



**FACULDADE DE COMPUTAÇÃO
UNIVERSIDADE FEDERAL DE UBERLÂNDIA**



Sérgio Francisco da Silva

Realimentação de Relevância via Algoritmos Genéticos aplicada à Recuperação de Imagens

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.
Área de concentração: Banco de Dados.

Orientadora:
Prof^a Dr^a Celia Aparecida Zorzo Barcelos

Uberlândia, MG
2007

Sérgio Francisco da Silva

Realimentação de Relevância via Algoritmos Genéticos aplicada à Recuperação de Imagens

Dissertação de Mestrado apresentada à Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Banco de Dados.

Aprovação em 11 de janeiro de 2007.

Banca Examinadora:

Orientadora: _____
Dr^a Celia Aparecida Zorzo Barcelos – UFU

Avaliador 1: _____
Dr^a Agma Juci Machado Traina – USP

Avaliador 2: _____
Dr^a Gina Maira Barbosa de Oliveira – UFU

Uberlândia, MG
2007

‘O Amor
que dedicam a seus filhos en-
che o meu coração. O sacrifício que
fazem pelos seus filhos alivia o meu
fardo. A confiança que depositam
em seus filhos me faz sentir a capaci-
dade de conquistar qualquer sonho.
A vocês, meus queridos pais,
dedico esta dissertação, por
tudo o que são, e por tudo
o que me possibili-
tam ser. Com
Amor.’



Agradecimentos

Agradeço a todos aqueles que contribuíram de alguma forma para o desenvolvimento dessa dissertação. Em especial agradeço a . . .

Celia Aparecida Zorzo Barcelos pelo carinho, pelo apoio em todos os aspectos, pelos conselhos, pela orientação, por me aplaudir, por “puxar minhas orelhas”. **Célia** muito obrigado por tudo!

Marcos Aurélio Batista pela presteza na colaboração em todas as fases deste trabalho, sempre com ótimas idéias, além de sua grande colaboração na revisão desta dissertação.

Emílio Barcelos por sua presteza ao apresentar um dos artigos aceitos deste trabalho em Budapest, Hungria.

aos Professores **Gina M. B. de Oliveira**, **Carlos R. Lopes**, **Denise Guliato**, **Ilmério R. da Silva** e **Márcia A. Fernandes**, pelas disciplinas de Computação Evolutiva, Inteligência Artificial, Processamento de Imagens, Recuperação de Informação e Análise de Algoritmos, respectivamente.

Carivaldo Ferreira da Silva e **Maria de Fátima Lima e Silva**, meus queridos pais, que me deram todo o suporte e motivação necessários, e tudo isso com toda a dignidade que se pode ter.

Márcio Dias, **Vinícius Borges**, **Cristiane Santos**, **Alexandre Fieno**, **Ivan Lopez** e **Juliana Franciscani** por estarem comigo nesta jornada e pela amizade.

Resumo

O principal objetivo de um sistema de recuperação de imagens é obter imagens que são o mais similar possível à requisição do usuário, de todas as imagens de uma coleção de referência. Tal objetivo é difícil de ser alcançado devido principalmente à subjetividade do conceito de similaridade entre imagens, visto que uma mesma imagem poder ser interpretada de diferentes maneiras por diferentes pessoas. Na tentativa de resolver este problema os sistemas de recuperação de imagens por conteúdo exploram as características de cor, forma e textura, quase sempre associadas à regiões e usam de mecanismos de realimentação de relevantes para ajustar uma busca aos critérios do usuário. Várias abordagens têm sido usadas em realimentação de relevância entre as quais os algoritmos genéticos têm se tornado bastante populares devido às suas habilidades adaptativas. Neste trabalho apresentamos um sistema de recuperação de imagens com base na similaridade de padrões locais, empregando as características de cor, forma e textura e com realimentação de relevância via algoritmo genético. A tarefa do algoritmo genético é inferir pesos para as características de cor, forma, textura e regiões que melhor ajustam a medida de similaridade entre imagens aos critérios de busca do usuário, fazendo com que o *ranking* final esteja de acordo com os critérios expressos na realimentação. Da teoria dos algoritmos genéticos é conhecido que a medida de aptidão exerce um papel essencial na performance destes algoritmos, uma vez que ela direciona o caminho da busca, por avaliar a aptidão dos indivíduos. Devido à falta de consenso acerca da medida de aptidão ideal na avaliação de *rankings* apresentamos uma análise de performance de dez medidas de aptidão. As funções de aptidão são classificadas em dois grupos: baseadas em ordem e não baseadas em ordem. Algumas destas funções são adaptadas do contexto de sistemas de recuperação de informação e outras são propostas neste trabalho. Os resultados experimentais mostram que as funções de aptidão baseadas em ordem são mais compatíveis aos interesses dos usuários uma vez que elas apresentam *rankings* superiores em precisão para baixos níveis de revocação e, conduzem mais rapidamente o AG na busca por uma solução heurísticamente ótima. Os resultados obtidos são superiores aos dos trabalhos de Stejić et al. que nos serviram de inspiração.

Palavras-chave: Recuperação de imagens por conteúdo, realimentação de relevância, algoritmos genéticos.

Abstract

The principal objective of an image retrieval system is to obtain images which are as similar as possible to the user's requirements, from all the images in the reference collection. Such an objective is difficult to reach due principally to the subjectivity of the image similarities. This is due to the fact the images can be interpreted in different ways by different people. With the aim of resolving this problem the content-based image retrieval systems explore the features of color, shape and texture. These are nearly always associated to the regions and use relevance feedback mechanisms to adjust a search to the user's criterions. Various approaches have been used in relevance feedback from those genetic algorithms have become quite popular due to their adaptive abilities. In this work we presented an image retrieval system based on the similarity of local patterns, working with the features of color, shape and texture as well as relevance feedback via a genetic algorithm. The task of this algorithm is infer weights to the features of color, shape, texture and regions which better adjust to the similarity found between images through the user's search criterions, thus producing a final ranking which is in accordance with the criterions expressed in the relevance feedback. The genetic algorithms theory states that the fitness measure applies an essential role upon the performance of these algorithms, once the fitness measure directs the search path for the evaluation of each individuals aptitude. Due to the lack of consensus about the best fitness measure in the ranking evaluation problem we present a performance analysis of ten fitness functions. The fitness functions are classified in two groups: order-based and non-order based. Some of these functions are adapted from text-based information retrieval systems and others are proposed in this work. The experimental results show that the order based fitness functions are more compatible to the user's interests, once they present superior rankings in terms of precision for low recall rates and conduct the quickest genetic algorithm in the search for an optimal heuristic solution. The results obtained are superior to those of the works of Stejić et al., which served as our inspiration.

Keywords: Content-based image retrieval, relevance feedback, genetic algorithms.

Sumário

Lista de Figuras	xi
Lista de Tabelas	xv
Lista de Algoritmos	xvii
Introdução	xix
I Revisão de Literatura	1
1 Algoritmos Genéticos	3
1.1 A Inspiração Biológica	3
1.2 Definições	4
1.3 Características dos AGs	5
1.4 Ciclo de Execução dos AGs	5
1.4.1 Representação Cromossômica	6
1.4.2 População Inicial	7
1.4.3 Medida de Aptidão	7
1.4.4 Seleção	9
1.4.5 <i>Crossover</i>	10
1.4.6 Mutação	13
1.4.7 Reinservação	13
1.4.8 Condições de Parada	14
1.5 Parâmetros de Controle	14
1.6 Problemas Práticos dos AGs	15
1.6.1 Convergência Prematura	15
1.6.2 Convergência Lenta	16
1.6.3 Tempo de Processamento Elevado	16
1.7 Diversidade Populacional e a Pressão Seletiva	16
1.7.1 Diversidade Populacional	16
1.7.2 Pressão Seletiva	17

1.8	Considerações Finais	17
2	Recuperação de Imagens	19
2.1	O Uso de Imagens	19
2.2	A Recuperação de Imagens	20
2.3	Recuperação de Imagens por Conteúdo	22
2.4	Caracterização	23
2.5	Cor	23
2.5.1	Momentos de Cor	24
2.5.2	Histograma de Cores	25
2.5.3	Vetor de Coerência de Cores	25
2.5.4	Correlograma de Cores	25
2.5.5	Características Invariantes de Cores	26
2.6	Forma	26
2.6.1	Técnicas Baseadas no Contorno	26
2.6.2	Técnicas Baseadas em Regiões	31
2.7	Textura	33
2.7.1	Vizinhança de Textura	35
2.7.2	Matrizes de Co-ocorrência	35
2.7.3	Características Tamura	35
2.7.4	Características <i>Wold</i>	37
2.7.5	Características da Filtragem <i>Gabor</i>	37
2.7.6	Características da Transformada <i>Wavelet</i>	38
2.8	Similaridade	39
2.8.1	Modelos de Similaridade	39
2.8.2	Medidas de Similaridade	39
2.9	Sistemas de Recuperação de Imagens	41
2.9.1	Metodologias de Busca	41
2.9.2	Realimentação de Relevância	43
2.10	Medidas de Performance	45
2.10.1	Curva Precisão-Revocação (<i>Precision-Recall</i>)	45
2.10.2	Precisão- <i>R</i>	46
2.10.3	<i>Average Normalized Modified Retrieval Rank</i>	46
2.10.4	<i>Average Normalized Similarity Metric Difference</i>	47
2.11	Considerações Finais	47
3	Realimentação de Relevância via Computação Evolutiva	49
3.1	Introdução	49
3.2	Fan et al.	50
3.2.1	O Problema	50
3.2.2	Programação Genética	50
3.2.3	Especificação do Algoritmo de Programação Genética	50
3.2.4	Funções de Aptidão	52
3.2.5	Avaliação Experimental	54

3.3	López-Pujalte et al.	55
3.3.1	O Problema	55
3.3.2	Especificação do Algoritmo Genético	55
3.3.3	Avaliação Experimental	56
3.4	Stejić et al.	57
3.4.1	Revisão e Problemas	57
3.4.2	Especificação do Algoritmo Genético	59
3.4.3	Detalhes Adicionais da Técnica de Stejić et al.	59
3.4.4	Avaliação Experimental	60
3.5	Torres et al.	61
3.5.1	O Problema	61
3.5.2	A Solução via Programação Genética	61
3.5.3	Avaliação Experimental	62
3.6	Considerações Finais	63
II	Nossa Contribuição	65
4	O Sistema Proposto	67
4.1	Estrutura Geral	67
4.2	Descritores de Características	68
4.2.1	Momentos de Cor	68
4.2.2	Histograma de Direção de Bordas	71
4.2.3	Vizinhança de Textura	72
4.3	O Modelo de Similaridade entre Imagens	72
4.3.1	Realimentação de Relevância Baseada em AGs	74
4.4	O Algoritmo Genético	74
4.4.1	A Codificação de Indivíduo	74
4.4.2	Os Operadores Genéticos	75
4.4.3	Os Parâmetros de Controle	76
4.4.4	As Funções de Aptidão	76
4.5	Considerações Finais	78
5	Estudo de Complexidade do Algoritmo Proposto	81
5.1	Introdução	81
5.1.1	<i>Insertion-sort</i>	83
5.1.2	Ordem de Crescimento de Funções e Notação Assintótica	85
5.2	Classes de Comportamento Assintótico	86
5.2.1	<i>Merge-sort</i>	88
5.3	Algoritmos do Sistema Proposto	91
5.4	Análise de Complexidade	95
5.4.1	Busca Inicial	95
5.4.2	Realimentação de Relevantes	95
5.5	Considerações Finais	96

6	Resultados Experimentais	97
6.1	Bancos de Dados de Teste	97
6.2	Avaliação Experimental	100
6.2.1	A Abordagem Proposta <i>versus</i> a de SjetiĆ et al.	101
6.2.2	Análise das Funções de Aptidão	102
6.2.3	Outros Resultados	107
6.3	Considerações Finais	113
7	Trabalhos Futuros e Perspectivas	119
7.1	Realimentação via Algoritmo Genético usando Função de Aptidão Multiobjetivos	119
7.1.1	Algoritmos Genéticos com Função de Aptidão Multiobjetivo	120
7.1.2	Exemplos Positivos e Negativos: Otimização Multiobjetivo	120
7.1.3	Resultados Preliminares	121
7.1.4	Complemento da Proposta	122
7.2	Trabalhos Futuros	122
7.3	Perspectivas	124
	Conclusão	125
	Referências bibliográficas	126
III	Apêndice	135
A	Funções de Aptidão Adaptadas do Contexto de Recuperação de Informação Textual	137
B	Outros Resultados	139

Lista de Figuras

1.1	Ciclo de execução básico de um AG	6
1.2	Ilustração da convergência de uma população	8
1.3	Ilustração de uma roleta	9
1.4	<i>Crossover</i> simples	11
1.5	<i>Crossover</i> múltiplo	11
1.6	<i>Crossover</i> uniforme	12
1.7	<i>Crossover PMX</i>	12
1.8	<i>Crossover</i> cíclico	12
1.9	Mutação simples	13
1.10	Classificação das técnicas de ajuste de parâmetros	15
2.1	Fluxograma de um sistema <i>CBIR</i> típico	22
2.2	Classificação geral das técnicas de representação de formas	27
2.3	Uma forma e sua assinatura obtida pelo método da distância ao centro	28
2.4	Código da cadeia	29
2.5	Aproximação por Polígonos	30
2.6	Análise sintática da forma de um cromossomo	30
2.7	Fecho convexo	33
2.8	Saliências	33
2.9	Imagens de texturas naturais	34
2.10	O Sistema <i>ImageGrouper</i>	42
2.11	O Sistema <i>PicSom</i>	42
2.12	O Sistema <i>CIRES</i>	44
3.1	Representação em árvore da função de <i>ranking TF-IDF</i>	51
3.2	Exemplificação de uma operação de <i>crossover</i> em árvores	52
3.3	Exemplificando o modelo <i>LSP</i>	57
3.4	Exemplo de uma possível função de combinação dos descritores d_1, d_2 e d_3 , representada por uma árvore	61
4.1	Fluxograma básico do sistema <i>CBIR</i> proposto	68
4.2	Sistema de cor <i>RGB</i>	69

4.3	Visualização dos canais de cor R , G e B	70
4.4	Sistema de cor HSI	70
4.5	Visualização dos canais de cor H , S e I	70
4.6	Imagem original e as imagens do gradiente da matiz, saturação e intensidade, respectivamente	71
4.7	Operadores de Sobel	72
4.8	Ilustração da divisão de uma imagem em regiões	72
4.9	Ilustração de uma operação de <i>crossover</i>	75
4.10	Ilustração de uma operação de mutação	76
5.1	Ordenando cartas manualmente conforme o algoritmo <i>insertion-sort</i>	84
5.2	Ilustração gráfica das notações O , Ω e Θ	86
5.3	Árvore de recursão do algoritmo <i>Merge-sort</i>	90
6.1	Amostra do banco de dados <i>Vistex-60</i>	98
6.2	Amostra do banco de dados <i>Brodatz-208</i>	98
6.3	Amostra do banco de dados <i>Corel-1000</i>	99
6.4	Amostra do banco de dados <i>BD-10000</i>	99
6.5	Amostra do banco de dados <i>BD-12750</i>	100
6.6	Curvas Precisão-Revocação para as várias funções de aptidão nos bancos de dados, <i>Vistex-167</i> , <i>Corel-1000</i> e <i>BD-10000</i> , respectivamente	103
6.7	Resultados de busca obtidos para a categoria África no banco de dados <i>Corel-1000</i> , através das funções de aptidão $F_{Stejic\ et\ al.}$, $F1_{Proposta}$, $F2_{Proposta}$ e $F3_{Fan\ et\ al.}$, respectivamente	104
6.8	Resultado de busca obtido para a categoria leopardo, no banco de dados <i>BD-10000</i> , usando as funções de aptidão $F_{Stejic\ et\ al.}$ e $F2_{Proposta}$, respectivamente	106
6.9	Resultado de uma busca por pássaros no banco de dados <i>BD-12750</i> usando as funções de aptidão $F2_{Proposta}$ e $F_{Stejic\ et\ al.}$, respectivamente	108
6.10	Resultado de uma busca por cavalos no banco de dados <i>BD-12750</i> , usando as funções de aptidão $F2_{Proposta}$ e $F_{Stejic\ et\ al.}$, respectivamente	108
6.11	Resultado de uma busca por aviões no banco de dados <i>BD-12750</i> , usando a função de aptidão $F2_{Proposta}$	110
6.12	Resultado de busca para a imagem de consulta construída pela metade de uma imagem de cavalo e pela metade de uma imagem de montanha, no banco de dados <i>Corel-1000</i>	111
6.13	Resultado de uma busca por árvores no banco de dados <i>BD-12750</i>	112
6.14	Resultado de uma busca por carros no banco de dados <i>BD-12750</i> , usando a função de aptidão $F2_{Proposta}$	112
6.15	Resultado de uma busca por ônibus no banco de dados <i>Corel-1000</i> , usando a função de aptidão $F2_{Proposta}$	114
6.16	Consultando faces no banco de dados <i>BD-12750</i>	115
6.17	Resultado de uma busca por flores no banco de dados <i>BD-12750</i> , usando a função de aptidão $F2_{Proposta}$	116
6.18	Recursos da interface da nossa aplicação	117

7.1	Interface básica para a classificação de exemplos em positivos e negativos	119
7.2	Resultado de uma busca por flores no banco de dados <i>BD-12750</i>	121
7.3	Resultado de uma busca por interiores de igrejas no banco de dados <i>BD-12750</i> .	123
7.4	Resultado de uma busca por zebras no banco de dados <i>BD-12750</i>	124
B.1	Resultados de busca por rosas vermelhas, mini-flores, casca de árvores, árvores, primavera e elefantes, respectivamente, no banco de dados <i>BD-12750</i>	140
B.2	Resultados de busca por cobras, torres, pássaros, textura, peixes e motos, respectivamente, no banco de dados <i>BD-12750</i>	141
B.3	Resultado de busca por imagens de montanhas polares no banco de dados <i>Corel-1000</i>	142
B.4	Resultado de busca por imagens de pôr-do-sol, no banco de dados <i>BD-12750</i> . .	142

Lista de Tabelas

2.1	Características deriváveis da matriz de co-ocorrência	35
3.1	Símbolos terminais usados no sistema PG de [Fan et al. 2004]	51
3.2	Resultados obtidos por [Fan et al. 2004]	55
3.3	Resultados obtidos por [López-Pujalte et al. 2003b]	57
3.4	Mutações experimentadas por [Stejić et al. 2003a]	60
3.5	Precisão de recuperação dos melhores métodos de [Stejić et al. 2003a]: $LSP-C\pm$ e $WLSP-C\pm$	60
3.6	Resultados obtidos por [Torres et al. 2005]	63
6.1	Precisão- R da técnica proposta para as várias funções de aptidão experimenta- das, em contraste com os modelos $WLSP-C\pm$ e $LSP-C\pm$ de [Stejić et al. 2003a]	102
6.2	Tempo médio de execução gasto para as várias funções de aptidão experimentadas	105
6.3	Número de gerações médio gasto pelo AG na busca do valor de aptidão máximo usando as funções de aptidão $F2_{Proposta}$ e $F_{Stejić et al.}$	107

Lista de Algoritmos

3.1	Sistema de Programação Genética de [Fan et al. 2004] para a descoberta de funções de <i>ranking</i>	53
3.2	Sistema de Programação Genética de [Torres et al. 2005] para a descoberta de funções de combinação de descritores de imagens	62
5.1	<i>Insertion-sort</i>	84
5.2	Algoritmo <i>merge-sort</i>	88
5.3	Passos principais do algoritmo genético proposto	91
5.4	Função de geração da população inicial	92
5.5	Função de avaliação da aptidão de um indivíduo	92
5.6	Função para a geração de um <i>ranking</i> considerando uma consulta q , um indivíduo C e a coleção de referência I	92
5.7	Fluxo do processo de reprodução: seleção, <i>crossover</i> e mutação	93
5.8	Função de montagem de uma roleta	93
5.9	Função de sorteio pelo método da roleta	93
5.10	Função de <i>crossover</i>	94
5.11	Função de Mutação	94
5.12	Função de reinserção	94

Introdução

Atualmente existem na literatura vários estudos inspirados em fenômenos da natureza, cuja idéia principal é reproduzir artificialmente comportamentos e ações inteligentes observados no dia-a-dia: redes neurais, computação evolutiva, estratégias presa-predador, colônias de insetos, etc. Uma das várias aplicações destes estudos, em especial as técnicas de computação evolutiva, é na área de recuperação de informação [Cordón et al. 2003].

Entre as técnicas de computação evolutiva destacam-se os algoritmos genéticos (AGs) [Goldberg 1989; Haupt and Haupt 1998; Holland 1975] que são métodos de busca inspirados na teoria de seleção natural e nos mecanismos genéticos. Os AGs são vistos há algum tempo como uma ferramenta útil para a recuperação de informação, devido ao seu poder de adaptação, podendo ser aplicados na adaptação de mecanismos de busca às necessidades dos usuários [Cordón et al. 2003; López-Pujalte et al. 2003b; Stejić et al. 2003a].

Paralelamente às pesquisas inspiradas na natureza, nota-se uma rápida ascensão da tecnologia multimídia, da *World Wide Web* e conseqüentemente o aumento das coleções de dados multimídia: de imagens, vídeos, músicas, etc. Este fato tem atraído inúmeras pesquisas sobre como organizar e recuperar de forma rápida e eficaz estas informações ou dados. Dentro deste contexto, este trabalho trata da recuperação de imagens, tendo como meta o desenvolvimento de um sistema que se adapte às necessidades de informação dos usuários, uma vez que esta tarefa é inquestionavelmente subjetiva [Castelli and Bergman 2002; Crucianu et al. 2004; Rui et al. 1997].

Para ser efetivo na tentativa de satisfazer as necessidades de informação dos usuários, um sistema de recuperação de imagens deve, de alguma forma, interpretar o conteúdo das imagens de uma coleção e ordená-las de acordo com o grau de relevância para a busca do usuário [Crucianu et al. 2004]. Esta interpretação do conteúdo de uma imagem envolve a extração de informações sintáticas e/ou semânticas das imagens e o uso destas na comparação com a requisição de informação do usuário. A dificuldade aqui é não somente a de extrair as informações mas também de como usá-las para decidir a relevância das imagens.

Devido à dificuldade e ao custo de produzir anotações textuais ricas e confiáveis para grandes coleções de imagens e ao gap-semântico ou subjetividade associada a estas anotações, praticamente todas as pesquisas atuais em recuperação de imagens utilizam-se do conteúdo visual das imagens para compor suas descrições – Recuperação de Imagens por Conteúdo (CBIR¹).

Juntamente com a substituição de anotações textuais por extração automática de características, popularizou-se o processo de consulta por exemplo visual (*query by visual example*) [Castelli and Bergman 2002; Crucianu et al. 2004] – o processo inicia pela apresentação de uma

¹CBIR: sigla derivada do inglês: *Content-Based Image Retrieval*.

imagem exemplo (consulta) ao *sistema de busca* que retornará as imagens visualmente mais similares à consulta.

Devido ao fato da percepção visual humana ser o padrão principal para comparação de eficiência na recuperação de imagens, o desenvolvimento de um “sistema ideal” é bastante desafiador. Assim como a percepção visual humana é um processo inteligente, capaz de extrair informações relevantes em uma cena e de fazer inferências e generalizações sobre conhecimentos prévios, acredita-se que um sistema de recuperação de imagem efetivo deva ser dotado de tais capacidades, ou seja, deva possuir mecanismos inteligentes, capazes de fazer inferências e/ou generalizações.

Partindo do julgamento de relevância do usuário podemos montar uma base de conhecimento, com a qual podemos fazer inferências para decidir quais são as informações de fato relevantes no contexto de uma busca. A idéia principal neste trabalho é inferir pesos de atributos (de cor, forma e textura) e de regiões na intenção de adaptar a medida de similaridade entre imagens aos critérios do usuário, de modo que mais imagens relevantes serão recuperadas [Crucianu et al. 2004; Salton and Buckley 1990]. Tal inferência de pesos será feita por um algoritmo genético projetado exclusivamente para este problema.

Da teoria dos algoritmos genéticos é conhecido que a medida de aptidão exerce um papel essencial no desempenho destes, uma vez que esta direciona o caminho da busca. Dado que nosso problema de realimentação de relevância é encarado como um problema de otimização do *ranking* final retornado ao usuário, a função da medida de aptidão é ponderar os indivíduos conforme a qualidade dos *rankings* proporcionados por estes. Como não há consenso acerca da medida de avaliação de um *ranking*, é apresentado neste trabalho uma análise de performance de dez função de aptidão aplicadas ao problema. Funções estas classificadas em dois grupos: baseadas em ordem e não baseadas em ordem, conforme seus princípios de construção.

Na intenção de expor com clareza os conceitos, as técnicas e os resultados obtidos neste trabalho, estruturamos este em três partes.

A primeira parte trata-se de uma revisão de literatura e é estrutura da seguinte forma:

- No **Capítulo 1** são apresentados as definições, inspirações, conceitos e fundamentos dos algoritmos genéticos.
- No **Capítulo 2** são apresentadas as noções básicas de recuperação de imagens, com foco voltado para a recuperação de imagens por conteúdo: descritores de imagens, medidas e modelos de similaridade, metodologias de busca, noções de realimentação de relevância e medidas de performance de recuperação são os principais tópicos deste capítulo.
- No **Capítulo 3** são apresentados descrições de quatro trabalhos recentes relacionados à aplicação de técnicas de computação evolutiva em mecanismos de realimentação de relevância. São eles:
 - [Fan et al. 2004] – analisaram, no contexto de recuperação de informação textual, os efeitos de funções de aptidão em programação genética destinada a descoberta de funções de *ranking* otimizadas conforme um conjunto de treinamento;
 - [López-Pujalte et al. 2003b] – analisaram, no contexto de recuperação de informação textual, a eficiência de funções de aptidão baseadas em ordem em um algoritmo

genético aplicado à realimentação de relevância. O objetivo do AG era inferir pesos de termos representados conforme o modelo espaço vetorial;

- [Stejić et al. 2003a] – utilizaram algoritmos genéticos para inferir, a partir da realimentação do usuário, pesos de características e de regiões de imagens, na tentativa de ajustar um modelo de similaridade entre imagens às necessidades momentâneas de busca do usuário;
- [Torres et al. 2005] – utilizaram programação genética para descobrir funções de combinação de descritores na recuperação de imagens.

A segunda parte é dedicada à descrição da técnica proposta, análise e comparações dos resultados obtidos, bem como propostas de trabalhos futuros:

- No **Capítulo 4** é apresentado o sistema de recuperação de imagens por conteúdo proposto;
- No **Capítulo 5** são apresentados em detalhes o algoritmo do sistema proposto e a análise de sua complexidade de tempo.
- No **Capítulo 6** são apresentadas as avaliações experimentais: comparação da nossa técnica com a de Stejić et al., comparação das várias funções de aptidão testadas através da medida de Precisão- R , curva Precisão-Revocação, custo computacional e velocidade de convergência do algoritmo genético. A boa performance da nossa técnica é ilustrada através de uma série de experimentos.
- No **Capítulo 7** são apresentados um trabalho em andamento referente à realimentação de relevância com exemplos positivos e negativos via algoritmos genéticos multi-objetivos, e propostas de trabalhos futuros.

E, finalmente, na terceira parte colocamos dois apêndices onde, no primeiro deles adaptamos para o nosso contexto, as funções de aptidão incorporadas ao problema de realimentação de relevância via algoritmo genético. No segundo apêndice apresentamos mais alguns resultados obtidos pela abordagem proposta.

Parte I

Revisão de Literatura

Algoritmos Genéticos

Os Algoritmos Genéticos são métodos de busca não determinísticos baseados na abstração do conceito de evolução natural. Eles operam sobre uma população de soluções candidatas, que evoluem de acordo com operadores genéticos concebidos a partir de metáforas biológicas, onde as melhores soluções têm maiores probabilidades de se combinarem e “sobreviverem”.

Atualmente, os Algoritmos Genéticos (AGs) proporcionam um meio bastante poderoso e simples de executar tarefas de otimização, simulações, auto-aprendizado (descoberta de regras de classificação), adaptação (ajuste de métricas) e estudos de fenômenos em um computador.

Neste Capítulo são apresentados os principais fundamentos dos AGs: a inspiração, definições, procedimentos, parâmetros de configuração, problemas e demais conceitos relacionados ao tema.

1.1 A Inspiração Biológica

Há tempos o homem busca na natureza inspirações para a criação de tecnologias que melhorem sua vivência no dia-a-dia. São exemplos popularmente conhecidos de inventos e suas respectivas inspirações: aviões e pássaros, submarinos e peixes e, sonares e morcegos.

Na comunidade científica atual são numerosos os estudos sobre métodos e técnicas inspiradas na natureza: Redes Neurais inspiradas no funcionamento do cérebro humano [Arbib 2003], sistemas de otimização inspirados no comportamento de colônias de insetos [Dorigo and Caro 1999], Computação Evolutiva inspirada na teoria da evolução das espécies [Bäck et al. 2000], etc. Estas pesquisas compõem uma das maiores ciências atuais, a Inteligência Artificial (IA), cuja idéia principal é reproduzir artificialmente comportamentos e ações inteligentes observados na natureza.

A técnica de Algoritmos Genéticos é umas das mais difundidas da Computação Evolutiva. Eles foram criados nos anos 60 pelo pesquisador John Holland, na Universidade de Michigan, USA. Com base nos estudos do biólogo Fisher, acerca da seleção natural [Fisher 1930], Holland percebeu um nítido elo entre a biologia e a computação: as máquinas poderiam se adaptar ao meio ambiente, assim como os seres vivos. Conforme sua convicção, a evolução natural era tal como a aprendizagem, uma forma de adaptação, sendo que a diferença entre elas estava apenas

na duração do processo, várias gerações, ao invés de uma vida. E também, já que a evolução funcionava tão bem para os seres vivos, o mesmo ocorreria com as máquinas.

