psi_{\text{opt},C}$ ser subótimo em Ψ_W .

A equação 3.13 significa que o erro de estimação de $\Psi_{\text{opt},W}$ é muito grande, tão grande que é maior do que a soma do aumento de erro de $\Psi_{\text{opt},C}$ em relação a $\Psi_{\text{opt},W}$, $\Delta(\Psi_{\text{opt},C}, \Psi_{\text{opt},W})$, e o erro de estimação de $\Psi_{\text{opt},C}$.

As equações apresentadas acima não podem ser utilizadas na prática pois elas dependem do conhecimento da verdadeira distribuição de probabilidade $P(\mathbf{X}, \mathbf{y})$. No entanto, elas são úteis para mostrar que a precisão das estimações pode afetar a precisão do operador projetado.

Analizamos aqui um dos gráficos apresentados no capítulo de introdução desta tese, destacando os conceitos discutidos nesta seção. Sejam W_1, W_2 duas janelas tais que $W_1 \subset W_2$. Neste caso, $\Psi_{W_1} \subset \Psi_{W_2}$. O gráfico da figura 3.2 compara o desempenho de operadores projetados nestes dois espaços, para diferentes números de exemplos de treinamento. Sejam $\hat{\Psi}_{\text{opt},W_1}$ e $\hat{\Psi}_{\text{opt},W_2}$ os operadores projetados em Ψ_{W_1} e Ψ_{W_2} , respectivamente. Para uma quantidade pequena de exemplos de treinamento temos $MAE\langle \hat{\Psi}_{\text{opt},W_1} \rangle < MAE\langle \hat{\Psi}_{\text{opt},W_2} \rangle$. A medida que a quantidade de exemplos aumenta, a diferença entre ambos diminui, até que a situação se inverte. Em termos da equação 3.13 podemos inferir que, para quantidades pequenas de exemplos de treinamento, $E[\Delta(\hat{\Psi}_{\text{opt},W_1}, \Psi_{\text{opt},W_1})]$ é pequeno em relação a $E[\Delta(\hat{\Psi}_{\text{opt},W_2}, \Psi_{\text{opt},W_2})]$, de forma que $E[\Delta(\hat{\Psi}_{\text{opt},W_2}, \Psi_{\text{opt},W_2})] > \Delta(\Psi_{\text{opt},W_1}, \Psi_{\text{opt},W_2}) + E[\Delta(\hat{\Psi}_{\text{opt},W_1}, \Psi_{\text{opt},W_1})]$. A medida que o número de exemplos de treinamento aumenta, $E[\Delta(\hat{\Psi}_{\text{opt},W_2}, \Psi_{\text{opt},W_2})]$ tende a diminuir e a partir de um dada quantidade de exemplos temos $E[\Delta(\hat{\Psi}_{\text{opt},W_2}, \Psi_{\text{opt},W_2})] < \Delta(\Psi_{\text{opt},W_1}, \Psi_{\text{opt},W_2}) + E[\Delta(\hat{\Psi}_{\text{opt},W_1}, \Psi_{\text{opt},W_1})]$.

Todas as abordagens que utilizam exemplos de treinamento, mesmo que não usem diretamente as probabilidades estimadas, dependem da expressividade estatística dos exemplos de treinamento. Em outras palavras, quando a quantidade de dados de treinamento é pequena em relação ao tamanho do espaço de hipóteses, em geral não se obtém bons resultados. Existem vários trabalhos que investigam este assunto tanto empírica [62, 135, 38, 68] quanto teoricamente [3, 73, 97, 160, 161].

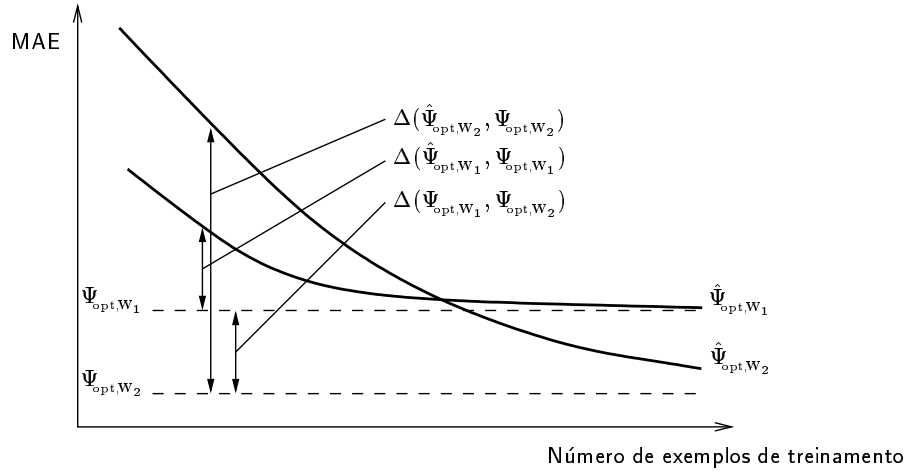


Figura 3.2: Erro de operadores projetados em espaço de tamanhos diferentes, para diferentes números de exemplos de treinamento.

3.6 Como Contornar as Limitações

De forma simplificada, podemos dizer que o problema de projeto de W-operadores consiste basicamente em escolhermos um operador ótimo no espaço de hipóteses considerado. Logo, se escolhermos um operador qualquer ao acaso, a probabilidade do operador ótimo ser escolhido é inversamente proporcional ao tamanho do espaço de hipóteses. Neste caso, quanto maior o espaço de hipóteses, menor é esta probabilidade. Para aumentar a probabilidade de escolhermos um operador ótimo devemos reduzir o espaço de possíveis escolhas. É importante notarmos que esta redução deve ser feita de forma adequada, isto é, de forma que o espaço resultante contenha o operador ótimo, ou pelo menos operadores bastante precisos (i.e., estatisticamente próximos do operador ótimo).

Equivalentemente, o problema de projetar W-operadores pode também ser entendido como um problema que consiste em definir corretamente a tabela-verdade (isto é, definir qual valor deve ser associado a cada uma das linhas da tabela). A seguir analisamos como conhecimentos sobre o problema que desejamos resolver podem ser utilizados para reduzir o espaço de possíveis candidatos.

Seja um espaço de hipóteses como o esquematizado na figura 3.3. Na ausência de qualquer conhecimento sobre o problema a ser resolvido, qualquer operador nesse espaço pode ser visto como um candidato. Em termos da função característica, podemos dizer que a tabela-verdade pode ser preenchida de qualquer forma.

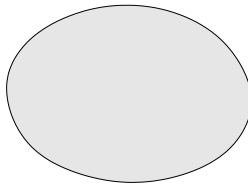


Figura 3.3: Os candidatos são todos os operadores.

A restrição do espaço de hipóteses a um subespaço naturalmente reduz o espaço de candidatos. Para que esta redução seja adequada, ela deve estar baseada em conhecimentos sobre o operador desejado. Por exemplo, o subespaço de operadores que satisfazem alguma propriedade algébrica pode ser considerado mediante o conhecimento de que o operador desejado satisfaz a mesma propriedade. Esquematicamente, podemos ilustrar esta idéia conforme a figura 3.4. Em termos da tabela-verdade da função característica, o fato do operador desejado possuir uma certa propriedade implica que podem existir certas regras para o preenchimento da tabela. Na página 15, podemos ver que a característica do núcleo define as regras para algumas classes de operadores.

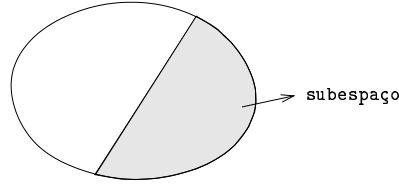


Figura 3.4: Os candidatos são os operadores no subespaço.

Mesmo que a redução do espaço de hipóteses não seja totalmente adequada, o operador projetado no subespaço pode ser mais preciso do que o operador projetado no espaço original devido a precisão da estimação estatística (no sentido da equação 3.13). Além disso, sob o ponto de vista de complexidade de tempo, as buscas podem ser mais rápidas no subespaço. As técnicas podem também explorar as particulares estruturas do espaço decorrentes das restrições impostas. Mais ainda, os algoritmos de aprendizado podem aproveitar informações sobre o problema que desejamos resolver para gerar generalizações mais precisas.

Um outro tipo de conhecimentos que podem ser explorados são aqueles referentes às imagens. Conhecimentos sobre a distribuição conjunta $P(\mathbf{X}, \mathbf{y})$ ou sobre características específicas das imagens podem ser levadas em consideração. Os exemplos de treinamento podem ser vistos como conhecimentos deste tipo. Para os algoritmos de aprendizado que selecionam um operador consistente com os exemplos de treinamento, a ausência de exemplos de treinamento implica que o espaço de candidatos é o próprio espaço de hipóteses; se apenas uma pequena quantidade de exemplos é observada, então o espaço de candidatos tende a ser bem grande enquanto um grande número de exemplos de treinamento implica poucos candidatos possíveis. Em termos de função característica, podemos pensar em tabelas-verdades cujo valor da função é conhecido para uma parcela das linhas, parcelas estas que podem ser pequenas ou grandes. Esta idéia pode ser esquematizada conforme ilustração da figura 3.5.

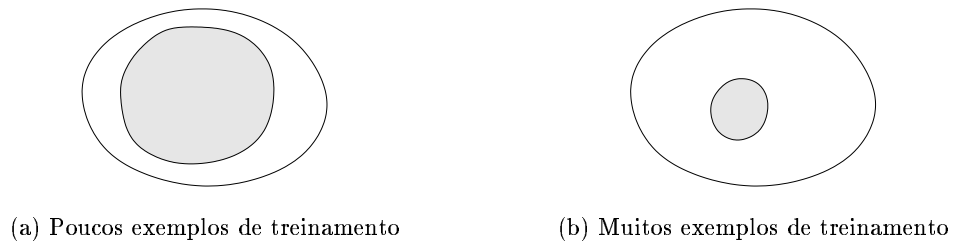


Figura 3.5: Os candidatos são os operadores consistentes com os exemplos de treinamento.

De uma forma geral, quando estas duas formas para reduzir o espaço de candidatos são conjugadas, tem-se um espaço menor de candidatos. Esta idéia pode ser esquematizada conforme ilustrações da figura 3.6.

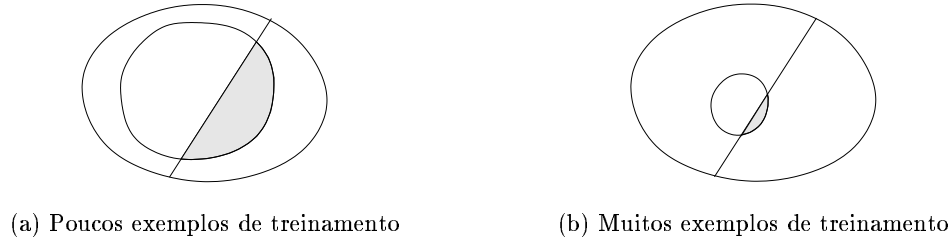


Figura 3.6: Os candidatos são os operadores no subespaço que são consistentes com os exemplos de treinamento.

A partir das considerações acima, entendemos que as soluções para contornar o problema de precisão dos operadores devem, portanto, considerar fundamentalmente abordagens para reduzir o tamanho do espaço de candidatos de forma adequada (ou, equivalentemente, preencher a maior parte possível da tabela de forma correta). O conhecimento do problema que desejamos resolver, seja com respeito ao operador ou às imagens, parecem ser fundamentais para isto.

3.7 Comentários

Em resumo, podemos dizer que projetar operadores precisos em espaços grandes (aqueles que dependem de uma janela W relativamente grande) a partir de uma quantidade limitada de exemplos de treinamento é uma das questões mais importantes sendo investigadas atualmente.

Neste capítulo apresentamos uma visão geral sobre o projeto de operadores morfológicos, enfatizando a questão sobre a precisão dos operadores projetados. Na última seção mostramos como os conhecimentos sobre o problema a ser resolvido, modelados no espaço dos operadores ou das imagens, podem ser vistos como uma potencial forma para aumentar a precisão do operador projetado.

Ao analisarmos os trabalhos existentes, verificamos que diferentes algoritmos (*algoritmos genéticos* [70, 159, 169]; *redes neurais* [166, 76, 5, 6]; *árvores de decisão binária* [86, 99]; *técnica de k -vizinhos mais próximos* [99]; *técnicas de programação linear* [33, 155]; *algoritmos de minimização lógica* [157, 15]), técnicas “fuzzy”, técnicas baseadas em multi-resolução [49, 67, 7, 30, 95, 90, 124, 129], projeto multi-estágio [144, 133, 171], entre outros, vem sendo aplicados. A visão geral apresentada neste capítulo, juntamente com a interpretação da seção anterior, podem ser úteis para entendermos as limitações dessas técnicas e, conseqüentemente, orientar a escolha por uma técnica adequada para um determinado problema.

Uma questão também importante com relação aos operadores projetados é a implementação eficiente dos mesmos. A implementação depende da arquitetura do computador considerado. Este assunto está além do escopo deste trabalho. Algumas considerações podem ser encontradas em [137, 113, 71].

Capítulo 4

O Algoritmo ISI

A noção de funções Booleanas e diversas formas de representação foram recordadas na seção 2.3. Neste capítulo revemos minimização de funções Booleanas e apresentamos novos resultados sobre o algoritmo ISI. Propomos e analisamos heurísticas para diminuir o tempo de processamento do ISI e analisamos a utilização do ISI como algoritmo de aprendizado para resolver problemas de processamento de imagens binárias. Alguns resultados comparativos de tempo de processamento, número de intervalos e capacidade de generalização são ilustrados através de gráficos.

4.1 Minimização de Funções Booleanas

Conforme mencionamos no capítulo 2, a minimização de funções Booleanas, i.e., a sua representação por expressões equivalentes minimais, é um assunto bastante estudado [77, 32, 120]. Uma das formas minimais mais estudadas é a forma minimal de soma de produtos.

Definição 4.1 (Expressão minimal) *Uma expressão escrita como soma de produtos é uma expressão minimal (ou MSOP) se (1) não existe outra expressão equivalente envolvendo um menor número de produtos, e (2) não existe nenhuma outra expressão equivalente envolvendo o mesmo número de produtos mas um número menor de literais.*

Sabemos que todos os produtos de uma função Booleana na forma MSOP são necessariamente implicantes primos. Portanto, para determinar a forma MSOP de uma função Booleana podemos primeiramente calcular todos os seus implicantes primos e em seguida selecionar, dentre todas as subcoleções de implicantes primos, uma que satisfaz a condição (2) da definição 4.1.

Seja $[A, B] \subseteq \mathcal{P}(W)$ um intervalo e seja $X \in \mathcal{P}(W)$. Dizemos que $[A, B]$ cobre X ou que X é coberto por $[A, B]$ se $X \in [A, B]$. A coleção de todos os elementos cobertos por uma coleção de intervalos \mathbf{I} é denotado por $\bigcup \mathbf{I}$, i.e., $\bigcup \mathbf{I} = \{X \in \mathcal{P}(W) : X \in [A, B], [A, B] \in \mathbf{I}\}$. Dizemos que uma coleção de intervalos \mathbf{I} de $\mathcal{P}(W)$ é uma cobertura de \mathcal{X} , $\mathcal{X} \subseteq \mathcal{P}(W)$, se existe, para todo $X \in \mathcal{X}$, um intervalo $[A, B] \in \mathbf{I}$ que cobre X .

Uma cobertura \mathbf{I} de \mathcal{X} é uma cobertura mínima se (1) não existe uma cobertura \mathbf{I}' de \mathcal{X} , $\mathbf{I} \neq \mathbf{I}'$ tal que $|\mathbf{I}'| < |\mathbf{I}|$ e (2) não existe nenhuma outra cobertura de \mathcal{X} possuindo a mesma cardinalidade de \mathbf{I} com intervalos de dimensão maior. A cobertura mínima não é necessariamente única. Uma

cobertura \mathbf{I} de \mathcal{X} é *minimal* se não existe uma subcoleção \mathbf{I}' própria de \mathbf{I} que também seja uma cobertura de \mathcal{X} . Uma cobertura mínima é necessariamente minimal.

Definição 4.2 (Coleção de intervalos maximais) *Seja \mathbf{I} uma coleção de intervalos de $\mathcal{P}(W)$. Um intervalo $[A, B] \in \mathbf{I}$ é maximal em \mathbf{I} se não existe outro intervalo $[A', B'] \in \mathbf{I}$ tal que $[A, B] \subset [A', B']$.*

Seja Max uma operação que determina todos os intervalos maximais de uma coleção de intervalos. O conjunto de todos os intervalos maximais contidos em \mathcal{X} ($\mathcal{X} \subseteq \mathcal{P}(W)$) denotado por $\mathbf{M}(\mathcal{X})$, é definido por :

$$\mathbf{M}(\mathcal{X}) = Max\{[A, B] : [A, B] \subseteq \mathcal{X}\}.$$

Por exemplo, $\mathbf{M}(\mathcal{P}(W)) = \{[\emptyset, W]\}$. No exemplo da figura 4.1, $\mathbf{M}(\{000, 001, 011, 111\}) = \{00X, 0X1, X11\}$ é o conjunto de implicantes primos. Note que o intervalo $0X1$ não é necessário para cobrir $\psi(1)$ pois este é coberto por $\{00X, X11\}$.

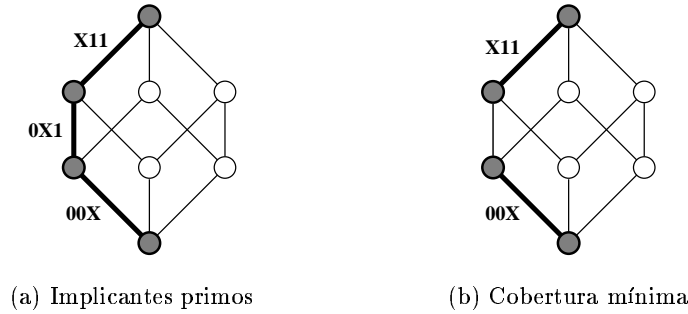


Figura 4.1: Implicantes primos e cobertura mínima.

Definimos uma relação \leq entre coleções de intervalos maximais da seguinte forma: sejam \mathbf{X} e \mathbf{Y} duas coleções de intervalos maximais, então

$$\mathbf{X} \leq \mathbf{Y} \iff \forall [A, B] \in \mathbf{X}, \exists [A', B'] \in \mathbf{Y} : [A, B] \subseteq [A', B'].$$

A relação \leq é uma relação de ordem parcial (i.e., reflexiva, anti-simétrica e transitiva).

Propriedade 4.3 *As seguintes propriedades são válidas:*

$P1 : Max(Max(\mathbf{I})) = Max(\mathbf{I})$ para qualquer coleção de intervalos \mathbf{I} .

$P2 : \bigcup \mathbf{M}(\mathcal{X}) = \mathcal{X}, \forall \mathcal{X} \subseteq \mathcal{P}(W)$.

$P3 : Para todo [A, B] \subseteq \mathcal{X}, existe [A', B'] \in \mathbf{M}(\mathcal{X}) tal que [A, B] \subseteq [A', B']$.

$P4 : \mathbf{I} \leq \mathbf{M}(\bigcup \mathbf{I}), para qualquer coleção de intervalos maximais \mathbf{I}$

$P5 : [A, B] \in \mathbf{M}(\mathcal{X}) \implies [A, B] \subseteq \mathcal{X}, \forall \mathcal{X}, [A, B]$.

$P6 : \mathcal{X}_1 \subseteq \mathcal{X}_2 \implies \mathbf{M}(\mathcal{X}_1) \leq \mathbf{M}(\mathcal{X}_2), \forall \mathcal{X}_1, \mathcal{X}_2$.

Conforme as notações utilizadas aqui, temos que $\mathcal{B}_W(\Psi) = \mathbf{M}(\mathcal{K}_W(\Psi))$, ou equivalentemente, que os implicantes primos de ψ são os intervalos em $\mathbf{M}(\psi(1))$. Portanto, a forma MSOP de ψ pode ser obtida calculando-se $\mathbf{M}(\psi(1))$ e em seguida uma cobertura mínima de $\psi(1)$ contida em $\mathbf{M}(\psi(1))$.

4.1.1 Algoritmos Clássicos

Dois intervalos são *adjacente* se eles diferem em apenas uma coordenada. Por exemplo, o 0-cubo 100 é adjacente aos 0-cubos 000, 110 e 101; o 1-cubo 1X0 é adjacente aos 1-cubos 0X0 e 1X1; e assim por diante. Dois cubos adjacentes podem ser combinados para formar um cubo maior : por exemplo, os 0-cubos 100 e 000 formam o 1-cubo X00, enquanto os 1-cubos 1X0 e 1X1 formam o 2-cubo 1XX.

Um algoritmo clássico para minimização de funções Booleanas é um algoritmo tabular conhecido por Quine-McCluskey (QM), consistindo de duas etapas, descritas a seguir.

1. cálculo de todos os implicants primos

Os implicants primos ou, equivalentemente, os intervalos maximais contidos em $\psi\langle 1 \rangle$, são calculados incrementalmente a partir dos 0-cubos correspondentes aos elementos de $\psi\langle 1 \rangle$. Quaisquer dois 0-cubos adjacentes são combinados para gerar um 1-cubo. Em seguida, quaisquer dois 1-cubos adjacentes são combinados para gerar 2-cubos, e assim por diante até que não existam mais cubos adjacentes. Os cubos que não puderam ser combinados com nenhum outro durante este processo são maximais e, portanto, implicants primos.

2. cálculo de uma cobertura mínima

Uma vez que os implicants primos são conhecidos, a segunda etapa deste algoritmo calcula uma cobertura mínima, isto é, seleciona um subconjunto dos implicants primos que são suficientes para cobrir todos os 0-cubos em $\psi\langle 1 \rangle$ e que satisfazem as condições de minimalidade.

A figura 4.2 ilustra o cálculo de implicants primos da função $\psi(x_1, x_2, x_3) = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3$ pelo algoritmo QM. Maiores detalhes sobre o algoritmo QM podem ser encontrados, por exemplo, em [77]. Algumas referências para outros algoritmos de minimização são [32, 120].

4.1.2 O Algoritmo ISI

Um outro algoritmo, denominado ISI (do inglês “Incremental Splitting of Intervals”), para minimização de funções Booleanas foi proposto recentemente em [157, 15]. Introduzimos aqui uma generalização desse algoritmo, cuja formulação engloba como um caso particular o algoritmo proposto anteriormente.

ISI é um algoritmo que segue um processo inverso ao QM. Isto é, em vez de agrupar pequenos cubos para gerar cubos maiores, ISI inicia o processo a partir do n -cubo, de onde extrai sucessivamente todos os elementos de $\psi\langle 0 \rangle$. A idéia básica do algoritmo consiste em representar o conjunto de elementos que resultam após uma operação de extração através de uma coleção de intervalos maximais. Assim, após a extração de todos os elementos de $\psi\langle 0 \rangle$, os elementos resultantes (i.e., elementos de $\psi\langle 1 \rangle$) estão representados através de intervalos maximais. Antes de descrever o algoritmo, apresentamos alguns resultados adicionais sobre representação de elementos de $\mathcal{P}(\mathcal{P}(W))$ em termos de intervalos maximais.

Seja Π_W o conjunto $\{\mathbf{M}(\mathcal{X}) : \mathcal{X} \subseteq \mathcal{P}(W)\}$. Os reticulados $(\mathcal{P}(\mathcal{P}(W)), \subseteq)$ e (Π_W, \leq) são isomorfos [16] pois $\mathcal{X} = \cup(\mathbf{M}(\mathcal{X}))$ para todo $\mathcal{X} \subseteq \mathcal{P}(W)$ e $\mathbf{M}(\cup \mathbf{I}) = \mathbf{I}$ para todo $\mathbf{I} \in \Pi_W$. Note que, denotando as operações de união, interseção e complementação, respectivamente, por \sqcup , \sqcap e $\bar{}$, valem

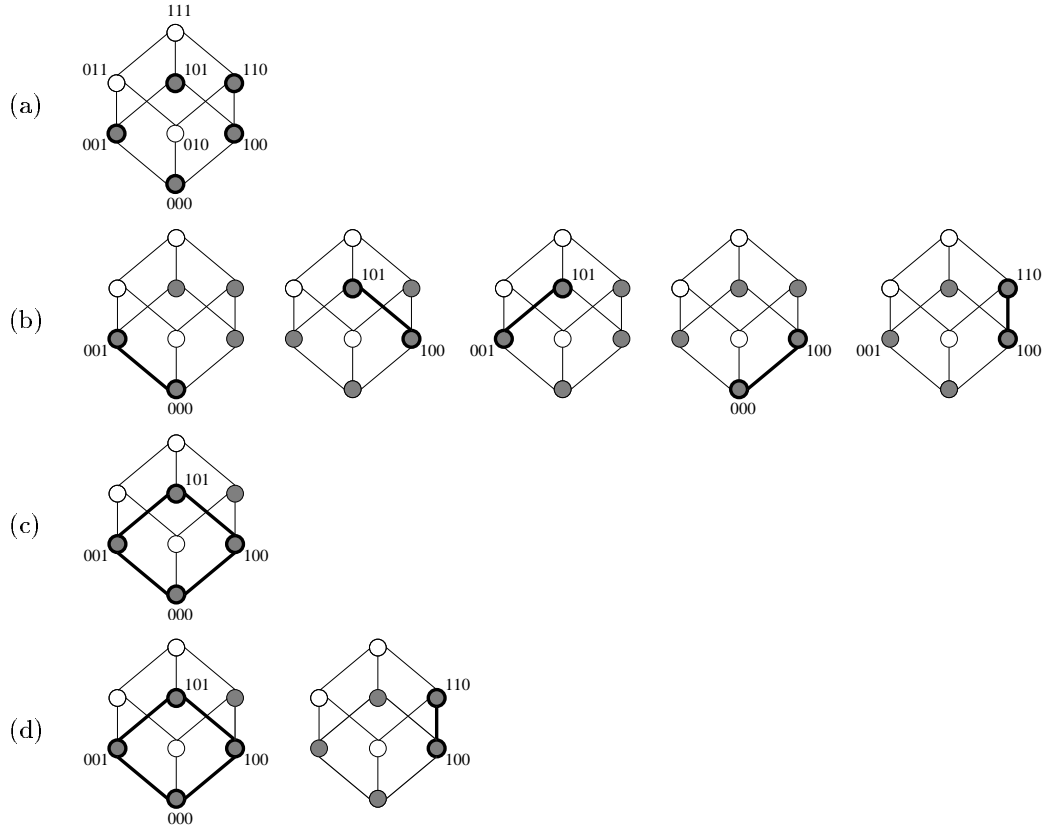


Figura 4.2: Minimização pelo algoritmo de Quine-McCluskey. (a) Os 0-cubos 000, 001, 100, 101, 110. (b) Os 1-cubos obtidos a partir dos 0-cubos em (a). (c) O 2-cubo obtido a partir dos 1-cubos em (b). (d) Os implicantes primos $X0X$ e $1X0$.

as relações :

$$\mathbf{X} \sqcup \mathbf{Y} = \mathbf{M}(\mathcal{X} \cup \mathcal{Y})$$

$$\mathbf{X} \sqcap \mathbf{Y} = \mathbf{M}(\mathcal{X} \cap \mathcal{Y})$$

$$\overline{\mathbf{X}} = \mathbf{M}(\mathcal{X}^c)$$

onde $\mathcal{X} = \cup \mathbf{X}$ e $\mathcal{Y} = \cup \mathbf{Y}$.

O complemento de um intervalo $[A, B] \subseteq \mathcal{P}(W)$ com relação a $\mathcal{P}(W)$ é definido por $[A, B]^c = \{X \in \mathcal{P}(W) : X \notin [A, B]\}$.

Proposição 4.4 *Seja $[A, B] \subseteq \mathcal{P}(W)$. Então,*

$$\mathbf{M}([A, B]^c) = \{[\emptyset, \{a\}^c] : a \in A\} \cup \{[\{b\}, W] : b \in B^c\}. \quad (4.1)$$

Dem.: *Veja demonstração em [16].* ■

Proposição 4.5 *Sejam $\mathbf{X}, \mathbf{Y} \in \Pi_w$. Então,*

$$\mathbf{X} \sqcap \mathbf{Y} = \text{Max}(\{[A \cup C, B \cap D] : [A, B] \in \mathbf{X}, [C, D] \in \mathbf{Y}\}).$$

Dem.: Veja demonstração em [16]. ■

A diferença entre dois intervalos $[A, B]$ e $[C, D]$, aqui denominado *extração de $[C, D]$ de $[A, B]$* , é definido como sendo o conjunto $[A, B] \setminus [C, D] = \{X \in \mathcal{P}(W) : X \in [A, B] \text{ e } X \notin [C, D]\}$, que também pode ser escrito como $[A, B] \setminus [C, D] = [A, B] \cap [C, D]^c$. A seguinte proposição mostra como a diferença entre dois intervalos pode ser expressa em termos de intervalos maximais.

Proposição 4.6 *Sejam $[A, B]$ e $[C, D]$ dois intervalos de $\mathcal{P}(W)$. Então,*

$$\mathbf{M}([A, B] \setminus [C, D]) = \{[A, B \cap \{c\}^c] : c \in C \cap A^c\} \cup \{[A \cup \{d\}, B] : d \in D^c \cap B\}. \quad (4.2)$$

Dem.:

$$\begin{aligned} \mathbf{M}([A, B] \setminus [C, D]) &\stackrel{(1)}{=} \mathbf{M}([C, D]^c \cap [A, B]) \\ &\stackrel{(2)}{=} \mathbf{M}([C, D]^c) \cap \mathbf{M}([A, B]) \\ &\stackrel{(3)}{=} \left(\{[\emptyset, \{c\}^c] : c \in C\} \cup \{[\{d\}, W] : d \in D^c\} \right) \cap \{[A, B]\} \\ &\stackrel{(4)}{=} \text{Max}(\{[A, \{c\}^c \cap B] : c \in C\} \cup \{[\{d\} \cup A, B] : d \in D^c\}) \\ &\stackrel{(5)}{=} \text{Max}(\{[A, B \cap \{c\}^c] : c \in C \cap A^c\} \cup \{[A \cup \{d\}, B] : d \in D^c \cap B\}) \\ &\stackrel{(6)}{=} \{[A, B \cap \{c\}^c] : c \in C \cap A^c\} \cup \{[A \cup \{d\}, B] : d \in D^c \cap B\} \end{aligned}$$

A igualdade (1) segue da definição, (2) do isomorfismo entre os reticulados $\mathcal{P}(\mathcal{P}(W))$ e Π_W , (3) da proposição 4.4, (4) da proposição 4.5, (5) pois $[A, B \cap \{c\}^c] \neq \emptyset, c \in C \iff A \subseteq B \cap \{c\}^c \iff A \subseteq \{c\}^c \iff c \notin A \iff c \in A^c$, e $[A \cup \{d\}, B] \neq \emptyset, d \in D^c \iff A \cup \{d\} \subseteq B \iff \{d\} \subseteq B \iff d \in B$ e (6) pois não existem dois intervalos em $\{[A, \{c\}^c \cap B] : c \in C\} \cup \{[A \cup \{d\}, B] : d \in D^c\}$ tais que um esteja propriamente contido em outro. ■

A *dimensão* de um intervalo $[A, B]$, $A \subseteq B$, é definido por $\dim([A, B]) = |B| - |A|$. Pode-se mostrar que um intervalo $[A, B]$ contém exatamente $2^{\dim([A, B])}$ elementos. Uma propriedade interessante da extração é a quantidade de intervalos gerados e a dimensão dos mesmos. Para mostrar esta propriedade, utilizamos o fato de que se U e V são dois conjuntos quaisquer, então $|U \cap V^c| = |U| - |U \cap V|$ e $|U \cap V| = |U| + |V| - |U \cup V|$. Utilizamos também a igualdade $[A, B] \cap [C, D] = [A \cup C, B \cap D]$ (ver demonstração em [16]).

Proposição 4.7 *Sejam $[A, B]$ e $[C, D]$ dois intervalos tais que $[A, B] \cap [C, D] \neq \emptyset$. Então*

$$a) \forall [A', B'] \in \mathbf{M}([A, B] \setminus [C, D]), \dim([A', B']) = \dim([A, B]) - 1,$$

$$b) |\mathbf{M}([A, B] \setminus [C, D])| = \dim([A, B]) - \dim([A, B] \cap [C, D]).$$

Dem.:

a) Se $[A', B'] \in \mathbf{M}([A, B] \setminus [C, D])$ então $[A', B'] \in \{[A, B \cap \{c\}^c] : c \in C \cap A^c\} \cup \{[A \cup \{d\}, B] : d \in D^c \cap B\}$.

Se $[A', B'] = [A, B \cap \{c\}^c]$, $c \in C \cap A^c$, então $\dim([A', B']) = |B'| - |A'| = |B \cap \{c\}^c| - |A| = |B| - |B \cap \{c\}| - |A| = |B| - |\{c\}| - |A| = |B| - 1 - |A| = \dim([A, B]) - 1$ (pois $C \subseteq B$ e, portanto, $c \in B$).

Da mesma forma, se $[A', B'] = [A \cup \{d\}, B]$, $d \in D^c \cap B$, temos $\dim([A', B']) = \dim([A, B]) - 1$.

b) Como $C \cap A^c$ e $D^c \cap B$ são disjuntos, a quantidade de intervalos em $\mathbf{M}([A, B] \setminus [C, D])$ é $|C \cap A^c| + |B \cap D^c|$. Isto é,

$$\begin{aligned}
 |\mathbf{M}([A, B] \setminus [C, D])| &= |C \cap A^c| + |B \cap D^c| \\
 &= |C| - |C \cap A| + |B| - |B \cap D| \\
 &= |C| - |A| - |C| + |A \cup C| + |B| - |B \cap D| \\
 &= |B| - |A| + |A \cup C| - |B \cap D| \\
 &= \dim([A, B]) - (|B \cap D| - |A \cup C|) \\
 &= \dim([A, B]) - \dim([A \cup C, B \cap D]) \\
 &= \dim([A, B]) - \dim([A, B] \cap [C, D])
 \end{aligned}$$

■

Um caso particular da proposição 4.6 é a extração de um elemento singular X do intervalo $[A, B]$ (originalmente proposto em [157]). De fato, uma vez que $\{X\} = [X, X]$, temos

$$\mathbf{M}([A, B] \setminus \{X\}) = \{[A, B \cap \{a\}^c] : a \in X \cap A^c\} \cup \{[A \cup \{b\}, B] : b \in B \cap X^c\}. \quad (4.3)$$

Todos os intervalos em $\mathbf{M}([A, B] \setminus \{X\})$ possuem dimensão $\dim([A, B]) - 1$ e a quantidade de intervalos em $\mathbf{M}([A, B] \setminus \{X\})$ é exatamente $\dim([A, B])$. A figura 4.3 mostra os 2-cubos gerados a partir da extração do elemento 011 do 3-cubo.

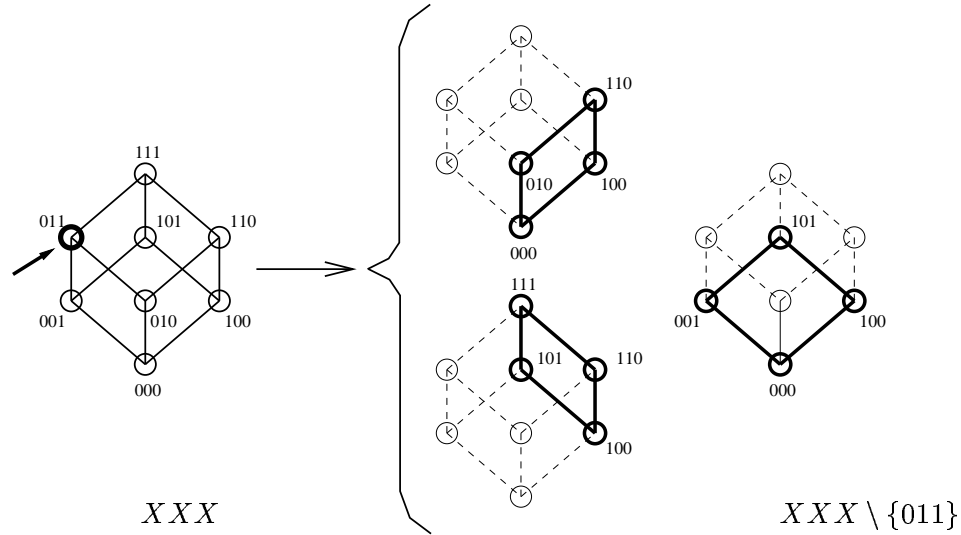


Figura 4.3: Extração do elemento 011 do 3-cubo resulta nos 2-cubos $XX0$, $1XX$, $X0X$.

O algoritmo (generalizado para extrair intervalos em vez de elementos) é descrito em Algoritmo 4.1. Todos os intervalos de $\mathbb{I}_{\psi\langle 0 \rangle}$, uma coleção de intervalos não necessariamente maximais tal que $\cup \mathbb{I}_{\psi\langle 0 \rangle} = \psi\langle 0 \rangle$, são extraídos sucessivamente a partir do n -cubo (se os elementos de $\psi\langle 0 \rangle$ não estiverem representados através de intervalos, podem ser facilmente convertidos para intervalos triviais). Dos intervalos novos gerados, aqueles que estão contidos em um intervalo maior existente

são eliminados, garantindo-se que todos os intervalos que são mantidos são maximais. A rotina $\text{min_cover}(\psi\langle 1 \rangle, \mathbb{I})$ calcula uma cobertura mínima de $\psi\langle 1 \rangle$ a partir dos intervalos em \mathbb{I} ¹.

Entrada: $W, \psi\langle 1 \rangle$ e $\mathbb{I}_{\psi\langle 0 \rangle}$ tal que $\cup \mathbb{I}_{\psi\langle 0 \rangle} = \psi\langle 0 \rangle$ e $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle = \mathcal{P}(W)$

Saída: Cobertura mínima de $\psi\langle 1 \rangle$ contida em $\mathbb{M}(\psi\langle 1 \rangle)$

```

1.  $\mathbb{I} \leftarrow \{\{\emptyset, W\}\}$  ;
2. Para cada  $[C, D] \in \mathbb{I}_{\psi\langle 0 \rangle}$  faça {
     $\mathbb{P} \leftarrow \{[A, B] \in \mathbb{I} : [A, B] \cap [C, D] = \emptyset\}$  ;
     $\text{tmp} \leftarrow \{[A, B] \in \mathbb{I} : [A, B] \cap [C, D] \neq \emptyset\}$  ;
     $\text{New} \leftarrow \emptyset$  ;
    Para cada  $[A, B] \in \text{tmp}$  faça {
         $\text{split} \leftarrow \mathbb{M}([A, B] \setminus [C, D])$  ;
        Para cada  $[A', B'] \in \text{split}$  {
            se não existe  $[E, F] \in \mathbb{P}$  tal que  $[A', B'] \subseteq [E, F]$ 
                então  $\text{New} \leftarrow \text{New} \cup \{[A', B']\}$  ;
        }
    }
     $\mathbb{I} \leftarrow \mathbb{P} \cup \text{New}$  ;
}
3.  $\mathbb{I} = \text{min\_cover}(\psi\langle 1 \rangle, \mathbb{I})$ .
```

Algoritmo 4.1 Algoritmo ISI.

Exemplo 4.8 Seja $n = 3$ e $\psi\langle 0 \rangle = \{111, 011, 010\}$. Os intervalos gerados pela extração dos elementos de $\psi\langle 0 \rangle$ são mostrados através de diagramas de Hasse na figura 4.4(a). Os mesmos intervalos são mostrados através de uma estrutura de árvore na figura 4.4(b). Na raiz da árvore encontra-se o 3-cubo. Na profundidade i da árvore estão todos os intervalos que foram gerados na iteração i do passo (2) do algoritmo. Dessas, aqueles que estão riscados foram eliminados por estarem contidos em intervalos maiores.

Para provar a corretude do algoritmo, apresentamos alguns resultados adicionais.

Proposição 4.9 Se \mathbb{I} é uma coleção de intervalos maximais então qualquer subcoleção $\mathbb{I}' \subset \mathbb{I}$ também é uma coleção de intervalos maximais.

Dem.: Suponha por absurdo que \mathbb{I}' não é uma coleção de intervalos maximais. Então existem ao menos dois intervalos $[A, B], [A', B'] \in \mathbb{I}'$ tal que $[A, B] \subset [A', B']$. Como $\mathbb{I}' \subset \mathbb{I}$, então $[A, B], [A', B'] \in \mathbb{I}$ e, portanto, \mathbb{I} não é uma coleção de intervalos maximais. Isto é um absurdo. ■

Proposição 4.10 Seja $[C, D] \subseteq \mathcal{P}(W)$ e $\mathbb{I} = \{[A_i, B_i] \subseteq \mathcal{P}(W) : [A_i, B_i] \cap [C, D] \neq \emptyset, i \in I\}$ uma coleção de intervalos maximais, onde I é uma coleção de índices. Então $\bigcup_{i \in I} \mathbb{M}([A_i, B_i] \setminus [C, D])$ é uma coleção de intervalos maximais.

¹O cálculo da cobertura mínima não será discutido neste texto. uma implementação do mesmo pode ser encontrado em [77]

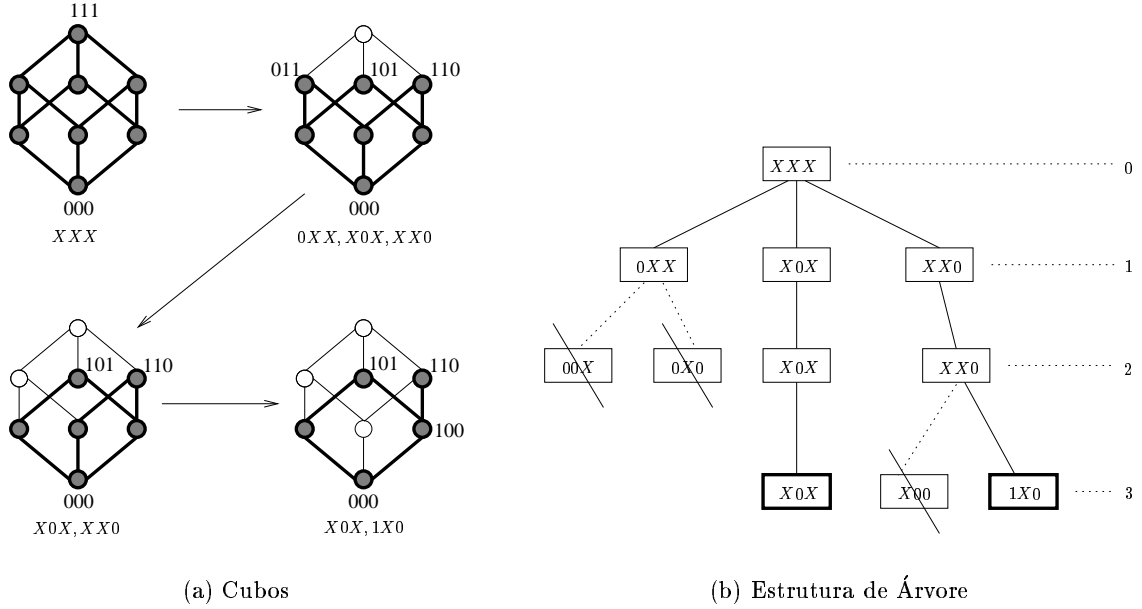


Figura 4.4: Cálculo de implicantes primos pelo algoritmo ISI. $\psi\langle 0 \rangle = \{010, 011, 111\}$ e $\psi\langle 1 \rangle = \{000, 001, 100, 101, 110\}$.

Dem.: Reescrevendo $\bigcup_{i \in I} \mathbf{M}([A_i, B_i] \setminus [C, D])$ em termos da proposição 4.6, temos

$$\bigcup_{i \in I} \mathbf{M}([A_i, B_i] \setminus [C, D]) = \bigcup_{i \in I} \{[A_i, B_i \cap \{c\}^c] : c \in C \cap A_i^c\} \cup \{[A_i \cup \{d\}, B_i] : d \in D^c \cap B_i\}.$$

Pela mesma proposição sabemos que os intervalos de $\mathbf{M}([A_i, B_i] \setminus [C, D])$ são maximais. Precisamos mostrar que nenhum intervalo de $\mathbf{M}([A_i, B_i] \setminus [C, D])$ está contido em um intervalo de $\mathbf{M}([A_j, B_j] \setminus [C, D])$, para $i \neq j$, isto é, precisamos mostrar que :

- não existe $c_1 \in C \cap A_i^c$ e $c_2 \in C \cap A_j^c$ tal que $[A_i, B_i \cap \{c_1\}^c] \subseteq [A_j, B_j \cap \{c_2\}^c]$,
- não existe $c \in C \cap A_i^c$ e $d \in D^c \cap B_j$ tal que $[A_i, B_i \cap \{c\}^c] \subseteq [A_j \cup \{d\}, B_j]$,
- não existe $d \in D^c \cap B_i$ e $c \in C \cap A_j^c$ tal que $[A_i \cup \{d\}, B_i] \subseteq [A_j, B_j \cap \{c\}^c]$ e
- não existe $d_1 \in D^c \cap B_i$ e $d_2 \in D^c \cap B_j$ tal que $[A_i \cup \{d_1\}, B_i] \subseteq [A_j \cup \{d_2\}, B_j]$.

Em primeiro lugar, mostramos que $A_i \subseteq D$ e $C \subseteq B_i$, $\forall i \in I$. De fato, $\forall i \in I$,

$$[A_i, B_i] \cap [C, D] \neq \emptyset \implies [A_i \cup C, B_i \cap D] \neq \emptyset \implies A_i \cup C \subseteq B_i \cap D \implies A_i \subseteq D \text{ e } C \subseteq B_i.$$

No caso (a), suponha por absurdo que existe $c_1 \in C \cap A_i^c$ e $c_2 \in C \cap A_j^c$ tal que $[A_i, B_i \cap \{c_1\}^c] \subseteq [A_j, B_j \cap \{c_2\}^c]$. Então, devemos ter que $A_j \subseteq A_i$ e $B_i \cap \{c_1\}^c \subseteq B_j \cap \{c_2\}^c$. Logo, todos os elementos de B_i que são diferentes de c_1 estão em B_j . Além disso, como $c_1 \in C \subseteq B_j$, temos que $B_i \subseteq B_j$, o que implica que $[A_i, B_i] \subseteq [A_j, B_j]$ e, portanto, que $[A_i, B_i]$ não é um intervalo maximal em \mathbf{I} , um absurdo.

No caso (b), suponha por absurdo que existe $c \in C \cap A_i^c$ e $d \in D^c \cap B_j$ tal que $[A_i, B_i \cap \{c\}^c] \subseteq [A_j \cup \{d\}, B_j]$. Então devemos ter $A_j \cup \{d\} \subseteq A_i$. Isto implica que $d \in A_i$, o que é um absurdo pois $A_i \subseteq D$ e $d \in D^c$.

No caso (c), suponha por absurdo que existe $d \in D^c \cap B_i$ e $c \in C \cap A_j^c$ tal que $[A_i \cup \{d\}, B_i] \subseteq [A_j, B_j \cap \{c\}^c]$. Então devemos ter $B_i \subseteq B_j \cap \{c\}^c$. Isto implica que $c \notin B_i$, o que é um absurdo pois $c \in C \subseteq B_i$.

No caso (d), suponha por absurdo que existe $d_1 \in D^c \cap B_i$ e $d_2 \in D^c \cap B_j$ tal que $[A_i \cup \{d_1\}, B_i] \subseteq [A_j \cup \{d_2\}, B_j]$. Então devemos ter $A_j \cup \{d_1\} \subseteq A_i \cup \{d_2\}$. Isto implica que $d_1 \in A_i$, o que é um absurdo pois $A_i \subseteq D$ e $d_1 \notin D$. ■

Teorema 4.11 *Sejam $\psi\langle 1 \rangle$ e $\psi\langle 0 \rangle$ dois subconjuntos de $\mathcal{P}(W)$ tais que $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle = \mathcal{P}(W)$. Seja também $\mathbf{I}_{\psi\langle 0 \rangle}$ tal que $\cup \mathbf{I}_{\psi\langle 0 \rangle} = \psi\langle 0 \rangle$. Se \mathbf{I} é a coleção de intervalos resultante após a aplicação do passo (2) do algoritmo ISI (algoritmo 4.1), então $\mathbf{I} = \mathbf{M}(\psi\langle 1 \rangle)$.*

Dem.: *Sejam $t = |\mathbf{I}_{\psi\langle 0 \rangle}|$ e $\mathbf{I}_{\psi\langle 0 \rangle} = \{[C_1, D_1], [C_2, D_2], \dots, [C_t, D_t]\}$. Para todo $i = 1, \dots, t$, sejam $\mathbf{X}_i = \{[C_1, D_1], [C_2, D_2], \dots, [C_i, D_i]\}$, \mathbf{I}_i a coleção de intervalos após a iteração i do passo (2) e $\mathcal{X}_i = \psi\langle 0 \rangle \setminus (\cup \mathbf{X}_i)$.*

Passo 0 : *Em particular, $\mathbf{X}_0 = \emptyset$, $\mathcal{X}_0 = \psi\langle 0 \rangle \setminus (\cup \mathbf{X}_0) = \psi\langle 0 \rangle$ e $\mathbf{I}_0 = \{[\emptyset, W]\} = \mathbf{M}(\mathcal{P}(W)) = \mathbf{M}(\psi\langle 1 \rangle \cup \psi\langle 0 \rangle) = \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_0)$.*

Passo da indução: *Suponha que $\mathbf{I}_{i-1} = \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_{i-1})$ e que $[C_i, D_i]$ é o intervalo a ser extraído na iteração i . Vamos mostrar que $\mathbf{I}_i = \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i)$.*

Para isso basta mostrarmos i), ii) e iii) .

i) \mathbf{I}_i é uma coleção de intervalos maximais.

Ao final da iteração i , $\mathbf{I}_i \leftarrow \mathbf{P} \cup \text{New}$. A coleção \mathbf{P} contém todos os intervalos de \mathbf{I}_{i-1} que não interceptam o intervalo $[C_i, D_i]$. Logo, \mathbf{P} é uma coleção de intervalos maximais (proposição 4.9). A coleção New é uma coleção de intervalos maximais (proposição 4.10) e, mais ainda, nenhum intervalo de New está coberto por algum intervalo de \mathbf{P} , pois tais intervalos foram descartados durante a construção de New . Além disso, nenhum intervalo de New cobre intervalos de \mathbf{P} , pois isto implicaria que os intervalos em \mathbf{I}_{i-1} não eram maximais. Logo, \mathbf{I}_i é uma coleção de intervalos maximais.

ii) $\mathbf{I}_i \leq \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i)$.

Sabemos que $\mathbf{I}_i \leq \mathbf{M}(\cup \mathbf{I}_i)$. Vamos mostrar que $\mathbf{M}(\cup \mathbf{I}_i) \leq \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i)$. Para isto basta mostrarmos que $\cup \mathbf{I}_i \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_i$.

Seja $X \in \cup \mathbf{I}_i$. Então existe $[A, B] \in \mathbf{I}_i$ tal que $X \in [A, B]$. Por construção de \mathbf{I}_i , para todo $[A, B] \in \mathbf{I}_i$ (a) $[A, B] \in \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_{i-1})$ ou (b) $[A, B] \in \mathbf{M}([A', B'] \setminus [C, D])$, $[A', B'] \in \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_{i-1})$.

ii.a) Se $[A, B] \in \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_{i-1})$ então $[A, B] \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_{i-1}$ e mais ainda, $[A, B] \cap [C_i, D_i] = \emptyset$. Logo, $[A, B] \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_i$.

ii.b) Se $[A, B] \in \mathbf{M}([A', B'] \setminus [C_i, D_i])$, então $[A, B] \subseteq [A', B'] \setminus [C_i, D_i] \subseteq [A', B']$. Como $[A', B'] \in \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_{i-1})$, então $[A, B] \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_{i-1}$. Logo $[A, B] \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_i$ pois $[A, B] \cap [C, D] = \emptyset$.

De ii.a) e ii.b) segue que $[A, B] \in \mathbf{I}_i \implies [A, B] \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_i$. Logo, $\cup \mathbf{I}_i \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_i$.

iii) $\mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i) \leq \mathbf{I}_i$.

Seja $[A, B] \in \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i)$. Como $\psi\langle 1 \rangle \cup \mathcal{X}_i \subseteq \psi\langle 1 \rangle \cup \mathcal{X}_{i-1}$, então $\mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i) \leq \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_{i-1})$. Logo, existe $[A', B'] \in \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_{i-1})$ tal que $[A, B] \subseteq [A', B']$. Se $[A', B'] \cap [C, D] = \emptyset$, então $[A', B'] \in \mathbf{I}_i$ por construção; senão se $[A', B'] \cap [C, D] \neq \emptyset$, então $[A, B] \subseteq [A', B'] \setminus [C_i, D_i]$ pois $[A, B] \cap [C_i, D_i] = \emptyset$. Logo, $\mathbf{M}([A, B]) \leq \mathbf{M}([A', B'] \setminus [C, D])$ e, portanto, existe $[A'', B''] \in \mathbf{M}([A', B'] \setminus [C, D])$.

$[C_i, D_i]$ tal que $[A, B] \subseteq [A'', B'']$. Por construção, ou $[A'', B''] \in \mathbf{I}_i$ ou existe $[E, F] \in \mathbf{I}_i$ tal que $[A'', B''] \subseteq [E, F]$.

Portanto, para todo $[A, B] \in \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i)$ existe $[A', B'] \in \mathbf{I}_i$ tal que $[A, B] \subseteq [A', B']$. Isto é, $\mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_i) \leq \mathbf{I}_i$.

Como $\mathbf{X}_0 = \emptyset$, $\mathcal{X}_0 = \psi\langle 0 \rangle \setminus (\cup \mathbf{X}_0) = \psi\langle 0 \rangle$ e $\mathbf{I}_0 = \{[\emptyset, W]\} = \mathbf{M}(\mathcal{P}(W)) = \mathbf{M}(\psi\langle 1 \rangle \cup \psi\langle 0 \rangle) = \mathbf{M}(\psi\langle 1 \rangle \cup \mathcal{X}_0)$, por indução em i temos que $\mathbf{I} = \mathbf{I}_t = \mathbf{M}(\psi\langle 1 \rangle)$. ■

No terceiro passo do algoritmo uma cobertura mínima de $\psi\langle 1 \rangle$ é calculada a partir dos intervalos obtidos no passo (2) do algoritmo, i.e., a partir de $\mathbf{I} = \mathbf{M}(\psi\langle 1 \rangle)$. Logo, o resultado do algoritmo ISI é uma cobertura mínima de $\psi\langle 1 \rangle$, cujos intervalos estão contidos em $\psi\langle 1 \rangle$.

4.2 Minimização na Existência de “Don’t Cares”

Em muitos casos, funções Booleanas não são completamente especificadas, i.e., existem elementos para os quais o valor da função não é conhecido ou não importa. Estes elementos são denominados *don’t cares* e são denotados por $\psi\langle * \rangle$. Logo, $\mathcal{P}(W) = \psi\langle 1 \rangle \cup \psi\langle 0 \rangle \cup \psi\langle * \rangle$.

A observação acima é especialmente verdadeira para problemas de aprendizado de funções Booleanas. Para n relativamente grande, em geral, a maior parte dos elementos de $\mathcal{P}(W)$ constituem *don’t cares*, isto é, não são observados nas imagens de treinamento. Logo, o número de elementos em $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle$ é, muitas vezes, relativamente pequeno se comparado ao total, 2^n . Isto significa que os 0-cubos podem estar esparsamente distribuídos no n -cubo e, portanto, o algoritmo QM pode não ser capaz de agrupar os sub-cubos, o que implica que uma redução significativa da representação não será alcançada. Para obter uma boa redução na representação através do algoritmo QM, podemos considerar na primeira etapa do algoritmo todos os 0-cubos em $\psi\langle 1 \rangle \cup \psi\langle * \rangle$ em vez dos 0-cubos em $\psi\langle 1 \rangle$ somente, pois isto aumenta as possibilidades de agrupar os cubos. No entanto, o tempo necessário para o cálculo de implicantes primos pode aumentar significativamente, devido à quantidade de dados adicionais a serem considerados. A tabela 4.1 (extraída de [157]) mostra algumas comparações entre os algoritmos QM e ISI. Na existência de don’t cares, o desempenho do algoritmo ISI é claramente superior ao QM.

n	$ \psi\langle 0 \rangle $	$ \psi\langle 1 \rangle $	$ \psi\langle * \rangle $	Intervalos	ISI		QM	
					Tempo	Mem.	Tempo	Mem.
11	655	1393	0	275	00:00:59	1991	00:00:51	12071
11	425	688	935	135	00:00:50	2082	00:01:12	21666
12	1035	3061	0	430	00:07:00	4580	00:07:07	38524
12	712	724	2660	90	00:00:24	2202	00:20:10	89824
15	16384	16384	0	1768	06:46:15	12728	07:19:54	213101
15	1706	1031	30031	110	00:26:18	21389	> 4,5 dias	-

Tabela 4.1: Comparação entre os algoritmos QM e ISI.

Nesta seção investigamos como os “don’t cares” são tratados pelo algoritmo ISI. Se $\mathbf{I}_{\psi\langle 0 \rangle} = \{[C_1, D_1], [C_2, D_2], \dots, [C_t, D_t]\}$, então os intervalos gerados pelo algoritmo ISI durante a etapa de

extração correspondem a uma sequência de funções Booleanas $\psi_0, \psi_1, \dots, \psi_t$, que são caracterizadas por :

$$\begin{array}{ll}
 \psi_0\langle 1 \rangle = \mathcal{P}(W) & \psi_0\langle 0 \rangle = \emptyset \\
 \psi_1\langle 1 \rangle = \mathcal{P}(W) \setminus [C_1, D_1] & \psi_1\langle 0 \rangle = [C_1, D_1] \\
 \psi_2\langle 1 \rangle = \mathcal{P}(W) \setminus (\cup\{[C_1, D_1], [C_2, D_2]\}) & \psi_2\langle 0 \rangle = \cup\{[C_1, D_1], [C_2, D_2]\} \\
 \vdots & \vdots \\
 \psi_t\langle 1 \rangle = \mathcal{P}(W) \setminus (\cup\{[C_1, D_1], [C_2, D_2] \dots, [C_t, D_t]\}) & \psi_t\langle 0 \rangle = \cup\{[C_1, D_1], [C_2, D_2] \dots, [C_t, D_t]\}
 \end{array}$$

Ao final da etapa de extração, i.e., após todos os intervalos de $\mathbf{I}_{\psi\langle 0 \rangle}$ terem sido extraídos, os intervalos resultantes cobrem todos os elementos de $\psi\langle 1 \rangle \cup \psi\langle * \rangle$, isto é, $\mathbf{I} = \mathbf{M}(\psi\langle 1 \rangle \cup \psi\langle * \rangle)$. Além disso, $\psi_0 > \psi_1 > \psi_2 > \dots > \psi_t$. A função resultante ψ_t é consistente com $\psi\langle 1 \rangle$ e $\psi\langle 0 \rangle$, i.e., $\psi_t(X) = \psi(X)$, $\forall X \in \psi\langle 1 \rangle \cup \psi\langle 0 \rangle$. Qualquer chaveamento de ψ_t sobre elementos em $\psi\langle * \rangle$ também resulta em funções consistentes com $\psi\langle 1 \rangle$ e $\psi\langle 0 \rangle$. Portanto, explorar estes chaveamentos possibilita algumas variantes do algoritmo ISI.

4.2.1 Variantes do Algoritmo ISI

O passo (3) do algoritmo ISI seleciona uma cobertura mínima de $\psi\langle 1 \rangle$ a partir dos implicantes primos calculados na etapa de extração. Se um intervalo gerado na etapa de extração não contém nenhum elemento de $\psi\langle 1 \rangle$, então nenhum de seus subintervalos os contém. Conseqüentemente, nenhum dos intervalos gerados a partir deste intervalo será necessário para cobrir algum elemento de $\psi\langle 1 \rangle$. Isto significa que, se existirem tais intervalos, eles certamente serão descartados na etapa de cálculo da cobertura mínima. Portanto, o algoritmo pode ser otimizado simplesmente eliminando-se, após cada iteração do passo (2), todos os intervalos que não cobrem nenhum elemento de $\psi\langle 1 \rangle$. Este procedimento não afeta o resultado final.

O mesmo raciocínio não vale para o algoritmo QM, uma vez que ao eliminarmos um intervalo no meio do processo de agrupamento podemos impedir que um intervalo de dimensão maior (que faria parte do MSOP) seja formado.

Variante 1: Calcular uma cobertura mínima de $\psi\langle 1 \rangle$ periodicamente

Quanto maior o número de intervalos em $\mathbf{I}_{\psi\langle 0 \rangle}$, maior tende a ser o número de intervalos gerados durante o passo (2) do algoritmo. Para n grande, o número de intervalos gerados pode tornar-se computacionalmente proibitivo. Uma heurística para manter o número de intervalos relativamente pequeno em qualquer iteração do passo (2) consiste em calcularmos uma cobertura mínima após cada número fixo de iterações. Se este número, que denominamos *período*, é 1, então o número de intervalos ao final de qualquer iteração do passo (2) é relativamente pequeno (não pode passar de $|\psi\langle 1 \rangle|$). Uma possível conseqüência é que os intervalos resultantes podem não ser uma cobertura mínima. Se o período considerado for $k = t = |\mathbf{I}_{\psi\langle 0 \rangle}|$, então recaímos no algoritmo original (i.e., a cobertura mínima só é calculada após a etapa de extração).

Variante 2: Interromper o processo de extração quando todos os elementos de $\psi\langle 1 \rangle$ estiverem cobertos

Em vez de gerar todos os subintervalos quando um exemplo está sendo extraído, pode-se gerar os subintervalos até que todos os elementos de $\psi\langle 1 \rangle$ estejam cobertos. Demais intervalos, mesmo aqueles que estão na fila para serem divididos, podem ser descartados. Este procedimento garante que todos os elementos de $\psi\langle 1 \rangle$ continuem cobertos pelo conjunto de intervalos mantidos. Novamente, deve-se notar que os intervalos resultantes podem não corresponder à expressão MSOP. Na prática observou-se que esta variante tende a gerar muito mais intervalos do que a variante anterior e, portanto, tende a ser relativamente lento.

4.2.2 ISI modificado para manipular “don’t cares”

Apresentamos a seguir uma modificação do algoritmo ISI para manipular os don’t cares adequadamente (algoritmo 4.2). Levando em conta que a primeira variante engloba o algoritmo original como um caso particular, e que a segunda variante não é eficiente, a modificação implementa a primeira variante e introduz o parâmetro de período, k . De agora em diante, qualquer menção ao algoritmo ISI refere-se ao algoritmo modificado.

Entrada: $W, \psi\langle 1 \rangle, \mathbf{I}_{\psi\langle 0 \rangle}$ tal que $\bigcup \mathbf{I}_{\psi\langle 0 \rangle} = \psi\langle 0 \rangle$, k (período para cálculo de cobertura mínima)
Saída: Cobertura minimal de $\psi\langle 1 \rangle$

```

1.  $\mathbf{I} \leftarrow \{\emptyset, W\}$  ;
2. Para cada  $[C, D] \in \mathbf{I}_{\psi\langle 0 \rangle}$  faça {
     $\mathbf{P} \leftarrow \{[A, B] \in \mathbf{I} : [A, B] \cap [C, D] = \emptyset\}$  ;
     $\mathbf{tmp} \leftarrow \{[A, B] \in \mathbf{I} : [A, B] \cap [C, D] \neq \emptyset\}$  ;
     $\mathbf{New} \leftarrow \emptyset$ ;
    Para cada  $[A, B] \in \mathbf{tmp}$  faça {
         $\mathbf{split} \leftarrow \mathbf{M}([A, B] \setminus [C, D])$  ;
        Para cada  $[A', B'] \in \mathbf{split}$  {
            se não existe  $[E, F] \in \mathbf{P}$  tal que  $[A', B'] \subseteq [E, F]$ 
            e se  $[A', B']$  cobre algum elemento de  $\psi\langle 1 \rangle$ 
            então  $\mathbf{New} \leftarrow \mathbf{New} \cup \{[A', B']\}$  ;
        }
    }
     $\mathbf{I} \leftarrow \mathbf{P} \cup \mathbf{New}$  ;
    Se o número de exemplos processados é múltiplo de  $k$ 
    ou se esta é a última iteração
    então  $\mathbf{I} \leftarrow \mathbf{min\_cover}(\psi\langle 1 \rangle, \mathbf{I})$  ;
}
3. Retorne  $\mathbf{I}$ .
```

Algoritmo 4.2 Algoritmo ISI (modificado).

Neste algoritmo modificado, $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle$ não precisa ser o conjunto $\mathcal{P}(W)$. A coleção de intervalos retornada é uma coleção de intervalos maximais que cobre $\psi\langle 1 \rangle$, mas não cobre nenhum elemento de $\psi\langle 0 \rangle$. Se $k < |\mathbf{I}_{\psi\langle 0 \rangle}|$ então o resultado é uma cobertura minimal mas não necessariamente mínima;

caso contrário, o resultado é uma cobertura mínima.

O desempenho do algoritmo foi avaliado para várias entradas (variando-se o número de variáveis n e o tamanho do conjunto $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle$). Os elementos de $\psi\langle 0 \rangle$ não foram agrupados em intervalos, mas poderiam ser agrupados usando um processo análogo ao do algoritmo QM. As figuras 4.5 a 4.8 mostram o desempenho do algoritmo ISI, para 4 entradas distintas. Para cada uma das entradas, diferentes períodos foram utilizados. Cada figura mostra gráficos que correspondem, respectivamente, ao tempo de processamento e ao tamanho da base. O período está indicado no eixo das abscissas de cada gráfico.

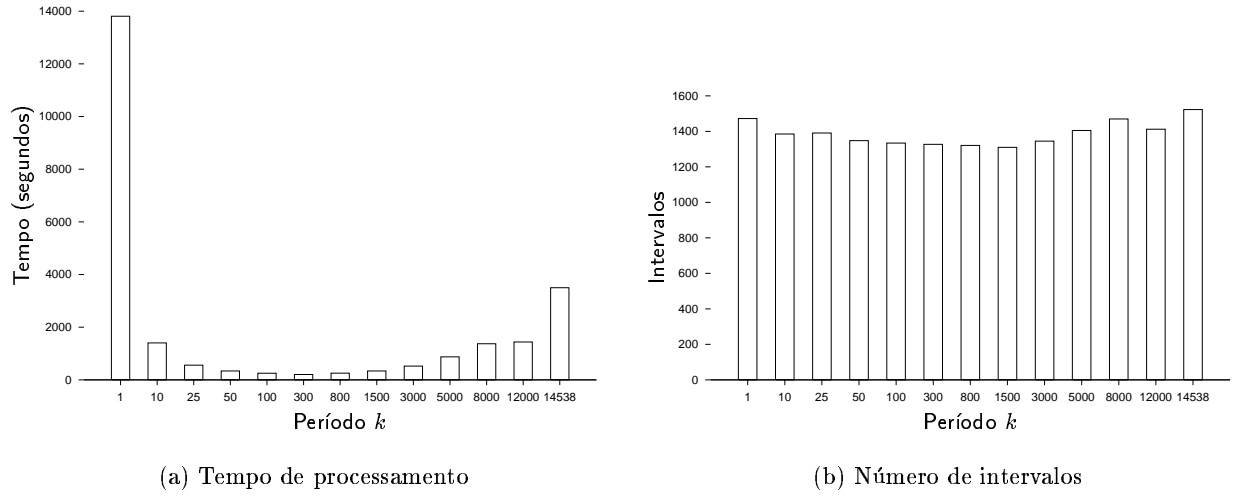


Figura 4.5: ISI com diferentes períodos ($n = 15$, $|\psi\langle 1 \rangle| = 9110$ e $|\psi\langle 0 \rangle| = 14538$).

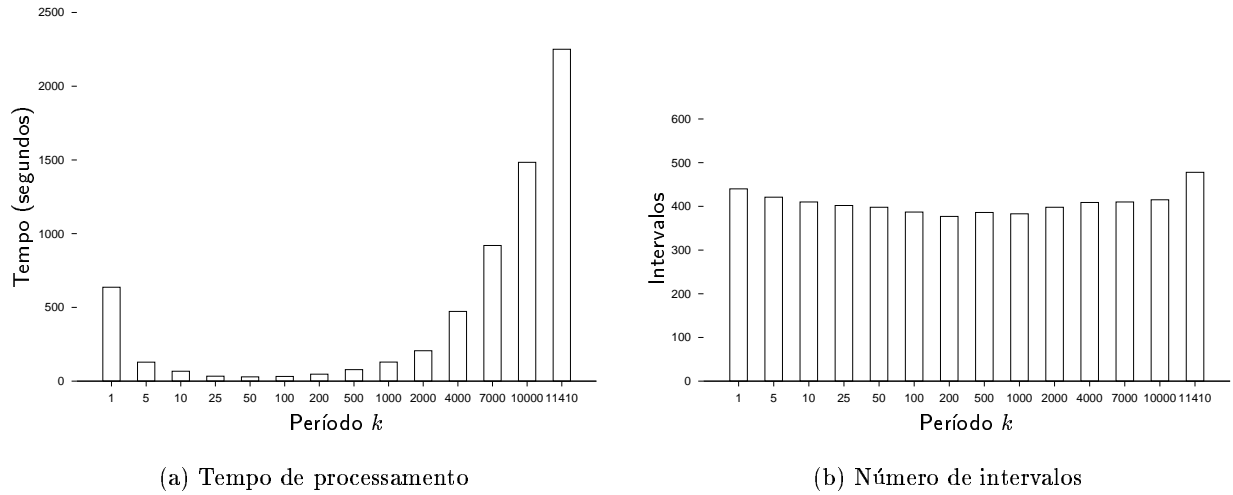


Figura 4.6: ISI com diferentes períodos ($n = 17$, $|\psi\langle 1 \rangle| = 1609$ e $|\psi\langle 0 \rangle| = 11410$).