O conceito de evolução natural, ou seleção natural, define a natureza como um processo de seleção de seres vivos. Numa determinada população, quando há escassez de recursos, sejam eles comida, espaço, ou outro recurso essencial, os indivíduos mais preparados para a competição dominam os mais fracos e sobrevivem. Isso acontece porque, dentre todas as características imprescindíveis à competição, os seres sobreviventes possuem algumas mais acentuadamente presentes que os outros. Por herança, essas características provavelmente passarão para seus descendentes, e assim, eles também terão grandes chances de saírem vencedores.

Na concepção da genética, um processo evolutivo natural só ocorre se: houver uma população de indivíduos, os indivíduos tiverem a capacidade de reproduzir, houver variedade entre os indivíduos, e a habilidade de sobrevivência estiver associada a essa variedade.

Para Holland, essa semântica da evolução natural poderia levar as máquinas a evoluírem, assim que fosse desenvolvida uma sintaxe artificial ou um modelo matemático que a suportasse.

Assim, o modo como a evolução foi implantada nas máquinas consistiu de partir de um conjunto de possíveis soluções ao acaso e aplicar sobre estas mecanismos inspirados na natureza, desta forma, emergindo um comportamento espontâneo. A evolução de uma população por várias gerações foi transcrito como um processo iterativo de melhoramento das soluções de um problema. As leis da natureza que determinam a sobrevivência dos mais aptos (seleção natural) foram representadas por operadores artificiais probabilísticos de seleção, sendo a aptidão de um indivíduo tomada a partir de alguma medida heurística acerca da qualidade da solução que ele representa.

Esta técnica, definida por Holland, foi denominada Algoritmo Genético (AG), e divulgada inicialmente em 1975 em seu livro intitulado: *Adaptation in Natural and Artificial Systems*. Posteriormente, a descoberta de Holland ganhou o mundo graças a seu aluno David E. Goldberg, através do livro, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Atualmente, os AGs dão suporte a várias aplicações computacionais (otimizações em geral, auto-aprendizado, auto-adaptação, simulação, etc.), nas mais variadas áreas do conhecimento (matemática, biologia, física, química, robótica, etc.).

1.2 Definições

Dentre as várias definições de Algoritmos Genéticos encontradas na literatura, vamos destacar algumas delas:

Segundo [Holland 1975], Algoritmos Genéticos são programas de computador que “*evoluem*” em um caminho que se assemelha à seleção natural podendo resolver problemas complexos, até mesmo aqueles que seus criadores não compreendem completamente.

Conforme [Goldberg 1989], Algoritmos Genéticos são métodos computacionais de busca, baseados nos mecanismos da evolução natural e na genética. Em AGs uma população de possíveis soluções para um problema, evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua.

De acordo com [Goldberg 1989], os AGs combinam a sobrevivência do melhor adaptado

dentre estruturas formadas por sequência de *bits*, com uma troca de informação randômica e estruturada para formar um algoritmo computacional com um “faro” inovador de busca. Apesar de aleatórios, os Algoritmos Genéticos não são uma simples caminhada aleatória. Eles exploram eficientemente informações históricas para especular novos pontos no espaço de busca com um aumento esperado de performance.

Segundo [Austin 1990], citado por [Oliveira 1999], na evolução biológica, a sobrevivência é uma medida de desempenho. Qualquer criatura viva pode ser considerada uma solução estrutural em seu ecossistema. Um Algoritmo Genético é um procedimento iterativo que mantém sempre uma população de estruturas que são candidatas à solução do problema. Durante cada incremento temporal, chamado geração, as estruturas na população corrente são avaliadas através de uma medida de desempenho que indica o quão próxima uma estrutura está de ser a solução do problema. Baseada nestas avaliações, uma nova população de soluções candidatas é formada, utilizando três operadores genéticos: seleção, *crossover* e mutação. Cada ponto no espaço de busca do problema é um indivíduo da população, representado no sistema por uma estrutura representando seus cromossomos, uma cadeia de símbolos de tamanho normalmente fixo.

1.3 Características dos AGs

[Goldberg 1989] afirma que os AGs se diferem da maioria dos procedimentos de busca e otimização em quatro princípios básicos:

- AGs operam num espaço de soluções codificadas (genótipos), e não diretamente no espaço de busca (fenótipos). Entretanto, muitos AGs atuais operam diretamente no espaço de busca.
- AGs operam num conjunto de pontos (população de indivíduos), e não a partir de um ponto isolado.
- AGs não necessitam de conhecimentos auxiliares além da representação das soluções e da estimação da qualidade destas.
- AGs usam regras de transição probabilísticas, e não regras determinísticas.

1.4 Ciclo de Execução dos AGs

O ciclo básico de execução de um AG é mostrado no fluxograma da Figura 1.1 [Oliveira 1999] e seu funcionamento básico é o seguinte.

Após eleita uma representação das possíveis soluções (indivíduos) gera-se uma população inicial. Os indivíduos desta população são avaliados e caso algum indivíduo atinja o critério de parada o ciclo é terminado. Senão parte dos indivíduos serão selecionados e passarão por um processo de reprodução (*crossover* e mutação). Os indivíduos descendentes serão avaliados e um número pré-estipulado T_p da população “pais + filhos” sobreviverão. Em seguida é verificado se existe algum indivíduo da população de sobreviventes (população corrente) que satisfaz o

critério de párada. Caso não exista, os processos de seleção para reprodução, avaliação e seleção de sobreviventes se repetem até que o critério de parada seja atingido. Maiores detalhes das operações deste ciclo são dadas nas subseções a seguir.

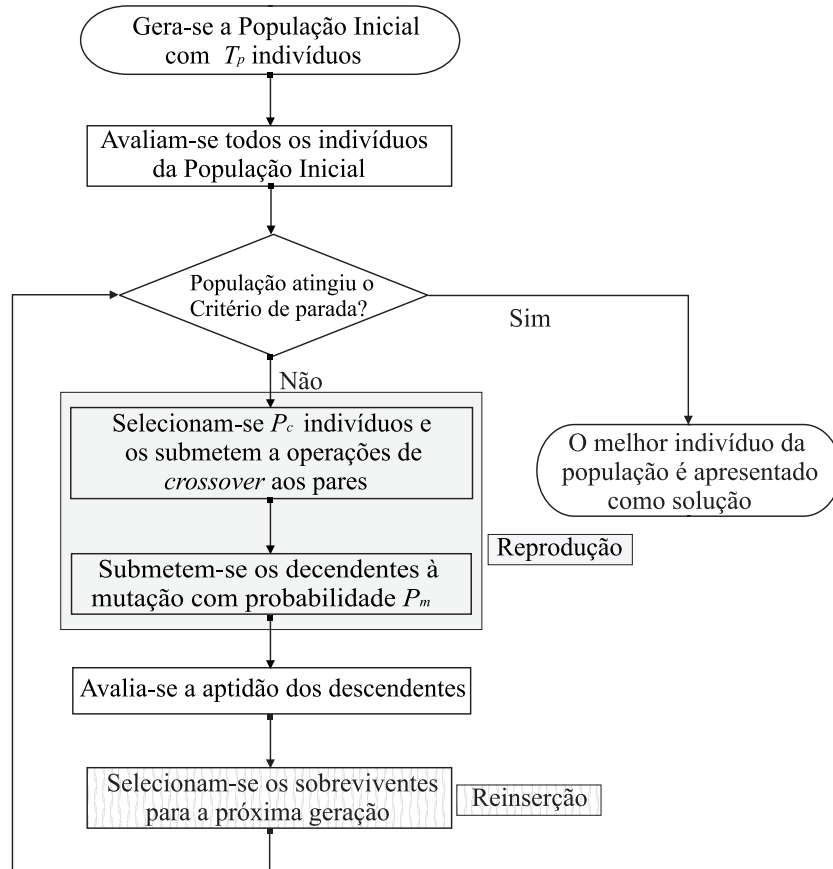


Figura 1.1: Ciclo de execução básico dos AGs. Baseado em [Oliveira 1999]

1.4.1 Representação Cromossômica

Definição 1.1: **Espaço de busca** é o conjunto, espaço ou região que compreende as soluções possíveis ou viáveis do problema.

O primeiro passo para a aplicação de um AG em um problema qualquer é eleger uma maneira de representar cada possível solução (indivíduo) no **espaço de busca** como uma seqüência finita de símbolos de um alfabeto. Os primeiros AGs utilizavam representação binária (alfabeto binário). Atualmente representações de indivíduos (cromossomos) baseadas em caracteres, números reais e inteiros são bastante utilizadas. Segundo [Silva 2002], novas pesquisas têm mostrado que alfabetos de alta-cardinalidade (ex., números reais) são mais efetivos devido ao seu maior poder de expressão. Normalmente aplicações complexas necessitam de alfabetos não-binários. Outra vantagem de representações não-binárias, particularmente as de valores reais, é a fácil definição de operadores específicos do AG.

Contudo, não há uma “receita” para uma boa codificação. Para se ter uma boa performance, os AGs devem ser feitos sob medida para o problema em questão. Assim, a melhor estratégia é usar conhecimentos específicos do domínio do problema sob análise para encontrar uma representação cromossômica apropriada.

1.4.2 População Inicial

Normalmente, uma população inicial de T_p indivíduos é gerada aleatoriamente ou a partir de algum processo heurístico. É importante que a população inicial cubra a maior área possível do espaço de busca. Fazendo uma analogia com a natureza, não ocorre evolução sem variedade, pois é necessário que os indivíduos tenham diferentes graus de aptidão para que possa ocorrer a seleção natural.

1.4.3 Medida de Aptidão

A medida de aptidão indica o quão bem adaptado está cada indivíduo da população ao ambiente. Alternativamente, ela pode ser uma função matemática, um programa de simulação, ou um especialista humano que decide a qualidade de um indivíduo.

Ao longo dos estudos sobre AGs, pesquisas têm mostrado que a especificação de uma medida de aptidão apropriada é crucial para a boa operação de um AG. É essencial que a medida de aptidão seja bastante representativa e diferencie na proporção correta as más soluções das boas. Se houver pouca precisão na avaliação, soluções de grande qualidade podem ser perdidas durante a execução do AG, que gastará mais tempo explorando soluções pouco promissoras, ou pior, pode ser encontrada uma solução de pouca qualidade.

Segundo [Michalewicz 1992], há vários fatores a serem considerados na elaboração de uma medida de aptidão:

- o problema é relacionado a maximização ou minimização?;
- ambientes não determinísticos *versus* ambientes determinísticos;
- a medida de aptidão pode mudar dinamicamente quando o AG é executado;
- a avaliação da aptidão deve consumir “pouco tempo”. Assim, em alguns casos a medida de aptidão é simplificada;
- a medida de aptidão deve alocar valores de aptidão diferentes para indivíduos diferentes, de forma a permitir um melhor trabalho dos operadores de seleção;
- ela deve considerar as restrições do problema;
- ela pode incorporar múltiplos objetivos;

Normalmente, a medida de aptidão é a parte da programação dos AGs que demanda o maior custo computacional, uma vez que ela avalia todos os indivíduos a cada geração. Pensando em diminuir essa carga computacional [Haupt and Haupt 1998] propôs alguns cuidados especiais

como: não gerar indivíduos idênticos na população inicial; manter a população com todos os indivíduos distintos entre si (garantir que a reprodução/evolução não gerará indivíduos idênticos); e criar uma memória para os algoritmos genéticos, para descartar os indivíduos gerados anteriormente. Na prática, quase sempre, somente o primeiro dos critérios citados é levado em conta na elaboração dos AGs.

Devido aos AGs partirem normalmente de soluções ao acaso, no início da busca iterativa, os valores de aptidão para os membros da população são randomicamente distribuídos. Quando a busca evolui, valores particulares para cada gene começam a prevalecer. Assim que a variância dos valores de aptidão decresce a população converge. A análise de convergência pode ser feita por várias medidas: em [Goldberg 1989] é usado o desvio padrão dos valores de aptidão. Na Figura 1.2 foram utilizados os valores de máximo, mínimo e a média das aptidões em cada geração, para a análise de convergência.

No caso ideal a população converge para a solução ótima (veja Figura 1.2). Porém, em algumas situações práticas o esperado não ocorre. Entre os problemas amplamente conhecidos acerca da convergência dos AGs tem-se a convergência prematura e o término lento. Estes problemas, suas possíveis causas e soluções são tratados na seção 1.6.

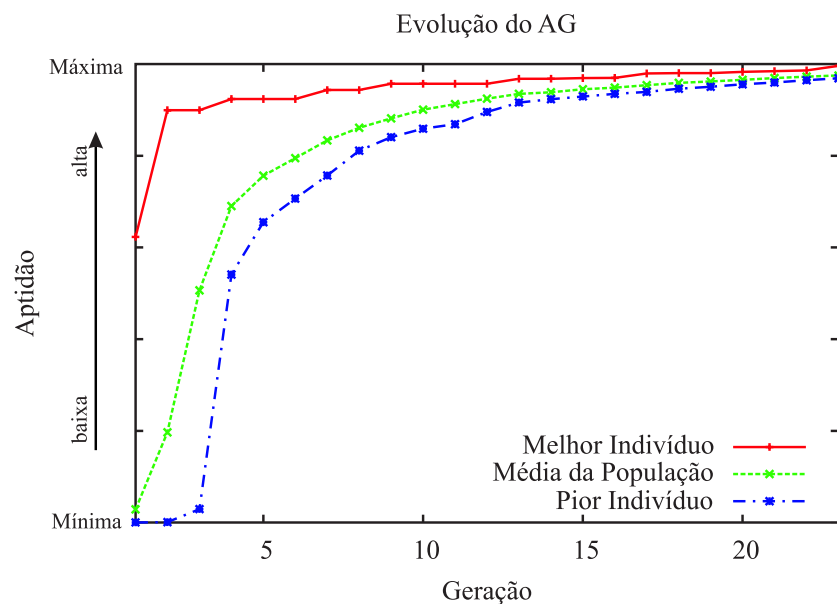


Figura 1.2: Ilustração da convergência de uma população. Pode-se verificar a convergência da população pela análise dos valores máximo, mínimo e a média das aptidões através das gerações.

Pensando nos problemas de convergência, [Cardoso 1998] desaconselha o uso direto da função objetivo¹ como medida de aptidão. Como a pressão seletiva (veja a seção 1.7.2), necessária para a convergência do AG, faz a diversidade da população cair rapidamente, cria-se a necessidade de tomar providências para evitar o problema da convergência prematura (seção 1.6.1). Para tanto [Cardoso 1998] e vários outros pesquisadores sugerem o uso de mapeamentos da função objetivo para se obter medidas de aptidões mais satisfatórias. As técnicas mais

¹Função objetivo refere-se a qualquer função ou métrica que mede a aptidão de um indivíduo

populares usadas para controlar a pressão seletiva através da medida de aptidão são baseadas nas funções de escalamento.

Funções de escalamento têm a finalidade de impedir que os melhores indivíduos tenham uma adequabilidade demasiadamente superior a outros no começo da busca e ampliar a vantagem dos melhores no final, onde a população como um todo está próxima do objetivo. Entretanto, este não é o único meio de se controlar a pressão seletiva. Veremos na próxima subseção um operador que desempenha muito bem este papel: a seleção.

1.4.4 Seleção

A seleção é feita após a medição da aptidão dos indivíduos. Ela desempenha o papel da seleção natural na evolução, selecionando prioritariamente para sobreviver e reproduzir os organismos melhores adaptados ao meio. A seleção é considerada o principal operador na determinação das características de convergência de um AG, sendo vital no controle da pressão seletiva.

A maneira pela qual os indivíduos são selecionados pode variar, dependendo do método de seleção utilizado. Entretanto, é certo que os indivíduos melhores adaptados terão, necessariamente, uma maior probabilidade de sobrevivência e reprodução que os menos adaptados.

Existem diversos operadores de seleção e aqui revisaremos alguns dos mais conhecidos:

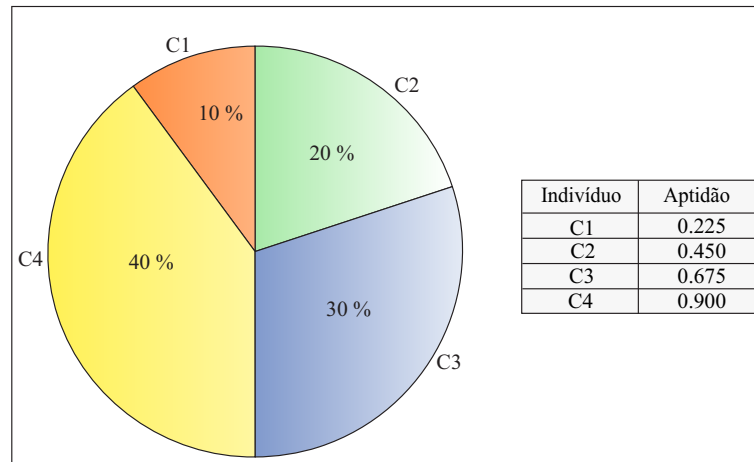


Figura 1.3: Ilustração de uma roleta.

- **Seleção estocástica com reposição** - também conhecido como **método de seleção por roleta**, é o método de seleção padrão dos AGs, proposto originalmente por Holland. A cada indivíduo da população corrente é atribuído uma fatia de uma roleta imaginária, sendo o tamanho desta fatia proporcional à aptidão do indivíduo (ver Figura 1.3). A cada giro desta roleta um indivíduo é selecionado. Se f_j é a aptidão do indivíduo j na população corrente, a probabilidade p_j do indivíduo j ser selecionado é

$$p_j = \frac{f_j}{\sum_{k=1}^{T_p} f_k}, \quad (1.1)$$

onde T_p é o número de indivíduos na população e f_k é a aptidão do k -ésimo indivíduo da população.

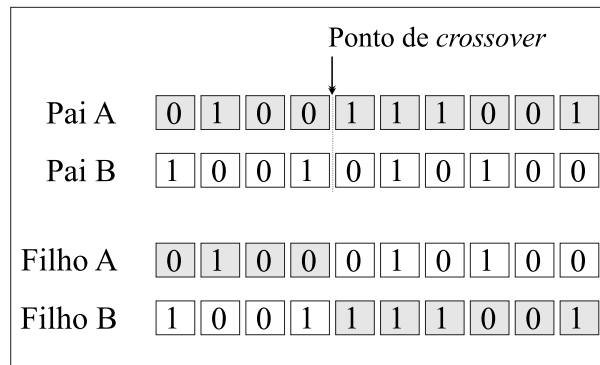
- **Seleção por torneio simples** - A idéia é promover um torneio entre um grupo de N ($N \geq 2$) indivíduos aleatoriamente tomados da população. O indivíduo com o maior valor de aptidão no grupo é selecionado, enquanto que os demais são descartados. Neste método a pressão seletiva pode ser controlada através do tamanho dos grupos do torneio.
- **Seleção por torneio estocástico** - segue a mesma idéia do torneio simples. A única diferença é que os indivíduos dos grupos são selecionados pelo método da roleta, aos invés de serem tomados aleatoriamente.
- **Seleção por truncamento** - um subconjunto dos melhores indivíduos são escolhidos para serem selecionados, com a mesma probabilidade.
- **Seleção por ordenação** - neste modelo, os indivíduos, são ordenados pelas suas aptidões, do pior para o melhor. Depois, a cada indivíduo j é atribuída uma probabilidade de seleção p_j , tomada de uma distribuição aplicada às posições dos indivíduos no *ranking*, com a restrição que $\sum p_j = 1$. As distribuições mais comuns são, respectivamente, a linear, $p_j = a \cdot pos(j) + b$, $a > 0$ e a exponencial, $p_j = a^b \cdot pos(j)^{+c}$, $a > 0, b > 0$, onde $pos(j)$ é a posição do indivíduo j no *ranking*.
- **Seleção elitista** - este modelo é normalmente acoplado a outros métodos de seleção, na tentativa de se aumentar a velocidade de convergência do algoritmo. Este processo simplesmente seleciona os N ($N \geq 1$) melhores indivíduos da população corrente, uma vez que, em um método seleção probabilístico, por azar estes poderiam não ser selecionados.

1.4.5 Crossover

É um processo inspirado na recombinação biológica (cruzamento), ou seja, na troca de material genético entre os pais na geração dos filhos. Os indivíduos que serão submetidos ao *crossover* são escolhidos pelos operadores de seleção, de forma que os indivíduos de maior aptidão tenham maiores chances de serem selecionados. Os pares selecionados para a recombinação (pais) darão origem a pares de descendentes (filhos). A frequência de recombinações é controlada pelo parâmetro p_{rec} , probabilidade de recombinação ou taxa de *crossover*, que diz a percentagem da população que será submetida ao *crossover* a cada geração.

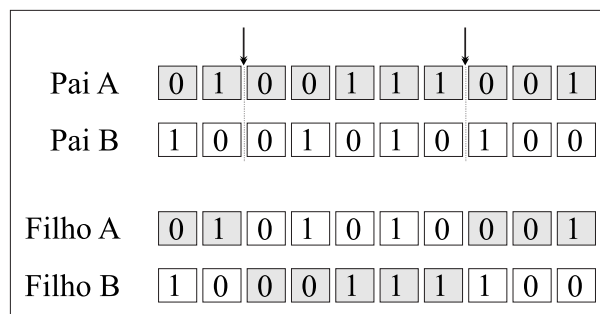
O modo em que a operação de *crossover* é realizada depende do domínio e das restrições do problema em questão. As operações de *crossover* mais usuais têm forte inspiração biológica, sendo os filhos formados a partir de trocas de material genético entre os pais. Nesta categoria temos basicamente três tipos de operações de *crossover*: o simples, o múltiplo e o uniforme.

- **Crossover simples** - um ponto dos cromossomos (biologicamente conhecido como *locus* e computacionalmente conhecido como ponto de *crossover*) é escolhido aleatoriamente. Ambos os cromossomos pais são cortados neste ponto. A primeira parte do Pai A é concatenada à segunda parte do Pai B, formando um dos filhos. O mesmo ocorre com as

Figura 1.4: *Crossover* simples.

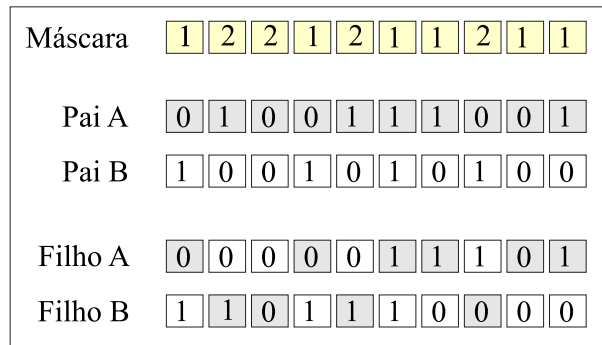
outras partes: a primeira parte do Pai B é ligada à segunda parte do Pai A, formando um segundo filho. Veja Figura 1.4.

- ***Crossover* múltiplo** - Dois ou mais pontos de *crossover* são escolhidos aleatoriamente. Os segmentos de genes, definidos pelos pontos de corte, são alternadamente trocados entre os pais na formação dos filhos. A Figura 1.5) ilustra este processo para dois pontos de corte.

Figura 1.5: *Crossover* múltiplo.

- ***Crossover* uniforme** - é um tipo de *crossover* múltiplo levado ao extremo. Ao invés de sortear pontos de corte, sorteia-se uma máscara do tamanho do cromossomo que indica qual cromossomo pai fornecerá cada gene ao primeiro filho. O segundo filho é gerado pelo complemento da máscara. A Figura 1.6 mostra um exemplo.

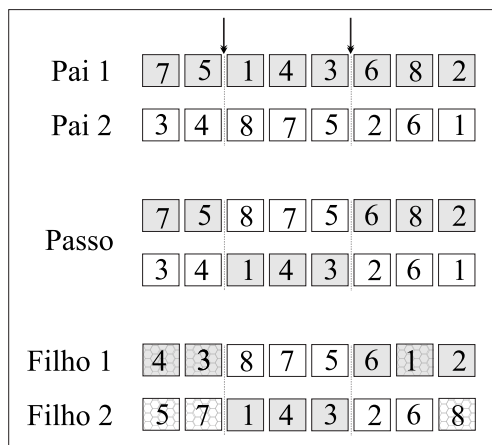
Em alguns domínios de problema a operação de *crossover* não deve gerar genes repetidos. Um exemplo é o problema do Caixeiro Viajante. Neste problema procura-se um trajeto em que o caixeiro passe uma vez por cada cidade, uma após a outra, na ordem em que o percurso total seja minimizado. Naturalmente, uma representação trivial para este problema é uma sequência de genes, cada um representando uma cidade, onde a ordem destes representa a ordem de visitação. Como a solução do problema requer uma única visita a cada cidade, e como os genes representam as cidades, necessita-se de uma operação de *crossover* que não gere genes

Figura 1.6: *Crossover* uniforme.

repetidos. Pensando em problemas como este foram propostos na literatura vários operadores de *crossover*. Aqui, veremos dois dos mais populares: o *PMX* e o cíclico.

- ***Crossover PMX*** - derivado do inglês: *Partially Matched Crossover* e conhecido na comunidade portuguesa como *Crossover* Parcial. Inicialmente sorteiam-se dois pontos de corte aleatórios. Os genes entre estes pontos são integralmente trocados entre os parceiros. Finalizando o processo, os alelos repetidos externos aos pontos de corte são substituídos pelos respectivos genes do outro parceiro, tomados dentro do segmento permutado. A Figura 1.7 ilustra esta operação.

Na biologia, alelos referem-se às variantes de um mesmo gene. Aqui neste contexto, a sentença, *alelos repetidos*, refere-se a genes com informações exatamente idênticas.

Figura 1.7: *Crossover PMX*.Figura 1.8: *Crossover* cíclico.

- ***Crossover cíclico*** - a partir de uma única posição, os genes são trocados entre os parceiros até formar um ciclo. Os demais genes fora do ciclo não são trocados. A Figura 1.8 exemplifica esta operação. Começando da primeira posição dos cromossomos, trocamos: 7 e 3, 3 e 5, 5 e 4, 4 e 7. Neste instante o processo se completa pois um ciclo foi formado: 7, 3, 5, 4, e 7.

1.4.6 Mutação

A mutação é uma operação que simplesmente modifica aleatoriamente alguma característica genética do indivíduo sobre o qual a mesma é aplicada (ver Figura 1.9). Esta é importante, pois permite criar novas características que não existiam ou apareciam em pequena quantidade na população em análise, introduzindo diversidade genética da população [Goldberg 1989]. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca não será nula. O operador de mutação é aplicado aos indivíduos através de uma taxa de mutação geralmente pequena. A probabilidade de mutação pode ser por indivíduo (cromossomo) ou por genes.

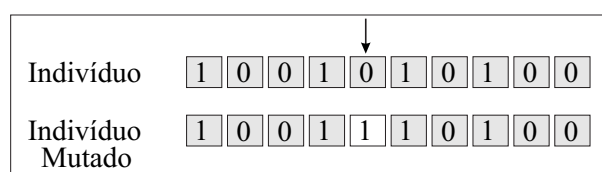


Figura 1.9: Mutação simples.

A operação de mutação, assim como a de *crossover* deve ser feita de modo a não violar as restrições do problema. Muitos procedimentos já foram propostos: substituição de um gene por um outro gerado aleatoriamente, perturbação de um gene, permutação de genes, etc. Para maior detalhes veja [Goldberg 1989; Haupt and Haupt 1998; Holland 1975; Michalewicz 1992; Mitchell 1997; Tanomaru 1995].

1.4.7 Reinservação

Após o processo de reprodução (*crossover* e mutação) e avaliadas as aptidões dos filhos, faz-se necessário mecanismos de seleção que elegerão os sobreviventes para a próxima geração. Os principais métodos de reinservação são :

- **Reinservação pura** - substitui toda a população pelos filhos. Esta estratégia é normalmente acompanhada de elitismo.
- **Elitismo** - uma parte da população (os melhores pais) é mantida para a próxima geração. Normalmente este procedimento é acompanhado por reinservação uniforme ou reinservação pura.
- **Reinservação uniforme** - selecionam-se a partir de qualquer um dos métodos de seleção tradicionais (seção 1.4.4) T_p indivíduos da população total (Pais + Filhos).
- **Baseada na aptidão** - a população total é ordenada e os T_p melhores indivíduos são selecionados.

1.4.8 Condições de Parada

Em se tratando de problemas de otimização, o ideal é que o AG pare assim que a solução ótima é descoberta [Oliveira 1999]. Entretanto, em um problema prático, muitas vezes não se pode afirmar que o AG encontra a solução ótima em tempo viável. Como consequência, utiliza-se vários outros critérios de parada como:

- Excedido o número máximo de gerações pré-estabelecido.
- Excedido o tempo máximo de processamento previamente estabelecido.
- Encontrado um indivíduo com a aptidão maior ou igual a um limiar pré-definido.
- Estagnação da população ou do melhor indivíduo após um determinado número de gerações.

1.5 Parâmetros de Controle

Os AGs têm seu funcionamento baseado em uma série de parâmetros: o tamanho da população T_p , a taxa de *crossover* P_c , a probabilidade de mutação P_m e até mesmo, o critério de parada. Estes parâmetros têm grande influência no comportamento de um AG, podendo levar estes a bons resultados ou ao fracasso. A intuição normalmente seguida na escolha destes parâmetros é a seguinte:

- Uma população muito pequena, implica em pouca cobertura do espaço de busca e conseqüentemente, maiores probabilidades de convergência prematura (subseção 1.6.1). Já uma população muito grande possibilita uma ampla cobertura do espaço de busca, prevenindo a convergência prematura, porém exige-se uma elevada carga computacional.
- Quanto maior a probabilidade de *crossover*, mais rapidamente novas estruturas serão introduzidas na população. Mas se esta for muito alta, estruturas com boas aptidões poderão ser destruídas mais rapidamente que a capacidade da seleção em mantê-las. Assim, normalmente são utilizadas operações de elitismo para garantir que o(s) melhor(es) indivíduo(s) não serão destruídos pelas operações de *crossover*.
- Mutações previnem que a busca fique estagnada em regiões pouco promissoras do espaço de busca (máximos locais), possibilitando que qualquer ponto do espaço de busca tenha uma probabilidade não nula de ser atingido. Entretanto, uma taxa muito alta de mutação torna a busca essencialmente aleatória.

A escolha dos parâmetros de controle dos AGs depende do problema que está sendo tratado. Muito autores aconselham que estes devam ser determinados com base em experimentações. Outros, motivados pela análise de problemas de convergência, acreditam que a variação dinâmica destes parâmetros faz com que os AGs tenham um melhor desempenho. Na literatura existe vários estudos relativos a especificação de parâmetros de controle [Bäck 1992; Goldberg 1989; Grefenstette 1986; Michalewicz and Fogel 2000; Smith and Fogarty 1996].

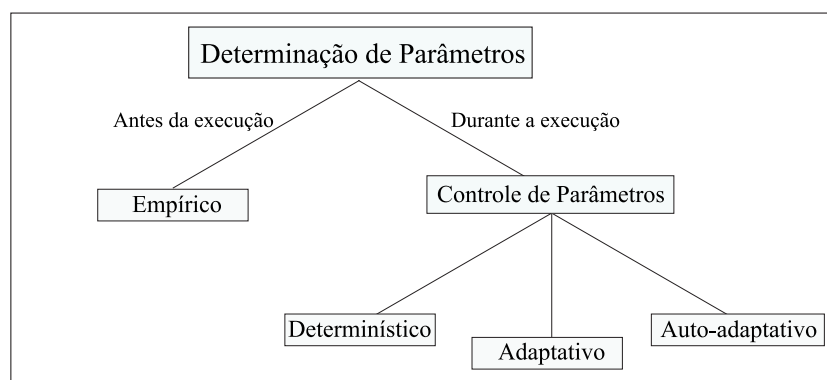


Figura 1.10: Classificação das técnicas de ajuste de parâmetros.

Segundo [Michalewicz and Fogel 2000] as técnicas de determinação dos parâmetros de controle podem ser classificadas conforme a Figura 1.10.

De acordo com essa classificação, antes da execução o ajuste é feito de modo empírico. Durante a execução o ajuste pode ser feito do modo determinístico, onde os valores dos parâmetros são alterados de acordo com alguma regra predeterminada, como por exemplo quando um certo número de iterações é atingido; pode ser feito também de modo adaptativo, isto é, de acordo com informações obtidas como retorno do próprio processo evolutivo; pode ainda ser feito de modo auto-adaptativo, onde as informações sobre os parâmetros são codificadas dentro dos indivíduos e também reproduzem e evoluem.

1.6 Problemas Práticos dos AGs

Dentre os problemas normalmente encontrados pelos pesquisadores de AGs, têm-se três amplamente conhecidos: a convergência prematura, a convergência lenta e o tempo de processamento elevado. Aqui, iremos nos concentrar nas descrições, causas e possíveis soluções destes.

1.6.1 Convergência Prematura

A convergência prematura, também conhecida como decepção, ocorre quando genes de alguns poucos indivíduos, com alto valor relativo de aptidão, rapidamente dominam a população, causando a convergência para um máximo local². Uma vez que a população tenha convergido, a habilidade do AG de continuar pesquisando por melhores soluções fica bastante restrita.

Este tipo de convergência normalmente ocorre devido à população reduzida e/ou à má distribuição da população em torno do ponto sub-ótimo, ou seja, indivíduos próximos de ótimos locais possuem valores de aptidão bastante superiores aos demais indivíduos da população. Com isso, o processo de seleção fará com que um destes indivíduos tenha grandes chances de dominar a população nas próximas gerações [Michalewicz 1992].

²Máximo local ou ótimo local é uma solução ótima em uma dada vizinhança de busca; já máximo global ou ótimo global é uma solução ótima considerando todo o espaço de busca do problema

1.6.2 Convergência Lenta

É o problema oposto à convergência prematura. Depois de muitas gerações, a população quase convergiu, porém, o máximo global não foi encontrado. A aptidão média é alta, e a diferença entre o melhor e o pior indivíduo é pequena. Contudo, não há variância suficiente nos valores de aptidão para localizar um máximo global.

1.6.3 Tempo de Processamento Elevado

Grande parte das aplicações de AGs absorvem um elevado tempo de processamento. Muitas vezes, o problema que se deseja otimizar é complexo, tornando demorada a avaliação de todos os indivíduos da população. Também pode ocorrer do espaço de busca ser muito grande, exigindo grandes populações, e conseqüentemente o AG demora a convergir.

Uma excelente solução para este problema é a utilização de processamento paralelo na execução dos AGs. Devido a natureza dos AGs ser intrinsecamente paralela, esta abordagem é bastante promissora. Por exemplo, como a medição da aptidão de um indivíduo é independente da medição de aptidão de qualquer outro, podemos pensar em uma rede de processadores paralelos, onde cada processador é responsável pela medição da aptidão de um indivíduo na população corrente. As operações de *crossover* e de mutação são também independentes e podem ser feitas em paralelo. Com o intuito de explorar estas propriedades, vários modelos para implementação de AGs paralelos têm sido propostos na literatura [Cantú-Paz 2001].

1.7 Diversidade Populacional e a Pressão Seletiva

Nesta seção são apresentados dois conceitos essenciais para o entendimento da funcionalidade dos AGs: a diversidade populacional e a pressão seletiva.

1.7.1 Diversidade Populacional

A diversidade populacional está relacionada com a variabilidade genética dos indivíduos da população: quanto maior a diversidade, maior o espalhamento das possíveis soluções (indivíduos) no espaço de busca. Normalmente o espalhamento é máximo no início do processo de busca, principalmente quando a população inicial é gerada aleatoriamente, e mínimo quando o AG converge em torno de uma solução.

A perda prematura de diversidade populacional implica na convergência prematura dos AGs. Uma maneira de aumentar a diversidade populacional é aumentar a taxa de mutação. Isso faz com que a taxa em que os genes dos indivíduos são alterados aumente, promovendo um afastamento um dos outros (espalhamento) na proporção do aumento da taxa de mutação. Outra maneira seria aumentar o tamanho da população. Entretanto, tais medidas podem afetar negativamente outros aspectos dos AGs, e devem ser melhor analisadas antes de serem tomadas.

1.7.2 Pressão Seletiva

A pressão seletiva é o grau em que os melhores indivíduos são favorecidos. Ela determina a velocidade e a “direção” que o AG vai tomar no espaço de busca. Sem este fator, o AG se comportaria como um algoritmo de busca aleatório, “sem direção”, “nem sentido”.

A pressão seletiva depende da medida de aptidão (função de adequação) e do método de seleção adotado. Ela modula o grau de privilégio de um indivíduo para sobreviver e reproduzir em detrimento dos demais.

Quando a aptidão de alguns indivíduos da população difere muito em relação a outros diz-se que a pressão seletiva é grande. Quando as aptidões dos indivíduos são próximas diz-se que a pressão seletiva é pequena.

Uma pressão seletiva nula torna os AGs equivalentes a um método de busca aleatório pois, uma vez que todos os indivíduos têm a mesma probabilidade de sobreviver ou de reproduzir, nenhum indivíduo possui vantagem sobre os demais, ficando o AG sem direção. Quando parte dos indivíduos tem maior aptidão que os demais, estes provavelmente guiarão as possíveis soluções pelo espaço de busca.

No entanto, a escolha de um número reduzido de indivíduos para sobreviver e reproduzir tende a causar perda de diversidade populacional. Por isso costuma-se dizer que a pressão seletiva trabalha em sentido oposto à diversidade populacional. No entanto são estas duas forças, de certo modo antagônicas, que “conduzem o AG”.

Observa-se que o aumento da pressão seletiva acelera a convergência do AG para algum ponto de ótimo (global ou local): se os melhores indivíduos estiverem próximos de um máximo global (ou de um resultado satisfatório) então a pressão seletiva alta terá um caráter benéfico, pois irá diminuir o tempo de busca, fazendo com que os indivíduos migrem rapidamente em direção a este ponto de máximo global.

Por outro lado, se os melhores indivíduos não estiverem em torno de um máximo global e sim próximos de um máximo local não-satisfatório, um aumento da pressão seletiva pode ser prejudicial, pois poderá ocorrer dos melhores indivíduos migrarem prematuramente para o “ponto errado” (convergência prematura). Essa migração causará uma perda da diversidade populacional, o que dificultará o AG em sua tarefa de encontrar um ponto ótimo global no espaço de busca.

Um modo de controlar a pressão seletiva pode ser implementado através do método de seleção ou pelo método de avaliação dos indivíduos da população. O método de escalamento da função de adequação tem exatamente este propósito. A escolha do método de seleção deve ser feita com base no julgamento do projetista acerca da pressão seletiva que o AG requer. Diferentes métodos de seleção exercem diferentes pressões seletivas.

1.8 Considerações Finais

A otimização é uma ferramenta imprescindível na resolução de problemas complexos. Dentre os métodos de otimização, os estocásticos se destacam por sua característica de buscar a solução ótima sem fazer o uso de todo o domínio de soluções candidatas. Isso é possível por causa do uso de técnicas probabilísticas para guiar a amostragem em direção às regiões mais promissoras do espaço de busca.

Dentro deste contexto, neste capítulo foi apresentado o conceito de Algoritmos Genéticos, uma técnica de busca estocástica, direcionada pela qualidade de soluções, melhoradas a cada iteração por operações inspiradas em operadores genéticos e por operadores seletivos baseados na abstração do conceito de seleção natural.

Esta técnica tem atraído várias pesquisas e propiciado aplicações de sucesso em recuperação de informação, a qual será abordada a seguir no contexto de imagens.

Recuperação de Imagens

A recuperação de imagens é um dos principais focos de pesquisa da recuperação de informação moderna. A recuperação de informação é a ciência da busca por informações em coleções de documentos, conforme a demanda do usuário. Os documentos podem ser textos, sons, imagens, vídeos, páginas *Web*, etc. Uma ação de recuperar informações parte de uma declaração formal da necessidade de informação pelo usuário (consulta). Esta pode ser feita de várias formas, dependendo do projeto e propósito do sistema de recuperação: por palavras-chave, navegação interativa, um exemplo da informação desejada (uma imagem, um som ou uma amostra de vídeo), etc. Os sistemas de recuperação de informação (SRIs) então comparam a representação da consulta do usuário com as representações dos documentos armazenados em coleções, e desta comparação têm-se os valores de similaridades utilizados para prover informações ao usuário, normalmente uma lista de documentos ordenados por ordem de similaridade à requisição do usuário – *ranking*.

Atualmente os SRIs são de extrema importância para o acesso eficiente à enorme quantidade de informações que povoa os bancos de dados. No contexto dos SRIs, neste capítulo é apresentada a recuperação de imagens. Inicialmente, com o intuito de motivar o leitor são apresentadas diversas possibilidades de aplicações relacionadas ao bom aproveitamento das imensas bases de dados atuais de imagens. Depois de contextualizar a recuperação de imagens por conteúdo (*Content-Based Image Retrieval* – *CBIR*) é apresentado uma revisão desta abordando seus procedimentos e aspectos principais. Finalmente, são apresentadas algumas medidas de performance usadas na avaliação dos sistemas *CBIR*.

2.1 O Uso de Imagens

O uso de imagens é pré-histórico. No período da pré-história muitos povos desenhavam cenas de seu cotidiano em paredes de cavernas, como uma forma de registrar seus feitos. Um fato notável é que hoje, milhares de anos depois, somos capazes de inferir fatos da vida de pessoas que viveram há tanto tempo, através da observação de ilustrações. Todavia, sabe-se que é possível que “pessoas diferentes” infiram coisas diferentes ao observar a mesma cena. Isto reflete propriedades importantes das imagens, como sua capacidade de quebrar barreiras temporais e seus diversos significados. Além disso, imagens transmitem informações, histórias

e humor, auxiliam na divulgação de cultura, na educação, etc. À medida que a humanidade evoluiu, muitas coisas aconteceram, fatos históricos, inventos, descobertas, surgindo a necessidade e possibilidade destes serem registrados. Hoje, fotos, gravuras, produções de cinema e de canais de TV, imagens de satélites, de raio-X, ultrasonografias, tomografias, anúncios comerciais, animações gráficas, imagens de museus, de radares são exemplos de registros de imagens de grande importância no nosso dia-a-dia. Estes registros são úteis a várias aplicações como educação, entretenimento, cultura, comércio, estratégias militares, na medicina, diagnóstico médico, controle de queimadas, de desmatamentos, previsão meteorológica, astronomia, prevenção ao crime, etc. No entanto, aplicações como estas só são possíveis graças ao desenvolvimento de técnicas computacionais que permitem a localização, busca e extração de informações úteis de forma automática de imensas coleções de imagens.

2.2 A Recuperação de Imagens

Nos últimos anos o número de coleções de imagens digitais tem crescido gigantesco. Bilhões de *bytes* de imagens são salvos diariamente em banco de dados para propósitos civis e militares e esta taxa é sempre crescente. Imagens são produzidas por satélites (civis e de defesa), inspeções militares e operações de vigilância, sistemas biométricos (impressões digitais, faces, etc), experimentos científicos, imageamento biomédico, sistemas de entretenimento e informação, entre outros. A evolução dos computadores, dos dispositivos de captura e visualização de imagens, e o surgimento da *Internet* são outros fatores que contribuíram para o aumento das coleções de imagens. Através da *internet* tornou-se possível o acesso e disponibilização de imagens em todas as partes do mundo, seja qual for sua finalidade.

Os acontecimentos citados anteriormente fizeram com que as técnicas automáticas de busca e extração de informações de imagens se tornassem uma necessidade. Assim originou quatro grandes áreas de pesquisa: recuperação de imagens, reconhecimento de imagens, análise de imagens e visão computacional. O foco deste capítulo é a recuperação de imagens. Dado que na literatura não é bem definido os limites desta área, aqui contextualizamos esta como a tarefa de buscar imagens relevantes em bancos de dados.

Atualmente a recuperação de imagens é aplicada em vários setores de nossa sociedade, como na educação, cultura, entretenimento, comércio, etc. especialmente na *Web* através dos sistemas de busca e de classificação. Outro fator que motiva as pesquisas em recuperação de imagens é a sua ligação com as áreas de visão computacional, reconhecimento e análise de imagens. Estas são áreas de pesquisa recentes que prometem aplicações fantásticas como: diagnóstico médico (reconhecimento de tumores, de lesões, etc.), prevenção ao crime (reconhecimento de faces, de íris, impressões digitais, etc), aplicações militares (reconhecimento de alvos), observação espacial (localização de áreas de desmatamentos, de queimadas, de culturas não permitidas), robótica (controle de robôs, navegação via satélite), etc.

Conforme a contextualização anteriormente estabelecida, a tarefa de um sistema de recuperação de imagem é operar sobre uma coleção de imagens e apresentar "itens relevantes" conforme a requisição do usuário (consulta). A consulta do usuário pode ser expressa de várias formas, dependendo do modo de operação do sistema de recuperação (ver Seção 2.9.1). A medida de similaridade entre a consulta e os documentos da coleção, dependerá da forma

de representação das características das imagens. As principais medidas de similaridade são baseadas em medidas de distância entre as representações numéricas das características de imagem ou em casamentos de palavras-chave, de *strings* e de grafos. Todas estas questões serão esclarecidas no decorrer deste Capítulo.

Primeiramente, o modo como o usuário expressa sua consulta e a maneira de computar a similaridade dos documentos da coleção em relação à consulta depende dentre outras coisas, da abordagem utilizada pelo sistema de recuperação de imagens: baseada em anotações textuais ou baseada no conteúdo visual (recuperação de imagens por conteúdo).

- **Recuperação baseada em anotações textuais** – utiliza-se de anotações manuais como forma de descrição do conteúdo das imagens. As consultas são expressas por meio de palavras-chave, às vezes combinadas por operadores booleanos, e efetuadas por meio de técnicas de gerenciamento de banco de dados. Tal abordagem é limitada por dois problemas, considerados insuperáveis por muitos pesquisadores. Primeiramente, anotações de imagens para grandes bancos de dados é praticamente impossível. Em segundo lugar, anotações de imagens são subjetivas – uma imagem pode ser interpretada de diferentes maneiras por diferentes pessoas e até mesmo pela mesma pessoa em diferentes situações. Revisões dos trabalhos de recuperação de imagens baseados em anotações textuais podem ser encontrados em [Chang and Hsu 1992] e [H. Tamura 1984]. Apesar do seu improvável sucesso para muitas aplicações, esta abordagem permanece aplicável para imagens de significado semântico especial como coleções de fotos de museus e de pinturas famosas.
- **Recuperação baseada no conteúdo visual** – visa recuperar imagens relevantes de bancos de dados com base em características automaticamente extraídas das imagens. Características extraídas das imagens podem ser de baixo nível ou de alto nível. Características de baixo nível trata-se da representações matemáticas das características visuais das imagens como cor, forma e textura. Características de alto nível refere-se à semântica, como objetos (suas propriedades e seus significados), interpretação de cenas, ações, sentimentos, etc [Liu et al. 2007]. Imagens são comparadas com base no grau de similaridade entre a requisição do usuário e as características extraídas dos itens do banco de dados. Nesta abordagem há várias possibilidades para a expressão da consulta do usuário (ver seção 2.9.1), sendo a mais comum, a consulta por exemplo visual.

Também, pode ser verificado a presença de uma outra categoria de sistemas de recuperação de imagens. Um exemplo desta é o serviço provido pela máquina de busca www.google.com. Nesta, imagens extraídas de documentos (geralmente páginas *html*) são indexadas com base nas informações textuais ou no *label* que as acompanham. Posteriormente utiliza-se técnicas tradicionais de recuperação de informação textual, para casar a consulta do usuário, provida por palavras-chave, com os índices das imagens. Entre os problemas deste tipo de sistema podemos destacar sua aplicabilidade (reduzida ao conjunto das imagens presentes nos documentos da *Web*) e sua fraca eficiência, proporcionada principalmente pelo fato de que na maioria das vezes o texto próximo a uma figura não a descreve fielmente.

O foco deste Capítulo são as técnicas de recuperação de imagens por conteúdo.

2.3 Recuperação de Imagens por Conteúdo

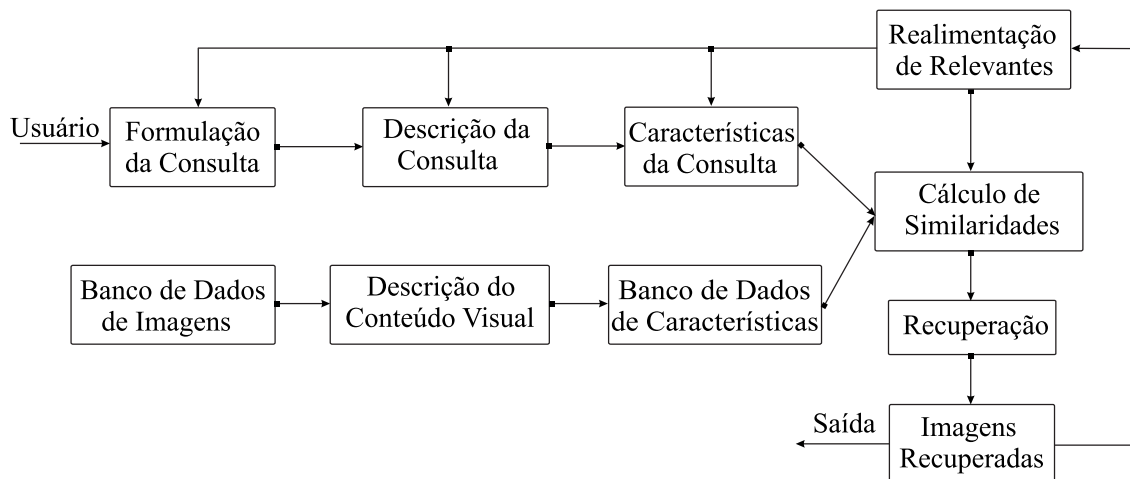


Figura 2.1: Fluxograma de um sistema *CBIR* típico [Long et al. 2003].

A dificuldade e o custo de produzir anotações textuais ricas e confiáveis para grandes bancos de dados de imagens, como também a subjetividade associada a estas anotações, explicam o fato da recuperação de imagens baseada diretamente no conteúdo visual, recuperação de imagens por conteúdo (*CBIR*), ser de grande interesse nos dias atuais [Crucianu et al. 2004].

Na recuperação de imagens por conteúdo o objetivo principal é encontrar “imagens relevantes” conforme a necessidade do usuário, através de características visuais automaticamente extraídas das imagens. Relevância para um usuário é normalmente associada à sua interpretação semântica das imagens, ou seja, uma imagem é relevante se esta satisfaz o critério semântico do usuário.

Como veremos neste trabalho, a interpretação semântica é um fator essencial para a recuperação de imagens e sua ligação às características automaticamente extraídas das imagens constitui um dos principais problemas desta área.

A Figura 2.1 mostra o fluxograma de um sistema padrão de recuperação de imagens por conteúdo. Primeiramente considera-se um banco de dados de imagens, também conhecido como coleção de referência¹. As imagens da coleção de referência são caracterizadas sintaticamente ou semanticamente, formando um banco de dados de características. Anterior a esta etapa, alguns sistemas *CBIR* incluem operações de pré-processamento (melhoramento da qualidade das imagens) e segmentação de imagens. Posterior à extração de características, sistemas *CBIR* que manipulam grandes coleções de imagens costumam incluir um processo de indexação que visa criar uma estruturação dos dados de forma a maximizar a velocidade do processo de busca por reduzir o número de comparações feitas pelo módulo Cálculo de Similaridades. Normalmente, o mesmo processo de descrição das características feito para o banco de dados é feito para a consulta do usuário, já que na maior parte dos sistemas *CBIR* as consultas são feitas por

¹Utiliza-se o termo coleção de referência devido a resposta à requisição do usuário levar em conta exclusivamente as imagens desta coleção, ou seja, a coleção de referência delimita o universo de busca de um sistema de recuperação.

imagens de exemplo. Dado, a descrição da consulta do usuário e as da coleção de referência, compara-se a imagem de consulta com as imagens da coleção através de medidas de similaridade (ou algoritmos de casamento). Com base nos valores de similaridade (ou grau de casamento) um *ranking* é formado e mostrado ao usuário. Muitos dos sistemas *CBIR* atuais fornecem aos usuários a possibilidade de refinar suas buscas através de realimentação de relevância.

Neste trabalho é dada uma atenção especial aos mecanismos de realimentação de relevância.

2.4 Caracterização

A recuperação de imagens por conteúdo baseia-se na descrição matemática de aspectos da visão humana tais como cor, forma e textura – características de baixo nível, ou na descrição de aspectos semânticos como objetos, ações, sentimentos, etc. – características de alto nível. A maior parte das pesquisas atuais em *CBIR* são voltadas para a exploração de características de baixo nível. A extração de características de alto nível tem se mostrado uma tarefa bastante complexa e com poucos resultados práticos [Rui et al. 1997]. No trabalho de [Liu et al. 2007] é apresentada uma revisão do estado arte da exploração de características de alto nível (semânticas). À descrição matemática de características de baixo nível dá-se o nome de vetores de características. Um vetor de característica tem por finalidade representar os aspectos visuais significativos de uma imagem.

Um vetor de características pode ser global ou local. Um vetor global usa as características visuais da imagem inteira, já um vetor local usa características de regiões ou objetos para descrever o conteúdo da imagem. Para se obter descrições (vetores) locais, uma imagem deve ser dividida em partes. O caminho mais simples de dividir uma imagem é usar partições seguindo algum critério pré-definido, como por exemplo, blocos de mesmo tamanho. Um caminho mais complexo é empregar técnicas de segmentação² para se obter objetos mais significantes semanticamente (como pessoas, bola de futebol, carro, casa, etc.).

Um bom vetor de característica pode ser caracterizado por várias propriedades. A precisão de recuperação, a separabilidade, a compactabilidade, aplicações, complexidade computacional e robustez compõem as propriedades de maior interesse.

A seguir veremos os principais descritores utilizados para representar as características visuais de baixo nível de imagens. Normalmente, as características de baixo nível que melhor satisfazem ao critério de separabilidade de imagens, tanto por humano quanto por máquina, são as de cor, forma e textura.

2.5 Cor

Cor é um dos principais aspectos da visão humana. O olho humano consegue discernir milhares de variações e intensidades de cores [Gonzalez and Woods 2002]. Características de cor são relativamente fáceis de representar e de grande importância na diferenciação de imagens por máquinas. Atualmente, cor é a característica visual mais usada em sistemas de recuperação

²Atualmente, segmentação automática de objetos para imagens de domínios gerais é uma tarefa de improvável sucesso.

de imagens por conteúdo. Grande parte dos sistemas *CBIR* utilizam informações de cor para comparar imagens, geralmente associada a outras características como forma e/ou textura.

A primeira decisão para se definir uma caracterização de cor é escolher o espaço de cor. Um espaço de cor é essencialmente uma especificação de um sistema de coordenadas tridimensionais e um subespaço dentro desse sistema, onde cada cor é representada por um único ponto. Os espaços de cores mais comuns na recuperação de imagens são o *Munsell*, *CIE L^*a^*b* , *CIE L^*u^*v* , *HSV* e *HSI* [Schettini et al. 2001]. Na literatura não há argumentos que permitam afirmar qual é o melhor espaço de cor, sendo a escolha de um deles, dependente da aplicação e do descritor. Contudo, na descrição das cores de uma imagem, umas das características desejáveis de um espaço de cor é a uniformidade – a proximidade entre duas cores deve ser diretamente relacionada a similaridade psicovisual entre elas.

A seguir são apresentados alguns descritores de cor comumente usados para descrever as informações de cores de imagens.

2.5.1 Momentos de Cor

Momentos de cor [Stricker and Orengo 1995] caracterizam as imagens em termos da distribuição das cores. Estes têm sido utilizados com sucesso em muitos sistemas *CBIR*, sendo o mais famoso, o *QBIC* [Flickner et al. 1995] da *IBM*.

Momentos de cor são normalmente dados por três medidas estatísticas: média, desvio padrão e inclinação, computadas sobre cada canal de cor de uma imagem. Sendo N o número de *pixels* de uma imagem e p_{ij} o valor do j -ésimo *pixel* no i -ésimo canal de cor, os três primeiros momentos: média, desvio padrão e inclinação são computados através das expressões 4.1, 4.2 e 4.3, respectivamente.

$$E_i = \frac{1}{N} \sum_{j=1}^N p_{ij} \quad (2.1)$$

$$\sigma_i = \sqrt{\left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - E_i)^2 \right)} \quad (2.2)$$

$$s_i = \sqrt[3]{\left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - E_i)^3 \right)} \quad (2.3)$$

Os momentos de cor são normalmente derivados dos espaços de cor *CIE L^*u^*v* , *CIE L^*a^*b* e *HSV* ou *HSI*. Devido a sua alta compactabilidade e por não considerar informações espaciais, momentos de cor têm poder de discriminação reduzido. Porém sua performance pode ser significativamente melhorada aplicando-o a regiões de imagem devido à descrição de características locais.

2.5.2 Histograma de Cores

Um histograma de cores [Swain and Ballard 1991] descreve a frequência global das cores de uma imagem. Em benefício à compactabilidade e performance, normalmente faz-se uma quantização do número de cores consideradas. Um dos modos de fazer a quantização de cores é usar métodos de clusterização para determinar as “ k melhores cores” (as mais frequentes, ou as mais discriminantes) no domínio de uma dada coleção de imagens. Cada destas melhores cores irão ser tomadas como uma das cores do histograma.

Entre os problemas dos histogramas de cores podem-se destacar sua alta dimensionalidade (necessária para sua eficiência) e o fato destes não considerarem a localização espacial das cores. Assim, imagens muito diferentes podem ter representações semelhantes. Novamente pode ser adotada a divisão das imagens em regiões, computando uma representação para cada região. Porém, tal procedimento aumenta ainda mais a dimensionalidade e carga computacional necessária para comparar os descritores.

2.5.3 Vetor de Coerência de Cores

O primeiro passo na tentativa de incorporar informações espaciais em um descritor de cor foi dado por [Pass and Zabith 1996] que propôs o método denominado vetor de coerência de cores (*CCV*). Cada *pixel* de uma imagem i é classificado como coerente ou incoerente. Um *pixel* será coerente se ele fizer parte de uma região de coloração uniforme (de tamanho pré-definido), caso contrário ele será incoerente. Para cada cor c_k o número de *pixels* coerentes α_{c_k} , e o número de *pixels* incoerentes β_{c_k} são computados. Cada componente do *CCV* é um par $(\alpha_{c_k}, \beta_{c_k})$, chamado par de coerência. O vetor de coerência é então definido como

$$CCV(i) = \left((\alpha_{c_1}, \beta_{c_1}), (\alpha_{c_2}, \beta_{c_2}), \dots, (\alpha_{c_n}, \beta_{c_n}) \right) \quad (2.4)$$

Note que $(\alpha_{c_1} + \beta_{c_1}, \alpha_{c_2} + \beta_{c_2}, \dots, \alpha_{c_n} + \beta_{c_n})$ é o histograma de cor da imagem.

Segundo [Long et al. 2003], os vetores de coerência de cores produzem melhores resultados de recuperação que os histogramas de cores, especialmente para imagens com muitas regiões de cor uniforme ou muita textura. Também conforme [Long et al. 2003], histogramas de cores e vetores de coerência de cores produzem melhores resultados quando aplicados ao espaço de cor *HSV*.

2.5.4 Correlograma de Cores

O correlograma de cores [Huang et al. 1997] é um histograma tridimensional que caracteriza a distribuição das cores e a correlação espacial entre pares de cores. A primeira e a segunda dimensão do histograma representam as cores de qualquer par de *pixels* e a terceira dimensão, a distância espacial entre eles. Um correlograma pode ser visto como uma tabela indexada por pares de cores, onde a k -ésima entrada para (i, j) especifica a probabilidade de encontrar um *pixel* de cor j a uma distância k de um *pixel* de cor i na imagem. Seja H o conjunto de *pixels* de uma imagem e $H_{c(j)}$ o conjunto de *pixels* cuja cor é $c(j)$. Então seu correlograma é definido como:

$$\gamma_{i,j}^k = \Pr_{p_1 \in H_{c(i)}, p_2 \in H} \left[p_2 \in H_{c(j)}, |p_1 - p_2| = k \right] \quad (2.5)$$

onde $i, j \in \{1, 2, \dots, N\}$, $k \in \{1, 2, \dots, d\}$, e $|p_1 - p_2|$ é a distância entre os *pixels* p_1 e p_2 , e $\Pr$ é a função probabilidade. Se considerarmos todas as possíveis combinações de pares de cores o tamanho do correlograma será muito grande ($O(N^2d)$), assim, uma versão simplificada chamada autocorrelograma é freqüentemente usada. O autocorrelograma de cores captura somente a correlação espacial entre cores idênticas e assim reduz a dimensão para $O(Nd)$.