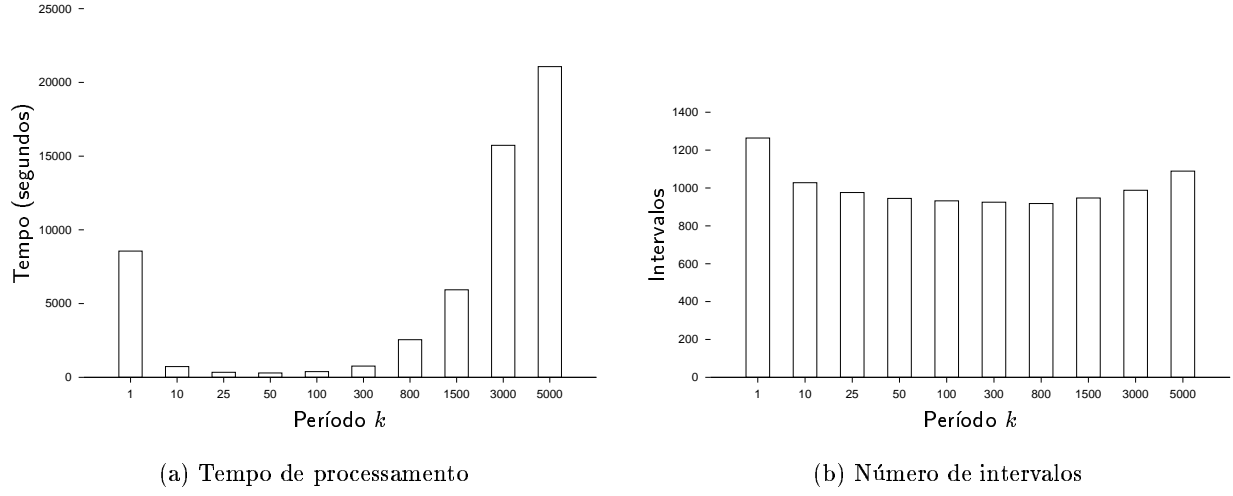


Figura 4.7: ISI com diferentes períodos ($n = 17$, $|\psi\langle 1 \rangle| = 8517$ e $|\psi\langle 0 \rangle| = 12251$).

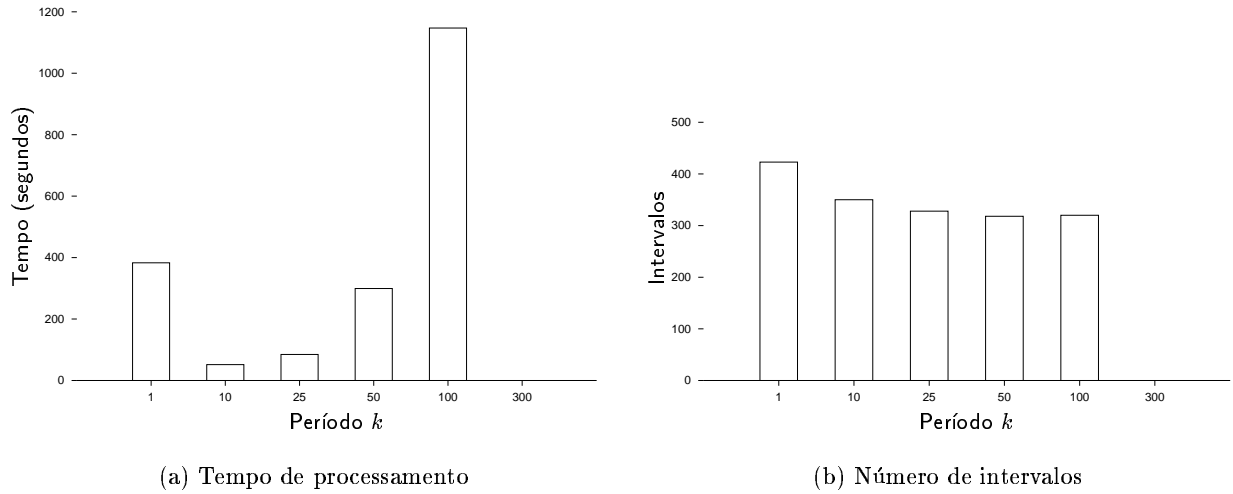


Figura 4.8: ISI com diferentes períodos ($n = 25$, $|\psi\langle 1 \rangle| = 1340$ e $|\psi\langle 0 \rangle| = 8126$).

Podemos observar que diferentes períodos afetam o tempo de processamento drasticamente, mas não afetam muito a quantidade de intervalos resultantes. O tempo de processamento é definido basicamente em função do tempo de extração de intervalos e o tempo de cálculo da cobertura mínima. Desta forma, para o período $k = 1$, o tempo de processamento pode ser grande pois a cobertura mínima é calculada a cada iteração do passo (2). Em geral, o tempo para cálculo de cobertura mínima depende de $|\psi\langle 1 \rangle|$ e de $|\mathbb{I}|$. Por outro lado, para períodos grandes, a quantidade de intervalos gerados pelas extrações entre duas etapas de cálculo de cobertura mínima pode ser muito grande, o que implica em maior tempo de processamento de extração. Períodos intermediários possuem tempo de processamento relativamente menor pois calculam a cobertura mínima poucas vezes e ao mesmo tempo não deixam a quantidade de intervalos gerados crescer demasiadamente.

O leitor atento deve ter notado que, nos gráficos, o número de intervalos para períodos $k \geq |\mathbb{I}_{\psi\langle 0 \rangle}|$ é maior que para alguns períodos menores, contrariando a observação feita no terceiro parágrafo anterior a esta. Cabe registrarmos que isto deve-se ao fato de que a implementação do algoritmo utilizada para cálculo de cobertura mínima usa algumas heurísticas e portanto o resultado pode ser subótimo.

Nas situações em que os elementos de $\psi\langle 0 \rangle$ se encontram agrupados, espera-se que o tempo de processamento seja menor em relação ao processamento sem agrupamento pois o número de iterações é menor no primeiro caso. Na tabela 4.2 mostramos o tempo de processamento comparando os dois casos para algumas situações. Os intervalos foram agrupados usando um processo similar ao do algoritmo QM. Pode-se observar que quando $|\mathbb{I}_{\psi\langle 0 \rangle}|$ é significativamente menor que $|\psi\langle 0 \rangle|$ o tempo de processamento para o primeiro caso é também significativamente menor. No entanto, se levarmos em conta o tempo necessário para agrupamento, em geral, não há vantagem.

n	$ \psi\langle 1 \rangle $	$ \psi\langle 0 \rangle $	Sem agrupamento Tempo (seg.)	Com agrupamento			
				$ \mathbb{I}_{\psi\langle 0 \rangle} $	agrup. (seg.)	ISI (seg.)	Total (agrup.+ISI)
17	6218	10279	2257	7459	562	1481	2043
17	1609	11410	637	11022	508	826	1334
21	21371	40589	31150	28611	21698	16469	38167
25	4202	37504	9469	33357	880	9290	10170

Tabela 4.2: ISI com e sem agrupamento de elementos.

4.3 Algoritmo ISI como um Algoritmo de Aprendizado

Para que um algoritmo seja interessante como algoritmo de aprendizado, uma das principais características de interesse é a sua capacidade de generalização, isto é, a capacidade de classificar corretamente os padrões não observados durante o treinamento.

Na etapa de cálculo da cobertura mínima são selecionados os intervalos suficientes para cobrir todos os elementos de $\psi\langle 1 \rangle$. Portanto, os don't cares que são cobertos por estes intervalos selecionados são também mapeados para 1; aqueles que não são cobertos são mapeados para 0. Tanto a variante original do ISI como o QM (quando todos os don't cares são considerados como 1) fazem a mesma generalização, uma vez que os intervalos calculados na primeira etapa são os mesmos. Além disso, pela observação já feita, o ISI com período $k \geq |\mathbb{I}_{\psi\langle 0 \rangle}|$ também faz a mesma generalização. Períodos menores que este podem produzir generalizações diferentes, uma vez que os intervalos calculados na

primeira etapa não são necessariamente a coleção de todos os implicantes primos.

As figuras 4.9 a 4.12 mostram gráficos com o erro MAE (estimado sobre um conjunto de imagens de validação) para os 4 casos acima.

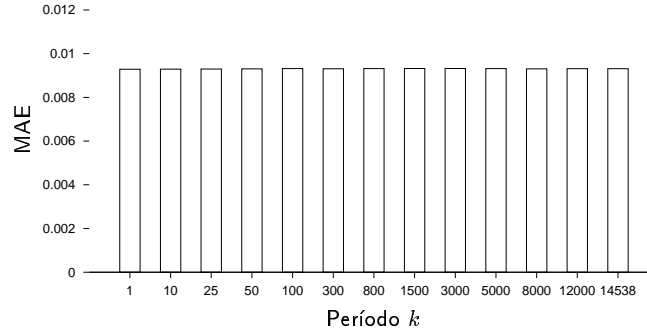


Figura 4.9: Erro MAE para ISI com diferentes períodos ($n = 15$, $|\psi\langle 1 \rangle| = 9110$ e $|\psi\langle 0 \rangle| = 14538$).

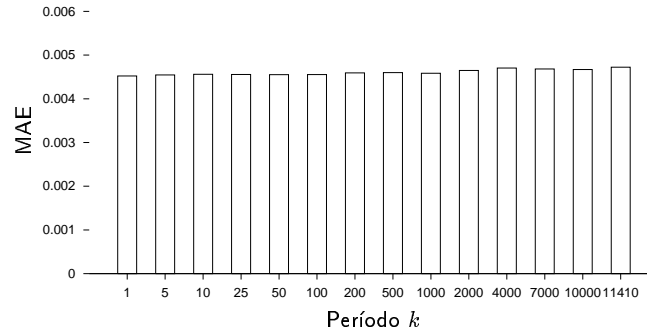


Figura 4.10: Erro MAE para ISI com diferentes períodos ($n = 17$, $|\psi\langle 1 \rangle| = 1609$ e $|\psi\langle 0 \rangle| = 11410$).

Podemos observar que há pouca diferença entre os erros MAE associados a diferentes períodos. A comparação do ISI com outros algoritmos de aprendizado não faz parte do escopo desta tese. Na prática, observou-se que o ISI apresenta uma boa capacidade de generalização para diversos problemas de processamento de imagens. Alguns resultados podem ser vistos no capítulo de aplicações ao final desta tese.

4.4 Paralelização do ISI

Os algoritmos de minimização são computacionalmente complexos, exigindo, em geral, muito tempo de processamento e espaço de memória. Uma possibilidade para reduzir esta complexidade seria a sua paralelização. Um k -cubo, $k > 0$, pode ser decomposto como a união de dois $(k - 1)$ -cubos disjuntos, os quais podem ser decompostos respectivamente como a união de dois $(k - 2)$ -cubos disjuntos e assim por diante. Portanto, pode-se considerar um k -cubo como uma união de cubos de dimensão menor. Isto sugere de imediato uma possível paralelização do algoritmo ISI. O n -cubo inicial pode ser particionado em t sub-cubos disjuntos, digamos C_1, C_2, \dots, C_t . Da mesma forma, o conjunto $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle$ pode ser subdividido em t sub-grupos, G_1, G_2, \dots, G_t . O conjunto G_i consiste de

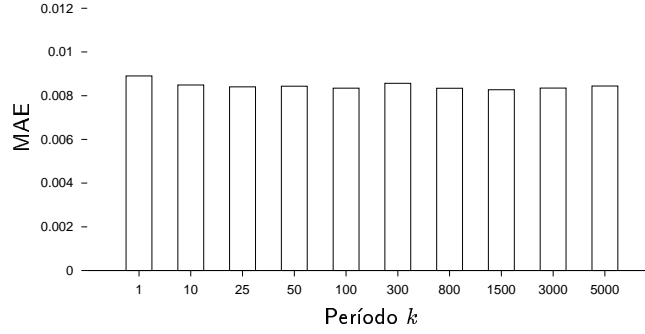


Figura 4.11: Erro MAE para ISI com diferentes períodos ($n = 17$, $|\psi\langle 1 \rangle| = 8517$ e $|\psi\langle 0 \rangle| = 12251$).

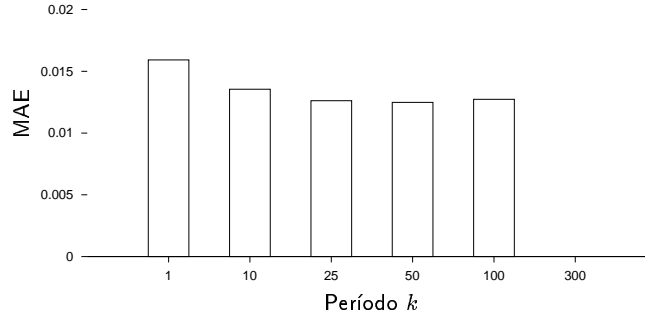


Figura 4.12: Erro MAE para ISI com diferentes períodos ($n = 25$, $|\psi\langle 1 \rangle| = 1340$ e $|\psi\langle 0 \rangle| = 8126$).

todos os padrões observados e que pertencem ao sub-cubo C_i . Supondo que existem t processadores disponíveis, cada um deles poderá processar um conjunto de exemplos G_i , a partir do intervalo inicial C_i .

Sejam dados $\mathcal{X} = \psi\langle 1 \rangle \cup \psi\langle 0 \rangle$ e o n -cubo inicial $[\emptyset, W]$. Descrevemos dois procedimentos para particionar $[\emptyset, W]$ e $\psi\langle 1 \rangle \cup \psi\langle 0 \rangle$.

- Sejam $[A, B] \subseteq \mathcal{P}(W)$, $\mathcal{X}_{[A, B]} = \{X \in \mathcal{X} : X \in [A, B]\}$ e $w_i \in W$. Particionar $[A, B]$ em w_i consiste em criar dois intervalos $[A_1, B_1] = [A, B \cap \{w_i\}^c]$ e $[A_2, B_2] = [A \cup \{w_i\}, B]$. Este particionamento implica na separação de $\mathcal{X}_{[A, B]}$ em duas subcoleções disjuntas $\mathcal{X}_{[A_1, B_1]} = \{X \in \mathcal{X}_{[A, B]} : X \in [A_1, B_1]\}$ e $\mathcal{X}_{[A_2, B_2]} = \{X \in \mathcal{X}_{[A, B]} : X \in [A_2, B_2]\}$.
- Procedimento 1 : seja dado um valor $nmax$, o número máximo de elementos permitidos em cada parte da partição. Primeiramente, escolhe-se um elemento $w_i \in W$ tal que o partionamento de $[\emptyset, W]$ em w_i separe mais eqüitativamente os elementos em \mathcal{X} . Particiona-se recursivamente cada umas das partes até que o número de elementos de \mathcal{X} contidos nela seja menor que $nmax$. A posição de particionamento w_i é sempre escolhida de forma a produzir uma divisão mais eqüitativa possível dos elementos contidos na parte sendo particionada.
- Procedimento 2 : seja dada uma ordenação dos elementos de W , w_1, w_2, \dots, w_n , estabelecendo a seqüência de posições de particionamento. Primeiramente, particiona-se $[\emptyset, W]$ em w_1 , em seguida particiona-se cada uma das partes em w_2 e assim por diante, até que o número total de partes seja maior que um dado valor especificado.

Todas as partes produzidas pelo primeiro procedimento tendem a conter aproximadamente $|\mathcal{X}|/nmax$ elementos. Esta regularidade não acontece no segundo procedimento. Um algoritmo recursivo que implementa o primeiro procedimento é apresentado a seguir (Algoritmo 4.3).

Entrada: $W, \psi\langle 1 \rangle \cup \psi\langle 0 \rangle, nmax$ (número máximo de elementos em cada parte da partição)
Saída: Partição $\{\mathcal{P}_i, i \in I\}$, subcoleções $\{\mathcal{X}_i : i \in I\}$.

1. $next \leftarrow 1$; $P_1 \leftarrow [\emptyset, W]$; $\mathcal{X}_1 \leftarrow \psi\langle 1 \rangle \cup \psi\langle 0 \rangle$;
2. $Particione([\emptyset, W], \mathcal{X}_1, 1)$;
3. **Retorne** partição $\{\mathcal{P}_i : 1 \leq i \leq next\}$ e subconjuntos $\{\mathcal{X}_i : 1 \leq i \leq next\}$.

```

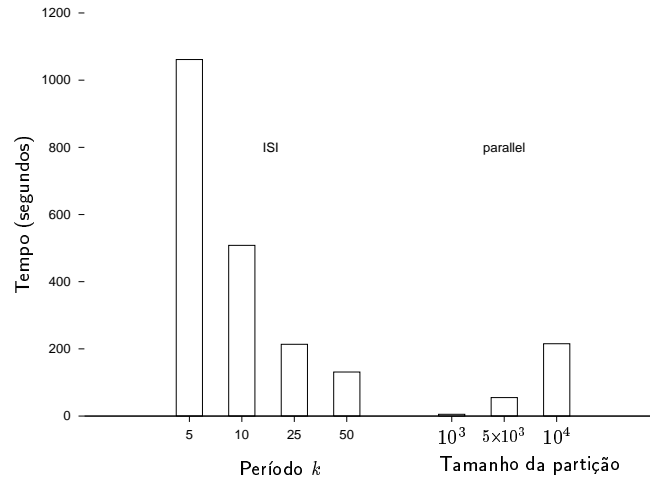
Particione([A, B],  $\mathcal{X}, i$ ) {
  Para todo  $w_j \in W$  seja
     $t_a \leftarrow |\{X \in \mathcal{X} : X \in [A, B \cap \{w_j\}^c]\}|$  ;
     $t_b \leftarrow |\{X \in \mathcal{X} : X \in [A \cup \{w_j\}, B]\}|$  ;
     $d[j] \leftarrow |t_a - t_b|$  ;
   $m \leftarrow arg_j min\{d[j] : 1 \leq j \leq |W|\}$  ;
   $\mathcal{X}_a \leftarrow \{X \in \mathcal{X} : X \in [A, B \cap \{w_m\}^c]\}$  ;
   $\mathcal{X}_b \leftarrow \{X \in \mathcal{X} : X \in [A \cup \{w_m\}, B]\}$  ;
  Se  $|\mathcal{X}_a| \leq nmax$  {
     $\mathcal{X}_i \leftarrow \mathcal{X}_a$  ;  $\mathcal{P}_i \leftarrow [A, B \cap \{w_m\}^c]$  ;
  }
  senão  $Particione([A, B \cap \{w_m\}^c], \mathcal{X}_a, i)$  ;
  Se  $|\mathcal{X}_b| \leq nmax$  {
     $next \leftarrow next + 1$  ;
     $\mathcal{X}_{next} \leftarrow \mathcal{X}_b$  ;  $\mathcal{P}_{next} \leftarrow [A \cup \{w_m\}, B]$  ;
  }
  senão  $Particione([A \cup \{w_m\}, B], \mathcal{X}_b, next)$  ;
}

```

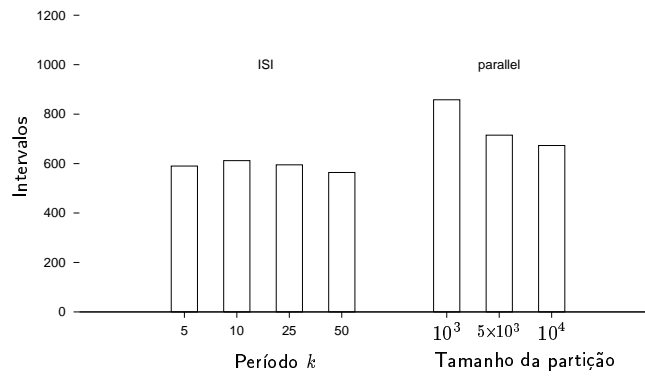
Algoritmo 4.3 Algoritmo de Particionamento.

A partição gerada pelo algoritmo pode ser interpretada como sendo a partição associada a uma expansão de Shannon, conforme visto na seção 2.3. O algoritmo ISI pode ser adaptado para que o intervalo inicial seja um intervalo qualquer. Logo, o ISI pode ser aplicado sobre cada uma das partes $[A_i, B_i]$ e seus respectivos elementos \mathcal{X}_i para gerar as funções ψ_i , associadas as partes da partição. A função sobre o intervalo $[\emptyset, W]$ é dada por $\psi = \bigvee_{i=1}^t \psi_i$.

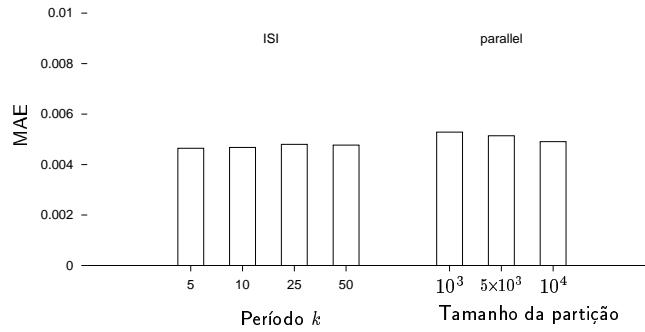
Os gráficos da figura 4.13 e 4.14 comparam o tempo de processamento, o tamanho da base e o erro MAE das funções geradas pelo algoritmo ISI e pelo algoritmo ISI que explora as partições de $\mathcal{P}(W)$. Os resultados referentes ao ISI estão mais a esquerda, enquanto os referentes ao ISI paralelo estão mais a direita. No eixo das abscissas está indicado o período k para o ISI e número máximo de elementos na partição para o ISI paralelo.



(a) Tempo de processamento

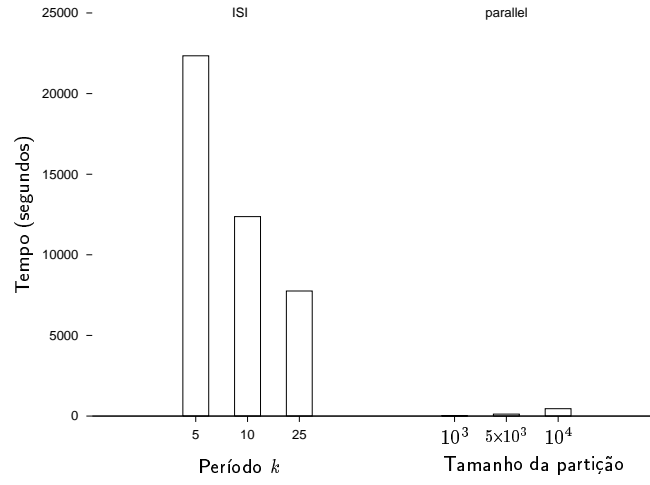


(b) Número de intervalos

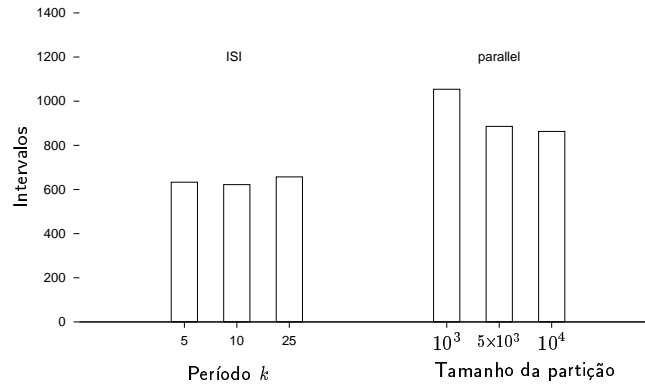


(c) Erro MAE

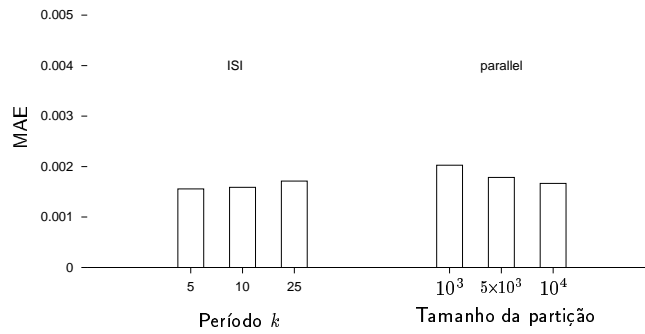
Figura 4.13: ISI \times ISI paralelo ($n = 17$, $|\psi\langle 1 \rangle| = 10351$ e $|\psi\langle 0 \rangle| = 16107$).



(a) Tempo de processamento



(b) Número de intervalos



(c) Erro MAE

Figura 4.14: ISI \times ISI paralelo ($n = 49$, $|\psi\langle 1 \rangle| = 9924$ e $|\psi\langle 0 \rangle| = 122284$).

Mesmo com a utilização de um único processador, o tempo total gasto por este algoritmo paralelo é inferior ao tempo gasto pelo ISI. Para explicar esta afirmação, analisamos as curvas no gráfico da figura 4.15. Seja T o número máximo de elementos em cada parte da partição. Se $|\psi\langle 1 \rangle \cup \psi\langle 0 \rangle| \leq T$

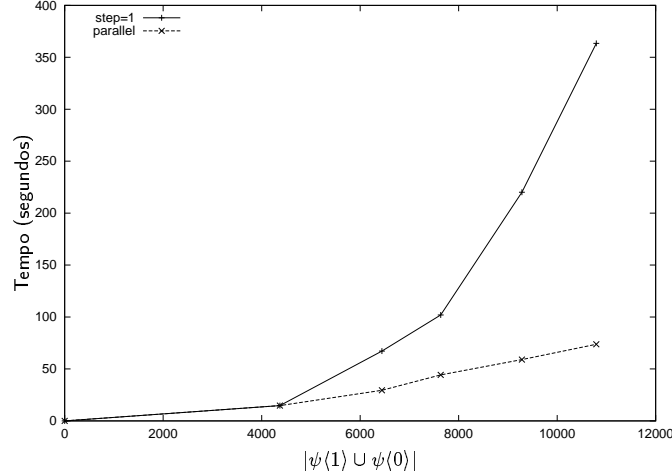


Figura 4.15: ISI \times ISI paralelo : tempo de processamento variando $|\psi\langle 1 \rangle \cup \psi\langle 0 \rangle|$.

então o ISI paralelo não cria partição e seu tempo de processamento é igual ao do ISI. Suponha que o tempo gasto pelo ISI para T elementos é tot_T . Se $|\psi\langle 1 \rangle \cup \psi\langle 0 \rangle| = m * T$, então o tempo gasto pelo ISI paralelo é de aproximadamente $m * tot_T$. No entanto, o tempo gasto pelo ISI não é linear no tamanho $|\psi\langle 1 \rangle \cup \psi\langle 0 \rangle|$, conforme mostra a curva da figura 4.15. Em contrapartida, tanto o tamanho da base como o MAE do operador produzido pelo algoritmo paralelo tendem a ser maiores do que o do ISI. Vale notar também que a disjunção das partes C_i pode ser explorada para uma implementação eficiente do operador projetado.

4.5 Comentários

Neste capítulo apresentamos detalhadamente o algoritmo ISI, com demonstração de sua corretude. A sua generalização (na qual os elementos extraídos são intervalos e não pontos isolados) bem como a prova da corretude são contribuições originais deste trabalho. Analisamos também algumas variações deste algoritmo e sua implicação no contexto de minimização de funções Booleanas (tamanho de representação da função) e no contexto de aprendizado computacional (capacidade de generalização expressa através da precisão MAE estimada sobre algumas imagens-teste).

O algoritmo apresentado originalmente em [157] considera quatro variantes do algoritmo : ISI-0 (que é a versão que não filtra os intervalos que não contém elementos de $\psi\langle 1 \rangle$), ISI-1 (que corresponde ao algoritmo modificado com $k = |\psi\langle 0 \rangle|$), ISI-2 (que corresponde a segunda variante discutida acima) e ISI-3 (que corresponde ao algoritmo modificado com $k = 1$). A variante ISI-3 foi mencionada como uma das mais eficientes em geral. Com o uso de um período k adequado, conseguimos uma redução de tempo de processamento da ordem de 10 ou mais em relação ao ISI-3 (período $k = 1$).

O algoritmo ISI pode, por um lado, ser essencialmente visto como um algoritmo alternativo para minimização de funções Booleanas. Ele é especialmente adaptado para os casos nos quais o número de “don’t cares” é muito grande relativamente ao total observado. A heurística (período para cálculo de cobertura) e a paralelização proposta permitem a minimização de funções Booleanas com um número

relativamente grande de variáveis. Observamos experimentalmente que os resultados obtidos com a utilização da heurística proposta são em geral muito parecidos com o resultado ótimo, em termos de número de intervalos. Por outro lado, no contexto de aprendizado computacional, o algoritmo ISI apresenta boa capacidade de generalização para problemas de processamento de imagens binárias. Para uma análise mais completa do algoritmo neste contexto, comparações com outros algoritmos seriam necessárias. Porém, por não fazer parte dos objetivos da tese, ela não é considerada neste trabalho.

Lembramos também que o critério utilizado pelos algoritmos para minimização de funções Booleanas é essencialmente algébrico. Na segunda parte dos algoritmos citados, a cobertura mínima é escolhida tomando-se como principais critérios o número de intervalos (que é minimizado) e o tamanho de cada intervalo (que é maximizado). Este critério tem como objetivo minimizar a representação da função Booleana resultante e encontra motivação na área de circuitos lógicos digitais, onde a utilização de um menor número de portas lógicas e pinos de entrada é uma preocupação constante. Porém, quando analisamos o problema no contexto de aprendizado, outros critérios poderiam ser mais interessantes. Por exemplo, os intervalos candidatos à cobertura poderiam ter como atributo um peso que depende das probabilidades de seus elementos. Assim, uma cobertura ótima não conteria necessariamente um menor número de intervalos, mas poderia ser calculada dando-se maior importância aos intervalos que agrupam objetos semelhantes (geometricamente) ou possuem elementos com características probabilísticas mais interessantes.

Capítulo 5

Projeto Multi-estágio de W-operadores

Um operador pode ser obtido compondo-se operadores mais simples. Por exemplo, um problema de contagem de objetos em uma imagem ruidosa poderia ser decomposto em quatro subproblemas : filtragem de ruído, realce de arestas, segmentação dos objetos, e contagem propriamente dita. Cada um destes subproblemas está associado a um objetivo específico. O operador final, que resolve o problema original, é obtido compondo-se seqüencialmente os operadores que resolvem os subproblemas.

Para projetar um operador podemos projetar cada um de seus componentes e compô-los de forma adequada. Esta forma de projeto é denominado *projeto multi-estágio*. O projeto de cada um dos componentes corresponde a um estágio do projeto. Em [144], os autores analisam várias formas de projeto dois-estágios. O projeto multi-estágio é também abordado no contexto de stack-filters [133].

No projeto multi-estágio, o projeto de um operador pode depender do resultado de um outro operador, como no exemplo citado acima. A tarefa de estabelecer os objetivos intermediários adequados requer conhecimentos específicos. Para evitar esta dificuldade, podemos considerar uma particular formulação na qual todos os objetivos intermediários coincidem com o objetivo final. Neste caso, o projeto multi-estágio pode ser visto como uma técnica de refinamento de soluções na qual o operador projetado aproxima-se cada vez mais do objetivo. Esta abordagem, restrita ao caso de operadores crescentes, aparece em [55] com o nome de *projeto iterativo*.

Conforme veremos na próxima seção, a composição de t operadores que dependem de janelas W_1, W_2, \dots, W_t , respectivamente, resulta em um operador que depende da janela $W_1 \oplus W_2 \oplus \dots \oplus W_t$. Portanto, projetar operadores sobre janelas pequenas iterativamente e depois compô-los é uma técnica que permite projetar operadores sobre janelas grandes. Esta técnica aproveita o fato de que o projeto de operadores sobre janelas pequenas é computacional e estatisticamente mais simples do que sobre uma janela grande.

Neste capítulo, vários aspectos da estratégia de projeto multi-estágio (no sentido iterado descrito acima), para operadores não necessariamente crescentes, são analisados sob um ponto de vista prático.

5.1 Considerações Iniciais

Seja $\Psi_1 : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ um W -operador e seja $\Psi_2 : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ um W' -operador, com as respectivas funções características ψ_1 e ψ_2 , onde $W = \{w_1, w_2, \dots, w_n\}$ e $W' = \{w'_1, w'_2, \dots, w'_m\}$. Então,

$$\begin{aligned}
 z \in \Psi_2 \Psi_1(S) &\iff \\
 &\iff \psi_2(\Psi_1(S)_{-z} \cap W') = 1 \\
 &\iff \psi_2(1_{\Psi_1(S)}(z + w'_1), 1_{\Psi_1(S)}(z + w'_2), \dots, 1_{\Psi_1(S)}(z + w'_m)) = 1 \\
 &\iff \psi_2(\psi_1(1_S(z + w'_1 + w_1), \dots, 1_S(z + w'_1 + w_n)), \dots, \psi_1(1_S(z + w'_m + w_1), \dots, 1_S(z + w'_m + w_n))) = 1
 \end{aligned} \tag{5.1}$$

O lado direito da expressão define uma função de $W \oplus W' \rightarrow \{0, 1\}$ que caracteriza o operador composição de Ψ_1 com Ψ_2 . Ele depende de valores de S na soma de Minkowski $W \oplus W'$. A figura 5.1 mostra os pontos da imagem dos quais depende o resultado da composição de dois operadores sobre a janela $W_{1 \times 3}$, cujo centro coincide com a origem.

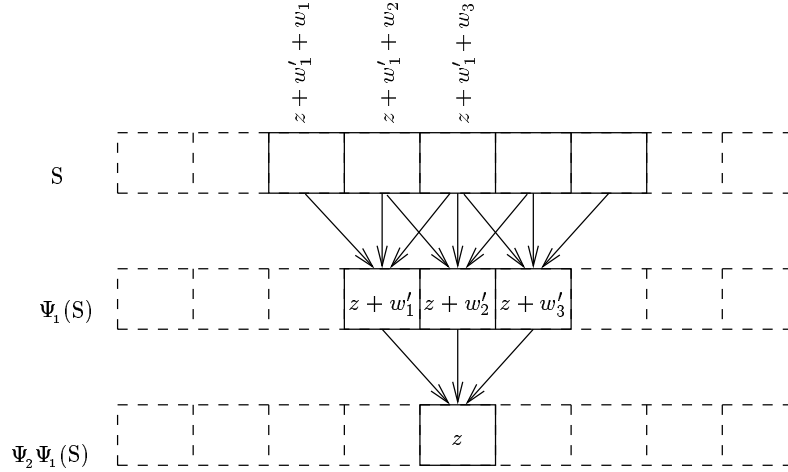


Figura 5.1: Composição de operadores.

A composição de t operadores, $\Psi_1, \Psi_2, \dots, \Psi_t$, localmente definidos, respectivamente, sobre as janelas W_i , $i = 1, 2, \dots, t$ é denotada $\Psi^t = \Psi_t \Psi_{t-1} \dots \Psi_2 \Psi_1$ (ou também $\Psi^t = \Psi_t \Psi^{t-1}$). O operador Ψ^t é um W^t -operador, onde $W^t = W_1 \oplus W_2 \oplus \dots \oplus W_t$. Se $W_i = W$ para todo $i = 1, 2, \dots, t$, então Ψ^t é um tW -operador, onde tW é a soma de Minkowski de W por ele mesmo $t - 1$ vezes.

Nem todos os tW -operadores podem ser expressos como uma composição de t operadores. Isto significa que a classe de operadores compostos Ψ^t é menor que a classe de tW -operadores. Consequentemente, se $\Psi_{\text{opt}, tW}$ denota o tW -operador ótimo e Ψ_{opt}^i denota o operador ótimo consistindo da composição de i W -operadores, vale a relação

$$MAE\langle \Psi_{\text{opt}, tW} \rangle \leq MAE\langle \Psi_{\text{opt}}^t \rangle \leq MAE\langle \Psi_{\text{opt}}^{t-1} \rangle \leq \dots \leq MAE\langle \Psi_{\text{opt}}^1 \rangle. \tag{5.2}$$

Vamos supor que Ψ_{opt}^t é o operador composto ótimo, para t iterações sobre uma janela W' . Note que Ψ_{opt}^t é composto por t operadores, cada qual definidos por $2^{|W'|}$ parâmetros. Logo, para projetar $\hat{\Psi}_{\text{opt}}^t$ (o estimador de Ψ_{opt}^t), o número total de parâmetros que precisam ser estimados é $t * 2^{|W'|}$. Por

outro lado, seja $\Psi_{\text{opt},W}$, $W = tW'$, o operador (não-iterado) ótimo sobre W . Neste caso, o número de parâmetros que precisam ser estimados para estimar $\hat{\Psi}_{\text{opt},W}$ é $2^{|W|}$. Por exemplo, se $W' = W_{3 \times 3}$ e $W = W_{5 \times 5}$, isto é, $t = 2$ e $W = 2W'$, então para projetar um operador de 2 iterações sobre W' precisamos estimar $2 \cdot 2^9$ parâmetros, enquanto que para projetar um operador sobre W precisamos estimar 2^{25} parâmetros.

Conseqüentemente, um operador composto ótimo Ψ_{opt}^t possui menos parâmetros a serem estimados e portanto a sua precisão, $E[\Delta(\hat{\Psi}_{\text{opt}}^t, \Psi_{\text{opt}}^t)]$, pode ser muito melhor que a precisão $E[\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})]$ de $\hat{\Psi}_{\text{opt},W}$. A iteração é estatisticamente vantajosa se

$$E[\Delta(\hat{\Psi}_{\text{opt}}^t, \Psi_{\text{opt}}^t)] + \Delta(\Psi_{\text{opt}}^t, \Psi_{\text{opt},W}) < E[\Delta(\hat{\Psi}_{\text{opt},W}, \Psi_{\text{opt},W})] \quad (5.3)$$

Se a desigualdade acima é satisfeita, isto significa que projetar operadores sobre uma janela grande através da composição de operadores projetados sobre janelas pequenas resulta em melhores operadores do que projetar um operador diretamente sobre a janela grande.

5.1.1 Detalhes do Projeto

Comentamos anteriormente que, do ponto de vista do usuário, uma das dificuldades do projeto multi-estágio seria a especificação dos objetivos intermediários e, portanto, uma alternativa seria fixar todos os objetivos intermediários iguais ao objetivo final. Analisamos aqui estas duas possibilidades sob o ponto de vista de projeto.

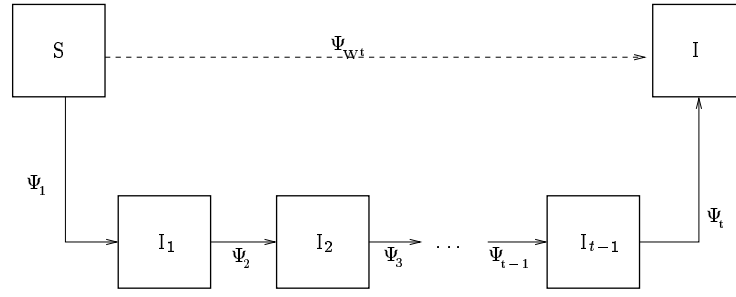
A primeira possibilidade, com objetivos intermediários distintos, é ilustrada na figura 5.2(a). Aqui, o objetivo final é a minimização global da diferença entre $\Psi^t(\mathbf{S}) = \Psi_t(\Psi_{t-1}(\dots(\Psi_1(\mathbf{S})))\dots)$ e \mathbf{I}_t . Este não é necessariamente obtido minimizando-se os erros entre $\Psi_1(\mathbf{S})$ e \mathbf{I}_1 , $\Psi_2(\mathbf{I}_1)$ e \mathbf{I}_2 , e assim por diante. Na prática, ele depende da minimização entre $\Psi_1(\mathbf{S})$ e \mathbf{I}_1 , $\Psi_2(\Psi_1(\mathbf{S}))$ e \mathbf{I}_2 , e assim por diante. Isto significa que a minimização envolve um processo de otimização global com relação ao espaço de todos os parâmetros que definem Ψ^t . Em outras palavras, os t componentes devem ser analisados conjuntamente, resultando em um problema combinatório altamente complexo. Em [144], algoritmos para $t = 2$, com restrições sobre o tipo de operadores em um dos estágios, foram propostos; porém eles não são aplicáveis para $t > 2$.

A segunda possibilidade, na qual todos os objetivos intermediários coincidem com o objetivo final, é ilustrada na figura 5.2(b). Neste caso, a otimização é local, isto é, minimizam-se as diferenças entre $\Psi^1(\mathbf{S})$ e \mathbf{I} , $\Psi^2(\mathbf{S})$ e \mathbf{I} , $\Psi^3(\mathbf{S})$ e \mathbf{I} , e assim por diante. Isto significa que cada um dos componentes pode ser projetado individualmente: \mathbf{S} é utilizada para projetar Ψ_1 , $\Psi_1(\mathbf{S})$ para projetar Ψ_2 , $\Psi^2(\mathbf{S})$ para projetar Ψ_3 , e assim por diante.

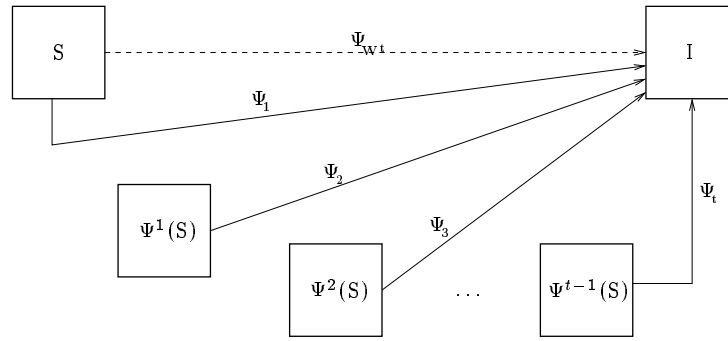
No projeto multi-estágio automático, cada um dos componentes do operador deve ser projetado automaticamente. Sob o ponto de vista de projeto, a segunda possibilidade pode ser facilmente implementada pois cada um dos componentes pode ser individualmente projetado. No caso da primeira possibilidade, além da dificuldade do ponto de vista de usuário, existe também a dificuldade do ponto de vista de projeto.

5.1.2 Detalhes dos Experimentos

O restante deste capítulo analisa vários aspectos relacionados ao projeto multi-estágio de operadores (no sentido iterativo). Estas análises são efetuadas a partir de observações experimentais.



(a) Objetivos intermediários distintos



(b) Objetivos intermediários iguais

Figura 5.2: Duas possibilidades para projeto multi-estágio.

O número de iterações t foi definido arbitrariamente para cada experimento assim como as respectivas janelas W_i , $i = 1, 2, 3, \dots, t$. Dadas algumas imagens de treinamento (S, I) , realizações de um processo (S, I) , um primeiro operador Ψ_1 é projetado utilizando-se a janela W_1 de forma que $\Psi_1(S)$ seja um estimador de I . Em seguida, um segundo operador Ψ_2 é projetado sobre a janela W_2 de modo que $\Psi_2(\Psi_1(S))$ seja um estimador de I . Da mesma forma, um terceiro operador é projetado de forma que $\Psi_3(\Psi_2(\Psi_1(S)))$ seja um estimador de I , e assim por diante. Portanto, para t iterações o processo todo é composto de t estágios de treinamento. Diversas janelas¹ foram utilizadas nos experimentos.

Existem algumas possibilidades quanto a questão de como os dados de treinamento são utilizados neste processo. Uma estratégia simples é a utilização de todo o conjunto de dados de treinamento em cada um dos estágios de treinamento. Uma outra possibilidade consiste em subdividir o conjunto de dados de treinamento em t subconjuntos (disjuntos ou não) e utilizar um para cada estágio de treinamento. Inicialmente consideramos o uso de todo o conjunto de imagens em cada estágio de treinamento, e na última seção analisamos o caso em que os conjunto de imagens de treinamento são subdivididos em subconjuntos.

Para os testes experimentais realizados, utilizamos tanto imagens sintéticas quanto imagens reais. Cada grupo de imagens foi separado em dois subgrupos: as *imagens de treinamento*, utilizadas para projetar os operadores, e as *imagens de validação*, utilizadas para avaliar o desempenho dos

¹Ver início do texto para as notações utilizadas.

operadores projetados. A seguir descrevemos e mostramos partes de algumas destas imagens.

Imagens com ruído sal-e-pimenta : este grupo contém imagens corrompidas por ruído do tipo sal-e-pimenta (ruídos aditivos e subtrativos pontuais), ambos com 15% de densidade (figura 5.3), isto é, a probabilidade de que um ponto da imagem tenha sido corrompido é 0.15. Diferentes realizações do mesmo tipo de ruído foram simuladas sobre uma mesma imagem, de tamanho 256×256 , formando vários pares de imagens observadas-ideais (neste caso as imagens ideais são sempre as mesmas).

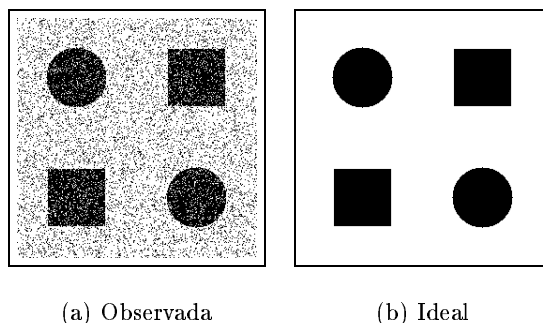


Figura 5.3: Imagens com ruído do tipo sal-e-pimenta.

Modelo Booleano [123] e ruído Booleano: este grupo consiste de imagens geradas seguindo-se o processo descrito a seguir. Primeiramente, alguns pontos na imagem são sorteados de acordo com o modelo binomial, com intensidade $D = 0.002$ sendo a probabilidade de um certo ponto ser escolhido. A seguir, consideramos um processo de grãos primários que geram conjuntos aleatórios, os quais são colocados sobre os pontos escolhidos. O grão primário considerado aqui é um quadrado $Z \times Z$, onde Z é uma aproximação discreta de uma variável aleatória Gaussiana com média $\mu = 11$ e variância $\sigma^2 = 4$. Os processos de ruído aditivo e subtrativo são também processos aleatórios discretos independentes do processo das imagens. O grão primário do processo de ruído é um subconjunto do quadrado 3×3 , com o tamanho de cada grão uniformemente distribuído sobre o intervalo $[1, 5]$. A intensidade dos ruídos aditivo e subtrativo são dados, respectivamente, por D_a e D_s . A figura 5.1.2 mostra parte de uma realização do processo de imagem e parte de três realizações de imagens ruidosas com diferente intensidade de ruído. O tamanho da imagem é 512×512 .

Reconhecimento de Caracteres: algumas páginas escolhidas ao acaso de dois livros foram digitalizadas a 200dpi (“dots per inch” ou seja, pontos por polegada), e em seguida binarizadas através de uma simples limiarização. Para cada imagem do primeiro livro, foi gerada uma imagem contendo somente as letras *s* minúsculas. Similarmente, para cada imagem do segundo livro foi gerada uma imagem contendo somente as letras ‘a’ minúsculas. Em seguida, todas as imagens foram reduzidas por um fator de 2, eliminando-se uma a cada duas colunas e uma a cada duas linhas das imagens. Uma pequena parte de duas destas imagens são mostradas nas figuras 5.5 e 5.6. Os tamanhos das imagens reduzidas são aproximadamente 375×310 para o primeiro livro, e 410×335 para o segundo.

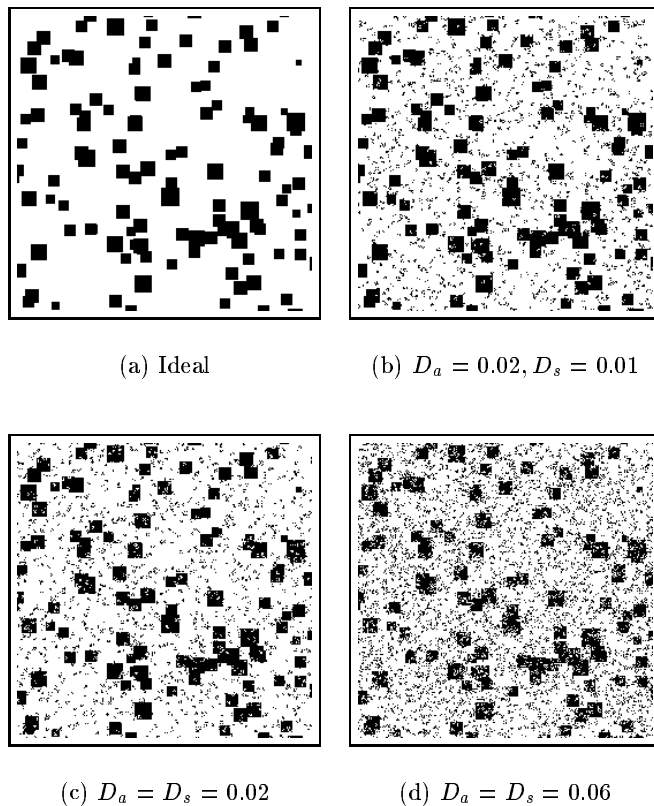


Figura 5.4: Modelo Booleano.

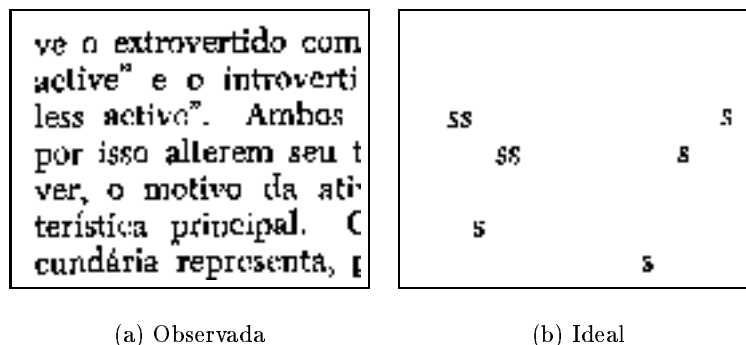
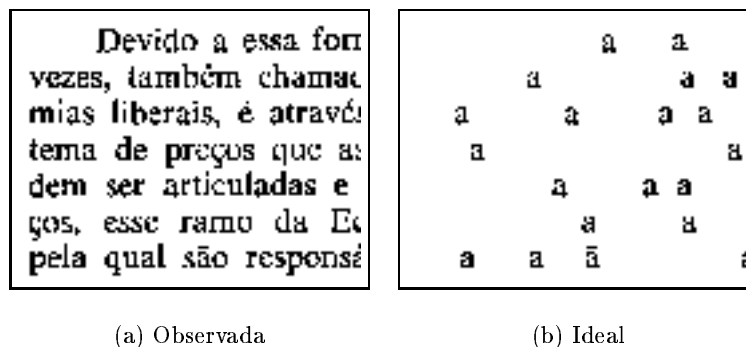
5.2 Número Crescente de Iterações

Apesar de sabermos que, do ponto de vista teórico, o MAE decresce a medida que o número de iterações aumenta (Eq. 5.2), na prática os operadores são projetados a partir de um número finito de dados e portanto não se pode afirmar o mesmo em relação aos operadores estimados. Nesta seção analisamos experimentalmente como os operadores estimados se comportam em termos de erro, para um número fixo de dados de treinamento e para uma janela fixa, a medida que o número de iterações aumenta.

A figura 5.7 mostra 4 gráficos de erro MAE. Cada gráfico mostra a variação do erro MAE a medida que o número de iterações aumenta. O erro das imagens originais não é ilustrado nos gráficos e, portanto, não podemos ver a diminuição de erro devido ao primeiro operador. Em geral, a redução de erro devido ao primeiro operador é significativo, muito mais do que a redução devida aos operadores subsequentes.

Podemos observar, na maioria dos gráficos, que existe uma redução visível do erro nas primeiras três ou quatro iterações e que, a partir desse ponto, a redução de erro torna-se desprezível (as curvas tornam-se quase horizontais). Em geral, o ponto no qual a curva começa a tornar-se horizontal pode estar indicando que iterações adicionais não serão úteis para reduzir o erro. Este ponto pode ser interpretado como o número de iterações a ser considerado para o problema específico em questão.

No segundo gráfico podemos observar um ligeiro aumento de erro da segunda para a terceira

Figura 5.5: Reconhecimento de letra *s* (livro 1).Figura 5.6: Reconhecimento de letra *a* (livro 2).

iteração. Observamos também casos nos quais as curvas apresentam pequenas oscilações. Estes comportamentos ocorrem pois os operadores são estimados a partir de dados. Além disso, quando as iterações são sobre janelas grandes, as curvas são quase horizontais mesmo nas primeiras iterações. Isto significa que praticamente não há redução de erro da primeira iteração para as iterações subseqüentes. Este caso será retomado mais adiante.

5.3 Número Crescente de Dados de Treinamento

Fixada uma janela, sabemos que os operadores projetados a partir de uma quantidade maior de dados de treinamento são mais precisos que os operadores projetados a partir de uma pequena quantidade de dados de treinamento.

Nesta seção analisamos o que acontece com a curva de erro ao longo das iterações se variamos a quantidade de dados de treinamento. Cada um dos gráficos das figuras 5.8 e 5.9 mostra as curvas de erro para diferentes quantidades de exemplos de treinamento. Os rótulos que acompanham as curvas indicam o número de pares de imagens utilizados no treinamento.

Em geral, o aumento de exemplos de treinamento implica em aumento da precisão do operador projetado, de forma consistente para todas as iterações. Esta relação é bastante óbvia enquanto a

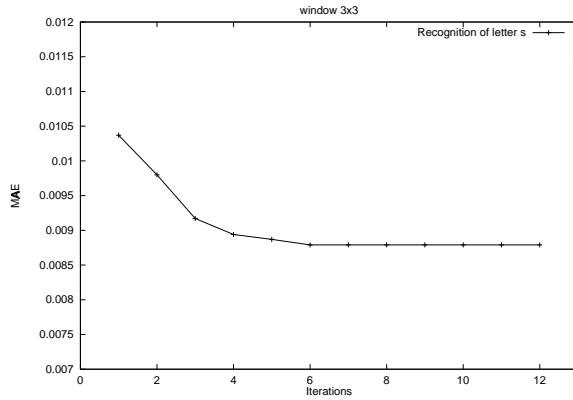
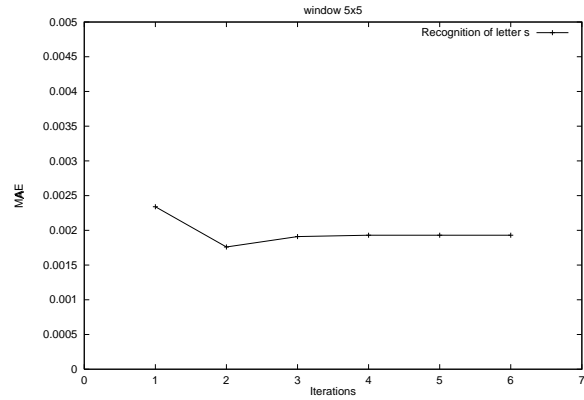
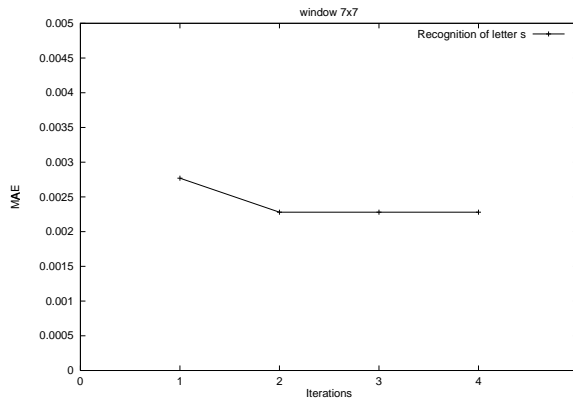
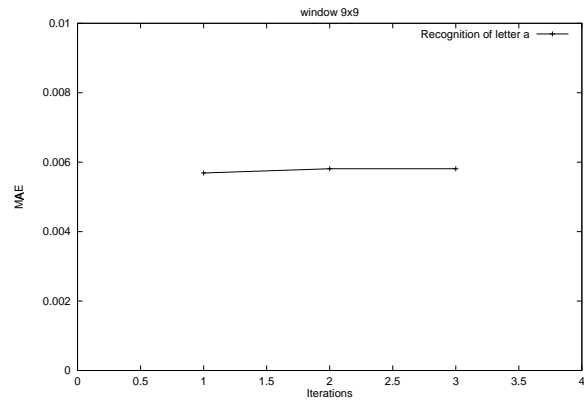
(a) Janela $W_{3 \times 3}$, reconhecimento de caractere(b) Janela $W_{5 \times 5}$, reconhecimento de caractere(c) Janela $W_{7 \times 7}$, reconhecimento de caractere(d) Janela $W_{9 \times 9}$, reconhecimento de caractere

Figura 5.7: Evolução das curvas de erro ao longo das iterações. Janela de iteração e quantidade de imagens de treinamento fixas ao longo das iterações.

quantidade de dados de treinamento é relativamente pequena. Por exemplo, no gráfico da figura 5.8b, a diferença de erro entre as duas curvas de cima (operadores projetados com um e dois pares de imagens de treinamento, respectivamente) é bastante significativa. No entanto, a diferença de erro entre as duas curvas de baixo (operadores projetados com 12 e 18 pares de imagens de treinamento, respectivamente) é bem pequena. A medida que a quantidade de dados de treinamento aumenta, as curvas tendem a convergir para a curva ótima. Conseqüentemente, a convergência pode estar indicando que o operador projetado já está próximo do operador ótimo para aquela janela. Observe que esta convergência não é aparente para janelas grandes, pois nesses casos o número de dados de treinamento necessários para projetar operadores precisos é, em geral, bem grande.

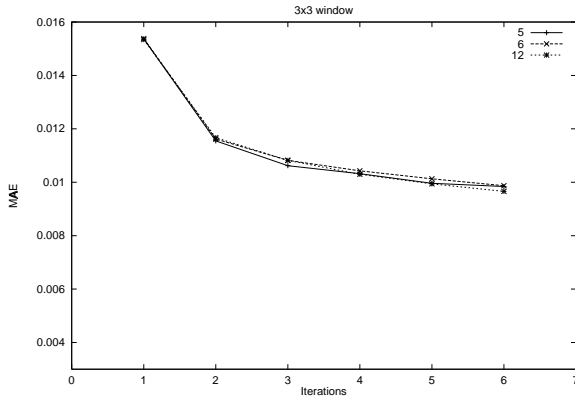
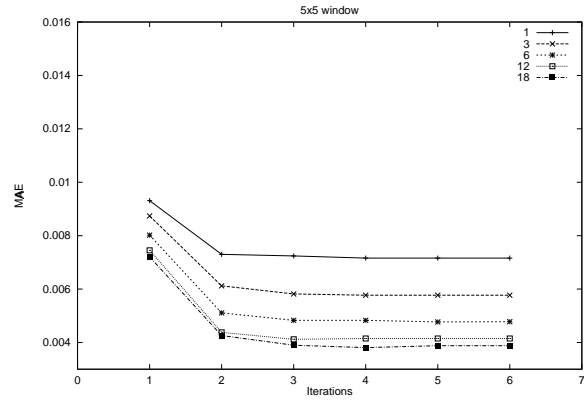
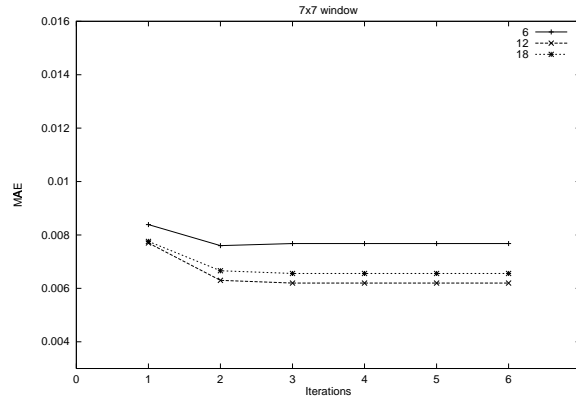
(a) Ruído Booleano, janela $W_{3 \times 3}$ (b) Ruído Booleano, janela $W_{5 \times 5}$ (c) Ruído Booleano, janela $W_{7 \times 7}$

Figura 5.8: Evolução das curvas de erro ao longo das iterações, para diferentes quantidades de exemplos de treinamento (modelo Booleano – $D_a = 0.02, D_s = 0.02$).

5.4 Iterações sobre Diferentes Janelas

Sabe-se que projetar operadores sobre janelas pequenas é muito mais simples tanto estatística quanto computacionalmente do que para janelas grandes. Nesta seção comparamos operadores iterados sobre diversas janelas, com especial atenção ao efeito devido à quantidade de exemplos disponíveis.

Primeiramente analisamos os casos que podem ocorrer quando comparamos iterações sobre uma janela W' e iterações sobre a janela $W = W' \oplus W'$.

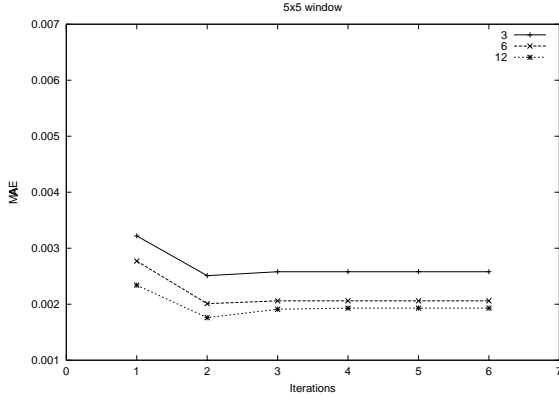
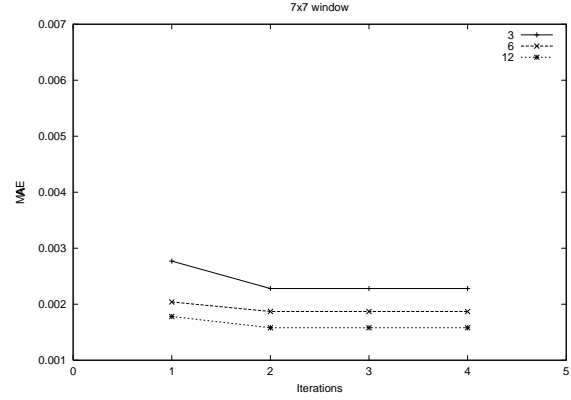
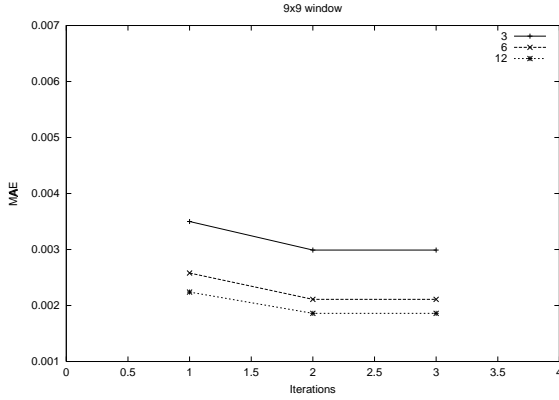
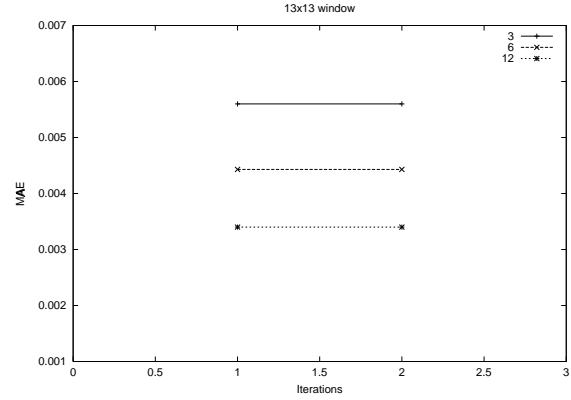
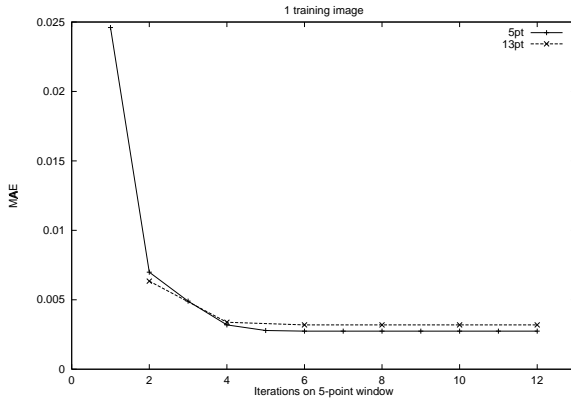
(a) Reconhecimento de caractere, janela $W_{5 \times 5}$ (b) Reconhecimento de caractere, janela $W_{7 \times 7}$ (c) Reconhecimento de caractere, janela $W_{9 \times 9}$ (d) Reconhecimento de caractere, janela $W_{13 \times 13}$

Figura 5.9: Evolução das curvas de erro ao longo das iterações, para diferentes quantidades de exemplos de treinamento (reconhecimento de letra *s*).

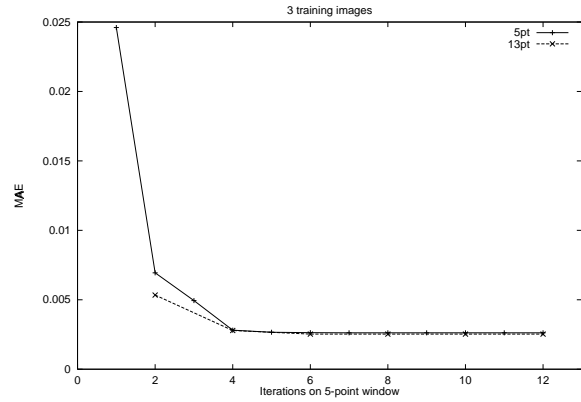
Caso a

Consideramos aqui as imagens com ruído do tipo sal-e-pimenta. A figura 5.10 ilustra o desempenho de operadores iterados sobre W_5 contra aqueles iterados sobre a janela W_{13} , para diferentes quantidades de exemplos de treinamento. Note que a medida que o número de exemplos de treinamento aumenta, as iterações sobre ambas as janelas convergem para um mesmo erro. Esta convergência pode ser um indicador de que a quantidade de dados de treinamento já é estatisticamente suficiente para ambas as janelas e que dados de treinamento adicionais irão melhorar a precisão do operador apenas por uma fração que pode ser desprezível. Neste caso, notamos que o operador iterado sobre a janela menor possui uma precisão similar, apenas um pouco inferior, ao operador iterado sobre a janela maior, independente da quantidade de exemplos utilizada.

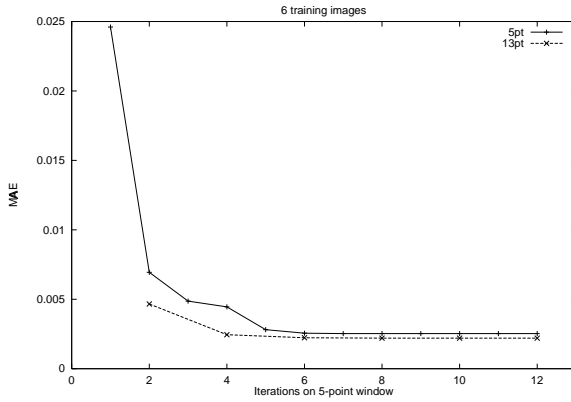
A figura 5.11 mostra os resultados de quatro iterações sobre a janela cruz, enquanto a Fig. 5.12 mostra os resultados de duas iterações sobre a janela W_{13} , ambos a partir de três pares de imagens



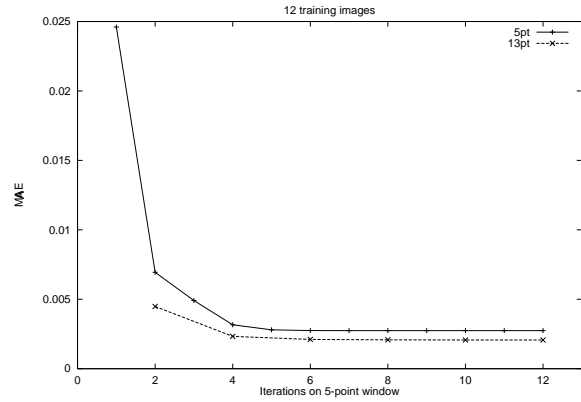
(a) Ruído sal-e-pimenta, 1 imagem de treinamento



(b) Ruído sal-e-pimenta, 3 imagens de treinamento



(c) Ruído sal-e-pimenta, 6 imagens de treinamento



(d) Ruído sal-e-pimenta, 12 imagens de treinamento

Figura 5.10: Filtragem de ruído sal-e-pimenta : iterações sobre $W_5 \times$ iterações sobre W_{13} .

de treinamento. Note que o resultado após duas iterações sobre a janela W_5 é similar ao resultado após uma iteração sobre a janela W_{13} , conforme mostram os gráficos anteriores.

Caso b

Consideramos agora o modelo Booleano descrito anteriormente. A figura 5.13 mostra o desempenho dos operadores iterados sobre a janela $W_{3 \times 3}$ contra os iterados sobre a janela $W_{5 \times 5}$, para diferente quantidade de exemplos de treinamento. Para uma pequena quantidade de exemplos de treinamento, os operadores projetados sobre a janela pequena superam em precisão aqueles projetados sobre a janela maior para a mesma quantidade de exemplos. No entanto, com o aumento de exemplos de treinamento, operadores projetados sobre a janela maior apresentam aumento significativo de precisão, enquanto aqueles projetados sobre a janela pequena apresentam apenas um pequeno aumento.

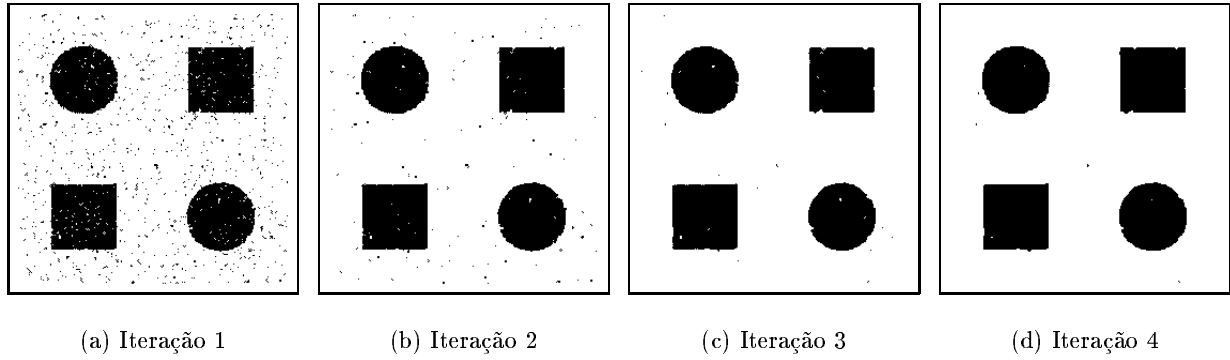


Figura 5.11: Resultados de 1 a 4-iterações (janela W_5).

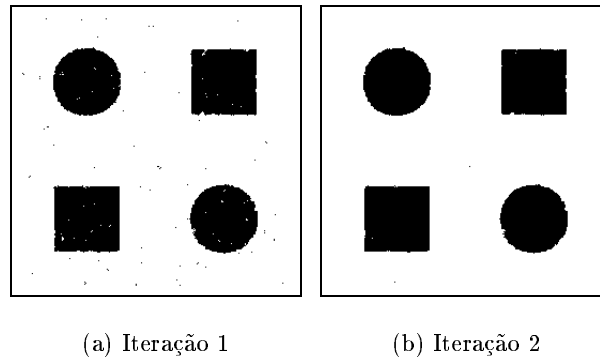
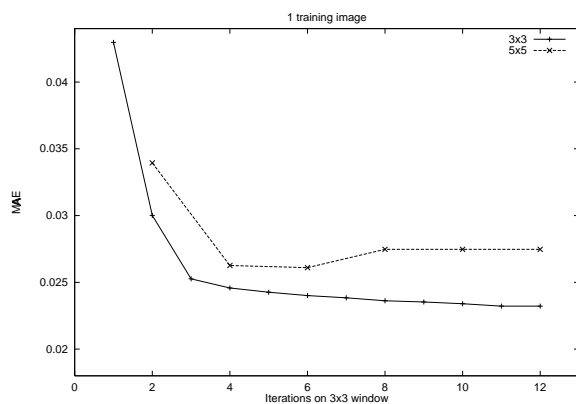


Figura 5.12: Resultados de 1 e 2-iterações (janela W_{13}).

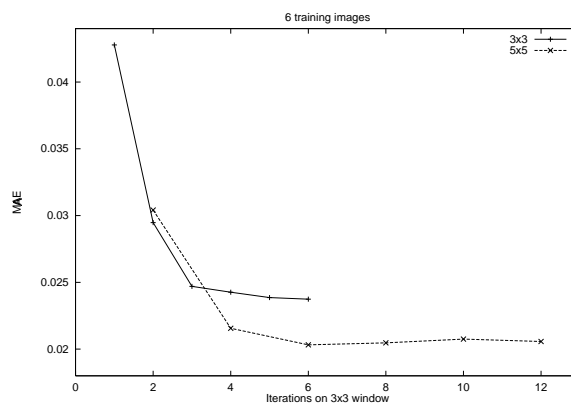
Com isso, para uma quantidade grande de exemplos de treinamento, os operadores iterados sobre a janela maior apresentam desempenho superior aos iterados sobre a janela menor.

Caso c

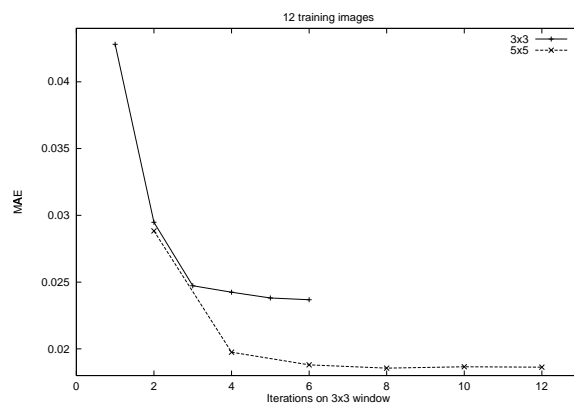
Neste caso, consideramos o problema de reconhecimento de caracteres. A figura 5.14 mostra o desempenho de operadores iterados sobre a janela $W_{3 \times 3}$ contra o desempenho dos operadores iterados sobre a janela $W_{5 \times 5}$, para diferente quantidade de exemplos de treinamento. Iterações sobre a janela pequena produzem operadores com desempenho muito fraco em relação aos operadores iterados sobre a janela maior, mesmo para quantidades pequenas de exemplos de treinamento. Conjeturamos que este desempenho pobre do operador sobre a janela menor é consequência da inabilidade do operador, em qualquer estágio de iteração, de capturar as características das formas (importantes nas letras), devido ao tamanho reduzido da janela.



(a) Ruído Booleano, 1 imagem de treinamento

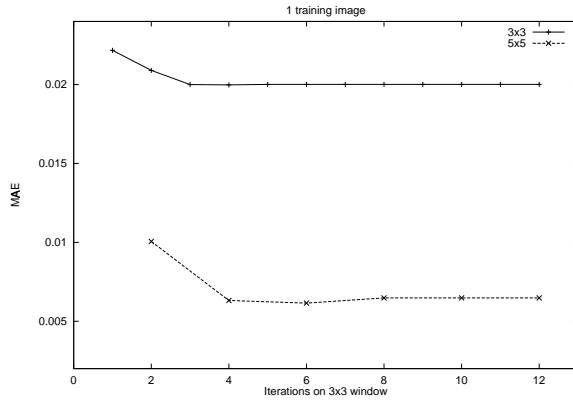


(b) Ruído Booleano, 6 imagens de treinamento

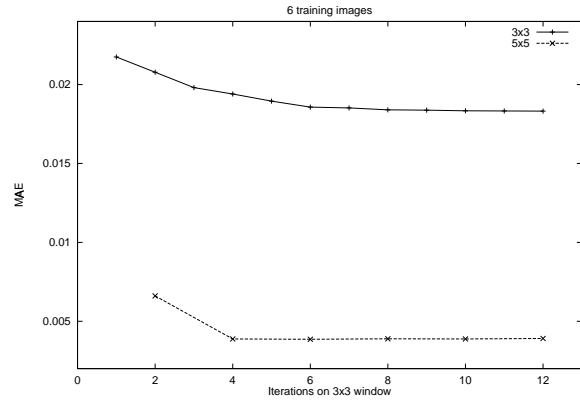


(c) Ruído Booleano, 12 imagens de treinamento

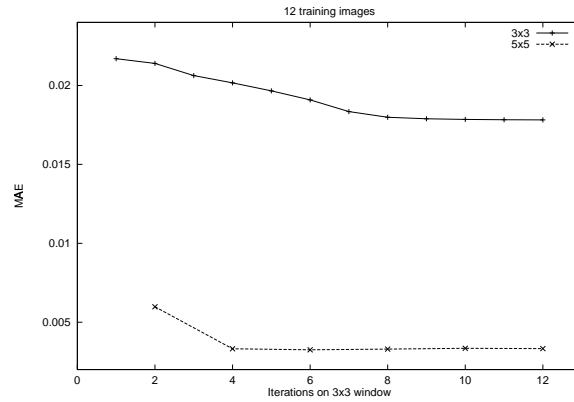
Figura 5.13: Filtragem de ruído Booleano: iterações sobre $W_{3 \times 3}$ \times iterações sobre $W_{5 \times 5}$.



(a) Reconhecimento de caracteres, 1 imagem de treinamento



(b) Reconhecimento de caracteres, 6 imagens de treinamento



(c) Reconhecimento de caracteres, 12 imagens de treinamento

Figura 5.14: Reconhecimento de caractere: iterações sobre $W_{3 \times 3} \times$ iterações sobre $W_{5 \times 5}$.

5.4.1 Exemplos

Suponha que o tamanho máximo da janela está estabelecido e, portanto, qualquer operador projetado não deve depender de janela maior que a fixada. Esta é uma hipótese razoável pois, na prática, a implementação de operadores pode estar sujeito a várias limitações, tais como a velocidade de aplicação dos mesmos e, principalmente, a quantidade de portas lógicas (ou circuitos) para a implementação de hardware dedicado, como o caso de impressoras. Desta forma, iteração sobre janelas pequenas e números menores de iterações tornam-se características desejáveis. Portanto, gostaríamos de saber qual é o melhor operador multi-estágio que pode ser projetado a partir de um conjunto de exemplos disponíveis e que cuja janela não ultrapassa o tamanho máximo estabelecido.

Aqui investigamos, experimentalmente, o comportamento de vários operadores iterados, sobre diferentes janelas e diferentes quantidades de exemplos de treinamento. A janela máxima é fixada como sendo um múltiplo da janela $W_{3 \times 3}$, e consideramos a tabela de equivalência 5.1 para efeitos de comparação.

2 $W_{3 \times 3}$	1 $W_{5 \times 5}$			
3 $W_{3 \times 3}$		1 $W_{7 \times 7}$		
4 $W_{3 \times 3}$	2 $W_{5 \times 5}$		1 $W_{9 \times 9}$	
5 $W_{3 \times 3}$				
6 $W_{3 \times 3}$	3 $W_{5 \times 5}$	2 $W_{7 \times 7}$		1 $W_{13 \times 13}$
7 $W_{3 \times 3}$				
8 $W_{3 \times 3}$	4 $W_{5 \times 5}$		2 $W_{9 \times 9}$	
9 $W_{3 \times 3}$		3 $W_{7 \times 7}$		
10 $W_{3 \times 3}$	5 $W_{5 \times 5}$			
11 $W_{3 \times 3}$				
12 $W_{3 \times 3}$	6 $W_{5 \times 5}$	4 $W_{7 \times 7}$	3 $W_{9 \times 9}$	2 $W_{13 \times 13}$

Tabela 5.1: Iterações equivalentes

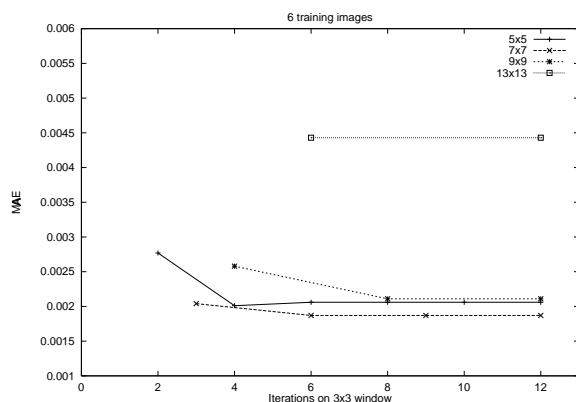
Por exemplo, a quinta linha da tabela indica que 6 iterações sobre a janela $W_{3 \times 3}$ são equivalentes a 3 iterações sobre a janela $W_{5 \times 5}$; o qual, por sua vez, é equivalente a 2 iterações sobre a janela $W_{7 \times 7}$, que é equivalente a 1 iteração sobre a janela $W_{13 \times 13}$. Se a janela máxima é $W_{13 \times 13}$, qual das possibilidades resulta no melhor operador ?

A figura 5.15 mostra o desempenho de operadores iterados sobre diferentes janelas (seguindo a tabela 5.1), para diferentes quantidades de exemplos de treinamento, para o problema de reconhecimento de letra “s”. Omitimos a curva para a janela $W_{3 \times 3}$ pois o erro é muito maior comparado ao das demais janelas, prejudicando a visualização da curvas.

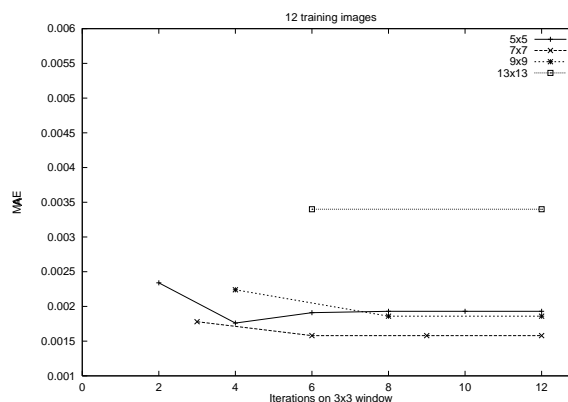
A figura 5.16 mostra uma pequena parte de uma imagem de teste e os resultados obtidos aplicando-se os operadores de 1 e 2 iterações sobre a janela $W_{7 \times 7}$.

A figura 5.17 mostra os desempenho dos operadores iterados sobre diferentes janelas, para diferentes quantidades de imagens de treinamento, para o modelo de ruído Booleano.

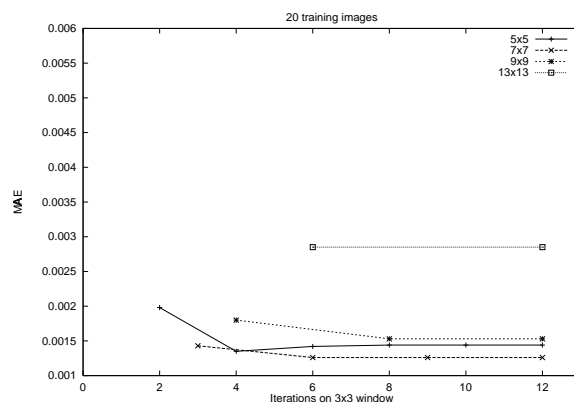
A figura 5.18 mostra uma pequena parte de uma imagem de teste e os resultados obtidos aplicando-se os operadores de 1 a 6 iterações, treinados a partir de 6 pares de imagens sobre uma janela $W_{5 \times 5}$, para o modelo $D_a = 0.06, D_s = 0.06$.



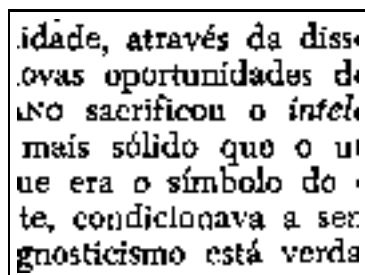
(a) 6 imagens de treinamento



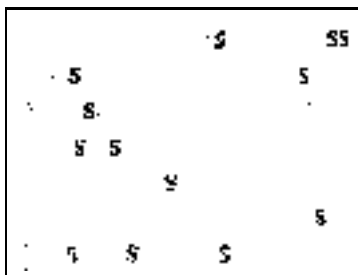
(b) 12 imagens de treinamento



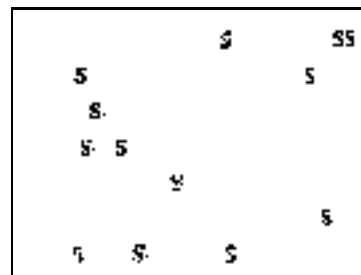
(c) 20 imagens de treinamento

Figura 5.15: Diferentes janelas de iteração para o reconhecimento da letra *s*.

(a) Teste



(b) Resultado da iteração 1



(c) Resultado da iteração 2

Figura 5.16: Resultados para reconhecimento de caractere “s”. Duas iterações sobre a janela $W_{7 \times 7}$, usando 6 pares de imagens de treinamento.

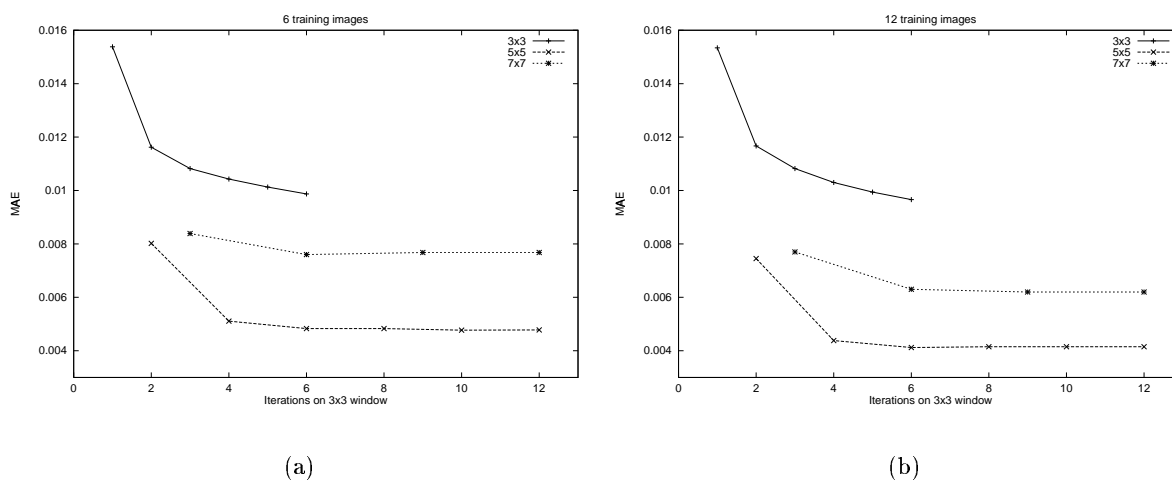


Figura 5.17: Diferentes janelas de iteração – modelo Booleano ($D_a = D_s = 0.02$).

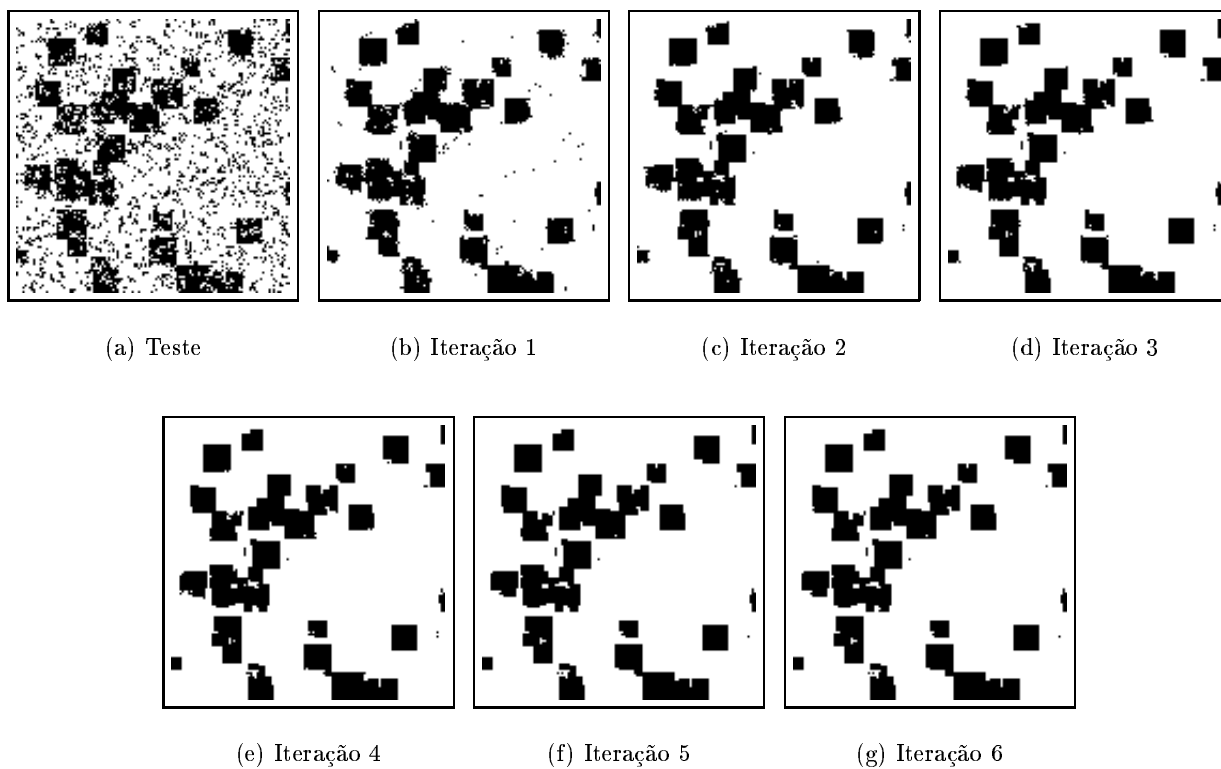


Figura 5.18: Modelo Booleano ($D_a = 0.06, D_s = 0.06$). Imagem de teste e resultados de 1 a 6 iterações.

5.5 Diferentes Formas de Treinamento

O erro dos operadores projetados sobre janelas grandes em relação as imagens de treinamento é, em geral, muito próximo de zero (veja a Fig. 5.19). Em outras palavras, para uma dada imagem de treinamento S , $\Psi(S)$ é muito próxima da respectiva imagem ideal I . Portanto, o segundo operador Ψ_2 projetado a partir dos pares $(\Psi_1(S), I)$ e os operadores das iterações subseqüentes pouco ou nada precisam fazer para melhorar $\Psi(S)$. Entretanto, quando estes operadores são aplicados sobre as imagens que não foram utilizadas para o treinamento, eles podem apresentar um desempenho bem diferente. O operador Ψ_1 provavelmente não produzirá erro tão próximo de zero e os operadores subseqüentes não serão, por sua vez, capazes de corrigir os erros deixados por Ψ_1 , uma vez que eles não foram treinados para isso. Esta é uma das razões que explicam porquê os operadores projetados sobre janelas grandes possuem curvas de erro quase horizontais, ou às vezes até crescentes.

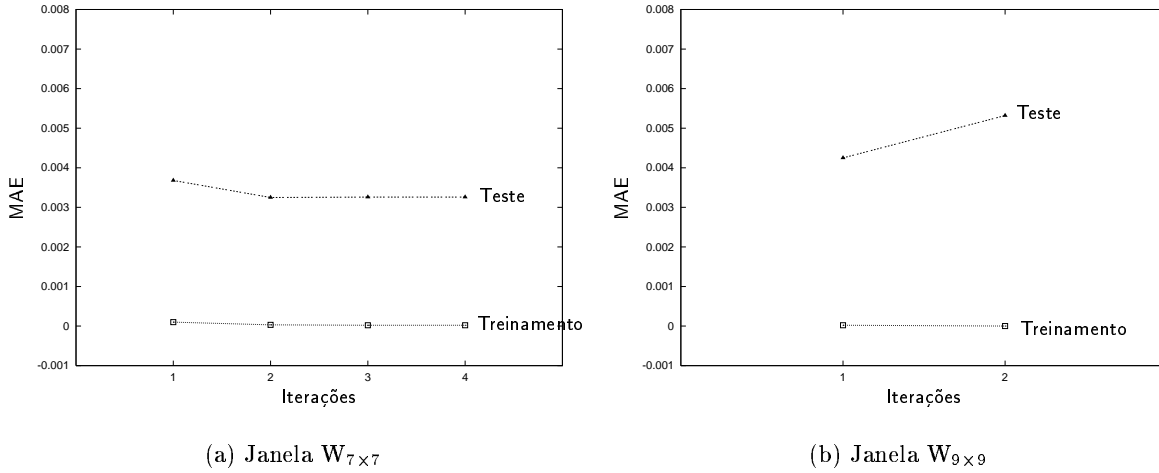


Figura 5.19: Erro sobre imagens de teste e de treinamento.

5.5.1 Subdividindo os Dados de Treinamento

O efeito descrito acima pode ser atenuado usando-se imagens de treinamento adicionais de um estágio de iteração para os subseqüentes. Entretanto, como o número de imagens de treinamento disponível é limitado em geral, este aumento de dados de uma iteração para outra não pode ser arbitrária. Aqui consideramos a seguinte estratégia: em vez de utilizarmos todas as N imagens para o treinamento, utilizamos ciclicamente um subconjunto de M ($M \leq N$) imagens em cada iteração. Fazendo isso, algumas imagens que não foram utilizadas em uma iteração serão utilizadas na iteração subseqüente. Por exemplo, se consideramos $N = 10$ e $M = 6$, então na primeira iteração as imagens a serem utilizadas são $\{1, 2, 3, 4, 5, 6\}$, na segunda iteração serão utilizadas as imagens $\{7, 8, 9, 10, 1, 2\}$, na terceira as imagens $\{3, 4, 5, 6, 7, 8\}$, e assim por diante. Se $N = 10$ e $M = 8$, então na primeira iteração serão utilizadas as imagens $\{1, 2, 3, 4, 5, 6, 7, 8\}$, na segunda as imagens $\{9, 10, 1, 2, 3, 4, 5, 6\}$, e assim por diante. Em outras palavras, as imagens são numeradas de 1 a N e são utilizadas ciclicamente.

As figuras de 5.20 a 5.24 mostram o desempenho dos operadores treinados para $N = 4$, $N = 6$,

$N = 11$, $N = 15$ e $N = 20$, respectivamente, sobre as imagens de teste, com M variando de $(N+1)/2$ a N . O eixo das abscissas indica o valor de M e os rótulos das curvas indicam o estágio de iteração. Nas primeiras iterações, quanto maior o valor de M , menor o erro MAE. No entanto, nas últimas iterações, o MAE é mínimo para M menor que N (isto é, para casos em que um subconjunto das imagens de treinamento foi utilizado em cada estágio de treinamento em vez do conjunto todo). Logo, o uso de todas as imagens de treinamento em todas os estágios de treinamento pode não resultar, necessariamente, no melhor operador multi-estágio (em termos estatísticos).

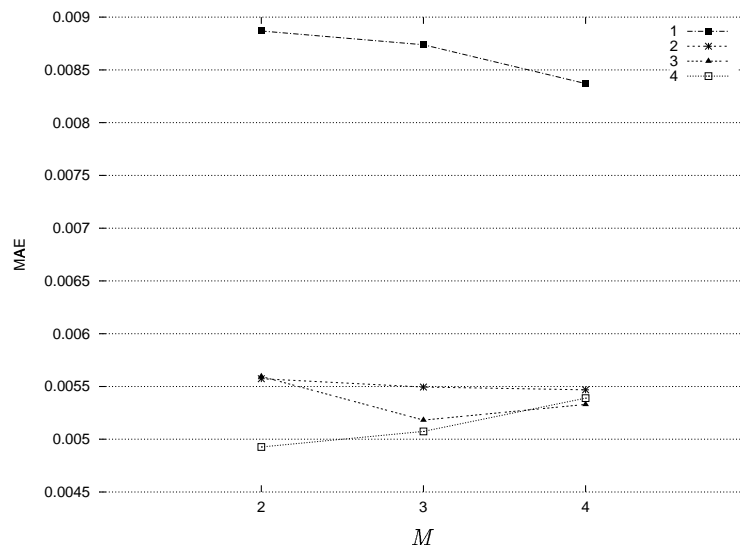


Figura 5.20: Desempenho sobre as imagens de teste ($N = 4$).

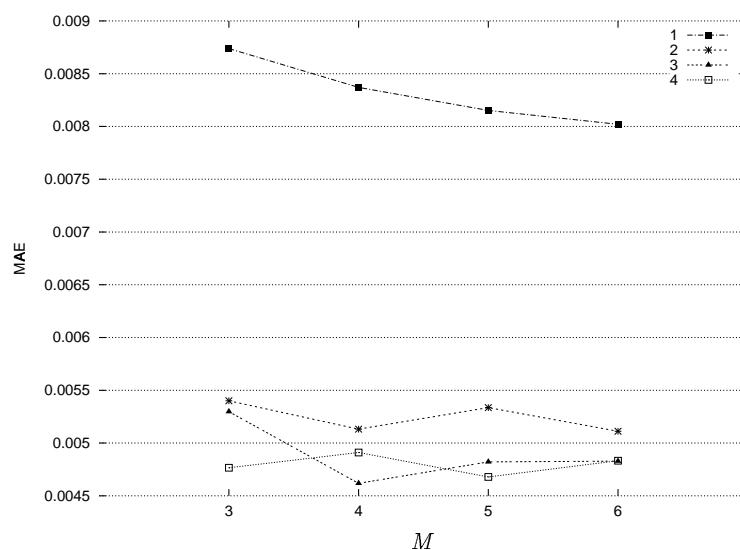


Figura 5.21: Desempenho sobre as imagens de teste ($N = 6$).

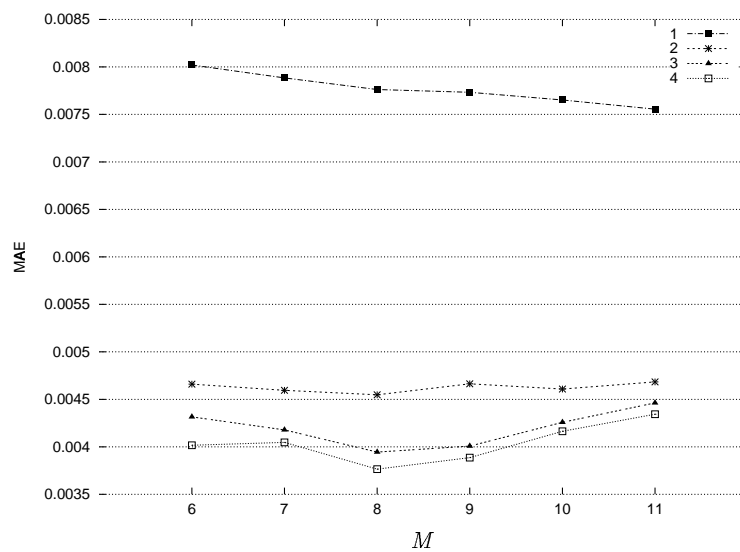


Figura 5.22: Desempenho sobre as imagens de teste ($N = 11$).

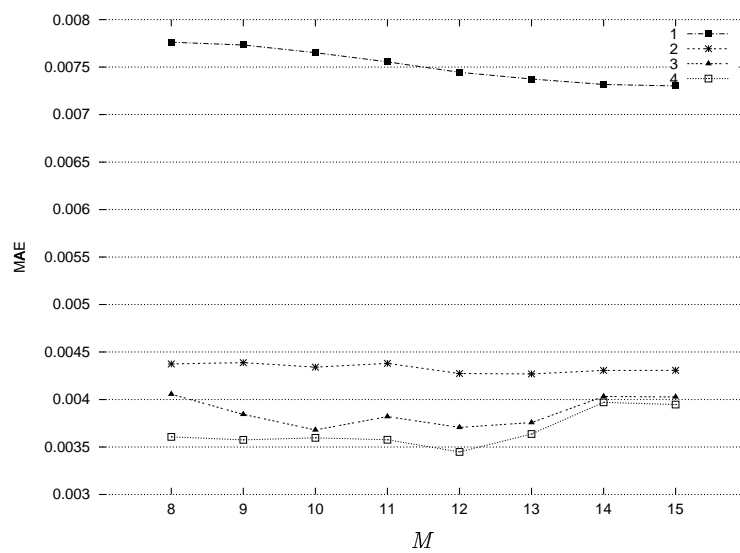


Figura 5.23: Desempenho sobre as imagens de teste ($N = 15$).

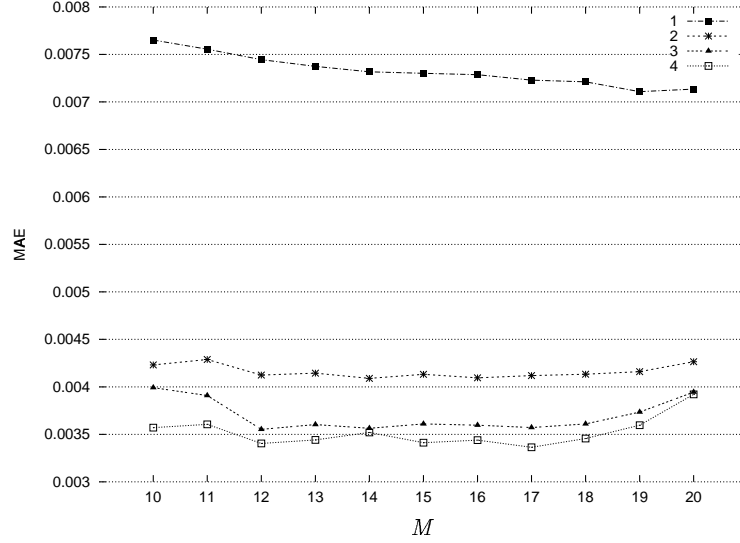


Figura 5.24: Desempenho sobre as imagens de teste ($N = 20$).

Por que subdividir os dados de treinamento pode ser vantajoso ?

Alguns dos resultados experimentais indicam que, para janelas grandes, operadores obtidos utilizando-se parte dos dados de treinamento em cada estágio de treinamento resulta em operadores com melhor desempenho do que aqueles obtidos utilizando-se todos os dados de treinamento. A seguir analisamos este fenômeno para entender porquê ele acontece. Para evitar notações complicadas, restringimo-nos ao caso de dois estágios de iteração, cuja análise pode ser estendida para um número de iterações quaisquer.

Seja \mathcal{T} uma coleção de dados de treinamento, isto é, $\mathcal{T} = \{(S_1, I_1), (S_2, I_2), \dots, (S_N, I_N)\}$, $N > 0$. No caso em que todos os exemplos de treinamento são utilizados em ambos os estágios de treinamento, o operador da primeira iteração $\hat{\Psi}_{1,\mathcal{T}}$ é estimado a partir de \mathcal{T} e o operador da segunda iteração $\hat{\Psi}_{2,\mathcal{T}}$ é estimado a partir de $\{(\hat{\Psi}_{1,\mathcal{T}}(S_1), I_1), (\hat{\Psi}_{1,\mathcal{T}}(S_2), I_2), \dots, (\hat{\Psi}_{1,\mathcal{T}}(S_N), I_N)\}$. Podemos escrever o erro MAE de $\hat{\Psi}_{1,\mathcal{T}}$ da seguinte forma :

$$MAE\langle\hat{\Psi}_{1,\mathcal{T}}\rangle = MAE\langle\hat{\Psi}_{2,\mathcal{T}}\hat{\Psi}_{1,\mathcal{T}}\rangle + \Delta(\hat{\Psi}_{1,\mathcal{T}}, \hat{\Psi}_{2,\mathcal{T}}\hat{\Psi}_{1,\mathcal{T}}). \quad (5.4)$$

No caso em que apenas parte dos exemplos de treinamento é utilizado em cada estágio de treinamento, seja $\mathcal{T}_1 \subset \mathcal{T}$ o conjunto de exemplos utilizados para projetar o operador de primeira iteração $\hat{\Psi}_{1,\mathcal{T}_1}$ e seja $\mathcal{T}_2 \subset \mathcal{T}$ o conjunto de exemplos utilizados para projetar o operador de segunda iteração $\hat{\Psi}_{2,\mathcal{T}_2}$. Neste caso, $\hat{\Psi}_{1,\mathcal{T}_1}$ é estimado a partir de $(S_i, I_i) \in \mathcal{T}_1$ e $\hat{\Psi}_{2,\mathcal{T}_2}$ é estimado a partir de $(\hat{\Psi}_{1,\mathcal{T}_1}(S_i), I_i)$ onde $(S_i, I_i) \in \mathcal{T}_2$. Em geral, temos que

$$MAE\langle\hat{\Psi}_{1,\mathcal{T}_1}\rangle \geq MAE\langle\hat{\Psi}_{1,\mathcal{T}}\rangle \quad (5.5)$$

pois $\mathcal{T}_1 \subset \mathcal{T}$. Podemos escrever,

$$MAE\langle\hat{\Psi}_{1,\mathcal{T}_1}\rangle = MAE\langle\hat{\Psi}_{1,\mathcal{T}}\rangle + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}}) \quad (5.6)$$

Além disso,

$$MAE\langle\hat{\Psi}_{1,\mathcal{T}_1}\rangle = MAE\langle\hat{\Psi}_{2,\mathcal{T}_2}\hat{\Psi}_{1,\mathcal{T}_1}\rangle + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2}\hat{\Psi}_{1,\mathcal{T}_1}) \quad (5.7)$$

Das equações 5.6 e 5.7 obtemos

$$MAE\langle\hat{\Psi}_{1,\mathcal{T}}\rangle + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}}) = MAE\langle\hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}\rangle + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}) \quad (5.8)$$

Substituindo $MAE\langle\hat{\Psi}_{1,\mathcal{T}}\rangle$ pela equação 5.4, obtemos

$$MAE\langle\hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}\rangle + \Delta(\hat{\Psi}_{1,\mathcal{T}}, \hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}) + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}}) = MAE\langle\hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}\rangle + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}) \quad (5.9)$$

$$MAE\langle\hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}\rangle - MAE\langle\hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}\rangle = \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}) - \Delta(\hat{\Psi}_{1,\mathcal{T}}, \hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}) - \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}}) \quad (5.10)$$

Logo,

$$\begin{aligned} MAE\langle\hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}\rangle - MAE\langle\hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}\rangle &\geq 0 \\ \Downarrow \\ \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}) &\geq \Delta(\hat{\Psi}_{1,\mathcal{T}}, \hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}) + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}}). \end{aligned}$$

Em outras palavras, a subdivisão de dados pode ser vantajosa ($MAE\langle\hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}\rangle < MAE\langle\hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}\rangle$) se, e somente se, a melhora obtida pelo operador de segunda iteração ($\Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1})$) for maior que a soma da melhora devida ao segundo operador (no caso em que se usa todos os exemplos de treinamento, $\Delta(\hat{\Psi}_{1,\mathcal{T}}, \hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}})$) mais o aumento de erro de $\hat{\Psi}_{1,\mathcal{T}_1}$ em relação ao erro de $\hat{\Psi}_{1,\mathcal{T}}$ ($\Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}})$) devido ao fato de utilizar menos exemplos na primeira iteração.

Analisamos agora em quais situações a condição acima pode ocorrer. Seja $m_0 = M(n, \delta)$ o maior número de exemplos de treinamento para o qual o erro de treinamento de $\hat{\Psi}_{1,\mathcal{T}}$ é menor que δ . Para um valor de δ fixo, $M(n, \delta)$ é uma função de n , crescente em n . Logo, para n grande, é possível termos um valor relativamente grande para m_0 e ainda assim termos erro de treinamento menor que δ . Se δ é bem pequeno, então $\hat{\Psi}_{1,\mathcal{T}}(S_i)$ é bem próximo de I_i e portanto $\hat{\Psi}_{2,\mathcal{T}} \approx I$ (I é o operador identidade). Neste caso,

$$MAE\langle\hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}\rangle \approx MAE\langle\hat{\Psi}_{1,\mathcal{T}}\rangle \quad (5.11)$$

e portanto $\Delta(\hat{\Psi}_{1,\mathcal{T}}, \hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}) \approx 0$. Relativo ao segundo caso, o erro de treinamento de $\hat{\Psi}_{1,\mathcal{T}_1}$ sobre \mathcal{T}_1 é menor que δ , mas sobre \mathcal{T} pode ser significativamente maior que δ . Logo, $\hat{\Psi}_{2,\mathcal{T}_2}$ pode ser bem diferente do operador identidade e em consequência $\Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1})$ pode ser significativo. Se tivermos $\Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}})$ também pequeno, então podemos ter $\Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{2,\mathcal{T}_2} \hat{\Psi}_{1,\mathcal{T}_1}) > \Delta(\hat{\Psi}_{1,\mathcal{T}}, \hat{\Psi}_{2,\mathcal{T}} \hat{\Psi}_{1,\mathcal{T}}) + \Delta(\hat{\Psi}_{1,\mathcal{T}_1}, \hat{\Psi}_{1,\mathcal{T}})$, que é a situação na qual subdividir os exemplos de treinamento é vantajoso.

5.5.2 Variando as Janelas de um Estágio para Outro

Até agora consideramos e analisamos casos nos quais os operadores em todos os estágios foram projetados sobre uma mesma janela. Porém, os operadores multi-estágio não precisam necessariamente ter todos os componentes projetados sobre uma mesma janela.

De fato, observamos experimentalmente casos nos quais a utilização de janelas diferentes de uma iteração para outra produz melhores resultados em comparação ao uso de uma única janela. A figura 5.25(a) mostra a curva de erro para 3 iterações sobre uma janela $W_{7 \times 7}$, contra a curva de erro do operador iterado sobre a sequência de janelas $W_{7 \times 7}$, $W_{5 \times 5}$ e $W_{3 \times 3}$. Note que $W_{7 \times 7} \oplus W_{7 \times 7} =$

$W_{7 \times 7} \oplus W_{5 \times 5} \oplus W_{3 \times 3}$. Portanto, tanto o operador de dois estágios sobre a janela $W_{7 \times 7}$, quanto o operador de três estágios sobre as janelas $W_{7 \times 7}$, $W_{5 \times 5}$ e $W_{3 \times 3}$, respectivamente, são operadores definidos sobre a janela $W_{7 \times 7} \oplus W_{7 \times 7}$.

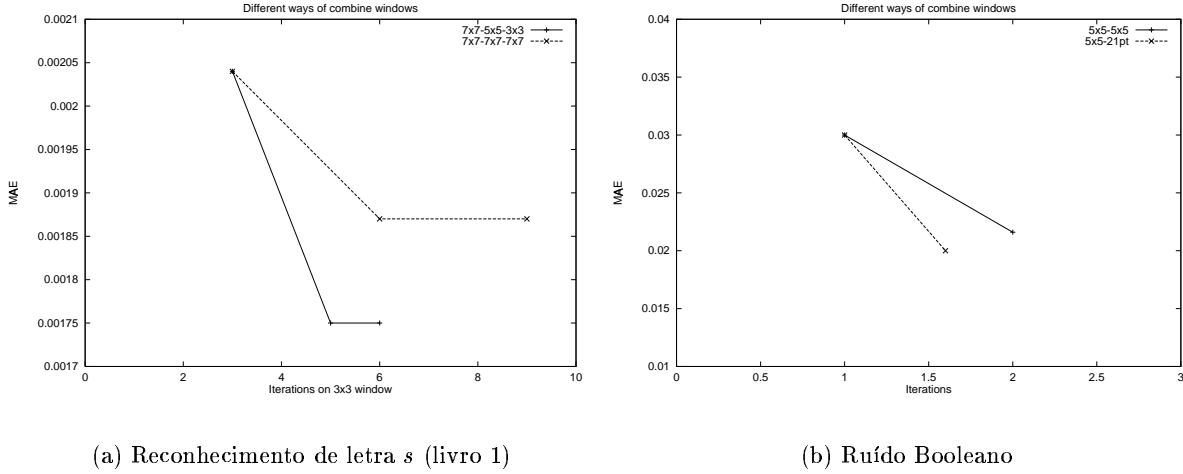


Figura 5.25: Diferentes janelas de uma iteração para outra.

A figura 5.25(b) mostra a curva de erro do operador de 2 iterações sobre a janela $W_{5 \times 5}$, contra a curva de erro do operador de dois estágios, sobre as janelas $W_{5 \times 5}$ e W_{21} pontos (a janela $W_{5 \times 5}$ sem os quatro pontos dos cantos), respectivamente. Mesmo que $W_{5 \times 5} \oplus W_{5 \times 5} \supseteq W_{5 \times 5} \oplus W_{21}$, o segundo operador apresenta desempenho superior ao primeiro.

Para concluir esta seção, vamos relembrar a questão da escolha da janela no caso de projeto simples (de um estágio). Vamos supor três janelas W_1 , W_2 e W_3 tais que $W_1 \subset W_2 \subset W_3$. Sejam Ψ_{opt, W_1} , Ψ_{opt, W_2} e Ψ_{opt, W_3} os operadores ótimos nos espaços Ψ_{W_1} , Ψ_{W_2} e Ψ_{W_3} , respectivamente. Suponha que existe um conjunto de dados de treinamento e três operadores estatisticamente ótimos, $\hat{\Psi}_{\text{opt}, W_1}$, $\hat{\Psi}_{\text{opt}, W_2}$ e $\hat{\Psi}_{\text{opt}, W_3}$, são projetados a partir destes exemplos.

É possível escolher as janelas de tal forma que tenhamos $MAE\langle \hat{\Psi}_{\text{opt}, W_2} \rangle < MAE\langle \hat{\Psi}_{\text{opt}, W_1} \rangle$ e $MAE\langle \hat{\Psi}_{\text{opt}, W_2} \rangle < MAE\langle \hat{\Psi}_{\text{opt}, W_3} \rangle$. No primeiro caso, $MAE\langle \Psi_{\text{opt}, W_1} \rangle - MAE\langle \Psi_{\text{opt}, W_2} \rangle > \Delta(\hat{\Psi}_{\text{opt}, W_2}, \Psi_{\text{opt}, W_2}) - \Delta(\hat{\Psi}_{\text{opt}, W_1}, \Psi_{\text{opt}, W_1})$, isto é, o erro do operador ótimo em W_1 é muito maior do que o erro do operador ótimo em W_2 ; o que acaba sendo decisivo para que o primeiro tenha desempenho inferior, mesmo que o erro de estimação no segundo seja maior. No segundo caso, $\Delta(\hat{\Psi}_{\text{opt}, W_3}, \Psi_{\text{opt}, W_3}) - \Delta(\hat{\Psi}_{\text{opt}, W_2}, \Psi_{\text{opt}, W_2}) > MAE\langle \Psi_{\text{opt}, W_2} \rangle - MAE\langle \Psi_{\text{opt}, W_3} \rangle$, isto é, o erro de estimação do terceiro operador é tão grande que a diferença de erro entre os ótimos Ψ_{opt, W_2} e Ψ_{opt, W_3} tem pouca importância na prática. Este raciocínio mostra que, para um dado número fixo de exemplos de treinamento, existe uma janela que produz o melhor operador estimado.

No caso de operadores multi-estágio iterados, existe uma janela ótima para o operador Ψ_1 da primeira iteração em relação às imagens de treinamento (\mathbf{S}, \mathbf{I}) . Uma vez que as imagens iniciais são processadas, as imagens $(\Psi_1(\mathbf{S}), \mathbf{I})$, que serão utilizadas para projetar o segundo operador, formam um outro par de processos aleatórios e, portanto, é natural que a janela ótima para este processo não seja necessariamente a mesma da primeira iteração. Portanto, saber selecionar uma janela ótima para a quantidade de dados de treinamento disponível é uma questão-chave para o problema de projeto

de operadores em geral.

5.6 Análise e Comentários

Conforme resultados já conhecidos e reforçados pelos experimentos realizados, a escolha de uma janela sobre a qual um operador será projetado não é trivial. No caso de projeto iterativo de operadores, não se pode dizer que a iteração sobre uma janela menor é sempre melhor, ou pior, do que a iteração sobre uma janela maior, pois os resultados dependem da quantidade de exemplos de treinamento disponíveis e também das imagens consideradas. Os experimentos apontam que os melhores resultados são obtidos escolhendo-se uma janela adequada para cada estágio de iteração.

Um outro aspecto que devemos considerar quando projetamos operadores multi-estágio é o custo associado à implementação do operador projetado. Como já vimos, o operador multi-estágio é um operador que consiste da composição de vários operadores. Na prática, um bom desempenho é o principal critério na escolha de um operador. Porém o custo de implementação pode ser um fator decisivo. Por exemplo, o caso (a) analisado na seção 5.4 mostra que duas iterações sobre uma janela menor resultam em um operador cujo desempenho é similar ao do operador projetado diretamente sobre a janela maior. Sabe-se que em implementações convencionais, o tempo gasto por dois W-operadores é menor que o tempo gasto pelo $W \oplus W$ -operador correspondendo à composição dos dois operadores.

No mesmo contexto, podemos considerar também o número de iterações a serem efetivamente consideradas. Em geral, $MAE\langle\hat{\Psi}^i\rangle > MAE\langle\hat{\Psi}^{i+1}\rangle$. No entanto, se $\Delta(\hat{\Psi}^i, \hat{\Psi}^{i+1})$ é pequeno, pode ser vantajoso escolhermos $\hat{\Psi}^i$ a $\hat{\Psi}^{i+1}$ para a implementação, pois o primeiro é um operador sobre uma janela menor e, portanto, seu custo de implementação é menor.

O estudo realizado neste capítulo é essencialmente sobre projeto multi-estágio de operadores com características sequenciais, isto é, projeto multi-estágio nos quais o projeto de cada um dos operadores projetados depende dos resultados do operador projetado no estágio anterior. No entanto, projeto multi-estágio de operadores com características paralelas, bem como combinações de ambos, também são possíveis. Por exemplo, a decomposição de um operador como a união entre um operador anti-extensivo e um anti-extensivo complementar permite que o operador possa ser obtido projetando-se seu componente anti-extensivo e seu componente anti-extensivo complementar.

Capítulo 6

O Uso de Operadores Projetados a Priori

Neste capítulo estamos interessados em saber se um operador projetado heurísticamente ou automaticamente pode ser utilizado para projetar operadores mais precisos.

No capítulo anterior vimos uma técnica iterativa de projeto. Cada iteração produz operadores que, compostos com os operadores das iterações anteriores, resultam em um operador mais preciso. Logo, um operador que tenha sido projetado, heurística ou automaticamente, pode ser utilizado como o operador da primeira iteração e operadores mais precisos podem ser obtidos a partir dele.

Neste capítulo estudamos uma outra técnica para melhorar a precisão de um dado operador a partir de exemplos adicionais de treinamento. A técnica proposta é baseada em chaveamentos, isto é, o valor do operador dado é alterado em alguns pontos de forma que o operador resultante seja mais preciso. Uma das motivações para este estudo é a possibilidade de termos uma representação mais compacta do operador projetado.

6.1 Descrição da Técnica Proposta

Seja Ψ_o um W -operador projetado (heurística ou automaticamente) para resolver um dado problema. Podemos expressar a diferença entre Ψ_o e o W -operador ótimo $\Psi_{opt,W}$ através da diferença simétrica $S[\Psi_o, \Psi_{opt,W}]$ (veja definição 3.2). Este pode ser expresso por $S[\Psi_o, \Psi_{opt,W}] = S^- \cup S^+$, onde

$$S^- = \{X \in \mathcal{P}(W) : \psi_o(X) = 1 \text{ e } \psi_{opt,W}(X) = 0\}$$

e

$$S^+ = \{X \in \mathcal{P}(W) : \psi_o(X) = 0 \text{ e } \psi_{opt,W}(X) = 1\}.$$

e onde ψ_o e $\psi_{opt,W}$ são, respectivamente, as funções características de Ψ_o e $\Psi_{opt,W}$. O conjunto S^- corresponde aos elementos que precisam ser retirados do núcleo de Ψ_o , enquanto S^+ são os que precisam ser acrescentados, para se obter $\psi_{opt,W}$ a partir de Ψ_o .

O operador que resulta ao chavearmos o valor de ψ_o nos elementos em $S[\Psi_o, \Psi_{opt,W}]$ é o próprio operador $\psi_{opt,W}$. Mais precisamente, sejam Ψ^- e Ψ^+ dois W -operadores definidos respectivamente

pelas funções características

$$\psi^-(X) = \begin{cases} 1, & \text{se } X \in \mathcal{S}^- \\ 0, & \text{se } X \notin \mathcal{S}^- \end{cases}$$

e

$$\psi^+(X) = \begin{cases} 1, & \text{se } X \in \mathcal{S}^+ \\ 0, & \text{se } X \notin \mathcal{S}^+ \end{cases}$$

Então,

$$\Psi_{\text{opt},W} = (\Psi_o - \Psi^-) \vee \Psi^+ \quad (6.1)$$

onde $\Psi_o - \Psi^- = \Psi_o \wedge \nu \Psi^-$.

Conseqüentemente, um operador ótimo pode ser projetado a partir de Ψ_o projetando-se os dois componentes, Ψ^- e Ψ^+ . O operador Ψ_o será denominado um **operador a priori**. Um caso particular deste, onde $\Psi_o = \iota$, foi proposto em [54], sob o nome de “differencing filters”. A generalização proposta aqui aparece também em [50, 14, 51].

Na prática, porém, a função $\psi_{\text{opt},W}$ não é conhecida e, portanto também não são conhecidos os conjuntos \mathcal{S}^- e \mathcal{S}^+ . Para utilizar a equação 6.1 podemos estimar os conjuntos \mathcal{S}^- e \mathcal{S}^+ a partir de exemplos de treinamento, da seguinte forma :

$$\hat{\mathcal{S}}^+ = \{X \in \mathcal{P}(W) : \psi_o(X) = 0 \text{ e } \hat{P}(1|X) > 0.5\}$$

e

$$\hat{\mathcal{S}}^- = \{X \in \mathcal{P}(W) : \psi_o(X) = 1 \text{ e } \hat{P}(1|X) \leq 0.5\}.$$

Estes conjuntos estimados podem ser utilizados para gerar os estimadores $\hat{\psi}^-$ e $\hat{\psi}^+$ de ψ^- e ψ^+ , respectivamente. Podemos, então, escrever o estimador $\hat{\psi}_{\text{opt},W}$ de $\psi_{\text{opt},W}$:

$$\hat{\psi}_{\text{opt},W} = (\psi_o - \hat{\psi}^-) \cup \hat{\psi}^+. \quad (6.2)$$

Neste caso, para os pontos X não observados na amostra de treinamento, $\hat{\psi}_{\text{opt},W}(X) = \psi_o(X)$.

6.2 Confiança sobre um Operador a Priori

Na construção apresentada acima, diferentes operadores podem ser utilizados como o operador a priori. Se um operador a priori está próximo do operador ideal, então não é desejável que pequenos erros sejam permitidos com frequência na construção dos conjuntos \mathcal{S}^+ e \mathcal{S}^- . Para possibilitar algum tipo de controle sobre o operador a priori, introduzimos um *fator de confiança* $0 \leq \Theta \leq 0.5$ e redefinimos $\hat{\mathcal{S}}^-$ e $\hat{\mathcal{S}}^+$ da seguinte forma :

$$\hat{\mathcal{S}}^+ = \{X \in \mathcal{P}(W) : \psi_o(X) = 0 \text{ e } \hat{P}(1|X) > 0.5 + \Theta\}$$

e

$$\hat{\mathcal{S}}^- = \{X \in \mathcal{P}(W) : \psi_o(X) = 1 \text{ e } \hat{P}(1|X) \leq 0.5 - \Theta\}$$

Quanto maior o valor de Θ , maior a resistência de ψ_o no sentido de que uma forte evidência ao contrário será necessária para que o valor de ψ_o seja alterado. Por outro lado, se $\Theta = 0$, temos então a formulação apresentada anteriormente, e o valor de ψ_o é modificado perante qualquer evidência mínima.

Para que o modelo permita alguma flexibilidade (i.e., permita independentes valores de Θ para diferentes X), definimos $\Theta_X = \theta_X/N_X$, onde $\theta_X \geq 0$ é um valor que define o fator de confiança associado a X e N_X é o número de vezes que o padrão X foi observado na amostra de treinamento. Para que Θ_X não seja maior que 0.5 convencionamos :

$$\Theta_X = \begin{cases} 0.5, & \text{se } \frac{\theta_X}{N_X} > 0.5, \\ \frac{\theta_X}{N_X}, & \text{caso contrário.} \end{cases}$$

A seguinte tabela ilustra o valor de Θ_X para diferentes valores de θ_X e N_X . A primeira linha contém os diferentes valores θ_X e a primeira coluna diferentes valores N_X .

		Valor de θ_X				
		0.1	1	10	50	100
Num. Observ. (N_X)	1	0.1	0.5	0.5	0.5	0.5
	10	0.01	0.1	0.5	0.5	0.5
	25	0.004	0.04	0.4	0.5	0.5
	50	0.002	0.02	0.2	0.5	0.5
	100	0.001	0.01	0.1	0.5	0.5
	500	0.0002	0.002	0.02	0.1	0.2
	1000	0.0001	0.001	0.01	0.05	0.1

Usando estas notações, temos :

$$\hat{S}^+ = \{X \in \mathcal{P}(W) : \psi_o(X) = 0 \text{ e } \hat{P}(1|X) > 0.5 + \Theta_X\} \quad (6.3)$$

e

$$\hat{S}^- = \{X \in \mathcal{P}(W) : \psi_o(X) = 1 \text{ e } \hat{P}(1|X) \leq 0.5 - \Theta_X\} \quad (6.4)$$

Diferentes valores de θ_X possibilitam diferentes graus de confiança : quanto maior for θ_X , maior será a confiança, isto é maior será a evidência necessária para que o valor de ψ_o seja alterado. A formulação acima leva em consideração o número de vezes que cada elemento é observado, isto é, se X é observado 1000 vezes, é muito provável que a estimativa $\hat{P}(1|X)$ seja mais próximo da verdadeira do que a de um outro elemento Y observado apenas 10 vezes.

6.3 Procedimento Prático

6.3.1 Escolha de uma Janela

Até agora consideramos apenas as situações nas quais, tanto o operador que queremos projetar como o operador ψ_o , são operadores que dependem de uma mesma janela. O que acontece quando a janela associada a eles é diferente, i.e., quando a janela W utilizada no processo difere da janela W_o (possivelmente desconhecida) associada ao operador Ψ_o ?

a) $W_o \subseteq W$

Neste caso Ψ_o é também um W -operador e $\mathcal{K}_W(\Psi_o) = \{X \in \mathcal{P}(W) : \psi_o(X \cap W_o) = 1\}$. Logo, os conjuntos \mathcal{S}^- e \mathcal{S}^+ estão bem definidos.

b) $W \subset W_o$

Neste caso o operador Ψ_o , que depende de uma janela W_o , será aproximado a um operador $\Psi_{opt,W}$ que depende de uma janela W menor que W_o . O operador Ψ_o pode ser melhorado desta forma se, e só se, o desempenho de Ψ_o é pior que o de $\Psi_{opt,W}$ (i.e., se $MAE\langle\Psi_o\rangle > MAE\langle\Psi_{opt,W}\rangle$). Isto significa que esta técnica só deve ser aplicada sobre operadores a priori que dependem de janela muito grande se o seu desempenho não é bom. Além disso, janelas W muito menores que aquelas associadas ao operador Ψ_o não devem ser utilizadas para tentá-lo melhorar, pois nesses casos dificilmente teremos uma situação na qual $MAE\langle\Psi_o\rangle > MAE\langle\Psi_{opt,W}\rangle$.

6.3.2 Algoritmo

Descrevemos a seguir um procedimento prático para implementar o operador conforme a construção apresentada. Seja $\mathcal{T}_1 = \{(S_i, I_i) : i \in I\}$ um conjunto de pares de imagens de treinamento e seja \mathcal{X} a coleção de todos os exemplos distintos de $\mathcal{P}(W)$ coletados a partir das imagens S_i . Para todo $X \in \mathcal{X}$:

$$class(X) = \begin{cases} 0, & \text{se } \hat{P}(1|X) \leq 0.5 - \Theta_X \\ 1, & \text{se } \hat{P}(1|X) > 0.5 + \Theta_X \\ 2, & \text{caso contrário} \end{cases}$$

onde $\hat{P}(1|X)$ é estimado a partir de \mathcal{T}_1 .

Os elementos X tais que $class(X) = 0$ ou $class(X) = 1$ são os que satisfazem o fator de confiança θ_X . Portanto,

$$\hat{\mathcal{S}}^+ = \{X \in \mathcal{X}_1 : \psi_o(X) = 0 \text{ e } class(X) = 1\}$$

e

$$\hat{\mathcal{S}}^- = \{X \in \mathcal{X}_1 : \psi_o(X) = 1 \text{ e } class(X) = 0\}$$

Nem sempre a função característica ψ_o de Ψ_o é dada. Para calcular $\hat{\mathcal{S}}^+$ e $\hat{\mathcal{S}}^-$, precisamos saber o valor de ψ_o para todos os padrões $X \in \mathcal{X}$. O que podemos fazer é estimá-lo a partir de um conjunto de pares de imagens $\mathcal{T}_2 = \{(S_i, \Psi_o(S_i)) : i \in I\}$. Para todo $X \in \mathcal{X}$,

$$\hat{\psi}_o(X) = \begin{cases} 1, & \text{se } \hat{P}(1|X) > 0.5, \\ 0, & \text{se } \hat{P}(1|X) \leq 0.5 \end{cases}$$

onde $\hat{P}(1|X)$ é estimado a partir de \mathcal{T}_2 . Se $W_o \subseteq W$ então $\hat{P}(1|X) = 0$ ou $\hat{P}(1|X) = 1$; caso contrário, $\hat{P}(1|X) \in [0, 1]$. Os conjuntos $\hat{\mathcal{S}}^+$ e $\hat{\mathcal{S}}^-$ podem ser estimados a partir de $class$ e $\hat{\psi}_o$, da seguinte forma:

Para cada $X \in \mathcal{X}$,

- se $class(X) = 0$ e $\hat{\psi}_o(X) = 1$ então coloque X em \mathcal{S}^-
- se $class(X) = 1$ e $\hat{\psi}_o(X) = 0$ então coloque X em \mathcal{S}^+ .

Nos casos em que $W_o \subseteq W$, a coleção \mathcal{X} de exemplos que precisam ser examinados pode ser restrito àqueles observados nos pontos da diferença simétrica $\Psi_o(S_i) \Delta I_i$.

6.4 Resultados Experimentais

Exemplo 6.1 A técnica proposta foi utilizada para a extração das bordas externas de caracteres com ruídos pontuais, como a mostrada na figura 6.1. Foram utilizados dois operadores a priori:

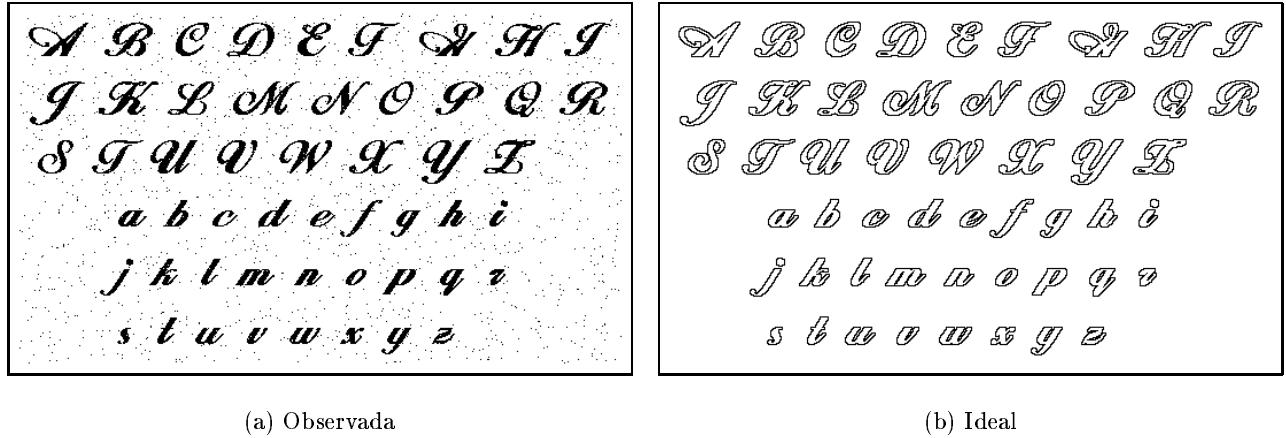


Figura 6.1: Imagem com ruídos pontuais e respectiva imagem ideal.

(1) Mediana pela janela $W_{3 \times 3}$ seguida de extração de borda externa (i.e., dilatação pelo elemento estruturante $B = 3 \times 3$ menos a identidade) e (2) um operador ótimo no espaço dos 3×3 -operadores, para projetar operadores no espaço de operadores com $W = W_{5 \times 5}$. Os resultados são comparados com o operador projetado pelo treinamento padrão. A tabela 6.1 mostra o resultado (tempo de treinamento e erro) dos operadores projetados usando a priori 1, a priori 2 e pelo treinamento padrão. Cada linha da tabela indica o número total de exemplos utilizados, a quantidade de diferentes padrões observados, e o tempo de treinamento e o respectivo erro, para cada um dos casos. Observe que o treinamento

		Priori 1		Priori 2		ISI	
Total	Distintos	Tempo	Erro	Tempo	Erro	Tempo	Erro
321912	14559	00:00:34	892	00:00:33	265	00:29:29	427
643824	20863	00:00:41	654	00:00:40	265	01:37:01	399
965736	26169	00:00:49	619	00:00:47	265	03:17:01	381
1287648	30722	00:00:58	580	00:00:55	265	05:15:45	373
1609560	35338	00:01:05	560	00:01:02	204	-	-
1931472	39074	00:01:13	535	00:01:09	201	-	-
2253384	42780	00:01:21	517	00:01:17	202	-	-
2575296	45983	00:01:29	504	00:01:24	199	-	-
2897208	49139	00:01:36	493	00:01:29	199	-	-

Tabela 6.1: Comparação entre uso de priori e não uso.

padrão foi interrompido em 1.3 milhão de exemplos pois o tempo de processamento excedeu 5 horas.

O erro da priori heurística (priori 1) é maior que o dos outros dois casos. O resultado da priori 2 é melhor que o do treinamento padrão.

A figura 6.2 mostra parte de uma imagem de teste, a correspondente imagem ideal, e os resultados para cada um dos casos acima.

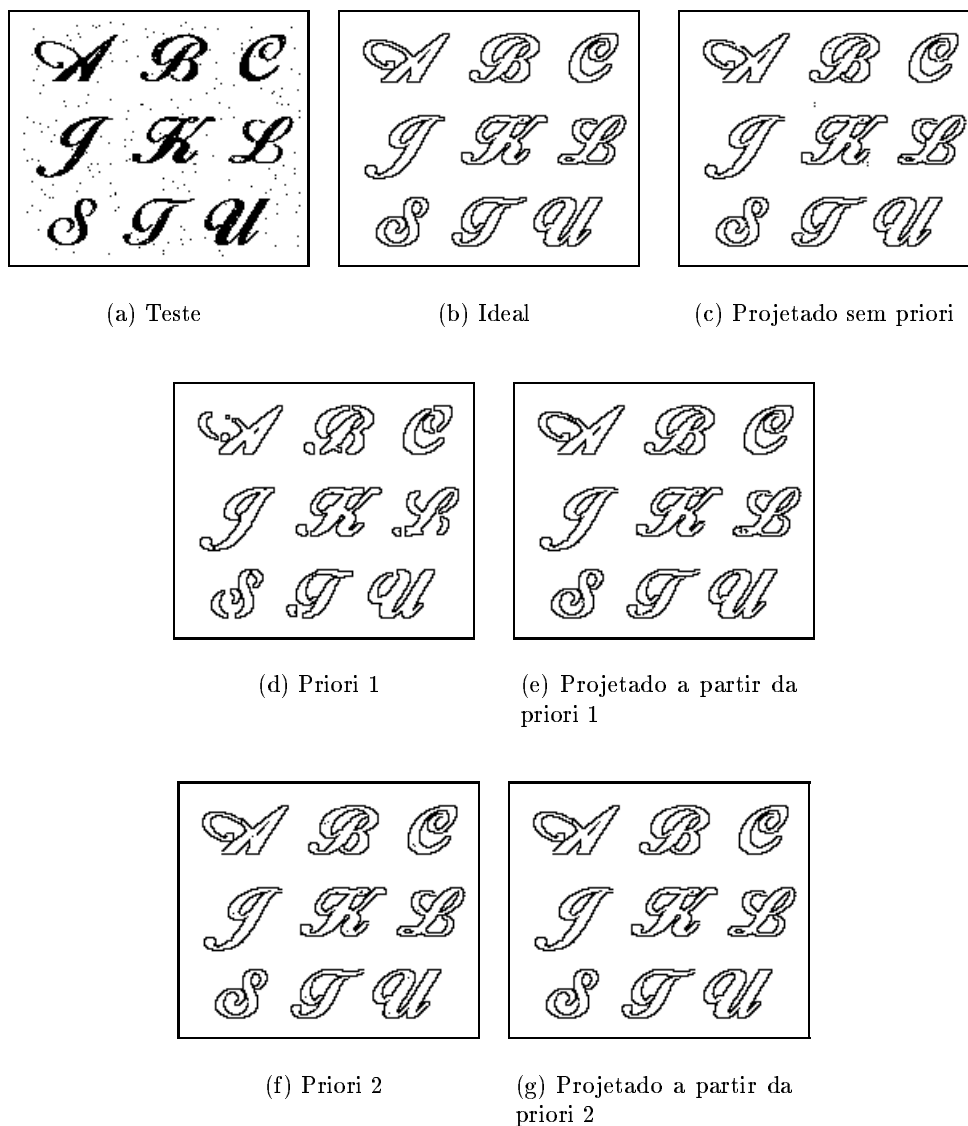


Figura 6.2: Extração de borda de uma imagem ruidosa.

Exemplo 6.2 A técnica proposta foi aplicada também para extração de segmentos de retas com inclinação entre 10 e 80 graus, de imagens com segmentos de retas com vários ângulos de inclinação. Novamente foram considerados dois operadores a priori e o treinamento padrão. A priori 1 é uma abertura por um segmento de reta de comprimento 5 e inclinação de 35 graus. A priori 2 é um operador treinado pelo método padrão para a janela $W_{5 \times 5}$. A janela utilizada no processo é a janela $W_{7 \times 7}$. A tabela 6.2 e a figura 6.3 mostram os resultados obtidos. Note que a janela associada ao operador priori 1 é maior que $W_{7 \times 7}$, mas como seu desempenho não é bom, foi possível melhorá-lo com a janela $W_{7 \times 7}$. Foi utilizado o fator de inércia $\theta_X = 0.2$. Neste caso o resultado do treinamento

Total	Distintos	Priori 1		Priori 2		ISI	
		Tempo	Erro	Tempo	Erro	Tempo	Erro
253000	23050	00:01:07	1455	00:01:07	250	00:01:39	45
506000	35690	00:01:18	538	00:01:17	249	00:04:18	28
759000	45369	00:01:29	480	00:01:28	248	00:07:14	26
1012000	53763	00:01:39	437	00:01:37	246	00:09:20	26
1265000	61291	00:01:50	370	00:01:48	205	-	-
1518000	67089	00:01:58	373	00:01:57	50	-	-
1771000	72856	00:02:09	364	00:02:05	46	-	-
2024000	77947	00:02:20	353	00:02:16	46	-	-
2277000	82796	00:02:27	353	00:02:25	45	-	-

Tabela 6.2: Comparação entre uso de priori e não uso.

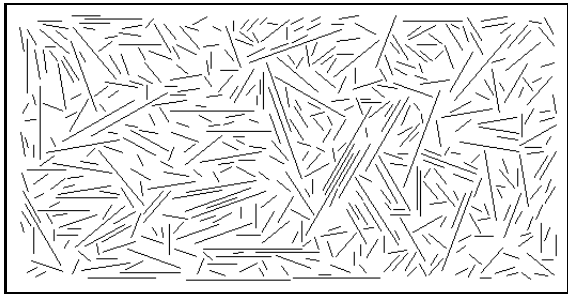
padrão é melhor que os operadores obtidos aplicando-se o método proposto para ambos os operadores a priori.

6.5 Comentários

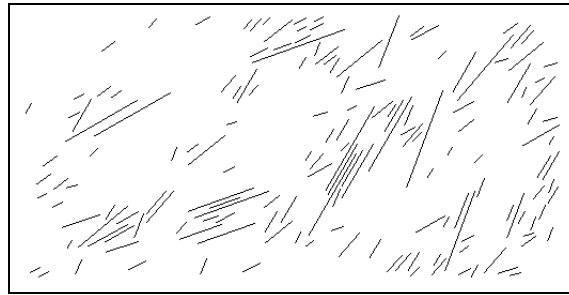
Se o operador a priori possui desempenho razoavelmente aceitável, então qualquer melhora deste operador pode ser obtido a partir de poucos chaveamentos, isto é, a representação de ψ^- e ψ^+ envolve poucos elementos. Nessas situações, a utilização da técnica proposta possibilita considerarmos grande quantidade de dados de treinamento e representação relativamente econômica.

Uma vez que o grande gargalo nos treinamentos padrões é o tamanho da janela, esta técnica pode ser utilizada para melhorar o desempenho de W-operadores obtidos por algum método de treinamento padrão, para janelas pequenas. Isto é, os pontos para os quais o operador projetado atribui valor incorreto podem ser corrigidos utilizando-se o método descrito com uma janela maior. Este processo pode ser iterado sucessivamente com janelas cada vez maiores.