Comparado ao histograma de cores e ao vetor de coerência de cores, o autocorrelograma de cores produz melhores resultados, mas também é computacionalmente o mais caro devido à sua alta dimensionalidade [Long et al. 2003].

2.5.5 Características Invariantes de Cores

Cor não somente reflete o material da superfície, mas também varia consideravelmente com mudanças de iluminação, orientação da superfície, e visão geométrica das câmeras. Em alguns casos é interessante levar em conta esta variabilidade. Representações invariantes de cores foram introduzidas na recuperação de imagens por conteúdo recentemente. Em [Gevers and Smeulders 2000], um conjunto características invariantes de cores para recuperação de objetos foi derivada das propriedades de reflexão de objetos. Em [Gevers and Smeulders 1999] é proposto um descritor de cor invariante à geometria das superfícies.

Estas características de cor invariantes, quando aplicadas na recuperação de imagens, podem produzir representações do conteúdo das imagens independentes da iluminação e geometria da cena, mas também podem levar a alguma perda do poder de discriminação entre imagens.

2.6 Forma

A cognição humana tem na percepção de forma o principal meio para o reconhecimento de objetos. Entretanto a extração de características de forma é um dos aspectos mais difíceis da recuperação de imagens por conteúdo, devido principalmente à dificuldade de segmentar os objetos de interesse presentes nas imagens, e pela exigência que a representação seja invariante à translação, rotação e escala.

Muitas técnicas de representação e descrição de formas foram e continuam sendo desenvolvidas. Conforme [Zhang and Lu 2004] estas técnicas podem ser classificadas em dois grupos: métodos baseados no contorno e métodos baseados em regiões (ver Figura 2.2). Em cada um destes grupos os diferentes métodos são divididos em abordagens estruturais e abordagens globais. Estas abordagens poderiam ainda ser distinguidas entre métodos no domínio do espaço e métodos no domínio de transformadas.

2.6.1 Técnicas Baseadas no Contorno

As técnicas baseadas no contorno exploram somente as informações de bordas (fronteiras). Há dois tipos de abordagens para a modelagem do contorno de formas: a abordagem contínua

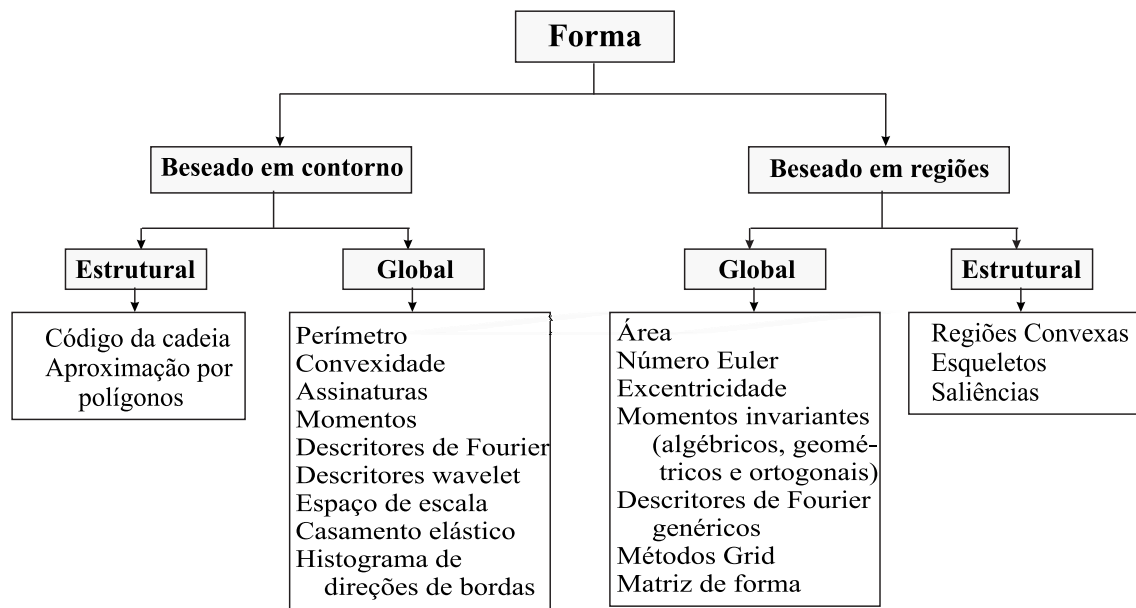


Figura 2.2: Classificação geral das técnicas de representação de formas [Zhang and Lu 2004].

(global) e a abordagem discreta (estrutural).

Na abordagem global, um vetor de característica numérico derivado de toda a borda é usado para descrever a forma. Medidas de similaridade entre formas são usualmente tomadas por alguma métrica de distância.

Abordagens estruturais quebram a borda em segmentos, chamados primitivas, usando algum critério particular. A representação final é usualmente uma *string* ou um grafo. A medida de similaridade é feita por casamento de *strings* ou de grafos.

Os descritores baseados no contorno supõem que o contorno da imagem em análise tenha sido detectado previamente.

Métodos Globais

Descritores mais simples nesta categoria incluem: excentricidade (comprimento do eixo maior / comprimento do eixo menor), orientação do eixo maior e energia da curvatura.

Descritores mais sofisticados incluem:

- **Histograma de direção de bordas** - é um dos descritores de forma mais populares [Brandt 1999]. Sua probabilidade de sucesso confia em sua simplicidade e no fato de não necessitar de segmentação de objetos. Primeiramente uma imagem é pré-processada para extrair as bordas e suas direções. Operadores de Canny, de Sobel, ou derivações Gaussianas podem ser usados na detecção de bordas. Depois deste pré-processamento um histograma é formado pela frequência das direções de borda. Histogramas de bordas são representações compactas, invariantes às operações de translação e de escala.
- **Assinatura de forma** - uma assinatura representa uma forma por uma função unidimensional derivada dos pontos de borda. Existem muitas assinaturas de forma: distância

ao centro, ângulo tangente, curvatura, etc. Assinaturas de formas normalizadas são invariantes a translação e escala, contudo, são sensíveis a ruído. A Figura 2.3 mostra uma forma e sua assinatura correspondente obtida pelo método da distância ao centro.

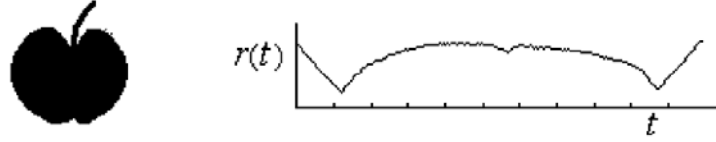


Figura 2.3: Uma forma e sua assinatura obtida pelo método da distância ao centro [Zhang and Lu 2004].

- **Momentos de borda** - são utilizados para reduzir as dimensões das representações de bordas. Por exemplo, assumindo que a borda de uma forma tenha sido representada por uma assinatura z o r -ésimo momento m_r e o momento central μ_r pode ser estimado como em [Zhang and Lu 2004] por:

$$m_r = \frac{1}{N} \sum_{i=1}^N [z(i)]^r \quad e \quad \mu_r = \frac{1}{N} \sum_{i=1}^N [z(i) - m_1]^r \quad (2.6)$$

onde N é o número de *pixels* de borda. Momentos de borda normalizados como $\bar{m}_r = m_r/(\mu_2)^{r/2}$ e $\bar{\mu}_r = \mu_r/(\mu_2)^{r/2}$ são invariantes às transformações de rotação, translação e escala [Zhang and Lu 2004].

- **Transformadas espectrais** - incluem entre outros os descritores de *Fourier* e descritores *wavelet*. Estes podem ser derivados de transformadas espectrais sobre assinaturas de formas unidimensionais ou derivados diretamente dos pontos do contorno [Zhang 2002]. Descritores de *Fourier* consistem na obtenção de um conjunto de coeficientes que representam uma dada borda. Transformadas espectrais produzem representações compactas, de fácil casamento e pouco sensíveis a ruído [Zhang and Lu 2004].
- Demais descritores nesta categoria incluem entre outros, curvatura de espaço de escala e casamento elástico [Zhang 2002].

Métodos Estruturais

Outro membro da família dos métodos de análise de bordas é a representação estrutural de forma. Com a abordagem estrutural, as formas são quebradas em segmentos de contorno chamados primitivas. Os métodos estruturais diferem entre si na seleção de primitivas e na organização destas para a representação de formas. Representações estruturais podem ser codificadas como uma *string* da forma:

$$S = s_1, s_2, \dots, s_n$$

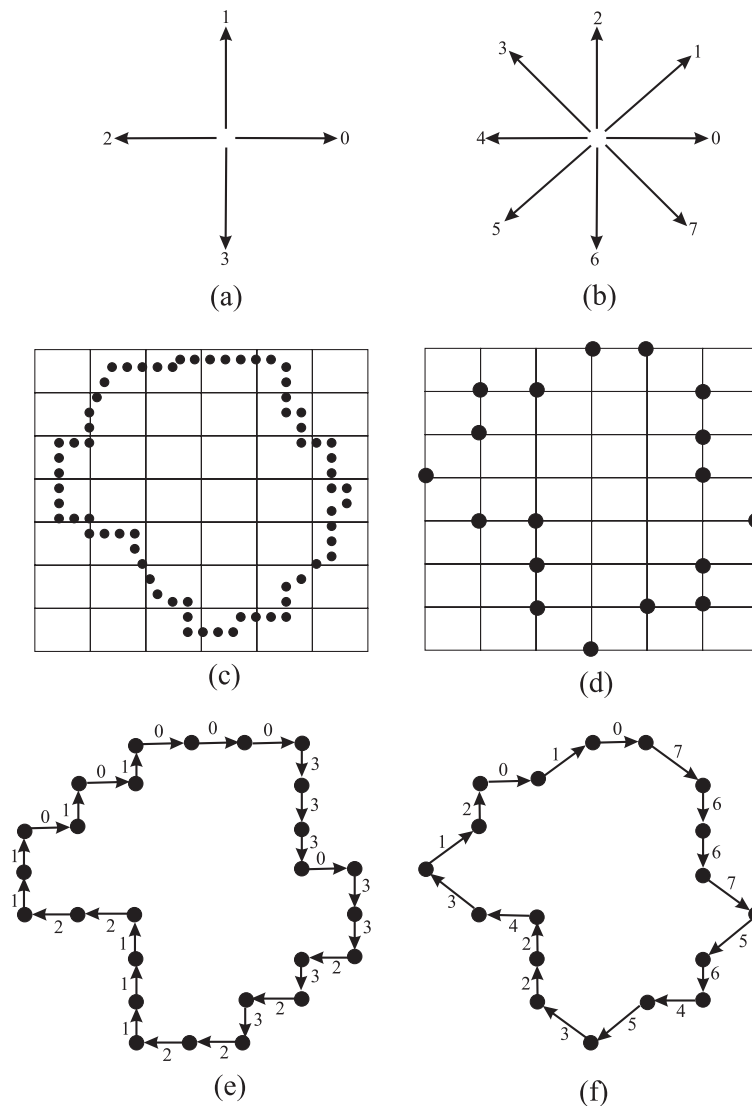


Figura 2.4: Código da cadeia; (a) e (b) códigos de 4 e 8 direções, respectivamente; (c) um contorno sobreposto por uma grade; (d) aproximação dos pontos do contorno para os pontos de intersecção da grade; (e) e (f) codificação do contorno pelo código de 4 e 8 direções, respectivamente.

onde s_i poder ser um elemento de um código da cadeia, um lado de um polígono, etc. s_i pode conter atributos como comprimento, curvatura, energia da curvatura, orientação, etc.

Uma *string* S pode ser comparada com outras para verificar o grau de casamento ou usada como entrada de um analisador sintático de alto nível como autômatos e máquina de *Turing*, que verificam se a forma descrita pela *string* S pertence a uma determinada categoria.

Entre os métodos estruturais mais conhecidos temos:

- **Código da cadeia** - descreve um contorno fechado por uma seqüência de segmentos de linha reta de tamanho unitário com uma dada orientação. Na implementação a borda da imagem é sobreposta por uma grade, onde um dado ponto de borda é aproximado para o ponto da grade mais próximo, obtendo assim uma borda amostrada. De um ponto

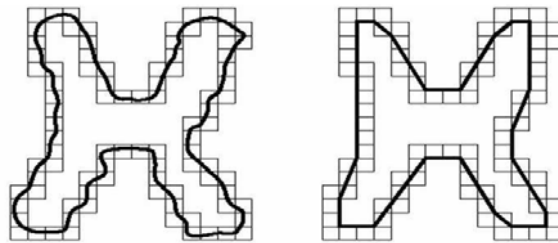


Figura 2.5: Aproximação por Polígonos [Zhang 2002]. À esquerda temos um contorno sobreposto por células. À direita temos a aproximação por polígono traçada internamente às células.

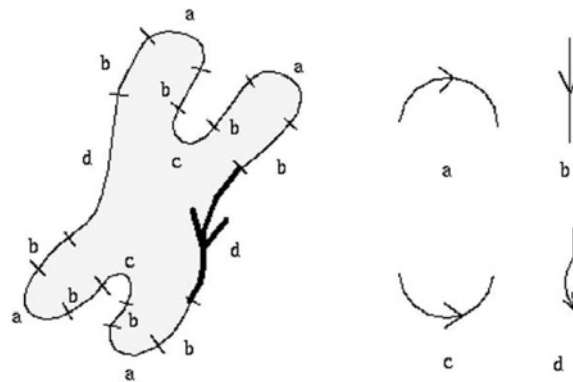


Figura 2.6: Análise sintática da forma de um cromossomo [Zhang and Lu 2004]. À direita temos os elementos primitivos. À esquerda temos o cromossomo representado a partir dos elementos primitivos

de partida selecionado, um código da cadeia pode ser gerado por seguir os pontos de bordas usando 4, 8 ou N ($N > 8$; $N = 2^k, k \in \mathbb{N}$) direções. O código da cadeia deve ser independente da escolha de seu ponto de partida. Uma maneira de fazer isso é tratar o código gerado como uma seqüência circular de números, sendo o ponto inicial dado de modo que o número formado seja de menor magnitude possível. Outras formas de normalização são apresentadas em [Zhang and Lu 2004]. A Figura 2.4 ilustra a aplicação de códigos 4 e 8 direcionais a uma dada forma. O tamanho da grade que sobrepõe o contorno da imagem original determina a dimensão do código da cadeia.

- **Aproximação por polígonos** - em tal abordagem um contorno fechado é aproximado por um polígono, onde se busca representar tal contorno com o menor número possível de segmentos poligonais, de acordo com a precisão que se deseje. O polígono obtido deve possuir perímetro mínimo, ou seja, dada uma fronteira conexa S , alguma técnica deve ser utilizada para que seja gerado um polígono que possua forma semelhante à original e o menor número possível de lados. Comumente se utiliza sobrepor a fronteira com um conjunto de células de tamanho pré-definido de maneira que o polígono possa ser traçado dentro das células [Zhang 2002]. A Figura 2.5 ilustra tal procedimento.
- **Análise Sintática** - parte do suposto que a composição de uma forma é análoga à composição de uma linguagem. Nos métodos sintáticos as formas são representadas por primi-

tivas e seus relacionamentos descritos por gramática [Zhang and Lu 2004]. Por exemplo, o cromossomo da Figura 2.6 pode ser representado por suas primitivas (a, b, c, d) e descrito pela *string*:

$$S = dbabcbabdbabcbab.$$

Neste caso a similaridade entre formas é medida por métodos de casamento de *strings*. Maiores detalhes sobre métodos sintáticos para descrição e casamento de formas podem ser encontrados em [Fu 1974].

2.6.2 Técnicas Baseadas em Regiões

Nas técnicas baseadas em regiões, todos os *pixels* dentro de uma região da imagem são levados em conta na representação de formas, ao invés de usar somente as informações do contorno. Similarmente aos métodos baseados em contorno, os métodos baseados em regiões podem ser divididos em métodos globais e métodos estruturais.

Métodos Globais

Tratam a forma como um todo. Estes métodos medem a distribuição dos *pixels* de regiões, sendo menos prováveis de serem afetados por ruídos ou variações. O resultado da descrição é um vetor de características numérico. Similaridades entre formas são medidas por métricas de distância. Descritores simples nesta categoria incluem: área, circularidade (perímetro²/área) e excentricidade.

- **Momentos invariantes** - os mais utilizados são os momentos geométricos e os momentos ortogonais de Zernike. Se o objeto R é representado como uma imagem binária, os momentos centrais de ordem $p + q$ são definidos como [Long et al. 2003]:

$$\mu_{p,q} = \sum_{(x,y) \in R} (x - x_c)^p (y - y_c)^q \quad (2.7)$$

onde (x_c, y_c) é o centro do objeto. O momento central pode ser normalizado para se tornar invariante a escala:

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^\gamma}, \quad \gamma = \frac{p + q + 2}{2} \quad (2.8)$$

Das equações 2.7 e 2.8 podem ser derivados um conjunto de **momentos geométricos invariantes** à translação, rotação e escala [Long et al. 2003]:

$$\begin{aligned} \phi_1 &= \eta_{2,0} + \eta_{0,2} \\ \phi_2 &= (\eta_{2,0} + \eta_{0,2})^2 + 4\eta_{1,1}^2 \\ \phi_3 &= (\eta_{3,0} - 3\eta_{1,2})^2 + (3\eta_{2,1} - \eta_{0,3})^2 \\ \phi_4 &= (\eta_{3,0} + \eta_{1,2})^2 + (\eta_{2,1} - \eta_{0,3})^2 \\ \phi_5 &= (\eta_{3,0} - 3\eta_{1,2})(\eta_{3,0} + \eta_{1,2})[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2] + 3(\eta_{2,1} - \eta_{0,3})(\eta_{2,1} + \end{aligned}$$

$$\begin{aligned} \phi_6 &= \eta_{0,3}[3(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2] \\ \phi_7 &= (\eta_{2,0} + \eta_{0,2})[(\eta_{3,0} + \eta_{1,2})^2 - (\eta_{2,1} + \eta_{0,3})^2] + 4\eta_{1,1}(\eta_{3,0} + \eta_{1,2})(\eta_{2,1} + \eta_{0,3}) \\ \phi_7 &= (3\eta_{2,1} - \eta_{0,3})(\eta_{3,0} + \eta_{1,2})[(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2] + (\eta_{3,0} - 3\eta_{1,2})(\eta_{2,1} + \eta_{0,3}) \\ &\quad \eta_{0,3}[3(\eta_{3,0} + \eta_{1,2})^2 - 3(\eta_{2,1} + \eta_{0,3})^2] \end{aligned}$$

Maiores detalhes sobre os momentos geométricos invariantes podem ser vistos em [Hu 1962].

Momentos Zernike utilizam funções de bases ortogonais e são menos sensíveis a ruídos que os momentos geométricos. Momentos Zernike bidimensionais [Long et al. 2003] são dados por:

$$A_{m,n} = \frac{m+1}{\pi} \int f(x,y)[V_{m,n}(x,y)]^* dx dy$$

onde,

$$x^2 + y^2 \leq 1$$

$$m = 0, 1, 2, \dots, \infty$$

$f(x, y)$ é a função a ser descrita (uma imagem)

$V_{m,n}$ denota a função de base Zernike

$*$ denota o complexo conjugado de $V_{m,n}$

n é um inteiro que representa a dependência angular ou rotação

Para maiores detalhes sobre os Momentos Zernike veja [Bin and Jia-Xiong 2002].

- Demais descritores nesta categoria incluem entre outros, descritores genéricos de Fourier, Métodos *Grid* e Matrizes de forma [Zhang and Lu 2004].

Métodos Estruturais

Um tanto similar aos métodos estruturais de contorno, os métodos estruturais baseados em regiões decompõem estas em partes que são então usadas para as descrições. Os métodos estruturais baseados em regiões mais conhecidos são fecho convexo e saliências.

- **Fecho Convexo** - uma região r é convexa se e somente se para quaisquer dois pontos $x_1, x_2 \in r$, o segmento de reta x_1x_2 estiver totalmente dentro da região r . O fecho convexo de uma região é a menor região convexa H que satisfaz a condição $R \subset H$. A diferença $H - R$ é chamada deficiência convexa D da região R [Zhang and Lu 2004]. Regiões e deficiências convexas podem ser descritas em forma de árvores, sendo estas representadas pelos nós folhas. A Figura 2.7 mostra uma forma composta de regiões, representada pelo método do fecho convexo.
- **Saliências** - são usadas para descrever os pontos do contorno de alta curvatura ou as extremidades dos esqueletos de forma. Na Figura 2.8 temos três ilustrações de saliências de uma folha. Em (a) temos os pontos de saliência do contorno, em (b) temos as saliências do esqueleto interno e em (c) as saliências do esqueleto externo. Maiores detalhes sobre saliências pode ser encontrados em [Torres and Falcão 2006].

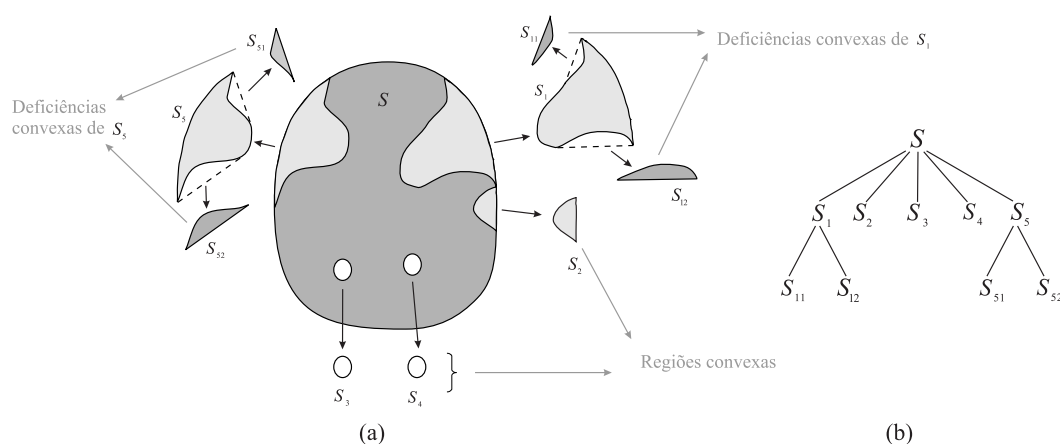


Figura 2.7: Fecho convexo; (a) Regiões e deficiências convexas (b) representação das regiões e deficiências convexas por árvore.

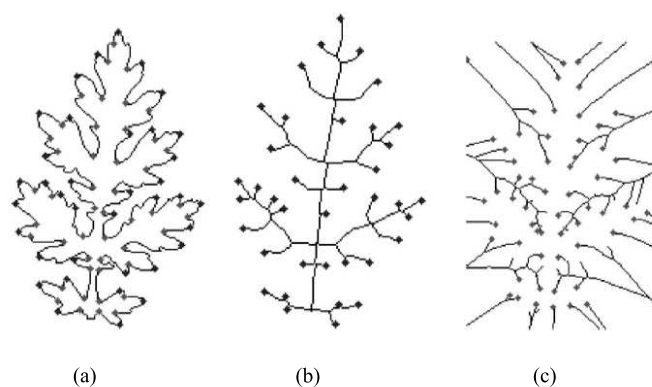


Figura 2.8: Pontos de saliências de uma folha; (a) saliências do contorno (b) e (c) saliências dos esqueletos internos e externos, respectivamente.

2.7 Textura

Texturas são aspectos presentes nas imagens, caracterizados pela relação entre seus *pixels*, ao contrário de cor que é uma propriedade individual de cada *pixel*. Texturas podem manifestar diversos aspectos como homogeneidade, randomicidade, granularidade, aspereza, etc., resultantes dos princípios de combinações de cores (ou níveis de cinza) que formam as imagens. Ela é uma característica fortemente discriminante pelo sistema visual humano e está presente em quase todo lugar na natureza. A Figura 2.9 mostra algumas imagens de textura encontradas na natureza.

Na literatura há três abordagens básicas para a descrição de textura:

- **Estrutural** - utilizam a idéia de que texturas são compostas de primitivas dispostas de forma aproximadamente regular, de acordo com regras bem definidas. Exemplos de tais texturas incluem telhados, estruturas celulares, amostra de tecido, parede de tijolo, etc. Contudo, esta abordagem é limitada, dado que a identificação automática destas primitivas é um problema difícil, e também porque esta é somente aplicada a texturas

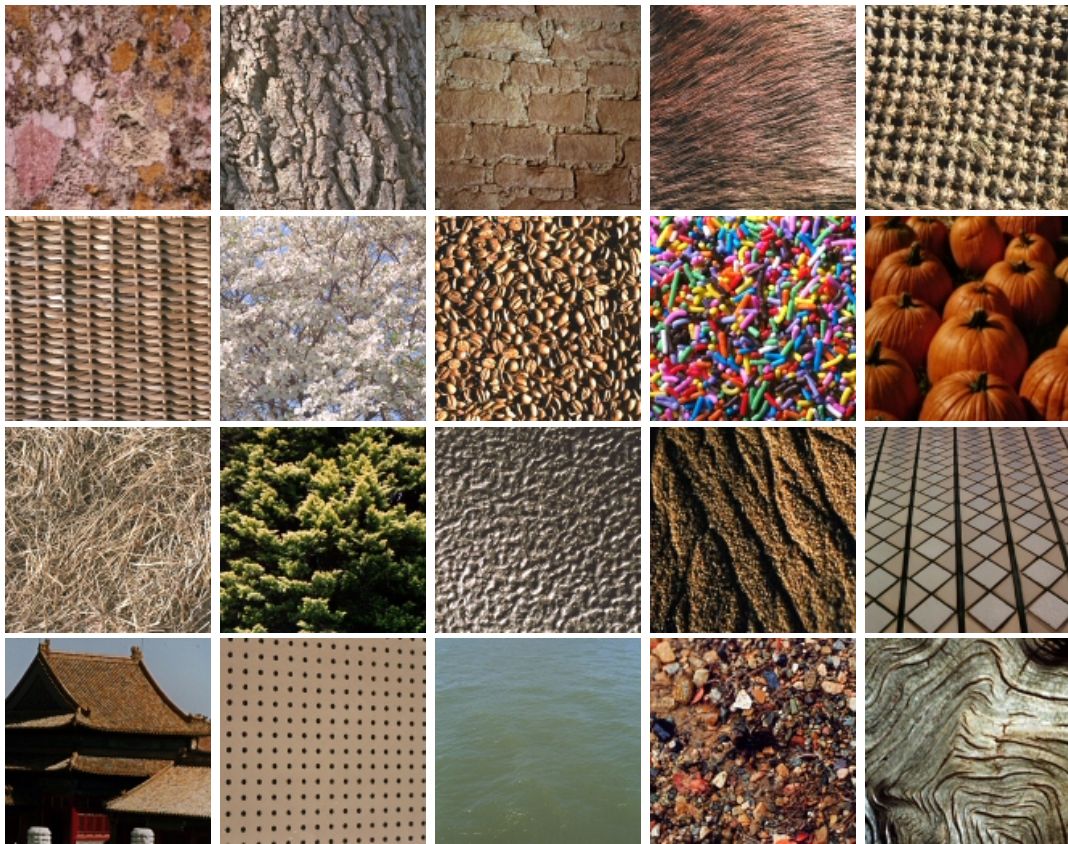


Figura 2.9: Imagens de texturas naturais

estritamente uniformes, o que é muito raro em imagens reais. Métodos estruturais para descrição de textura incluem operadores morfológicos, grafos de adjacência e diagramas de Voronoi.

- **Estocástica** - texturas são assumidas como a realização de um processo estocástico governado por alguns parâmetros. A análise de textura é efetuada por definir um modelo e estimar seus parâmetros de modo que este processo estocástico possa reproduzir o padrão de textura. Os parâmetros estimados servem como características de textura para a recuperação, classificação e segmentação. A dificuldade desta modelagem de textura é que muitas texturas naturais não obedecem às restrições de um modelo particular. Métodos estocásticos incluem cadeias randômicas de Markov, cadeias randômicas de Gibbs e fractais. Estes métodos são bastante úteis na síntese de textura, mas nem sempre são satisfatórios para a recuperação e classificação de imagens.
- **Estatística** - as propriedades das texturas são descritas por medidas estatísticas tais como: contraste, correlação, entropia, uniformidade, densidade, aspereza, rugosidade, regularidade, linearidade, direcionalidade e frequência. Estas medidas são fortemente baseadas nos aspectos da percepção humana de textura. Devido a grande popularidade dos métodos estatísticos na descrição de textura, revisaremos a seguir alguns deles.

2.7.1 Vizinhança de Textura

É calculada a partir dos níveis de cinza de uma imagem. Examina-se a vizinhança de oito de cada *pixel* da imagem e obtém uma estimativa da probabilidade para cada posição relativa de que um *pixel* vizinho tenha maior intensidade de brilho que o *pixel* central [Laaksonen 2000].

2.7.2 Matrizes de Co-ocorrência

A matriz de co-ocorrência [Haralick 1979] é a abordagem mais tradicional para a análise de textura. Esta se baseia na caracterização de uma imagem de textura a partir de dados estatísticos associados à ocorrência simultânea dos níveis de cinza dos *pixels* na imagem.

Uma matriz de co-ocorrência $P_D(i, j)$ contabiliza a co-ocorrência de *pixels* de níveis de cinza i e j em uma dada distância D . A distância D é definida em coordenadas polares (d, α) , onde d é a magnitude e $\alpha = \{0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 315^\circ\}$ é a orientação. Os valores de d são definidos de acordo com a granularidade das imagens da coleção a ser descrita. A matriz de co-ocorrência $P_D(i, j)$ pode ser definida como:

$$P_D(i, j) = \Pr \left\{ (i(p_1) = i) \wedge (i(p_2) = j) \wedge (\|p_1 - p_2\| = D) \right\} \quad (2.9)$$

onde $\Pr$ denota a probabilidade, p_1 e p_2 denotam as posições de dois *pixels* de uma dada imagem, e $i(p)$ denota a intensidade do *pixel* p .