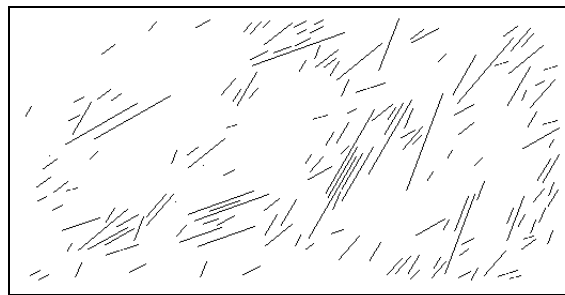
Os experimentos realizados mostram que a escolha heurística de um operador a priori não é uma tarefa fácil. Em ambos os casos a utilização do operador subótimo, projetado pelo treinamento padrão sobre uma janela menor, resultou em operadores mais precisos do que a utilização de um operador heurístico.



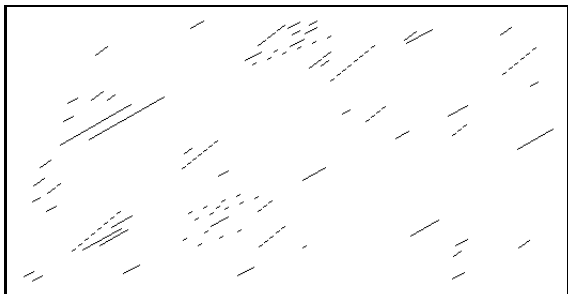
(a) Teste



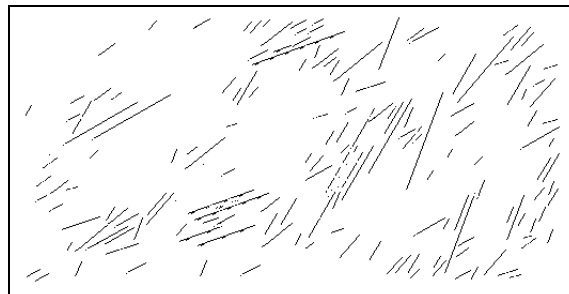
(b) Ideal



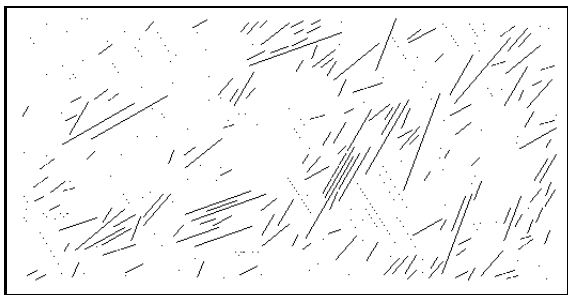
(c) Projetado sem priori



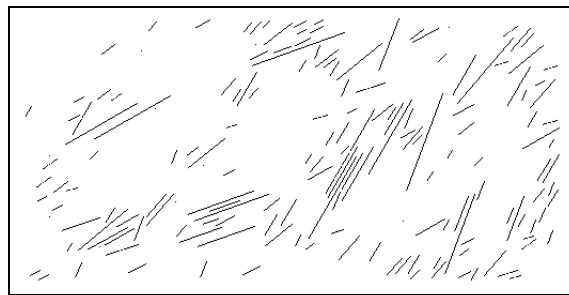
(d) Priori 1



(e) Projetado a partir da priori 1



(f) Priori 2



(g) Projetado a partir da priori 2

Figura 6.3: Reconhecimento de segmento de linhas a certo grau de inclinação.

Capítulo 7

Projeto de Operadores Crescentes Ótimos

O interesse em projetar operadores crescentes tem motivação em alguns fatos :

- operadores crescentes podem ser expressos como união de erosões e portanto sua implementação é fácil ;
- a interpretação do efeito de um operador crescente através da análise dos elementos estruturantes da sua base é relativamente simples ;
- operadores crescentes constituem uma classe largamente utilizada na prática (envolvem as aberturas, fechamentos e os filtros em geral).
- apesar da maior parte dos trabalhos existentes sobre projeto de operadores ser restrito à classe de operadores crescentes, não se conhece um algoritmo eficiente para projetá-los.

Sob o ponto de vista de projeto de operadores, restringir o espaço de operadores ao espaço de operadores crescentes implica duas consequências antagônicas: por um lado, reduz-se o tamanho do espaço de busca, o que é uma característica desejável; mas por outro lado, a restrição pode resultar em subotimalidade. Esta segunda consequência é, a primeira vista, uma característica indesejável. No entanto, na prática, a precisão do operador projetado em um espaço reduzido pode ser maior do que daquele projetado sobre o espaço todo (ver capítulo 3 para maiores detalhes). Esta é uma das razões que justificam o projeto de operadores com restrições. Além disso, operadores com restrição algébrica podem possuir estruturas de representação interessantes que podem ser exploradas pelos algoritmos de busca.

Neste capítulo apresentamos um algoritmo eficiente para projeto de operadores crescentes, utilizando o método de chaveamentos (uma idéia originalmente proposta em [119]). Analisamos também a estimação de custos de chaveamento no contexto Bayesiano.

Apresentamos, a seguir, uma formulação do problema de projeto de operadores crescentes estatisticamente ótimos e uma breve revisão das técnicas para projeto de operadores crescentes encontradas na literatura consultada.

7.1 Descrição do Problema e Algumas Técnicas de Solução

Um operador $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ é *crescente* sse $\Psi(X) \subseteq \Psi(Y)$ sempre que $X \subseteq Y$, $\forall X, Y \in \mathcal{P}(E)$. Se Ψ é um W -operador caracterizado pela função Booleana ψ , esta condição é equivalente a dizer que $\psi(X) \leq \psi(Y)$, $\forall X, Y$ sempre que $X \subseteq Y$.

Qualquer operador crescente e i.t. pode ser expresso como o supremo de um conjunto de erosões [118] (veja também a seção 2.2.3). Além disso, a função Booleana característica ψ pode ser expressa como soma de produtos onde nenhuma variável aparece complementada. Outra importante propriedade é que estes termos (os produtos) correspondem aos elementos minimais de $\psi\langle 1 \rangle$. Mais ainda, se um dado elemento $X \in \mathcal{P}(W)$ pertence a $\psi\langle 1 \rangle$, então todos os elementos $Y \in \mathcal{P}(W)$ tais que $Y \geq X$ também pertencem a $\psi\langle 1 \rangle$. Portanto, projetar W -operadores crescentes equivale a escolher um conjunto de elementos que correspondem aos elementos minimais do núcleo do operador. Isto equivale, por sua vez, a escolher o conjunto de produtos que fazem parte da expressão Booleana ou equivalentemente, o conjunto de erosões que definem a decomposição de Matheron [118].

Utilizamos uma notação simplificada para a base de um operador crescente : em vez de $\mathcal{B}(\Psi)$ (uma coleção de intervalos maximais contidos em $\mathcal{P}(W)$) denotamos a base de operadores crescentes por $Bas[\Psi]$ (uma coleção de elementos minimais de $\mathcal{K}_W(\Psi)$, mais precisamente, todas as extremidades inferiores dos intervalos em $\mathcal{B}(\Psi)$).

No contexto visto no capítulo 3, a subclasse de operadores Ψ_c considerada aqui é a classe de W -operadores crescentes. Estamos interessados em encontrar um operador MAE-ótimo em Ψ_c . O tamanho deste espaço não é conhecido para um n genérico. Este é um problema em aberto, conhecido como *Problema de Dedekind* e apenas alguns limitantes são conhecidos [101].

Um dos primeiros trabalhos sobre projeto de operadores crescentes estatisticamente ótimos coincide com os primeiros trabalhos sobre projeto de operadores morfológicos no contexto de estimação estatística [39, 40]. Neles, uma formulação baseada na representação de Matheron, a saber a representação de quaisquer operadores crescentes como uma união de erosões, é apresentada. O problema fundamental consiste em escolher uma coleção de elementos estruturantes cuja união de erosões resulte em um operador estatisticamente ótimo no sentido visto no capítulo 3. A solução apresentada é o método da força-bruta.

Uma primeira tentativa de sistematização para resolver o problema é baseado em um teorema, denominado *Teorema do MAE* [111, 109]. A idéia básica do teorema é uma fórmula recursiva que permite o cálculo do MAE de uma união de m -erosões a partir do MAE da união de $(m - 1)$ -erosões e dos MAEs individuais. Inicialmente, o MAE de uma erosão pode ser facilmente calculado a partir das probabilidades conjuntas de \mathbf{X} e \mathbf{y} :

$$MAE\langle E_B \rangle = E[|\mathbf{y} - E_B(\mathbf{X})|] = \sum_{\{\mathbf{X} : \mathbf{X} \subseteq B\}} P(\mathbf{X}, 0) + \sum_{\{\mathbf{X} : \mathbf{X} \not\subseteq B\}} P(\mathbf{X}, 1) \quad (7.1)$$

A partir dos MAEs individuais, o MAE de qualquer operador crescente pode ser calculado através de uma fórmula recursiva. Isto é, se Ψ é um operador crescente cuja base é dada por $Bas[\Psi] = \{B_1, B_2, \dots, B_m\}$, então seu MAE pode ser expresso como :

$$MAE\langle \Psi \rangle = MAE\langle \Psi' \rangle + MAE\langle E_{B_m} \rangle - MAE\langle \Phi \rangle \quad (7.2)$$

onde Ψ' e Φ são caracterizadas respectivamente pelas bases dadas por :

$$Bas\langle \Psi' \rangle = \{B_1, B_2, \dots, B_{m-1}\}$$

e

$$Bas\langle\Phi\rangle = \{B_1 \cup B_m, B_2 \cup B_m, \dots, B_{m-1} \cup B_m\}$$

Esta fórmula recursiva pode ser implementada, por exemplo, utilizando técnicas de programação dinâmica (veja, por exemplo [31], capítulo 16) para evitar repetição de cálculos. No entanto, este teorema não reduz o número de combinações de elementos estruturantes que devem ser analisados, o qual é de ordem exponencial.

Abordagens alternativas que tentam evitar a busca no espaço de todas as combinações possíveis começaram a ser propostas. Em [110, 109], três abordagens que impõem restrições sobre o espaço de busca são propostas :

1. Fixar o número máximo de elementos estruturantes na base do operador a ser projetado. Com isso, ao invés de todas as possíveis combinações de elementos estruturantes, apenas combinações que envolvem um número máximo (fixo) de elementos estruturantes são analisadas.
2. Em vez de considerar todos os subconjuntos de uma janela W como sendo os possíveis elementos estruturantes, consideram-se apenas subconjuntos de uma subjanela $W' \subset W$.
3. Reduzir o conjunto de elementos estruturantes segundo algum critério. Uma possibilidade é uma *biblioteca especializada* contendo os elementos estruturantes que possuem características adequadas ao problema que se deseja resolver. Este tipo de conhecimento pode vir, por exemplo, de experiências acumuladas com problemas semelhantes. Outra possibilidade é a *biblioteca de primeira-ordem*, ou seja, o conjunto formado por elementos estruturantes que individualmente apresentam os menores MAEs.

Na prática é comum o emprego simultâneo das restrições em 1 e 3. Estas abordagens fornecem resultados sub-ótimos.

O conjunto de operadores crescentes forma um reticulado completo, e este pode ser representado por um grafo. Por exemplo, para uma janela de 3 pontos, o grafo de todos os operadores crescentes está mostrado na figura 7.1. Neste grafo, a diferença entre o núcleo de dois operadores vizinhos

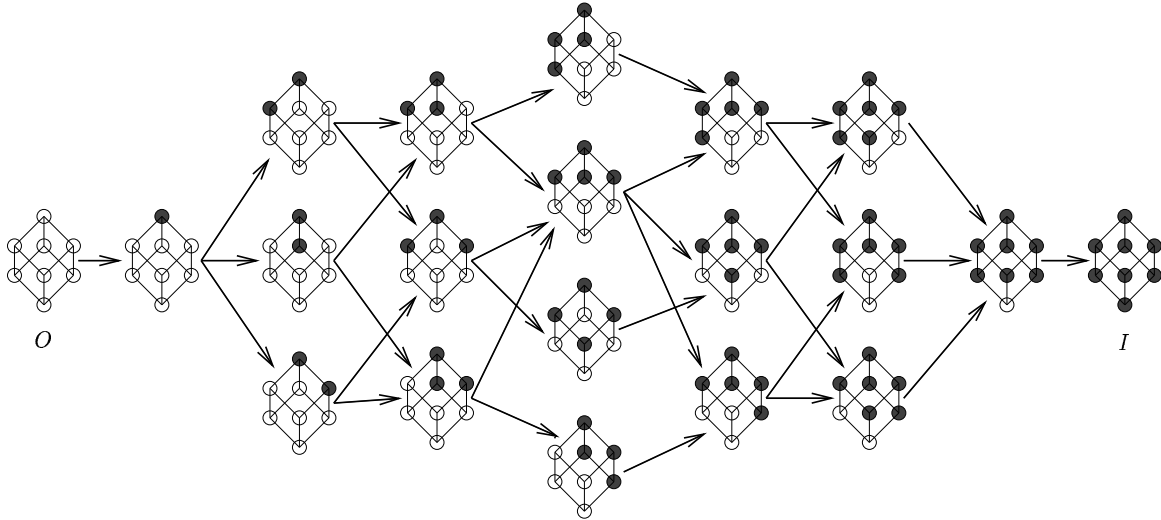


Figura 7.1: Grafo representando o espaço de operadores crescentes.

quaisquer é de apenas um elemento. O MAE de um operador crescente Ψ_b obtido acrescentando-se apenas um elemento X ao núcleo de um operador crescente Ψ_a pode ser expresso da seguinte forma :

$$MAE\langle\Psi_b\rangle = MAE\langle\Psi_a\rangle - P(X, 1) + P(X, 0)$$

Observe, no entanto, que este elemento X não pode ser arbitrário, ele deve ser um elemento tal que Ψ_b também é crescente. Um algoritmo que realiza uma busca percorrendo este grafo a partir do nó que corresponde ao operador cujo núcleo é $\mathcal{K}(\Psi_a) = \{W\}$ é proposto em [69]. A partir de um dado nó Ψ_a , examinam-se todos os nós vizinhos Ψ_b e percorre-se o grafo pela aresta que conduz ao vizinho Ψ_b (com um elemento a mais no núcleo) que mais reduz o MAE em relação a Ψ_a . Algumas características interessantes propostas nesse artigo são a propriedade ‘greedy’ de passeio, isto é, dado um conjunto de elementos que podem ser adicionados ao núcleo de Ψ_a de forma que Ψ_b é crescente, a propriedade greedy garante que a escolha daquele elemento X que mais diminui o MAE, se existir algum, é o caminho que leva ao operador com MAE mínimo no grafo todo. Se não existirem elementos que diminuam o MAE, então os candidatos devem ser empilhados e processados um a um. A outra característica do algoritmo é a condição de “bound”. Isto é, se a partir de um vértice, todos os vértices que podem ser atingidos possuem apenas elementos que aumentam o MAE, então não há necessidade de se prosseguir o percorrimto a partir daquele vértice (neste caso, supõe-se que os vértices que fazem parte do caminho a frente deste vértice já haviam sido percorridos a partir de outro vértice processado anteriormente). Embora esta represente uma técnica mais elaborada que evita cálculos desnecessários, é possível encontrarmos casos para os quais o algoritmo percorre todo o espaço de operadores crescentes.

É possível também formular o problema como um problema de programação linear (PL) [127]. Esta abordagem foi proposta no contexto de projeto de “stack filters” [33] mas foi abandonada devido a sua complexidade [108, 170].

Outra abordagem é sugerida em [119]. Trata-se da abordagem baseada em chaveamentos. Esse artigo apresenta a formulação do problema no contexto de chaveamentos, a noção de conjuntos violadores (referidos como conjuntos de inversão), custos de chaveamento e conjuntos de chaveamento ótimo em relação a propriedade crescente. Porém, nenhum algoritmo para escolha do conjunto de chaveamento ótimo é dado nesse artigo.

A generalização destes conceitos para restrições algébricas quaisquer é apresentada na seção a seguir. O algoritmo que propomos neste capítulo, utilizando estes conceitos, é justamente um algoritmo para calcular um conjunto de chaveamento ótimo em relação a propriedade crescente.

7.2 O Método de Chaveamentos

Seja Ψ um W -operador. Ele pode ser modificado chaveando-se seu valor para algumas de suas entradas (i.e., mudando o valor de $\psi(X)$ de 0 para 1 ou de 1 para 0, para alguns elementos $X \in \mathcal{P}(W)$). Um chaveamento sobre Ψ pode ser visto como uma modificação no núcleo de Ψ : um elemento é removido de $\psi\langle 1 \rangle$ e agregado ao conjunto $\psi\langle 0 \rangle$, ou vice-versa. Uma sucessão de t chaveamentos sobre Ψ define uma sequência de operadores $\psi_0 = \psi, \psi_1, \dots, \psi_t$. Estes chaveamentos podem ser efetuados de modo que o operador resultante, ψ_t , esteja contido em algum subespaço Ψ_C . Os elementos para os quais o valor de Ψ é chaveado é denominado *conjunto de chaveamento*.

Na seção 3.3.3, vimos que um operador subótimo em Ψ_W com relação a um subespaço Ψ_C é aquele que possui o menor aumento de erro em relação a $\Psi_{\text{opt}, W}$. Portanto, um operador ótimo em Ψ_C

pode ser projetado aplicando-se chaveamentos sobre $\Psi_{\text{opt},W}$ de forma que o operador resultante esteja em Ψ_C e seu aumento de erro seja mínimo. Esta idéia está mostrada na figura 7.2. Se denotamos

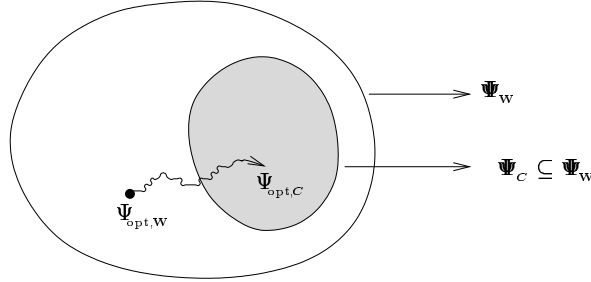


Figura 7.2: O método de chaveamentos.

o conjunto de chaveamentos por \mathcal{A} , o operador resultante do chaveamento sobre $\Psi_{\text{opt},W}$ é denotado $\Psi_{\text{opt},W,\mathcal{A}}$, e sua função característica é dada por :

$$\psi_{\text{opt},W,\mathcal{A}}(X) = \begin{cases} \psi_{\text{opt},W}(X), & \text{se } X \notin \mathcal{A}, \\ 1 - \psi_{\text{opt},W}(X), & \text{se } X \in \mathcal{A}. \end{cases} \quad (7.3)$$

O aumento de erro de $\psi_{\text{opt},W,\mathcal{A}}$ em relação a $\psi_{\text{opt},W}$ pode ser escrito em função de \mathcal{A} :

$$\Delta(\psi_{\text{opt},W,\mathcal{A}}, \psi_{\text{opt},W}) = \sum_{X \in \mathcal{A}} c(X) \quad (7.4)$$

onde $c(X)$ é o aumento de erro devido a X , denominado *custo de chaveamento* de X . No caso específico do risco MAE, $c(X) = |2P(1|X) - 1|P(X)$. Um *conjunto de chaveamento ótimo* é um conjunto de chaveamento para o qual o somatório da Eq. 7.4 é mínimo.

O método de chaveamentos é útil para projetar operadores com restrições algébricas. O procedimento para obter um operador ótimo em Ψ_C a partir de chaveamentos sobre $\psi_{\text{opt},W}$ pode ser esquematizado como segue :

1. Estimar $\hat{\psi}_{\text{opt},W}$.
2. Calcular o custo de chaveamento $c(X)$ para cada $X \in \mathcal{P}(W)$. Se um particular elemento X não foi observado nas amostras de treinamento, então podemos considerar $P(X) = 0$.
3. Escolher um conjunto de chaveamento ótimo \mathcal{A} .
4. Calcular $\hat{\psi}_{\text{opt},W,\mathcal{A}}$, o operador ótimo em Ψ_C .

Conjunto Violador

Seja Ψ um W -operador e seja $\Psi_C \subset \Psi_W$. O *conjunto violador* de Ψ com respeito a Ψ_C , denotado \mathcal{Q}_ψ , é o conjunto de elementos de $\mathcal{P}(W)$ que violam a condição que deve ser satisfeita por Ψ .

Por exemplo, o conjunto violador de Ψ com respeito ao subespaço dos operadores anti-extensivos é o conjunto $\mathcal{Q}_\psi = \{X \in \psi\langle 1 \rangle : o \notin X\}$; e com respeito ao subespaço dos operadores crescentes é conjunto de todos os elementos de $\psi\langle 1 \rangle$ que são menores do que pelo menos um elemento de $\psi\langle 0 \rangle$, e por todos os elementos de $\psi\langle 0 \rangle$ que são maiores do que pelo menos um elemento de $\psi\langle 1 \rangle$.

Um operador Ψ é um elemento de Ψ_C se, e somente se, o conjunto violador de Ψ com respeito a Ψ_C é vazio. Chaveamentos podem ser efetuados de forma que o conjunto violador do operador resultante seja menor em relação ao conjunto violador do operador inicial.

Em alguns casos, o conjunto de chaveamento ótimo pode ser determinado facilmente; é o próprio conjunto violador. Por exemplo, para se obter um operador anti-extensivo, basta que todos os elementos de $\psi_{\text{opt},W}\langle 1 \rangle$, que não contém a origem, sejam chaveados de 1 para 0. Neste exemplo, o conjunto de chaveamentos é o conjunto $\mathcal{A} = \{X \in \psi_{\text{opt},W}\langle 1 \rangle : o \notin X\}$. Em outros casos, como no caso em que a restrição é a propriedade crescente ou idempotente, a determinação de um conjunto de chaveamento ótimo pode ser extremamente complexa. Neste capítulo apresentamos um novo algoritmo, baseado no método de chaveamentos, para projetar operadores crescentes.

7.3 Um Novo Algoritmo Baseado no Método de Chaveamentos

Escolher um conjunto de chaveamento para gerar um operador crescente ótimo não é trivial. Por exemplo, se existe um elemento Y em $\psi\langle 0 \rangle$ maior do que um elemento X em $\psi\langle 1 \rangle$, então existem duas possibilidades quando estes dois elementos são analisados isoladamente: ou chaveia-se $\psi(X)$ de 1 para 0 ou chaveia-se $\psi(Y)$ de 0 para 1. Entretanto, ao se chavear $\psi(X)$ para 0, todos os elementos de $\psi\langle 1 \rangle$ que são menores do que X também precisam ser chaveados para 0; por outro lado, ao se manter $\psi(X) = 1$, todos os elementos de $\psi\langle 0 \rangle$ que são maiores que X precisam ser chaveados para 1. Portanto, a decisão de chavear X ou Y pode afetar as decisões sobre outros elementos no conjunto violador e, estes por sua vez, podem afetar sucessivamente a decisão sobre outros elementos [119].

A figura 7.3(a) mostra os conjuntos $\psi\langle 1 \rangle$ e $\psi\langle 0 \rangle$. O conjunto violador é mostrado na Fig. 7.3(b). A Fig. 7.3(c) mostra um exemplo de um conjunto de chaveamento (os elementos indicados por dupla borda), e a Fig. 7.3(d) mostra o operador crescente resultante após os chaveamentos. A Fig. 7.4(a) mostra um subconjunto do mesmo conjunto violador que não é, entretanto, um conjunto de chaveamento (válido). O operador resultante após os chaveamentos, que não é crescente, é mostrado na Fig. 7.4(b).

7.3.1 Reestruturação do Problema como um Problema de Partição

A partir daqui, vamos assumir implicitamente que um operador ótimo ψ_{opt} é dado. Qualquer menção a conjuntos violadores, conjuntos de chaveamentos e custos de chaveamentos devem ser entendidos como referentes a ψ_{opt} .

Dado um conjunto violador \mathcal{Q} , um par ordenado (L, U) tal que $L \cup U = \mathcal{Q}$, $L \cap U = \emptyset$, e tal que nenhum elemento de L é maior do que um elemento de U (ou equivalentemente, nenhum elemento de U é menor do que um elemento de L) é denominado uma *partição válida* de \mathcal{Q} . Os conjuntos L e U são denominados respectivamente, *parte inferior* e *parte superior* da partição.

Proposição 7.1 *Seja \mathcal{Q} um conjunto violador, $\mathcal{A} \subseteq \mathcal{Q}$ um conjunto de chaveamento, e (L, U) uma partição válida de \mathcal{Q} . Considere os conjuntos $L_{\mathcal{A}} = (\mathcal{Q}\langle 0 \rangle \setminus \mathcal{A}\langle 0 \rangle) \cup \mathcal{A}\langle 1 \rangle$, $U_{\mathcal{A}} = (\mathcal{Q}\langle 1 \rangle \setminus \mathcal{A}\langle 1 \rangle) \cup \mathcal{A}\langle 0 \rangle$, e $\mathcal{A}_{(L,U)} = (L \cap \mathcal{Q}\langle 1 \rangle) \cup (U \cap \mathcal{Q}\langle 0 \rangle)$. Então,*

- (a) $(L_{\mathcal{A}}, U_{\mathcal{A}})$ é uma partição válida de \mathcal{Q} ,
- (b) $\mathcal{A}_{(L,U)}$ é um conjunto de chaveamento,

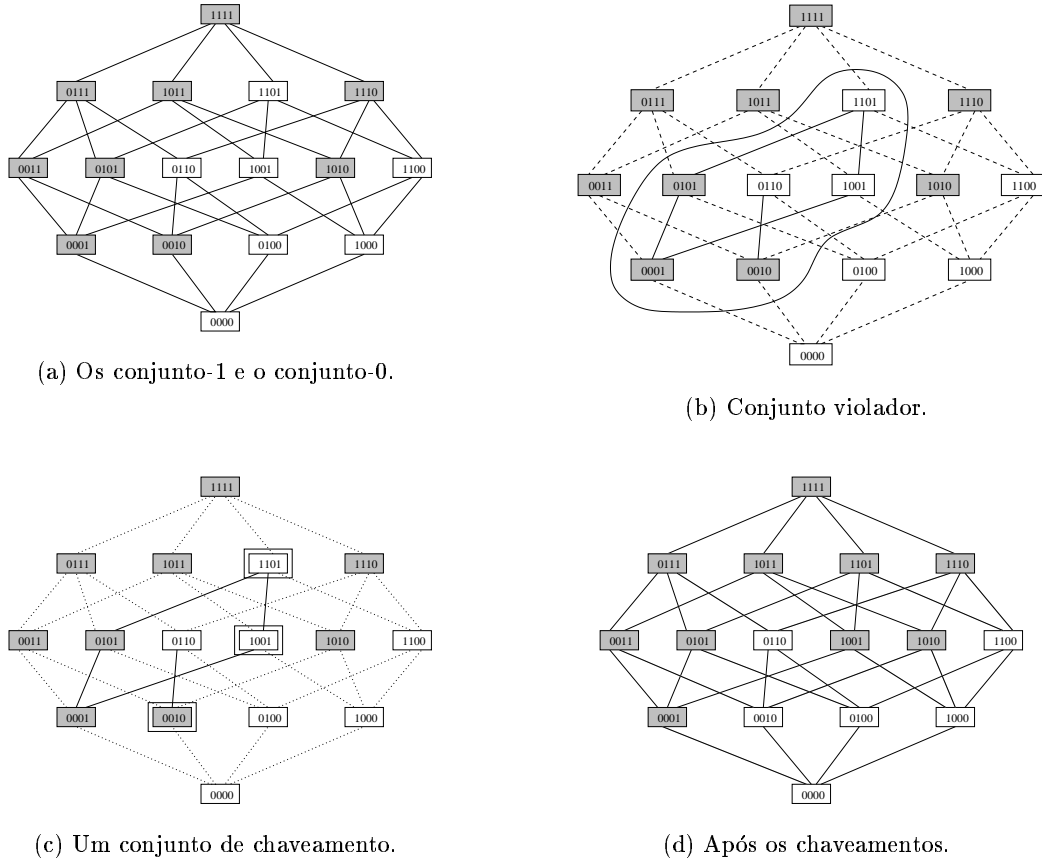


Figura 7.3: Um operador crescente produzido por chaveamentos.

$$(c) \mathcal{A}_{(L_{\mathcal{A}}, U_{\mathcal{A}})} = \mathcal{A} \text{ e}$$

$$(d) (L_{\mathcal{A}_{(L, U)}}, U_{\mathcal{A}_{(L, U)}}) = (L, U).$$

Dem.: (a) Provamos que $L_{\mathcal{A}} \cup U_{\mathcal{A}} = \mathcal{Q}$, $L_{\mathcal{A}} \cap U_{\mathcal{A}} = \emptyset$, e que nenhum elemento de $U_{\mathcal{A}}$ é menor que qualquer elemento de $L_{\mathcal{A}}$.

$$\begin{aligned} L_{\mathcal{A}} \cup U_{\mathcal{A}} &= [(\mathcal{Q}\langle 0 \rangle \setminus \mathcal{A}\langle 0 \rangle) \cup \mathcal{A}\langle 1 \rangle] \cup [(\mathcal{Q}\langle 1 \rangle \setminus \mathcal{A}\langle 1 \rangle) \cup \mathcal{A}\langle 0 \rangle] \\ &= [(\mathcal{Q}\langle 0 \rangle \setminus \mathcal{A}\langle 0 \rangle) \cup \mathcal{A}\langle 0 \rangle] \cup [(\mathcal{Q}\langle 1 \rangle \setminus \mathcal{A}\langle 1 \rangle) \cup \mathcal{A}\langle 1 \rangle] \\ &= \mathcal{Q}\langle 0 \rangle \cup \mathcal{Q}\langle 1 \rangle = \mathcal{Q} \end{aligned}$$

$$\begin{aligned} L_{\mathcal{A}} \cap U_{\mathcal{A}} &= [(\mathcal{Q}\langle 0 \rangle \setminus \mathcal{A}\langle 0 \rangle) \cup \mathcal{A}\langle 1 \rangle] \cap [(\mathcal{Q}\langle 1 \rangle \setminus \mathcal{A}\langle 1 \rangle) \cup \mathcal{A}\langle 0 \rangle] \\ &= [(\mathcal{Q}\langle 0 \rangle \cap \mathcal{A}\langle 0 \rangle^c) \cup \mathcal{A}\langle 1 \rangle] \cap [(\mathcal{Q}\langle 1 \rangle \cap \mathcal{A}\langle 1 \rangle^c) \cup \mathcal{A}\langle 0 \rangle] \\ &= [(\mathcal{Q}\langle 0 \rangle \cap \mathcal{A}\langle 0 \rangle^c) \cap [(\mathcal{Q}\langle 1 \rangle \cap \mathcal{A}\langle 1 \rangle^c) \cup \mathcal{A}\langle 0 \rangle]] \\ &\quad \cup [\mathcal{A}\langle 1 \rangle \cap [(\mathcal{Q}\langle 1 \rangle \cap \mathcal{A}\langle 1 \rangle^c) \cup \mathcal{A}\langle 0 \rangle]] \\ &= \emptyset \cup \emptyset = \emptyset \end{aligned}$$

Suponha por absurdo que existe $Y \in L_{\mathcal{A}}$ tal que $\exists X \in U_{\mathcal{A}}$, $X < Y$. Como $L_{\mathcal{A}} \subseteq (\psi\langle 0 \rangle \setminus \mathcal{A}\langle 0 \rangle) \cup \mathcal{A}\langle 1 \rangle = \psi_{\mathcal{A}}\langle 1 \rangle$ e $U_{\mathcal{A}} \subseteq (\psi\langle 1 \rangle \setminus \mathcal{A}\langle 1 \rangle) \cup \mathcal{A}\langle 0 \rangle = \psi_{\mathcal{A}}\langle 1 \rangle$, o operador $\psi_{\mathcal{A}}$ (segundo a definição dada pela

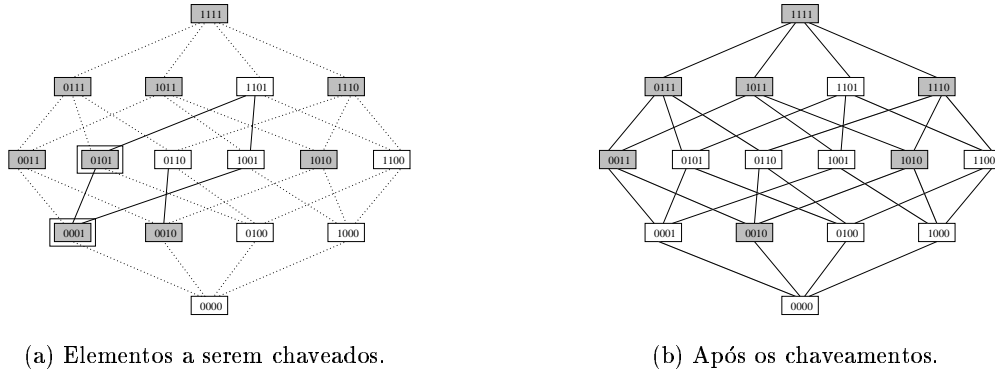


Figura 7.4: Chaveamentos que resultam em um operador não-crescente.

Eq. 7.3) é tal que $\psi_{\mathcal{A}}(X) = 1$ e $\psi_{\mathcal{A}}(Y) = 0$, o que é um absurdo porque $\psi_{\mathcal{A}}$ deveria ser crescente (pois \mathcal{A} é um conjunto de chaveamento).

(b) Por definição, $\mathcal{A}_{(L,U)}$ é um conjunto de chaveamento se e somente se $\psi_{\mathcal{A}_{(L,U)}}$ é um operador crescente. Suponha por absurdo que $\mathcal{A}_{(L,U)}$ não é um conjunto de chaveamento. Então, $\psi_{\mathcal{A}_{(L,U)}}$ não é crescente, i.e., existe X e Y , $X < Y$, tal que $X \in \psi_{\mathcal{A}_{(L,U)}}\langle 1 \rangle$ e $Y \in \psi_{\mathcal{A}_{(L,U)}}\langle 0 \rangle$, onde

$$\begin{aligned}\psi_{\mathcal{A}_{(L,U)}}\langle 1 \rangle &= [\psi\langle 1 \rangle \setminus (L \cap \mathcal{Q}\langle 1 \rangle)] \cup (U \cap \mathcal{Q}\langle 0 \rangle) = [\psi\langle 1 \rangle \setminus L\langle 1 \rangle] \cup U\langle 0 \rangle \\ \psi_{\mathcal{A}_{(L,U)}}\langle 0 \rangle &= [\psi\langle 0 \rangle \setminus (U \cap \mathcal{Q}\langle 0 \rangle)] \cup (L \cap \mathcal{Q}\langle 1 \rangle) = [\psi\langle 0 \rangle \setminus U\langle 0 \rangle] \cup L\langle 1 \rangle\end{aligned}$$

Portanto, por um lado, ou $X \in \psi\langle 1 \rangle \setminus L\langle 1 \rangle$ ou $X \in U\langle 0 \rangle$ e, por outro lado, ou $Y \in \psi\langle 0 \rangle \setminus U\langle 0 \rangle$ ou $Y \in L\langle 1 \rangle$.

Mostraremos que todas as possibilidades implicam $X \in U$ e $Y \in L$, o que é um absurdo porque (L,U) é uma partição válida de \mathcal{Q} .

(i) O caso $X \in U\langle 0 \rangle$ e $Y \in L\langle 1 \rangle$ é trivial.

(ii) se $X \in U\langle 0 \rangle$ e $Y \in \psi\langle 0 \rangle \setminus U\langle 0 \rangle$, então $X \in \mathcal{Q}$ (pois $U\langle 0 \rangle \subseteq \mathcal{Q}$), i.e., existe $Z \in \mathcal{P}(W)$, $Z < X$, tal que $\psi(Z) = 0$ e $\psi(Z) = 1$. Portanto, $Y \in \mathcal{Q}$ (pois $Y > X > Z$ e $\psi(Y) = 0$), i.e., $Y \in [\psi\langle 0 \rangle \setminus U\langle 0 \rangle] \cap \mathcal{Q} = L\langle 0 \rangle$.

(iii) se $X \in \psi\langle 1 \rangle \setminus L\langle 1 \rangle$ e $Y \in L\langle 1 \rangle$, então $Y \in \mathcal{Q}$, $X \in \mathcal{Q}$ também, e portanto $X \in [\psi\langle 1 \rangle \setminus L\langle 1 \rangle] \cap \mathcal{Q} = U\langle 1 \rangle$, pelo mesmo raciocínio em (ii).

(iv) se $X \in \psi\langle 1 \rangle \setminus L\langle 1 \rangle$ e $Y \in \psi\langle 0 \rangle \setminus U\langle 0 \rangle$, então $\psi(X) = 1$ e $\psi(Y) = 0$, e portanto $X, Y \in \mathcal{Q}$ (pois $Y > X$), o que significa que $X \in [\psi\langle 1 \rangle \setminus L\langle 1 \rangle] \cap \mathcal{Q} = U\langle 1 \rangle$ e $Y \in [\psi\langle 0 \rangle \setminus U\langle 0 \rangle] \cap \mathcal{Q} = L\langle 0 \rangle$.

(c)

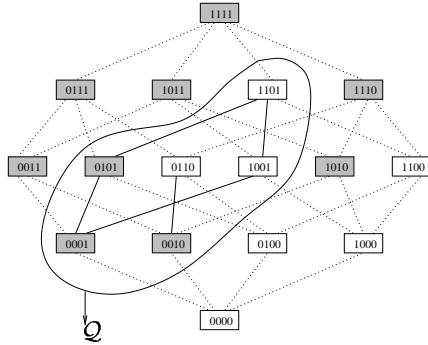
$$\begin{aligned}\mathcal{A}_{(L_{\mathcal{A}}, U_{\mathcal{A}})} &= (L_{\mathcal{A}} \cap \mathcal{Q}\langle 1 \rangle) \cup (U_{\mathcal{A}} \cap \mathcal{Q}\langle 0 \rangle) \\ &= [((\mathcal{Q}\langle 0 \rangle \setminus \mathcal{A}\langle 0 \rangle) \cup \mathcal{A}\langle 1 \rangle) \cap \mathcal{Q}\langle 1 \rangle] \\ &\quad \cup [((\mathcal{Q}\langle 1 \rangle \setminus \mathcal{A}\langle 1 \rangle) \cup \mathcal{A}\langle 0 \rangle) \cap \mathcal{Q}\langle 0 \rangle] \\ &= \mathcal{A}\langle 1 \rangle \cup \mathcal{A}\langle 0 \rangle = \mathcal{A}\end{aligned}$$

(d)

$$\begin{aligned}
L_{\mathcal{A}_{(L,U)}} &= (\mathcal{Q}\langle 0 \rangle \setminus \mathcal{A}_{(L,U)}\langle 0 \rangle) \cup \mathcal{A}_{(L,U)}\langle 1 \rangle \\
&= (\mathcal{Q}\langle 0 \rangle \setminus (U \cap \mathcal{Q}\langle 0 \rangle)) \cup (L \cap \mathcal{Q}\langle 1 \rangle) \\
&= (\mathcal{Q}\langle 0 \rangle \setminus U\langle 0 \rangle) \cup L\langle 1 \rangle \\
&= L\langle 0 \rangle \cup L\langle 1 \rangle = L
\end{aligned}$$

A igualdade $U_{\mathcal{A}_{(L,U)}} = U$ pode ser provada de forma similar. ■

Para ilustrar os conceitos definidos acima, seja $\mathcal{Q} = \{0001, 0010, 0101, 0110, 1001, 1101\}$, conforme Fig. 7.5(a). Neste caso, $\mathcal{Q}\langle 0 \rangle = \{0110, 1001, 1101\}$ e $\mathcal{Q}\langle 1 \rangle = \{0001, 0010, 0101\}$. Considere o conjunto de chaveamento $\mathcal{A} = \{0001, 0010, 1101\}$. Então, $\mathcal{A}\langle 0 \rangle = \{1101\}$, $\mathcal{A}\langle 1 \rangle = \{0001, 0010\}$, e a partição correspondente é $L_{\mathcal{A}} = (\mathcal{Q}\langle 0 \rangle \setminus \mathcal{A}\langle 0 \rangle) \cup \mathcal{A}\langle 1 \rangle = \{0001, 0010, 0110, 1001\}$ e $U_{\mathcal{A}} = (\mathcal{Q}\langle 1 \rangle \setminus \mathcal{A}\langle 1 \rangle) \cup \mathcal{A}\langle 0 \rangle = \{0101, 1101\}$. Por outro lado, considere a partição $(L, U) = (\{0001, 0010, 0110, 1001\}, \{0101, 1101\})$, ilustrado na Fig. 7.5(b). Então, o conjunto de chaveamento correspondente é $\mathcal{A}_{(L,U)} = (L \cap \mathcal{Q}\langle 1 \rangle) \cup (U \cap \mathcal{Q}\langle 0 \rangle) = \{0001, 0010\} \cup \{1101\} = \{0001, 0010, 1101\}$. As figuras 7.5(c) e 7.5(d) mostram, respectivamente, dois outros conjuntos de chaveamento e as respectivas partições.



(a) Conjunto violador.

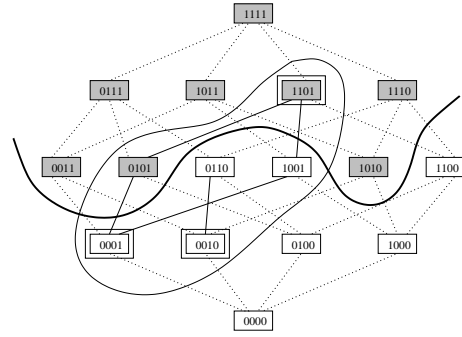
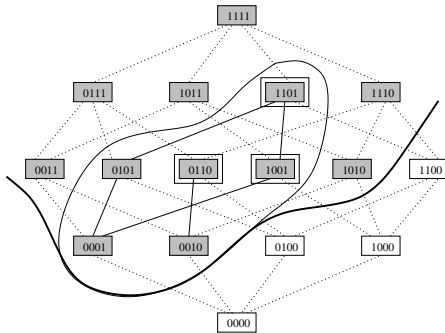
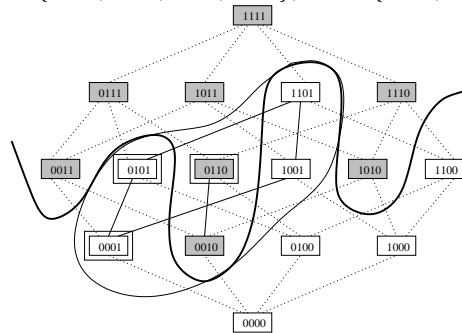
(b) $\mathcal{A} = \{0001, 0010, 1101\}$,
 $L_{\mathcal{A}} = \{0001, 0010, 0110, 1001\}$, $U_{\mathcal{A}} = \{0101, 1101\}$ (c) $\mathcal{A}_1 = \{0110, 1001, 1101\}$,
 $L_{\mathcal{A}_1} = \emptyset$, $U_{\mathcal{A}_1} = \mathcal{Q}$ (d) $\mathcal{A}_2 = \{0001, 0101, 0110\}$,
 $L_{\mathcal{A}_2} = \{0001, 0101, 1001, 1101\}$, $U_{\mathcal{A}_2} = \{0010, 0110\}$

Figura 7.5: Conjuntos de Chaveamento e suas respectivas partições.

Devido à equivalência entre conjuntos de chaveamentos e partições válidas, dado um conjunto de chaveamento \mathcal{A} , o aumento de erro de $\psi_{\text{opt}_{\mathcal{A}}}$ (Eq. 7.4) pode ser reescrito em termos da partição

válida $(L_{\mathcal{A}}, U_{\mathcal{A}})$ como :

$$\tilde{\Delta}(L_{\mathcal{A}}, U_{\mathcal{A}}) = \sum_{X \in U_{\mathcal{A}}\langle 0 \rangle} c(X) + \sum_{X \in L_{\mathcal{A}}\langle 1 \rangle} c(X) \quad (7.5)$$

i.e., $\tilde{\Delta}(L_{\mathcal{A}}, U_{\mathcal{A}}) = \Delta(\psi_{\text{opt}_{\mathcal{A}}}, \psi_{\text{opt}})$.

Em suma, podemos dizer que os seguintes problemas são equivalentes :

- Calcular um operador crescente ótimo.
- Calcular um operador crescente com menor aumento de erro em relação a ψ_{opt} .
- Calcular um conjunto de chaveamento ótimo, isto é, aquele que minimiza Eq. 7.4.
- Calcular uma partição ótima, isto é, aquela que minimiza Eq. 7.5.

7.3.2 O Problema da Partição Ótima

O problema original de projetar um operador crescente ótimo é equivalente ao problema de calcular uma partição ótima do conjunto violador de ψ_{opt} .

O Problema da Partição Válida Ótima (OVP): Sejam dados $\psi_{\text{opt}} : \mathcal{P}(W) \rightarrow \mathcal{P}(W)$ um W-operador com conjunto violador \mathcal{Q} e os custos de chaveamento $c(X)$, $X \in \mathcal{Q}$. Encontrar uma partição válida de \mathcal{Q} que minimiza a Eq. 7.5.

O espaço de soluções para este problema consiste de todas as partições válidas de \mathcal{Q} . Uma vez que o tamanho de \mathcal{Q} pode ser bem grande, algoritmos de força-bruta não são indicados para este problema. Trata-se de um problema essencialmente combinatório. Um bom algoritmo para resolver problemas de otimização combinatória deve ser capaz de encontrar uma solução ótima após analisar apenas poucos candidatos à solução. Para tal, ele deve saber tirar vantagens de alguma propriedade ou estrutura especial do espaço de soluções.

A seguir apresentamos uma série de definições e proposições que serão utilizadas para mostrar que o espaço de soluções do problema de partição ótima possui uma propriedade “greedy”, o qual será explorado pelo algoritmo que propomos mais adiante.

Proposição 7.2 *Seja (L, U) uma partição válida de \mathcal{Q} e seja F um subconjunto de \mathcal{Q} tal que $(\mathcal{Q} \setminus F, F)$ é também uma partição válida de \mathcal{Q} . As seguintes afirmações são verdadeiras.*

- (a) *Se $F \subseteq U$, então $(L, U \setminus F)$ é uma partição válida de $\mathcal{Q} \setminus F$ (Fig. 7.6(a)).*
- (b) *Se $F \subseteq L$, então $(L \setminus F, U \cup F)$ é uma partição válida de \mathcal{Q} (Fig. 7.6(b)).*
- (c) *Se $F \not\subseteq U$ e $F \not\subseteq L$, então $(L \setminus (L \cap F), U \cup (L \cap F))$ é uma partição válida de \mathcal{Q} e $(L \cap F, U \cap F)$ é uma partição válida de F (Fig. 7.6(c)).*
- (d) *Se (L', U') é uma partição válida de $\mathcal{Q} \setminus F$, então $(L', U' \cup F)$ é uma partição válida de \mathcal{Q} (Fig. 7.6(d)).*
- (e) *Se (L'', U'') é uma partição válida de L , então $(L'', U \cup U'') = (L'', \mathcal{Q} \setminus L'')$ é uma partição válida de \mathcal{Q} (Fig. 7.6(e)).*

Dem.: (a) Como $L \cup (U \setminus F) = L \cup (U \cap F^c) = Q \setminus F$ e $L \cap (U \setminus F) = \emptyset$, então $(L, U \setminus F)$ é uma partição de $Q \setminus F$. Além disso, como nenhum elemento de $U \setminus F$ é menor que um elemento de L , $(L, U \setminus F)$ é uma partição válida.

(b) Note que $(L \setminus F) \cup (U \cup F) = Q$. Como $(Q \setminus F, F)$ é uma partição válida de Q , nenhum elemento de F é menor que um elemento de $Q \setminus F$ e, portanto, nenhum elemento de F é menor que um elemento de $L \setminus F$. Além disso, como (L, U) é uma partição válida de Q , nenhum elemento de U é menor que um elemento de L e, portanto, nenhum elemento de U é menor que um elemento de $L \setminus F$ (pois $L \setminus F \subseteq L$). Portanto, nenhum elemento de $U \cup F$ é menor que um elemento de $L \setminus F$. Logo, $(L \setminus F, U \cup F)$ é uma partição válida de Q .

(c) Como $(L \setminus (L \cap F)) \cup (U \cup (L \cap F)) = Q$ e $(L \setminus (L \cap F)) \cap (U \cup (L \cap F)) = \emptyset$, segue que $(L \setminus (L \cap F), U \cup (L \cap F))$ é uma partição de Q . Além disso, como $(Q \setminus F, F)$ é uma partição válida de Q , nenhum elemento de F é menor que um elemento de $L \setminus (L \cap F)$ (pois $L \setminus (L \cap F) \subseteq Q \setminus F$) e, então, nenhum elemento de $L \cap F$ é menor que um elemento de $L \setminus (L \cap F)$ (pois $L \cap F \subseteq F$). Mais ainda, como (L, U) é uma partição válida de Q , nenhum elemento de U é menor que um elemento de $L \setminus (L \cap F)$ (pois $L \setminus (L \cap F) \subseteq L$). Logo, nenhum elemento de $U \cup (L \cap F)$ é menor que um elemento de $L \setminus (L \cap F)$.

O par $(L \cap F, U \cap F)$ é uma partição válida de F pois $(L \cap F) \cup (U \cap F) = F$ e $(L \cap F) \cap (U \cap F) = \emptyset$. Além disso, como $(L \cap F) \subseteq L$ e $U \cap F \subseteq U$, nenhum elemento de $U \cap F$ é menor que um elemento de $L \cap F$.

(d) Como $L' \cup (U' \cup F) = (L' \cup U') \cup F = (Q \setminus F) \cup F = Q$ e $L' \cap (U' \cup F) = (L' \cap U') \cup (L' \cap F) = \emptyset \cup \emptyset = \emptyset$, então $(L', U' \cup F)$ é uma partição de Q . Para mostrar que nenhum elemento de $U' \cup F$ é menor que um elemento de L' , basta mostrarmos que nenhum elemento de F é menor que um elemento de L' , já que nenhum elemento de U' é menor que um elemento de L' (pois (L', U') é uma partição válida de $Q \setminus F$). Mas isto segue do fato de que $(Q \setminus F, F)$ é uma partição válida de Q .

(e) Como $L'' \subseteq L$, nenhum elemento de L'' é menor que um elemento de U . Além disso, como (L'', U'') é uma partição válida de L , então nenhum elemento de L'' é menor que um elemento de U'' . Logo, nenhum elemento de L'' é menor que um elemento de $U \cup U'' = Q \setminus L''$. ■

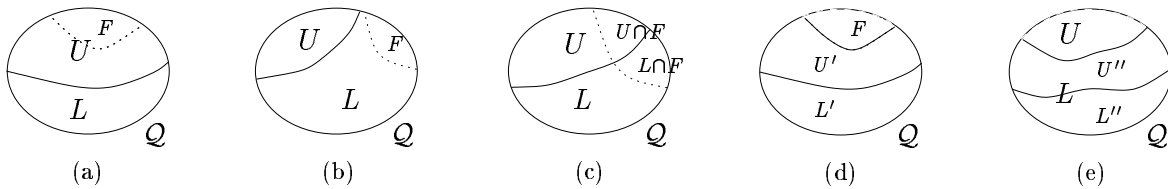


Figura 7.6: Ilustração para a proposição 7.2.

Resultados duais aos da proposição acima são também válidos em relação a subconjuntos F de Q tais que $(F, Q \setminus F)$ é uma partição válida de Q .

Proposição 7.3 Considere a partição válida trivial (Q, \emptyset) de Q . Então

$$\tilde{\Delta}(Q, \emptyset) = \sum_{X \in Q(1)} c(X)$$

■

Dualmente, com relação a outra partição válida trivial (\emptyset, \mathcal{Q}) vale que $\tilde{\Delta}(\emptyset, \mathcal{Q}) = \sum_{X \in \mathcal{Q}\langle 0 \rangle} c(X)$. Estes são os aumentos de erro nos casos em que, respectivamente, todos os elementos de $\mathcal{Q}\langle 1 \rangle$ são chaveados para 0 e todos os elementos de $\mathcal{Q}\langle 0 \rangle$ são chaveados para 1.

Se (L, U) é uma partição válida de \mathcal{Q} , então o seu aumento de erro $\tilde{\Delta}(L, U)$ (Eq. 7.5) pode ser reescrito em termos de $\tilde{\Delta}(\mathcal{Q}, \emptyset)$, como

$$\tilde{\Delta}(L, U) = \tilde{\Delta}(\mathcal{Q}, \emptyset) + \sum_{X \in U\langle 0 \rangle} c(X) - \sum_{X \in U\langle 1 \rangle} c(X) \quad (7.6)$$

ou, dualmente, como

$$\tilde{\Delta}(L, U) = \tilde{\Delta}(\emptyset, \mathcal{Q}) + \sum_{X \in L\langle 1 \rangle} c(X) - \sum_{X \in L\langle 0 \rangle} c(X) \quad (7.7)$$

Logo, podemos dizer que uma partição (L, U) é preferível à partição (\mathcal{Q}, \emptyset) se e somente se $\sum_{X \in U\langle 0 \rangle} c(X) - \sum_{X \in U\langle 1 \rangle} c(X) < 0$; ela é preferível à partição (\emptyset, \mathcal{Q}) se e somente se $\sum_{X \in L\langle 1 \rangle} c(X) - \sum_{X \in L\langle 0 \rangle} c(X) < 0$.

Para simplificar a apresentação de alguns resultados, definimos a noção de *peso* de um subconjunto.

Definição 7.4 O peso de um subconjunto Z de \mathcal{Q} é definido como

$$\omega(Z) = \sum_{X \in Z\langle 0 \rangle} c(X) - \sum_{X \in Z\langle 1 \rangle} c(X)$$

Note que, em particular,

$$\omega(\{X\}) = \begin{cases} c(X), & \text{se } X \in \mathcal{Q}\langle 0 \rangle \\ -c(X), & \text{se } X \in \mathcal{Q}\langle 1 \rangle \end{cases}$$

Dado um subconjunto $Z \subseteq \mathcal{Q}$, suponha que apenas duas escolhas são permitidas: ou chavear todos os elementos de $Z\langle 0 \rangle$ para 1, ou então, chavear todos os elementos de $Z\langle 1 \rangle$ para 0. Neste caso, se o objetivo é minimizar o aumento de erro, deve-se escolher aquele conjunto de chaveamentos com o menor custo de chaveamento. Por exemplo, se $\sum_{X \in Z\langle 1 \rangle} c(X) > \sum_{X \in Z\langle 0 \rangle} c(X)$, i.e, $\omega(Z) < 0$, então é melhor que os elementos de $Z\langle 0 \rangle$ sejam chaveados para 1 do que o contrário. Se $\omega(Z) > 0$, então é melhor que os elementos de $Z\langle 1 \rangle$ sejam chaveados para 0 do que o contrário. Se $\omega(Z) = 0$, ambas as escolhas produzem o mesmo aumento de erro. Neste caso, utilizamos a convenção de sempre chavear de 1 para 0. Os dois primeiros casos estão ilustrados nas figuras 7.7(a) e 7.7(b), respectivamente. Os números ao lado de cada elemento representam o custo de chaveamento.

Proposição 7.5 Dados dois subconjuntos disjuntos Z_1 e Z_2 de \mathcal{Q} , $\omega(Z_1 \cup Z_2) = \omega(Z_1) + \omega(Z_2)$.

■

Proposição 7.6 Seja (L, U) uma partição válida de \mathcal{Q} . Então

$$\tilde{\Delta}(L, U) = \tilde{\Delta}(\mathcal{Q}, \emptyset) + \omega(U)$$



Figura 7.7: Custo de chaveamento e peso.

Dem.:

$$\begin{aligned}
 \tilde{\Delta}(L, U) &= \sum_{X \in L\langle 1 \rangle} c(X) + \sum_{X \in U\langle 0 \rangle} c(X) \\
 &= \sum_{X \in L\langle 1 \rangle} c(X) + \sum_{X \in U\langle 0 \rangle} c(X) + \sum_{X \in U\langle 1 \rangle} c(X) - \sum_{X \in U\langle 1 \rangle} c(X) \\
 &= \tilde{\Delta}(\mathcal{Q}, \emptyset) + \omega(U)
 \end{aligned}$$

■

A proposição 7.6 reexpressa a Eq. 7.6 em termos de peso e afirma que o aumento de erro de uma partição pode ser expressa em termos do aumento de erro da partição trivial e o peso da parte superior da partição. Dualmente, com relação à Eq. 7.7, vale que $\tilde{\Delta}(L, U) = \tilde{\Delta}(\emptyset, \mathcal{Q}) - \omega(L)$.

Proposição 7.7 *Seja (L, U) uma partição válida de \mathcal{Q} , e (L', U') uma partição válida de L (veja figura 7.8(a)). Então, $(L \setminus U', U \cup U')$ é uma partição válida de \mathcal{Q} e*

$$\tilde{\Delta}(L \setminus U', U \cup U') = \tilde{\Delta}(L, U) + \omega(U')$$



Figura 7.8: Ilustração para a proposição 7.7.

Dem.: O par $(L \setminus U', U \cup U')$ é uma partição válida de \mathcal{Q} (Proposição 7.2(e)). Além disso,

$$\begin{aligned}
 \tilde{\Delta}(L \setminus U', U \cup U') &= \tilde{\Delta}(\mathcal{Q}, \emptyset) + \omega(U \cup U') \\
 &= \tilde{\Delta}(\mathcal{Q}, \emptyset) + \omega(U') + \omega(Y) \\
 &= \tilde{\Delta}(L' \cup U', U) + \omega(U') \\
 &= \tilde{\Delta}(L, U) + \omega(U')
 \end{aligned}$$

■

A Proposição 7.7 é uma generalização da proposição 7.6. Quando um conjunto de elementos é transferido da parte inferior para a parte superior, a variação no aumento de erro entre a partição original e a nova partição é dado pelo peso do subconjunto transferido. Dualmente, se (L'', U'') é uma partição válida de U , então $(L \cup L'', U \setminus L'')$ é uma partição válida de Q e $\tilde{\Delta}(L \cup L'', U \setminus L'') = \tilde{\Delta}(L, U) - \omega(L'')$ (veja Fig. 7.8(b)). Este corresponde ao caso onde um subconjunto da parte superior é transferido para a parte inferior.

Para encontrar uma partição ótima, poderia-se escolher uma partição inicial e transferir, sucessivamente, pequenos subconjuntos da parte superior para a parte inferior e vice-versa, tomando-se o cuidado de que o aumento de erro da partição resultante (Proposição 7.7) sempre diminua. Entretanto, existem certas questões não triviais com respeito a escolha da partição inicial e o número de iterações necessárias até que haja uma convergência. Também não se sabe se a escolha inicial pode ou não afetar a convergência para a partição ótima.

Propomos aqui uma outra abordagem que consiste em analisar pequenas porções do conjunto violador de cada vez. Isto é, se pequenas porções são analisadas, pode ser possível decidir se esta porção deve pertencer à parte superior ou inferior da partição ótima. Supondo que tais decisões sejam possíveis, o procedimento consistiria em iniciar o processo com ambas as partes, superior e inferior, vazias e remover as pequenas porções do conjunto violador para a parte adequada da partição. Esta é uma abordagem construtiva, na qual os elementos são pouco a pouco agregados às partes da partição, enquanto o conjunto violador vai diminuindo.

Note que na segunda abordagem o tamanho do problema (tamanho do conjunto violador) é gradualmente reduzido, enquanto que na primeira abordagem o problema é sempre do mesmo tamanho.

Para ilustrar as idéias básicas da segunda abordagem, consideramos o conjunto violador e os respectivos custos de chaveamento mostrados na Fig. 7.9(a). Se analisamos o subconjunto $\{g, h, i\}$, podemos verificar que o custo para chavear g de 0 para 1 é menor que o custo para chavear ao mesmo tempo h e i de 1 para 0. Uma vez que chavear g de 0 para 1 não afeta outros elementos no conjunto violador, podemos afirmar que $\{g, h, i\}$ pode ser transferido para a parte superior com segurança (Fig. 7.9(b)). Os elementos removidos do conjunto violador são indicados por linhas pontilhadas, dentre os quais aqueles que são escuros indicam elementos da parte superior, enquanto aqueles não hachurados indicam os elementos da parte inferior. A seguir, como o ato de chavear f de 1 para 0 não afeta outros elementos, e como o custo para chavear f para 0 é menor do que o custo de não chaveá-lo (pois isto implicaria que pelo menos d teria de ser chaveado para 1) o subconjunto $\{d, f\}$ pode ser removido para a parte inferior com segurança (Fig. 7.9(c)). Analisando o restante do conjunto violador, $\{a, b, c, e\}$, podemos ver, por exemplo, que $\{a, c\}$ pode ser removido com segurança para a parte superior (Fig. 7.9(d)). Neste caso, os dois últimos elementos não precisam ser chaveados e portanto podem ser movidos para a parte superior também (Fig. 7.9(e)). A partição resultante é mostrada na Fig. 7.9(f), onde os elementos chaveados estão indicados por círculos duplos.

Note que no exemplo anterior a seqüência na qual os subconjuntos foram removidos do conjunto violador não é única possível. Além disso, os subconjuntos removidos não precisam ser necessariamente aqueles ilustrados no exemplo. Por exemplo, em vez de se escolher $\{a, c\}$ e depois $\{b, e\}$, poderíamos ter escolhido $\{a, b, c, e\}$, ou $\{a, b\}$ e então $\{c, e\}$, e ainda assim obteríamos o mesmo resultado. Mais ainda, estes subconjuntos devem ser escolhidos com bastante cuidado. Por exemplo, no caso mostrado na Fig. 7.10, poderia-se simplesmente remover todos os elementos para a parte superior já que $\omega < 0$; porém a escolha ótima é $\{a, c\}$ na parte superior e $\{b, d\}$ na parte inferior.

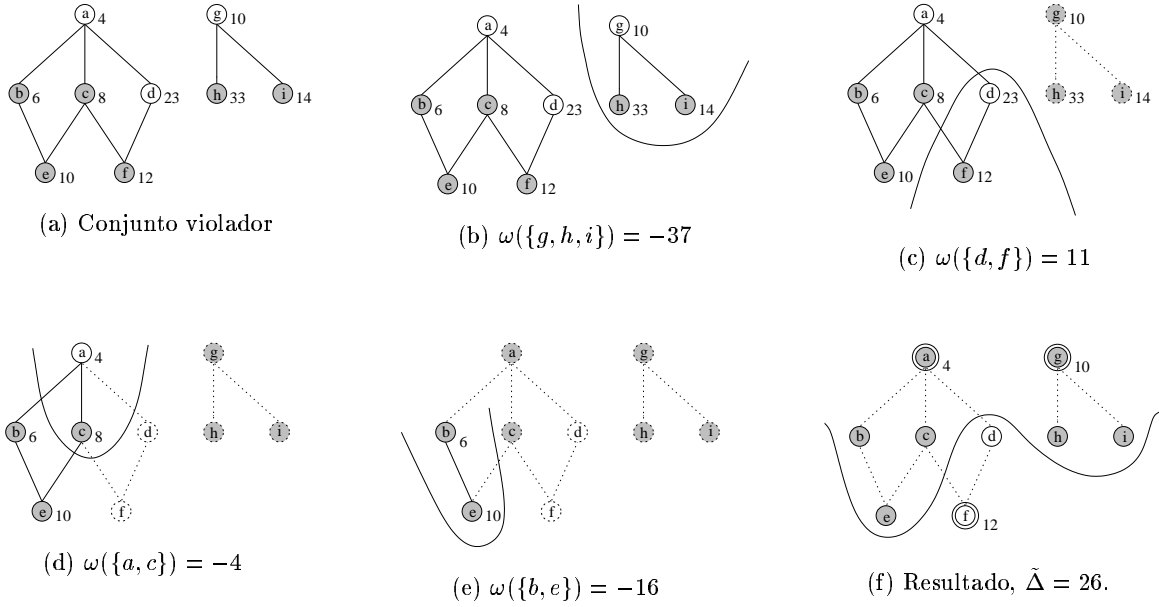


Figura 7.9: Dinâmica do particionamento.

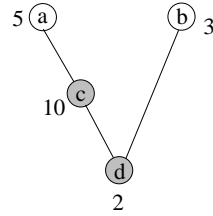


Figura 7.10: Um conjunto violador.

7.3.3 Conjuntos Viáveis

Para formalizar a noção de remover um conjunto com segurança e eliminar as ambigüidades na escolha dos subconjuntos a serem removidos, introduzimos a noção de *conjuntos viáveis*. Depois mostramos que um algoritmo que sempre remove um conjunto viável gera uma partição ótima de \mathcal{Q} .

Definição 7.8 Seja \mathcal{F}_U a classe dos subconjuntos não-vazios F de \mathcal{Q} (veja Fig. 7.11(a)) satisfazendo

1. $(\mathcal{Q} \setminus F, F)$ é uma partição válida de \mathcal{Q} , e
2. $\omega(F) < 0$.

Um subconjunto $F \in \mathcal{F}_U$ é U-viável se e somente se F é minimal em \mathcal{F}_U com relação a \subseteq .

Definição 7.9 Seja \mathcal{F}_L a classe dos subconjuntos não-vazios F de \mathcal{Q} (veja Fig. 7.11(b)) satisfazendo

1. $(F, \mathcal{Q} \setminus F)$ é uma partição válida de \mathcal{Q} , e
2. $\omega(F) \geq 0$.

Um subconjunto $F \in \mathcal{F}_L$ é L-viável se e somente se F é minimal em \mathcal{F}_L com relação a \subseteq .



Figura 7.11: Ilustração para as definições 7.8 e 7.9.

Por conveniência, (\emptyset, \emptyset) é considerada uma partição válida ótima de \emptyset . Para nos referirmos aos conjuntos U - ou L -viáveis, sem uma distinção explícita, usamos simplesmente o termo *conjuntos viáveis*.

Lema 7.10 (a) Se um conjunto violador Q não contém um subconjunto U -viável, então ele contém um subconjunto L -viável.

(b) Se um conjunto violador Q não contém um subconjunto L -viável, então ele contém um subconjunto U -viável.

Dem.: (a) Se Q não contém um subconjunto U -viável, então, pela Def. 7.8, para qualquer partição válida (L, U) de Q , tem-se $\omega(U) \geq 0$. Em particular, temos que $(L, U) = (\emptyset, Q)$, o que significa que \mathcal{F}_L é não-vazio.

(b) A prova é análoga. ■

Teorema 7.11 Seja F um conjunto viável de Q , e seja (L', U') uma partição ótima de $Q \setminus F$. Então,

(a) se F é U -viável, então $(L', U' \cup F)$ é uma partição ótima de Q , e

(b) se F é L -viável, então $(L' \cup F, U')$ é uma partição ótima de Q .

Dem.: (a) Para provar que $(L', U' \cup F)$ é uma partição ótima de Q , precisamos apenas provar que ela é ótima, uma vez que já sabemos, pela Proposição 7.2(a), que ela é uma partição válida de Q . Vamos supor, por absurdo, que ela não é ótima, e então mostraremos que esta hipótese sempre conduz a uma contradição.

Se $(L', U' \cup F)$ não é uma partição ótima de Q , então existe alguma partição válida (L, U) de Q tal que $\tilde{\Delta}(L, U) < \tilde{\Delta}(L', U' \cup F)$. Vamos escolher, em particular, uma partição ótima, que denotaremos (L^*, U^*) .

Vamos analisar a relação de F com U^* e L^* , e mostrar que todos os possíveis casos ((1) $F \subseteq L^*$, (2) $F \subseteq U^*$, e (3) $F \not\subseteq U^*$ e $F \not\subseteq L^*$), conduzem-nos a alguma contradição.

Caso 1: Se $F \subseteq L^*$, então $(L^* \setminus F, U^* \cup F)$ é uma partição válida de Q (proposição 7.2(c)). Da Proposição 7.7 e do fato de que F é um conjunto U -viável segue que

$$\tilde{\Delta}(L^* \setminus F, U^* \cup F) = \tilde{\Delta}(L^*, U^*) + \omega(F) < \tilde{\Delta}(L^*, U^*).$$

Isto é um absurdo, pois (L^*, U^*) é uma partição ótima de Q .

Caso 2: Se $F \subseteq U^*$, então $(L^*, U^* \setminus F)$ é uma partição válida de Q' (Proposição 7.2(b)). Mais ainda, da Proposição 7.7, temos que

$$\begin{aligned} \tilde{\Delta}(L^*, U^*) &= \tilde{\Delta}(L^*, F \cup (U^* \setminus F)) = \\ \tilde{\Delta}(L^* \cup (U^* \setminus F), F) &+ \omega(U^* \setminus F) = \tilde{\Delta}(Q', F) + \omega(U^* \setminus F) \end{aligned}$$

e

$$\tilde{\Delta}(L', U' \cup F) = \tilde{\Delta}(Q' \setminus U', F \cup U') = \tilde{\Delta}(Q', F) + \omega(U').$$

Como (L^*, U^*) é uma partição ótima de Q , então $\tilde{\Delta}(L^*, U^*) \leq \tilde{\Delta}(L', U' \cup F)$, o que significa que das duas expressões acima segue que $\omega(U^* \setminus F) \leq \omega(U')$. Note que a igualdade não pode valer pois $(L', U' \cup F)$ seria uma partição ótima de Q , contradizendo a hipótese. Logo, deve ser verdade que $\omega(U^* \setminus F) < \omega(U')$, e então, da Proposição 7.6, segue que

$$\tilde{\Delta}(L^*, U^* \setminus F) = \tilde{\Delta}(Q', \emptyset) + \omega(U^* \setminus F) < \tilde{\Delta}(Q', \emptyset) + \omega(U') = \tilde{\Delta}(L', U')$$

o que é um absurdo pois (L', U') é uma partição ótima de Q' .

Case 3: Se $F \not\subseteq L^*$ e $F \not\subseteq U^*$, então seja $X = F \cap L^*$. Da Proposição 7.2(d), $(L^* \setminus X, U^* \cup X)$ é uma partição válida de Q . Também sabemos que $\omega(X) \geq 0$ pois em caso contrário, pela Proposição 7.7, $(L^* \setminus X, U^* \cup X)$ seria uma partição válida de Q com aumento de erro menor que (L^*, U^*) , o que é um absurdo. Como $\omega(F) < 0$ (pois F é um conjunto U -viável), da Proposição 7.5 concluímos que $\omega(F \setminus X) + \omega(X) = \omega(F) < 0$. Disto, e do fato de que $\omega(X) \geq 0$, segue que $\omega(F \setminus X) < -\omega(X) \leq 0$. Isto, $\omega(F \setminus X) < 0$, é um absurdo pois $F \setminus X \subseteq F \in \mathcal{F}_U$, o que viola a condição de minimalidade de F .

(b) A prova é análoga. ■

7.3.4 O Algoritmo

O teorema 7.11 é a base para o algoritmo que propomos. O algoritmo começa com as partes inferior e superior vazias e sucessivamente remove subconjuntos viáveis do conjunto violador para uma das partes. Como o processo é guloso, uma vez que um subconjunto é removido do conjunto violador, ele nunca será colocado de volta ao conjunto violador. O lema 7.10 mostra que sempre existe pelo menos um subconjunto viável no conjunto violador restante, portanto pode-se garantir que o algoritmo termina.

Teorema 7.12 *Dado um conjunto violador Q e custos de chaveamento $c(X)$ para cada elemento X de Q , o seguinte algoritmo (algoritmo OVP) produz uma solução ótima para o problema OVP.*

Entrada: Um conjunto violador Q ; custos de chaveamento $c(X)$, $\forall X \in Q$
Saída: Partição ótima (L, U) de Q .

-
1. Faça $U \leftarrow \emptyset$ e $L \leftarrow \emptyset$.
 2. Se Q é vazio, então devolva (L, U) e pare.
 3. Procure um conjunto viável F em Q .
 Se F é U -viável, então faça $U \leftarrow U \cup F$;
 se F é L -viável, então faça $L \leftarrow L \cup F$.
 Faça $Q \leftarrow Q \setminus F$ e retorne ao passo 2.

Algoritmo 7.1 *Algoritmo OVP.*

Dem.: O caso $\mathcal{Q} = \emptyset$ é trivial. Caso contrário, existe pelo menos um conjunto viável em \mathcal{Q} (Lema 7.10). Seja então F_1, F_2, \dots, F_t , $t \geq 1$, a seqüência de conjuntos viáveis encontrados pelo algoritmo. Note que $\mathcal{Q} = F_1 \cup F_2 \cup \dots \cup F_t$.

Se F_t é um conjunto U -viável, então (\emptyset, F_t) é uma partição ótima de F_t e, se F_t é um conjunto L -viável então (F_t, \emptyset) é uma partição ótima de F_t (Teorema 7.11), com $(L', U') = (\emptyset, \emptyset)$.

Agora suponha que (L', U') é uma partição ótima de $F_{i+1} \cup \dots \cup F_t$. Pelo teorema 7.11, se F_i é um conjunto U -viável, então $(L', U' \cup F_i)$ é uma partição ótima de $F_i \cup F_{i+1} \cup \dots \cup F_t$, e se F_i é um conjunto L -viável então $(L' \cup F_i, U')$ é uma partição ótima de $F_i \cup F_{i+1} \cup \dots \cup F_t$, $i = t-1, t-2, \dots, 1$. Em outras palavras, dada uma partição ótima de $F_{i+1} \cup \dots \cup F_t$, a partição ótima de $F_i \cup F_{i+1} \cup \dots \cup F_t$ pode ser construída adicionando-se F_i à parte superior ou inferior, dependendo se F_i é um conjunto U ou L -viável, respectivamente.

Logo, a partição de \mathcal{Q} tal que suas partes superior e inferior são formadas por todos os conjuntos U e L -viáveis, respectivamente, é uma partição ótima de \mathcal{Q} . Como esta é exatamente a partição devolvida pelo algoritmo, ela é ótima. ■

As figuras 7.12 e 7.13 mostram dois exemplos de aplicação do algoritmo proposto. Elas ilustram uma seqüência de subconjuntos viáveis sendo removidos de dois conjuntos violadores diferentes. As figuras mostram, respectivamente, o estado inicial do conjunto violador, os estados intermediários gerados pelo algoritmo (onde os subconjuntos viáveis encontrados são evidenciados por uma curva e os elementos removidos são indicados por linhas pontilhadas), e a partição resultante.

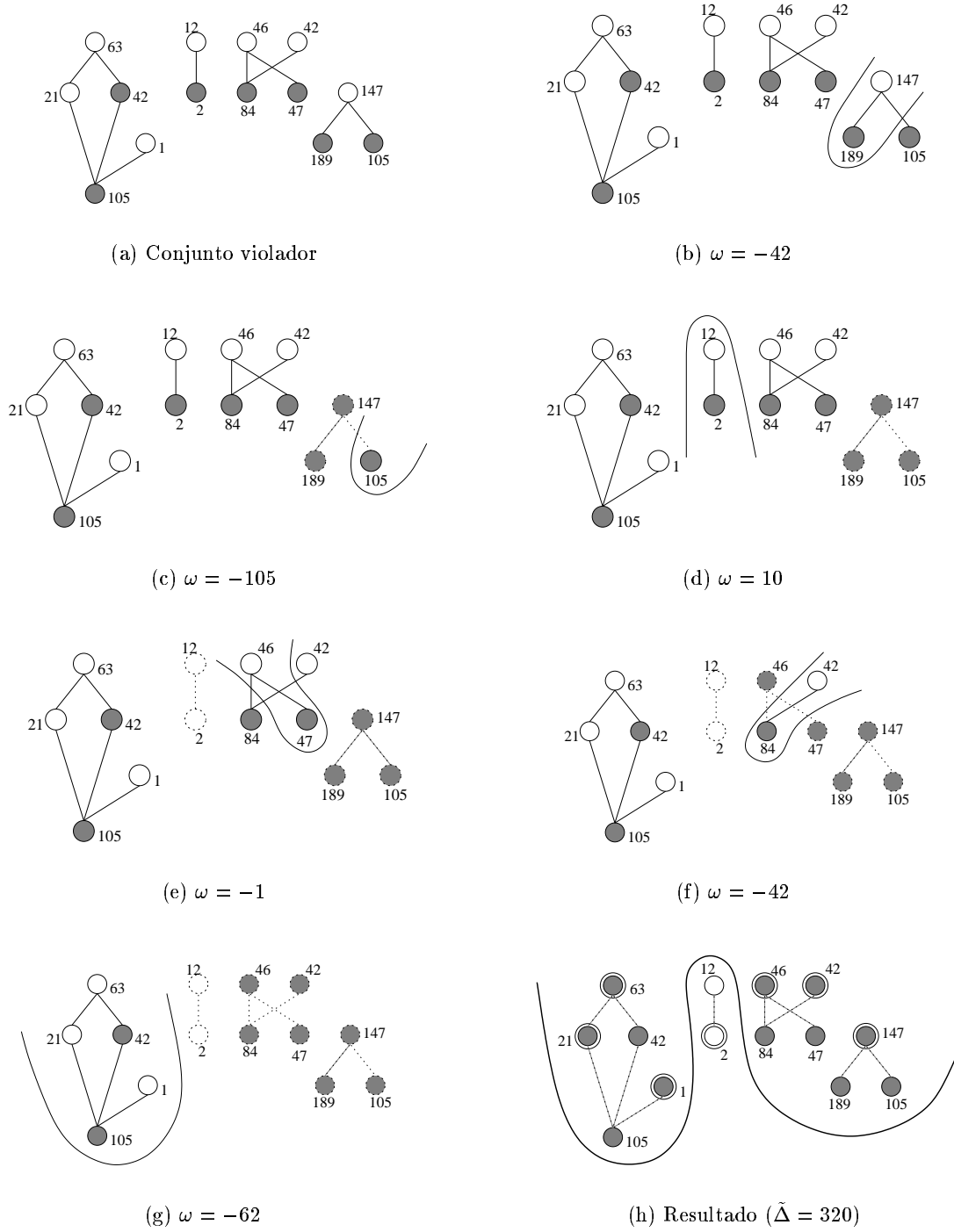
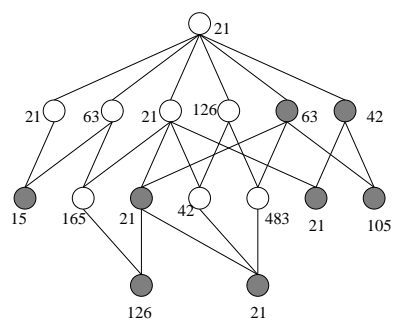


Figura 7.12: Aplicação do algoritmo OVP - exemplo 1



(a) Conjunto violador

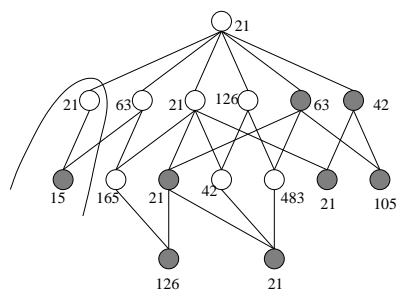
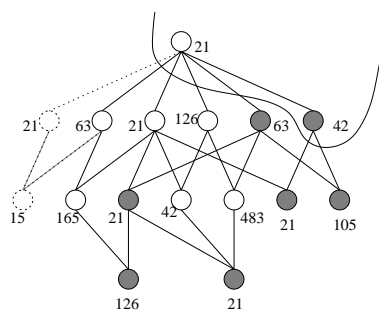
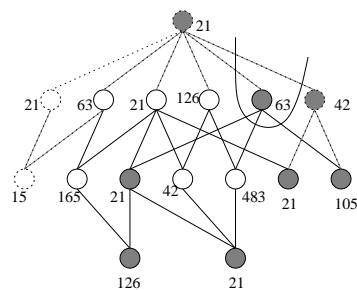
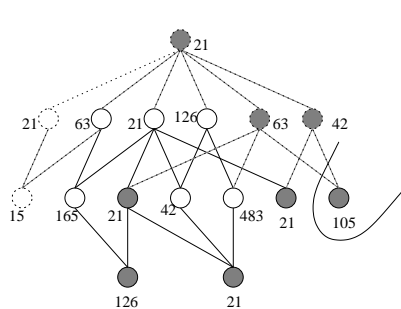
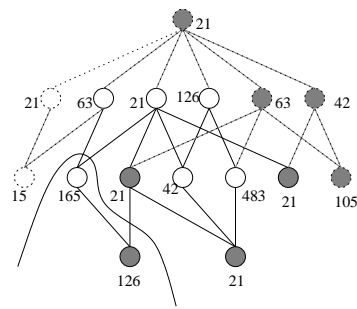
(b) $\omega = 6$ (c) $\omega = -21$ (d) $\omega = -63$ (e) $\omega = -105$ (f) $\omega = 39$

Figura 7.13: Aplicação do algoritmo OVP - exemplo 2.

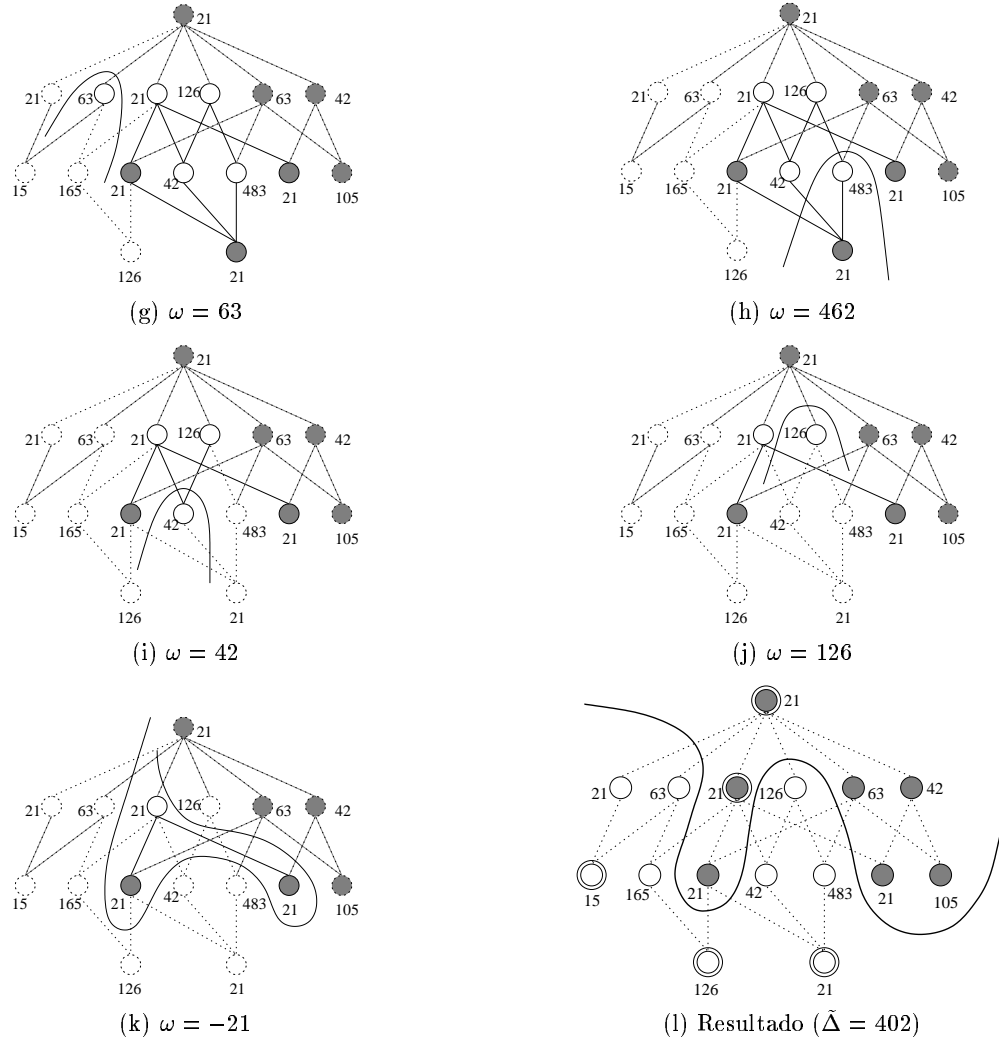


Figura 7.13: Aplicação do algoritmo OVP - exemplo 2 (continuação).

7.3.5 Como Encontrar Conjuntos Viáveis

Nesta seção apresentamos uma estratégia de busca por conjuntos viáveis e discutimos uma implementação relaxada, cujo objetivo é controlar a complexidade do algoritmo.

Definição 7.13 *Seja $T = \{Z_1, Z_2, \dots, Z_k\}$, $k \geq 1$, um subconjunto do conjunto violador \mathcal{Q} . Definimos*

$$\begin{aligned} U[T] &= \{X \in \mathcal{Q} : \exists Z \in T, X \geq Z\} \\ L[T] &= \{X \in \mathcal{Q} : \exists Z \in T, X \leq Z\} \end{aligned}$$

Proposição 7.14 (a) *Se $(\mathcal{Q} \setminus F, F)$ é uma partição válida de \mathcal{Q} e X_1, X_2, \dots, X_k são os elementos minimais de F , então $F = U[\{X_1, X_2, \dots, X_k\}]$.*

(b) *Se $(F, \mathcal{Q} \setminus F)$ é uma partição válida de \mathcal{Q} e X_1, X_2, \dots, X_k são os elementos maximais de F , então $F = L[\{X_1, X_2, \dots, X_k\}]$.*

Dem.: (a) *Se $X \in F$, então, pela condição de minimalidade, $X \geq X_i$ para algum $i \in \{1, 2, \dots, k\}$. Logo, $X \in U[\{X_1, X_2, \dots, X_k\}]$.*

Por outro lado, se $X \in U[\{X_1, X_2, \dots, X_k\}]$, então existe X_i , para algum $i \in \{1, 2, \dots, k\}$, tal que $X \geq X_i$. Suponha por absurdo que $X \notin F$. Neste caso, teríamos $X \in \mathcal{Q} \setminus F$, e $(\mathcal{Q} \setminus F, F)$ não seria uma partição válida de \mathcal{Q} , pois $X_i \in F$ é menor que $X \in \mathcal{Q} \setminus F$. Isto é um absurdo.

(b) *A prova é análoga.* ■

Proposição 7.15 (a) *Todos os elementos minimais de um conjunto U -viável F de \mathcal{Q} são elementos de $\mathcal{Q}\langle 1 \rangle$.*

(b) *Todos os elementos maximais de um conjunto L -viável F de \mathcal{Q} são elementos de $\mathcal{Q}\langle 0 \rangle$.*

Dem.: (a) *Suponha por absurdo que alguns elementos minimais de F estão em $\mathcal{Q}\langle 0 \rangle$, e seja X um deles. Por definição, $\omega(\{X\}) \geq 0$. Logo,*

$$\begin{aligned} \omega(F) &= \omega(F \setminus \{X\}) + \omega(\{X\}) \\ \omega(F \setminus \{X\}) &= \omega(F) - \omega(\{X\}) \leq \omega(F) < 0 \end{aligned}$$

Então, $(\{X\}, F \setminus \{X\})$ é uma partição válida de F e $\omega(F \setminus \{X\}) < 0$, o que contradiz o fato de que F é um conjunto U -viável, pois $F \setminus \{X\} \subseteq F$.

(b) *A prova é análoga.* ■

Das proposições 7.14 e 7.15, fica claro que quando procurando \mathcal{Q} por conjuntos U -viáveis, um algoritmo precisa considerar apenas os subconjuntos da forma $U[\{X_1, X_2, \dots, X_k\}]$, $k \geq 1$, onde $X_i \in \mathcal{Q}\langle 1 \rangle$, para todo $i \in \{1, 2, \dots, k\}$. Mais ainda, devido a condição de minimalidade de conjuntos viáveis, o algoritmo precisa garantir que nenhum subconjunto próprio do conjunto sendo analisado é um conjunto U -viável. Em outras palavras, $U[\{X_1, X_2, \dots, X_k\}]$ deve ser testado somente após todos os seus subconjuntos da forma $U[\{X_{i_1}, X_{i_2}, \dots, X_{i_j}\}]$, $1 \leq j < k$, $i_l \in \{1, 2, \dots, k\}$, $1 \leq l \leq j$, $i_l \neq i_t$ se $l \neq t$, terem sido testados. O mesmo tipo de raciocínio vale também para conjuntos L -viáveis.

Na prática, o procedimento de busca é realizado sobre o grafo correspondente a \mathcal{Q} . Os elementos de \mathcal{Q} correspondem aos nós do grafo. Dois nós correspondendo aos elementos X e Y , onde $X < Y$, são ligados por uma aresta se e somente se não existe nenhum outro elemento $Z \in \mathcal{Q}$ tal que $X < Z < Y$. Um nó correspondendo a um elemento X será referenciado por X também.

Para facilitar a descrição do procedimento de busca, definimos a noção de *nível* de um nó. Os nós maximais de $\mathcal{Q}\langle 1 \rangle$ são definidos como nós no nível 0. Um nó $X \in \mathcal{Q}\langle 1 \rangle$ está no nível 1 se e somente se todos os 1-elementos em $U[\{X\}] \setminus \{X\}$ estão no nível 0. Em geral, se t é o nível máximo de todos os nós em $(U[\{X\}] \setminus \{X\}) \cap \mathcal{Q}\langle 1 \rangle$, então o nível de X é definido como $t + 1$. Denotamos o nível de X por $l(X)$. Logo, $\forall X \in \mathcal{Q}\langle 1 \rangle$, tal que X não é maximal em $\mathcal{Q}\langle 1 \rangle$,

$$l(X) = \max\{l(Y) : Y \in (U[\{X\}] \setminus \{X\}) \cap \mathcal{Q}\langle 1 \rangle\} + 1.$$

Suponha que $U[\{X\}]$, $X \in \mathcal{Q}\langle 1 \rangle$ e $l(X) = t$. Os subconjuntos de $U[\{X\}]$ que potencialmente podem ser U -viáveis são aqueles cujos elementos minimais estão em $(U[\{X\}] \setminus \{X\}) \cap \mathcal{Q}\langle 1 \rangle$, o que significa que o nível desses nós é menor do que t . Portanto, contanto que todos os subconjuntos correspondentes a combinações de elementos de $\mathcal{Q}\langle 1 \rangle$ com nível menor do que t tenham sido examinados (e não sejam U -viáveis) o algoritmo poderá proceder para examinar o conjunto $U[\{X\}]$; ele apenas precisa verificar se este satisfaz a condição (2) da Def. 7.8, i.e., se $\omega(U[\{X\}]) < 0$. Mais precisamente, o algoritmo pode proceder o teste com $U[\{X\}]$ se todos os subconjuntos correspondendo às combinações de elementos em $\mathcal{Q}\langle 1 \rangle \cap (U[\{X\}] \setminus \{X\})$ já tiverem sido examinados. Generalizando esta idéia, um subconjunto $U[\{X_1, X_2, \dots, X_k\}]$ deve ser examinado somente após todos os seus subconjuntos correspondendo às combinações de elementos em $(\bigcup_{i=1}^k U[\{X_i\}]) \cap \mathcal{Q}\langle 1 \rangle$ terem sido examinados.

Temos, portanto, o seguinte procedimento de busca. Primeiramente, examinam-se todos os subconjuntos da forma $U[\{X\}]$, onde $X \in \mathcal{Q}\langle 1 \rangle$ e $l(X) = 0$. A seguir, examinam-se subconjuntos da forma $U[\{X_1, X_2\}]$, onde tanto X_1 e X_2 estão em $\mathcal{Q}\langle 1 \rangle$ e tem nível 0. Prossegue-se tomando combinações de três, quatro e mais elementos do nível 0. Uma vez que todas as combinações de elementos no nível 0 forem examinadas, examinam-se os subconjuntos com um elemento minimal no nível 1. Antes de examinar combinações de dois ou mais elementos no nível 1, cada elemento X no nível 1 precisa ser combinado com cada combinação de elementos do nível 0. A seguir, combinações de dois elementos no nível 1 precisam ser combinados com todas as possíveis combinações de elementos no nível 0, e assim por diante. O número de possíveis combinações cresce rapidamente com o aumento dos níveis. Porém, cada vez que um conjunto viável é encontrado, o grafo e o nível do elementos afetados precisam ser atualizados e o procedimento de busca deve ser reinicializado no nível 0 sobre o novo grafo. Estritamente falando, somente as combinações que envolvem elementos X tais que alguma porção de $U[\{X\}]$ foi removido é que precisam ser analisados novamente. No caso dos conjuntos L -viáveis, a mesma idéia se aplica; neste caso, os níveis aumentam de baixo para cima.

Observa-se na prática que, em geral, os conjuntos viáveis possuem poucos elementos minimais/maximais. Portanto, o procedimento de busca não precisa, em geral, testar todas as possíveis combinações descritas anteriormente. Entretanto, é possível que em alguns casos, em algum passo do algoritmo, todos os conjuntos viáveis existentes sejam muito grandes. Nestes casos, o algoritmo precisará examinar um número grande de combinações até encontrar um conjunto viável, constituindo uma situação crítica em termos de tempo de processamento necessário.

7.3.6 Busca Relaxada

Para controlar a complexidade combinatória do problema de busca, impomos algumas restrições sobre o número de elementos minimais/maximais em um conjunto viável.

Definição 7.16 *Seja $\mathcal{F}_U(k)$ uma classe de subconjuntos não-vazios F de \mathcal{Q} satisfazendo*

1. $(\mathcal{Q} \setminus F, F)$ é uma partição válida de \mathcal{Q} ,
2. $\omega(F) < 0$, e
3. F tem no máximo k elementos minimais.

Um subconjunto $F \in \mathcal{F}_U(k)$ é U- k -viável se e somente se F é minimal em $\mathcal{F}_U(k)$ com relação a \subseteq .

Conjuntos L - k -viáveis são definidos de forma similar, apenas trocando os elementos minimais por maximais.

Estas duas definições introduzem o que chamamos simplesmente de *conjuntos k -viáveis*. Se somente conjuntos k -viáveis são procurados, então o procedimento de busca descrito anteriormente precisa verificar combinações de no máximo k elementos minimais/maximais, e portanto o tempo de processamento pode ser controlado através do parâmetro k . Este relaxamento pode introduzir sub-otimalidade, no sentido de o resultado não ser ótimo. Vale notar, entretanto, que sempre é possível escolher um k suficientemente grande para obter um resultado ótimo.

Daqui para frente, referimos ao algoritmo que procura somente conjuntos k -viáveis como *algoritmo k -relaxado* em oposição ao algoritmo original (não-relaxado). Analisamos a seguir algumas situações que podem decorrer devido ao relaxamento.

Seja \mathcal{Q} o conjunto violador mostrado na Fig. 7.14(a). Existe apenas um subconjunto 1-viável, $U[\{i\}]$, em \mathcal{Q} . Como $U[\{i\}] = \mathcal{Q}$, após remover \mathcal{Q} para a parte superior o algoritmo 1-relaxado para, retornando como resultado a partição (\emptyset, \mathcal{Q}) , como mostra a Fig. 7.14(b). Entretanto, $U[\{i\}]$ não é um conjunto viável no sentido estrito pois ele contém os subconjuntos $U[\{d, f\}]$, $L[\{g, h\}]$, $L[\{g, e\}]$ e $L[\{e, h\}]$ que são viáveis. Todos estes conjuntos viáveis possuem mais de 1 elemento minimal/maximal, e isto explica porque eles não podem ser encontrados pelo algoritmo 1-relaxado. O resultado ótimo ($\tilde{\Delta} = 9$) é mostrado na Fig. 7.14(c). Note que o algoritmo 2-relaxado é capaz de encontrar todos os conjuntos viáveis, isto é, $k = 2$ seria suficiente, neste caso, para se calcular a partição ótima de \mathcal{Q} .

Seja F um conjunto k -viável sendo removido de \mathcal{Q} pelo algoritmo k -relaxado. Como o algoritmo k -relaxado não examina subconjuntos com mais de k elementos minimais/maximais, se existir pelo menos um subconjunto com mais de k 1-elementos minimais em F e se F é um conjunto U - k -viável, ou se existirem subconjuntos com mais de k 0-elementos maximais em F e se F for um conjunto L - k -viável, então existe a possibilidade de que conjuntos viáveis (no sentido estrito) não serem encontrados. Note, entretanto, que a existência de tais subconjuntos não necessariamente implica que os mesmos são viáveis (no sentido estrito).

Em alguns casos, mesmo quando o algoritmo relaxado perde (deixa de identificar) um conjunto viável, o resultado final pode ser ótimo. Consideramos o conjunto violador \mathcal{Q} da Fig. 7.15(a) para ilustrar um caso desses. Se o algoritmo 1-relaxado for aplicado sobre \mathcal{Q} , então dois conjuntos 1-viáveis, a saber $U[\{i\}]$ e $U[\{k\}]$ (o segundo em $\mathcal{Q} \setminus U[\{i\}]$), serão encontrados. Novamente, $U[\{i\}]$ não é um conjunto viável no sentido estrito. Apesar disso, o resultado final, mostrado na Fig. 7.15(b),

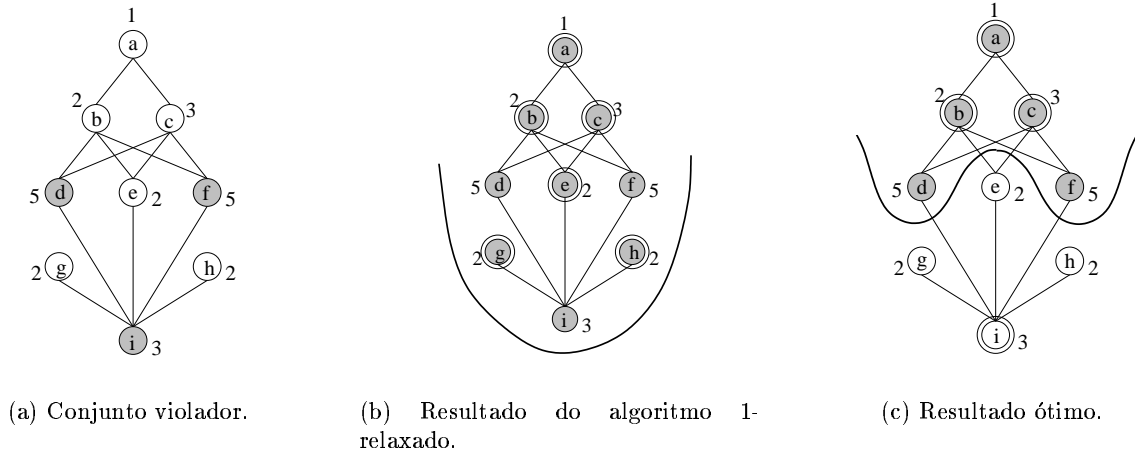


Figura 7.14: Resultado do algoritmo relaxado não é ótimo.

é ótimo (compare com o resultado do algoritmo 2-relaxado na Fig. 7.15(c)). Este exemplo mostra que, contanto que nenhum elemento seja removido para a parte errada da partição, existem chances de que o resultado final seja ótimo, mesmo que durante o processo alguns conjuntos viáveis no sentido estrito tenham sido perdidos.

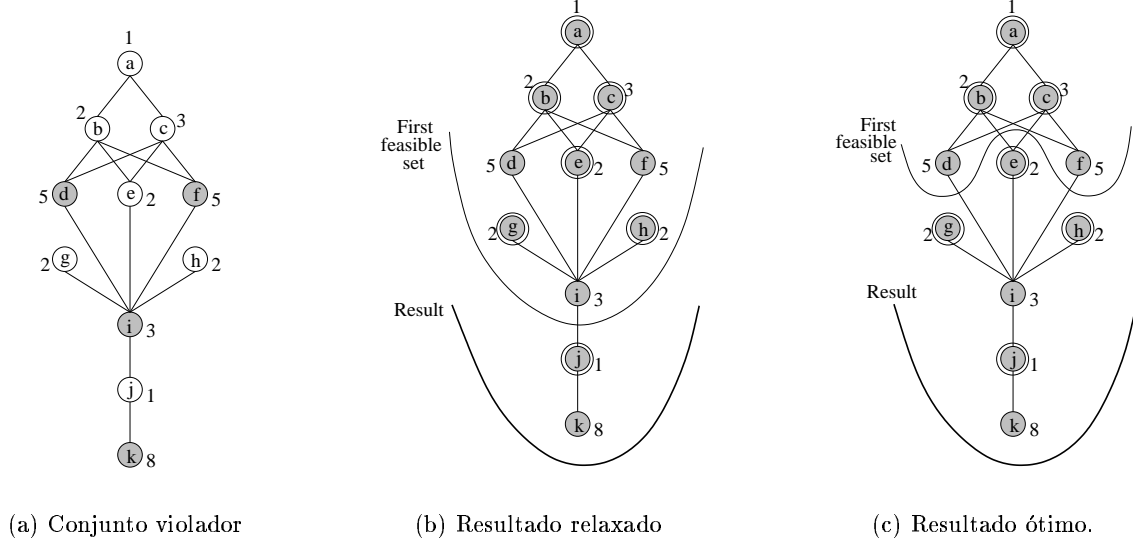


Figura 7.15: Resultado do algoritmo relaxado é ótimo.

Seja Q o conjunto violador mostrado na Fig. 7.16(a). O algoritmo 1-relaxado encontra apenas um conjunto 1-viável, $L[\{e\}]$ (Fig. 7.16(b)). Não existem conjuntos 1-viáveis no restante do conjunto violador $\{a, b, c, d, f\}$ (Fig. 7.16(c)). Em termos gerais, se em algum momento durante o processamento o algoritmo k -relaxado não encontra nenhum conjunto k -viável no conjunto violador restante, então esta porção restante não poderá ser processada. Tais porções serão denominadas *porções irreduzíveis*.

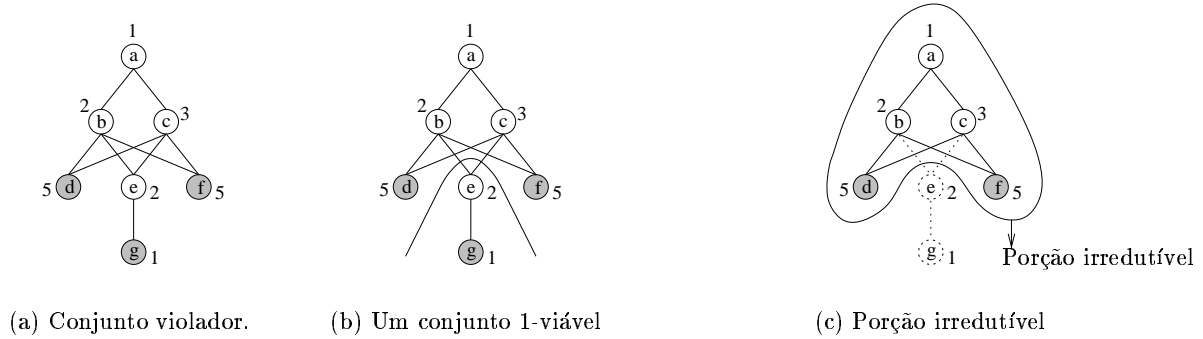


Figura 7.16: Exemplo de uma porção irredutível

Para processar uma porção irredutível, digamos Q' , uma estratégia simples consiste em calcular $\tilde{\Delta}(Q', \emptyset)$ e $\tilde{\Delta}(\emptyset, Q')$, e então remover Q' para o conjunto superior caso $\tilde{\Delta}(Q', \emptyset) < \tilde{\Delta}(\emptyset, Q')$ ou para o conjunto inferior em caso contrário. Uma outra alternativa, quando Q' é relativamente pequeno, seria aplicar o método da força-bruta, i.e., testar todas as possíveis partições válidas de Q' e escolher uma com o menor aumento de erro. O algoritmo OVP relaxado é apresentado a seguir :

Entrada: Um conjunto violador Q ; custos de chaveamento $c(X)$, $\forall X \in Q$, fator de relaxamento k , $k > 0$

Saída: Partição ótima (L, U) de Q .

1. Faça $U \leftarrow \emptyset$ e $L \leftarrow \emptyset$.

2. Se Q é vazio, então devolva (L, U) e pare.

3. Procure um conjunto k -viável F em Q .

• Se F for encontrado, e se ele é U -viável, então faça $U \leftarrow U \cup F$; se F é L -viável, então faça $L \leftarrow L \cup F$. Faça $Q \leftarrow Q \setminus F$.

• Se F não for encontrado, calcule $\tilde{\Delta}(Q, \emptyset)$ e $\tilde{\Delta}(\emptyset, Q)$. Se $\tilde{\Delta}(Q, \emptyset) \leq \tilde{\Delta}(\emptyset, Q)$ então faça $L \leftarrow L \cup Q$, senão faça $U \leftarrow U \cup Q$. Faça $Q \leftarrow \emptyset$.

Volte ao passo 2.

Algoritmo 7.2 Algoritmo OVP relaxado.

Suponha uma situação na qual nenhum conjunto viável (no sentido estrito) foi perdido, até o momento em que uma porção irredutível é encontrada. Mesmo após o processamento da porção irredutível conforme descrito acima, o resultado final pode ser ótimo. Considere a porção irredutível Q' mostrada na Fig. 7.17(a) (nenhum de seus subconjuntos $L[\{a\}]$, $L[\{b\}]$, ou $U[\{c\}]$ são 1-viáveis.) O algoritmo 1-relaxado quando aplicado sobre Q' , calcula $\tilde{\Delta}(Q', \emptyset) = 4$ e $\tilde{\Delta}(\emptyset, Q') = 6$ e decide remover Q' para o conjunto inferior. Como $L[\{a, b\}] = Q'$ é um conjunto viável (no sentido estrito) ele seria removido para o conjunto inferior pelo algoritmo não relaxado também (Fig. 7.17(b).)

Por outro lado, a mesma estratégia pode produzir resultados não-ótimos, como mostramos no seguinte exemplo. Seja Q' a porção irredutível mostrada na Fig. 7.18(a). Então, $\tilde{\Delta}(Q', \emptyset) = 15$ e



Figura 7.17: Porção irredutível é processada corretamente.

$\tilde{\Delta}(\emptyset, Q') = 13$. Logo, Q' será removido para a parte superior conforme ilustrado na Fig. 7.18(b). Entretanto, a solução ótima ($\tilde{\Delta} = 11$) é ilustrada na Fig. 7.18(c).

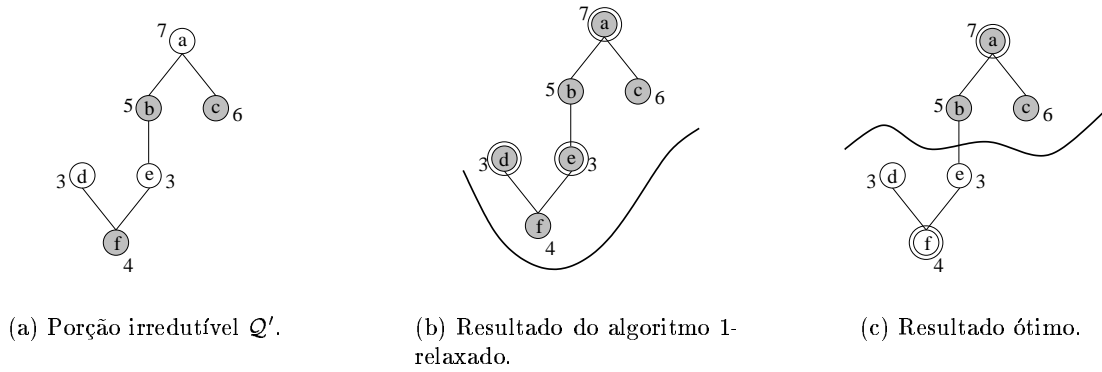


Figura 7.18: Porção irredutível é processado incorretamente.

Em resumo, o relaxamento pode produzir resultados não-ótimos devido aos conjuntos viáveis que não são detectados, ou devido ao processamento incorreto da porção irredutível. Entretanto, deve-se manter em mente que a existência de tais subconjuntos não necessariamente implica que o resultado é sub-ótimo. Estas situações nas quais existem porções irredutíveis, ou nas quais subconjuntos viáveis podem não ter sido detectados, serão denominados *casos incertos*. A ausência de casos incertos implica que o resultado é ótimo.

7.3.7 Limitantes para o Aumento de Erro

O resultado do algoritmo relaxado pode ser sub-ótimo, conforme mostramos na seção anterior. Nesta seção apresentamos limitantes para o aumento de erro do operador projetado em relação ao operador crescente ótimo. Seja $\tilde{\Delta}$ o aumento de erro do operador crescente ótimo (aquele que corresponde à partição ótima de Q). Então,

$$0 \leq \tilde{\Delta} \leq \min\{\tilde{\Delta}(Q, \emptyset), \tilde{\Delta}(\emptyset, Q)\}$$

Agora, sejam U' , L' o conjunto de elementos na parte superior e inferior, respectivamente, quando o algoritmo depara com um caso incerto pela primeira vez durante o processamento de Q . Nesta fase do processamento, sabe-se que os elementos de U' e L' são, respectivamente, elementos das partes superior e inferior da partição ótima. O aumento de erro devido a U' e L' são dados, respectivamente, por

$$c^+(U') = \sum_{X \in U'(0)} c(X) \quad \text{e} \quad c^-(L') = \sum_{X \in L'(1)} c(X)$$

Portanto, o aumento de erro no momento em que o algoritmo depara com um caso incerto pela primeira vez é dado por $c^+(U') + c^-(L')$, o que constitui um limitante inferior para o verdadeiro valor de $\tilde{\Delta}$, i.e.,

$$0 \leq c^+(U') + c^-(L') \leq \tilde{\Delta}$$

Por outro lado, denote por $\tilde{\Delta}_k$ o aumento de erro da partição final calculada pelo algoritmo. Então, $\tilde{\Delta}_k$ é menor ou igual a $\min\{\tilde{\Delta}(\mathcal{Q}, \emptyset), \tilde{\Delta}(\emptyset, \mathcal{Q})\}$. Portanto, temos a seguinte relação :

$$0 \leq c^+(U') + c^-(L') \leq \tilde{\Delta} \leq \tilde{\Delta}_k \leq \min\{\tilde{\Delta}(\mathcal{Q}, \emptyset), \tilde{\Delta}(\emptyset, \mathcal{Q})\} \quad (7.8)$$

Conforme já dissemos anteriormente, se o algoritmo não depara com um caso incerto, então o resultado por ele produzido é ótimo e, neste caso, teremos $\tilde{\Delta}_k = \tilde{\Delta}$. Veremos nos resultados experimentais que muitas vezes $\tilde{\Delta}_k$ é muito próximo de $\tilde{\Delta}$.

7.3.8 Resultados Experimentais

Testamos o uso de diferentes valores para k . Os experimentos foram conduzidos em um computador Pentium II, 300 MHz, usando diferentes modelos de imagens (A,B,C, e D), ilustrados nas Figs. 7.19 a 7.22.

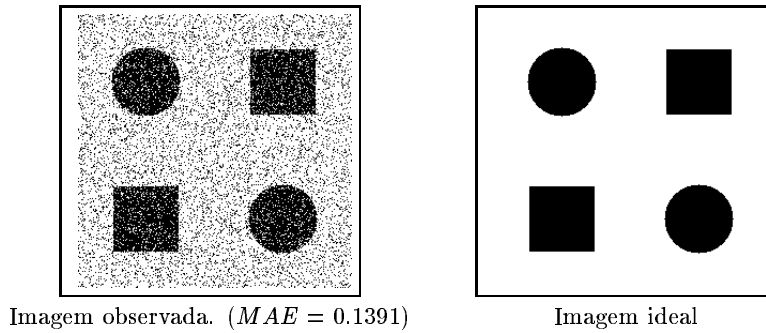


Figura 7.19: Grupo A : 15% de ruído sal-e-pimenta

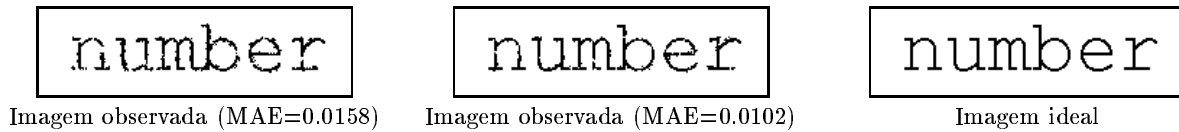


Figura 7.20: Grupo B: Ruído de borda, diferente densidades.

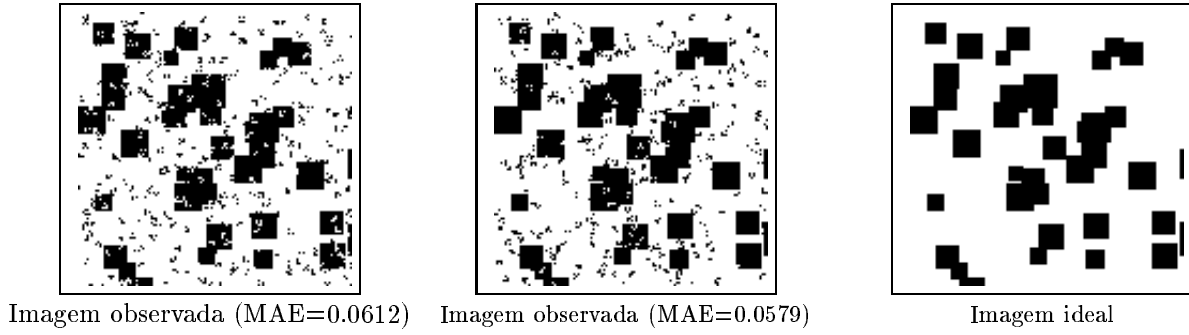


Figura 7.21: Grupo C: Modelo Booleano com grãos quadrados e subconjuntos do quadrado 3×3 como ruído.

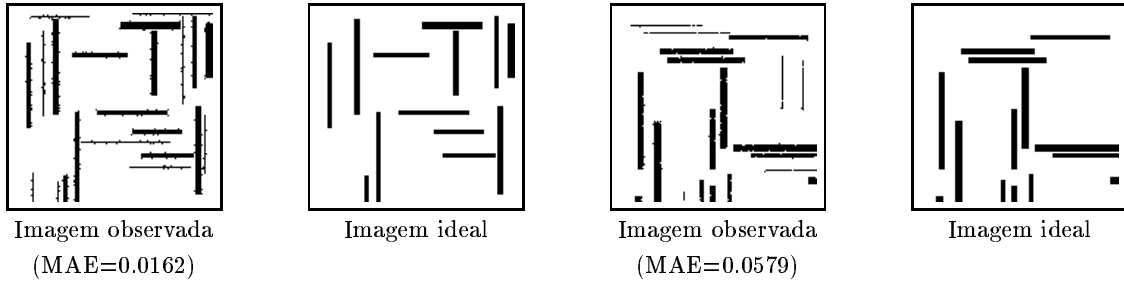


Figura 7.22: Grupo D : Retângulos com ruído de borda, com diferentes densidades.

Algoritmo OVP

Se, durante o processamento, o algoritmo k -relaxado não depara com um caso incerto, então $\tilde{\Delta}_k$ é ótimo. Além disso, se $\tilde{\Delta}_k$ é ótimo, então $\tilde{\Delta}_l = \tilde{\Delta}_k$, para todo $l \geq k$. Note que, mesmo nos casos em que o algoritmo depara com um caso incerto, o resultado pode ainda ser ótimo, conforme discutimos anteriormente.

A tabela 7.1 mostra os resultados relativos a 67 experimentos com a janela $W_{3 \times 3}$. Aqui enfatizamos a relação entre $\tilde{\Delta}_1$, $\tilde{\Delta}_2$ e $\tilde{\Delta}_3$. Estes resultados estão separados em dois grupos principais : os ótimos (aqueles nos quais o algoritmo não deparou com um caso incerto para pelo menos um dos k , $k = 1, 2, 3$) e os *incertos* (aqueles nos quais o algoritmo deparou com um caso incerto para $k = 1, 2$, e 3). Cada um desses dois grupos estão subdivididos em subcasos. Para a classe dos ótimos, a coluna 2 indica qual o menor valor de k para o qual o algoritmo não deparou um caso incerto. Isto significa que o resultado ótimo foi encontrado e que o algoritmo, se executado com k maior, iria produzir o mesmo resultado. A coluna 3 indica quantas vezes cada um desses casos foi observado. A coluna 4 indica a relação entre $\tilde{\Delta}_1$, $\tilde{\Delta}_2$ e $\tilde{\Delta}_3$, e a coluna 5 indica quantas vezes cada uma das relações indicadas foi observada. Finalmente, as duas últimas colunas especificam o modelo e a quantidade de imagens utilizadas nos experimentos. A tabela 7.2 mostra o mesmo para 69 experimentos com a janela W_{13} .

A tabela 7.3 fornece maiores detalhes dos 17 casos, da janela W_{13} , onde $\Delta_1 > \Delta_3$. A coluna 1 mostra o MAE do operador ótimo ψ_{opt} e as colunas 2 e 3 mostram, respectivamente, o MAE do operador crescente obtido usando $k = 3$ e $k = 1$. Observe que a variação do MAE entre estes dois operadores não é significativa, como indicado na coluna 4.

Grupo	Caso	Ocorr.	subcaso	Ocorr.	Modelo	Ocorr.
ótimo	$k = 1$	34	$\Delta_1 (= \Delta_2 = \Delta_3)$	34	A	1
					C	3
					D	30
	$k = 2$	11	$\Delta_1 = \Delta_2 (= \Delta_3)$	11	D	11
	$k = 3$	8	$\Delta_1 = \Delta_2 = \Delta_3$	6	D	6
			$\Delta_1 > \Delta_2 = \Delta_3$	2	B	1
					D	1
incerto		14	$\Delta_1 = \Delta_2 = \Delta_3$	13	B	2
					D	11
			$\Delta_1 > \Delta_2 = \Delta_3$	1	B	1
total		67		67		67

Tabela 7.1: Desempenho do algoritmo OVP: janela $W_{3 \times 3}$.

Grupo	k	Ocorr.	subcaso	Ocorr.	Modelo	Ocorr.
ótimo	$k = 1$	22	$\Delta_1 (= \Delta_2 = \Delta_3)$	22	A	1
					D	21
	$k = 2$	8	$\Delta_1 = \Delta_2 (= \Delta_3)$	7	C	1
					D	6
			$\Delta_1 > \Delta_2 (= \Delta_3)$	1	A	1
	$k = 3$	8	$\Delta_1 = \Delta_2 = \Delta_3$	6	D	6
			$\Delta_1 > \Delta_2 = \Delta_3$	2	C	2
incerto		31	$\Delta_1 = \Delta_2 = \Delta_3$	17	B	2
					D	15
			$\Delta_1 > \Delta_2 = \Delta_3$	7	B	1
					D	6
			$\Delta_1 = \Delta_2 > \Delta_3$	1	D	1
			$\Delta_1 > \Delta_2 > \Delta_3$	6	B	1
					D	5
total		69		69		69

Tabela 7.2: Desempenho do Algoritmo OVP: janela W_{13} .

$MAE\langle\psi_{\text{opt}}\rangle$	$MAE\langle\psi_{\text{opt}}\rangle + \Delta_3$	$MAE\langle\psi_{\text{opt}}\rangle + \Delta_1$	$\Delta_1 - \Delta_3$
5.6226e-03	6.0367e-03	6.0368e-03	1e-07
7.6559e-03	8.1047e-03	8.1048e-03	1e-07
5.2408e-04	7.8244e-04	7.8259e-04	1.5e-07
4.9300e-03	5.3283e-04	5.3265e-04	1.8e-07
3.9749e-04	4.4473e-04	4.4492e-04	1.9e-07
5.1621e-03	5.6847e-03	5.6849e-03	2e-07
7.1254e-03	7.6549e-03	7.6551e-03	2e-07
1.8423e-04	2.7528e-04	2.7552e-04	2.4e-07
6.2505e-03	8.9047e-03	8.9051e-03	4e-07
1.5442e-03	1.9622e-03	1.9629e-03	7e-07
3.1494e-03	3.2061e-03	3.2069e-03	8e-07
2.1721e-03	2.7239e-03	2.7248e-03	9e-07
1.9718e-03	2.2409e-03	2.2425e-03	1.6e-06
2.1792e-03	2.7267e-03	2.7289e-03	2.2e-06
5.0711e-04	9.9452e-04	9.9836e-04	3.84e-06
5.2177e-04	9.9219e-04	9.9659e-04	4.4e-06
1.0535e-02	1.4451e-02	1.4465e-02	1.4e-05

Tabela 7.3: Aumento de Erro: fator de relaxamento $k = 1$ e $k = 3$.

A tabela 7.4 mostra o tempo de processamento para variadas janelas e diferentes tamanhos do conjunto violador. O tempo é dado em segundos. Para simplificar, eles foram aproximados para o valor inteiro não nulo mais próximo. Quando o número é marcado pelo símbolo *, significa que o resultado correspondente é ótimo, enquanto o símbolo \times indica que o tempo não foi calculado para aquele caso. O tempo de processamento é o tempo gasto para a) o cálculo do conjunto violador, b) construção de um grafo correspondendo ao conjunto violador, c) aplicação do algoritmo OVP, d) cálculo dos elementos minimais de $\psi_{opt_A}(1)$, e) I/O. Cabe notar que cada um dos experimentos com a janela $W_{3 \times 3}$ levaram menos de 0.27 segundos.

Tamanho da janela	Exemplos	Conjunto violador	Arestas	Base	Tempo ± 0.5 (em segundos)		
					$k = 1$	$k = 2$	$k = 3$
13	3671	622	1516	41	1	1	1
	8180	902	1084	596	2	2	2*
	8185	2452	8778	215	3	6	96
	8185	70	42	831	1*	\times	\times
	7892	3090	12718	165	3	8	40
	5864	4492	22762	69	4	15	1125
16	6605	5020	26368	81	5	26	1483
	47648	5277	20139	1856	66	67	72
17	56706	14063	13209	1485	223	393	18621
	74253	7066	10532	2820	137	148	161
	68011	18656	52716	1995	397	1028	\times
25	95439	24539	14549	2153	779	2595	\times
	52615	5101	73541	212	87	88*	\times
	290892	5376	97878	5183	288	289	311
	74914	10090	42771	198	203	204	204
	92565	10899	45789	8	449	465	1571
	597965	19885	68891	9384	3749	4075	5830
	368717	87732	808285	6151	11459	66518	\times
49	836334	198168	2040243	8375	16 horas	7 dias	\times
	1173140	445	426	37376	678	671*	\times

Tabela 7.4: Tempo de processamento do algoritmo OVP.

Algoritmo OVP versus o Algoritmo Baseado na Representação MAE

Comparamos a seguir o algoritmo OVP com outro algoritmo cuja implementação é baseada no teorema de representação do MAE. Neste segundo, utilizamos uma implementação já citada em estudos anteriores, que consiste dos seguintes passos :

1. Calcular o MAE para cada um dos 2^n elementos estruturantes (ou seja, cada elemento de $\mathcal{P}(W)$) (Eq. 7.1.)
2. Selecionar 100 elementos estruturantes com os menores valores MAE para construir a biblioteca de primeira-ordem.

3. Calcular o MAE para todas as combinações de no máximo m elementos estruturantes desta biblioteca (m é o tamanho máximo da base), usando um algoritmo recursivo que implementa a Eq. 7.2.

Implementações mais eficientes da Eq. 7.2, que não são recursivas, são possíveis. Por exemplo, um algoritmo baseado em programação dinâmica poderia utilizar o fato de que o MAE de um operador com m erosões pode ser computado a partir do MAE de dois operadores com $(m - 1)$ erosões e mais o MAE de um elemento estruturante [111]. Porém, uma dificuldade nesta abordagem é a quantidade de dados que precisam ser armazenados, uma vez que a quantidade de operadores consistindo de $(m - 1)$ erosões é enorme. Mesmo que o espaço para armazenamento seja suficiente, outra dificuldade está relacionado com a forma de armazenamento: os MAEs de todos os operadores constituídos por $(m - 1)$ erosões precisam ser armazenados de tal maneira que os mesmos possam ser localizados facilmente.

A tabela 7.5 mostra algumas comparações entre o desempenho do algoritmo OVP e o algoritmo baseado na representação MAE, cuja implementação foi descrita acima. Para o algoritmo baseado na representação MAE, o tamanho da base foi restrito conforme indicado na coluna *Restrição de tamanho* da tabela. A superioridade do algoritmo OVP é evidente. O algoritmo OVP foi utilizado para janelas de 25 e de 49 pontos, com o tamanho da base na ordem de milhares. Mesmo em aplicações industriais, para aplicar o algoritmo baseado na representação MAE, o tamanho da base é limitado a um número pequeno de elementos estruturantes selecionados de uma biblioteca também relativamente pequena [112].

Tamanho da janela	OVP			representação MAE			
	Tamanho da base	Δ	tempo	Restrição de tamanho	Tamanho da base	Δ	tempo
9	4	2.0488e-04	1	6	4	2.0488e-04	2202
	4	1.2631e-03	1	6	4	1.2631e-03	2190
	10	4.5525e-05	1	6	6	3.4783e-04	2499
	22	2.2020e-06	1	6	6	2.5304e-04	4541
	100	0.0	1	6	6	1.1874e-02	8433
	100	0.0	1	8	8	1.3307e-02	13.7 dias
	24	6.8646e-04	1	6	6	1.0895e-03	4158
	34	0.0	1	6	6	4.7806e-04	7047
13	276	1.3440e-04	1	6	4	2.1947e-02	19152
	831	6.8237e-06	1	6	5	1.8189e-02	19192

Tabela 7.5: Algoritmo OVP contra algoritmo baseado na representação MAE.

7.4 Distribuições a Priori para as Probabilidades Condicionais

Para janelas relativamente grandes, as estimativas das probabilidades condicionais $p_X = P(y = 1|X)$ e das probabilidades $P(X)$ geralmente não são muito precisas pois a quantidade de dados a partir do qual elas são estimadas é pequena. Para determinar o operador ótimo, a precisão de \hat{p}_X é fundamental. Uma abordagem proposta para melhorar a estimativa destas probabilidades considera que p_X são variáveis aleatórias que seguem uma distribuição a priori $f(p_X)$ [47]. Dada uma amostra

de treinamento, a distribuição a posteriori de p_X é denotada $f(p_X | \hat{p}_X)$, onde \hat{p}_X é a probabilidade condicional estimada a partir desta amostra.

Estas distribuições a priori são diferentes para diferentes observações $X \in \mathcal{P}(W)$. Para simplificar a notação, eliminamos X daqui em diante (assim, p significa p_X , para um dado X). Seja m a quantidade de vezes que a configuração X foi observada na amostra de treinamento, das quais u vezes associada ao valor 1. Supondo a estimação usual, $\hat{p} = u/m$, a densidade condicional de p dado \hat{p} , é dada por

$$f(p | \hat{p}) = \frac{f(p, \hat{p})}{f(\hat{p})} = \frac{f(\hat{p} | p) f(p)}{\int_0^1 f(\hat{p} | p) f(p) dp} \quad (7.9)$$

O operador ótimo pode levar em consideração a distribuição a priori juntamente com os dados de treinamento. Como $f(p | \hat{p})$ é a distribuição de p dado \hat{p} , então o operador ótimo, a ser denotado ψ_{pri} é dado por

$$\psi_{pri}(X) = \begin{cases} 1, & \text{se } E[p | \hat{p}] > 0.5 \\ 0, & \text{caso contrário.} \end{cases} \quad (7.10)$$

onde $E[p | \hat{p}]$ é o valor esperado da distribuição $f(p | \hat{p})$.

7.4.1 Custos de Chaveamento a Partir de Distribuições a Priori

Propomos aqui a utilização de distribuições a priori das probabilidades condicionais para estimar os custos de chaveamento. Estimativas mais precisas das probabilidades condicionais implicam custos de chaveamento mais precisos e, conseqüentemente, operadores mais precisos.

Se $E[p | \hat{p}] \leq 0.5$ e $\psi_{pri}(X)$ é chaveado de 0 para 1, para cada valor $p \leq 0.5$ há um aumento de erro de $|2p - 1| P(X)$, e para cada valor $p > 0.5$ há um decréscimo de erro de $|2p - 1| P(X)$. Logo, o custo de chaveamento total relativo a X , quando $\psi_{pri}(X)$ é chaveado de 0 para 1, é dado por

$$c_1(X) = \left[\int_0^{0.5} |2p - 1| f(p | \hat{p}) dp - \int_{0.5}^1 |2p - 1| f(p | \hat{p}) dp \right] P(X) \quad (7.11)$$

Analogamente, o custo de chaveamento quando $\psi_{pri}(X)$ é chaveado de 1 para 0 é dado por :

$$c_2(X) = \left[\int_{0.5}^1 |2p - 1| f(p | \hat{p}) dp - \int_0^{0.5} |2p - 1| f(p | \hat{p}) dp \right] P(X) \quad (7.12)$$

A expressão 7.11 pode ser simplificada conforme mostramos a seguir :

$$\begin{aligned} c_1(X) &= \left[\int_0^{0.5} |2p - 1| f(p | \hat{p}) dp - \int_{0.5}^1 |2p - 1| f(p | \hat{p}) dp \right] P(X) \\ &= \left[\int_0^{0.5} (1 - 2p) f(p | \hat{p}) dp - \int_{0.5}^1 (2p - 1) f(p | \hat{p}) dp \right] P(X) \\ &= \left[\int_0^{0.5} f(p | \hat{p}) dp - 2 \int_0^{0.5} p f(p | \hat{p}) dp - 2 \int_{0.5}^1 p f(p | \hat{p}) dp + \int_{0.5}^1 f(p | \hat{p}) dp \right] P(X) \\ &= \left[\int_0^{0.5} f(p | \hat{p}) dp + \int_{0.5}^1 f(p | \hat{p}) dp - 2 \left(\int_0^{0.5} p f(p | \hat{p}) dp + \int_{0.5}^1 p f(p | \hat{p}) dp \right) \right] P(X) \\ &= \left[1 - 2 \int_0^1 p f(p | \hat{p}) dp \right] P(X) \\ &= \left[1 - 2 E[p | \hat{p}] \right] P(X) \end{aligned} \quad (7.13)$$

De forma análoga, a Eq. 7.12 também pode ser simplificada :

$$c_2(X) = [2 E[p|\hat{p}] - 1] P(X) \quad (7.14)$$

Uma vez que na Eq. 7.13 temos que $E[p|\hat{p}] \leq 0.5$ e na Eq. 7.14 temos que $E[p|\hat{p}] > 0.5$, ambos podem ser expressos como $c(X) = |2 E[p|\hat{p}] - 1| P(X)$.

Levando em conta os resultados obtidos até agora, o cálculo do custo de chaveamento depende somente de $E[p|\hat{p}]$ e $P(X)$. Daqui em diante utilizamos uma particular distribuição a priori, a exemplo do que é feito em [47]. A distribuição considerada é a distribuição beta, que possui dois parâmetros $\alpha, \beta > 0$:

$$f(p) = \begin{cases} \frac{1}{B(\alpha, \beta)} p^{\alpha-1} (1-p)^{\beta-1}, & \text{if } 0 \leq p \leq 1 \\ 0, & \text{caso contrário.} \end{cases}$$

onde $B(\cdot, \cdot)$ é a função beta definida por

$$B(\alpha, \beta) = \int_0^1 t^{\alpha-1} (1-t)^{\beta-1} dt$$

Supondo que $f(p)$ é uma distribuição beta com parâmetros α e β e dados $m = m_X$, $u = u_X$ e $\hat{p} = m/u$, a equação 7.9 pode ser reescrita como :

$$\begin{aligned} f(p|\hat{p}) &= \frac{\binom{m}{u} p^u (1-p)^{m-u} \frac{p^{\alpha-1} (1-p)^{\beta-1}}{B(\alpha, \beta)}}{\int_0^1 \binom{m}{u} p^u (1-p)^{m-u} \frac{p^{\alpha-1} (1-p)^{\beta-1}}{B(\alpha, \beta)} dp} \\ &= \frac{\frac{\binom{m}{u}}{B(\alpha, \beta)} p^{\alpha+u-1} (1-p)^{\beta+m-u-1}}{\frac{\binom{m}{u}}{B(\alpha, \beta)} \int_0^1 p^{\alpha+u-1} (1-p)^{\beta+m-u-1} dp} \\ &= \frac{p^{\alpha+u-1} (1-p)^{\beta+m-u-1}}{B(\alpha+u, \beta+m-u)} \end{aligned}$$

que é a distribuição beta a posteriori (com parâmetros $\alpha' = \alpha + u$ e $\beta' = \beta + m - u$). Se a distribuição a priori de X não é conhecida, podemos usar a distribuição uniforme (i.e., $\alpha = \beta = 1$, e neste caso a equação 7.10 é equivalente a equação 3.4, o caso sem distribuições a priori).

O valor esperado de $p|\hat{p}$, $E[p|\hat{p}]$, é dado por

$$E[p|\hat{p}] = \frac{\alpha'}{\alpha' + \beta'}$$

Logo o custo de chaveamento é dado por :

$$c(X) = \left| 2 \frac{\alpha'}{\alpha' + \beta'} - 1 \right| P(x). \quad (7.15)$$

7.4.2 Resultados Experimentais

No exemplo a seguir, consideramos ruído de borda em imagens de texto. Supomos que o ruído de borda tem intensidade parametrizada por δ , o qual segue uma distribuição conforme ilustrada na

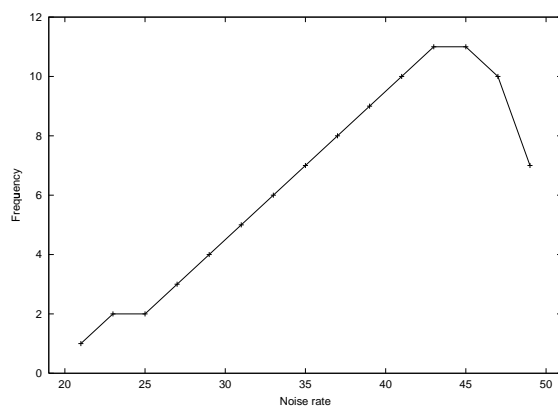


Figura 7.23: Distribuição de δ (parâmetro da intensidade de ruído).

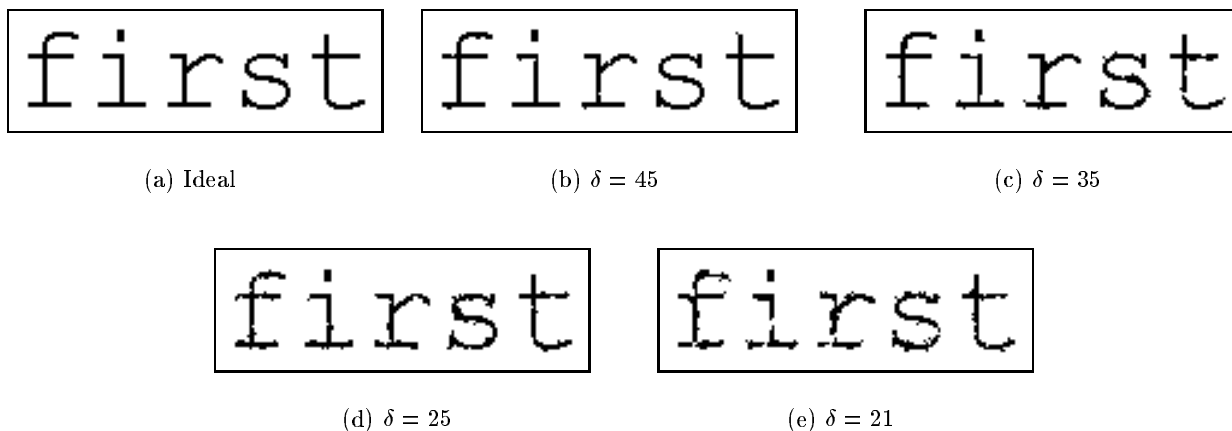


Figura 7.24: Exemplos de ruído de borda.

figura 7.23. A intensidade de ruído é inversamente proporcional ao valor de δ . A figura 7.24 mostra, respectivamente, parte de uma imagem ideal, e simulação de ruído para $\delta = 45, 35, 25$ e 21 .

Se consideramos uma amostra de imagens com ruído de borda seguindo esta distribuição, imagens com $\delta = 21$ (bastante ruidosa) são observadas com muito menor frequência do que imagens com $\delta = 45$, por exemplo.

Como escolher uma distribuição a priori é uma questão importante, muitas vezes polêmica, que não será discutida neste trabalho. O objetivo no momento é apenas mostrar que o uso de distribuições a priori adequadas é útil para obter operadores precisos, principalmente quando a quantidade de exemplos de treinamento é pequena.

Para obter uma distribuição a priori para cada X , foram obtidas diversas estimações para p_X a partir de uma amostra de treinamento, cujas imagens com densidades de ruído diferentes aparecem de acordo com a proporção da figura 7.23. Para cada m imagens com densidade de ruído $\delta = 21$ foram consideradas $2m$ imagens com densidade $\delta = 23$ e $\delta = 25$, $3m$ imagens com densidade $\delta = 27$ e assim por diante. Para cada grupo de m imagens foi estimado um valor de p_X , e em seguida foi calculada uma distribuição beta cujas realizações mais se aproximam destes dados, utilizando a

técnica de máxima verossimilhança.

Três distribuições a priori foram obtidas a partir de valores distintos para m . A priori 1 foi obtida a partir de $m = 2$, a priori 2 a partir de $m = 10$ e a priori 3 a partir de $m = 50$. Comparamos o desempenho de operadores crescentes projetados utilizando estas prioris em relação ao desempenho de operadores projetados sem o uso de prioris para particulares valores de δ . Utilizamos a janela $W_{5 \times 3}$, e o algoritmo OVP proposto no capítulo 7, com o operador ótimo dado pela equação 7.10 e com os custos de chaveamento conforme equação 7.15.

Para $\delta = 45$ (pouco ruído), o uso da priori 3 resulta em operadores bons a tal ponto que o treinamento com número crescente de exemplos não tem efeito algum. A priori 2 também resulta em operadores bons. A priori 1 resulta em operadores que funcionam melhor que o caso sem priori, e melhora ligeiramente à medida que o exemplos de treinamento aumentam. O treinamento sem priori, aproxima-se do da priori 1 quando utiliza uma grande quantidade de exemplos. No limite, isto é, infinitos exemplos de treinamento, todas as 4 curvas devem convergir para o mesmo erro (erro mínimo).

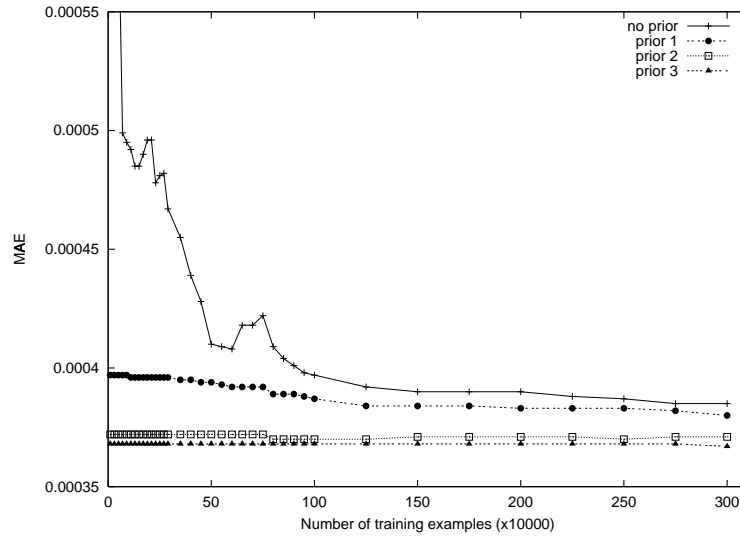


Figura 7.25: Curva de erro para $\delta = 45$.

Para o caso $\delta = 35$, a priori 3 também funciona bem de forma que o treinamento quase não tem efeito. A melhora dos operadores resultantes da priori 2, à medida que aumenta a quantidade de exemplos de treinamento, é mais evidente que para o caso $\delta = 45$. A priori 1 começa com erro inicial menor que o caso sem priori, mas este alcança a priori 1 e passa a ser melhor quando a quantidade de exemplos de treinamento é relativamente grande.

Para o caso $\delta = 25$, a priori 3 resulta em operadores que melhoram com o aumento da quantidade de exemplos de treinamento. A melhora é mais evidente para a priori 2. A priori 1 não tem bom desempenho no começo e rapidamente é alcançada pelo caso sem priori. O fato do caso sem priori funcionar melhor que a priori 1 mostra que o uso da priori está sendo prejudicial. Isto ilustra um exemplo de uso de priori não adequada.

Para o caso $\delta = 21$, a priori 1 ajuda basicamente quando não há treinamento algum. Rapidamente é ultrapassado pelo caso sem priori. Para quantidade grande de exemplos de treinamento, tanto a

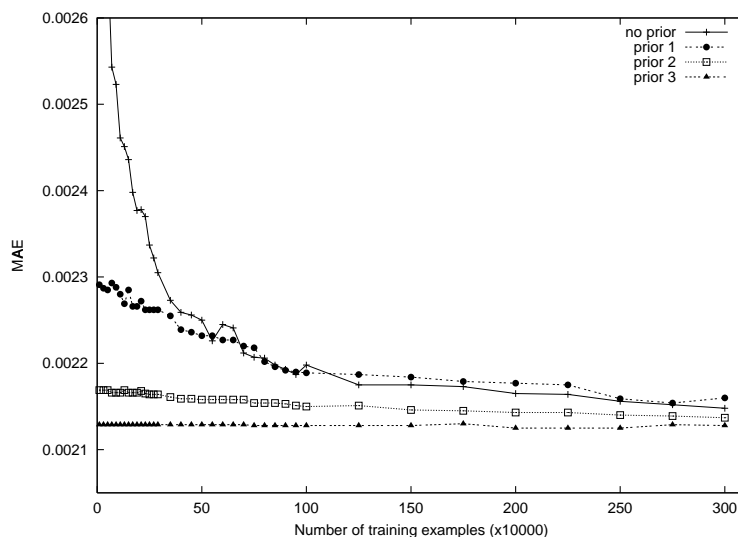


Figura 7.26: Curva de erro para $\delta = 35$.

priori 2 como a priori 3 são alcançadas pelo sem priori.