Uma vez computada a matriz de co-ocorrência, várias características de textura podem ser extraídas desta. [Haralick 1979] propôs um conjunto de 14 estatísticas obtidas da matriz de co-ocorrência para caracterizar texturas. Destas as mais utilizadas na recuperação de imagens são as listadas na tabela 2.1, onde μ_i e μ_j são as médias e σ_i e σ_j são os desvios padrões de P .

Característica	Equação
Contraste	$f_1 = \sum_i \sum_j (i - j)^2 P(i, j)$
Energia	$f_2 = \sum_i \sum_j P^2(i, j)$
Entropia	$f_3 = \sum_i \sum_j P(i, j) - \log P(i, j)$
Homogeneidade	$f_4 = \sum_i \sum_j \frac{P(i, j)}{(1 + i - j)}$
Correlação	$f_5 = \frac{1}{\sigma_i \sigma_j} \sum_i \sum_j (i - \mu_i)(j - \mu_j) P(i, j)$

Tabela 2.1: Características deriváveis da matriz de co-ocorrência.

Características extraídas da matriz de co-ocorrência apresentam boa capacidade de discriminação de texturas. A principal desvantagem é o seu alto custo computacional.

2.7.3 Características Tamura

As características Tamura [Tamura et al. 1978] incluem as medidas de rugosidade, contraste, direcionalidade, semelhança de linhas, regularidade e aspereza. Estas medidas foram projetadas em conformidade com estudos psicovisuais acerca da percepção humana de textura. Os três primeiros componentes das características Tamura foram usados em dois *CBIRs* populares: o *QBIC* [Flickner et al. 1995] e o *Photobook* [Pentland et al. 1996]. A computação destas

três características são dadas a seguir. As demais medidas (semelhança de linhas, regularidade e aspereza) podem ser vistas em [Tamura et al. 1978].

Rugosidade - é uma medida da granularidade da textura. Inicialmente, para cada *pixel* (x, y) são calculadas as médias $A_k(x, y)$ dos níveis de cinza em cada vizinhança 2^k , $k = 2, 3, \dots, L$. Posteriormente, para cada *pixel* e cada vizinhança é calculado a diferença entre pares de médias vizinhas não-sobrepostas nas direções horizontal h e vertical v ,

$$\begin{aligned} E_{k,h}(x, y) &= \left| A_k(x + 2^{k-1}, y) - A_k(x - 2^{k-1}, y) \right| \\ E_{k,v}(x, y) &= \left| A_k(x, y + 2^{k-1}) - A_k(x, y - 2^{k-1}) \right| \end{aligned} \quad (2.10)$$

Depois disso, é calculado para cada *pixel* o tamanho da vizinhança S_{best} que maximiza o valor de E ,

$$S_{best}(x, y) = 2^k \quad (2.11)$$

Finalmente, a rugosidade é definida como a média dos valores de S_{best} ,

$$F_{rugosidade} = \frac{1}{m \times n} \sum_{i=1}^m \sum_{j=1}^n S_{best}(i, j) \quad (2.12)$$

onde $m \times n$ corresponde à dimensão da imagem.

Contraste - é calculado pela seguinte formula:

$$F_{contraste} = \frac{\sigma}{\alpha_4^{1/4}} \quad (2.13)$$

onde $\alpha_4 = \mu_4/\sigma^4$ é o curtose, μ_4 é o quarto momento sobre a média, e σ^2 é a variância. Esta fórmula pode ser aplicada à imagem inteira ou a regiões.

Direcionalidade - para computar a direcionalidade é feito uma convolução entre a imagem

e os operadores direcionais $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ e $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$, e em seguida é calculado

um vetor gradiente para cada *pixel*. A magnitude $|\Delta G|$ e o ângulo θ de cada vetor são dados por:

$$\begin{aligned} |\Delta G| &= (|\Delta_H| + |\Delta_V|)/2 \\ \theta &= \tan^{-1}(\Delta_V/\Delta_H) + \pi/2 \end{aligned} \quad (2.14)$$

onde Δ_H e Δ_V são as diferenças horizontais e verticais da convolução.

Em seguida é construído um histograma dos valores de θ (histograma de direções), considerando somente os valores de magnitude $|\Delta G|$ maior que um limiar T . Este histograma H_D apresenta fortes picos para imagens altamente direcionais e é relativamente aplanado para imagens sem fortes orientações. O histograma inteiro é então resumido por um valor de direcionalidade por:

$$F_{\text{direcionalidade}} = \sum_{\forall p \in P} \sum_{\phi \in w_p} (\phi - \phi_p)^2 H_D(\phi) \quad (2.15)$$

onde P é o conjunto de picos do histograma, w_p é o conjunto de intervalos direcionais entre dois vales que delimitam um pico, e ϕ_p é a direção onde se encontra o pico.

2.7.4 Características *Wold*

A decomposição *Wold* [Liu and Picard 1996] provê outra abordagem para a descrição de textura em termos de propriedades perceptivas. As três componentes *Wold*, harmonia, *evanescence* e indeterminismo, correspondem a periodicidade, direcionalidade e randomicidade de texturas, respectivamente. Texturas periódicas têm um forte componente harmônico, texturas altamente direcionais têm um forte componente *evanescence*, e texturas menos estruturadas tendem a ter um forte componente indeterminístico.

A decomposição *Wold* bidimensional permite que um campo randômico discreto e homogêneo $\{y(m, n), (m, n \in \mathbb{Z})\}$, seja decomposto em três componentes mutuamente ortogonais:

$$y(m, n) = u(m, n) + d(m, n) = u(m, n) + h(m, n) + e(m, n) \quad (2.16)$$

onde $u(m, n)$ é o componente indeterminístico, e $d(m, n)$ é o componente determinístico que poder ser decomposto no componente harmônico $h(m, n)$ e no componente *evanescence* $e(m, n)$. No domínio da frequência temos que:

$$F_y(\xi, \eta) = F_u(\xi, \eta) + F_d(\xi, \eta) = F_u(\xi, \eta) + F_h(\xi, \eta) + F_e(\xi, \eta) \quad (2.17)$$

onde $F_y(\xi, \eta)$, $F_u(\xi, \eta)$, $F_d(\xi, \eta)$, $F_h(\xi, \eta)$, $F_e(\xi, \eta)$ são as funções de distribuição espectral de $\{y(m, n)\}$, $\{u(m, n)\}$, $\{d(m, n)\}$, $\{h(m, n)\}$ e $\{e(m, n)\}$, respectivamente.

No domínio espacial, os três componentes ortogonais podem ser obtidos pela estimação de probabilidade máxima, que consiste em ajustar um processo auto-regressivo de alta ordem, minimizando uma função custo e resolver um conjunto de equações lineares. No domínio da frequência, componentes *Wold* podem ser obtidos por limiares globais de magnitudes do espectro de Fourier das imagens.

2.7.5 Características da Filtragem *Gabor*

O filtro *Gabor* é bastante usado para extrair características de imagem, especialmente de textura. Existe muitas propostas de caracterização de textura com base na filtragem *Gabor*, no entanto estas seguem a mesma idéia básica.

Uma função *Gabor* bidimensional é definida como:

$$g(x, y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp \left[-\frac{1}{2} \left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right) + 2\pi j W x \right] \quad (2.18)$$

onde σ_x e σ_y são os desvios padrões dos núcleos Gaussianos ao longo das direções x e y . Então um conjunto de filtros *Gabor* podem ser obtidas por dilatações e rotações de $g(x, y)$:

$$\begin{aligned} g_{mn}(x, y) &= a^{-m} g(x', y') \\ x' &= a^{-m} (x \cos \theta + y \sin \theta) \\ y' &= a^{-m} (-x \sin \theta + y \cos \theta) \end{aligned} \quad (2.19)$$

onde $a > 1$, $\theta = n\pi/K$, $n = 0, 1, \dots, K-1$, e $m = 0, 1, \dots, S-1$. K e S são os números de orientações e escalas.

Dada uma imagem $i(x, y)$, sua transformada de Gabor é definida como:

$$W_{mn}(x, y) = \int i(x, y) g_{mn}^*(x - x_1, y - y_1) dx_1 dy_1 \quad (2.20)$$

onde $*$ indica o complexo conjugado. Então a média μ_{mn} e o desvio padrão σ_{mn} da magnitude de $W_{mn}(x, y)$, isto é, $f = [\mu_{00}, \sigma_{00}, \dots, \mu_{mn}, \sigma_{mn}, \dots, \mu_{S-1K-1}, \sigma_{S-1K-1}]$ podem ser usados para representar as características de textura de uma imagem.

2.7.6 Características da Transformada *Wavelet*

Similar à filtragem Gabor, a transformada *wavelet* [Mallat 1992] produz uma abordagem multi-resolução para a análise e classificação de textura. A transformada *wavelet* decompõe um sinal com uma família de funções bases $\Psi_{mn}(x)$ obtidas através de translações e rotações de uma *wavelet mãe* $\Psi(x)$, isto é,

$$\Psi_{mn}(x) = 2^{-m/2} \Psi(2^{-m}x - n) \quad (2.21)$$

onde m e n são os parâmetros de dilatação e translação, respectivamente. Um sinal $f(x)$ pode ser representado como:

$$f(x) = \sum_{m,n} c_{mn} \Psi_{mn}(x) \quad (2.22)$$

A computação da transformada *wavelet* de um sinal 2D envolve filtragem recursiva e sub-amostragem. Em cada nível, o sinal é decomposto em quatro sub-bandas de frequência, *LL*, *LH*, *HL* e *HH*, onde *L* denota frequência baixa e *H* denota frequência alta. Os dois tipos principais de transformada *wavelet* usadas para análise de textura são a transformada *wavelet* estruturada em pirâmide (*pyramid-structured wavelet transform - PWT*) e a transformada *wavelet* estruturada em árvore (*tree-structured wavelet transform - TWT*). A *PWT* decompõe recursivamente a banda *LL*. Contudo, para algumas texturas a informação mais importante freqüentemente aparece nos canais de média frequência. Para superar este problema, a *TWT* decompõe outras bandas tais como *LH*, *HL* ou *HH* quando necessário.

Depois da decomposição, vetores de características podem ser construídos usando a média e o desvio padrão da distribuição de energia de cada sub-banda em cada nível.

2.8 Similaridade

A similaridade, no contexto dos CBIRs, é uma estimativa da semelhança entre duas imagens. Esta pode ser caracterizada por dois fatores: a semelhança de características e a importância de cada uma destas. Desta forma, os sistemas *CBIRs* modernos costumam incluir, além das tradicionais medidas de similaridades de características, um modelo de similaridade entre imagens. A principal função de um modelo de similaridade é definir como as características serão comparadas para a obtenção do grau de similaridade entre imagens.

2.8.1 Modelos de Similaridade

Dado que imagens podem ser representadas por múltiplas características como cor, forma e textura, extraídas da imagem inteira ou de regiões, e como a similaridade de cada uma destas características e regiões podem ter diferentes importâncias, faz-se necessário modelos que agreguem as similaridades parciais para compor um valor final que indica a similaridade entre duas imagens.

Na literatura existem inúmeros modelos de similaridade de imagem. Boa parte destes são baseados em médias ponderadas de similaridades parciais de características e/ou regiões, sendo que há várias maneiras para se definir os pesos: heurísticas, o usuário no momento da busca, pesos ajustáveis por mecanismos de realimentação de relevância, etc. Por exemplo, o sistema *CIRES* [Iqbal and Aggarwal 2002] permite ao usuário definir as importâncias das características de cor, forma e textura no instante da consulta, e posteriormente ajustam suas importâncias através de realimentação de relevância (ver Figura 2.12). O sistema *SIMPLicity* [Wang et al. 2001] utiliza características de cor, forma e textura, extraídas de regiões obtidas por segmentação de imagem e agregadas através da medida *IRM* [Li et al. 2000], onde as características de cor e de textura têm maiores importâncias que as de forma, conforme regras heurísticas. [Chow et al. 2006] utilizou um modelo de similaridade de imagem estruturado em árvore onde, primeiramente as imagens são comparadas por suas características globais, armazenadas em nós raízes, e em seguida por suas características locais, extraídas de regiões obtidas por segmentação, armazenadas em nós folhas. Desta forma, praticamente todos os sistemas *CBIR* atuais têm em seus projetos, um modelo de similaridade entre imagens.

2.8.2 Medidas de Similaridade

Uma vez que as características de uma imagem tenham sido extraídas e armazenadas em vetores de característica, faz-se necessário medidas que comparem a similaridade entre vetores de característica. Medidas de similaridade são normalmente baseadas em medidas de distância. Algumas das mais utilizadas são dadas a seguir:

Distância Minkowski - é apropriada quando cada coordenada dos vetores de características são independentes entre si e de igual importância.

$$D(X, Y) = \left[\sum_{i=1}^N |X_i - Y_i|^\lambda \right]^{1/\lambda} \quad (2.23)$$

- **Distância Euclidiana** - é um caso particular da Distância Minkowski, quando $\lambda = 2$.

$$D(X, Y) = \sqrt{\sum_{i=1}^N (X_i - Y_i)^2} \quad (2.24)$$

- **Distância em Quadras (*city-block*)** - outro caso particular da Distância Minkowski, quando $\lambda = 1$.

$$D(X, Y) = \sum_{i=1}^N |X_i - Y_i| \quad (2.25)$$

Distância Quadrática - foi introduzida para resolver o problema da distância Minkowski que trata as coordenadas dos vetores de características de forma inteiramente independente e não levam em conta o fato que certas coordenadas corresponderem a características perceptivas mais importantes que outras. A distância quadrática entre X e Y é dada por:

$$D(X, Y) = \sqrt{(X - Y)^T A (X - Y)} \quad (2.26)$$

onde $A = [a_{ij}]$ e a_{ij} denota a similaridade das características de índices i e j .

Distância Mahalanobis - é apropriada quando as coordenadas dos vetores de características são independentes e de diferentes importâncias.

$$D(X, Y) = \sqrt{(X - Y)^T S^{-1} (X - Y)} \quad (2.27)$$

onde S é a matriz de covariância dos vetores de característica

Canberra, Chebyshev, Bottleneck, Hausdorff, Fréchet, Earth Mover's, Banach-Mazar e Cosseno fazem parte da imensa lista de medidas de distância utilizadas na comparação de vetores de características.

Destas medidas de distância a mais utilizada é a Minkowski:

- No sistema MARS [Rui et al. 1997] foi utilizado distância Euclidiana para computar a similaridade de características de textura entre imagens.
- No sistema Netra [Ma and Manjunath 1999] foi utilizado distância Euclidiana para computar as similaridades de características de cor e forma, e distância em Quadras para computar a similaridade de características de textura.

- No trabalho de [Voorhees and Poggio 1988] foi utilizado distância Minkowski com $\lambda = \infty$ para computar as similaridades de textura entre imagens.
- No trabalho de [Swain and Ballard 1991] foi usada a distância em Quadras para computar as similaridades de cor entre imagens.
- Nos sistemas *CIRES* [Iqbal and Aggarwal 2002] e *Blobworld* [Carson et al. 1999] foram utilizados distância Euclidiana para a comparação de características de forma e textura.

Assim como estes, vários outros sistemas *CBIR* utilizaram e/ou utilizam distância Minkowski para comparar vetores de características.

2.9 Sistemas de Recuperação de Imagens

Existe um grande número de sistemas *CBIR* [Veltkamp and Tanase 2002] e novos sistemas são desenvolvidos continuamente. Também pode-se notar grandes diversidades conceituais ligadas ao desenvolvimento destes sistemas. Aqui veremos que os sistemas podem ser classificados quanto a metodologia de busca e quanto a forma de realimentação de relevância.

2.9.1 Metodologias de Busca

Classificações quanto ao tipo de busca suportado:

- **Consulta por exemplo Visual** - o usuário tem uma imagem exemplo e busca por imagens visualmente similares. Usualmente, o sistema *CBIR* retorna as imagens do banco de dados ordenadas conforme a medida de similaridade empregada. Grande parte dos *CBIR* suportam consultas por exemplo visual.
- **Consulta por grupo de exemplo** - permite ao usuário selecionar ou informar múltiplas imagens de consulta. O sistema busca por imagens que melhor casam às características comuns do grupo de exemplo. Sistemas nesta categoria incluem entre outros, o *Image-Group* [Nakazato et al. 2002] e o *PicSom* [Laaksonen et al. 2000], ilustrados pelas Figuras 2.10 e 2.11, respectivamente.
- **Consulta por esboço** - o sistema permite ao usuário desenhar a imagem que será utilizada como consulta. A maioria destes sistemas permitem que a busca seja refinada por realimentação de relevância. *ImageScape* [Lew et al. 1997] e *DrawSearch* [Sciascio et al. 1999] são sistemas nesta categoria.
- **Consulta por palavras-chave** - empregada em sistemas *CBIR* que extraem características de alto nível (semânticas). Um exemplo desta categoria de sistema pode ser visto em [Li et al. 2004]. Normalmente é limitado o conjunto de palavras-chave permitidas, dado que estes sistemas são de aplicações restritas.

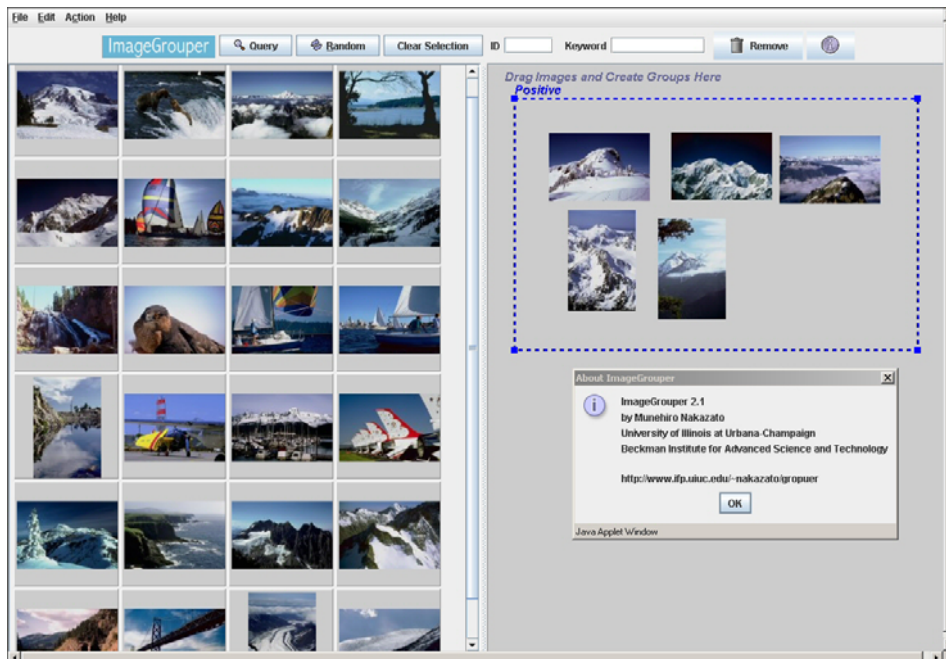


Figura 2.10: O Sistema *ImageGrouper*.

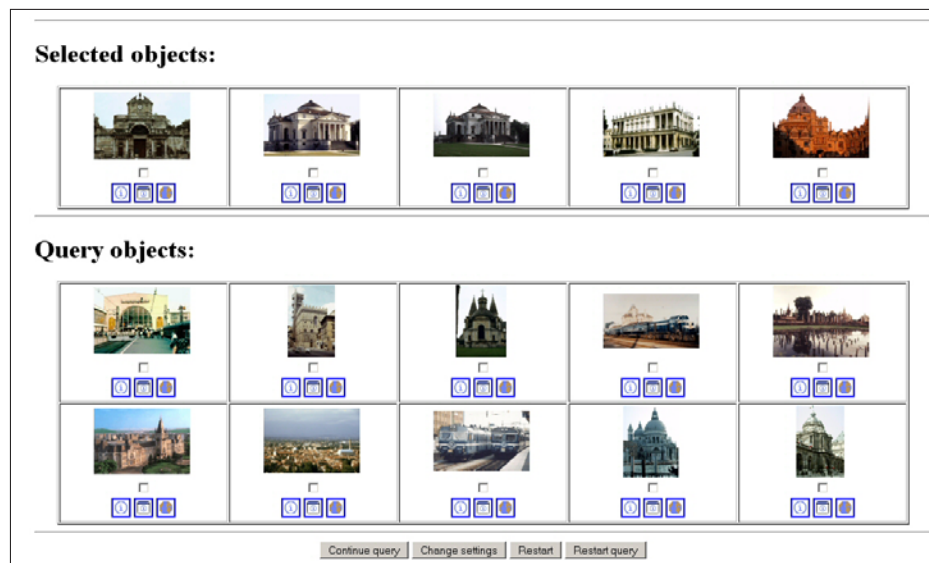


Figura 2.11: O Sistema *PicSom*.

- **Busca por alvo** - o usuário busca por uma imagem particular que ele tem em mente. [de Ves et al. 2006] desenvolveu um sistema onde imagens são mostradas aleatoriamente ao usuário e a partir destas, pode-se realimentar sucessivamente o sistema que irá se adaptar gradativamente aos interesses do usuário em perseguição a seu alvo.
- **Busca por categoria** - o usuário busca por um conjunto de imagens visualmente similares a um exemplo (categoria). Vários sistemas provêm busca por categoria, entre eles

o SIMPLIcity [Wang et al. 2001].

2.9.2 Realimentação de Relevância

No contexto da recuperação de informação, dado uma consulta do usuário, a meta principal é recuperar documentos que podem ser úteis ou relevantes para o usuário.

Para ser efetivo na tentativa de satisfazer a necessidade de informação do usuário, os sistemas de recuperação de imagens devem, de alguma forma, interpretar o conteúdo das imagens de uma coleção e ordená-las de acordo com o grau de similaridade à requisição do usuário. Esta interpretação do conteúdo de uma imagem envolve a extração de características sintáticas e/ou semânticas e o uso destas na comparação com a informação do usuário. A dificuldade do processo, além da extração de informações, é de como utilizar estas informação para decidir o grau de relevância das imagens. Assim, a noção de relevância está no centro de qualquer tarefa de recuperação de informação. Contudo, é o usuário quem melhor conhece o que é relevante no contexto de uma busca em particular. Daí a justificativa para a exploração do conceito de relevância do usuário em um mecanismo conhecido como realimentação de relevância.

Conforme [Baeza-Yates and Ribeiro-Neto 1999] e [Salton and Buckley 1990], realimentação de relevância é uma forma controlada de modificar consultas. A idéia principal é a de usar as informações fornecidas pelo usuário acerca da relevância dos documentos previamente recuperados para realizar uma nova busca, de forma que mais documentos relevantes sejam recuperados.

A realimentação de relevância é motivada pelo fato de que grande parte dos usuários têm dificuldades de formular consultas que atendem de imediato seus propósitos de recuperação. De um modo geral, o desconhecimento do repertório da coleção referência e a falta de familiaridade com o ambiente de recuperação são apontados como os principais complicadores na execução de buscas satisfatórias.

Conforme os princípios da realimentação de relevância, a primeira operação de recuperação é vista como uma tentativa, cuja meta é porventura recuperar alguns elementos úteis da coleção. Estes elementos inicialmente recuperados podem então ser examinados e após avaliada suas relevâncias, podem ser usados para realizar uma nova busca, numa tentativa de encontrar documentos que melhor satisfaçam a necessidade do usuário.

Na recuperação de imagens a realimentação de relevância é estimulada pelo fato de que, ou os usuários formulam consultas compostas de características que não casam com aquelas dos índices da imagens relevantes, ou eles não indicam (e muitas vezes não têm como indicar) qual a relevância dos vários aspectos de uma imagem. Pelo próprio modo de operação de um sistema *CBIR*, existe uma subjetividade inerente no processo de busca. Quando o usuário fornece como consulta uma imagem exemplo, o sistema vê esta imagem como um conjunto de características de baixo nível, já o usuário a vê pela sua semântica. Também uma mesma consulta (imagem exemplo) pode ser interpretada de diferentes formas por diferentes pessoas – subjetividade. Para tratar estes problemas, mecanismos de realimentação de relevância, herdados dos sistemas de recuperação de informação textual, foram introduzidos à recuperação de imagens. Com realimentação de relevância é possível amenizar o *gap* entre conceitos de alto nível e características de baixo nível, o que é conhecido como *gap-semântico*, e o problema da subjetividade.

Sistemas de realimentação de relevância podem trabalhar com exemplos positivos ou po-



Mammals: Elephants



[CIRES](#)
[Query Help](#)
[Sample Queries](#)
[Feedback](#)
[Sitemap](#)

Please select one of the following images:

nat_anm_mam_ele_art_ae0003.jpg

Weights (should sum to 1):

Perceptual grouping: Color: Texture:

Texture:

☒ L, A and B channels
☐ L channel only (~Grayscale texture)

Retrieved Images

				
☐ Yes ☐ NS ☒ No	☒ Yes ☐ NS ☐ No	☒ Yes ☐ NS ☐ No	☒ Yes ☐ NS ☐ No	☐ Yes ☐ NS ☒ No
				
☐ Yes ☐ NS ☒ No	☐ Yes ☐ NS ☒ No	☒ Yes ☐ NS ☐ No	☐ Yes ☐ NS ☒ No	☐ Yes ☐ NS ☒ No
				
☐ Yes ☒ NS ☐ No	☐ Yes ☒ NS ☐ No	☐ Yes ☐ NS ☒ No	☐ Yes ☐ NS ☒ No	☐ Yes ☐ NS ☒ No

Figura 2.12: O Sistema *CIRES*.

sitivos e negativos. Os documentos relevantes para o usuário devem ser identificados como exemplos positivos e os indesejáveis como exemplos negativos (ver ilustração da interface do Sistema *CIRES* na Figura 2.12). Num processo de realimentação de relevância a questão chave é como utilizar exemplos positivos ou positivos e negativos, para refinar a busca do usuário.

Na tentativa de capturar os requerimentos do usuário, os mecanismos de realimentação de relevância ou modificam a representação da consulta do usuário ou modificam a medida de similaridade utilizada para recuperar e ordenar os documentos da coleção de referência

[Doulamis and Doulamis 2006].

Há duas formas principais de modificação de consultas: (1) computar uma nova representação para a consulta (*query point movement*) [Ciocca and Schettini 1999] ou (2) expandir a consulta (*query expansion*), normalmente utilizando múltiplas representações desta (*multiple points query movement*) [French and Jin 2004].

As formas de modificação da medida de similaridade entre imagens são voltadas principalmente para o ajuste de pesos de características e de regiões conforme o conceito de relevância do usuário. Existem diversas abordagens e ferramentas empregados para tal tarefa: abordagens estatísticas (como correlação e desvio padrão), redes neurais, algoritmos genéticos, redes bayesianas, etc. Outra forma de realimentação de relevância recente é a descoberta de funções de *ranking* ou funções de combinação de descritores, via Programação Genética.

No Capítulo 3 revisaremos alguns trabalhos sobre realimentação de relevância através da modificação ou descoberta da medida de similaridade via computação bioinspirada.

2.10 Medidas de Performance

As medidas mais comumente utilizadas para avaliar a performance de um sistema de recuperação de informação são a eficiência (custo computacional) e a eficácia (exatidão) de recuperação. Nesta seção serão apresentadas as principais medidas de eficácia de recuperação utilizadas na análise de desempenho de sistemas *CBIR*.

2.10.1 Curva Precisão-Revocação (*Precision-Recall*)

O critério quantitativo mais usado para avaliação de sistemas de recuperação de informação é a curva precisão-revocação [Baeza-Yates and Ribeiro-Neto 1999]. A precisão para uma consulta q (Eq. 2.28) é definida como a fração entre o número de imagens relevantes recuperadas, $Ra(q)$, e o número de imagens recuperadas, M .

$$Pr(q) = \frac{Ra(q)}{M} \quad (2.28)$$

Por outro lado, a revocação (Eq. 2.29) é definida como a fração entre o número de imagens relevantes recuperadas $Ra(q)$, e o número total de imagens relevantes na coleção, $A(q)$.

$$Re(q) = \frac{Ra(q)}{A(q)} \quad (2.29)$$

Para um sistema “ideal”, ambos $Pr(q)$ e $Re(q)$ devem ser altos. Contudo, em um sistema de recuperação de imagens, quando o número M de imagens retornadas para o usuário aumenta, a precisão diminui, enquanto a revocação aumenta. Por causa disto, ao invés de usar um simples valor de $Pr(q)$ e/ou $Re(q)$ para caracterizar a performance de um sistema, usualmente utiliza-se a curva precisão-revocação.

Nas equações (2.28) e (2.29), precisão e revocação são estimadas para uma dada consulta q . Usualmente os sistemas de recuperação de informação são avaliados para várias consultas

distintas. Para avaliar a performance de recuperação sobre todas as consultas de teste, a taxa média de precisão em cada nível de revocação é adotada:

$$\overline{Pr}(r) = \sum_{i=1}^{N_q} \frac{Pr_i(r)}{N_q} \quad (2.30)$$

onde $\overline{Pr}(r)$ é a precisão média no nível de revocação r , N_q é o número de consultas de teste e $Pr_i(r)$ é a precisão no nível de revocação r para a i -ésima consulta.

2.10.2 Precisão- R

A medida de Precisão- R é dada pela precisão entre os R primeiros documentos do *ranking*, onde R é o número total de documentos relevantes para a consulta em análise. A medida de precisão é definida pela equação 2.28, na subseção anterior.

Uma outra medida derivada da análise de precisão e revocação é a precisão média em níveis de revocação: a precisão média em 11 níveis de revocação e a precisão média em três níveis de revocação (0.25, 0.50, 0.75) são muito utilizadas.

2.10.3 *Average Normalized Modified Retrieval Rank*

Considerando que os sistemas *CBIR* são avaliados principalmente para baixos níveis de revocação, o *Moving Picture Expert Group (MPEG)* propôs uma medida de performance alternativa chamada *average normalized modified retrieval rank (ANMRR)* [Doulamis and Doulamis 2006]. Este critério resume a performance de um sistema por um único número, que leva em conta a posição no *ranking* das imagens relevantes recuperadas.