Em todos os casos, a priori 3 é superior a priori 2, que por sua vez é superior a priori 1, como esperado. No entanto, para alta intensidade de ruído ($\delta = 21, 25$), a partir de um certo momento, o uso de priori começa a ser prejudicial. Isto tem explicação no fato de que a priori foi obtida de um conjunto de imagens onde predominou ruídos de intensidade baixa.

7.5 Comentários

Uma das características interessantes do algoritmo proposto neste capítulo é que ele manipula apenas os elementos de $\mathcal{P}(W)$ que foram observados nas imagens de treinamento, enquanto as demais técnicas conhecidas manipulam todo o espaço $\mathcal{P}(W)$. Esta característica permite que operadores sobre janelas relativamente grandes sejam projetados.

Além disso, ele é extremamente eficiente nos casos em que o tamanho do conjunto violador é pequeno. Heurísticas, ou fatos baseados nos conhecimentos sobre o problema que desejamos resolver, podem ser utilizados para reduzir o tamanho do conjunto violador e, conseqüentemente, o tempo de treinamento.

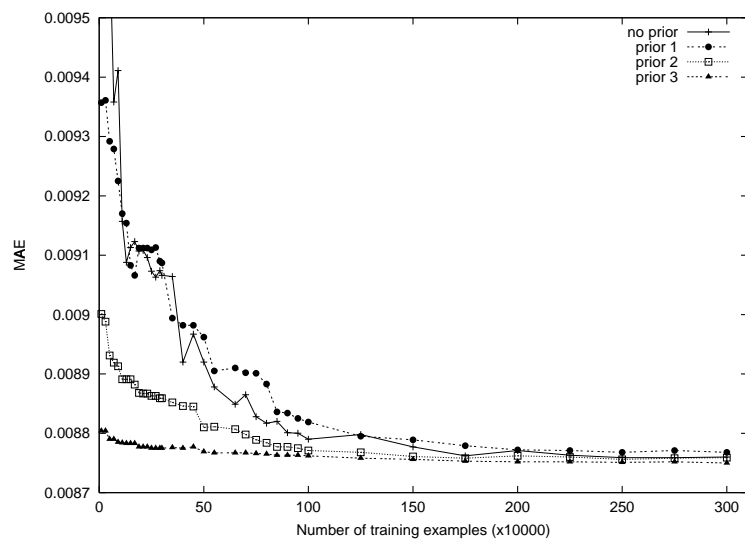


Figura 7.27: Curva de erro para $\delta = 25$.

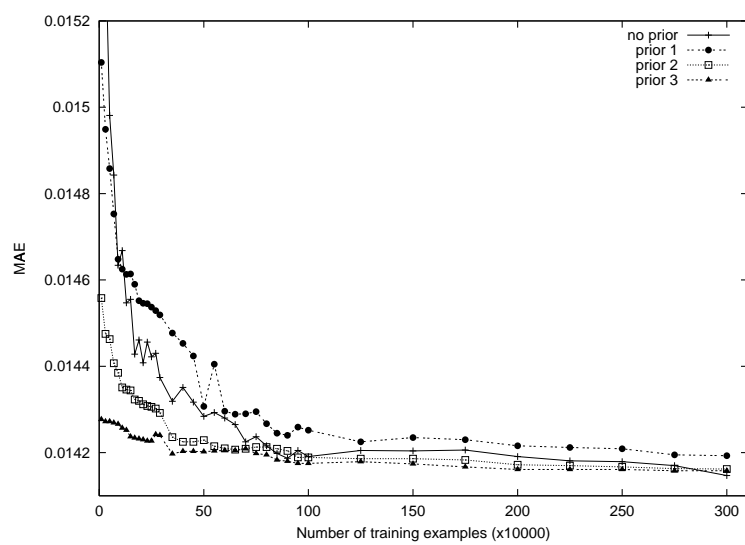


Figura 7.28: Curva de erro para $\delta = 21$.

Capítulo 8

Filtros “Stack”

Filtros “Stack”¹ formam uma classe de operadores sobre imagens em níveis de cinza que podem ser implementados via operadores sobre imagens binárias [165]. Um exemplo clássico de filtro stack é o *filtro da mediana*. Pode-se mostrar que a mediana de uma imagem em níveis de cinza é igual à soma das medianas (binárias) de cada um de seus cortes [60]. No contexto de morfologia matemática, os filtros stack aparecem relacionados a uma classe de operadores denominados “flat” [75] e também à classe de operadores que comutam com o threshold [114, 115].

Um fato interessante é que o erro MAE de um filtro stack pode ser expresso como a soma do erro MAE (binário) sobre cada um dos cortes. Isto significa que, para projetar estes operadores, basta projetarmos os operadores binários que o implementam.

Neste capítulo apresentamos (1) filtros stack no contexto de morfologia matemática [114, 115, 75], porém restrito ao domínio das imagens em níveis de cinza com número de níveis finito e discreto, (2) uma prova construtiva da equivalência entre filtros stack e funções Booleanas positivas baseada em alguns resultados conhecidos e (3) revemos a representação de filtros stack em termos de união de erosões. Analisamos filtros stack ótimos segundo o critério MAE e mostramos como e porquê o algoritmo OVP pode ser utilizado para projetar filtros stack ótimos. Por fim, descrevemos um algoritmo para o projeto de filtros stack.

Os operadores binários são representados por letras gregas (Ψ , Φ , ...), conforme no restante do texto. Os operadores de imagens em níveis de cinza são representados pelas mesmas letras, porém com um apóstrofe, como em Ψ' , Φ' , etc.

8.1 Definições e Propriedades

Vimos que dado $E = Z^2$ e $K = \{0, 1, \dots, k\}$, onde k é um inteiro positivo, uma *imagem em níveis de cinza* é uma função f de E em K . O conjunto de todas as imagens em níveis de cinza definidos sobre E é denotado por K^E , e o conjunto K é o conjunto dos valores de níveis de cinza. O conjunto $(E, +)$, onde $+$ é a adição usual de vetores, é um grupo abeliano. A *translação* de $f \in K^E$ por um vetor $z \in E$ é denotado f_z e definido por, $\forall x \in E$, $f_z(x) = f(x - z)$.

Seja $W \subseteq E$ uma janela (conforme utilizada até agora). A *restrição* de $f \in K^E$ por W em torno

¹Optamos por utilizar a grafia em inglês.

de z define uma imagem $f_{-z}|_W$ in K^W dado por $(f_{-z}|_W)(w) = f(w+z)$, para qualquer $w \in W$. Note que se $W = E$, então $f_{-z}|_W = f_{-z}$. Dados $f, g \in K^W$, $f \leq g \Leftrightarrow f(x) \leq g(x), \forall x \in W$.

Recordamos brevemente, por conveniência, algumas definições e resultados:

- o *corte* de uma imagem $f \in K^W$, no nível i , é o conjunto $T_i[f] = \{x \in W : f(x) \geq i\}$;
- a função $1_X \in \{0,1\}^W$ é definida por $1_X(x) = 1 \Leftrightarrow x \in X$, para todo $X \in \mathcal{P}(W)$,
- se $f \in K^W$ é uma imagem em níveis de cinza, $1_{T_i[f]}$ é uma imagem binária;
- se $f \in \{0,1\}^W$ então $1_{T_1[f]} = f$.

Consideramos também a limiarização de um valor $v \in K$. O limiar de v no nível i é definida por

$$T_i(v) = \begin{cases} 1, & \text{se } v \geq i, \\ 0, & \text{se } v < i. \end{cases} \quad (8.1)$$

Note que enquanto $T_i[]$ resulta em um conjunto, $T_i()$ resulta em um valor binário.

Proposição 8.1 (Decomposição threshold [165]) *Qualquer imagem $f \in K^W$ pode ser representada através da soma de uma seqüência de imagens correspondentes aos cortes $T_i[f]$, $i = 1, 2, \dots, k$, i.e.,*

$$f = \sum_{i=1}^k 1_{T_i[f]}. \quad (8.2)$$

■

A figura 8.1 mostra a decomposição threshold de um sinal. Ao lado esquerdo está o sinal e ao lado direito os cortes do sinal nos níveis 1, 2 e 3. Ao se somar os cortes, obtém-se o sinal original.

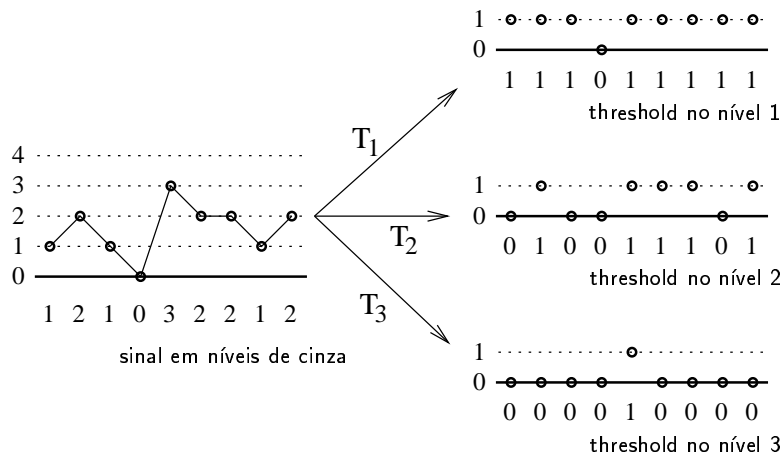


Figura 8.1: Decomposição threshold.

Proposição 8.2 *Seja $W \subseteq E$, $t_1, t_2 \in K$ e $f \in K^W$. Se $t_1 \leq t_2$, então $T_{t_2}[f] \subseteq T_{t_1}[f]$ ou, equivalentemente, $1_{T_{t_2}[f]} \leq 1_{T_{t_1}[f]}$ [33].*

■

Proposição 8.3 Para quaisquer $f, g \in K^W$, $f \leq g \iff T_t[f] \subseteq T_t[g], \forall t \in K$. Equivalentemente, $f \leq g \iff 1_{T_t[f]} \leq 1_{T_t[g]}, \forall t \in K$.

Dem.: (\implies) se $f \leq g$, então $f(x) \leq g(x), \forall x \in W$. Logo, $\{y \in W : f(y) \geq t\} \subseteq \{y \in W : g(y) \geq t\}$ para todo $t \in K$, i.e., $T_t[f] \subseteq T_t[g], \forall t \in K$.

(\impliedby) para qualquer $x \in W$, $x \in T_{f(x)}[f]$. Se $T_t[f] \subseteq T_t[g], \forall t \in K$, então para qualquer $x \in W$, $x \in T_{f(x)}[f] \Rightarrow x \in T_{f(x)}[g] = \{y \in E : g(y) \geq f(x)\}$. Logo, $g(x) \geq f(x)$, para qualquer $x \in W$, i.e., $f \leq g$. ■

Definição 8.4 (Operadores crescentes) Um operador $\Psi' : K^E \rightarrow K^E$ é crescente se e somente se, para quaisquer $f, g \in K^E$, se $f \leq g$, então $\Psi'(f) \leq \Psi'(g)$.

Seja $\Psi' : K^E \rightarrow K^E$ um operador de imagens em níveis de cinza e seja $\Psi : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ um operador definido por, $\forall S \in \mathcal{P}(E)$,

$$\Psi(S) = T_1[\Psi'(1_S)] \quad (8.3)$$

O mapeamento que associa Ψ a Ψ' é denotado por b , isto é, $b(\Psi') = \Psi$.

Definição 8.5 (Operadores que comutam com o threshold) Seja $\Psi' : K^E \rightarrow K^E$ e $\Psi = b(\Psi')$. Dizemos que Ψ' comuta com o threshold sse

$$T_i[\Psi'(f)] = \Psi(T_i[f]) \quad (8.4)$$

para todo $i \in K$ e $f \in K^E$.

Proposição 8.6 (Propriedades de operadores que comutam com o threshold) Se um operador $\Psi' : K^E \rightarrow K^E$ comuta com o threshold então

- (a) tanto Ψ' como $\Psi = b(\Psi')$ são crescentes e
- (b) vale a decomposição

$$\Psi'(f)(x) = \max\{t \in K : x \in \Psi(T_t[f])\} \quad (8.5)$$

para qualquer $f \in K^E$ e $x \in E$.

Dem.: Veja demonstração em [114] (Theorem 3, pag. 1156). ■

Qualquer operador $\Psi' : K^E \rightarrow K^E$ para o qual existe um operador binário crescente Ψ tal que a equação 8.5 vale é denominado um *operador plano* (Heijmans [75], página 364). Se Ψ' é um operador plano então, por construção, Ψ' comuta com o threshold e, além disso, $\Psi(X) = T_1[\Psi'(1_X)]$. Logo operadores que comutam com o threshold e operadores planos são os mesmos elementos. Observe que se $\Psi(\emptyset) = \emptyset$ então

$$\Psi'(1_X) = 1_{\Psi(X)} \quad (8.6)$$

isto é, nestes casos, imagens binárias são mapeadas em imagens binárias.

Filtros Stack são definidos como uma classe de operadores que obedecem a estrutura de decomposição threshold e que possuem a propriedade stack ² [33, 170].

²Em [33, 170], funções Booleanas positivas são referidas como funções que possuem a propriedade stack.

Proposição 8.7 (Estrutura de decomposição threshold) *Se $\Psi' : K^E \rightarrow K^E$ comuta com o threshold, então Ψ' obedece a estrutura de decomposição threshold, i.e., $\forall f \in K^E$,*

$$\Psi'(f) = \sum_{i=1}^k 1_{\Psi(T_i[f])}, \quad (8.7)$$

onde $\Psi = b(\Psi')$.

Dem.: *Da proposição 8.1, segue que $\Psi'(f) = \sum_{i=1}^k 1_{T_i[\Psi'(f)]}$, pois $\Psi'(f) \in K^E$. Por outro lado, fazendo o somatório em ambos os lados da Eq. 8.4, temos*

$$\sum_{i=1}^k 1_{T_i[\Psi'(f)]} = \sum_{i=1}^k 1_{\Psi(T_i[f])}$$

A partir das duas igualdades, temos o resultado. ■

Proposição 8.8 (Comuta com o threshold) *Um operador $\Psi' : K^E \rightarrow K^E$ comuta com o threshold sse Ψ' obedece a estrutura de decomposição threshold e é crescente.*

Dem.: (\implies) *já foi provada nas proposições 8.7 e 8.6.*

(\Leftarrow) *Por um lado, $\Psi'(f) = \sum_{i=1}^k 1_{T_i[\Psi'(f)]}$ (proposição 8.1). Por outro lado, $\Psi'(f) = \sum_{i=1}^k 1_{\Psi(T_i[f])}$ pois Ψ' obedece a estrutura de decomposição threshold. Isto é,*

$$\sum_{i=1}^k 1_{T_i[\Psi'(f)]} = \sum_{i=1}^k 1_{\Psi(T_i[f])}$$

onde $\Psi = b(\Psi')$.

Como tanto os termos do lado esquerdo quanto os do lado direito obedecem a propriedade stack, o primeiro por construção e o segundo por hipótese (pois Ψ é crescente), então $1_{T_i[\Psi'(f)]} = 1_{\Psi(T_i[f])}$, para todo $i \in K$. Isto é, $T_i[\Psi'(f)] = \Psi(T_i[f])$. ■

Note que a estrutura de decomposição threshold juntamente com a propriedade crescente de Ψ implica a equação 8.5. Esta observação e a anterior implicam que operadores planos, operadores que comutam com o threshold e filtros stack são os mesmos objetos.

8.2 Filtros W-Stack

Relembramos a noção de operadores i.t., operadores l.d. e W-operadores, no contexto de operadores sobre imagens em níveis de cinza.

Definição 8.9 (Invariante por translação) *Um operador $\Psi' : K^E \rightarrow K^E$ é invariante por translação (i.t.) se e somente se, para qualquer $f \in K^E$ e $z \in E$,*

$$\Psi'(f_z) = [\Psi'(f)]_z \quad (8.8)$$

Definição 8.10 (Localmente definido) Um operador $\Psi' : K^E \rightarrow K^E$ é localmente definido (l.d.) em W se e somente se, para qualquer $x \in E$ e quaisquer $f, g \in K^E$, se $f_{-x}|_W = g_{-x}|_W$ então

$$\Psi'(f)(x) = \Psi'(g)(x). \quad (8.9)$$

Definição 8.11 (W-operador) Um operador $\Psi' : K^E \rightarrow K^E$ é um W-operador sse Ψ' é i.t. e l.d. em W .

Proposição 8.12 (Caracterização de W-operadores através de funções) Um operador $\Psi' : K^E \rightarrow K^E$ é um W-operador sse existe um mapeamento $\psi' : K^W \rightarrow K$ tal que, para todo $f \in K^E$ e $x \in E$,

$$\Psi'(f)(x) = \psi'(f_{-x}|_W). \quad (8.10)$$

Dem.: Veja maiores detalhes em [8]. ■

De forma análoga ao caso binário, operadores de imagens em níveis de cinza que são invariantes por translação podem ser vistos como W-operadores utilizando-se $W = E$. Logo, todos os resultados apresentados a seguir para W-operadores são válidos para operadores i.t. em geral.

Definição 8.13 (Filtro W-stack) Um W-operador $\Psi' : K^E \rightarrow K^E$ é um filtro W-stack sse Ψ' comuta com o threshold.

Apesar da denominação filtros W-stack, estes operadores são aqueles conhecidos usualmente como “stack filters”. Adotamos este nome para uniformizar a nomenclatura com o restante deste texto e também para explicitar a existência de uma janela W na sua definição.

Proposição 8.14 Se $\Psi' : K^E \rightarrow K^E$ é um W-operador, caracterizado pela função $\psi' : K^W \rightarrow K$, então $\Psi = b(\Psi')$ é um W-operador (binário) caracterizado pela função definida por, $\forall X \in \mathcal{P}(W)$,

$$\psi(X) = T_1(\psi'(1_X)) \quad (8.11)$$

Dem.: De fato, para todo $x \in E$ e $S \in \mathcal{P}(E)$,

$$\begin{aligned} x \in \Psi(S) &\stackrel{(1)}{\iff} x \in T_1[\Psi'(1_S)] \\ &\stackrel{(2)}{\iff} \Psi'(1_S)(x) \geq 1 \\ &\stackrel{(3)}{\iff} \psi'(1_{S-x}|_W) \geq 1 \\ &\stackrel{(4)}{\iff} T_1(\psi'(1_{S-x}|_W)) = 1 \\ &\stackrel{(5)}{\iff} \psi(S_{-x} \cap W) = 1 \end{aligned}$$

A equivalência (1) segue da definição de b (eq. 8.3), (2) da definição de threshold, (3) pois Ψ' é um W-operador caracterizado por ψ' , (4) da eq. 8.1 e (5) pela eq. 8.11. ■

Observe que um W-operador $\Psi' : K^E \rightarrow K^E$ é um filtro W-stack (comuta com o threshold) sse a sua função característica também comuta com o threshold, isto é, sse para qualquer $i \in K$ e $f \in K^W$,

$$T_i(\psi'(f)) = \psi(T_i[f]). \quad (8.12)$$

Exemplo 8.15 Um exemplo clássico de um filtro W -stack é o filtro da mediana. A figura 8.2 ilustra a equivalência entre a mediana, com $W = \{-1, 0, 1\}$, calculada diretamente sobre o sinal em níveis de cinza e a soma das medianas calculadas sobre os sinais binários correspondentes aos cortes do sinal.

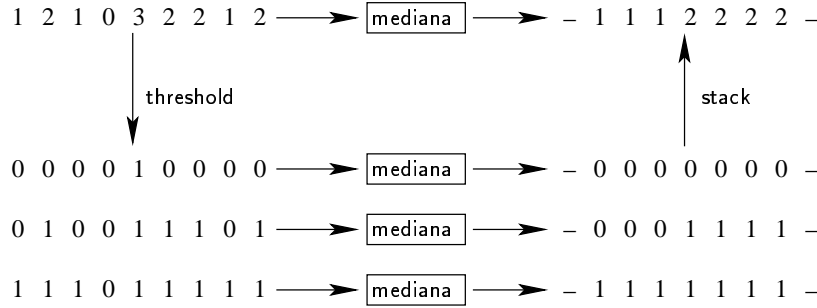


Figura 8.2: Propriedade stack.

A seguir mostraremos que o espaço das funções Booleanas positivas e dos filtros W -stack são isomorfos. Primeiramente, mostramos que o filtro W -stack associado a uma função Booleana positiva ψ é o W -operador caracterizado pela função $\psi'(f)(x) = \max\{t \in K : \psi(T_t[f]) = 1\}$ e esta associação será denotada por s , isto é, $\psi' = s(\psi)$. A função Booleana associada a um filtro W -stack Ψ' é a função dada por $\psi(X) = T_1(\psi'(1_X))$ e esta associação é denotada também por b , isto é, $\psi = b(\psi')$. Mostraremos que os mapeamentos s e b definem um isomorfismo de reticulados.

Lema 8.16 Seja $W \in \mathcal{P}(E)$ e seja $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ uma função crescente. Então a função dada por, para qualquer $f \in K^W$,

$$\psi'(f) = [s(\psi)(f)] = \max\{t \in K : \psi(T_t[f]) = 1\} \quad (8.13)$$

caracteriza um filtro W -stack.

Dem.: Inicialmente mostramos que $b(s(\psi)) = \psi$. De fato, para qualquer $X \in \mathcal{P}(W)$,

$$\begin{aligned} [b(s(\psi))](X) &\stackrel{(1)}{=} T_1(s(\psi)(1_X)) \\ &\stackrel{(2)}{=} T_1(\max\{t \in K : \psi(T_t[1_X]) = 1\}) \\ &\stackrel{(3)}{=} \begin{cases} T_1(0), & \text{se } \psi = \mathbf{0}, \\ T_1(0), & \text{se } \psi = \mathbf{1}, \\ T_1(\psi(T_1[1_X])), & \text{caso contrário.} \end{cases} \\ &\stackrel{(4)}{=} \psi(X) \end{aligned}$$

A igualdade (1) segue da Eq. 8.11, (2) da Eq. 8.13, (3) do fato de que $T_t[1_X] = \emptyset$ para qualquer $t \geq 2$, e (4) do fato de que $T_1[1_X] = X$.

Agora mostramos que $s(\psi)$ comuta com o threshold. Para qualquer $g \in K^W$ e $i \in K$,

$$T_i(s(\psi)(g)) = 1 \iff s(\psi)(g) \geq i$$

$$\begin{aligned}
&\stackrel{(2)}{\iff} \max\{t \in K : [b(s(\psi))](T_t[g]) = 1\} \geq i \\
&\stackrel{(3)}{\iff} \max\{t \in K : \psi(T_t[g]) = 1\} \geq i \\
&\stackrel{(4)}{\iff} \psi(T_i[g]) = 1
\end{aligned}$$

A equivalência (1) segue da Eq. 8.1, (2) da Eq. 8.13, (3) do fato de que $b(s(\psi)) = \psi$, e (4) da propriedade do \max . ■

Proposição 8.17 (Isomorfismo entre filtros stack e funções Booleanas crescentes) O espaço de todos os filtros W-stack e o espaço das funções Booleanas positivas com $|W|$ variáveis são isomorfos.

Dem.: Para mostrar que estes dois espaços são isomorfos, precisamos mostrar que existe uma bijeção entre eles que preserva a relação de ordem. Sejam os mapeamentos definidos pelas equações 8.11 e 8.13 (veja também Fig. 8.3).

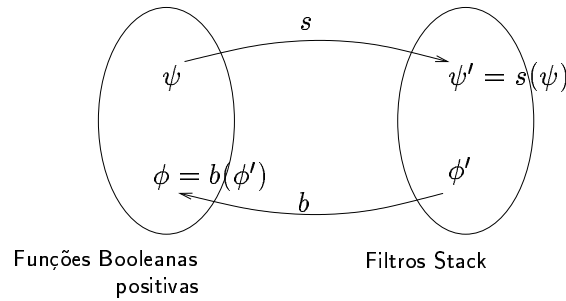


Figura 8.3: Isomorfismo entre funções Booleanas positivas e filtros stack.

Primeiramente, mostramos que os mapeamentos s e b constituem uma bijeção entre os dois espaços. Por um lado, seja ψ uma função Booleana positiva. Então, pelo lema 8.16, $s(\psi)$ caracteriza um filtro stack. Além disso, pelo mesmo lema, sabemos que $b(s(\psi)) = \psi$. Por outro lado, seja Φ' um filtro stack caracterizado por ϕ' . Então, $b(\phi')$ é uma função Booleana por construção e além disso, ela é positiva conforme a proposição 8.6. Mais ainda, $s(b(\phi')) = \phi'$, como mostramos a seguir.

$$\begin{aligned}
[s(b(\phi'))](f) &\stackrel{(1)}{=} \max\{t \in K : b(\phi')(T_t[f]) = 1\} \\
&\stackrel{(2)}{=} \max\{t \in K : T_1(\phi'(1_{T_t[f]})) = 1\} \\
&\stackrel{(3)}{=} \max\{t \in k : b(\phi')(T_1[1_{T_t[f]}]) = 1\} \\
&\stackrel{(4)}{=} \max\{t \in K : b(\phi')(T_t[f]) = 1\} \\
&\stackrel{(5)}{=} \max\{t \in K : T_t(\phi'(f)) = 1\} \\
&\stackrel{(6)}{=} \phi'(f)
\end{aligned}$$

A igualdade (1) segue da Eq. 8.13, (2) da Eq. 8.11, (3) e (5) porque ϕ' comuta com o threshold, (4) porque $T_1[1_X] = X$; e (6) da definição de threshold.

Mostramos agora que s e b preservam a relação de ordem parcial. Para quaisquer filtros stack $\Phi'_1, \Phi'_2 : K^E \rightarrow K^E$ e $X \in \mathcal{P}(W)$, se $\Phi'_1 \leq \Phi'_2$, então

$$b(\phi'_1)(X) \stackrel{(1)}{=} T_1(\phi'_1(1_X))$$

$$\begin{aligned}
& \stackrel{(2)}{\leq} T_1(\phi'_2(1_X)) \\
& \stackrel{(3)}{=} b(\phi'_2)(X)
\end{aligned}$$

A equivalência (1) segue da Eq. 8.11; (2) porque $\Phi'_1 \leq \Phi'_2 \Leftrightarrow \phi'_1 \leq \phi'_2$; e (3) da Eq. 8.11.

Para quaisquer funções Booleanas positivas $\psi_1, \psi_2 : \mathcal{P}(W) \rightarrow \{0, 1\}$, se $\psi_1 \leq \psi_2$, então para qualquer $f \in K^W$ e $x \in W$,

$$\begin{aligned}
s(\psi_1)(f) & \stackrel{(1)}{=} \max\{t \in K : \psi_1(T_t[f]) = 1\} \\
& \stackrel{(2)}{\leq} \max\{t \in K : \psi_2(T_t[f]) = 1\} \\
& = s(\psi_2)(f)
\end{aligned}$$

A igualdade (1) segue da Eq. 8.13; (2) é verdade porque $\psi_1 \leq \psi_2$. ■

A proposição 8.17 implica que cada filtro stack é unicamente caracterizado por uma função Booleana positiva e, analogamente, cada função Booleana positiva unicamente caracteriza um filtro stack. Os mapeamentos s e b mostram, respectivamente, como construir o filtro stack correspondente a uma dada função Booleana positiva e como construir uma função Booleana a partir de um dado filtro stack.

8.3 Representação de Filtros W-Stack

Filtros stack podem ser representados como o supremo de erosões por elementos estruturantes planos. Em [115] existe uma seção dedicada ao estudo da relação entre a representação Booleana e morfológica de filtros stack. Esse resultado é rerepresentado aqui explicitamente. Para isto revemos algumas definições.

Definição 8.18 (Erosão) *Seja $B \subseteq W$, B finito. O mapeamento $E'_B : K^E \rightarrow K^E$ dado por, para qualquer $f \in K^E$ e $x \in E$,*

$$E'_B(f)(x) = \varepsilon'_B(f_{-x}|_W) \quad (8.14)$$

onde para qualquer $g \in K^W$,

$$\varepsilon'_B(g) = \min\{g(y) : y \in B\} \quad (8.15)$$

é denominado a erosão de f pelo conjunto B .

Note que E'_B é um W -operador.

A erosão binária $E_B : \mathcal{P}(E) \rightarrow \mathcal{P}(E)$ pode ser definida a partir de E'_B da seguinte forma : para todo $S \in \mathcal{P}(E)$,

$$x \in E_B(S) \iff E'_B(1_S)(x) = 1. \quad (8.16)$$

A função característica ε_B que corresponde a E_B é dada por, para qualquer $X \subseteq W$,

$$\varepsilon_B(X) = 1 \iff \varepsilon'_B(1_X) = 1 \quad (8.17)$$

Note que $\varepsilon_B(X) = 1 \iff B \subseteq X$. Além disso, tanto a erosão como o supremo (máximo) de erosões comutam com o threshold [114].

As funções Booleanas positivas podem ser representadas como soma de produtos onde nenhuma variável complementada aparece. Conforme recordamos no capítulo 2, cada implicante primo de uma função Booleana positiva corresponde a um intervalo maximal do núcleo do operador correspondente. Todos estes intervalos possuem extremidade superior igual ao conjunto W e, portanto, na representação morfológica a função pode ser expressa como união de erosões, onde os elementos estruturantes são a extremidade inferior dos intervalos. Ou seja, se $\psi : \mathcal{P}(W) \rightarrow \{0, 1\}$ é uma função Booleana positiva caracterizando um W -operador binário Ψ , então, para qualquer $X \in \mathcal{P}(W)$,

$$\psi(X) = \max\{\varepsilon_B(X) : B \in \text{Bas}(\Psi)\}. \quad (8.18)$$

Proposição 8.19 (Representação de filtros stack) *Se $\Psi' : K^E \rightarrow K^E$ é um filtro W -stack, então para qualquer $x \in E$ e $f \in K^E$,*

$$\Psi'(f)(x) = \max\{\varepsilon'_B(f_{-x}|_W) : B \in \text{Bas}(\Psi)\} \quad (8.19)$$

Dem.: Como $\Psi'(f)(x) = \psi'(f_{-x}|_W)$, $\forall f \in K^E$ e $x \in E$, devemos apenas mostrar que

$$\psi'(f_{-x}|_W) = \max\{\varepsilon'_B(f_{-x}|_W) : B \in \text{Bas}(\Psi)\}$$

De fato, $\forall g \in K^W$,

$$\begin{aligned} \psi'(g) &\stackrel{(1)}{=} \max\{y \in K : T_y(\psi'(g)) = 1\} \\ &\stackrel{(2)}{=} \max\{y \in K : \psi(T_y[g]) = 1\} \\ &\stackrel{(3)}{=} \max\{y \in K : \max\{\varepsilon_B(T_y[g]) : B \in \text{Bas}(\Psi)\} = 1\} \\ &\stackrel{(4)}{=} \max\{y \in K : \max\{T_y(\varepsilon'_B(g)) : B \in \text{Bas}(\Psi)\} = 1\} \\ &= \max\{\max\{y \in K : T_y(\varepsilon'_B(g)) = 1\} : B \in \text{Bas}(\Psi)\} \\ &\stackrel{(5)}{=} \max\{\max\{y \in K : \varepsilon'_B(g) \geq y\} : B \in \text{Bas}(\Psi)\} \\ &= \max\{\varepsilon'_B(g) : B \in \text{Bas}(\Psi)\} \end{aligned}$$

A igualdade (1) segue da definição de threshold e max, (2) porque ψ comuta com o threshold, (3) da equação 8.18, (4) pois a erosão por elemento estruturante plano comuta com o threshold, e (5) da definição de threshold. ■

Proposição 8.20 *Seja $\Psi' : K^E \rightarrow K^E$ um W -operador cuja função característica é ψ' . Então as seguintes afirmações são equivalentes :*

1. Ψ' é um filtro W -stack.
2. Ψ' comuta com o threshold.
3. Ψ' obedece a estrutura de decomposição threshold e é crescente.
4. $\psi = b(\psi')$ é uma função Booleana positiva.
5. Ψ' é o supremo de erosões por elementos estruturantes planos.

Dada uma imagem em níveis de cinza f , um filtro W -stack Ψ' e $\Psi = b(\Psi')$, pode-se computar $\Psi'(f)$, $\forall x \in E$, das seguintes formas :

- calculando-se o supremo das erosões pelos elementos estruturantes na base de Ψ , i.e.,

$$\Psi'(f)(x) = \max\{\varepsilon'_{B_i}(f_{-x}|_W) : B_i \in \text{Bas}(\Psi)\}$$

- determinadno-se o maior nível de corte para o qual o resultado em x é 1, i.e.,

$$\Psi'(f)(x) = \max\{t \in K : \psi(T_k[f_{-x}|_W]) = 1\}$$

- ou somando-se o resultado binário correspondente a cada um dos cortes, i.e.,

$$\Psi'(f)(x) = \sum_{i=1}^k 1_{\psi(T_k[f_{-x}|_W])}$$

8.4 Filtros W-Stack Ótimos

O erro MAE tem sido largamente utilizado para avaliar o desempenho de filtros stack, pois o MAE de um filtro stack é igual à soma dos MAEs da correspondente função Booleana sobre cada um dos cortes [33].

Sejam $f, f_0 \in K^E$ uma imagem a ser filtrada e sua correspondente imagem ideal. O MAE de um filtro stack Ψ' , caracterizado por ψ' , é dado por

$$MAE\langle\Psi'\rangle = E\left[|f_0(z) - \psi'(f_{-z}|_W)|\right]. \quad (8.20)$$

Seja Ψ o operador binário correspondente a Ψ' , caracterizado por ψ . O MAE de Ψ no corte i é dado por

$$MAE_i\langle\Psi\rangle = E\left[|T_i(f_0(z)) - \psi(T_i[f_{-z}|_W])|\right]. \quad (8.21)$$

Proposição 8.21 *Seja Ψ' um filtro stack e Ψ o operador binário correspondente. A seguinte igualdade é verdadeira (veja [33]).*

$$MAE\langle\Psi'\rangle = \sum_{i=1}^k MAE_i\langle\Psi\rangle \quad (8.22)$$

■

Se f e f_0 são conjuntamente estacionários, o valor $1_{T_i[f_0(z)]}$ pode ser considerado uma realização de uma variável aleatória binária \mathbf{y}_i e o conjunto $T_i[f_{-z}|_W]$ uma realização de um conjunto aleatório \mathbf{X}_i . Denotamos por $P_i(\mathbf{X})$ a probabilidade de $\mathbf{X}_i = \mathbf{X}$ (i.e., $P_i(\mathbf{X}) = P(\mathbf{X}_i = \mathbf{X})$) e por $P_i(y|\mathbf{X})$ a probabilidade condicional de $\mathbf{y}_i = y$ dado que $\mathbf{X}_i = \mathbf{X}$ (i.e., $P_i(y|\mathbf{X}) = P(\mathbf{y}_i = y|\mathbf{X}_i = \mathbf{X})$), onde $\mathbf{X} \in \mathcal{P}(W)$ e $y \in \{0, 1\}$. Usando estas notações, a Eq. 8.21 pode ser reescrita como

$$MAE_i\langle\Psi\rangle = \sum_{\mathbf{X}} P_i(\mathbf{X}) \sum_y |y - \psi(\mathbf{X})| P_i(y|\mathbf{X}). \quad (8.23)$$

Expandindo a Eq. 8.22, obtemos

$$\begin{aligned} MAE\langle\Psi'\rangle &= \sum_{i=1}^k \sum_{\mathbf{X}} P_i(\mathbf{X}) \sum_y |y - \psi(\mathbf{X})| P_i(y|\mathbf{X}) \\ &= \sum_{\mathbf{X}} \underbrace{\sum_{i=1}^k P_i(\mathbf{X}) \sum_y |y - \psi(\mathbf{X})| P_i(y|\mathbf{X})}_{M_{\mathbf{X}}(\psi)}. \end{aligned} \quad (8.24)$$

Um filtro stack é MAE-ótimo se ele possui o menor erro MAE dentre todos os filtros stack. A minimização da Eq. 8.20 é equivalente à minimização da Eq. 8.24. Nenhuma solução simples para encontrar um filtro stack ótimo é conhecido. A dificuldade está relacionada ao fato de que ψ precisa ser crescente, o que significa que os termos $M_X(\psi)$ não podem ser minimizados independentemente; se $\psi(X)$ é fixado como 1, então $\psi(Y)$ para qualquer $Y > X$ também precisa ser fixado em 1. Inversamente, se $\psi(X)$ é fixado em 0, então $\psi(Y)$ para qualquer $Y < X$ também precisa ser fixado em 0.

A minimização da equação 8.24 pode ser formulada em termos de custos. A seguir inserimos este problema num contexto similar ao do projeto de operadores crescentes que foi discutido no capítulo 7.

Seja $M(\psi) = \sum_{i=1}^k MAE_i(\psi) = \sum_{i=1}^k M_X(\psi)$ (note que $M(\psi) = MAE(\Psi')$ se, e somente se, ψ é crescente, i.e., se Ψ' é um filtro stack). Consideramos primeiramente o problema de determinar uma função Booleana (não necessariamente positiva) que minimiza M . M é minimizado quando M_X é minimizado para cada X . Como $y \in \{0, 1\}$,

$$M_X(\psi) = \sum_{i=1}^k P_i(X) \psi(X) P_i(0|X) + \sum_{i=1}^k P_i(X) (1 - \psi(X)) P_i(1|X).$$

Se fixamos $\psi(X) = 0$, então o valor com o qual X contribui para o valor total de $M(X)$ é

$$c_0(X) = \sum_{i=1}^k P_i(X) P_i(1|X)$$

e, por outro lado, se fixamos $\psi(X) = 1$, então este valor é

$$c_1(X) = \sum_{i=1}^k P_i(X) P_i(0|X).$$

Logo, $M(\psi)$ é minimizado pela seguinte função Booleana :

$$\psi_{opt}(X) = \begin{cases} 1, & \text{se } c_0(X) > c_1(X), \\ 0, & \text{se } c_0(X) \leq c_1(X). \end{cases} \quad (8.25)$$

Note que esta função pode não ser positiva.

Se examinarmos o aumento de erro $M_X(\psi) - M_X(\psi_{opt})$ de uma função Booleana qualquer ψ com relação a ψ_{opt} , então o total que X contribui para este aumento é dado por

$$\begin{aligned} c(X) &= M_X(\psi) - M_X(\psi_{opt}) \\ &= \sum_{i=1}^k P_i(X) \psi(X) P_i(0|X) + \sum_{i=1}^k P_i(X) (1 - \psi(X)) P_i(1|X) \\ &\quad - \sum_{i=1}^k P_i(X) \psi_{opt}(X) P_i(0|X) - \sum_{i=1}^k P_i(X) (1 - \psi_{opt}(X)) P_i(1|X) \\ &= \sum_{i=1}^k P_i(X) P_i(0|X) [\psi(X) - \psi_{opt}(X)] + \sum_{i=1}^k P_i(X) P_i(1|X) [\psi_{opt}(X) - \psi(X)] \\ &= c_1(X) [\psi(X) - \psi_{opt}(X)] + c_0(X) [\psi_{opt}(X) - \psi(X)] \end{aligned}$$

$$\begin{aligned}
&= \begin{cases} (c_1(X) - c_0(X))\psi(X), & \text{se } \psi_{opt}(X) = 0, \\ (c_1(X) - c_0(X))\psi(X) + c_0(X) - c_1(X), & \text{se } \psi_{opt}(X) = 1 \end{cases} \\
&= \begin{cases} c_1(X) - c_0(X), & \text{se } \psi_{opt}(X) = 0 \text{ e } \psi(X) = 1, \\ c_0(X) - c_1(X), & \text{se } \psi_{opt}(X) = 1 \text{ e } \psi(X) = 0, \\ 0, & \text{se } \psi_{opt}(X) = \psi(X). \end{cases}
\end{aligned}$$

Encontrar uma função Booleana positiva que minimiza a Eq. 8.20 é equivalente a encontrar uma função Booleana positiva ψ que minimiza $\Delta_M(\psi, \psi_{opt}) = \sum_{\{X: \psi_{opt}(X) \neq \psi(X)\}} c(X)$, onde ψ_{opt} é dada pela Eq. 8.25 e os custos $c(X)$ são dados, para qualquer $X \in \mathcal{P}(W)$, por

$$c(X) = \begin{cases} c_1(X) - c_0(X), & \text{se } \psi_{opt}(X) = 0, \\ c_0(X) - c_1(X), & \text{se } \psi_{opt}(X) = 1. \end{cases} \quad (8.26)$$

8.5 Projeto de Filtros W-Stack

Um filtro stack pode ser projetado utilizando-se o algoritmo OVP, com custos de chaveamento dados pela equação 8.26.

Sejam dados um conjunto de imagens de treinamento (pares de imagens observadas-ideais) e sejam

- N_i o número total de observações nos cortes de nível i
- $N_i(X)$ o número de vezes em que X foi observado nos cortes de nível i
- $N_i(X, 1)$ o número de vezes em que X foi observado nos cortes de nível i com $y = 1$
- $N_i(X, 0)$ o número de vezes em que X foi observado nos cortes de nível i com $y = 0$

Consideramos a estimação usual para as probabilidades $P_i(X)$, $P_i(1|X)$ e $P_i(0|X)$. Estes são dados, respectivamente, por

$$\hat{P}_i(X) = \frac{N_i(X)}{N_i},$$

$$\hat{P}_i(1|X) = \frac{N_i(X, 1)}{N_i(X)}$$

e

$$\hat{P}_i(0|X) = \frac{N_i(X, 0)}{N_i(X)}.$$

A partir destas estimações, um estimador $\hat{\psi}_{opt}$ para a função Booleana ótima pode ser obtida apenas substituindo-se as probabilidades da Eq. 8.25 pelas respectivas estimativas, i.e., para qualquer $X \in \mathcal{P}(W)$,

$$\hat{\psi}_{opt}(X) = \begin{cases} 1, & \text{se } \hat{c}_0(X) > \hat{c}_1(X), \\ 0, & \text{se } \hat{c}_0(X) \leq \hat{c}_1(X). \end{cases} \quad (8.27)$$

onde

$$\hat{c}_1(X) = \sum_{i=1}^k \hat{P}_i(X) \hat{P}_i(0|X)$$

e

$$\hat{c}_0(\mathbf{X}) = \sum_{i=1}^k \hat{P}_i(\mathbf{X}) \hat{P}_i(1|\mathbf{X}).$$

O estimador para os custos da Eq. 8.26 é dado por

$$\hat{c}(\mathbf{X}) = \begin{cases} \hat{c}_1(\mathbf{X}) - \hat{c}_0(\mathbf{X}), & \text{if } \hat{\psi}_{opt}(\mathbf{X}) = 0, \\ \hat{c}_0(\mathbf{X}) - \hat{c}_1(\mathbf{X}), & \text{if } \hat{\psi}_{opt}(\mathbf{X}) = 1 \end{cases} \quad (8.28)$$

Em particular, as imagens binárias correspondentes a cada corte da imagem em níveis de cinza possuem o mesmo tamanho, i.e., $N_i = N$ para todo $i \in K$ (N é um inteiro positivo). Logo, $\hat{\psi}_{opt}$ e $\hat{c}(\mathbf{X})$ podem ser simplificados. Obtemos :

$$\begin{aligned} \hat{\psi}_{opt}(\mathbf{X}) &= \begin{cases} 1, & \text{se } \sum_{i=1}^k \hat{P}_i(\mathbf{X}) \hat{P}_i(1|\mathbf{X}) > \sum_{i=1}^k \hat{P}_i(\mathbf{X}) \hat{P}_i(0|\mathbf{X}), \\ 0, & \text{se } \sum_{i=1}^k \hat{P}_i(\mathbf{X}) \hat{P}_i(1|\mathbf{X}) \leq \sum_{i=1}^k \hat{P}_i(\mathbf{X}) \hat{P}_i(0|\mathbf{X}) \end{cases} \\ &= \begin{cases} 1, & \text{se } \sum \frac{N_i(\mathbf{X})}{N_i} \frac{N_i(\mathbf{X},1)}{N_i(\mathbf{X})} > \sum \frac{N_i(\mathbf{X})}{N_i} \frac{N_i(\mathbf{X},0)}{N_i(\mathbf{X})}, \\ 0, & \text{se } \sum \frac{N_i(\mathbf{X})}{N_i} \frac{N_i(\mathbf{X},1)}{N_i(\mathbf{X})} \leq \sum \frac{N_i(\mathbf{X})}{N_i} \frac{N_i(\mathbf{X},0)}{N_i(\mathbf{X})} \end{cases} \\ &= \begin{cases} 1, & \text{se } \frac{1}{N} \sum N_i(\mathbf{X},1) > \frac{1}{N} \sum N_i(\mathbf{X},0), \\ 0, & \text{se } \frac{1}{N} \sum N_i(\mathbf{X},1) \leq \frac{1}{N} \sum N_i(\mathbf{X},0). \end{cases} \\ &= \begin{cases} 1, & \text{se } \sum N_i(\mathbf{X},1) > \sum N_i(\mathbf{X},0), \\ 0, & \text{se } \sum N_i(\mathbf{X},1) \leq \sum N_i(\mathbf{X},0). \end{cases} \end{aligned}$$

e

$$\hat{c}(\mathbf{X}) = \begin{cases} \frac{1}{N} [\sum N_i(\mathbf{X},0) - \sum N_i(\mathbf{X},1)], & \text{if } \hat{\psi}_{opt}(\mathbf{X}) = 0, \\ \frac{1}{N} [\sum N_i(\mathbf{X},1) - \sum N_i(\mathbf{X},0)], & \text{if } \hat{\psi}_{opt}(\mathbf{X}) = 1. \end{cases} \quad (8.29)$$

Isto mostra que não há necessidade de se estimar as probabilidades para cada um dos níveis de corte. Os k cortes de tamanho N obtidos de uma imagem de tamanho N podem ser considerados como uma única grande imagem com kN pontos.

Conseqüentemente, (1) podemos aplicar o algoritmo OVP sobre os custos estimados acima para projetarmos um filtro W-stack ótimo e (2) podemos considerar um custo alternativo $c_{alt}(\mathbf{X}) = \gamma c(\mathbf{X})$ ($\gamma > 0$) para transformar custos em custos inteiros, que teremos um problema de otimização equivalente.

A seguir descrevemos o algoritmo que propomos para projetar filtros stack MAE-ótimos a partir de uma coleção de imagens de treinamento :

Entrada: Pares de imagens de treinamento (S_i, I_i) , janela W
Saída: Base de um operador crescente ótimo (estimado)

1. Estimar as probabilidades $\hat{P}(X) = \frac{\sum_{i=1}^k N_i(X)}{kN}$ e $\hat{P}(1|X) = \frac{\sum_{i=1}^k N_i(X,1)}{\sum_{i=1}^k N_i(X)}$ a partir das imagens de treinamento.
2. Calcular $\hat{\psi}_{opt}(X) = 1 \Leftrightarrow \hat{P}(1|X) > 0.5$
3. Calcular o conjunto violador $Q = Q_{\hat{\psi}_{opt}}$
4. Calcular os custos de chaveamento $\hat{c}(X)$ como especificado na Eq. 8.29
5. Aplicar o algoritmo OVP sobre Q . Denotemos por \mathcal{A} o conjunto de chaveamento resultante.
6. Calcule os elementos minimais B de $\hat{\psi}_{opt_{\mathcal{A}}}(1)$, que são os elementos da base do operador crescente ótimo estimado.

Algoritmo 8.1 Algoritmo para o projeto de filtros stack.

Os elementos da base correspondem aos elementos estruturantes que caracterizam um filtro stack (Eq. 8.19).

8.6 Alguns Resultados Experimentais

8.6.1 Filtragem de Ruído do Tipo Impulso

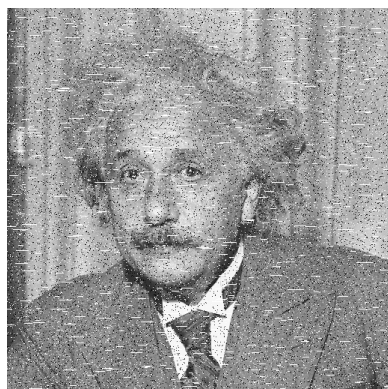
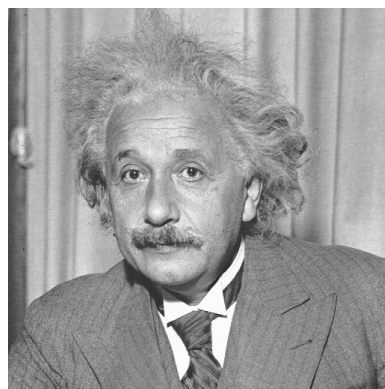
No exemplo a seguir ilustramos o uso de filtros stack para filtragem de ruído tipo impulso de imagens em níveis de cinza. O ruído considerado é uma composição de pulsos aditivos e subtrativos (os quais são uniformemente distribuídos com probabilidade de ocorrência de 10% e amplitude 200) com pulsos aditivos contínuos (i.e., segmentos de linhas horizontais de intensidade 255 com probabilidade de ocorrência 0.35%, cujo comprimento segue uma distribuição normal com média 5 e variância 49). A figura 8.4 mostra o par de imagens observada-ideal que foi utilizada para treinamento de um filtro stack sobre uma janela de 17 pontos. O tempo de treinamento foi de aproximadamente 5 minutos. O número de exemplos distintos foi 129254 e o número de elementos estruturantes do operador projetado foi 833.

A figura 8.5 mostra a mesma imagem corrompida por uma outra realização do mesmo processo de ruído e a respectiva imagem filtrada pelo operador projetado.

A figura 8.6 mostra uma outra imagem corrompida com uma outra realização do mesmo processo de ruído e a respectiva imagem filtrada pelo mesmo operador. Podemos notar que o operador projetado é bastante robusto.

8.6.2 Filtragem de Ruído *Speckle*

Speckle é um tipo de ruído que aparece em imagens obtidos por sistemas de imageamento coerente tais como os baseados na tecnologia de radares de abertura sintética (“synthetic aperture radar”

(a) Observada ($MAE = 13.934361$)

(b) Ideal

Figura 8.4: Imagens de treinamento.

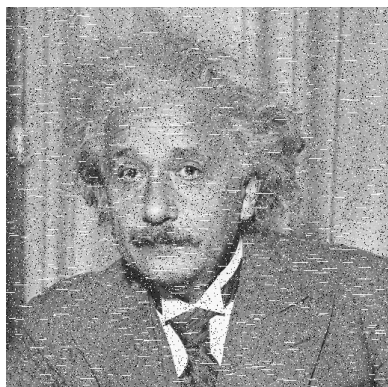
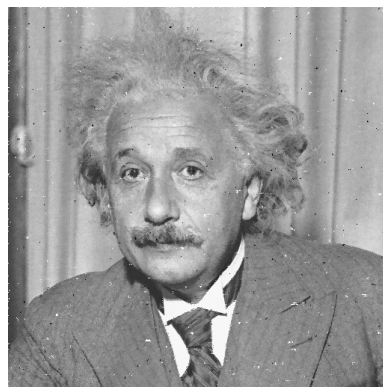
(a) Teste ($MAE = 14.0028$)(b) Resultado ($MAE = 3.2352$)

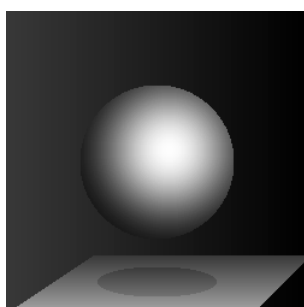
Figura 8.5: Filtragem de ruído impulso.

(SAR)) [106, 116]. Os ruídos speckle podem ser de amplitude ou intensidade e estar associados a diferentes números de visadas.

As imagens SAR de intensidade com 1 visada foram simuladas de acordo com a descrição apresentada em [106]. Utilizamos o método iterativo apresentado no capítulo 5 para projetar um operador de 4 iterações sobre a janela de 21 pontos, utilizando um total de 5 pares de imagens de treinamento, 4 pares em cada iteração. O tempo total de treinamento foi de aproximadamente 15 horas. O número de elementos estruturantes dos operadores projetados foram, respectivamente, 42305, 6292, 1726 e 483. A figura 8.7 mostra uma imagem sintética, uma imagem ruidosa (teste), e os resultados obtidos pelas iterações de 1 a 4, respectivamente.

(a) Teste ($MAE = 13.2344$)(b) Resultado ($MAE = 1.4053$)

Figura 8.6: Filtragem de ruído impulso - robustez.



(a) Ideal

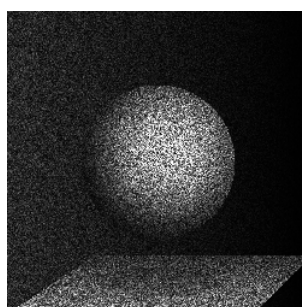
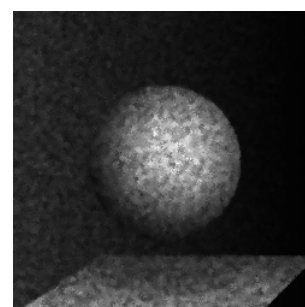
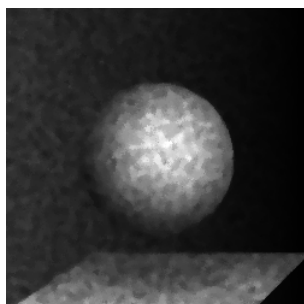
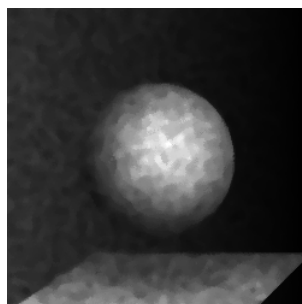
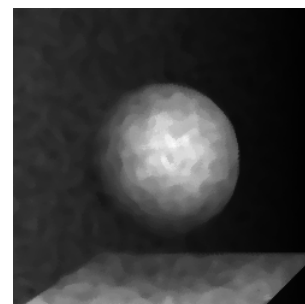
(b) Teste ($MAE = 21.8212$)(c) Iteração 1 ($MAE = 6.8640$)(d) Iteração 2 ($MAE = 4.5953$)(e) Iteração 3 ($MAE = 3.8472$)(f) Iteração 4 ($MAE = 3.5682$)

Figura 8.7: Filtragem de ruído speckle (intensidade com 1 visada).

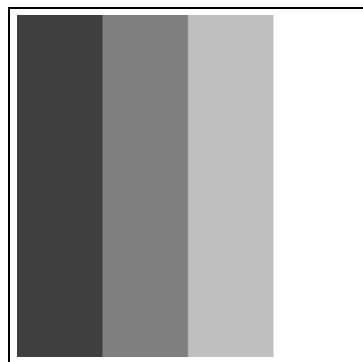
Um experimento similar foi repetido para filtragem de ruído speckle de amplitude com 4-visadas. Um operador de 5 iterações sobre a janela de 17 pontos foi projetado. Cada um dos 5 filtros foi projetado a partir de 6 pares de imagens de treinamento, de um total de 10 pares. O tempo total de treinamento total foi de aproximadamente 30 minutos. A figura 8.8 mostra, respectivamente, a imagem ideal (composta de quatro regiões verticais com valores de níveis de cinza crescentes), uma simulação de imagem SAR de amplitude com 4-visadas (imagem observada), e os resultados obtidos pelas iterações de 1 a 5. O número de elementos estruturantes dos 5 operadores foram, respectivamente, 10995, 2875, 1194, 984 e 286. O índice médio de redução de ruído speckle $E(\beta)$, onde $\beta = \frac{\sqrt{Var(\Psi(f)(x))}}{E(\Psi(f)(x))}$, sobre as 4 regiões homogêneas é 0.2622 para a imagem ruidosa original, 0.07968 para o resultado do filtro de 1 iteração e 0.04326 para o de 5 iterações.

8.7 Comentários

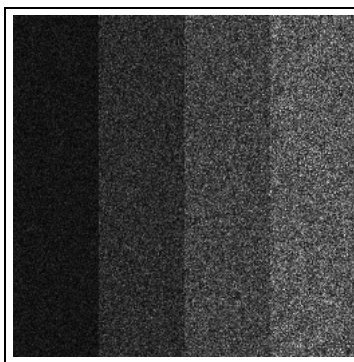
Neste capítulo apresentamos uma visão geral sobre filtros stack, unificando as definições utilizadas no contexto de morfologia matemática e fora dele. Uma prova construtiva da equivalência entre filtros stack e funções Booleanas positivas foi apresentada.

Mostramos também que a soma dos MAEs binários relativos aos cortes é proporcional ao MAE binário quando os exemplos são considerados sem vínculo ao nível de corte que o originou. Com isto, mostramos que o algoritmo de chaveamento proposto no capítulo 7 para projetar W -operadores binários e crescentes pode ser diretamente aplicado para projetar uma função Booleana positiva que caracteriza um filtro stack ótimo.

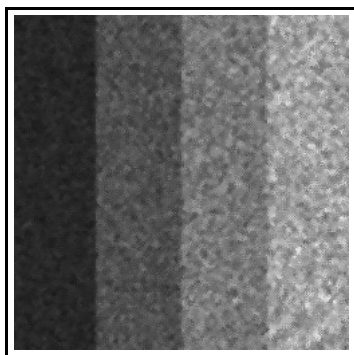
Para concluir este capítulo, lembramos que existem vários algoritmos para projeto de filtros stack. Entre eles merecem destaque os desenvolvidos por um grupo da Purdue University e da Tampere University. Inicialmente [33] o problema foi modelado como um problema de programação linear [127], porém esta abordagem foi abandonada [108]. Um algoritmo adaptativo [108], no sentido de que os dados são processados a medida que são obtidos e o filtro sendo projetado é adaptativamente modificado para ajustar-se aos dados obtidos, foi proposto em seguida. Uma versão modificada do mesmo algoritmo é proposto em [107], onde a adaptação ocorre a cada ciclo regular e não necessariamente após cada exemplo observado. A mais recente proposta em torno deste algoritmo é sua paralelização [170]. Na Tampere University, a abordagem proposta apresenta uma construção em parte similar a nossa proposta, isto é, eles também utilizam a noção de conjunto violador. O algoritmo proposto por eles é denominado FASTAF [155] e uma melhora deste mesmo algoritmo, denominado FASTAR, é proposto em [154]. A grande diferença entre estes algoritmos e o nosso é o fato de que estes manipulam todos os 2^W elementos da janela, enquanto o algoritmo OVP manipula apenas os padrões observados nas imagens de treinamento.



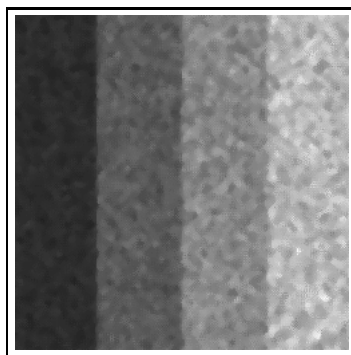
(a) Ideal



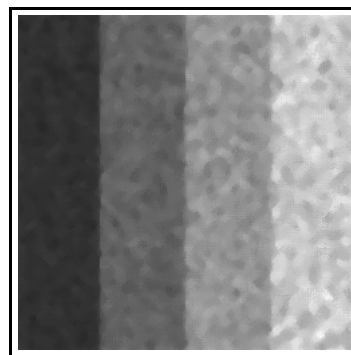
(b) Teste ($MAE = 16.5794$)



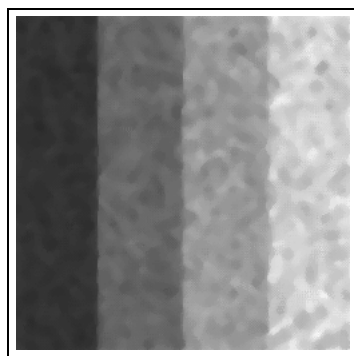
(c) Resultado da iteração 1
($MAE = 5.4542$)



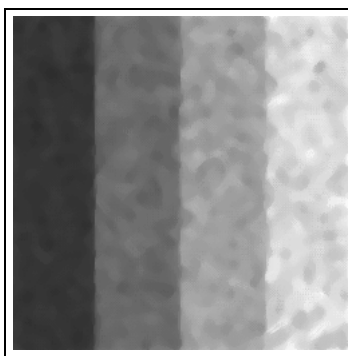
(d) Resultado da iteração 2
($MAE = 3.4711$)



(e) Resultado da iteração 3
($MAE = 2.7515$)



(f) Resultado da iteração 4
($MAE = 2.3977$)



(g) Resultado da iteração 5
($MAE = 2.1779$)

Figura 8.8: Filtragem de ruído speckle (amplitude com 4 visadas).

Capítulo 9

Exemplos de Aplicações

Neste capítulo apresentamos várias aplicações que ilustram a utilização de operadores projetados pelas técnicas apresentadas. Todos os experimentos foram realizados utilizando-se computadores com processadores Pentium II ou III, de 133MHz a 450Mhz, com 128Mb a 192Mb de memória RAM. A indicação do tempo de processamento não é precisa e serve apenas como uma referência.

9.1 Filtragem de Ruído Tipo Impulso

Os ruídos do tipo impulso consistem de falhas espalhadas irregularmente na imagem, geralmente consistindo de pulsos de alta ou baixa intensidade e de pouca duração. Este exemplo mostra a aplicação de operadores binários crescentes para filtragem deste tipo de ruído em imagens sintéticas. A figura 9.1 mostra um par de imagens de treinamento, 15% de ruído aditivo e 15% de ruído subtrativo (ver detalhes do experimento na tabela 9.1).

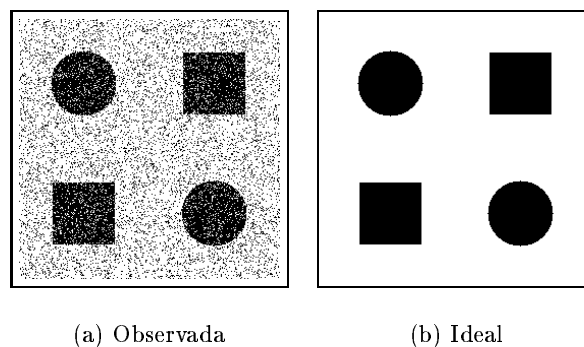


Figura 9.1: Filtragem de ruído: exemplo de imagens de treinamento.

A figura 9.2 mostra uma imagem de teste, o respectivo resultado obtido aplicando-se o operador projetado e a diferença simétrica entre o resultado e a imagem ideal, sobreposta sobre a imagem ideal.

Problema: Filtragem de ruído sal-e-pimenta	
Propriedade do operador	crescente
Janela	$W_{5 \times 5}$
Função de perda	MAE
Algoritmo	OVP
Exemplos distintos	288604
$ \psi(0) $	193190
Intervalos	5134
Imagens de treinamento	20 pares
Tempo de treinamento	50 segundos

Tabela 9.1: Filtragem de ruído sal-e-pimenta: detalhes do treinamento.

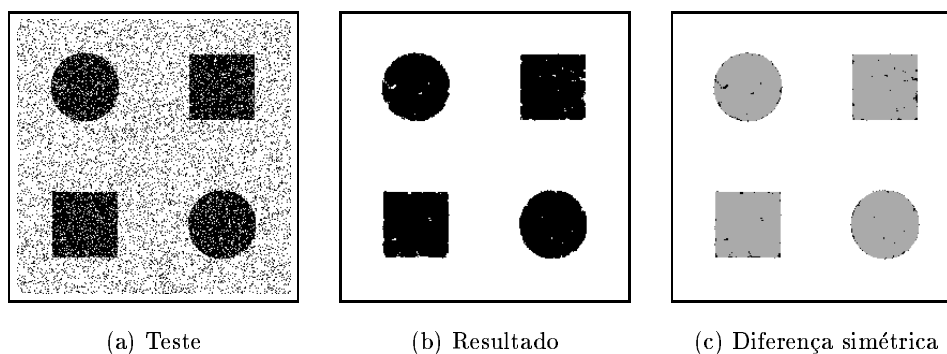


Figura 9.2: Imagem de teste, resultado, e diferença simétrica (erro 335 pixels).

Para este problema em particular, filtros de ordem¹, com a escolha adequada do parâmetro de ordem, produzem resultados bastante satisfatórios. A figura 9.3 mostra o resultado do filtro de ordem 15 composto com o filtro de ordem 11, ambos com relação a janela $W_{5 \times 5}$, para a imagem teste da figura 9.2.

O operador gerado pelo algoritmo OVP produz imagens com algumas falhas no interior do círculo e do quadrado, enquanto o filtro de ordem gera imagens com erro na borda destes objetos. A combinação destes dois resultados, através da união, gera uma imagem com erro menor, conforme mostrado na figura 9.4.

A figura 9.5 mostra o resultado para outra imagem de teste.

¹Filtro de ordem: veja, por exemplo, [75], pág.98, ou [1]

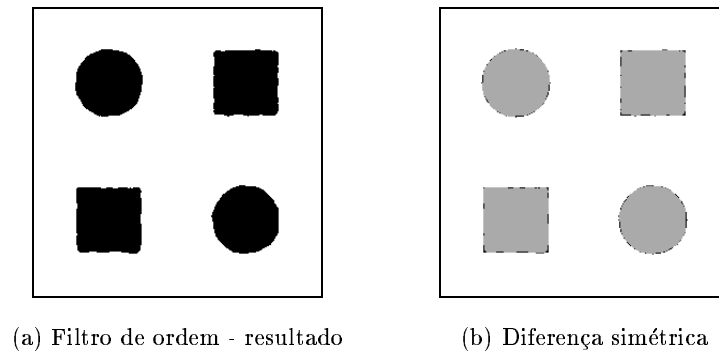


Figura 9.3: Resultado do filtro de ordem e diferença simétrica (erro 324 pixels).

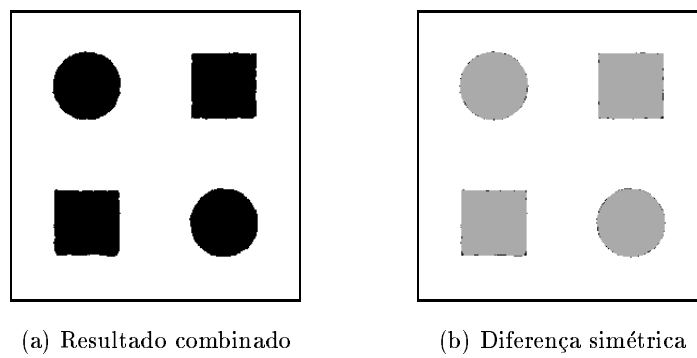
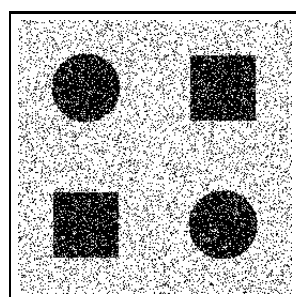
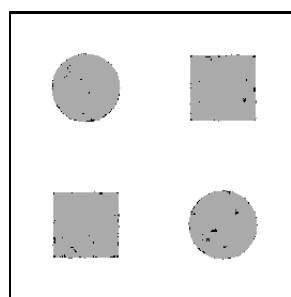
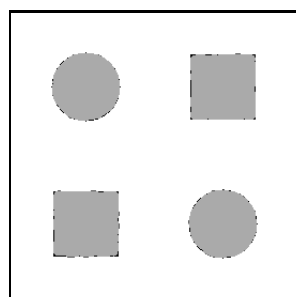


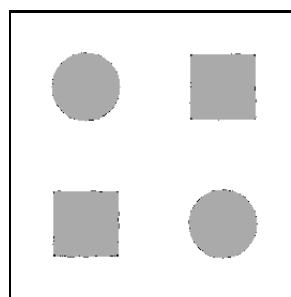
Figura 9.4: Resultado da combinação, diferença simétrica (erro 199 pixels).



(a) Teste (S)

(b) $\hat{\Psi}_{inc}(S)$ 

(c) Filtro de ordem - resultado



(d) União de (b) e (c)

Figura 9.5: Imagem de teste, resultado do operador projetado (erro 303 pixels), resultado do filtro de ordem (erro 258 pixels), e união dos dois resultados (erro 175 pixels), respectivamente.

9.2 Reconhecimento de Padrões em Diagramas

Os *diagramas funcionais* são largamente utilizados para expressar as principais características de processos e fenômenos em diversas áreas. Localizar certos elementos funcionais nestes diagramas pode ser interessante para planejar sua implementação ou automatizar sua especificação dentro de um banco de dados, por exemplo.

Consideramos diagramas funcionais de sistemas de controle binário e o problema de localizar elementos circulares que correspondem a somadores/multiplicadores de sinais. A figura 9.6 mostra um par de imagens de treinamento.

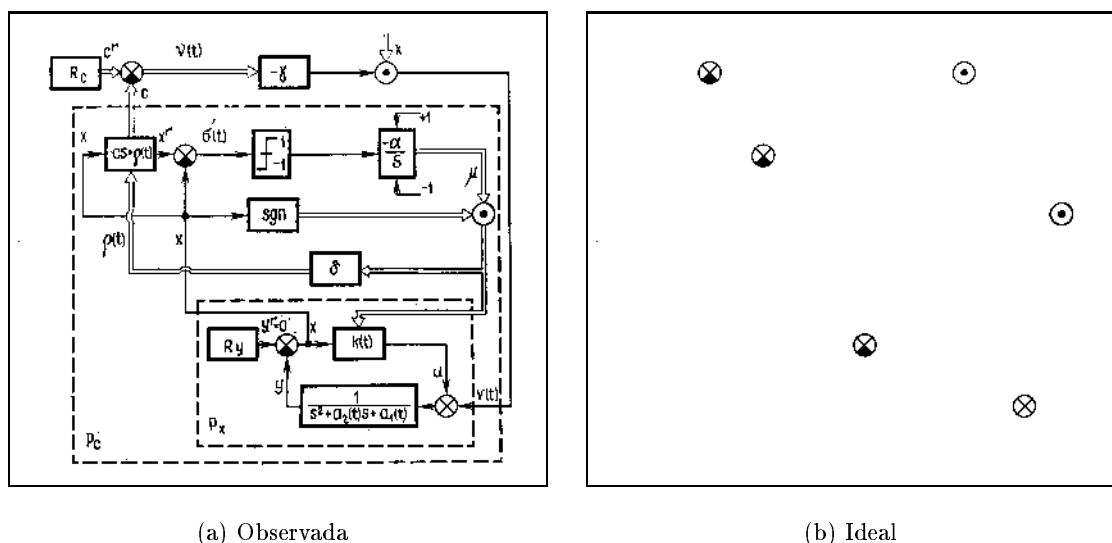


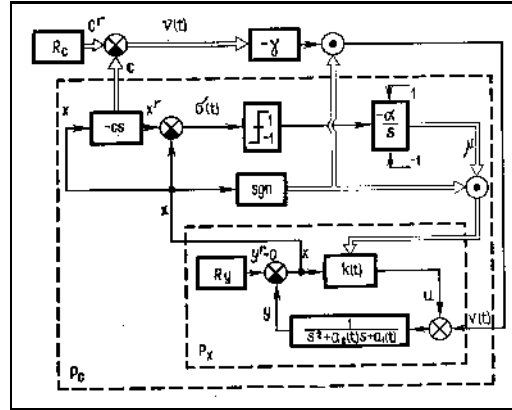
Figura 9.6: Reconhecimento de padrões: exemplo de imagem de treinamento.

Problema: Reconhecimento de somadores/multiplicadores		
	Iteração 1	Iteração 2
Propriedade do operador	anti-extensivo	anti-extensivo
janela	$W_{9 \times 7}$	$W_{7 \times 5}$
Função de perda	MAE	MAE
Algoritmo	ISI com $p = 2$ e $k = 25$	ISI com $k = 25$
Exemplos distintos	28688	8234
Intervalos	313	149
Imagens de treinamento	6 pares	6 pares
Tempo de treinamento	110497 segundos	31 segundos

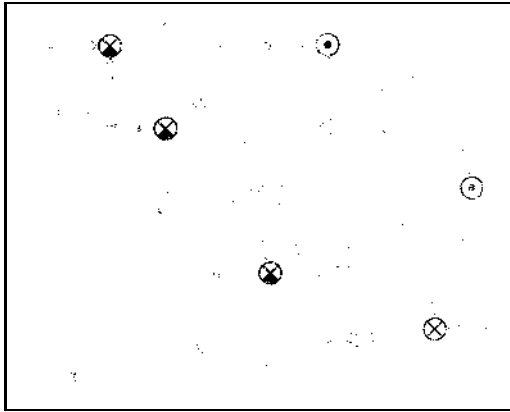
Tabela 9.2: Reconhecimento de somadores/multiplicadores: detalhes do treinamento.

Para resolver este problema, foram utilizados um total de 10 pares de imagens de treinamento, semelhantes aos mostrados na figura 9.6. Um operador de duas iterações foi projetado sobre as janelas $W_{9 \times 7}$ e $W_{7 \times 5}$, respectivamente. O primeiro operador foi projetado a partir dos 6 primeiros

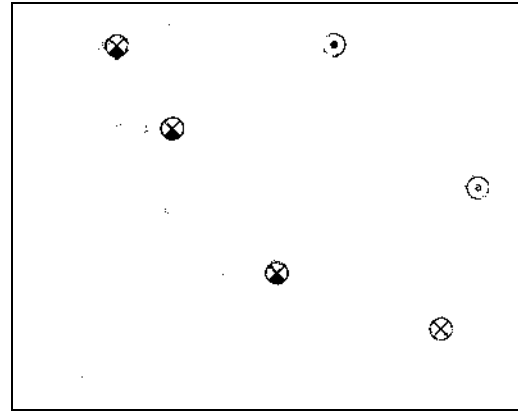
pares e o segundo a partir dos 4 últimos mais os 2 primeiros pares. O resultado deste operador para uma imagem de teste é mostrado na figura 9.7.



(a) Teste



(b) Resultado da iteração 1



(c) Resultado da iteração 2

Figura 9.7: Reconhecimento de padrão em diagramas: imagem (teste) observada, resultado da primeira iteração e resultado da segunda iteração, respectivamente.

Os pequenos pontos isolados (ruídos) remanescentes podem ser eliminados por uma abertura por área², por exemplo. No entanto ela também elimina os pontos isolados que fazem parte do objeto que desejamos extrair. Da mesma forma, maior número de iterações também tendem a causar o mesmo efeito (o qual pode ser observado no resultado da segunda iteração em relação ao resultado da primeira iteração). Para filtrar o ruído remanescente, sem estragar o objeto extraído, utilizamos o seguinte filtro :

1. Filtro de ordem³ 12 por um círculo de raio 9. Este tem o efeito de marcar as regiões da imagem que correspondem aos objetos de interesse. Este efeito é mostrada na figura 9.8.

²Abertura por área: veja, por exemplo, [142]

³Filtro de ordem: veja, por exemplo, [75], pág.98.

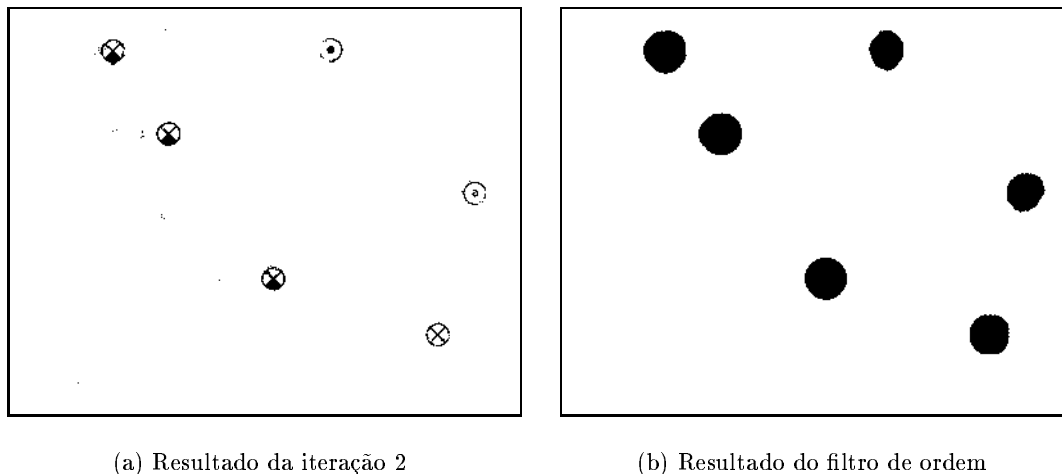


Figura 9.8: Reconhecimento de padrão em diagramas: resultado do operador e filtro de ordem 12 por uma janela circular de raio 9, respectivamente.

2. Reconstrução⁴ do resultado do operador projetado a partir do resultado do filtro de ordem. O resultado da reconstrução é mostrada na figura 9.9.

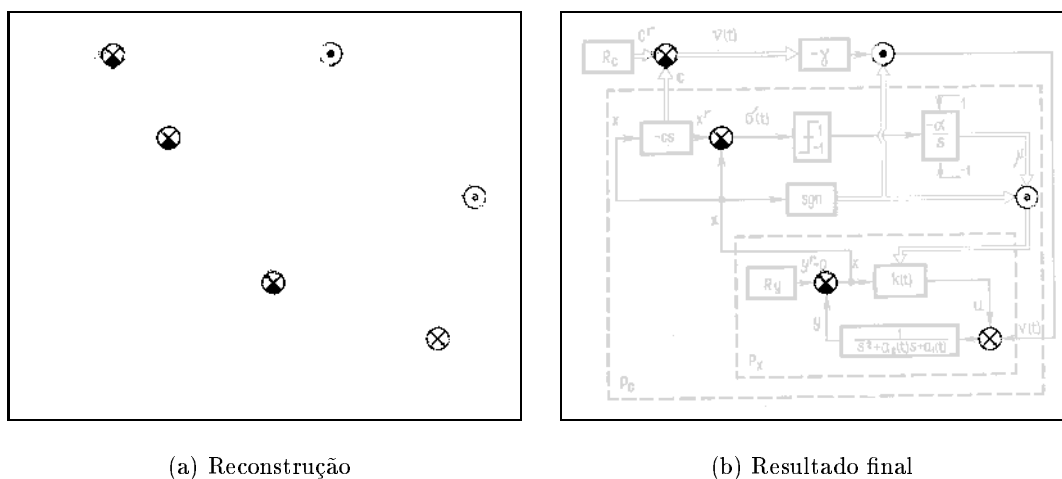


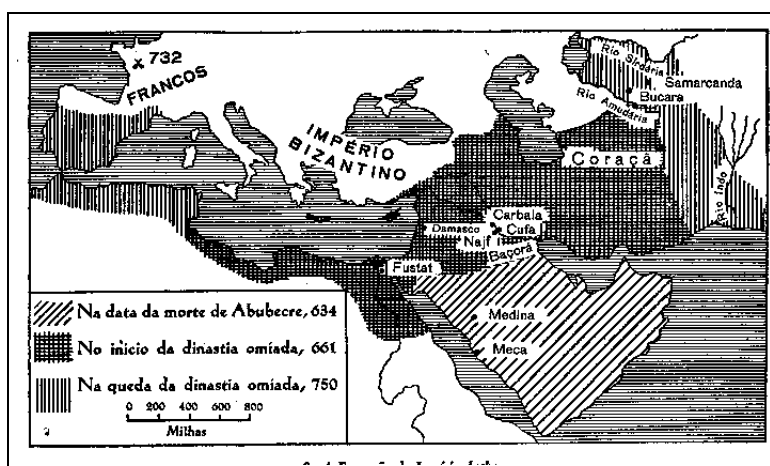
Figura 9.9: Reconhecimento de padrão em diagramas: resultado da reconstrução (resultado final) e o mesmo sobreposto à imagem original, respectivamente.

Resultados obtidos para outras imagens de teste são apresentados no apêndice A.

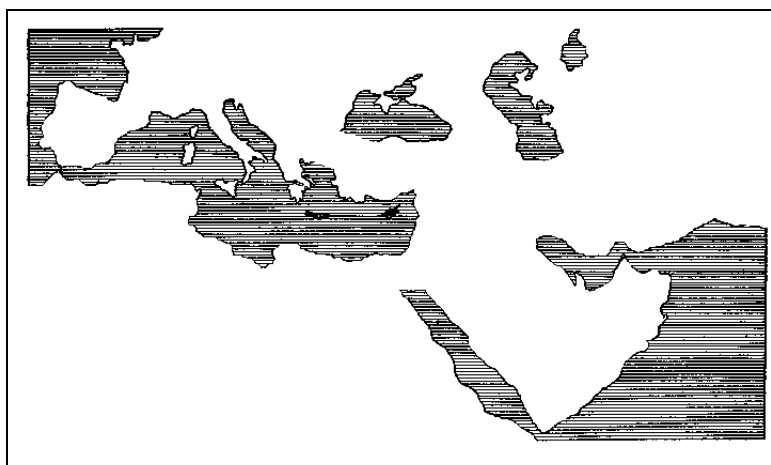
⁴Reconstrução: veja, por exemplo, [11], pág.139.

9.3 Reconhecimento de Textura

Uma textura pode caracterizar um objeto ou uma região de interesse nas imagens. Quando estas são distinguíveis em imagens binárias, as técnicas apresentadas podem ser utilizadas para segmentá-las. As figuras 9.10 e 9.11 mostram pares de imagens de treinamento. O objetivo aqui é segmentar a região da imagem com listras horizontais. Observe que as listras não são perfeitas.



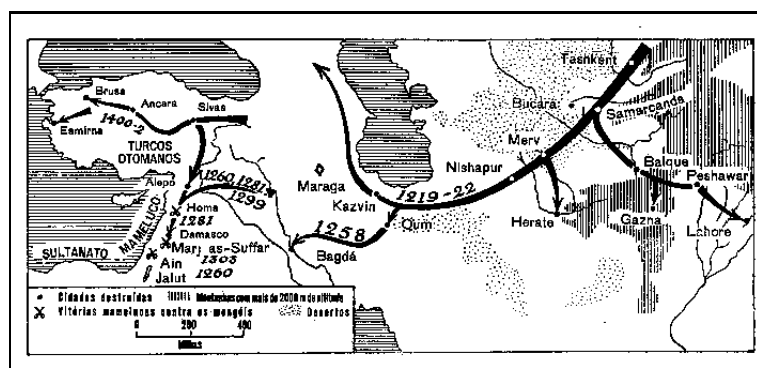
(a) Observada



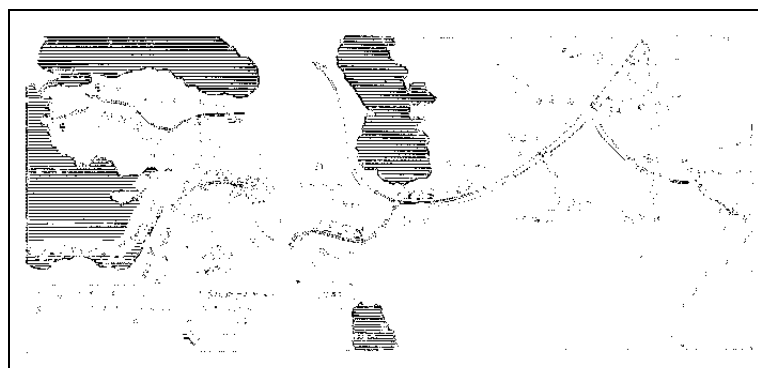
(b) Ideal

Figura 9.10: Imagem de treinamento para reconhecimento de textura.

A tabela 9.3 mostra os detalhes do experimento. O resultado do operador de duas iterações (ambos anti-extensivos) são mostrados nas figuras 9.12 e 9.13.



(a) Teste

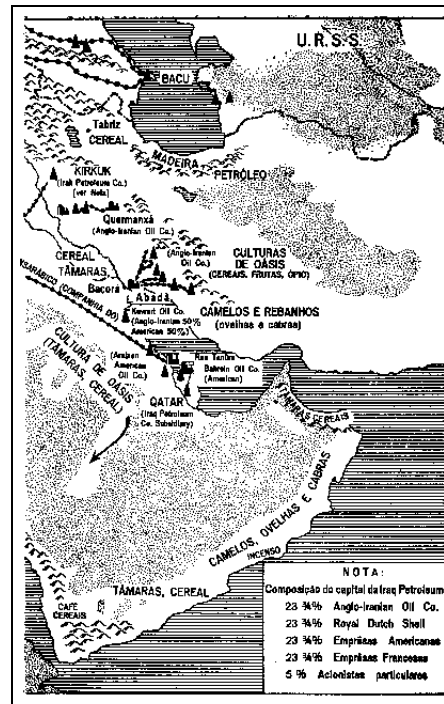


(b) Resultado da iteração 1

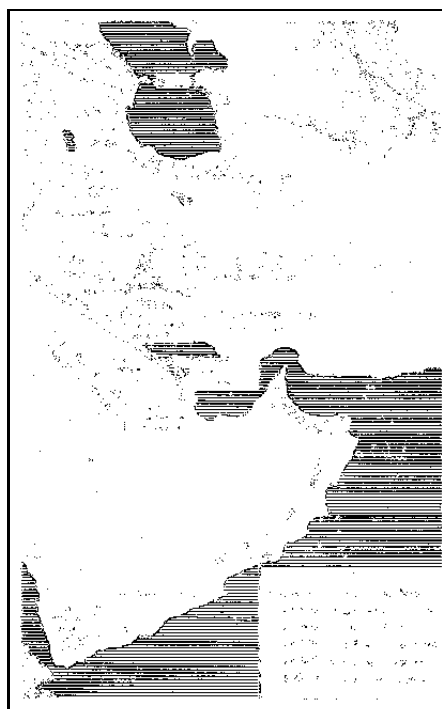


(c) Resultado da iteração 2

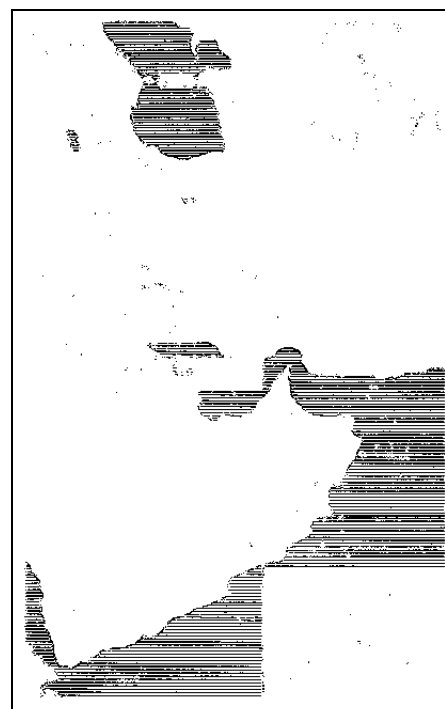
Figura 9.12: Reconhecimento de textura: imagem teste 1, resultado da primeira e segunda iterações, respectivamente.



(a) Teste



(b) Resultado da iteração 1



(c) Resultado da iteração 2

Figura 9.13: Reconhecimento de textura: imagem teste 2, resultado da primeira e segunda iterações, respectivamente.

9.4 Reconhecimento de Caracteres

Reconhecimento de caracteres é um problema clássico na área de processamento de documentos. Páginas escolhidas aleatoriamente de um livro foram digitalizadas através de um “scanner”, com resolução de 200dpi. Estas imagens foram binarizadas por uma operação de limiarização simples e em seguida seu tamanho foi reduzido pela metade. Uma imagem ideal, consistindo de todas os caracteres “a” minúsculos (com e sem acentuação), e outra consistindo de todas os caracteres “s” minúsculos foram geradas para cada imagem.

Para o reconhecimento do caractere “a” foi projetado um operador de 2-iterações (ambos anti-extensivos), sobre as janelas $W_{9 \times 7}$ e $W_{7 \times 5}$, respectivamente. Para projetar o primeiro operador foram utilizadas 8 imagens e para projetar o segundo, além das 8, foram utilizadas mais duas imagens.

Problema: Reconhecimento de caractere “a”		
	Iteração 1	Iteração 2
Propriedade do operador	anti-extensivo	anti-extensivo
janela	$W_{9 \times 7}$	$W_{7 \times 5}$
Função de perda	MAE	MAE
Algoritmo	ISI com $p = 8$ (nmax=30000) e $k = 25$	ISI com $k = 25$
Imagens de treinamento	8 pares	8 pares
Intervalos	1012	99
Tempo de treinamento	12438 segundos	7.65 segundos

Tabela 9.4: Reconhecimento de caracteres “a”: detalhes do treinamento.

A figura 9.14 mostra o resultado da primeira e segunda iterações, respectivamente, para o reconhecimento do caractere “a”, em uma página de teste. O resultado está sobreposto sobre a imagem observada para melhor visualização. O operador da primeira iteração gera marcadores para todas as ocorrências do caractere “a”. No entanto, aparecem alguns pequenos marcadores sobre outros caracteres. A maior parte destes pequenos marcadores são filtrados pelo operador da segunda iteração, enquanto os marcadores sobre os caracteres de interesse são mantidos. Daqui em diante, não mostraremos o resultado do operador da primeira iteração, mas apenas os da segunda iteração.

A figura 9.15 mostra o resultado para uma outra página de teste. Os círculos sobre a imagem destacam os erros, i.e., caracteres que foram marcados pelo operador projetado mas que não são os caracteres de interesse.

O treinamento foi repetido para o reconhecimento do caractere “s” (minúsculo) nestas mesmas imagens e também em um segundo livro. Os resultados são apresentados no apêndice A.

Para abranger as dimensões deste novo universo teórico, a *Análise Macroeconômica* se desdobraria em dois conjuntos principais: *Teoria dos Agregados* e *Teoria Geral do Equilíbrio e do Crescimento*. No campo da Teoria dos Agregados são conceituados e calculados os principais indicadores do desempenho da economia como um todo: o Produto Nacional e a Renda Nacional, bem como cada um de seus principais componentes, são aqui definidos e avaliados através de processos especiais de medição e de aferição da atividade econômica global. No campo da Teoria Geral do Equilíbrio e do Crescimento são

(a) Resultado da iteração 1

dagem do sistema econômico a partir de seus agentes individuais — consumidores e produtores —. KEYNES enveredou para a análise de conceitos agregados, como a Renda Nacional, o consumo, a poupança e o investimento globais, os volumes das exportações e das importações, os dispêndios e as receitas totais do governo. Os níveis dos preços, os volumes do emprego, o suprimento de moeda passaram a ser vistos globalmente. A década de 30 assistiria, assim, à passagem da Economia individualista e da empresa para a Economia agregativa, considerada a partir de suas magnitudes globais.

(b) Resultado da iteração 2

Figura 9.14: Reconhecimento do caractere “a”: resultado da primeira e segunda iterações, respectivamente.

Ao longo deste percurso há, entretanto, um ponto de maximização do lucro. Segundo se observa no modelo numérico da Tabela 11.7, ele está situado no intervalo de 1 600 a 1 800 unidades produzidas. Nesse ponto, será observada a máxima distância positiva entre a receita e os custos totais. As demonstrações gráficas da Figura 11.7 permitem a clara visualização desse ponto. No gráfico (a) estão representadas as curvas do custo total e da receita total. Esta última, para a empresa perfeitamente competitiva, é identificada por uma reta que passa pela origem. A perfeita elasticidade da curva da procura

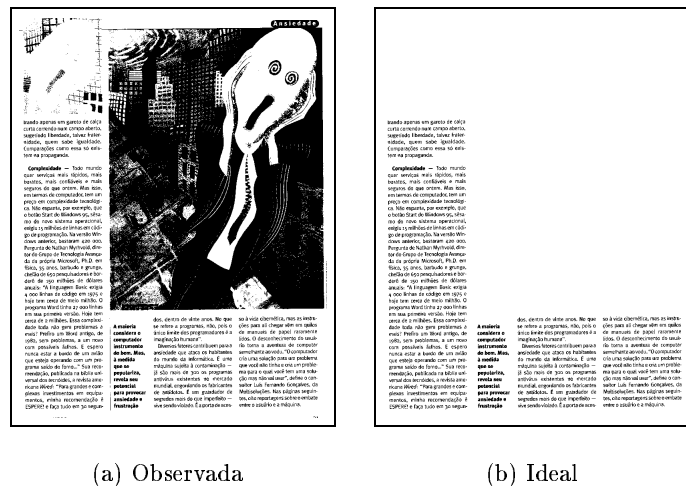
(a) Resultado da iteração 2

Figura 9.15: Reconhecimento do caractere “a”: resultado do operador projetado.

9.5 Segmentação de Texto

Uma página de documento (livro, revista, etc) contém, em geral, objetos tais como texto, figuras, tabelas, fotos, entre outros. A tarefa de um sistema de reconhecimento de caracteres pode ser facilitada através de um pré-processamento que segmenta as regiões da imagem que correspondem ao texto, das demais regiões [130, 58, 100, 4, 28]. Nesta seção mostramos o uso das técnicas propostas nesta tese para projetar operadores para segmentação de texto.

Algumas páginas de uma revista foram digitalizadas a 200dpi e binarizadas através de uma limiarização simples. Em seguida elas foram reduzidas em tamanho de modo a se obter imagens correspondentes a 100 dpi aproximadamente. Um total de cinco pares de imagens (como as ilustradas na figura 9.16) foram utilizadas para projetar um operador de duas iterações. Uma vez que a imagem



(a) Observada

(b) Ideal

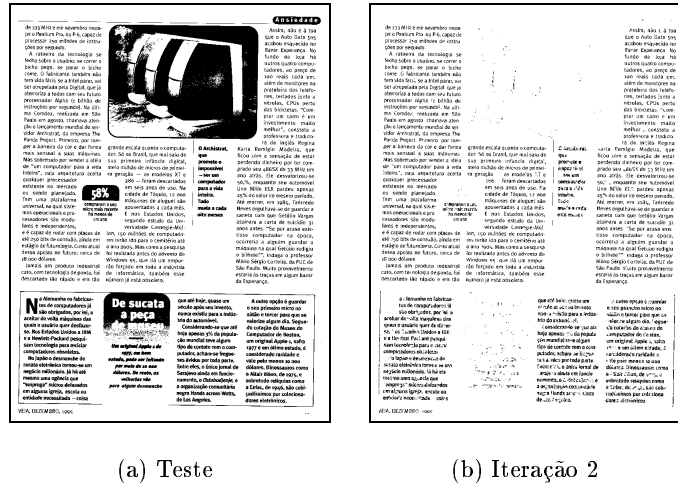
Figura 9.16: Um par de imagem de treinamento para segmentação de texto.

ideal é necessariamente um subconjunto da imagem observada, os operadores foram projetados de modo a satisfazer a propriedade anti-extensiva. O primeiro operador foi projetado sobre a janela $W_{7 \times 5}$ (usando as 4 primeiras imagens), e o segundo sobre a janela W_{21} (usando o resultado do primeiro operador para as 4 últimas imagens).

Problema: segmentação de texto		
	Iteração 1	Iteração 2
Propriedade do operador	anti-extensivo	anti-extensivo
janela	$W_{7 \times 5}$	W_{21}
Função de perda	MAE	MAE
Algoritmo	ISI com $p = 14$ e $k = 15$	ISI com $p = 2$ e $k = 15$
Imagens de treinamento	4/5 pares	4/5 pares
Intervalos	4992	1328
Tempo de treinamento	25004 segundos	1179 segundos

Tabela 9.5: Segmentação de Texto: detalhes do treinamento.

A figura 9.17(b) mostra o resultado do operador para uma particular imagem de teste (figura 9.17(a)).



(a) Teste

(b) Iteração 2

Figura 9.17: Segmentação de texto : (a) imagem (teste) observada e (b) resultado do operador projetado.

A densidade de pixels acesos é muito maior nas áreas correspondentes ao texto do que nas demais áreas. Portanto, um filtro de pós-processamento que aproveita esta característica foi projetado heurísticamente. O filtro consiste de :

1. filtro de ordem⁵ 18 com relação a janela $W_{7 \times 20}$,
2. fechamento⁶ por uma linha vertical de comprimento 10,
3. operador *preenche buracos*⁷, e
4. abertura por área⁸ (supomos que objetos com área menor que um dado valor não correspondem a região de texto).

Verificamos que os parâmetros dos dois primeiros filtros é globalmente ajustável (isto é, um mesmo valor para o conjunto de todas as imagens de um mesmo domínio). No entanto, o parâmetro da abertura por área precisa ser manualmente ajustado para cada imagem.

A figura 9.18 mostra os resultados destes filtros para uma imagem de teste. O resultado do filtro de pós-processamento foi satisfatório para todas as páginas de teste (um total de 15 páginas). No entanto, como já comentamos anteriormente, o parâmetro do filtro de abertura por área teve de ser ajustado manualmente, página por página. Além disso, pequenas áreas de texto (como por exemplo, a página) foram perdidas em alguns casos por se tratar de uma área menor do que uma área correspondente a ruído.

⁵Filtro de ordem: veja, por exemplo, [75], pág.98.

⁶Fechamento: veja, por exemplo, [147], pág. 50.

⁷Preenche buracos: veja, por exemplo, [11], pág. 202.

⁸Abertura por área: veja, por exemplo, [142]

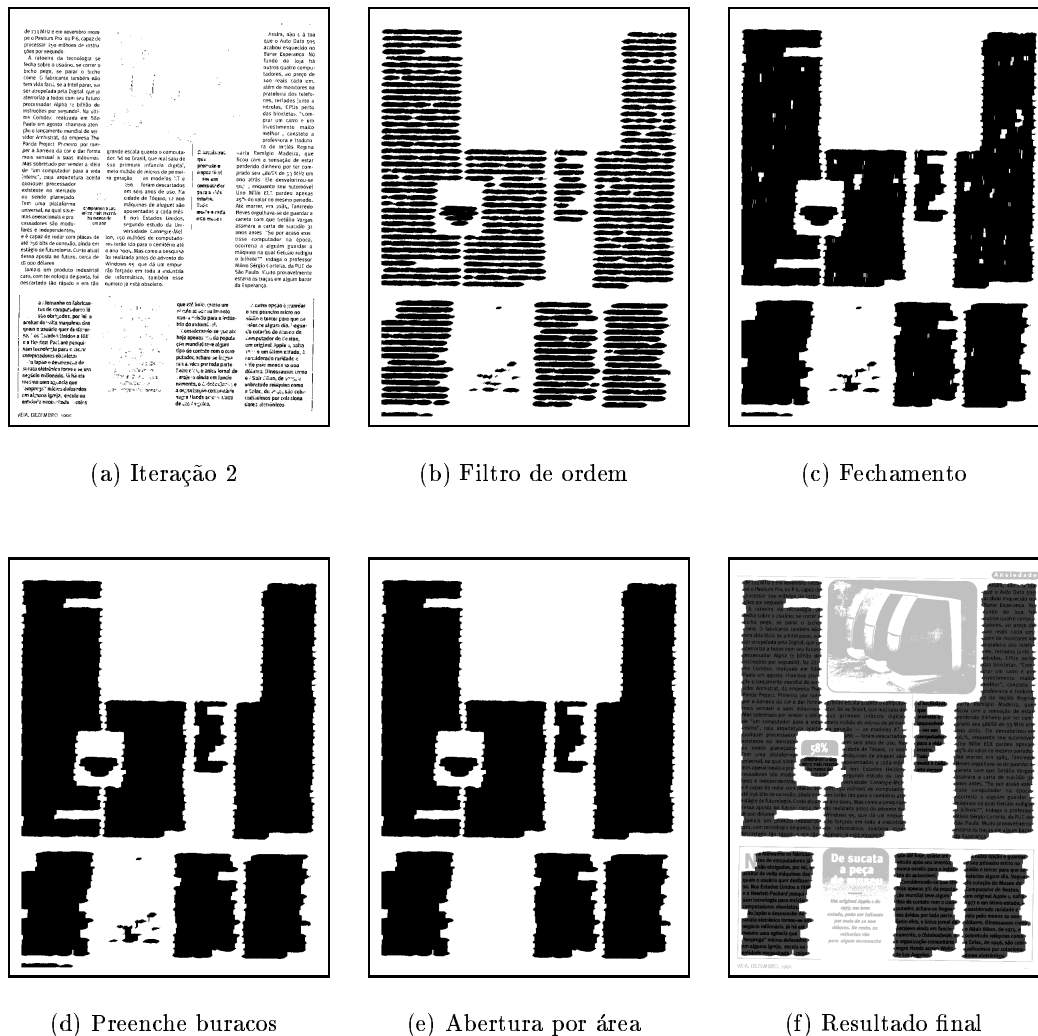


Figura 9.18: Efeitos do filtro de pós-processamento para segmentação de texto: resultado do (a) operador, (b) filtro de ordem, (c) fechamento vertical, (d) preenche buracos, (e) abertura por área e (f) resultado final sobreposto à imagem observada.

Um outro filtro foi utilizado para gerar os marcadores para as componentes que correspondem às áreas de texto, com o intuito de maximizar a automatização do processo de segmentação. Para as imagens de teste, este filtro funcionou com os mesmos parâmetros para 14 das 15 imagens de teste. O filtro corresponde aos seguintes subfiltros :

- abertura por área (área 3),
- Fechamento seguido de abertura por uma linha horizontal de comprimento 3,
- Fechamento seguido de abertura por uma linha vertical de comprimento 5,
- Fechamento seguido de abertura por uma linha horizontal de comprimento 12

O resultado deste filtro foi usado como marcador para reconstruir o resultado do operador preenche buracos do filtro anterior. A figura 9.19 mostra os resultados intermediários deste filtro para uma particular imagem de teste.

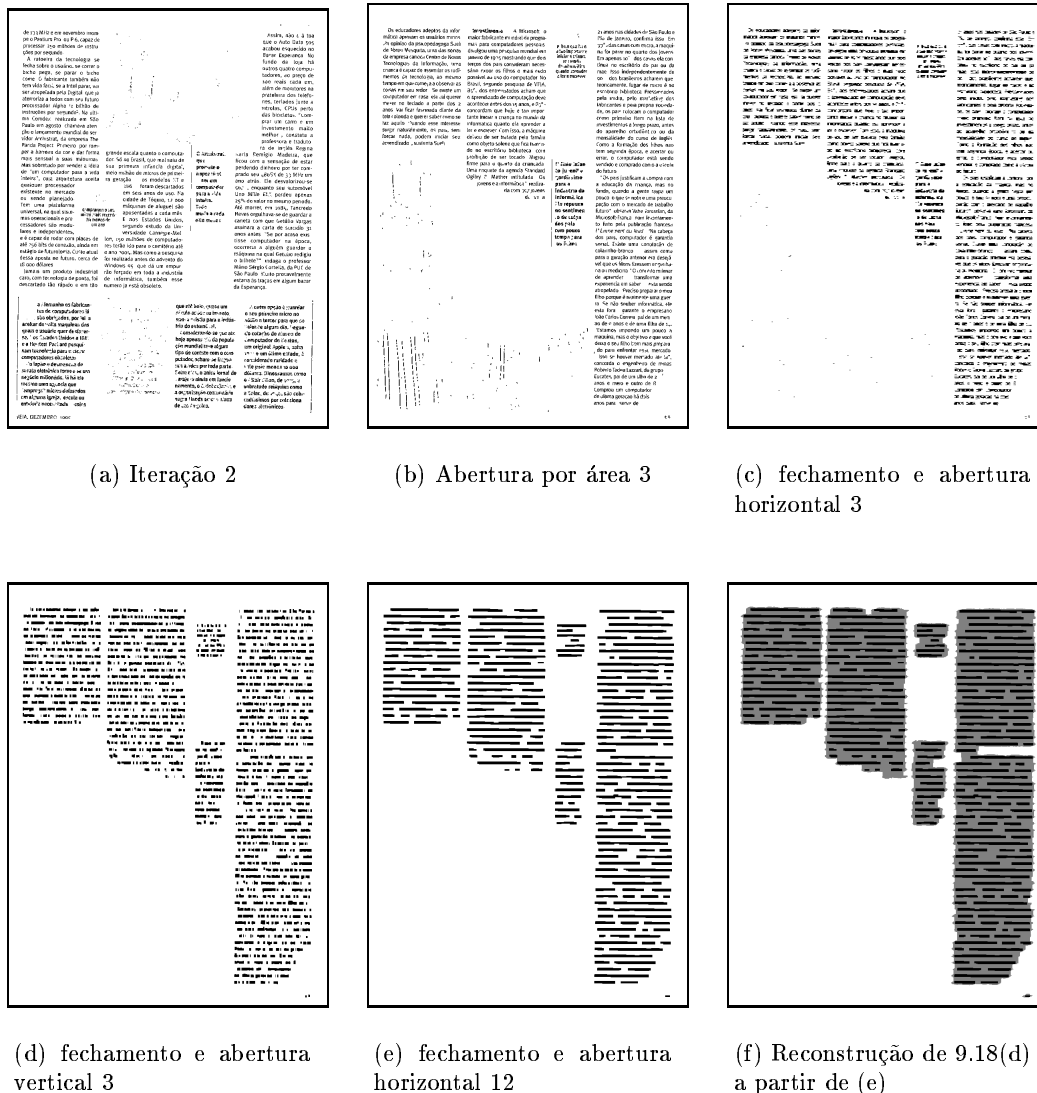


Figura 9.19: Efeitos do filtro para geração de marcadores para a segmentação de texto.

Um segundo grupo de imagens foi obtido digitalizando-se algumas páginas de um livro a 300dpi. Estes também foram reduzidos de forma a obtermos imagens correspondentes a 100dpi. Um total de 8 imagens de treinamento foram utilizadas para projetar um operador de três iterações, sobre as janelas $W_{7 \times 5}$, $W_{5 \times 9}$ e $W_{7 \times 5}$, respectivamente. Na primeira iteração foram utilizados 6 pares de imagens de treinamento e nas demais foram utilizados todos os 8 pares. O mesmo filtro de pós-processamento utilizado para o primeiro grupo de imagens também foi aplicado para as imagens deste grupo. A figura 9.20 mostra o resultado para uma página de teste.

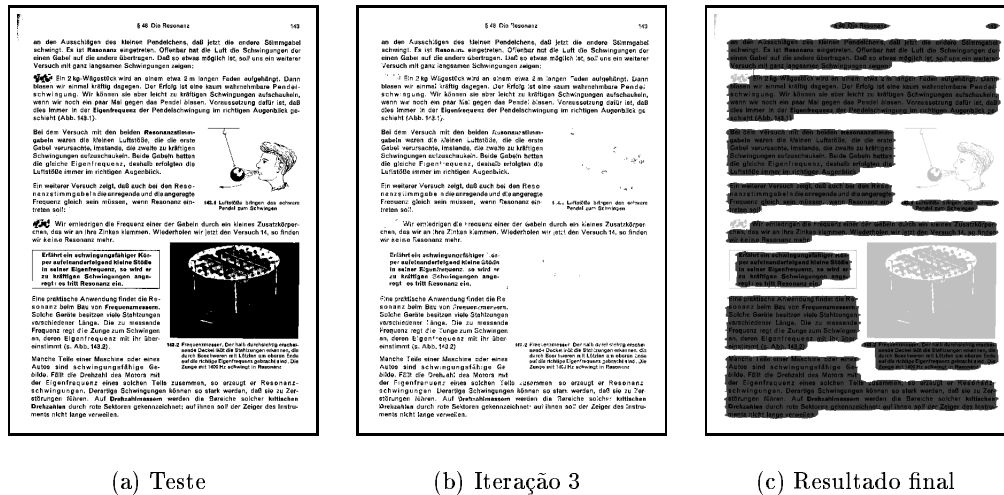


Figura 9.20: Segmentação de texto : imagem observada, resultado do operador e resultado final, respectivamente.

Resultados para várias imagens de teste, dos dois grupos de imagens, são apresentados no apêndice A.

Capítulo 10

Conclusões

Neste trabalho estudamos o problema de projeto de W -operadores, com ênfase sobre a questão da precisão de operadores projetados a partir de uma quantidade limitada de exemplos de treinamento. Algumas técnicas que exploram conhecimentos a priori para projetar operadores mais precisos e algoritmos eficientes foram propostos.

Uma técnica de projeto baseada no modelo de aprendizado PAC foi utilizada como ponto de partida para as investigações [18, 15]. Formas para contornar as limitações desta técnica, quais sejam, a imprecisão dos operadores projetados (devida ao número limitado de exemplos de treinamento) e a complexidade de tempo dos algoritmos de aprendizado, constituíram o objetivo deste estudo.

10.1 Resumo das Contribuições

Vimos que a utilização de conhecimentos sobre o problema que desejamos resolver, quando adequadamente modelados, permitem a redução do espaço de operadores e, portanto, podem ser explorados para projetar operadores mais precisos. Para que estes conhecimentos sejam efetivamente utilizados, as estratégias de busca devem ser capazes de explorar as estruturas e propriedades dos espaços de hipóteses bem como aproveitar características das imagens consideradas. Além disso, elas devem estar apoiadas em algoritmos de busca eficientes.

Introduzimos **modificações no algoritmo ISI** para torná-lo eficiente e conseguimos redução de tempo da ordem de 10 em relação as implementações anteriores. As principais modificações introduzidas são: (1) a sua generalização para trabalhar com a extração de um conjunto de elementos (intervalo) em uma iteração em vez de um único elemento, (2) a introdução de um parâmetro que define o passo para cálculo de cobertura mínima e (3) a paralelização do algoritmo. Apresentamos também uma prova formal de sua correteude.

A técnica de **projeto iterativo** [84, 80] (vista no capítulo 5) foi investigada como forma para projetar operadores sobre janelas grandes a partir da composição de operadores projetados sobre janelas menores. Verificamos que se os exemplos de treinamento forem distribuídos de forma adequada entre as iterações em vez de todos eles serem utilizados em cada uma das iterações, os operadores resultantes podem ser mais precisos. A técnica de projeto iterativo pode ser vista como um caso particular do projeto multi-estágio. Cada estágio adicional corresponde a um refinamento do operador em termos de precisão.

O uso de operadores projetados a priori [50, 14, 51] (capítulo 6) pode ser também visto no mesmo contexto de refinamento de solução. A técnica proposta consiste em melhorar a precisão de um operador a partir de exemplos de treinamento. Introduzimos um fator de confiança, controlado pelo usuário, que indica quanta evidência os exemplos de treinamento devem fornecer para que o valor do operador em um determinado ponto seja alterado. A técnica pode ser vista como uma generalização dos “differencing filters” [54], onde o operador a priori é o operador identidade. Estudamos também questões ligadas à escolha de uma janela e mostramos que uma escolha inadequada pode ser crítica.

O projeto multi-estágio é interessante se a técnica de refinamento utilizada garante que a cada estágio o operador obtido é mais preciso (em termos de erro) do que o operador do estágio anterior. A escolha de um operador inicial adequado, de preferência próximo da solução ótima, é importante para a obtenção de um operador preciso.

Apresentamos um **novo algoritmo eficiente** [81, 82] para projetar operadores crescentes ótimos, a partir de uma formulação nova do problema. A grande vantagem do algoritmo proposto é que ele depende apenas dos padrões observados nas imagens de treinamento e portanto pode ser aplicado para projetar operadores sobre janelas relativamente grandes. Os outros algoritmos existentes são formulados de tal forma que dependem de todos os $2^{|W|}$ possíveis padrões dentro de uma janela W . Logo eles são extremamente ineficientes para janelas relativamente grandes.

Uma consequência natural deste algoritmo foi a utilização do mesmo para o **projeto de filtros “stack”** [78], pois estes são caracterizados por operadores crescentes binários. Nos estudos envolvendo filtros “stack”, unificamos as notações utilizadas no contexto de morfologia matemática e fora dele. Mostramos também que a formulação que considera cada um dos cortes da imagem em níveis de cinza como um conjunto aleatório e a que considera a união dos cortes como um único conjunto aleatório são equivalentes sob o ponto de vista de minimização do erro MAE de um filtro stack. Como existem poucos trabalhos publicados sobre filtros stack no contexto de morfologia matemática, o estudo apresentado neste trabalho é uma contribuição neste sentido.

Uma abordagem Bayesiana [83] foi utilizada no contexto de estimação dos custos de chaveamento para projeto de operadores crescentes (seção 7.4). As probabilidades condicionais foram consideradas variáveis aleatórias com função de densidade Beta. Constatamos que em situações nas quais as distribuições a priori das probabilidades condicionais são conhecidas, os resultados obtidos são notadamente melhores.

Quanto mais e mais conhecimentos são considerados, a estruturação do espaço de hipóteses bem como as técnicas de busca tendem a tornar-se mais específicas e complexas. Neste trabalho tentamos, dentro do processo de incorporar conhecimentos, manter uma preocupação pela generalidade das técnicas. Por exemplo, no caso do método de chaveamentos, a técnica para encontrar o conjunto de chaveamentos ótimos é específica para cada tipo de restrição, porém o conceito de chaveamento não depende da particular restrição considerada.

A sintetização e análise dos diversos aspectos relacionados com projeto de W -operadores, em um único material, é também uma das contribuições desta tese.

Resultados práticos

As técnicas propostas foram implementadas¹, testadas e aplicadas para resolver problemas reais de processamento de imagens tais como segmentação de texto [79], reconhecimento de caracteres [19], reconhecimento de objetos em diagramas funcionais, reconhecimento de padrões e filtragem. As publicações decorrentes deste trabalho de pesquisa estão listadas no apêndice B.

10.2 Comentários Finais

Para que as técnicas propostas possam ser aplicadas adequadamente, é preciso entendermos qual é o mecanismo comum a estas técnicas que são importantes para o aumento da precisão do operador projetado. Sob a perspectiva apresentada na última seção do capítulo 3, as restrições impostas ou as informações oriundas dos conhecimentos a priori são importantes para definir um subespaço de operadores candidatos. Entender qual é o espaço de candidatos associado a cada técnica é fundamental para a escolha de uma técnica adequada para um particular problema.

Além disso, a partir do fato de que conhecimentos sobre os problemas podem ser utilizados para reduzir o espaço de candidatos (ou, equivalentemente, preencher um maior número de linhas da tabela-verdade que definem as funções características), novas técnicas para gerar operadores mais precisos podem ser desenvolvidos.

A extensão do método de chaveamentos para o projeto de operadores auto-duais, idempotentes e outros, assim como a do projeto multi-estágio para projetos não necessariamente iterativos e a do projeto de operadores binários para operadores níveis de cinza [40, 86, 88] são alguns dos passos futuros desta pesquisa. Outras questões que merecem ser investigadas são vários aspectos do problema, específicos ou genéricos, tais como a robustez dos operadores projetados, a questão da escolha de uma janela ótima para uma quantidade fixa de dados de treinamento, a utilização de algoritmos paralelos, a implementação eficiente dos operadores projetados [113], uso de técnicas multi-resolução, modelagem de dados, decomposição funcional dos operadores, complexidade de exemplos de treinamento, entre outros.

Desafios muito maiores aparecem quando pensamos em modelagens que sejam capazes de combinar diferentes tipos de conhecimentos e também técnicas que sejam capazes de explorar propriedades das imagens ideais (e não apenas das imagens observadas). Devemos notar também que a complexidade computacional no caso de projeto de operadores não-binários é muito maior, assim como a imprecisão decorrente da quantidade limitada de exemplos de treinamento. Portanto, o uso de conhecimentos a priori ocupa um papel ainda mais relevante neste caso.

Finalmente, um objetivo das pesquisas futuras nesta área deve ser a busca de uma formulação que possibilite a integração das técnicas existentes e as que venham a ser desenvolvidas. Acreditamos que o presente trabalho traz algumas contribuições neste sentido.

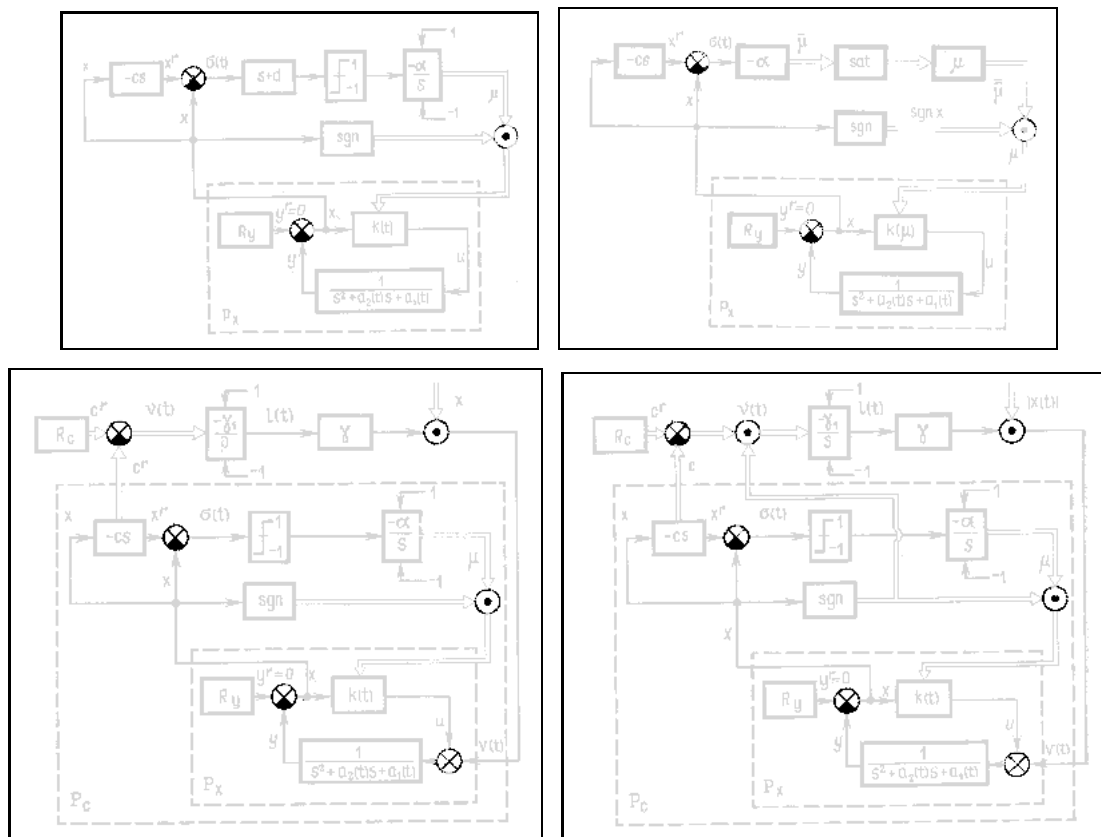
¹Todos os programas computacionais foram implementados no sistema Khoros [104], com a preocupação de mantê-los independentes de plataforma. Mais precisamente, apenas a parte de interface dos programas depende desse sistema. Todos os programas foram escritos em linguagem C padrão e atualmente constituem, excluindo a parte correspondente aos códigos de interface e juntamente com rotinas relativas ao projeto de operadores em níveis de cinza, uma biblioteca de rotinas com aproximadamente 45000 linhas de código e documentação.

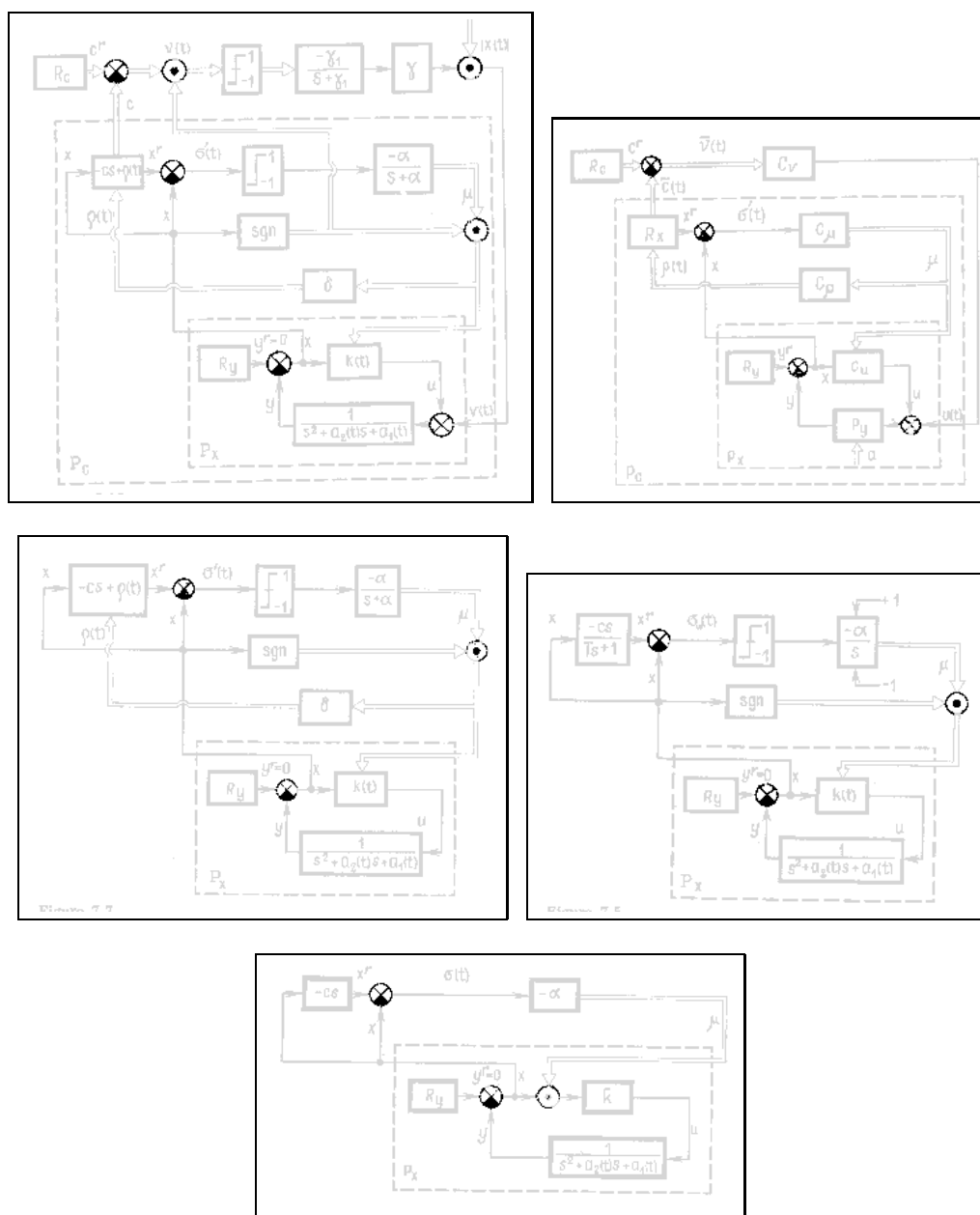
Apêndice A

Resultados (Imagens) Adicionais das Aplicações

Este apêndice é um complemento ao capítulo 9. Para as aplicações de reconhecimento de somadores/multiplicadores em diagramas funcionais, reconhecimento de caracteres e segmentação de texto apresentados naquele capítulo, mostramos os resultados obtidos para várias imagens de teste.

A.1 Reconhecimento de Padrões em Diagramas





A.2 Reconhecimento de Caracteres

As imagens a seguir mostram os resultados obtidos pelo operador projetado para reconhecimento do caractere “s” em imagens de páginas de um livro. O círculo sobre a imagem destaca um erro.

dagem do sistema económico a partir de seus agentes individuais — consumidores e produtores —. KEYNES encredeou para a análise de conceitos agregados, como a Renda Nacional, o consumo, a poupança e o investimento globais, os volumes das exportações e das importações, os dispêndios e as receitas totais do governo. Os níveis dos preços, os volumes do emprego, o suprimento de moeda passaram a ser vistos globalmente. A década de 30 assistiria, assim, à passagem da Economia individualista e da empresa para a Economia agregativa, considerada a partir de suas magnitudes globais.

Ao longo deste percurso há, entretanto, um ponto de maximização do lucro. Segundo se observa no modelo numérico da Tabela 11.7, ele está situado no intervalo de 1 600 a 1 800 unidades produzidas. Nesse ponto, será observada a máxima distância positiva entre a receita e os custos totais. As demonstrações gráficas da Figura 11.7 permitem a clara visualização desse ponto. No gráfico (a) estão representadas as curvas do custo total e da receita total. Esta última, para a empresa perfeitamente competitiva, é identificada por uma reta que passa pela origem. A perfeita elasticidade da curva da procura

Os resultados obtidos pelo operador treinado para um segundo livro são mostradas a seguir. O círculo sobre a imagem destaca um erro, enquanto os retângulos destacam o reconhecimento do caractere quando este aparece em diferente tipo de fonte (itálico ou “small cap”).

manifesta através de um ténue véu que mal esconde o frio e calculado propósito. Quem procurar entender o introvertido, convencer-se-á facilmente de que só com muito esforço ele pode dominar e vigiar sua paixão tumultuosa, mediante uma aparência calma e razoável.

Ambos os critérios são verdadeiros e falsos. O critério é falso quando o ponto de vista consciente, ou melhor, quando a consciência possui, simplesmente, perante o inconsciente, a força e a capacidade de resistência necessárias; e é verdadeira

pela Igreja. A condenação definitiva foi póstuma, decerto, uma vez que, já velho e alquebrado pelo martírio que sofreu durante a perseguição de Décio, morreu pouco depois, em consequência das torturas. Foi condenado no ano de 399 pelo Papa ANASTASIO I e, em 543, um sínodo convocado por EVENCANO ratificou com maldição a sua heresia, sentença que foi respeitada por todos os concílios posteriores.

ORIGENES é um representante clássico do tipo extrovertido. Sua orientação fundamental dirige-se ao objeto; isto

A.3 Segmentação de Texto

[illegible][illegible]

83%
de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea.

CONTEXTO

El uso de Internet en España ha crecido de forma espectacular en los últimos meses. Según un estudio realizado por el Observatorio de las Comunicaciones de la Universidad de Valencia, el 83% de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea. Este dato refleja el creciente interés de los españoles por acceder a la información a través de Internet.

El estudio también revela que el 75% de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea. Este dato refleja el creciente interés de los españoles por acceder a la información a través de Internet.

El estudio también revela que el 75% de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea. Este dato refleja el creciente interés de los españoles por acceder a la información a través de Internet.

83%
de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea.

CONTEXTO

El uso de Internet en España ha crecido de forma espectacular en los últimos meses. Según un estudio realizado por el Observatorio de las Comunicaciones de la Universidad de Valencia, el 83% de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea. Este dato refleja el creciente interés de los españoles por acceder a la información a través de Internet.

El estudio también revela que el 75% de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea. Este dato refleja el creciente interés de los españoles por acceder a la información a través de Internet.

El estudio también revela que el 75% de los usuarios de Internet en España dicen que el uso de Internet les ayuda a encontrar información sobre el mundo que les rodea. Este dato refleja el creciente interés de los españoles por acceder a la información a través de Internet.

Asamblea de los Estados Unidos que una de las reuniones más importantes de la historia se celebró en la ciudad de Nueva York, en la sede de la Organización de las Naciones Unidas. En esta ocasión, los representantes de los Estados Unidos, de México y de Colombia se reunieron para discutir la posibilidad de que los Estados Unidos, México y Colombia se unieran para formar una nueva nación, la Unión Americana. Esta unión se celebró en la ciudad de Nueva York, en la sede de la Organización de las Naciones Unidas. En esta ocasión, los representantes de los Estados Unidos, de México y de Colombia se reunieron para discutir la posibilidad de que los Estados Unidos, México y Colombia se unieran para formar una nueva nación, la Unión Americana.

El primer paso fue la creación de la Asamblea de los Estados Unidos, que se celebró en la ciudad de Nueva York, en la sede de la Organización de las Naciones Unidas. En esta ocasión, los representantes de los Estados Unidos, de México y de Colombia se reunieron para discutir la posibilidad de que los Estados Unidos, México y Colombia se unieran para formar una nueva nación, la Unión Americana. Esta unión se celebró en la ciudad de Nueva York, en la sede de la Organización de las Naciones Unidas. En esta ocasión, los representantes de los Estados Unidos, de México y de Colombia se reunieron para discutir la posibilidad de que los Estados Unidos, México y Colombia se unieran para formar una nueva nación, la Unión Americana.

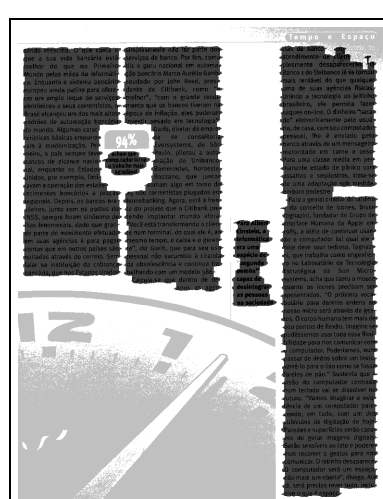
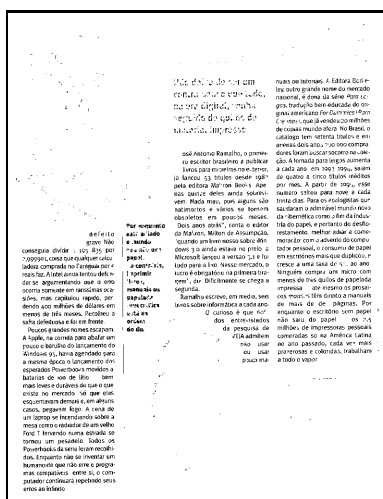
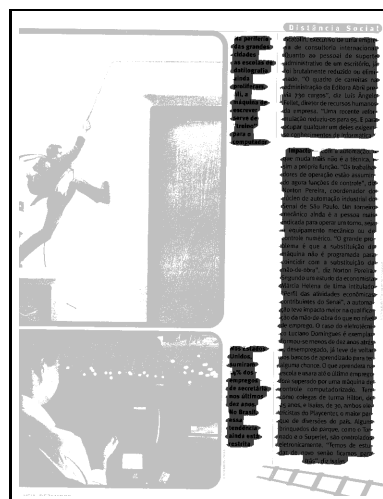
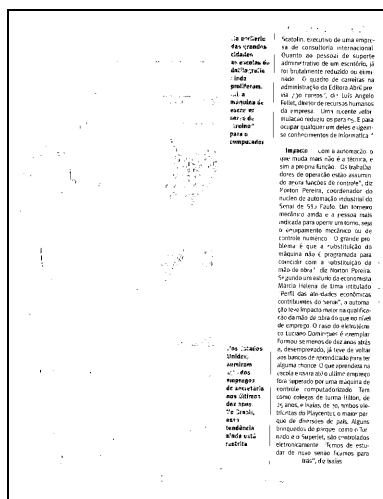
El primer paso fue la creación de la Asamblea de los Estados Unidos, que se celebró en la ciudad de Nueva York, en la sede de la Organización de las Naciones Unidas. En esta ocasión, los representantes de los Estados Unidos, de México y de Colombia se reunieron para discutir la posibilidad de que los Estados Unidos, México y Colombia se unieran para formar una nueva nación, la Unión Americana. Esta unión se celebró en la ciudad de Nueva York, en la sede de la Organización de las Naciones Unidas. En esta ocasión, los representantes de los Estados Unidos, de México y de Colombia se reunieron para discutir la posibilidad de que los Estados Unidos, México y Colombia se unieran para formar una nueva nación, la Unión Americana.

127

[illegible][illegible][illegible]

88%
Source: Intel

[illegible][illegible][illegible]

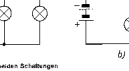


Os resultados de segmentação obtidos para o segundo grupo de imagens são mostrados a seguir.

234

I. Grundwissenschaften am Stromkreis

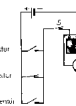
Vorl. Auch die beiden Schaltungen nach Abb. 234.1 mit einer Batterie als Stromquelle auf Wirkungsweise aus sich? Wo findet man sie im Haushalt?



234.1 Vergleiche die beiden Schaltungen.

V.12: Eine Klingellampe (Abb. 234.2) mit einer Tasterbetriebsartem nach! Suche die mögliche Stromstärke auf Weiche Bedienung bei der Schalter 52 Wege, wenn zwei Kontakte gleichzeitig geschlossen werden!

Lernziele des Stromkreises in einer Taschenlampe und vergleiche mit Abb. 234.1. Zwei, das Gehäuse ein Teil des Stromkreises sein.



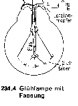
234.2 Strömung der elektrischen Ladung in der Taschenlampe.

V.12: Verfolge die möglichen Stromstärke bei der Klingellampe nach einem durchschalten.

Bei der **Strombahnke Ruch** nach Abb. 234.3 der Strom durch die Oberleitung und über den Stromabnehmer zum Ende des Gleises von dort zu den Häfen. Die Schein leuchtende Rückleitung.

3. Gefahren des elektrischen Stroms, Endabsch.

Durch fauchte Hand (Vorarlach im Bad und beim Schwitzen) ersagt der Strom besonders leicht ins Innere unseres Körpers. Das kann sehr gefährlich werden und schon dann mitwirken, wenn man nur einen kleinen Lader oder einen Pol einer Steckdose berührt. Dies zeigt der folgende Versuch:



234.4 Glühbirne mit Fassung.

V.14: Wovon sind bestimmter Pol der Steckdose ohne einen Zusammenhang mit dem Wasser- oder Gasleitung verbunden so trachten die Lampe hell auf Glücken Versuche, dass es auf keinen Fall selbst ausbleiben. Am anderen Pol der Steckdose bleibt die Lampe dunkel dargen. Die, so am fährnde Leitung hat den Elektr-

234.5 Auch die Stromstrome benutzt zur Leuchten

[illegible][illegible]

250

10. Leitung und Spannung

Durch die Blitze in Gewitterwolken entstehen bequamen und ungelährlichen Weg. Schläge der Blitze in ein ungeschütztes Haus, so sucht er entlang zur Leiter, also längs der Wände – bis an die Giebel, der Kanne oder Antennen steht. Wie man deshalb in solchen Häusern bei Gewittern das Gefährdungswort

29A.2 Die Blitze führen gefährliche
Ladungen ins

29A.3 Die Blitzableiter trägt nur
eine Ladung

3. Schlägt der Blitz auf metallisch ungeschlossene Räume (wie aus dem Aussehen des Flugplatzes), so ordnen die Fliesen beim Schauen. Das zeigt bereits Faraday (engl. Phänomen 1781 bis 1867), wenn er sich in einem Metallgefäß setzte und von außen Funken aufzufangen ließ. Auch wenn er von innen das Metall berührte, ging keine Ladung auf ihn über. Die Ausstattung der folgenden Tabelle

Blitz Im Inneren befindet sich ein Metallgefäß, das durch einen Kondaktor aufgeführt. Mit einem zweiten Kondaktor kann man aus dem Inneren hermetisch Ladung entnehmen, auch nicht bei der Ladung. Die Ladung tritt auf der äußeren Oberfläche (Abb. 29A.3) ungelährlich auf eine geladene Kugel, die man ins Innere eines Faraday-Bereichers bringt, also Ladung ab. Dies ist nicht der Fall, wenn man die Außenfläche berührt.

4. Im **Bleisenderker** läuft ein endloses Gefäß aus zwei Wänden. Manches besteht aus einem Metallgefäß. Die Wände sind als Faraday-Bereich und sammeln die Ladungen an ihrer Oberfläche an. Die Ladungsentladung muß am Boden der Wände stattfinden. An der Spitze, z. B. im Inneren Blitz, wenn Teil der gesamten Ladungen ausgeht. Wegen der geringsten Influenzverfälschung, welche die Ladungsentladung unterbindet, können wir die Einzelheiten nicht genauer erklären. Dies gilt auch für die Prüfer und besetzte **metallische Maschine**.

29A.4 Faraday-Bereich


522

II. Ladung und Spannung


Durch die Leiter die Geleitzdrähte fließen, so suchen sie unangelegentlich weiter, schlägt der Blitz in einen ungelegentliches Haus, so sucht er entlang jeder Leiter, also längs der Wasser- und Gasrohre, der Kamine oder Antennen seiner Weg. Man würde deshalb in diesen Häusern bei einem Gewitter diese Geleitzdrähte

12. 2. 20. 1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112. 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126. 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140. 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154. 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168. 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182. 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196. 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210. 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224. 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238. 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252. 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266. 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280. 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294. 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308. 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322. 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336. 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350. 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364. 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378. 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392. 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406. 407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 420. 421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 434. 435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 448. 449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 462. 463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 476. 477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 490. 491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 504. 505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 518. 519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 532. 533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 546. 547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 560. 561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 574. 575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 588. 589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 602. 603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 616. 617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 630. 631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 644. 645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 658. 659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 672. 673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 686. 687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 700. 701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 714. 715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 728. 729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 742. 743. 744. 745. 746. 747. 748. 749. 750. 751. 752. 753. 754. 755. 756. 757. 758. 759. 760. 761. 762. 763. 764. 765. 766. 767. 768. 769. 770. 771. 772. 773. 774. 775. 776. 777. 778. 779. 780. 781. 782. 783. 784. 785. 786. 787. 788. 789. 790. 791. 792. 793. 794. 795. 796. 797. 798. 799. 800. 801. 802. 803. 804. 805. 806. 807. 808. 809. 810. 811. 812. 813. 814. 815. 816. 817. 818


schlich der Blitz in ein angesprochenes Haus, so sucht er entlang jeder Leiter, also Messing-, Eisen-, Wasser- und Gasrohr, die Kamine oder Antennen seiner Weg. Man muß deshalb in solchen Häusern bei einem solchen Gewitter alle Leitungswege




mit einem Blitzableiter versehen. Ein solches Blitzableiter-System besteht aus einem Leiter, der an einem Punkt am Haus befestigt ist, und einem Erdanker, der in den Boden reicht. Ein Pfeil zeigt den Stromfluss von der Leiter zum Erdanker.



mit einem Blitzableiter versehen. Ein solches Blitzableiter-System besteht aus einem Leiter, der an einem Punkt am Haus befestigt ist, und einem Erdanker, der in den Boden reicht. Ein Pfeil zeigt den Stromfluss von der Leiter zum Erdanker.



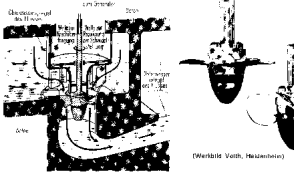
mit einem Blitzableiter versehen. Ein solches Blitzableiter-System besteht aus einem Leiter, der an einem Punkt am Haus befestigt ist, und einem Erdanker, der in den Boden reicht. Ein Pfeil zeigt den Stromfluss von der Leiter zum Erdanker.



mit einem Blitzableiter versehen. Ein solches Blitzableiter-System besteht aus einem Leiter, der an einem Punkt am Haus befestigt ist, und einem Erdanker, der in den Boden reicht. Ein Pfeil zeigt den Stromfluss von der Leiter zum Erdanker.

§15 Zusammensetzung und Zerlegung von Kräften

Figur stehenden Wassermengen reißt man die Schaufeln durch einen in die Achse eingebauten Mechanismus mehr oder weniger stark. Dem angesprochenen Gesetzwert zur Stromrichtung zeigt Abb. 209.3.



§16.1 Kaplan-Turbinen, Gesamtanzahl blinder bei 2 Stellungen der Schaufeln


Aufgaben:

- Zeichne an Hand von Ringen die Energiestromform, in die sich die von der Sonne zu uns kommende Energie umwandeln kann!
- Welche Leistung liefert eine Freistromturbine, die bei einem Gefälle von 150 m in 1 s von 5,4 m³ Wasser durchläuft, wenn eine Verluste über! Woran? 5, betragen also die Verluste in dem in Text angegebenen Beispiel?

§15 Zusammensetzung und Zerlegung von Kräften

1. Zusammensetzung von Kräften. Wenn zwei Kräfte (z. B. 25 kg und 18 kg) in gleicher Richtung am selben Punkt angreifen, addieren wir einfach ihre Beträge und erhalten als „Resultat“ hinter Kräfte die Resultierende 43 kg (Abb. 56.2).

Nach Abb. 56.3 ziehen zwei Hände am Arm eines Herrn nach verschiedenen Richtungen.



§16.2 Der rechte Winkel mit 60° und 30° bei der Resultierenden aus 25 kg und 18 kg des Glanzwerts

§16.3 Zusammensetzung von Kräften

[illegible][illegible]

§ 98 Messung der elektrischen Stromstärke

Fließt in der Zeit t die Ladung Q durch einen Querschnitt des Leiters, so berechnet man die Stromstärke I als Quotient aus der Ladung Q und der Zeit:

$$I = \frac{Q}{t} \quad (98.1)$$

Die Einheit der elektrischen Stromstärke ist 1 Ampere (A). Unter 1 A versteht man die Stromstärke, bei der 1 Coulomb in 1 s durch einen Querschnitt des Leiters fließt.