Seja n_Q o número de consultas e n_I o número de imagens da coleção de referência. Para uma consulta q , considere $A(q)$ o conjunto de imagens relevantes na coleção e $|A(q)|$ o número de elementos de $A(q)$. Então, *ANMRR* é computada como

$$ANMRR = \frac{1}{n_Q} \sum_{q=1}^{n_Q} \frac{MRR(q)}{C(q) + 0.5 - 0.5 \times |A(q)|} \quad (2.31)$$

onde

$$C(q) = \min\{4 \times |A(q)|, 2 \times \max_{k=1}^{n_Q} |A(k)|\}$$

$$MRR(q) = \frac{1}{|A(q)|} \left[\sum_{k=1}^{n_I} Rank(k, q) \right] - 0.5 - \frac{|A(q)|}{2}.$$

A função $Rank(k, q)$ dá a utilidade de uma imagem que aparece na k -ésima posição do *ranking* da consulta q da seguinte forma:

$$Rank(k, q) = \begin{cases} k, & \text{se } k \leq C(q) \text{ e a } k\text{-ésima imagem} \in A(q) \\ C(q) + 1, & \text{se } k > C(q) \text{ e a } k\text{-ésima imagem} \in A(q) \\ 0, & \text{caso contrário.} \end{cases}$$

O valor de $ANMRR$ estará no intervalo $[0.0, 1.0]$. Um baixo valor de $ANMRR$ indica uma alta taxa de recuperação. Idealmente $ANMRR = 0$ se todas as imagens relevantes aparecem nas primeiras posições do *ranking*. Por outro lado, $ANMRR = 1$ representa o pior caso possível de eficiência de recuperação onde nenhuma imagem relevante aparece entre as primeiras do *ranking*.

2.10.4 Average Normalized Similarity Metric Difference

No artigo de Doulamis [Doulamis and Doulamis 2006] foi proposta uma nova medida para avaliar a performance de um sistema *CBIR*, por computar o erro absoluto entre a medição de similaridade atual e a “ideal” sobre todas as imagens relevantes e irrelevantes recuperadas. Idealmente, uma métrica de similaridade retorna um valor próximo de 1 para todas as imagens relevantes e próximo de zero para todas as imagens irrelevantes. Assim, o erro absoluto para uma consulta q , $e(q)$, é definido como:

$$e(q) = \frac{1}{M} \sum_{k \in S_M} |d_n(f_q, f_k) - \tau(q, k)|_2 \quad (2.32)$$

onde $d_n(f_q, f_k)$ expressa a métrica de similaridade atual normalizada entre q e a k -ésima imagem das M primeiras recuperadas, $\tau(q, k)$ refere a medida de similaridade “ideal” entre as respectivas imagens e $|\cdot|_2$ corresponde a norma L_2 (Distância Euclidiana).

Várias consultas são submetidas ao sistema e então o erro médio sobre todas as consultas é usado para estimar a performance. Considera-se que, quanto menor o erro médio, melhor a performance do sistema. Particularmente, assumindo que Q diferentes consultas tenham sido submetidas, a *average normalized similarity metric difference* é dada por:

$$E = \frac{1}{Q} \sum_{q=1}^Q e(q) \quad (2.33)$$

2.11 Considerações Finais

Neste capítulo foi apresentado uma revisão acerca dos conceitos e técnicas de recuperação de imagens. Inicialmente vimos que há duas abordagens principais de sistemas de recuperação de imagens: baseada em anotações textuais e baseada na extração automática de características visuais. Vimos que as primeiras possuem dois gargalos: a subjetividade das anotações e a dificuldade de aplicações destas para grandes coleções de imagens. Posteriormente foi apresentado uma revisão conceitual das técnicas de recuperação de imagens por conteúdo, onde características automaticamente extraídas das imagens tentam contornar os problemas das anotações manuais.

No contexto dos sistemas de recuperação de imagens por conteúdo (*CBIR*) vimos que há dois focos principais de pesquisas: a exploração de características de baixo nível (de cor, forma e textura) e a exploração de características de alto nível (aspectos semânticos). O primeiro foco concentra grande parte das pesquisas atuais enquanto o segundo tem se mostrado uma

tarefa bastante complexa e com poucos resultados práticos. No contexto da exploração de características de baixo nível foram apresentadas diversas representações para as características de cor, forma e textura. Vimos também que a maioria dos sistemas *CBIR* que exploram características de baixo nível normalmente empregam múltiplas características, às vezes extraídas de regiões. As características locais e/ou individuais são agregadas por modelos de similaridade de imagem que proporcionam a estimativa de similaridade entre imagens. Estes sistemas apresentam dois problemas principais: gap-semântico e subjetividade. Realimentação de relevância tem se tornado a principal alternativa para amenizar tais problemas.

Finalizando o Capítulo foi apresentado as principais medidas de eficácia de recuperação usadas para avaliar os sistemas de recuperação de imagens.

Realimentação de Relevância via Computação Evolutiva

3.1 Introdução

Conforme apresentado no capítulo 2, a função principal de um mecanismo de realimentação de relevância é capturar os requerimentos do usuário para realizar uma nova busca que melhor satisfaça suas necessidades. Vimos que existem duas ações principais consideradas pelos mecanismos de realimentação de relevância, (1) modificar a representação da consulta do usuário e (2) modificar ou descobrir a medida de similaridade usada para ordenar os documentos da coleção de referência [Doulamis and Doulamis 2006].

Também foi visto que existem variadas abordagens para cada uma destas ações. Na modificação do centro da consulta podemos ou calcular uma nova representação para a consulta (*query point movement*) [Ciocca and Schettini 1999] ou expandir a consulta (*query expansion*), adicionando e/ou removendo termos (características) e utilizando múltiplas representações da consulta (*multiple points query movement*) [French and Jin 2004]. Já as ações de modificar a medida de similaridade entre imagens são voltadas ou para o ajuste de pesos de atributos (termos, características, regiões, etc.), ou para a descoberta de funções de *ranking* ou de combinação de descritores.

Existem diversas abordagens e ferramentas empregadas para a tarefa de ajuste de pesos: abordagens estatísticas (como correlação e desvio padrão), redes neurais, redes bayesianas, **computação evolutiva** (algoritmos genéticos, estratégias evolutivas e estratégias coevolutivas), etc. Já a descoberta de funções de *ranking* ou de funções de combinação de descritores é um campo de pesquisa recente onde todos os estudos de nosso conhecimento fazem uso da técnica Programação Genética.

Neste capítulo são apresentados quatro trabalhos recentes relacionados à aplicação de técnicas de computação evolutiva em mecanismos de realimentação de relevância. Técnicas de computação evolutiva são aplicáveis a várias tarefas da recuperação de informação. Conforme a revisão de [Cordón et al. 2003] estas técnicas são aplicadas em: indexação automática de documentos, clusterização de termos e de documentos, definição de consultas, aprendizagem de funções de casamento, recuperação de imagens, projeto de perfis de usuários para recuperação

de informação na *internet*, classificação de páginas *Web* e projeto de agentes de busca na *internet*.

Os trabalhos apresentados neste capítulo são: (1) [Fan et al. 2004] - analisaram no contexto de recuperação de informação textual, os efeitos de funções de aptidão em programação genética destinada à descoberta de funções de *ranking* otimizadas conforme um conjunto de treinamento, (2) [López-Pujalte et al. 2003b] - analisaram a eficiência de diferentes funções de aptidão baseadas em ordem em um algoritmo genético (AG) aplicado à realimentação de relevância em busca textual, sendo que o objetivo do AG era inferir os pesos de termos representados conforme o modelo de espaço vetorial, (3) [Stejić et al. 2003a] - utilizaram AG para inferir, a partir da realimentação do usuário, pesos de características (cor, forma e textura) e de regiões de imagens, na tentativa de ajustar um modelo de similaridade de imagens às necessidades momentâneas de busca do usuário, (4) [Torres et al. 2005] utilizaram programação genética para descobrir funções de combinação de descritores na recuperação de imagens.

3.2 Fan et al.

3.2.1 O Problema

Fan, Fox, Pathak e Wu analisaram em [Fan et al. 2004], o efeito de várias funções de aptidão em programação genética (PG) aplicada à descoberta de funções de *ranking*, no contexto da recuperação de informação textual. Função de *ranking* é a função usada por uma engenharia de busca para ordenar documentos de acordo com sua similaridade a uma consulta particular. Uma vez que não é conhecido qual a melhor função de *ranking* para uma consulta [Salton and Buckley 1990] e [Zobel and Moffat 1998], os autores modelaram este problema como um problema de busca de PG: uma função de *ranking* otimizada foi inferida a partir de um conjunto de treinamento, onde para cada consulta eram conhecidos os documentos relevantes.

3.2.2 Programação Genética

A programação genética (PG), assim como os AGs, são técnicas de busca baseadas na teoria da evolução natural. A principal diferença entre estas técnicas é a representação dos indivíduos (as possíveis soluções): em AG cada indivíduo é comumente representado (embora, nem sempre) por uma *string* (numérica ou de caracteres) de comprimento fixo; já a PG utiliza estruturas de dados mais complexas como árvores, listas e pilhas, de tamanhos variados [Langdon 1998].

No trabalho de [Fan et al. 2004] os indivíduos foram representados por árvores que correspondem às funções de *ranking*. Por exemplo, a famosa função de *ranking* *TF-IDF* pode ser representada pela árvore da Figura 3.1.

3.2.3 Especificação do Algoritmo de Programação Genética

Os vários componentes necessários para o desenvolvimento de um sistema de programação genética (PG) são definidos a seguir:

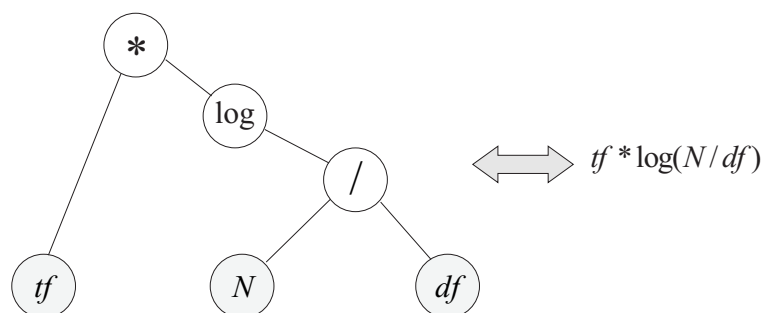


Figura 3.1: Representação em árvore da função de *ranking TF-IDF*. Veja na tabela 3.1 o significado dos símbolos tf , N e df .

- **Terminais** - nós folha na estrutura de árvore, por exemplo, tf , df e N na Figura 3.1. A Tabela 3.1 ilustra os terminais considerados por [Fan et al. 2004].

Terminais	Significado
tf	número de ocorrências de um termo em um documento
tf_{max}	tf máximo para um documento
tf_{avg}	média de tf para um documento
$tf_{doc_{max}}$	tf máximo na coleção de documentos
df	número de documentos que contêm um dado termo
df_{max}	df máximo para uma dada consulta
N	número total de documentos na coleção
$length$	número de termos de um documento
$length_{avg}$	número de termos médio dos documento da coleção
R	uma constante real randômica gerada pelo sistema PG
n	número de termos distintos de um documento

Tabela 3.1: Símbolos terminais usados no sistema PG de [Fan et al. 2004].

- **Funções** - nós não-folha usados para combinar os nós folha. Comumente usa-se operações numéricas como $+$, $-$, $*$ e $/$. Os autores usaram os operadores $+$, $*$, $/$ e $\log$.
- **Função de aptidão** - mede a qualidade dos indivíduos (possíveis soluções para o problema). Os autores desenvolveram quatro funções de aptidão baseadas em ordem e adicionalmente utilizaram três funções de aptidão baseadas em ordem da literatura. Apresentaremos estas na próxima subseção.
- **Seleção** - um operador genético que seleciona os indivíduos com base na sua aptidão, onde normalmente os mais aptos têm maiores probabilidades de serem os escolhidos.
- **Crossover** - um operador genético que troca aleatoriamente sub-árvores de dois pais para formar dois filhos. Veja a Figura 3.2.
- **Mutação** - um operador genético que seleciona randomicamente uma sub-árvore e substitui ela por uma outra criada randomicamente.

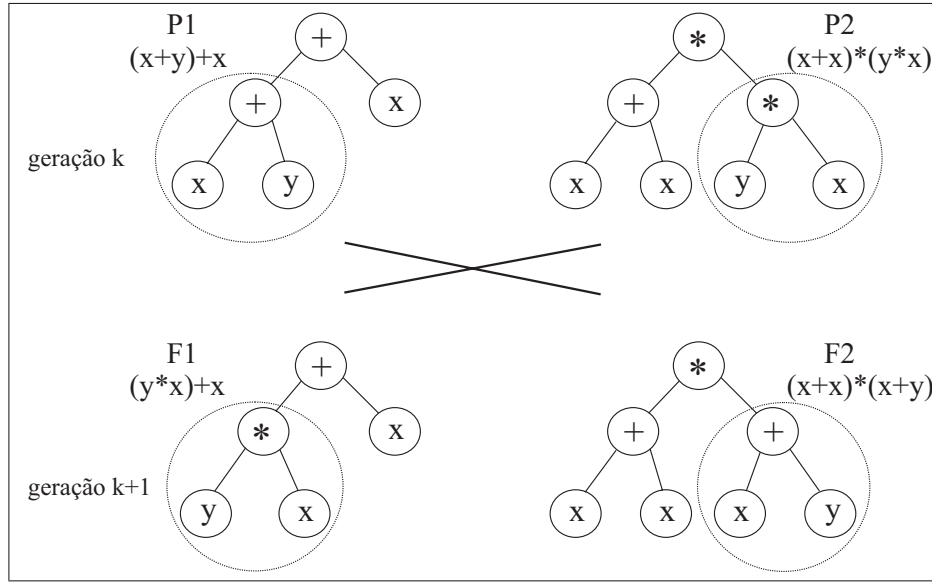


Figura 3.2: Exemplificação de uma operação de *crossover* em árvores.

O fluxo do sistema PG de [Fan et al. 2004] é ilustrado pelo Algoritmo 3.1.

3.2.4 Funções de Aptidão

Em [Fan et al. 2004] foram elaboradas quatro funções de aptidão baseadas em ordem seguindo os princípios da teoria da utilidade [Fishburn 1998]. De acordo com a teoria da utilidade, existe uma função que atribui um valor de utilidade para cada documento no *ranking*. Em geral, assume-se que a utilidade de um documento relevante decresce conforme a sua ordem no *ranking*. Dada uma função de utilidade $U(x)$, e dois documentos quaisquer x_1 e x_2 , com a posição de x_1 menor que a posição de x_2 a condição $U(x_1) > U(x_2)$ é chamada de condição de preservação de ordem. Com base nesta condição [Fan et al. 2004] desenvolveu quatro funções de aptidão, equações 3.1, 3.2, 3.3 e 3.4, onde: $r(d)$ é o indicador de relevância de um documento, sendo 1 se o documento é relevante e 0 caso contrário; $|D|$ é o número total de documentos recuperados; $k_1, \dots, k_9$ são fatores de escala calculados experimentalmente pelos autores e setados com os valores 6, 1.2, 2, 3.65, 0.1, 0.4, 27.32, 7 e 0.982, respectivamente.

$$F1_{Fan \text{ et al.}} = \sum_{i=1}^{|D|} r(d_i) k_1 \ln^{-1}(i + k_2) \quad (3.1)$$

$$F2_{Fan \text{ et al.}} = \sum_{i=1}^{|D|} r(d_i) k_3 \log_{10}^{-1}(1000/i) \quad (3.2)$$

$$F3_{Fan \text{ et al.}} = \sum_{i=1}^{|D|} r(d_i) k_4^{-1} (e^{-k_5 \ln(i) + k_6} - k_7) \quad (3.3)$$

-
- 1 Gerar uma população inicial de árvores de *ranking* de forma randômica
 - 2 Executar os seguintes sub-passos sobre um conjunto de documentos de treinamento por N_g gerações:
 - 2.1 Usar cada árvore como função de *ranking*, separadamente para múltiplas consultas
 - 2.2 Calcular a aptidão de cada árvore de *ranking*
 - 2.3 Registrar as N_{top} melhores árvores de *ranking*
 - 2.4 Criar uma nova população por:
 - a) Seleção
 - b) *Crossover*
 - c) Mutação
 - 3 Aplicar os candidatos registrados (árvores de *ranking*) em um conjunto de documentos de validação e selecionar a melhor árvore como a resposta do problema de busca.
-

Algoritmo 3.1: Funcionamento global do sistema PG de [Fan et al. 2004] para a descoberta de funções de *ranking*. Em todos os seus experimentos foram considerados $N_g = 30$ e $N_{top} = 10$. Os autores não mencionam o tamanho de população utilizado.

$$F4_{Fan \ et \ al.} = \sum_{i=1}^{|D|} r(d_i) k_8 k_9^i \quad (3.4)$$

As outras três funções de aptidão experimentadas por [Fan et al. 2004] são dadas pelas equações 3.5, 3.6 e 3.7. A equação 3.5 é uma padronização da precisão média, uma medida comumente usada em avaliações TREC¹. As equações 3.6 e 3.7 foram usadas no trabalho de [López-Pujalte et al. 2003b]. Nestas equações, $TRel$ é o número total de documentos relevantes na coleção, $r_B(d)$ é uma função que retorna a relevância de um documento d , sendo 1 se d é relevante, e -1 caso contrário.

$$F_{P_{AVG}} = \frac{\sum_{i=1}^{|D|} \left(r(d_i) \left(\frac{\sum_{j=1}^i r(d_j)}{i} \right) \right)}{TRel} \quad (3.5)$$

$$F1_{Lopez-Pujalte \ et \ al.} = \frac{1}{|D|} \sum_{i=1}^{|D|} \left(r(d_i) \sum_{j=i}^{|D|} \frac{1}{j} \right) \quad (3.6)$$

$$F2_{Lopez-Pujalte \ et \ al.} = \left(\sum_{i=1}^{|D|} \left(r_B(d_i) \frac{1}{A} \left(\frac{A-1}{A} \right)^{i-1} \right) \right) \frac{\sum_{i=1}^{|D|} r(d_i)}{|D|} \quad (3.7)$$

¹TREC: do inglês: *Text REtrieval Conference*

3.2.5 Avaliação Experimental

Fan, Fox, Pathak e Wu compararam os resultados obtidos pelo sistema de PG, com os obtidos com a função de *ranking* Okapi BM25, segundo os autores uma função de *ranking* com performance muito boa em competições TREC.

Dada uma consulta q e um documento d a fórmula de *ranking* Okapi BM25 é definida como:

$$Sim_{Okapi} = \sum_{\forall t \in q} \frac{(K_1 + 1)tf}{K_1 \left((1 - b) + b \frac{length}{length_{avg}} \right) + tf} \log \frac{N - df + 0.5}{df + 0.5} qtf \quad (3.8)$$

onde,

tf é a frequência de um termo t no documento d ;

qtf é a frequência de um termo t na consulta q ;

N é o número total de documentos na coleção;

df é o número de documentos na coleção nos quais o termo sob consideração t está presente;

$length$ é o tamanho do documento d em palavras;

$length_{avg}$ é o tamanho médio dos documentos da coleção (em palavras);

K_1 e b são parâmetros usados para ajustar a performance. Os autores usaram $K_1 = 2$ e $b = 0.75$.

Para avaliação [Fan et al. 2004] usaram a coleção TREC 10GB, que contém 1.69 milhões de documentos, com uma média de 311 palavras em cada um. Foram usadas 100 consultas com comprimento de 2 a 13 palavras. Todos os documentos e consultas foram indexados conforme o modelo de espaço vetorial padrão, depois de aplicar a remoção de *stop-word* e *stemming* [Salton 1989].

Os autores dividiram a coleção TREC em três partes: dados de treinamento (49%), de validação (21%), e de teste (30%). Segundo os autores o processo de validação, ajudará a aliviar o problema de *overfitting* da PG, de forma a selecionar uma função de *ranking* de melhor generalização. O sistema foi executado 10 vezes para cada experimento, utilizando sementes randômicas. O melhor resultado entre as 10 execuções foram os reportados.

Na avaliação de performance foram utilizadas a precisão média não interpolada padrão (equação 3.5) e a precisão nos 10 primeiros documentos (*precision at the top 10 hits*). Os resultados reportados para estas duas medidas trata-se da média sobre 100 consultas. Todos os experimentos foram efetuados em um servidor *Linux* de 1.2GHz com 2GB de memória.

A tabela 3.2 mostra os resultados obtidos em [Fan et al. 2004]. Pode-se notar que as funções de aptidão $F2$, $F3$ e $F4$ propostas por [Fan et al. 2004] sobressaíram, obtendo uma melhora acima de 20% sobre a conceituada fórmula Okapi BM25. Nota-se também que a melhor performance é obtida pela função de aptidão $F4_{Fan et al.}$ com uma melhora de aproximadamente 30% em relação à fórmula Okapi BM25.

[Fan et al. 2004] também verificaram (como mostrado na Tabela 3.2) que entre as funções de aptidão, a $F4_{Fan et al.}$ foi a que gastou o menor tempo de treinamento. Como em programação genética (PG) a complexidade de tempo é altamente dependente do tamanho da árvore (indivíduo), uma vez que árvores complexas levam mais tempo para serem avaliadas do que árvores simples, os autores concluíram que as árvores geradas com o uso de $F4_{Fan et al.}$ tendem a ser menos complexas do que as obtidas com o uso de qualquer outra das funções de

Método	P_{AVG}	P10	Tempo de Treinamento
Okapi BM25	0.2230	0.2404	—
PG com $F1_{Fan et al.}$	0.2203 (-1.21%)	0.2420 (+0.67%)	75 horas
PG com $F2_{Fan et al.}$	0.2895 (+29.82%)	0.2932 (+21.96%)	80 horas
PG com $F3_{Fan et al.}$	0.2899 (+30.00%)	0.2932 (+21.96%)	67 horas
PG com $F4_{Fan et al.}$	0.2995 (+34.30%)	0.3114 (+29.53%)	61 horas
PG com $F_{P_{AVG}}$	0.2509 (+12.51%)	0.2511 (+4.45%)	69 horas
PG com $F1_{L-Pujalte et al.}$	0.2666 (+19.55%)	0.2580 (+7.32%)	70 horas
PG com $F2_{L-Pujalte et al.}$	0.2136 (-4.22%)	0.2386 (-0.75%)	84 horas

Tabela 3.2: Resultados obtidos por [Fan et al. 2004].

aptidão disponíveis. Tal conclusão se fundamenta no fato do número de gerações ser fixo para todos os experimentos.

3.3 López-Pujalte et al.

3.3.1 O Problema

López-Pujalte, Guerrero-Bote e Moya-Anegón avaliaram em [López-Pujalte et al. 2003b] a eficiência de diferentes funções de aptidão baseadas em ordem em um AG aplicado à realimentação de relevância em busca textual. Seus experimentos foram baseados no modelo de espaço vetorial e os resultados comparados com os do método clássico de realimentação de relevância *Ide dec-hi* [Salton and Buckley 1990]. A tarefa do AG era encontrar os pesos dos termos empregados na medida de similaridade (equação 3.9), conforme a realimentação do sistema. Anteriormente, os autores [López-Pujalte et al. 2003b] tinham constatado a partir dos trabalhos [López-Pujalte et al. 2002] e [López-Pujalte et al. 2003a] que entre as funções de aptidão disponíveis na literatura para este problema, as que obtêm os melhores resultados são aquelas que levavam em conta não somente a quantidade de documentos relevantes recuperados, mas também a ordem em que estes aparecem no *ranking* – funções de aptidão baseadas em ordem.

$$sim(d_j, q) = \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| |\vec{q}|} = \frac{\sum_{i=1}^t \omega_{i,j} \omega_{i,q}}{\sqrt{\sum_{i=1}^t \omega_{i,j}^2} \sqrt{\sum_{i=1}^t \omega_{i,q}^2}} \quad (3.9)$$

3.3.2 Especificação do Algoritmo Genético

As características do AG de [López-Pujalte et al. 2003b] são dadas a seguir:

- **Representação de indivíduo** - os vetores correspondentes aos documentos analisados pelo usuário são transformados nos indivíduos do AG. O número de genes dos indivíduos

corresponde ao número de termos não nulos na consulta e nos documentos da realimentação. A população inicial é construída dos seguintes vetores: a consulta inicial; os documentos relevantes; os documentos não-relevantes; e os documentos não-relevantes com seus pesos negados.

- **Seleção** - é utilizado o método da roleta com elitismo de 1.
- **Operadores genéticos** - foram utilizados: *crossover* simples e mutação randômica (um gene selecionado para mutação é substituído por outro gerado aleatoriamente, dentro do seu domínio)
- **Parâmetros de controle** - conforme [López-Pujalte et al. 2003b] todos os parâmetros de controle foram obtidos experimentalmente. A taxa de *crossover* considerada foi de 80%, probabilidade de mutação de 20%². A população evolui por 20 gerações.
- **As soluções** - o AG retorna como solução o melhor indivíduo encontrado e o centróide daqueles indivíduos com aptidão igual à aptidão máxima na geração final.

As funções de aptidão experimentadas por [López-Pujalte et al. 2003b] são dadas pelas equações 3.6 e 3.7, descritas anteriormente no trabalho de [Fan et al. 2004], e pelo cálculo da precisão média em nove níveis de revocação, sendo as duas últimas de sua própria autoria e a primeira da autoria de [Horng and Yeh 2000].

3.3.3 Avaliação Experimental

Nos experimentos [López-Pujalte et al. 2003b] utilizaram 1398 documentos da coleção *Cranfield*, relacionados à engenharia aeronáutica. Para uma dada consulta são examinados e determinadas as relevâncias dos 15 primeiros documentos e esta informação é fornecida ao algoritmo como a realimentação. Foram utilizadas 33 consultas que têm pelo menos três documentos relevantes entre os 15 primeiros e que faltam cinco ou mais documentos relevantes a serem recuperados. Assim como no trabalho anterior, os documentos foram indexados conforme o modelo de espaço vetorial padrão, depois da remoção de *stop-word* e aplicação da operação de *stemming* [Salton 1989].

Nos experimentos foi utilizado o método da coleção residual e a performance foi medida através da precisão média em três níveis de revocação (0.25, 0.5 e 0.75). Os resultados experimentais de [López-Pujalte et al. 2003b] são ilustrados pela Tabela 3.3. A percentagem de melhoria é tomada em relação ao método sem realimentação.

Os autores concluem que a aplicação de algoritmos genéticos com funções de aptidão baseadas em ordem obteve resultados satisfatórios. Com relação à forma de calcular a solução (melhor indivíduo ou centróide) pode-se perceber que o centróide pode ser uma boa alternativa. Com respeito ao tempo de computação os autores comentam que o método *Ide dec-hi* é muito mais rápido que os demais métodos usando AG. O tempo de resposta usando AG foi de 2 segundos por consulta, em um computador Pentium III, 700MHz e 256 MB de memória, sendo que não houve diferenças significativas de uma função aptidão para outra.

²Segundo os autores a razão desta taxa de mutação, demasiadamente alta para os padrões normais, é evitar a convergência prematura do algoritmo.

Método	Melhor Indivíduo		Centróide	
	Média	Melhoria	Média	Melhoria
Sem realimentação	0.098	—	—	—
<i>Ide dec-hi</i>	0.218	120.8%	—	—
AG com $F1_{L-Pujalte et al.}$	0.218	120.8%	0.220	123.6%
AG com $F2_{L-Pujalte et al.}$	0.218	120.8%	0.210	114.2%
AG com $F3_{L-Pujalte et al.}$	0.218	120.8%	0.224	127.2%

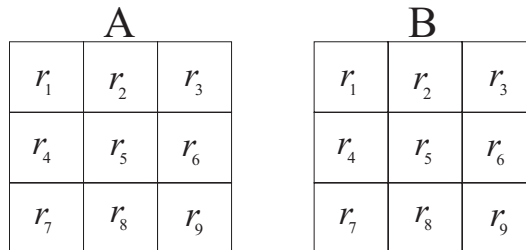
Tabela 3.3: Resultados obtidos por [López-Pujalte et al. 2003b].

3.4 Stejić et al.

Os trabalhos de Stejić et al. descritos nesta seção são relacionados à recuperação de imagens através do modelo de similaridade de padrões locais (*Local Similarity Pattern – LSP*). O modelo de similaridade de padrões locais (*LSP*) baseia-se na comparação de regiões correspondentes para obter o grau de similaridade final entre imagens. Em todos os trabalhos dos autores, descritos nesta seção, as imagens são divididas em regiões retangulares uniformes e comparadas através de suas regiões correspondentes. Por exemplo, supondo que as imagens sejam divididas em 9 regiões retangulares uniformes (veja Figura 3.3), a similaridade entre as imagens A e B será dada por:

$$Sim(A, B) = \frac{\sum_{i=1}^9 Sim(r_i(A), r_i(B))}{9} \quad (3.10)$$

As características utilizadas para a comparação de imagens foram as de cor, forma e textura, descritas por Momentos de Cor, Histograma de Direção de Bordas e Vizinhança de Textura, respectivamente. Estes descritores foram apresentados no Capítulo 2.

Figura 3.3: Exemplificando o modelo *LSP*.

3.4.1 Revisão e Problemas

A partir de 2002, Stejić, Takama e Hirota começaram a desenvolver sistemas de recuperação de imagens usando múltiplas características e regiões de imagens (modelo *LSP*), e com realimentação de relevância via algoritmos genéticos. O primeiro sistema criado pelo autores [Stejić et al. 2002] tinha como meta usar a realimentação do usuário para obter ótimas combinações de características para regiões de imagem. A similaridade de uma dada região era

comparada usando características de: cor, ou forma, ou textura, ou cor+forma, ou cor+textura ou forma+textura. A combinação de características usadas no modelo de similaridade entre imagens era inferida por um AG de forma que a precisão de recuperação fosse maximizada. Depois disso, Stejić et. al. propôs em 2003 o modelo de similaridade *WLSP* (*Weighted Local Similarity Pattern*) [Stejić et al. 2003b], como uma extensão do modelo *LSP*. No modelo *WLSP* além da combinação de características, agora com pesos discretos, eram também considerados pesos (discretos) de regiões. Com base nos modelos *LSP* e *WLSP*, ainda em 2003, Stejić et. al. desenvolveram duas novas extensões [Stejić et al. 2003a]: (1) os modelos *LSP-C+*/*WLSP-C+*, com pesos contínuos não-negativos e (2) os modelos *LSP-C±*/*WLSP-C±* com pesos contínuos positivos e negativos.