Nach § 95 bedeutet dies:

Ein Strom der Stärke 1 Ampere schaltet 0,775 cm³ Kupfaleit in 1 s ab (bei 0°C und 760 Torr) oder 1,118 mg Silber in 1 s:

$$1 \text{ A} = \frac{1}{1.118} \quad (98.2)$$

$$1 \text{ mA (Milliampere)} = \frac{1}{1000} \text{ A} \quad (98.3)$$

Fließen 10 C in 2 s, so ist die Stromstärke $I = Q/t = 10 \text{ C} / 2 \text{ s} = 5 \text{ A}$. In 20 s fließt bei diesem Strom die Ladung:

$$Q = I \cdot t = 5 \text{ A} \cdot 20 \text{ s} = 100 \text{ C} \quad (98.4)$$

2. Bei einer bestimmten Stromstärke fließt in der doppelten bzw. dreifachen Zeit die doppelte bzw. dreifache Ladung. Der Quotient $Q/t = 2Q/2t = 3Q/3t$ bleibt konstant. Bei größerer Stromstärke fließt dagegen in der gleichen Zeit mehr Ladung. Der Strom und die Ladung sind auch in Sekunden mehr Wärme und gibt eine stärkere magnetische Wirkung aus. Da sich der Hitzefaktor einer Vorrichtung, kann man das Hitzefaktorinstrument unmittelbar in Ampere (= Coulomb je Sekunde) eichen und als Strommesser verwenden. Das gleiche gilt für Maßgeräte auf magnetischer Grundlage (§ 84):

Wir schalten eine Knallgaszelle, ein Hitzefaktorinstrument und einen auf der magnetischen Stromwirkung beruhenden Strommesser hintereinander. Diese Geräte werden alle in der gleichen Zeit t von der gleichen Ladung Q durchflossen, unabhängig von der Reihenfolge. Die Stromstärke $I = Q/t$ ist in ihnen gleich; die Angabe der Instrumente kann also an der Gesamtwirkung nachgeprüft werden. Ein Strommesser gestattet mit einem Blick die Stromstärke zu messen, ohne daß man Ladung und Zeit einzeln bestimmt. Dies entspricht dem Tachometer im Auto, das ohne Messung von Weg (s) und Zeit (t) die Geschwindigkeit $v = s/t$ unmittelbar anzeigt.

Im unverzweigten Stromkreis ist die Stromstärke überall gleich groß.

§ 98 Messung der elektrischen Stromstärke

Fließt in der Zeit t die Ladung Q durch einen Querschnitt des Leiters, so berechnet man die Stromstärke I als Quotient aus der Ladung Q und der Zeit:

$$I = \frac{Q}{t} \quad (98.1)$$

Die Einheit der elektrischen Stromstärke ist 1 Ampere (A). Unter 1 A versteht man die Stromstärke, bei der 1 Coulomb in 1 s durch einen Querschnitt des Leiters fließt.

Nach § 95 bedeutet dies:

Ein Strom der Stärke 1 Ampere schaltet 0,775 cm³ Kupfaleit in 1 s ab (bei 0°C und 760 Torr) oder 1,118 mg Silber in 1 s:

$$1 \text{ A} = \frac{1}{1.118} \quad (98.2)$$

$$1 \text{ mA (Milliampere)} = \frac{1}{1000} \text{ A} \quad (98.3)$$

Fließen 10 C in 2 s, so ist die Stromstärke $I = Q/t = 10 \text{ C} / 2 \text{ s} = 5 \text{ A}$. In 20 s fließt bei diesem Strom die Ladung:

$$Q = I \cdot t = 5 \text{ A} \cdot 20 \text{ s} = 100 \text{ C} \quad (98.4)$$

2. Bei einer bestimmten Stromstärke fließt in der doppelten bzw. dreifachen Zeit die doppelte bzw. dreifache Ladung. Der Quotient $Q/t = 2Q/2t = 3Q/3t$ bleibt konstant. Bei größerer Stromstärke fließt dagegen in der gleichen Zeit mehr Ladung. Der Strom und die Ladung sind auch in Sekunden mehr Wärme und gibt eine stärkere magnetische Wirkung aus. Da sich der Hitzefaktor einer Vorrichtung, kann man das Hitzefaktorinstrument unmittelbar in Ampere (= Coulomb je Sekunde) eichen und als Strommesser verwenden. Das gleiche gilt für Maßgeräte auf magnetischer Grundlage (§ 84):

Wir schalten eine Knallgaszelle, ein Hitzefaktorinstrument und einen auf der magnetischen Stromwirkung beruhenden Strommesser hintereinander. Diese Geräte werden alle in der gleichen Zeit t von der gleichen Ladung Q durchflossen, unabhängig von der Reihenfolge. Die Stromstärke $I = Q/t$ ist in ihnen gleich; die Angabe der Instrumente kann also an der Gesamtwirkung nachgeprüft werden. Ein Strommesser gestattet mit einem Blick die Stromstärke zu messen, ohne daß man Ladung und Zeit einzeln bestimmt. Dies entspricht dem Tachometer im Auto, das ohne Messung von Weg (s) und Zeit (t) die Geschwindigkeit $v = s/t$ unmittelbar anzeigt.

Im unverzweigten Stromkreis ist die Stromstärke überall gleich groß.

§ 98 Messung der elektrischen Stromstärke

Fließt in der Zeit t die Ladung Q durch einen Querschnitt des Leiters, so berechnet man die Stromstärke I als Quotient aus der Ladung Q und der Zeit:

$$I = \frac{Q}{t} \quad (98.1)$$

Die Einheit der elektrischen Stromstärke ist 1 Ampere (A). Unter 1 A versteht man die Stromstärke, bei der 1 Coulomb in 1 s durch einen Querschnitt des Leiters fließt.

Nach § 95 bedeutet dies:

Ein Strom der Stärke 1 Ampere schaltet 0,775 cm³ Kupfaleit in 1 s ab (bei 0°C und 760 Torr) oder 1,118 mg Silber in 1 s:

$$1 \text{ A} = \frac{1}{1.118} \quad (98.2)$$

$$1 \text{ mA (Milliampere)} = \frac{1}{1000} \text{ A} \quad (98.3)$$

Fließen 10 C in 2 s, so ist die Stromstärke $I = Q/t = 10 \text{ C} / 2 \text{ s} = 5 \text{ A}$. In 20 s fließt bei diesem Strom die Ladung:

$$Q = I \cdot t = 5 \text{ A} \cdot 20 \text{ s} = 100 \text{ C} \quad (98.4)$$

2. Bei einer bestimmten Stromstärke fließt in der doppelten bzw. dreifachen Zeit die doppelte bzw. dreifache Ladung. Der Quotient $Q/t = 2Q/2t = 3Q/3t$ bleibt konstant. Bei größerer Stromstärke fließt dagegen in der gleichen Zeit mehr Ladung. Der Strom und die Ladung sind auch in Sekunden mehr Wärme und gibt eine stärkere magnetische Wirkung aus. Da sich der Hitzefaktor einer Vorrichtung, kann man das Hitzefaktorinstrument unmittelbar in Ampere (= Coulomb je Sekunde) eichen und als Strommesser verwenden. Das gleiche gilt für Maßgeräte auf magnetischer Grundlage (§ 84):

Wir schalten eine Knallgaszelle, ein Hitzefaktorinstrument und einen auf der magnetischen Stromwirkung beruhenden Strommesser hintereinander. Diese Geräte werden alle in der gleichen Zeit t von der gleichen Ladung Q durchflossen, unabhängig von der Reihenfolge. Die Stromstärke $I = Q/t$ ist in ihnen gleich; die Angabe der Instrumente kann also an der Gesamtwirkung nachgeprüft werden. Ein Strommesser gestattet mit einem Blick die Stromstärke zu messen, ohne daß man Ladung und Zeit einzeln bestimmt. Dies entspricht dem Tachometer im Auto, das ohne Messung von Weg (s) und Zeit (t) die Geschwindigkeit $v = s/t$ unmittelbar anzeigt.

Im unverzweigten Stromkreis ist die Stromstärke überall gleich groß.

V. Einige technische Anwendungen der Elektrizität

Wenn man auf diese Gegenkraft vorrichtet, dreht sich die Spule so weit, bis ihr Südpol dem Nordpol des Hufeisenmagneten gegenübersteht (Abb. 290.1). Ansort man in diesem Augenblick die Stromrichtung, so stehen sich zwei gleichnamige Pole gegenüber und abstoßen sich ab. Infolge ihres Schwingens dreht sich die Spule in der ursprünglichen Richtung weiter. Man muß nur immer dann den Strom in ihr umpolen, wenn der Spulen-

290.1 Prinzip des Elektromotors mit Stromwender und Bürsten

290.2 Ein Motor nach Abb. 290.1 wirkt vollständig hebeln. Durch viele gegenüberverordnete Windungen besteht der beiderseitige Bürstenkontakt

pol am Magnetpol vorbeiführt. Das besorgt selbständig der Kommutator (Stromwender). Er besteht aus zwei gegenüberliegenden Halbkugeln, die sich mit der Spule (Abb. 290.1) durch 1 Windung abwechselnd drehen. Auf diese schalten zwei feststehende Kohlekontakte, Bürsten genannt. Durch die Bürste fließt der Strom in den Halbkugeln und damit der Spule zu, daß ihre obere Fläche ein Südpol ist und deshalb nach rechts gezogen wird (Abb. 290.1). Wenn er dort ankommt, wechselt jede Bürste auf den andern Halbkugel über. Dann ist die sich nunmehr nach oben drehende Fläche Südpol; die Spule wird fortgesetzt. — Um die Kräfte zu vergrößern, ersetzt man den Dauermagneten durch einen Elektromagneten und füllt auch das Innere der Spule mit Eisen aus. Der Luftspalt zwischen den kreisförmig angeordneten Polschuh des Magneten und dem zylinderförmigen, rotierenden Anker wird, wie beim Elektromagneten, möglichst klein gehalten (Abb. 290.2).

§ 109 Die elektromagnetische Induktion

1. Faraday entdeckte 1830, daß der elektrische Strom von einem Magnetfeld umgeben ist. Es erzeugt im Elektromotor Kraft und Bewegung. Umgekehrt gelang es 1831 M. Faraday (S. 50), aus Bewegung in Magnetfeldern Spannung zu erzeugen:

1. Führe nach Abb. 291.1 den roten Leiter in das ruhende Magnetfeld von oben ein (induziert, einführen). Der angeschlossene Spannungsmesser zeigt, daß Spannung

V. Einige technische Anwendungen der Elektrizität

Wenn man auf diese Gegenkraft vorrichtet, dreht sich die Spule so weit, bis ihr Südpol dem Nordpol des Hufeisenmagneten gegenübersteht (Abb. 290.1). Ansort man in diesem Augenblick die Stromrichtung, so stehen sich zwei gleichnamige Pole gegenüber und abstoßen sich ab. Infolge ihres Schwingens dreht sich die Spule in der ursprünglichen Richtung weiter. Man muß nur immer dann den Strom in ihr umpolen, wenn der Spulen-

290.1 Prinzip des Elektromotors mit Stromwender und Bürsten

290.2 Ein Motor nach Abb. 290.1 wirkt vollständig hebeln. Durch viele gegenüberverordnete Windungen besteht der beiderseitige Bürstenkontakt

pol am Magnetpol vorbeiführt. Das besorgt selbständig der Kommutator (Stromwender). Er besteht aus zwei gegenüberliegenden Halbkugeln, die sich mit der Spule (Abb. 290.1) durch 1 Windung abwechselnd drehen. Auf diese schalten zwei feststehende Kohlekontakte, Bürsten genannt. Durch die Bürste fließt der Strom in den Halbkugeln und damit der Spule zu, daß ihre obere Fläche ein Südpol ist und deshalb nach rechts gezogen wird (Abb. 290.1). Wenn er dort ankommt, wechselt jede Bürste auf den andern Halbkugel über. Dann ist die sich nunmehr nach oben drehende Fläche Südpol; die Spule wird fortgesetzt. — Um die Kräfte zu vergrößern, ersetzt man den Dauermagneten durch einen Elektromagneten und füllt auch das Innere der Spule mit Eisen aus. Der Luftspalt zwischen den kreisförmig angeordneten Polschuh des Magneten und dem zylinderförmigen, rotierenden Anker wird, wie beim Elektromagneten, möglichst klein gehalten (Abb. 290.2).

§ 109 Die elektromagnetische Induktion

1. Faraday entdeckte 1830, daß der elektrische Strom von einem Magnetfeld umgeben ist. Es erzeugt im Elektromotor Kraft und Bewegung. Umgekehrt gelang es 1831 M. Faraday (S. 50), aus Bewegung in Magnetfeldern Spannung zu erzeugen:

1. Führe nach Abb. 291.1 den roten Leiter in das ruhende Magnetfeld von oben ein (induziert, einführen). Der angeschlossene Spannungsmesser zeigt, daß Spannung

V. Einige technische Anwendungen der Elektrizität

Wenn man auf diese Gegenkraft vorrichtet, dreht sich die Spule so weit, bis ihr Südpol dem Nordpol des Hufeisenmagneten gegenübersteht (Abb. 290.1). Ansort man in diesem Augenblick die Stromrichtung, so stehen sich zwei gleichnamige Pole gegenüber und abstoßen sich ab. Infolge ihres Schwingens dreht sich die Spule in der ursprünglichen Richtung weiter. Man muß nur immer dann den Strom in ihr umpolen, wenn der Spulen-

290.1 Prinzip des Elektromotors mit Stromwender und Bürsten

290.2 Ein Motor nach Abb. 290.1 wirkt vollständig hebeln. Durch viele gegenüberverordnete Windungen besteht der beiderseitige Bürstenkontakt

pol am Magnetpol vorbeiführt. Das besorgt selbständig der Kommutator (Stromwender). Er besteht aus zwei gegenüberliegenden Halbkugeln, die sich mit der Spule (Abb. 290.1) durch 1 Windung abwechselnd drehen. Auf diese schalten zwei feststehende Kohlekontakte, Bürsten genannt. Durch die Bürste fließt der Strom in den Halbkugeln und damit der Spule zu, daß ihre obere Fläche ein Südpol ist und deshalb nach rechts gezogen wird (Abb. 290.1). Wenn er dort ankommt, wechselt jede Bürste auf den andern Halbkugel über. Dann ist die sich nunmehr nach oben drehende Fläche Südpol; die Spule wird fortgesetzt. — Um die Kräfte zu vergrößern, ersetzt man den Dauermagneten durch einen Elektromagneten und füllt auch das Innere der Spule mit Eisen aus. Der Luftspalt zwischen den kreisförmig angeordneten Polschuh des Magneten und dem zylinderförmigen, rotierenden Anker wird, wie beim Elektromagneten, möglichst klein gehalten (Abb. 290.2).

§ 109 Die elektromagnetische Induktion

1. Faraday entdeckte 1830, daß der elektrische Strom von einem Magnetfeld umgeben ist. Es erzeugt im Elektromotor Kraft und Bewegung. Umgekehrt gelang es 1831 M. Faraday (S. 50), aus Bewegung in Magnetfeldern Spannung zu erzeugen:

1. Führe nach Abb. 291.1 den roten Leiter in das ruhende Magnetfeld von oben ein (induziert, einführen). Der angeschlossene Spannungsmesser zeigt, daß Spannung

IV. Elektrische Gesetze und Größen

Wenn zwischen den Enden eines Leiters die Spannung U liegt und in ihm Strom der Stärke I fließt, berechnet man den Widerstand R des Leiters nach:

$$R = \frac{U}{I} \quad (\text{Definition des Widerstandes}) \quad (99.1)$$

Die Einheit des Widerstandes ist 1 Ohm (Ω) = 1 Volt/Ampere.

3. G. S. Ohm untersuchte auch, wie sich der Widerstand eines Drahtes beim Erhitzen ändert:

Erhielt man den Drahtbrenner eine Wendel aus dünnem Eisenblech, die nach Abb. 292.1 in einem Stromkreis liegt. Die Stromstärke I sinkt erheblich, obwohl die Spannung U konstant bleibt. Nach $R = U/I$ steigt also beim Erhitzen der Widerstand des Eisens (Abb. 292.2). Konstanten behält nach Abb. 292.3 beim Erhitzen konstanter Widerstand. Deshalb fließt in V der Widerstand trotz der Erwärmung des Konstantenstroms durch den Strom konstant. Aus dem gleichen Grund benutzt man diese und ähnliche Legierungen in Maßgeräten. Bei Kohle sinkt beim Erhitzen der Widerstand.

292.1 Beim Erhitzen sinkt der Widerstand des Eisens

292.2 Beim Erhitzen sinkt der Widerstand des Eisens

292.3 Beim Erhitzen sinkt der Widerstand des Eisens

Mit nach V des Widerstand einer Glühlampe bei sehr kleiner Stromstärke. Auf welchen Wert steigt er bei normaler Belastung infolge der Erwärmung?

Der Widerstand von Metallen nimmt im allgemeinen beim Erhitzen zu.

Sollt man durch Versuch den Zusammenhang zwischen Widerstand und Temperatur fest, so kann man nachher aus dem gemessenen Widerstand auf die Temperatur schließen. Diese Widerstandsthermometer werden auf elektrischen Weg Temperatur zu ermitteln oder schoner ausgerechneten. Oder: man er größer oder zu messen (Flugzeugmotoren, Öfen). Siehe auch § 112.

IV. Elektrische Gesetze und Größen

Wenn zwischen den Enden eines Leiters die Spannung U liegt und in ihm Strom der Stärke I fließt, berechnet man den Widerstand R des Leiters nach:

$$R = \frac{U}{I} \quad (\text{Definition des Widerstandes}) \quad (99.1)$$

Die Einheit des Widerstandes ist 1 Ohm (Ω) = 1 Volt/Ampere.

3. G. S. Ohm untersuchte auch, wie sich der Widerstand eines Drahtes beim Erhitzen ändert:

Erhielt man den Drahtbrenner eine Wendel aus dünnem Eisenblech, die nach Abb. 292.1 in einem Stromkreis liegt. Die Stromstärke I sinkt erheblich, obwohl die Spannung U konstant bleibt. Nach $R = U/I$ steigt also beim Erhitzen der Widerstand des Eisens (Abb. 292.2). Konstanten behält nach Abb. 292.3 beim Erhitzen konstanter Widerstand. Deshalb fließt in V der Widerstand trotz der Erwärmung des Konstantenstroms durch den Strom konstant. Aus dem gleichen Grund benutzt man diese und ähnliche Legierungen in Maßgeräten. Bei Kohle sinkt beim Erhitzen der Widerstand.

292.1 Beim Erhitzen sinkt der Widerstand des Eisens

292.2 Beim Erhitzen sinkt der Widerstand des Eisens

292.3 Beim Erhitzen sinkt der Widerstand des Eisens

Mit nach V des Widerstand einer Glühlampe bei sehr kleiner Stromstärke. Auf welchen Wert steigt er bei normaler Belastung infolge der Erwärmung?

Der Widerstand von Metallen nimmt im allgemeinen beim Erhitzen zu.

Sollt man durch Versuch den Zusammenhang zwischen Widerstand und Temperatur fest, so kann man nachher aus dem gemessenen Widerstand auf die Temperatur schließen. Diese Widerstandsthermometer werden auf elektrischen Weg Temperatur zu ermitteln oder schoner ausgerechneten. Oder: man er größer oder zu messen (Flugzeugmotoren, Öfen). Siehe auch § 112.

IV. Elektrische Gesetze und Größen

Wenn zwischen den Enden eines Leiters die Spannung U liegt und in ihm Strom der Stärke I fließt, berechnet man den Widerstand R des Leiters nach:

$$R = \frac{U}{I} \quad (\text{Definition des Widerstandes}) \quad (99.1)$$

Die Einheit des Widerstandes ist 1 Ohm (Ω) = 1 Volt/Ampere.

3. G. S. Ohm untersuchte auch, wie sich der Widerstand eines Drahtes beim Erhitzen ändert:

Erhielt man den Drahtbrenner eine Wendel aus dünnem Eisenblech, die nach Abb. 292.1 in einem Stromkreis liegt. Die Stromstärke I sinkt erheblich, obwohl die Spannung U konstant bleibt. Nach $R = U/I$ steigt also beim Erhitzen der Widerstand des Eisens (Abb. 292.2). Konstanten behält nach Abb. 292.3 beim Erhitzen konstanter Widerstand. Deshalb fließt in V der Widerstand trotz der Erwärmung des Konstantenstroms durch den Strom konstant. Aus dem gleichen Grund benutzt man diese und ähnliche Legierungen in Maßgeräten. Bei Kohle sinkt beim Erhitzen der Widerstand.

292.1 Beim Erhitzen sinkt der Widerstand des Eisens

292.2 Beim Erhitzen sinkt der Widerstand des Eisens

292.3 Beim Erhitzen sinkt der Widerstand des Eisens

Mit nach V des Widerstand einer Glühlampe bei sehr kleiner Stromstärke. Auf welchen Wert steigt er bei normaler Belastung infolge der Erwärmung?

Der Widerstand von Metallen nimmt im allgemeinen beim Erhitzen zu.

Sollt man durch Versuch den Zusammenhang zwischen Widerstand und Temperatur fest, so kann man nachher aus dem gemessenen Widerstand auf die Temperatur schließen. Diese Widerstandsthermometer werden auf elektrischen Weg Temperatur zu ermitteln oder schoner ausgerechneten. Oder: man er größer oder zu messen (Flugzeugmotoren, Öfen). Siehe auch § 112.

LEHRE VOM MAGNETISMUS

§75 **Wichtige Eigenschaften der Magnete**

Bereits im Altertum kannte man Eisenzerstückchen, die Eisen, Kobalt und Nickel anziehen, nicht aber andere Stoffe wie Holz, Messing oder Blei. Nach ihrem angeblichen Fundort, der Stadt Magnesia (s. 1) in Kleinasien, nennt man sie Magnete. Diese natürlichen Magneteisensteine (Abb. 218.1) sind sehr schwach magnetisch. Deshalb benutzt man heute künstliche Magnete aus Stahl oder bestimmten Legierungen (Abb. 219.2). Wie man sie herstellt, werden wir auf S. 220 und 241 erfahren.

218.1 Magnetstein aus Eisenkies
218.2 Der künstliche Magnet aus Gussstahl (Eisen-Nickel-Kobalt-Legierung) trägt das übliche, seine Eigenschaften
219.2 Stabmagnet mit Eisenkern

Für Auge und Teststift unterscheidet sich ein Magnet in nichts von einem gewöhnlichen Stab. Steht man ihn aber in Eisenstäben, so überzieht er sich an seinen Enden, den Polen, mit dicken Bärten (Abb. 219.3). Die Mitte des Magneten, die Indifferenzzone (indifferent, lat.; gleichgültig, wirkungslos), übt dagegen nur unbedeutende magnetische Kräfte aus.

Die Stellen starker Anziehung eines Magneten nennt man Pole.

Hänge nach Abb. 219.4 einen Stabmagnet an einem dünnen Faden waagrecht und leicht darüber auf. Er zeigt nach einigen Schwingungen mit dem einen Pol, dem sog. Nordpol N, anziehend nach Norden. Der nach Süden wiesende Pol heißt Südpol S. Beim Kompaß ist ein solcher nadelförmiger Stabmagnet auf einer Spitze leicht drehbar gelagert (s. Abb. 220.1 und 222.3).

219.3 Wo ist Norden?

1) Heute Manne, nordöstlich Ismer (Tahiti).

LEHRE VOM MAGNETISMUS

§75 **Wichtige Eigenschaften der Magnete**

Bereits im Altertum kannte man Eisenzerstückchen, die Eisen, Kobalt und Nickel anziehen, nicht aber andere Stoffe wie Holz, Messing oder Blei. Nach ihrem angeblichen Fundort, der Stadt Magnesia (s. 1) in Kleinasien, nennt man sie Magnete. Diese natürlichen Magneteisensteine (Abb. 218.1) sind sehr schwach magnetisch. Deshalb benutzt man heute künstliche Magnete aus Stahl oder bestimmten Legierungen (Abb. 219.2). Wie man sie herstellt, werden wir auf S. 220 und 241 erfahren.

218.1 Magnetstein aus Eisenkies
218.2 Der künstliche Magnet aus Gussstahl (Eisen-Nickel-Kobalt-Legierung) trägt das übliche, seine Eigenschaften
219.2 Stabmagnet mit Eisenkern

Für Auge und Teststift unterscheidet sich ein Magnet in nichts von einem gewöhnlichen Stab. Steht man ihn aber in Eisenstäben, so überzieht er sich an seinen Enden, den Polen, mit dicken Bärten (Abb. 219.3). Die Mitte des Magneten, die Indifferenzzone (indifferent, lat.; gleichgültig, wirkungslos), übt dagegen nur unbedeutende magnetische Kräfte aus.

Die Stellen starker Anziehung eines Magneten nennt man Pole.

Hänge nach Abb. 219.4 einen Stabmagnet an einem dünnen Faden waagrecht und leicht darüber auf. Er zeigt nach einigen Schwingungen mit dem einen Pol, dem sog. Nordpol N, anziehend nach Norden. Der nach Süden wiesende Pol heißt Südpol S. Beim Kompaß ist ein solcher nadelförmiger Stabmagnet auf einer Spitze leicht drehbar gelagert (s. Abb. 220.1 und 222.3).

219.3 Wo ist Norden?

1) Heute Manne, nordöstlich Ismer (Tahiti).

LEHRE VOM MAGNETISMUS

§75 **Wichtige Eigenschaften der Magnete**

Bereits im Altertum kannte man Eisenzerstückchen, die Eisen, Kobalt und Nickel anziehen, nicht aber andere Stoffe wie Holz, Messing oder Blei. Nach ihrem angeblichen Fundort, der Stadt Magnesia (s. 1) in Kleinasien, nennt man sie Magnete. Diese natürlichen Magneteisensteine (Abb. 218.1) sind sehr schwach magnetisch. Deshalb benutzt man heute künstliche Magnete aus Stahl oder bestimmten Legierungen (Abb. 219.2). Wie man sie herstellt, werden wir auf S. 220 und 241 erfahren.

218.1 Magnetstein aus Eisenkies
218.2 Der künstliche Magnet aus Gussstahl (Eisen-Nickel-Kobalt-Legierung) trägt das übliche, seine Eigenschaften
219.2 Stabmagnet mit Eisenkern

Für Auge und Teststift unterscheidet sich ein Magnet in nichts von einem gewöhnlichen Stab. Steht man ihn aber in Eisenstäben, so überzieht er sich an seinen Enden, den Polen, mit dicken Bärten (Abb. 219.3). Die Mitte des Magneten, die Indifferenzzone (indifferent, lat.; gleichgültig, wirkungslos), übt dagegen nur unbedeutende magnetische Kräfte aus.

Die Stellen starker Anziehung eines Magneten nennt man Pole.

Hänge nach Abb. 219.4 einen Stabmagnet an einem dünnen Faden waagrecht und leicht darüber auf. Er zeigt nach einigen Schwingungen mit dem einen Pol, dem sog. Nordpol N, anziehend nach Norden. Der nach Süden wiesende Pol heißt Südpol S. Beim Kompaß ist ein solcher nadelförmiger Stabmagnet auf einer Spitze leicht drehbar gelagert (s. Abb. 220.1 und 222.3).

219.3 Wo ist Norden?

1) Heute Manne, nordöstlich Ismer (Tahiti).

§17 Der Hebel

3. Zusammensetzung und Zerlegung paralleler Kräfte an festen Körpern. a) Die Wirkung eines Hebels hängt wesentlich davon ab, wie die Kräfte zur Drehachse liegen. Was sie mit ihr zu tun haben, wollen wir jetzt untersuchen.

V.37: Hänge nach Abb. 47.1 die Drehachse eines Hebels an einem Kraftmesser. Er wird wie zu erwarten nicht nur durch das Hebelgewicht (100 g), sondern auch durch die Resultierende $L = F_1 + F_2$ (s. 100 g + 200 g) der beiden Einzelkräfte belastet. Diese Resultierende wirkt von der festen Drehachse aufgenommen. Dann herrscht Gleichgewicht. Für die Abstände a_1 und a_2 der Resultierenden L von den Komponenten F_1 und F_2 gilt nach dem Hebelgesetz: $a_1 \cdot F_1 = a_2 \cdot F_2$ oder $a_1/a_2 = F_2/F_1$. Die Resultierende L zweier paralleler Einzelkräfte liegt um so näher bei der größeren Einzelkraft, je mehr diese die kleinere überwiegt.

An einem festen Körper setzen sich zwei parallele Einzelkräfte F_1 und F_2 zu einer ihrer parallelen Resultierenden zusammen.

Für die Abstände der Resultierenden L von den Einzelkräften gilt:

$$a_1 \cdot F_1 = a_2 \cdot F_2 \quad (47.2)$$

Am Hebel herrscht Gleichgewicht, wenn die Drehmomente auf der Resultierenden liegen.

b) Am Traghebel (Abb. 47.2 und 47.3) wird umgekehrt nach L in parallele Einzelkräfte zerlegt und auf die beiden Träger verteilt.

V.38: Hänge eine 100 cm lange, waagrechte Stange an ihren Enden mit zwei Kraftmessern auf Balken so, dass 30 cm vom linken Kraftmesser entfernt mit $L = 100$ g die Last L fest sich auf links in $L = 70$ g Traghebel. 30 g und rechts in $L = 70$ g. Es gilt: $F_1 = L$ und $F_2 = a_1/a_2 \cdot F_1 = 70$ g. Diese Gleichungen stellen die Umkehrung der Gl. (47.1) und (47.2) dar.

V.39: Prüfe die beiden aufgestellten Gleichungen nach, wenn die Last in der Mitte des Traghebels, wenn sie 50 cm vom rechten bzw. linken Ende entfernt hängt.

8. Die Drehwaage (Abb. 47.4): Die Last L (210 g) liegt auf der Brücke und tritt sich nach dem Gewicht des Traghebels in die Waage F_1 auf $F_1 = F_2 = L$. Dabei wird F_2 umverteilt durch die Stange Z auf den oberen Hebel übertragen. An der Waage, die am 10cm langen Hebelarm hängt, ist zum Auf-

47.1 Hebel

47.2 Traghebel

47.3 Traghebel

47.4 Drehwaage

§17 Der Hebel

3. Zusammensetzung und Zerlegung paralleler Kräfte an festen Körpern. a) Die Wirkung eines Hebels hängt wesentlich davon ab, wie die Kräfte zur Drehachse liegen. Was sie mit ihr zu tun haben, wollen wir jetzt untersuchen.

V.37: Hänge nach Abb. 47.1 die Drehachse eines Hebels an einem Kraftmesser. Er wird wie zu erwarten nicht nur durch das Hebelgewicht (100 g), sondern auch durch die Resultierende $L = F_1 + F_2$ (s. 100 g + 200 g) der beiden Einzelkräfte belastet. Diese Resultierende wirkt von der festen Drehachse aufgenommen. Dann herrscht Gleichgewicht. Für die Abstände a_1 und a_2 der Resultierenden L von den Komponenten F_1 und F_2 gilt nach dem Hebelgesetz: $a_1 \cdot F_1 = a_2 \cdot F_2$ oder $a_1/a_2 = F_2/F_1$. Die Resultierende L zweier paralleler Einzelkräfte liegt um so näher bei der größeren Einzelkraft, je mehr diese die kleinere überwiegt.

An einem festen Körper setzen sich zwei parallele Einzelkräfte F_1 und F_2 zu einer ihrer parallelen Resultierenden zusammen.

Für die Abstände der Resultierenden L von den Einzelkräften gilt:

$$a_1 \cdot F_1 = a_2 \cdot F_2 \quad (47.2)$$

Am Hebel herrscht Gleichgewicht, wenn die Drehmomente auf der Resultierenden liegen.

b) Am Traghebel (Abb. 47.2 und 47.3) wird umgekehrt nach L in parallele Einzelkräfte zerlegt und auf die beiden Träger verteilt.

V.38: Hänge eine 100 cm lange, waagrechte Stange an ihren Enden mit zwei Kraftmessern auf Balken so, dass 30 cm vom linken Kraftmesser entfernt mit $L = 100$ g die Last L fest sich auf links in $L = 70$ g Traghebel. 30 g und rechts in $L = 70$ g. Es gilt: $F_1 = L$ und $F_2 = a_1/a_2 \cdot F_1 = 70$ g. Diese Gleichungen stellen die Umkehrung der Gl. (47.1) und (47.2) dar.

V.39: Prüfe die beiden aufgestellten Gleichungen nach, wenn die Last in der Mitte des Traghebels, wenn sie 50 cm vom rechten bzw. linken Ende entfernt hängt.

8. Die Drehwaage (Abb. 47.4): Die Last L (210 g) liegt auf der Brücke und tritt sich nach dem Gewicht des Traghebels in die Waage F_1 auf $F_1 = F_2 = L$. Dabei wird F_2 umverteilt durch die Stange Z auf den oberen Hebel übertragen. An der Waage, die am 10cm langen Hebelarm hängt, ist zum Auf-

47.1 Hebel

47.2 Traghebel

47.3 Traghebel

47.4 Drehwaage

§17 Der Hebel

3. Zusammensetzung und Zerlegung paralleler Kräfte an festen Körpern. a) Die Wirkung eines Hebels hängt wesentlich davon ab, wie die Kräfte zur Drehachse liegen. Was sie mit ihr zu tun haben, wollen wir jetzt untersuchen.

V.37: Hänge nach Abb. 47.1 die Drehachse eines Hebels an einem Kraftmesser. Er wird wie zu erwarten nicht nur durch das Hebelgewicht (100 g), sondern auch durch die Resultierende $L = F_1 + F_2$ (s. 100 g + 200 g) der beiden Einzelkräfte belastet. Diese Resultierende wirkt von der festen Drehachse aufgenommen. Dann herrscht Gleichgewicht. Für die Abstände a_1 und a_2 der Resultierenden L von den Komponenten F_1 und F_2 gilt nach dem Hebelgesetz: $a_1 \cdot F_1 = a_2 \cdot F_2$ oder $a_1/a_2 = F_2/F_1$. Die Resultierende L zweier paralleler Einzelkräfte liegt um so näher bei der größeren Einzelkraft, je mehr diese die kleinere überwiegt.

An einem festen Körper setzen sich zwei parallele Einzelkräfte F_1 und F_2 zu einer ihrer parallelen Resultierenden zusammen.

Für die Abstände der Resultierenden L von den Einzelkräften gilt:

$$a_1 \cdot F_1 = a_2 \cdot F_2 \quad (47.2)$$

Am Hebel herrscht Gleichgewicht, wenn die Drehmomente auf der Resultierenden liegen.

b) Am Traghebel (Abb. 47.2 und 47.3) wird umgekehrt nach L in parallele Einzelkräfte zerlegt und auf die beiden Träger verteilt.

V.38: Hänge eine 100 cm lange, waagrechte Stange an ihren Enden mit zwei Kraftmessern auf Balken so, dass 30 cm vom linken Kraftmesser entfernt mit $L = 100$ g die Last L fest sich auf links in $L = 70$ g Traghebel. 30 g und rechts in $L = 70$ g. Es gilt: $F_1 = L$ und $F_2 = a_1/a_2 \cdot F_1 = 70$ g. Diese Gleichungen stellen die Umkehrung der Gl. (47.1) und (47.2) dar.

V.39: Prüfe die beiden aufgestellten Gleichungen nach, wenn die Last in der Mitte des Traghebels, wenn sie 50 cm vom rechten bzw. linken Ende entfernt hängt.

8. Die Drehwaage (Abb. 47.4): Die Last L (210 g) liegt auf der Brücke und tritt sich nach dem Gewicht des Traghebels in die Waage F_1 auf $F_1 = F_2 = L$. Dabei wird F_2 umverteilt durch die Stange Z auf den oberen Hebel übertragen. An der Waage, die am 10cm langen Hebelarm hängt, ist zum Auf-

47.1 Hebel

47.2 Traghebel

47.3 Traghebel

47.4 Drehwaage

Apêndice B

Publicações Decorrentes da Pesquisa

Os artigos que foram produzidos a partir das pesquisas realizadas durante a elaboração desta tese estão listados a seguir.

- [1] J. Barrera, E. R. Dougherty, and N. S. **Tomita**¹. Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory. *Electronic Imaging*, 6(1):54–67, January 1997.
- [2] E. R. Dougherty, J. Barrera, and N. S. **Tomita**. Optimal Filters from Prior Filters. In *Nonlinear Image Processing VIII*, volume 3026 of *Proc. of SPIE*, pages 2–7, San Jose, CA, USA, February 1997.
- [3] J. Barrera, E. R. Dougherty, and N. S. T. **Hirata**². Design of Optimal Morphological Operators from Prior Filters. *Acta Steriologica*, 16(3):193–200, 1997. Special issue on Mathematical Morphology.
- [4] J. Barrera, R. Terada, R. A. Lotufo, N. S. T. **Hirata**, R. Hirata Jr., and F. A. Zampirolli. An OCR based on Mathematical Morphology. In *Nonlinear Image Processing IX*, volume 3304 of *Proceedings of SPIE*, pages 197–208, San Jose, CA, January 1998.
- [5] E. R. Dougherty, A. M. Grigoryan, J. Barrera, and N. S. T. **Hirata**. Binary Filter Design: Optimization, Prior Information, and Robustness. In *Mathematical Modeling and Estimation Techniques in Computer Vision*, volume 3457 of *Proc. SPIE's International Symposium on Optical Science, Engineering and Instrumentation*, pages 230–241, San Diego, CA, July 1998.
- [6] N. S. T. **Hirata**, E. R. Dougherty, and J. Barrera. Design of Large-Window Binary Filters via Iteration. In *Intelligent Robots and Computer Vision XVII: Algorithms, Techniques, and Active Vision*, volume 3522 of *Proc. of SPIE*, pages 173–182, Boston, Massachusetts, November 1998.
- [7] N. S. T. **Hirata**, E. R. Dougherty, and J. Barrera. Efficient Switching Algorithm for Designing Increasing Binary Filters. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Image Processing X*, volume 3646 of *Proc. SPIE*, pages 185–196, San Jose, CA, January 1999.

¹Este é um artigo decorrente do trabalho de mestrado [157], que foi premiado como a melhor dissertação de mestrado [158] no *Concurso de Teses e Dissertações* da Sociedade Brasileira de Computação, no ano de 1996.

²Alteração do estado civil.

- [8] N. S. T. **Hirata**, J. Barrera, and E. R. Dougherty. Design of Statistically Optimal Stack Filters. In J. Stolfi and C. L. Tozzi, editors, *Proc. of Sibgrapi'99*, pages 265–274, Campinas, SP, Brazil, 1999.
- [9] H. M. F. Madeira, J. Barrera, R. Hirata Jr, and N. S. T. **Hirata**. A New Paradigm for the Architecture of Morphological Machines : Binary Decision Diagrams. In J. Stolfi and C. L. Tozzi, editors, *Proc. of Sibgrapi'99*, pages 283–292, Campinas, SP, Brazil, 1999.
- [10] J. Barrera, R. Terada, R. Hirata Jr, and N. S. T. **Hirata**. Automatic Programming of Morphological Machines by PAC Learning. *Fundamenta Informaticae*, 41(1-2):229–258, January 2000.
- [11] N. S. T. **Hirata**, E. R. Dougherty, and J. Barrera. A Switching Algorithm for Design of Optimal Increasing Binary Filters Over Large Windows. *Pattern Recognition*, 33(6):1059–1081, June 2000. Special Issue on Mathematical Morphology & Nonlinear Image Processing.
- [12] N. S. T. **Hirata**, E. R. Dougherty, and J. Barrera. Bayesian Switching Algorithm for the Optimal increasing Binary Filter. In *Proceedings of EUSIPCO - 2000*, volume IV, pages 1889–1892, 2000.
- [13] N. S. T. **Hirata**, J. Barrera, and R. Terada. Text Segmentation by Automatically Designed Morphological Operators. In *Proc. of SIBGRAPI'2000*, pages 284–291, Gramado, Brazil, 2000.
- [14] N. S. T. **Hirata**, E. R. Dougherty, and J. Barrera. Iterative Design of Morphological Binary Image Operators. To appear in *Optical Engineering*.

Referências Bibliográficas

- [1] S. Agaian, J. Astola, and K. Egiazarian. *Binary Polynomial Transforms and Nonlinear Digital Filters*. Marcel Dekker, 1995.
- [2] I. Andreadis, A. Gasteratos, and P. Tsalides. An ASIC for fast grey-scale dilation. *Microprocessors and Microsystems*, 20(2):89–95, April 1996.
- [3] M. Anthony and N. Biggs. *Computational Learning Theory - An Introduction*. Cambridge University Press, 1992.
- [4] A. Antonacopoulos. Page Segmentation Using the Description of the Background. *Computer Vision and Image Understanding*, 70(3):350–369, June 1998.
- [5] A. Asano, K. Matsumura, K. Itoh, Y. Ichioka, and S. Yokozeki. Optimization of Morphological Filters by Learning. *Optics Communications*, 112:265–270, 1994.
- [6] A. Asano, T. Yamashita, and S. Yokozeki. Learning Optimization of Morphological Filters with Gray Scale Structuring Elements. *Optical Engineering*, 35(8):2203–2213, August 1996.
- [7] A. Asano and S. Yokozeki. Multiresolution Pattern Spectrum and its Application to Optimization of Nonlinear Filter. In *International Conference on Image Processing*, Proc. of IEEE, pages 387–390, Lausanne, Switzerland, 1996.
- [8] G. J. F. Banon. Characterization of Translation Invariant Elementary Operators for Gray-level Morphology. In E. R. Dougherty, F. Preteux, and S. Shen, editors, *Neural, Morphological, and Stochastic Methods in Image and Signal Processing*, SPIE Proceedings, pages 68–79, 1995.
- [9] G. J. F. Banon and J. Barrera. Minimal Representations for Translation-Invariant Set Mappings by Mathematical Morphology. *SIAM J. Appl. Math.*, 51(6):1782–1798, December 1991.
- [10] G. J. F. Banon and J. Barrera. Decomposition of Mappings between Complete Lattices by Mathematical Morphology, Part I. General Lattices. *Signal Processing*, 30:299–327, 1993.
- [11] G. J. F. Banon and J. Barrera. Bases da Morfologia Matemática para Análise de Imagens Binárias. IX Escola de Computação, Pernambuco, Julho 1994.
- [12] J. Barrera, G. J. F. Banon, R. A. Lotufo, and R. Hirata Jr. MMach: a Mathematical Morphology Toolbox for the Khoros System. *Electronic Imaging*, 7(1):174–210, 1998.
- [13] J. Barrera, F. S. C. da Silva, and G. J. F. Banon. Automatic Programming of Binary Morphological Machines. In *Image Algebra and Morphological Image Processing*, volume 2300 of *Proc. of SPIE*, pages 229–240, San Diego, 1994.

- [14] J. Barrera, E. R. Dougherty, and N. S. T. Hirata. Design of Optimal Morphological Operators from Prior Filters. *Acta Steriologica*, 16(3):193–200, 1997. Special issue on Mathematical Morphology.
- [15] J. Barrera, E. R. Dougherty, and N. S. Tomita. Automatic Programming of Binary Morphological Machines by Design of Statistically Optimal Operators in the Context of Computational Learning Theory. *Electronic Imaging*, 6(1):54–67, January 1997.
- [16] J. Barrera and G. P. Salas. Set Operations on Closed Intervals and Their Applications to the Automatic Programming of Morphological Machines. *Electronic Imaging*, 5(3):335–352, July 1996.
- [17] J. Barrera, R. Terada, F. S. C. da Silva, and N. S. Tomita. Automatic Programming of Morphological Machines for OCR. In *Mathematical Morphology and its Applications to Image and Signal Processing*, pages 385–392, Atlanta, GA, May 1996. International Symposium on Mathematical Morphology, Kluwer Academic Publishers.
- [18] J. Barrera, R. Terada, R. Hirata Jr, and N. S. T. Hirata. Automatic Programming of Morphological Machines by PAC Learning. *Fundamenta Informaticae*, 41(1-2):229–258, January 2000.
- [19] J. Barrera, R. Terada, R. A. Lotufo, N. S. T. Hirata, R. Hirata Jr., and F. A. Zampiroli. An OCR based on Mathematical Morphology. In *Nonlinear Image Processing IX*, volume 3304 of *Proceedings of SPIE*, pages 197–208, San Jose, CA, January 1998.
- [20] G. A. Baxes. *Digital Image Processing - Principles and Applications*. John Wiley and Sons, Inc., 1994.
- [21] S. Beucher. *Segmentation D’Images et Morphologie Mathématique*. PhD thesis, l’Ecole Nationale Supérieure des Mines de Paris, 1990.
- [22] S. Beucher and F. Meyer. *Mathematical Morphology in Image Processing*, chapter 12. The Morphological Approach to Segmentation: The Watershed Transformation, pages 433–481. Marcel Dekker, 1992.
- [23] G. Birkhoff. *Lattice Theory*. American Mathematical Society, Providence, Rhode Island, 3rd edition, 1967.
- [24] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford : Clarendon, 1996.
- [25] A. Bleau, J. De Guise, and A.-R. LeBlanc. A New Set of Fast Algorithms for Mathematical Morphology – I. Idempotent Geodesic Transforms. *CVGIP : Image Understanding*, 56(2):178–209, September 1992.
- [26] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. The Wadsworth Statistics/Probability Series. Wadsworth International Group, 1984.
- [27] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.
- [28] L. Cinque, L. Lombardi, and G. Manzini. A Multiresolution Approach for Page Segmentation. *Pattern Recognition Letters*, 19(2):217–225, February 1998.

- [29] M. L. Comer and E. J. Delp. An Empirical Study of Morphological Operators in Color Image Enhancement. In *Proceedings of the SPIE Conference on Image Processing Algorithms and Techniques III*, volume 1657, pages 314–325, San Jose, California, February 10–13 1992.
- [30] M. L. Comer and E. J. Delp. Multiresolution Image Segmentation. In *Proceedings of the 1995 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2415–2418, Detroit, MI, May 1995.
- [31] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.
- [32] O. Coudert. Two-level Logic Minimization: an Overview. *Integration, the VLSI Journal*, 17(2):97–140, October 1994.
- [33] E. J. Coyle and J.-H. Lin. Stack Filters and the Mean Absolute Error Criterion. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(8):1244–1254, August 1988.
- [34] E. J. Coyle, J.-H. Lin, and M. Gabbouj. Optimal Stack Filtering and the Estimation and Structural Approaches to Image Processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(12):2037–2066, December 1989.
- [35] J. Crespo. *Morphological Connected Filters and Intra-Region Smoothing for Image Segmentation*. PhD thesis, Georgia Institute of Technology, November 1993.
- [36] H. A. Curtis. A Functional Canonical Form. *Journal of the ACM*, 6(2):245–258, April 1959.
- [37] M. Davio, J.P. Deschamps, and A. Thayse. *Discrete and Switching Functions*. Advanced Book Program. McGraw-Hill, 1978.
- [38] L. Devroye. Automatic Pattern Recognition: A Study of the Probability of Error. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(4):530–543, July 1988.
- [39] E. R. Dougherty. Optimal Mean-Square N-Observation Digital Morphological Filters I. Optimal Binary Filters. *CVGIP: Image Understanding*, 55(1):36–54, January 1992.
- [40] E. R. Dougherty. Optimal Mean-Square N-Observation Digital Morphological Filters II. Optimal Gray-Scale Filters. *CVGIP: Image Understanding*, 55(1):55–72, January 1992.
- [41] E. R. Dougherty, editor. *Mathematical Morphology in Image Processing*. Marcel Dekker, 1993.
- [42] E. R. Dougherty. Binary Filter Estimation for Large Windows. In *SIBGRAPI-Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens*, Rio de Janeiro, October 1998. IEEE Press.
- [43] E. R. Dougherty. *Random Processes for Image and Signal Processing*. SPIE and IEEE Presses, Bellingham, 1998.
- [44] E. R. Dougherty. The Granulometric Size Density in Filter Optimization. In J. Stolfi and C. L. Tozzi, editors, *Proc. of Sibgrapi'99*, pages 257–264, Campinas, SP, Brazil, 1999.
- [45] E. R. Dougherty and J. T. Astola, editors. *Nonlinear Filters for Image Processing*. SPIE—The International Society for Optical Engineering, 1999.

- [46] E. R. Dougherty and J. Barrera. Morphological Paradigm for Loss-function-based Design of Digital Filters. In R. D. Juday and S. K. Park, editors, *Visual Information Processing V*, volume 2753 of *SPIE Proceedings*, pages 90–97, Orlando, Florida, April 1996. SPIE - The International Society for Optical Engineering.
- [47] E. R. Dougherty and J. Barrera. Bayesian Design of Optimal Morphological Operators Based on Prior Distributions for Conditional Probabilities. *Acta Stereologica*, 16(3):167–174, 1997.
- [48] E. R. Dougherty and J. Barrera. Logical Image Operators. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Filters for Image Processing*, pages 1–60. SPIE and IEEE Press, Bellingham, 1999.
- [49] E. R. Dougherty, J. Barrera, G. Mozelle, S. Kim, and M. Brun. Multiresolution analysis for optimal binary filters. submitted, January 2000.
- [50] E. R. Dougherty, J. Barrera, and N. S. Tomita. Optimal Filters from Prior Filters. In *Nonlinear Image Processing VIII*, volume 3026 of *Proc. of SPIE*, pages 2–7, San Jose, CA, USA, February 1997.
- [51] E. R. Dougherty, A. M. Grigoryan, J. Barrera, and N. S. T. Hirata. Binary Filter Design: Optimization, Prior Information, and Robustness. In *Mathematical Modeling and Estimation Techniques in Computer Vision*, volume 3457 of *Proc. SPIE's International Symposium on Optical Science, Engineering and Instrumentation*, pages 230–241, San Diego, CA, July 1998.
- [52] E. R. Dougherty and R. P. Loce. Optimal Mean-Absolute-Error Hit-or-Miss Filters: Morphological Representation and Estimation of the Binary Conditional Expectation. *Optical Engineering*, 32(4):815–827, April 1993.
- [53] E. R. Dougherty and R. P. Loce. Precision of Morphological-Representation Estimator for Translation-invariant Binary Filters: Increasing and Nonincreasing. *Signal Processing*, 40:129–154, 1994.
- [54] E. R. Dougherty and R. P. Loce. Optimal Binary Differencing Filters: Design, Logic Complexity, Precision Analysis, and Application to Digital Document Processing. *Electronic Imaging*, 5(1):66–86, January 1996.
- [55] E. R. Dougherty, Y. Zhang, and Y. Chen. Optimal Iterative Increasing Binary Morphological Filters. *Optical Engineering*, 35(12):3495–3507, December 1996.
- [56] E. R. Dougherty and D. Zhao. Model-based Characterization of Statistically Optimal Design for Morphological Shape Recognition Algorithms via the Hit-or-Miss Transform. *Visual Communication and Image Representation*, 3(2):147–160, June 1992.
- [57] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973.
- [58] K.-C. Fan, L.-S. Wang, and Y.-K. Wang. Page segmentation and identification for intelligent signal processing. *Signal Processing*, 45(3):329–346, September 1995.
- [59] E. A. Filho. *Teoria Elementar dos Conjuntos*. Livraria Nobel S.A., São Paulo, 1980.

- [60] J. P. Fitch, E. J. Coyle, and N. C. Gallagher Jr. Median Filtering by Threshold Decomposition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-32(6):1183–1188, December 1984.
- [61] F. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 2nd edition, 1990.
- [62] K. Fukunaga and R. R. Hayes. Effects of Sample Size in Classifier Design. *IEEE on Pattern Analysis and Machine Intelligence*, 11(8), August 1989.
- [63] P. K. Ghosh. The Indecomposability Problem in Binary Morphology: An Algorithmic Approach. *Mathematical Imaging and Vision*, 6(2/3):169–198, June 1996.
- [64] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.
- [65] J. Goutsias. Morphological Analysis of Discrete Random Shapes. *Journal of Mathematical Imaging and Vision*, 2:193–215, 1992.
- [66] J. Goutsias and S. Batman. *Handbook of Medical Imaging: Volume 3 - Progress in Medical Image Processing and Analysis*, chapter Morphological Methods for Biomedical Image Analysis. SPIE Optical Engineering Press, 2000 (to appear).
- [67] J. Goutsias and H. J. A. M. Heijmans. Multiresolution Signal Decomposition Schemes. Part 1: Linear and Morphological Pyramids. Technical Report PNA-R9810, CWI, October 1998.
- [68] I. Guyon, J. Makhouli, R. Schwartz, and V. Vapnik. What Size Test Set Gives Good Error Rate Estimates? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):52–64, January 1998.
- [69] C. C. Han and K. C. Fan. Finding of Optimal Binary Morphological Erosion Filter via Greedy and Branch & Bound Searching. *Mathematical Imaging and Vision*, 6(4):335–353, December 1996.
- [70] N. R. Harvey and S. Marshall. The Use of Genetic Algorithms in Morphological Filter Design. *Signal Processing: Image Communication*, 8(1):55–71, January 1996.
- [71] R. F. Hashimoto. *Mudança de Estrutura de Representação de Operadores em Morfologia Matemática*. PhD thesis, Instituto de Matemática e Estatística - USP, 2000. in Portuguese.
- [72] M. H. Hassoun. *Fundamentals of Artificial Neural Net*. MIT Press, 1995.
- [73] D. Haussler. Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Information and Computation*, 100:78–150, 1992.
- [74] S. Haykin. *Neural Networks – A Comprehensive Foundation*. Mcmillan, 1994.
- [75] H. J. A. M. Heijmans. *Morphological Image Operators*. Academic Press, Boston, 1994.
- [76] C. B. Herwig and R. J. Schalkoff. Morphological Image Processing Using Artificial Neural Networks. *Control and Dynamic Systems*, 67, 1994.

- [77] F. J. Hill and G. R. Peterson. *Introduction to Switching Theory and Logical Design*. John Wiley, 3rd edition, 1981.
- [78] N. S. T. Hirata, J. Barrera, and E. R. Dougherty. Design of Statistically Optimal Stack Filters. In J. Stolfi and C. L. Tozzi, editors, *Proc. of Sibgrapi'99*, pages 265–274, Campinas, SP, Brazil, 1999.
- [79] N. S. T. Hirata, J. Barrera, and R. Terada. Text Segmentation by Automatically Designed Morphological Operators. In *Proc. of SIBGRAPI'2000*, pages 284–291, Gramado, Brazil, October 2000.
- [80] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. Design of Large-Window Binary Filters via Iteration. In *Intelligent Robots and Computer Vision XVII: Algorithms, Techniques, and Active Vision*, volume 3522 of *Proc. of SPIE*, pages 173–182, Boston, Massachusetts, November 1998.
- [81] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. Efficient Switching Algorithm for Designing Increasing Binary Filters. In E. R. Dougherty and J. T. Astola, editors, *Nonlinear Image Processing X*, volume 3646 of *Proc. SPIE*, pages 185–196, San Jose, CA, January 1999.
- [82] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. A Switching Algorithm for Design of Optimal Increasing Binary Filters Over Large Windows. *Pattern Recognition*, 33(6):1059–1081, June 2000. Special Issue on Mathematical Morphology & Nonlinear Image Processing.
- [83] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. Bayesian Switching Algorithm for the Optimal increasing Binary Filter. In *Proceedings of EUSIPCO - 2000*, volume IV, pages 1889–1892, 2000.
- [84] N. S. T. Hirata, E. R. Dougherty, and J. Barrera. Iterative Design of Morphological Binary Image Operators. *to appear*, 2000.
- [85] R. Hirata Jr., J. Barrera, and R. A. Lotufo. A Tutorial on Image Segmentation via Mathematical Morphology with Khoros. In *Khoros '97 proceedings*, pages 192–206. Khoros Research, Khoros Research Inc., Editors, March 1997.
- [86] R. Hirata Jr., E. R. Dougherty, and J. Barrera. Aperture Filters. *Signal Processing*, 80(4):697–721, April 2000.
- [87] R. Hirata Jr., E. R. Dougherty, and J. Barrera. Design of gray-scale nonlinear filters via multiresolution apertures. In *Proceedings of EUSIPCO 2000*, volume IV, pages 1905–1908, 2000.
- [88] R. Hirata Jr., E. R. Dougherty, and J. Barrera. Some Applications of Aperture Filters. In J. Goutsias, L. Vincent, and D. S. Bloomberg, editors, *Mathematical Morphology and its Applications to Image and Signal Processing*, volume 18, pages 119–128. Kluwer Academic Publishers, 2000. Fifth ISMM.
- [89] M. Iwanowski and J. Serra. Morphological Interpolation and Color Images. In *Proceedings of the 10th International Conference on Image Analysis and Processing*, 1999.

- [90] P. T. Jackway and M. Deriche. Scale-Space Properties of the Multiscale Morphological Dilation-Erosion. *Transactions on Pattern Analysis and Machine Intelligence*, 18(1):38–51, January 1996.
- [91] A. K. Jain, R. P.W. Duin, and J. Mao. Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), January 2000.
- [92] H. Joo. *Automatic Morphology*. PhD thesis, University of Washington, 1991.
- [93] H. Joo, R. M. Haralick, and L. G. Shapiro. Toward the Automatic Generation of Mathematical Morphology Procedures Using Predicate logic. In *Proc. of the Third International Conference on Computer Vision*, pages 156–165, Osaka, Japan, 1990.
- [94] B. K. Jenkins K. S. Huang and A. A. Sawchuk. Binary Image Algebra and Optimal Cellular Logic Processor Design. *Computer Vision, Graphics and Image Processing*, 45:295–345, 1989.
- [95] V. G. Kamat, E. R. Dougherty, and J. Barrera. Multiresolution Bayesian Design of Binary Filters. *submitted*, 2000.
- [96] R. H. Katz. *Contemporary Logic Design*. Benjamin Cummings, 1994.
- [97] M. J. Kearns. *The Computational Complexity of Machine Learning*. The MIT Press, 1990.
- [98] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.
- [99] H. Y. Kim. *Construção Automática de Operadores Morfológicos por Aprendizagem Computacional*. PhD thesis, Escola Politécnica da Universidade de São Paulo, 1997.
- [100] K. Kise, A. Sato, and M. Iwata. Segmentation of Page Images Using the Area Voronoi Diagram. *Computer Vision and Image Understanding*, 70(3):370–382, June 1998.
- [101] D. J. Kleitman and G. Markowsky. On Dedekind’s Problem: The Number of Isotone Boolean Functions II. *Transactions of the American Mathematical Society*, 213:373–390, November 1975.
- [102] A. F. Kohn. *Reconhecimento de Padrões - Uma Abordagem Estatística*. EPUSP, 1999.
- [103] T. Y. Kong and A. Rosenfeld. Digital Topology: Introduction and Survey. *Computer Vision, Graphics and Image Processing*, 48:357–393, 1989.
- [104] K. Konstantinides and J. Rasure. The KHOROS Software Development Environment for Image and Signal Processing. *IEEE Transactions on Image Processing*, 3(3):243–252, 1994.
- [105] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann Publishers, 1996.
- [106] J. S. Lee, I. Jurkevich, P. Dewaele, P. Wambacq, and A. CosterLinck. Speckle Filtering of Synthetic Aperture Radar Images : A Review. *Remote Sensing Reviews*, 8:313–340, 1994.
- [107] J.-H. Lin and Y.-T. Kim. Fast Algorithms for Training Stack Filters. *IEEE Transactions on Signal Processing*, 42(4):772–781, April 1994.

- [108] J.-H. Lin, T. M. Sellke, and E. J. Coyle. Adaptive Stack Filtering Under the Mean Absolute Error Criterion. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38(6):938–954, June 1990.
- [109] R. P. Loce. *Morphological Filter Mean-Absolute-Error Representation Theorems and Their Application to Optimal Morphological Filter Design*. PhD thesis, Center of Image Processing - Rochester Institute of Technology, 1993.
- [110] R. P. Loce and E. R. Dougherty. Facilitation of Optimal Binary Morphological Filter Design Via Structuring Element Libraries and Design Constraints. *Optical Engineering*, 31(5):1008–1025, May 1992.
- [111] R. P. Loce and E. R. Dougherty. Optimal Morphological Restoration: The Morphological Filter Mean-Absolute-Error Theorem. *Journal of Visual Communication and Image Representation*, 3(4):412–432, December 1992.
- [112] R. P. Loce and E. R. Dougherty. *Enhancement and Restoration of Digital Documents: Statistical Design of Nonlinear Algorithms*. SPIE - The International Society for Optical Engineering, Bellingham, 1997.
- [113] H. M. F. Madeira, J. Barrera, R. Hirata Jr, and N. S. T. Hirata. A New Paradigm for the Architecture of Morphological Machines : Binary Decision Diagrams. In J. Stolfi and C. L. Tozzi, editors, *Proc. SIBGRAPI'99 - XII Brazilian Symposium in Computer Graphics and Image Processing*, pages 283–292, Campinas, SP, Brazil, November 1999.
- [114] P. Maragos and R. W. Schafer. Morphological Filters: Part I: Their Set-Theoretic Analysis and Relations to Linear Shift-Invariant Filters. *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-35:1153–1169, August 1987.
- [115] P. Maragos and R. W. Schafer. Morphological Filters: Part II: Their Relations to Median, Order Statistic, and Stack-Filters. *IEEE Trans. Acoust. Speech Signal Process.*, ASSP-35:1170–1184, August 1987. Corrections in Vol. ASSP-37, April 1989, p. 597.
- [116] N. D. A. Mascarenhas. An Overview of Speckle Noise Filtering in SAR Images. In *Image Processing Techniques*, Proc. of First Latino-American Seminar on Radar Remote Sensing, pages 71–79, Buenos Aires, Argentina, 2-4 December 1996.
- [117] N. D. A. Mascarenhas and F. R. D. Velasco. *Processamento Digital de Imagens*. Escola de Computação, IME-USP, 1984.
- [118] G. Matheron. *Random Sets and Integral Geometry*. John Wiley, 1975.
- [119] A. V. Mathew, E. R. Dougherty, and V. Swarnakar. Efficient Derivation of the Optimal Mean-Square Binary Morphological Filter from the Conditional Expectation Via a Switching Algorithm for Discrete Power-Set Lattice. *Circuits, Systems, and Signal Processing*, 12(3):409–430, 1993.
- [120] P. C. McGreer, J. Sanghavi, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Espresso-Signature : A New Exact Minimizer for Logic Functions. *IEEE trans. on VLSI*, 1(4):432–440, December 1993.

- [121] F. Meyer and S. Beucher. Morphological Segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21–46, September 1990.
- [122] T. M. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. McGraw-Hill, March 1997.
- [123] I. Molchanov. *Statistics of the Boolean Model for Practitioners and Mathematicians*. John Wiley and Sons, 1997.
- [124] S. Mukhopadhyay and B. Chanda. A multiscale morphological approach to local contrast enhancement. *Signal Processing*, 80(4):685–696, April 2000.
- [125] J. R. F. Oliveira. *O uso de algoritmos genéticos na decomposição morfológica de operadores invariantes em translação aplicados a imagens digitais*. PhD thesis, Instituto Nacional de Pesquisas Espaciais - INPE, Av. dos Astronautas, 1758 - Jd da Granja CEP 12227.010 São José dos Campos - SP - Brasil, December 1998.
- [126] S. K. Pal and P. P. Wang, editors. *Genetic Algorithms for Pattern Recognition*. CRC Press, 1996.
- [127] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, 1982.
- [128] H. Park and R. T. Chin. Decomposition of Arbitrarily Shaped Morphological Structuring Elements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):2–15, January 1995.
- [129] K.-R. Park and C.-N. Lee. Scale-Space Using Mathematical Morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11), November 1996.
- [130] T. Pavlidis. Page Segmentation and Classification. *CVGIP: Graphical Models and Image Processing*, 54(6):484–496, November 1992.
- [131] M.A. Perkowski, T. Luba, S. Grygiel, P. Burkey, M. Burns, N. Iliev, M. Kolsteren, R. Lisanke, R. Malvi, Z. Wang, H. Wu, F. Yang, S. Zhou, , and J.S. Zhang. Unified Approach to Functional Decompositions of Switching Functions. Technical report, PSU Electrical Engineering Department, December 1995.
- [132] L. I. Perlovsky. Conundrum of Combinatorial Complexity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):666–670, June 1998.
- [133] D. Petrescu, I. Tăbuș, and M. Gabbouj. Adaptive Skeletonization Using Multistage Boolean and Stack Filtering. In *Proc. EUSIPCO*, volume VII, pages 951–954, September 1994.
- [134] W. K. Pratt. *Digital Image Processing*. John Wiley and Sons, 1991.
- [135] S. J. Raudys and A. K. Jain. Small Sample Size Effects in Statistical Pattern Recognition : Recommendations for Practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):252–264, March 1991.
- [136] C. H. Richardson and R. Schafer. A Lower Bound for Structuring Element Decompositions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):365–369, April 1991.

- [137] L. Robert and G. Malandain. Fast Binary Image Processing Using Binary Decision Diagrams. *Computer Vision and Image Understanding*, 72(1):1–9, October 1998.
- [138] C. Ronse. Fourier Analysis, Mathematical Morphology, and Vision. Technical Report Working Document WD54, Philips Research Laboratory, Brussels, June 1989.
- [139] K. A. Ross and C. R. B. Wright. *Discrete Mathematics*. Prentice Hall, 3rd edition, 1992.
- [140] R. Sabourin, G. Genest, and F. J. Prêteux. Off-line Signature Verification by Local Granulometric Size Distributions. *IEEE on Pattern Analysis and Machine Intelligence*, 19(9):976–988, September 1997.
- [141] F. Safa and G. Flouzat. Speckle Removal on Radar Imagery Based on Mathematical Morphology. *Signal Processing*, 16(4):319–333, 1989. Special issue on Advances in Mathematical Morphology.
- [142] P. Salembier and J. Serra. Flat Zones Filtering, Connected Operators, and Filters by Reconstruction. *IEEE Transactions on Image Processing*, 4(8):1153–1160, August 1995.
- [143] O. V. Sarca, E.R. Dougherty, and J. Astola. Optimal Binary Filters with Linearly Separable Preprocessing. In *Nonlinear Image Processing*, Proc. of SPIE, January 1998.
- [144] O. V. Sarca, E.R. Dougherty, and J. Astola. Two-stage Binary Filters. *Electronic Imaging*, 8(3):219–232, July 1999.
- [145] M. Schmitt. Mathematical morphology and artificial intelligence: an automatic programming system. *Signal Processing*, 16(4):389–401, April 1989.
- [146] D. Schonfeld. Optimal Structuring Elements for the Morphological Pattern Restoration of Binary Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):589–601, June 1994.
- [147] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.
- [148] J. Serra. *Image Analysis and Mathematical Morphology. Volume 2: Theoretical Advances*. Academic Press, 1988.
- [149] K. Sivakumar. *Morphological Analysis of Random Fields: Theory and Applications*. PhD thesis, Dept. of Electrical Engineering, The Johns Hopkins University, Baltimore, MD, 1997.
- [150] K. Sivakumar and J. Goutsias. On the Discretization of Morphological Operators. *Journal of Visual Communication and Image Representation*, 8(1):39–49, March 1997.
- [151] P. Soille. *Morphological Image Analysis*. Springer-Verlag, Berlin, 1999.
- [152] R. P. Sousa. *Projetos de Operadores Invariantes a Translação via treinamento de redes neurais*. PhD thesis, Universidade Federal da Paraíba, Campina Grande, Paraíba, Brasil, fevereiro 2000.
- [153] R. P. Sousa, J. M. Carvalho, F. M. Assis, and L. F. C. Pessoa. Designing translation invariant operations via neural network training. In *Proceedings of the International Conference on Image Processing - ICIP 2000*. IEEE Signal Processing Society, IEEE Signal Processing Society, 2000.

- [154] I. Tăbuș and B. Dumitrescu. A New Fast Method for Training Stack Filters. In *IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP'99)*, Antalya, Turkey, June 1999.
- [155] I. Tăbuș, D. Petrescu, and M. Gabbouj. A training Framework for Stack and Boolean Filtering – Fast Optimal Design Procedures and Robustness Case Study. *IEEE Transactions on Image Processing*, 5(6):809–826, June 1996.
- [156] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1998.
- [157] N. S. Tomita. Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC. Master's thesis, Instituto de Matemática e Estatística - Universidade de São Paulo, São Paulo, SP - Brasil, março 1996.
- [158] N. S. Tomita. Programação Automática de Máquinas Morfológicas Binárias baseada em Aprendizado PAC. In *Anais do XXIII Seminário Integrado de Software e Hardware - IX Concurso de Teses e Dissertações - XV Concurso de Trabalhos de Iniciação Científica*, pages 517–525, Recife, PE, Agosto 1996. XVI Congresso da SBC, Sociedade Brasileira de Computação.
- [159] P. Undrell and K. Delibasis. Stack Filter Design for Image Restoration - An Application of Genetic Algorithms. In *Proceedings of the 1997 International Conference on Image Processing (ICIP '97)*, volume II, 1997.
- [160] L. G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
- [161] V. N. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.
- [162] L. Vincent. Morphological Image Processing and Network Analysis of Cornea Endothelial Cell Images. In *Image Algebra and Morphological Image Processing III*, volume 1769, pages 212–226. SPIE, 1992.
- [163] L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, June 1991.
- [164] R. Vogt. *Automatic Generation of Morphological Set Recognition Algorithms*. Springer-Verlag, 1989.
- [165] P. D. Wendt, E. J. Coyle, and N. C. Gallagher Jr. Stack Filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-34(4):898–911, August 1986.
- [166] S. S. Wilson. Training Structuring Elements in Morphological Networks. In E. R. Dougherty, editor, *Mathematical Morphology in Image Processing*, chapter 1, pages 1–41. Marcel Dekker, 1993.
- [167] D. H. Wolpert, editor. *The Mathematics of Generalization*. Addison-Wesley, 1995. The Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning.
- [168] J. Xu. Decomposition of Convex Polygonal Morphological Structuring Elements into Neighborhood Subsets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):153–162, February 1991.

- [169] I. Yoda, K. Yamamoto, and H. Yamada. Automatic Acquisition of Hierarchical Mathematical Morphology Procedures by Genetic Algorithms. *Image and Vision Computing*, 17(10):749–760, August 1999.
- [170] J. Yoo, K. L. Fong, J.-J. Huang, E. J. Coyle, and G. B. Adams III. A Fast Algorithm for Designing Stack Filters. *IEEE Trans. on Image Processing*, 8(8):1014–1028, August 1999.
- [171] B. Zeng, Y. A. Neuvo, and A. Venetsanopoulos. Optimal Multistage Stack Filtering for Image Restoration. In *Proc. IEEE Intl. Symp. Circuits and Systems*, pages 117–120, November 1992.
- [172] X. Zhuang. Decomposition of Morphological Structuring Elements. *Mathematical Imaging and Vision*, 4(1):5–18, January 1994.
- [173] X. Zhuang and R. M. Haralick. Morphological Structuring Elements Decomposition. *Computer Vision, Graphics and Image Processing*, 35:370–382, 1986.