Em [Stejić et al. 2005] foram testados um total de 67 operadores de agregação para compor a medida de similaridade entre imagens e desenvolveu um AG para a escolha dos operadores de agregação que melhor ajustassem o modelo de similaridade ao critério do usuário manifestado durante a realimentação. Apesar de terem mostrado que a escolha de operadores de agregação tem uma grande importância na modelagem do critério de similaridade do usuário, a tentativa de adaptação dos operadores de agregação através de AG obteve resultados inferiores àqueles das extensões dos modelos *LSP*/*WLSP* [Stejić et al. 2003a]. Em 2004, [Stejić et al. 2004] fez uma tentativa de usar um algoritmo genético hierárquico para adaptar ambos, os operadores de agregação e os pesos de regiões e de características, contudo, novamente os resultados obtidos foram inferiores aos dos modelos *LSP-C+*/*WLSP-C+* e *LSP-C±*/*WLSP-C±* [Stejić et al. 2003a].

Dado que os modelos *LSP-C±*/*WLSP-C±* [Stejić et al. 2003a] foram os que obtiveram os melhores resultados, a partir deste instante voltamos nossa atenção exclusivamente para estes, descrevendo em detalhes a técnica e os resultados obtidos.

O modelo de similaridade de imagens utilizado é definido pelas equações 3.11 e 3.12. A equação 3.11 representa uma soma ponderada de similaridades de regiões sendo estas obtidas por uma soma ponderada das similaridades de cor, forma e textura (equação 3.12). Nestas equações q denota a imagem consulta, i uma dada imagem do banco de dados, R o conjunto de regiões consideradas e $F = \{cor, forma, textura\}$.

$$\underbrace{S_I(q, i)}_{sim. \text{ imagem}} = \sum_{r \in R} \underbrace{\omega_R(q, i, r)}_{\text{peso regioao}} \underbrace{S_R(q, i, r)}_{sim. \text{ regioao}}, \text{ tal que } \sum_{r \in R} |\omega_R(q, i, r)| = 1 \quad (3.11)$$

onde,

$$\underbrace{S_R(q, i, r)}_{sim. \text{ regioao}} = \sum_{f \in F} \underbrace{\omega_F(q, i, r, f)}_{\text{peso caract.}} \underbrace{S_F(q, i, r, f)}_{sim. \text{ caract.}}, \text{ tal que } \sum_{f \in F} |\omega_F(q, i, r, f)| = 1 \quad (3.12)$$

Nos modelos de similaridade *LSP-C±*/*WLSP-C±* as funções pesos, de características e de regiões têm intervalos contínuos onde $\omega_F^{LSP-C\pm/WLSP-C\pm} \rightarrow [-1, 1]$, $\omega_R^{LSP-C\pm} \rightarrow \{1\}$, e $\omega_R^{WLSP-C\pm} \rightarrow [-1, 1]$. Segundo os autores, a introdução de pesos negativos permite ao modelo expressar o fato de uma certa similaridade ser mais ou menos indesejável, em adição ao nível de relevância expresso por pesos positivos e a irrelevância expressa por um peso nulo.

Os autores adotam a seguinte normalização para obter valores de similaridade de imagem

e de regiões no intervalo $[0, 1]$:

$$\begin{aligned}\overbrace{S_R(q, i, r)}^{[0,1]} &= \frac{1}{2}(\overbrace{S_R^*(q, i, r)}^{[-1,1]} + 1), \text{ e} \\ \overbrace{S_I(q, i)}^{[0,1]} &= \frac{1}{2}(\overbrace{S_I^*(q, i)}^{[-1,1]} + 1)\end{aligned}\quad (3.13)$$

Nos modelos de [Stejić et al. 2003a] as similaridades de características dependem exclusivamente da consulta e da coleção de referência. Porém, as importâncias das características podem variar com a necessidade de busca do usuário. Assim, [Stejić et al. 2003a] propôs a adaptação do modelo de similaridade através da inferência de pesos de características. Estes pesos têm a função de representar os critérios do usuário expressos na realimentação de relevância.

3.4.2 Especificação do Algoritmo Genético

Para a tarefa de inferência de pesos, [Stejić et al. 2003a] propuseram o seguinte algoritmo genético:

- **Codificação de Indivíduo** - um indivíduo representa a área da imagem consulta uniformemente particionada em n_R regiões. Cada gene de um indivíduo corresponde a uma região, sendo mapeada por um conjunto de quatro pesos: de cor, forma, textura e o peso da região. Estes pesos simulam uma tradução dos critérios do usuário na busca por imagens. Como neste trabalho considera-se 16 regiões, cada indivíduo contém 16 genes.
- **Medida de Aptidão** - é dada pela medida de precisão- R de recuperação. Para facilidade de nomenclatura denominamos esta função como $F_{Stejić et al.}$.
- **Operadores Genéticos**
 - *Crossover* - foram experimentados *crossover* uniforme, simples e múltiplo (de dois pontos).
 - *Mutação* - foram experimentados sete operadores de mutação. Estes são ilustrados pela tabela 3.4.2.

3.4.3 Detalhes Adicionais da Técnica de Stejić et al.

Distâncias entre pares de vetores de características são computadas por distância Euclidiana para momentos de cor, e distância em Quadras para histograma de direção de bordas e vizinhança de textura. Similaridades de características S_F são dadas por $S_F = 1 - D_F$, onde $D_F \in [0, 1]$ denota a distância normalizada entre um par de vetores de características.

Dada uma seqüência de valores de similaridade entre a imagem consulta q e todas as imagens do banco de dados i_K ($k = 1, \dots, n_I$) com respeito a região r e a característica f – $S_F(q, r, f) = [S_F(q, i_1, r, f), S_F(q, i_2, r, f), \dots, S_F(q, i_{n_I}, r, f)]$ – e tendo computado a média

Tipo de Mutação	Operação
Uniforme	$\omega' = \text{random}(-1 + \epsilon, 1 - \epsilon)$
Inversa	$\omega' = \text{rndsgn}() \times (1/\omega)$
Complemento	$\omega' = \text{rndsgn}() \times (1 - \omega)$
Complemento do quadrado	$\omega' = \text{rndsgn}() \times (1 - \omega^2)$
Inversa Combinada	$\omega' = \begin{cases} \text{rndsgn}() \times (1/\omega) & \text{com prob. 0.8} \\ \text{random}(-1 + \epsilon, 1 - \epsilon) & \text{com prob. 0.2} \end{cases}$
Complemento Combinado	$\omega' = \begin{cases} \text{rndsgn}() \times (1 - \omega) & \text{com prob. 0.8} \\ \text{random}(-1 + \epsilon, 1 - \epsilon) & \text{com prob. 0.2} \end{cases}$
Complemento do quadrado combinado	$\omega' = \begin{cases} \text{rndsgn}() \times (1 - \omega^2) & \text{com prob. 0.8} \\ \text{random}(-1 + \epsilon, 1 - \epsilon) & \text{com prob. 0.2} \end{cases}$

Tabela 3.4: Mutações experimentadas por [Stejić et al. 2003a]. O valor de ϵ é 10^{-6} . A função $\text{random}(l, h)$, onde $l, h \in \mathbb{R} : l < h$, retorna um número randômico no intervalo $[l, h]$. A função $\text{rndsgn}()$ significa “sinal randômico”, retornando ou $+1$ ou -1 com a mesma probabilidade.

$\mu(q, r, f)$ e o desvio padrão $\sigma(q, r, f)$ desta seqüência, a seqüência normalizada é computada por transformar cada elemento da seguinte forma:

$$\underbrace{S'_F(q, i_k, r, f)}_{[-1,1]} = \frac{S_F(q, i_k, r, f) - \mu(q, r, f)}{4\sigma(q, r, f)} \quad (3.14)$$

e

$$\underbrace{S''_F(q, i_k, r, f)}_{[0,1]} = \frac{S'_F(q, i_k, r, f) + 1}{2} \quad (3.15)$$

Possíveis valores de S''_F fora do intervalo $[0,1]$ são simplesmente mapeados para o limite do intervalo mais próximo.

3.4.4 Avaliação Experimental

Stejić, Takama e Hirota avaliaram seus sistemas em cinco bancos de dados de imagens: *Vistex-60*, *Vistex-167*, *Brodatz-207*, *Corel-1000-A* e *Corel-1000-B*. Como medidas de performance foram utilizadas a precisão de recuperação e o tempo de computação. A tabela 3.5 mostra a precisão de recuperação média obtida pelos métodos *LSP-C $\pm$* e *WLSP-C $\pm$* .

Modelo	Precisão Média				
<i>LSP-C$\pm$</i>	98	86	96	76	68
<i>WLSP-C$\pm$</i>	95	84	95	75	66
<i>Banco de Dados</i>	<i>V-60</i>	<i>V-167</i>	<i>B-208</i>	<i>C-1000-A</i>	<i>C-1000-B</i>

Tabela 3.5: Precisão de recuperação dos melhores métodos de [Stejić et al. 2003a]: *LSP-C $\pm$* e *WLSP-C $\pm$*

Com relação ao tempo de computação os autores apresentam apenas o tempo de reali-

mentação para uma simples consulta no banco de dados *Corel-1000-A*, onde o modelo *LSP-C±* gasta 61.2 s e o *WLSP-C±* gasta 63.5 s. Infelizmente os autores não mencionam o sistema (*hardware+software*) utilizado.

3.5 Torres et al.

3.5.1 O Problema

Torres, Falcão, Gonçalves, Zhang, Fan e Fox propuseram em [Torres et al. 2005], o uso de Programação Genética (PG) na geração de funções destinadas a combinar descritores de imagens. Este trabalho pode ser visto como uma adaptação do trabalho de [Fan et al. 2004] para o contexto de recuperação de imagens por conteúdo.

As funções de combinação de descritores geradas pela técnica de PG são representadas por árvores, cujos nós não-folha são dados pelos operadores numéricos ($+$, $*$, $/$, $\sqrt{}$) e os nós folha são dados pelos valores de similaridades $d_i, i = 1, \dots, k$, retornados pelos descritores de imagem. A Figura 3.4 mostra uma possível combinação dos valores de similaridade de três descritores (d_1, d_2 e d_3).

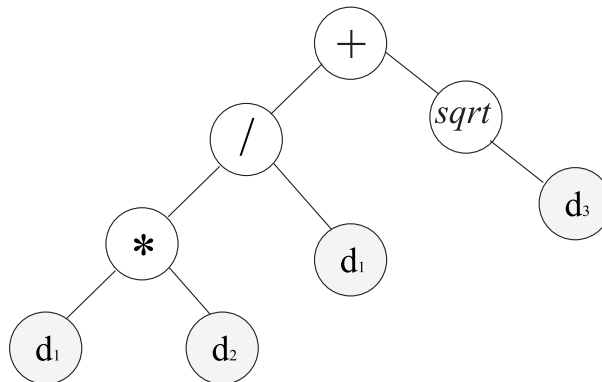


Figura 3.4: Exemplo de uma possível função de combinação dos descritores d_1, d_2 e d_3 , representada por uma árvore.

3.5.2 A Solução via Programação Genética

Como já descrevemos os conceitos básicos de Programação Genética na subseção 3.2.2, iremos diretamente à solução proposta por [Torres et al. 2005] para a combinação de descritores. A solução geral é descrita pelo Algoritmo 3.2.

A população inicial de árvores foi restringida para ter uma profundidade máxima de quatro níveis e gerada pelo método *ramped half-and-half*, que estipula que metade das árvores geradas randomicamente têm todas as ramificações, respeitando a profundidade máxima estabelecida. Na geração da outra metade das árvores considera-se apenas a restrição de profundidade máxima. Os autores não mencionam o tamanho da população inicial de árvores e nem o valor de N_{top} .

-
- 1 Gerar uma população inicial de árvores
 - 2 Executar os seguintes sub-passos sobre um conjunto de documentos de treinamento por N_g gerações:
 - 2.1 Usar cada árvore como função de combinação para múltiplas consultas
 - 2.2 Calcular a aptidão de cada árvore
 - 2.3 Registrar as N_{top} melhores árvores
 - 2.4 Criar uma nova população por:
 - a) Elitismo
 - b) Seleção
 - c) *Crossover*
 - d) Mutação
 - 3 Aplicar a “melhor árvore” sobre o conjunto de imagens de teste.
-

Algoritmo 3.2: Funcionamento global do sistema PG para descoberta de funções de combinação de descritores [Torres et al. 2005]

Adicionalmente os autores utilizaram: seleção por torneio de tamanho 6, a operação de *crossover* consiste de emparelhar duas árvores e trocar randomicamente suas subárvores, a operação de mutação consiste em “deletar” randomicamente uma subárvore e substituir esta por uma outra subárvore gerada aleatoriamente. A probabilidade de mutação considerada foi de 0.25. O critério de párada é 50 gerações.

3.5.3 Avaliação Experimental

Embora a técnica proposta tenha sido especificada de modo genérico, permitindo a combinação de quaisquer descritores (de cor, forma, textura, etc.), os experimentos relatados foram realizados para descritores baseados em formas. Os descritores utilizados foram: *Beam Angle Statistics (BAS)* com 40 amostras, *BAS* com 60 amostras, *Moments Invariants*, *Fourier Descriptors* e *Contour Multiscale Fractal Dimension*. A proposta foi validada pela medida de precisão entre as 10 primeiras imagens retornadas, onde foi utilizado dois bancos de dados: *Fish Shape Database* (1000 imagens, agrupadas em 100 classes de 10 imagens cada, onde as imagens de uma classe consistem de alterações de escala e rotação aplicadas à imagem original) e *MPEG-7 Part B Database* (1400 imagens agrupadas em 70 classes de 20 imagens cada). Os bancos de dados foram divididos randomicamente pela metade em conjunto de treinamento e conjunto de teste.

A tabela 3.6 mostra os resultados obtidos por [Torres et al. 2005] no conjunto de treinamento e no conjunto de teste, comparados aos resultados obtidos pelo uso do descritor *BAS* com 60 amostras. O tempo de treinamento para o banco de dados *Fish Shapes* é em média de 30 minutos e para o banco de dados *MPEG-7 Part B* de 40 minutos. Os testes foram executados em um processador *Pentium 4 3.2GHz* com 2G de *RAM*.

Descritor	<i>MPEG-7</i> Precisão-10		<i>Fish Shapes</i> Precisão-10	
	Treinamento	Teste	Treinamento	Teste
<i>BAS60</i>	66.27	64.84	93.25	92.30
GP com F_{PAVG}	70.56 (6.47%)	69.21 (6.74 %)	93.75 (0.54 %)	92.75 (0.49 %)
GP com $F1_{Fan et al.}$	70.92 (7.02%)	69.59 (7.33 %)	94.20 (1.02 %)	93.30 (1.08 %)
GP com $F2_{Fan et al.}$	70.79 (6.82%)	69.76 (7.59 %)	94.30 (1.13 %)	93.35 (1.14 %)
GP com $F3_{Fan et al.}$	70.75 (6.76%)	69.44 (7.09 %)	94.05 (0.86 %)	93.30 (1.08 %)
GP com $F4_{Fan et al.}$	70.40 (6.23%)	68.97 (6.37 %)	94.05 (0.86 %)	93.30 (1.08 %)
GP com $F1_{L-Pujalte et al.}$	70.73 (6.73%)	66.78 (2.99 %)	94.20 (1.02 %)	93.30 (1.08 %)
GP com $F2_{L-Pujalte et al.}$	70.86 (6.93%)	70.90 (9.35 %)	94.15 (0.97 %)	93.20 (0.98 %)

Tabela 3.6: Resultados obtidos por [Torres et al. 2005].

3.6 Considerações Finais

Neste capítulo foram apresentados quatro trabalhos relacionados ao uso de técnicas de computação evolutiva em mecanismos de realimentação de relevância. Pelos resultados reportados nos trabalhos revisados, podemos perceber que as técnicas de computação evolutiva são alternativas altamente promissoras para a realimentação de relevância.

Parte II

Nossa Contribuição

O Sistema Proposto

Neste Capítulo é apresentado o sistema *CBIR* proposto. Inicialmente é dada uma visão geral do sistema proposto, seus módulos principais e seu funcionamento. Posteriormente inicia-se a apresentação dos componentes do sistema: os descritores, as medidas de similaridade, o modelo de similaridade e a realimentação de relevância via algoritmo genético.

O principal enfoque deste trabalho é o mecanismo de realimentação de relevância via algoritmo genético. Como contribuições podemos destacar: a elaboração de um algoritmo genético sob-medida para o problema de realimentação de relevância, proposta de duas funções de aptidão e a comparação de performance de dez funções de aptidão aplicadas ao problema. Adicionalmente estas funções são classificadas em dois grupos conforme seus princípios de construção: baseadas em ordem e não baseadas em ordem. Para comparar as diferenças nos princípios de desenvolvimento, comparamos estas matematicamente, supondo algumas situações de *ranking*.

4.1 Estrutura Geral

A Figura 4.1 mostra a estrutura geral do sistema *CBIR* desenvolvido. Inicialmente considera-se um banco de dados de imagens onde são armazenadas todas as imagens do universo de busca do sistema. O banco de dados de imagens alimenta o módulo de extração de características, que produz como saída estruturas contendo o código de identificação e os vetores de características de cada imagem do banco de dados. Estes dados (código de identificação + características) são armazenados no banco de dados de características. Este processo de caracterização das imagens do banco de dados, ilustrado pela setas pontilhadas, é feito *off-line*.

Quando o usuário faz uma busca (*on-line*), vetores de características são extraídos da imagem de consulta pelo módulo de extração de características e comparados através de medidas de similaridade/distância às características das imagens do banco de dados, armazenadas no banco de dados de características. O módulo medidor de similaridade retorna um valor de similaridade para cada imagem do banco de dados, em relação à imagem de consulta. Com base nestes valores as imagens são postas em ordem decrescente de similaridade (*ranking*) e mostradas ao usuário. Não satisfeito com o resultado da busca, o usuário pode realimentar o sistema, indicando as imagens relevantes para sua tarefa de recuperação, conforme seu ponto de vista.

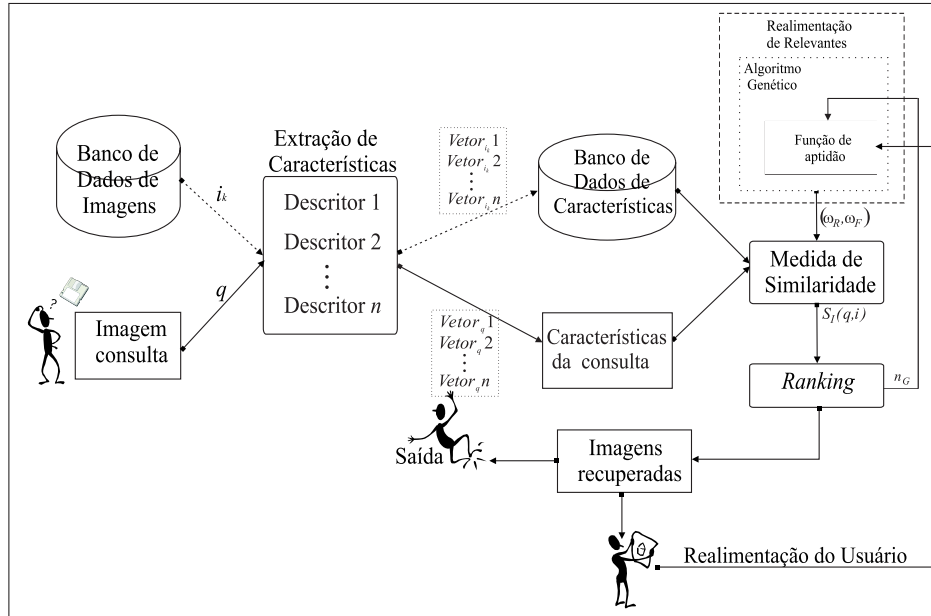


Figura 4.1: Fluxograma básico do sistema *CBIR* proposto.

Com base na realimentação do usuário, o mecanismo de realimentação de relevância irá ajustar a medida de similaridade aos critérios do usuário, através de pesos de regiões e de características. O mecanismo de realimentação de relevantes tem como base um algoritmo genético de código real, projetado para o problema de inferência de pesos, de modo a maximizar a eficiência de recuperação de acordo com os requisitos do usuário.

4.2 Descritores de Características

Nesta seção serão apresentados três descritores, de cor, forma e textura, utilizados na caracterização das imagens, assim como as métricas utilizadas para a comparação dos seus correspondentes vetores de características.

4.2.1 Momentos de Cor

Momentos de cor [Stricker and Orengo 1995] caracterizam as imagens em termos da distribuição das cores. As imagens são convertidas do espaço de cor *RGB* (*Red, Green and Blue*) para o espaço de cor *HSI* (*Hue, Saturation and Intensity*) e cada canal de cor é interpretado por três medidas estatísticas.

Sendo N o número de *pixels* da imagem e p_{ij} o valor do i -ésimo canal de cor para o j -ésimo *pixel* da imagem, média, desvio padrão e inclinação são computados através das expressões 4.1, 4.2 e 4.3, respectivamente.

$$E_i = \frac{1}{N} \sum_{j=1}^N p_{ij} \quad (4.1)$$

$$\sigma_i = \sqrt{\left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - E_i)^2\right)} \quad (4.2)$$

$$s_i = \sqrt[3]{\left(\frac{1}{N} \sum_{j=1}^N (p_{ij} - E_i)^3\right)} \quad (4.3)$$

Distâncias entre pares de descritores de cor $(E_H, \sigma_H, s_H, E_S, \sigma_S, s_S, E_I, \sigma_I, s_I)$ são computadas por Distância Euclidiana (equação 2.24).

O Sistema de Cor *RGB*

O sistema de cor *RGB* é definido por um espaço tridimensional cujas componentes representam as intensidades de vermelho (*Red*), verde (*Green*) e azul (*Blue*) que compõem uma dada cor. O sistema de cor RGB é bastante utilizado por dispositivos tais como *scanners* e monitores coloridos. Qualquer cor no espaço *RGB* pode ser vista como uma mistura das três cores primárias: vermelho, verde e azul. Por exemplo, a cor ciano é dada pela combinação das cores verde e azul. A Figura 4.2 ilustra o espaço de cor *RGB* através de um cubo. As cores vermelho, verde e azul são representadas pelas arestas principais do cubo. Uma posição tomada ao longo deste cubo corresponde a cor formada pela mistura das respectivas intensidades de vermelho, verde e azul.

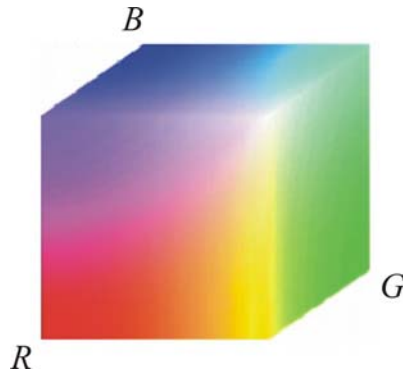


Figura 4.2: Sistema de cor *RGB*.

A Figura 4.3 mostra uma imagem colorida e a visualização de seus canais de cor *R*, *G* e *B*.

O Sistema de Cor *HSI*

O sistema de cor *HSI* é definido pelos canais de cor *Hue*, *Saturation* e *Intensity*. *Hue* (matiz) é um atributo associado ao comprimento de onda dominante que representa uma cor pura como vermelho, laranja, rosa, azul, etc. *Saturation* (saturação) está relacionada à quantidade de luz branca misturada à matiz. Quanto menor a quantidade de luz branca, maior é a saturação. *Intensity* (intensidade) é desacoplada das informações de cor, correspondendo a uma representação da imagem em níveis de cinza [Gonzalez and Woods 2002].



Figura 4.3: (a) Imagem original; (b), (c) e (d) visualização dos canais R , G e B , respectivamente.

A Figura 4.4 ilustra o espaço de cor HSI . A Figura 4.5 mostra uma imagem colorida e a visualização de seus canais de cor H , S e I .

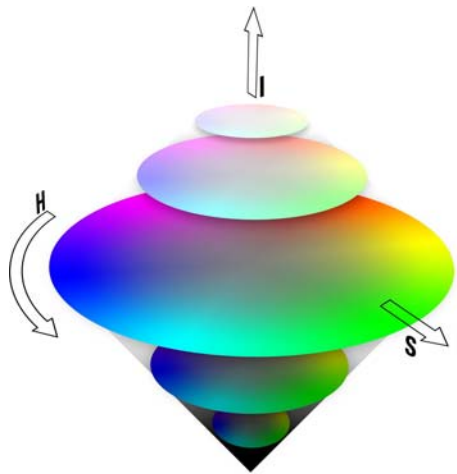


Figura 4.4: Sistema de cor HSI .



Figura 4.5: (a) Imagem original; (b), (c) e (d) visualização dos canais H , S e I , respectivamente.

Conversão RGB/HSI

As expressões 4.4, 4.5 e 4.6 são utilizadas para a conversão do sistema de cor RGB para o HSI. Elas obtêm valores para as componentes H , S e I no intervalo $[0, 1]$ a partir dos valores de R , G e B no mesmo intervalo.

$$I = \frac{1}{3}(R + G + B) \quad (4.4)$$

$$S = 1 - \frac{3}{(R + G + B)} \min(R, G, B) \quad (4.5)$$

$$H = \cos^{-1} \left[\frac{\frac{1}{2}[(R - G) + (R - B)]}{\sqrt{[(R - G)^2 + (R - B)(G - B)]}} \right] \quad (4.6)$$

onde $H = 2\pi - H$ se $\frac{B}{I} > \frac{G}{I}$. Para normalizar H no intervalo $[0, 1]$, calcula-se $H = \frac{H}{2\pi}$.

4.2.2 Histograma de Direção de Bordas

O histograma de direção de bordas [Brandt 1999] modela similaridade entre imagens em termos da similaridade da distribuição das direções de bordas. As imagens são convertidas para o espaço de cor *HSI* (*Hue, Saturation and Intensity*), onde somente os canais S e I são usados, na identificação das direções de bordas. A razão do descarte do canal H (a matiz) se dá pelo fato deste negligenciar bordas importantes em regiões ou objetos com a mesma tonalidade de cor. Tal fato pode ser observado através do experimento da Figura 4.6. Na Figura 4.6-(a) temos uma imagem original e em (b), (c) e (d) a visualização da resposta das máscaras de sobel para os canais H , S , I , respectivamente. O valor de limiar de todos os gradientes foi fixado como sendo 15% da resposta máxima.

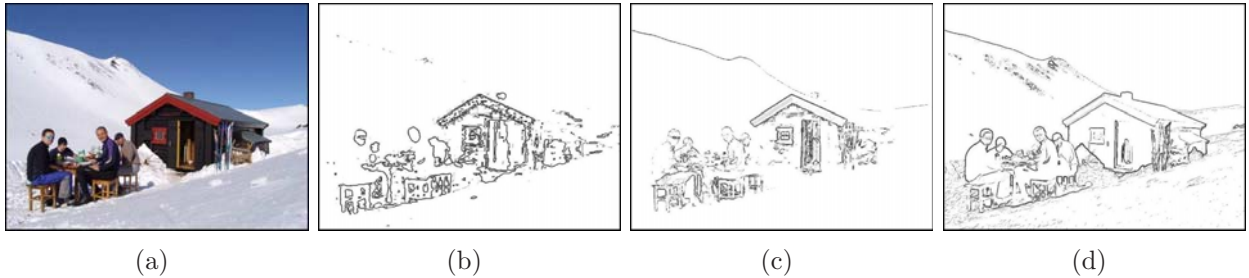


Figura 4.6: Imagem original e as imagens do gradiente da matiz, saturação e intensidade, respectivamente.

Operadores de Sobel de oito direções (ver Figura 4.7) são aplicados sobre os dois canais e é definido um descritor representando as oito direções. Cada *pixel*, de cada canal de cor, é analisado segundo as 8 máscaras de Sobel e o maior valor obtido define a direção da borda naquele *pixel* caso o valor para alguma máscara ultrapassar 35% ou 15% para os canais S e I , em relação à resposta máxima das máscaras nos respectivos canais. Cada vez que a condição acima for atendida o contador correspondente àquela direção é incrementado. Finalmente o histograma de direção de bordas é normalizado pelo número de *pixels* de borda. Distâncias entre pares de histogramas de direção de bordas ($H_0^o, H_{45}^o, H_{90}^o, H_{135}^o, H_{180}^o, H_{225}^o, H_{270}^o, H_{315}^o$) são computadas através da Distância em Quadras (equação 2.25).

<table><tr><td>-1</td><td>0</td><td>1</td></tr><tr><td>-2</td><td>0</td><td>2</td></tr><tr><td>-1</td><td>0</td><td>1</td></tr></table> 0°	-1	0	1	-2	0	2	-1	0	1	<table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>-1</td><td>0</td><td>1</td></tr><tr><td>-2</td><td>-1</td><td>0</td></tr></table> 45°	0	1	2	-1	0	1	-2	-1	0	<table><tr><td>1</td><td>2</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>-1</td><td>-2</td><td>-1</td></tr></table> 90°	1	2	1	0	0	0	-1	-2	-1	<table><tr><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>-1</td></tr><tr><td>0</td><td>-1</td><td>-2</td></tr></table> 135°	2	1	0	1	0	-1	0	-1	-2
-1	0	1																																					
-2	0	2																																					
-1	0	1																																					
0	1	2																																					
-1	0	1																																					
-2	-1	0																																					
1	2	1																																					
0	0	0																																					
-1	-2	-1																																					
2	1	0																																					
1	0	-1																																					
0	-1	-2																																					
<table><tr><td>1</td><td>0</td><td>-1</td></tr><tr><td>2</td><td>0</td><td>-2</td></tr><tr><td>1</td><td>0</td><td>-1</td></tr></table> 180°	1	0	-1	2	0	-2	1	0	-1	<table><tr><td>0</td><td>-1</td><td>-2</td></tr><tr><td>1</td><td>0</td><td>-1</td></tr><tr><td>2</td><td>1</td><td>0</td></tr></table> 225°	0	-1	-2	1	0	-1	2	1	0	<table><tr><td>-1</td><td>-2</td><td>-1</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>2</td><td>1</td></tr></table> 270°	-1	-2	-1	0	0	0	1	2	1	<table><tr><td>-2</td><td>-1</td><td>0</td></tr><tr><td>-1</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>2</td></tr></table> 315°	-2	-1	0	-1	0	1	0	1	2
1	0	-1																																					
2	0	-2																																					
1	0	-1																																					
0	-1	-2																																					
1	0	-1																																					
2	1	0																																					
-1	-2	-1																																					
0	0	0																																					
1	2	1																																					
-2	-1	0																																					
-1	0	1																																					
0	1	2																																					

Figura 4.7: Operadores de Sobel.

4.2.3 Vizinhança de Textura

No método de vizinhança de textura [Laaksonen 2000] a imagem é representada em níveis de cinza. Um descritor de dimensão oito é gerado da seguinte forma: se o valor de um *pixel* na vizinhança de oito for maior que o *pixel* central então, o contador direcional indicando a respectiva direção é incrementado. Isto é feito para todos os *pixels* na imagem, e em seguida o descritor é normalizado. Distâncias entre pares de descritores de vizinhança de textura, $(V_0^o, V_{45}^o, V_{90}^o, V_{135}^o, V_{180}^o, V_{225}^o, V_{270}^o, V_{315}^o)$, são computadas pela Distância em Quadras.

4.3 O Modelo de Similaridade entre Imagens

O modelo de similaridade entre imagens é baseado na abordagem de padrões locais conforme a proposta de [Stejić et al. 2003a]. As imagens são divididas em regiões uniformes¹ (veja Figura 4.8) e a similaridade entre imagens é medida através da comparação de regiões correspondentes.

r_1	r_2	r_3	r_4
r_5	r_6	r_7	r_8
r_9	r_{10}	r_{11}	r_{12}
r_{13}	r_{14}	r_{15}	r_{16}

Figura 4.8: Ilustração da divisão de uma imagem em regiões, i.e., $i = \bigcup r_j, j = 1, \dots, 16$.

Seja q uma imagem de consulta, i uma dada imagem do banco de dados, r uma região de imagem, f uma característica, I o banco de dados de imagem, R o conjunto de regiões de imagem e $F = \{cor, forma, textura\}$.

¹Dada uma imagem i de $N \times M$ *pixels*, dividimos esta em 16 regiões não sobrepostas de $(N/4) \times (M/4)$ *pixels* onde a união de todas as regiões dá a imagem original i

A função similaridade de características, $S_F(q, i, r, f)$, mede o grau de similaridade entre um par de imagens q e i , com relação à característica f da região r e é dada pela Eq.4.7:

$$\underbrace{S_F(q, i, r, f)}_{\text{sim. caract.}} = 1 - \underbrace{D_F(q, i, r, f)}_{\text{dist. caract.}} \quad (4.7)$$

onde $D_F(q, i, r, f) \in [0, 1]$ é a distância² normalizada entre um par de vetores de características.

A função peso de característica $\omega_F(r, f)$ pondera com valores reais no intervalo $[-1, 1]$ a importância da característica f da região r , de acordo com o critério do usuário, definido pela imagem de consulta e o conjunto de imagens relevantes. Baseado nas funções similaridade de características e peso de características, a função similaridade de região $S_R(q, i, r)$ encontra o grau de similaridade de região entre q e i com relação à região r . Complementarmente a função peso de região $\omega_R(r)$ pondera com valores reais, no intervalo $[-1, 1]$, a importância da região r conforme os requisitos do usuário. Finalmente, baseado nas funções similaridade de região e peso de região, a função similaridade de imagem $S_I(q, i)$, equação 4.8, retorna o grau de similaridade entre as imagens q e i .

$$\underbrace{S_I(q, i)}_{\text{sim. imagem}} = \sum_{r \in R} \left(\underbrace{\omega_R(r)}_{\text{peso regioao}} \times \underbrace{\sum_{f \in F} \left(\underbrace{\omega_F(r, f)}_{\text{peso caract.}} \underbrace{S_F(q, i, r, f)}_{\text{sim. caract.}} \right)}_{\text{sim. regioao.} = S_R(q, i; r)} \right) \quad (4.8)$$

Ao contrário do modelo de [Stejić et al. 2003a] que tem as restrições: $\sum_{f \in F} |\omega_F(r, f)| = 1$ e $\sum_{r \in R} |\omega_R(r)| = 1$, no modelo apresentado as únicas restrições é que os valores de $\omega_F(r, f)$ e de $\omega_R(r)$ estejam no intervalo $[-1, 1]$. Esta simplificação elimina a necessidade de um operador de normalização no algoritmo genético.

Como no modelo apresentado o conjunto de regiões e o conjunto de características são fixos, a função similaridade de característica produzirá valores de similaridade fixos para uma dada coleção de imagens I . Contudo, o que faz o modelo flexível às necessidades de recuperação de informação do usuário são as funções peso de características e de regiões, que têm como meta encontrar um conjunto de pesos onde, dados a imagem de consulta e as imagens relevantes, aumente a eficiência de recuperação do ponto de vista do usuário. A introdução de pesos contínuos positivos e negativos [Stejić et al. 2003a], permite ao usuário expressar os conceitos de relevância, irrelevância e indesejabilidade em vários graus.

Embora o modelo apresentado seja aparentemente simples, ele é mais complexo do que parece ser à primeira vista. A razão de sua complexidade é a dificuldade de estabelecer uma correspondência óbvia entre as características de baixo nível e os conceitos de alto nível dos usuários – gap-semântico [Crucianu et al. 2004; Stejić et al. 2003a]. Assim o tratamento do gap-semântico é o ponto chave para o sucesso do processo de recuperação e aqui este foi alcançado graças ao processo de adaptação da medida de similaridade entre imagens via Algoritmo Genético.

²Distância entre vetores de características de cor é computada por distância Euclidiana e entre vetores de características de forma e textura por distância em quadras.

4.3.1 Realimentação de Relevância Baseada em AGs

Realimentação de relevância é uma aplicação natural de algoritmos genéticos devido a sua força de adaptação onde há alterações ou diferentes formas de contextualização do ambiente. Atualmente há vários trabalhos relacionados aos uso de algoritmos genéticos em mecanismos de realimentação de relevância como [Cordón et al. 2003], [López-Pujalte et al. 2003b] e [Stejić et al. 2003a].

Os trabalhos relacionados a aplicação de AG em mecanismos de realimentação de relevância são em sua maioria direcionados ao ajuste de parâmetros (pesos, variáveis) de medidas de similaridade que levam a uma recuperação mais eficiente. Esta tarefa normalmente parte de um conjunto de possíveis soluções que evoluem através de operações genéticas e de seleção por várias gerações na tentativa de encontrar uma solução otimizada conforme os requerimentos do usuário. Para guiar esta evolução na busca de possíveis soluções, a questão chave é projetar uma medida de aptidão que avalia quão boa estas soluções são, conforme a realimentação do usuário.

Neste trabalho o processo de realimentação usando o modelo de similaridade de imagem (equação 4.8) é modelado como um problema de otimização de inferência de pesos ($\omega_F(r, f)$ e $\omega_R(f)$) tal que $r = 1, 2, \dots, 16$ e $f = \{cor, forma, textura\}$. Dado que um sistema gera em resposta a uma dada consulta e uma atribuição de pesos C (indivíduo), um *ranking* com todas as imagens do banco de dados, a medida da qualidade deste *ranking* comparada à realimentação do usuário será o alvo de otimização do AG. Ou seja, a realimentação do usuário é o ponto de referência do AG para inferir os pesos que ajustarão a medida de similaridade aos critérios do usuário.

4.4 O Algoritmo Genético

Nós usamos um AG projetado e ajustado exclusivamente para o problema de realimentação de relevância. Inicialmente serão descritas as características deste AG e em seguida as funções de aptidão usadas para guiar o algoritmo no processo de busca.

4.4.1 A Codificação de Indivíduo

Para aplicar um AG para um dado problema, a primeira decisão a ser feita é a escolha de qual genótipo o problema necessita, isto é, qual a representação de indivíduo. Ou seja, uma decisão deve ser feita com relação ao modo em que os parâmetros do problema serão mapeados em uma *string* finita de símbolos (genes), definida a partir de um dado alfabeto, codificando as possíveis soluções em um dado espaço de problema. Neste trabalho, um indivíduo representa a área da imagem de consulta uniformemente particionada em n_R regiões. Cada gene (G_i) de um indivíduo corresponde a uma região de imagem, sendo mapeado por (1) uma sequência de pesos de características atribuídos às características daquela região e (2) o peso de região, expressando suas importâncias na medida de similaridade:

$$G_i \equiv \left(\omega_F(r_i, cor), \omega_F(r_i, forma), \omega_F(r_i, textura), \omega_R(r_i) \right).$$

Como neste trabalho cada imagem é particionada em 16 regiões, cada indivíduo (C) conterà 16 genes:

$$C = (G_1, G_2, G_3, \dots, G_{16}) \quad (4.9)$$

4.4.2 Os Operadores Genéticos

Nesta subseção, as operações de seleção, *crossover* e mutação usadas neste trabalho são apresentadas. A seleção implementa a sobrevivência dos indivíduos mais aptos de acordo com uma medida de aptidão pré-definida. Assim, os indivíduos de maiores aptidões têm maiores chances de sobreviverem e reproduzirem, enquanto que aqueles de baixa aptidão têm maiores chances de desaparecerem. As operações de *crossover* e mutação representa uma analogia à reprodução natural e têm como meta na prática, explorar o espaço de soluções.

Seleção

- *Seleção para recombinação* – é aplicada para selecionar os indivíduos que irão reproduzir. Foi utilizado seleção proporcional ao valor de aptidão pelo método da roleta. Tal procedimento de seleção pode ser revisto na seção 1.4.4.
- *Seleção para reinserção* – os T_p melhores indivíduos entre pais e filhos, de acordo com o valor de aptidão, sobrevivem de uma geração para a outra.

Crossover

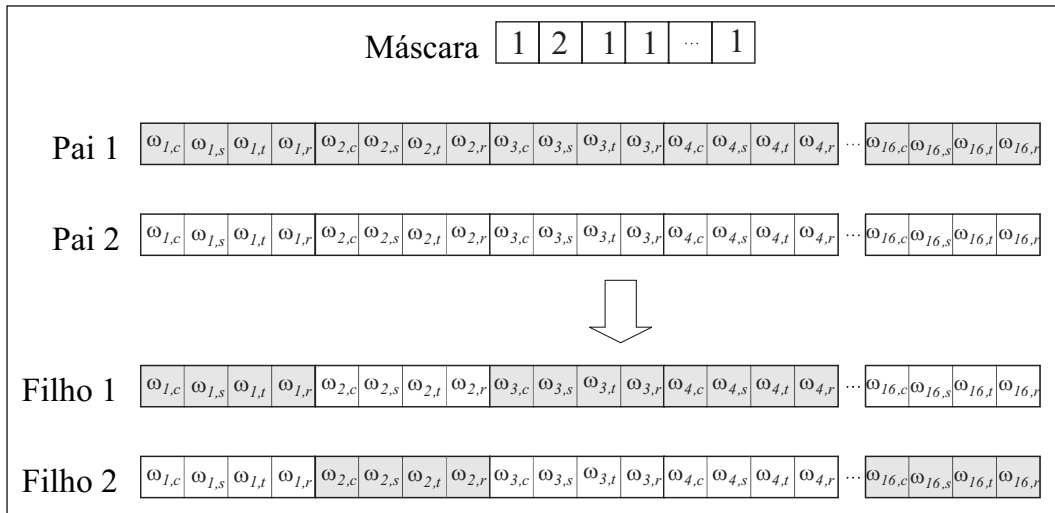


Figura 4.9: Ilustração de uma operação de *crossover*.

Crossover representa o cruzamento de dois indivíduos (pais) para formar dois novos indivíduos (filhos). Neste trabalho foi utilizado *crossover* uniforme (seção 1.4.5) onde é permi-

tido somente a divisão entre genes. A Figura 4.9 ilustra uma operação de *crossover* uniforme considerando nossa representação de indivíduo.

Mutação

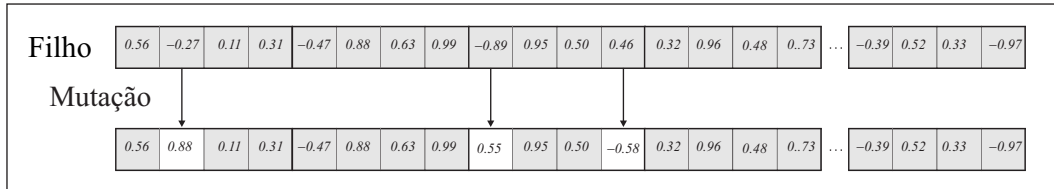


Figura 4.10: Ilustração de uma operação de mutação.

Foi usada mutação uniforme onde um campo de um gene selecionado para mutação é substituído por um outro gerado randomicamente dentro do domínio do campo (em nosso caso, um valor real $\in [-1, 1]$). A Figura 4.10 ilustra uma operação de mutação.

4.4.3 Os Parâmetros de Controle

Todos os parâmetros de controle foram fixados experimentalmente, executando várias tentativas para obter um ajuste ótimo. Os parâmetros de controle que levam aos melhores resultados são: taxa de *crossover* p_c de 60%; probabilidade de mutação p_m de 3%; tamanho da população T_p de 50 indivíduos que evoluem ao longo de 250 gerações, caso nenhum indivíduo de aptidão máxima seja encontrado durante o processo evolutivo.

4.4.4 As Funções de Aptidão

As funções de aptidão são empregadas para ponderar a qualidade dos indivíduos na população. O AG descrito acima foi executado para dez diferentes funções de aptidão. Todas as funções de aptidão são aplicadas para os documentos ordenados em ordem de relevância para a consulta. As funções de aptidão foram classificadas em duas categorias: baseadas em ordem e não baseadas em ordem. Nas medidas de aptidão $F(q, C)$ denota o valor de aptidão do indivíduo C para a consulta q .

Funções de Aptidão não Baseadas em Ordem

Funções de aptidão não baseadas em ordem são medidas onde o valor de utilidade de um documento tem uma frágil relação com sua posição no *ranking*. Por exemplo, se um documento relevante aparece antes de um valor de limiar no *ranking* pode-se atribuir a este um valor de utilidade máximo, caso contrário, se um documento relevante aparece depois deste valor de limiar, um valor de utilidade mínimo ou nulo é atribuído. Nós avaliamos duas funções de aptidão pertencente a esta categoria: $F1_{Stejić et. al}$ utilizado no trabalho de [Stejić et al. 2003a] e $F1_{Proposta}$ que foi proposta neste trabalho.

$F1_{Stejjic\ et.\ al.}$: esta função de aptidão é dada pela medida de Precisão- R , como descrito na seção 2.10.

$F1_{Proposta}$: esta função é baseada numa análise dos números de *true positive* (Rr – relevante e recuperado), *false positive* (Rn – recuperado e não relevante) e *false negative* (Nr – relevante e não recuperado). $|D|$ é o número de documentos conhecidos como relevantes para a consulta q .

$$F1_{Proposta}(q, \omega) = (2|D|) + Rr - Rn - Nr \quad (4.10)$$

Funções de Aptidão Baseadas em Ordem

Funções de aptidão baseadas em ordem são medidas onde o valor de utilidade de um documento é inversamente proporcional a sua posição no *ranking*. Estas funções de aptidão são baseadas no fato de não ser o mesmo um documento relevante aparecer no início ou no fim da lista de documentos recuperados. Também, é conhecido que os usuários preferem que os documentos relevantes apareçam nas primeiras posições do *ranking*, sugerindo que as funções de aptidão baseadas em ordem têm grandes probabilidades de obter sucesso. Neste trabalho experimentamos as funções de aptidão $F1_{Fan\ et\ al.}$, $F2_{Fan\ et\ al.}$, $F3_{Fan\ et\ al.}$ e $F4_{Fan\ et\ al.}$, propostas por [Fan et al. 2004], $F1_{Lopez-Pujalte\ et\ al.}$ e $F2_{Lopez-Pujalte\ et\ al.}$ propostas por [López-Pujalte et al. 2003b], $F_{P_{AVG}}$ utilizada em [Fan et al. 2004], [Torres et al. 2005] e [López-Pujalte et al. 2003b], e $F2_{Proposta}$ proposta neste trabalho. Todas estas funções, com exceção $F2_{Proposta}$, foram descritas no Capítulo 3. Devido à utilização destas em um contexto diferente e por considerar que o *ranking* retornado (documentos recuperados) consiste de todas as imagens da coleção ordenadas conforme sua similaridade à consulta, reescrevemos-as conforme o nosso contexto no Apêndice A. A função $F2_{Proposta}$ é dada pela seguinte equação:

$$F2_{Proposta}(q, C) = \frac{Utilidade(q, C)}{\sum_{j=1}^{|D|} \frac{1}{j}} \quad (4.11)$$

onde,

$$Utilidade(q, C) = \sum_{vi \in D} \frac{1}{\text{Posição de } i \text{ no } ranking(q, C)} \quad (4.12)$$

onde D representa o conjunto de documentos conhecidos como relevantes para a consulta q , $|D|$ representa o número de elementos de D , $ranking(q, C)$ representa o *ranking* de imagens recuperadas para a consulta q usando o indivíduo C . Finalmente, $\sum_{j=1}^{|D|} \frac{1}{j}$ representa a utilidade máxima de um *ranking* para q com respeito ao conjunto D , e $Utilidade(q, C)$, representa utilidade real do *ranking* em resposta à q usando C .

Função de Aptidão Baseadas em Ordem *versus* não Baseadas em Ordem

Nesta seção fizemos algumas considerações com o intuito de comparar matematicamente o comportamento das funções de aptidão baseadas em ordem com as não baseadas em ordem. Suponha que o sistema CBIR seja realimentado com cinco imagens, i.e., $|D| = 5$, e considere R – imagem relevante, e N – imagem não relevante.

Considerando uma função de aptidão não baseada em ordem, ambos os *rankings* abaixo (1 e

2) são matematicamente similares em termos de performance de recuperação, uma vez que eles apresentam o mesmo número de N 's e R 's entre as cinco primeiras imagens. Consequentemente os indivíduos responsáveis por estes terão a mesma aptidão. Já considerando uma função de aptidão baseada em ordem, o *ranking 1* é significativamente superior ao *ranking 2*, o que implica em um maior valor de aptidão para o indivíduo responsável pelo *ranking 1*.

Ranking 1:

$$\left\{ \overbrace{\underbrace{R}_{1^a}, \underbrace{R}_{2^a}, \underbrace{R}_{3^a}, \underbrace{N}_{4^a}, \underbrace{N}_{5^a}}^{|D| \text{ first image of the ranking}}, \underbrace{R}_{6^a}, \underbrace{R}_{7^a}, \underbrace{N}_{8^a}, \dots, \underbrace{N}_{n_I^a} \right\}$$

Ranking 2:

$$\left\{ \overbrace{\underbrace{N}_{1^a}, \underbrace{N}_{2^a}, \underbrace{R}_{3^a}, \underbrace{R}_{4^a}, \underbrace{R}_{5^a}}^{|D| \text{ first image of the ranking}}, \underbrace{N}_{6^a}, \dots, \underbrace{N}_{(n_I-2)^a}, \underbrace{R}_{(n_I-1)^a}, \underbrace{R}_{n_I^a} \right\}$$

Um caso interessante ocorre quando não há imagens relevantes entre as $|D|$ primeiras posições do *ranking*, por exemplo, *rankings 3* e *4*. Novamente supomos que o sistema seja realimentado com cinco imagens e que R denota uma imagem relevante e N uma imagem não relevante. Considerando uma função de aptidão não baseada em ordem, ambos os indivíduos responsáveis pelos *rankings 3* e *4*, terão o valor de aptidão mínimo possível, apesar do *ranking 3* ser claramente superior ao *ranking 4*. Já se considerarmos qualquer função de aptidão baseada em ordem, o valor de aptidão atribuído ao indivíduo responsável pelo *ranking 3* será consideravelmente superior ao valor de aptidão do indivíduo responsável pelo *ranking 4*.

Ranking 3:

$$\left\{ \overbrace{\underbrace{N}_{1^a}, \underbrace{N}_{2^a}, \underbrace{N}_{3^a}, \underbrace{N}_{4^a}, \underbrace{N}_{5^a}}^{|D| \text{ first image of the ranking}}, \underbrace{R}_{6^a}, \underbrace{R}_{7^a}, \underbrace{R}_{9^a}, \underbrace{R}_{10^a}, \underbrace{R}_{11^a}, \underbrace{N}_{12^a}, \dots, \underbrace{N}_{n_I^a} \right\}$$

Ranking 4:

$$\left\{ \overbrace{\underbrace{N}_{1^a}, \underbrace{N}_{2^a}, \underbrace{N}_{3^a}, \underbrace{N}_{4^a}, \underbrace{N}_{5^a}}^{|D| \text{ first image of the ranking}}, \dots, \underbrace{N}_{(n_I-5)^a}, \underbrace{R}_{(n_I-4)^a}, \underbrace{R}_{(n_I-3)^a}, \underbrace{R}_{(n_I-2)^a}, \underbrace{R}_{(n_I-1)^a}, \underbrace{R}_{n_I^a} \right\}$$

4.5 Considerações Finais

Atualmente a subjetividade e o gap semântico constituem os principais problemas acerca da recuperação de imagens por conteúdo para domínios gerais.

Na tentativa de contornar este problema foi apresentado um algoritmo genético embutido no processo de realimentação de relevância para ajustar o modelo de similaridade de imagem utilizado às necessidades do usuário.

Considerando que o problema consiste na otimização do *ranking* mostrado ao usuário, conforme seu ponto de vista e principalmente por estar ciente de que (1) há várias medidas para

análise da qualidade de *rankings*, onde não existe uma unanimidade sobre qual é a melhor, (2) a exatidão da medida de aptidão é o fator chave para o bom funcionamento de um AG, propomos duas e experimentamos um total de dez medidas de aptidão. A avaliação de performance do sistema é apresentada nos próximos capítulos.

A performance de um sistema CBIR pode ser avaliada através de dois aspectos:

- Eficiência – medida através do tempo de execução de uma consulta;
- Eficácia – medida pela exatidão de recuperação.

A compactabilidade dos descritores e o esquema de indexação são os principais responsáveis pela eficiência de um sistema CBIR. O projeto de descritores, de medidas de similaridade/distâncias ou algoritmos de casamento e modelos de similaridade são os responsáveis pela eficácia.

No próximo Capítulo analisamos a eficiência do nosso sistema através da análise da complexidade do algoritmo correspondente. No Capítulo seguinte avaliamos a eficácia do sistema pelas medidas de Precisão- R e curva Precisão-Revocação.

Estudo de Complexidade do Algoritmo Proposto

Neste capítulo é apresentado inicialmente uma breve revisão sobre análise de complexidade de algoritmos. Tal revisão é ilustrada pela análise dos algoritmos de ordenação: por inserção (*insertion-sort*) e por intercalação (*merge-sort*). Em seguida apresentamos os procedimentos e funções necessários para a implementação do sistema CBIR, com ênfase no processo de realimentação de relevância via Algoritmo Genético. Finalmente é feita uma análise da complexidade do algoritmo de realimentação, da qual concluímos que sua complexidade de tempo é de $O(n \log n)$.

5.1 Introdução

A análise de algoritmos tem como meta determinar os recursos computacionais necessários para executar um dado algoritmo e principalmente comparar a complexidade temporal (eficiência) de algoritmos alternativos.

A maior parte dos algoritmos são pensados para trabalhar com entradas de tamanho arbitrário. O tempo despendido por um algoritmo é normalmente dependente da entrada: a ordenação de um milhão de números demora mais que a ordenação de dez números. Em geral, o tempo de execução de um algoritmo cresce com o tamanho da entrada; assim, é tradicional descrever o tempo de execução de um algoritmo em função do tamanho de sua entrada. Para isso, é preciso definir com cuidado os termos “tempo de execução” e “tamanho de entrada”.

A noção de tamanho de entrada depende do problema que está sendo analisado. Em muitos problemas, como é o caso da ordenação de números e da recuperação de imagens, a medida mais natural é o número de itens na entrada, por exemplo, o tamanho do arranjo n para ordenação, ou o número de imagens em um banco de dados. Outros problemas como a multiplicação de matrizes, operações em grafos, etc., exigem uma análise mais criteriosa sobre o tamanho da entrada.

O tempo de execução de um algoritmo para uma determinada entrada corresponde ao número de operações básicas ou etapas executadas. Consideramos como operação básica qualquer operação que é executada em um tempo constante, ou seja, a execução da operação não

depende do tamanho da entrada. A não ser quando especificado de outra maneira, cada linha de nossos pseudocódigos correspondem a uma operação básica. Uma linha pode gastar um período diferente de outra linha, assim vamos considerar que cada execução da i -ésima linha leva um tempo c_i , onde c_i é uma constante. Este ponto de vista reflete o que ocorrerá quando nossos pseudocódigos forem implementados em computadores reais.

Em geral, a eficiência ou complexidade de um algoritmo é dada por uma função cujo crescimento é proporcional ao tamanho da entrada (tamanho do problema) n . A complexidade entre algoritmos alternativos pode ser diferenciada pela ordem de crescimento das funções que descrevem os tempos de execução dos algoritmos em relação ao tamanho da entrada n . Assim têm-se algoritmos de complexidade linear (n), logarítmica ($\log n$), $n \log n$, polinomial (n^2, n^3, n^c), $n^c \log n$, exponencial (c^n), onde c é uma constante. Normalmente estamos interessados na ordem de crescimento das funções e não em fatores constantes não dependentes do tamanho da entrada. Uma vez que o tamanho da entrada n se torna grande o suficiente, as funções dependentes de n superarão com sobras as diferenças em fatores constantes. Vamos ilustrar melhor este fato através de um exemplo [Cormen et al. 2002].

Considere dois algoritmos. O algoritmo A que leva um tempo aproximadamente igual a $c_1 n^2$ para resolver um problema de n itens de entrada, onde c_1 é uma constante que não depende de n ; e o algoritmo B que leva um tempo aproximadamente igual a $c_2 n \log n$ para resolver o mesmo problema, onde c_2 é outra constante que também não depende de n . Mesmo que c_1 seja significantemente menor que c_2 , quando o tamanho da entrada n se tornar grande o suficiente, a vantagem do algoritmo B de $\log n$ contra n compensará com sobras a diferença entre as constantes c_1 e c_2 , ou seja, sempre haverá um ponto de passagem além do qual o algoritmo B será mais rápido que o algoritmo A. Este mesmo critério empregado para eliminar fatores constantes também é empregado para eliminar funções aditivas de mais baixa ordem.

Como um exemplo concreto de comparação entre os algoritmos A e B vamos considerar que o algoritmo A seja executado por um computador “rápido” A, enquanto o algoritmo B é executado por um computador “lento” B. Cada um deles recebe um milhão de itens como dados de entrada. Suponha que o computador A execute um bilhão de instruções por segundo e que o computador B execute apenas dez milhões de instruções por segundo; assim, o computador A é 100 vezes mais rápido que o computador B em capacidade bruta de computação. Para tornar a diferença ainda mais drástica, suponha que um programador especialista codifique o algoritmo A em uma linguagem de máquina para o computador A, e que o código resultante exija $2n^2$ instruções para uma entrada de tamanho n (de forma que $c_1 = 2$). Por outro lado, suponha que o algoritmo B seja programado no computador B por um programador que utiliza uma linguagem de alto nível com um compilador ineficiente, com o código resultante totalizando $50n \log n$ instruções (de forma que $c_2 = 50$). Para ordenar um milhão de números o computador A demora

$$\frac{2 \cdot (10^6)^2 \text{ instruções}}{10^9 \text{ instruções/segundo}} = 2000 \text{ segundos},$$

enquanto o computador B demora

$$\frac{50 \cdot 10^6 \log 10^6 \text{ instruções}}{10^7 \text{ instruções/segundo}} \approx 100 \text{ segundos}.$$

Mesmo utilizando um computador e compilador inferiores, o algoritmo B foi executado aproximadamente 20 vezes mais rápido que o algoritmo A. À medida que o tamanho do problema aumenta (número de itens de entrada), também aumenta a vantagem relativa do algoritmo B.

O exemplo anterior mostra que os algoritmos, assim como os *hardware* de computadores, constituem uma tecnologia. O desempenho total de um sistema depende tanto da escolha de algoritmos eficientes quanto da escolha de *hardware* rápidos [Cormen et al. 2002].

Para entendermos melhor o processo de análise de tempo de um algoritmo, vamos considerar o seguinte problema: ordenar uma seqüência de números em ordem não crescente. Esse problema surge com frequência na prática, inclusive em nosso problema de realimentação de relevância (na construção de *rankings*), e oferece um solo fértil para a introdução de muitas técnicas de projeto padrão e ferramentas de análise. Vejamos como definir formalmente o problema de ordenação:

Entrada: uma seqüência de n números $\langle a_1, a_2, \dots, a_n \rangle$

Saída: uma permutação (reordenação) $\langle a'_1, a'_2, \dots, a'_n \rangle$ da seqüência de entrada, tal que $a'_1 \geq a'_2 \geq \dots \geq a'_n$

Dada uma seqüência de entrada como $\langle 31, 41, 59, 26, 41, 58 \rangle$, um algoritmo de ordenação conforme o especificado retorna como saída a seqüência $\langle 59, 58, 41, 41, 31, 26 \rangle$. Uma seqüência de entrada como essa é chamada uma instância do problema de ordenação. Em geral, uma instância de um problema consiste em uma entrada, que satisfaz a quaisquer restrições impostas no enunciado do problema, necessária para se calcular uma solução para o problema. Adicionalmente, um algoritmo é dito correto se para cada instância de entrada, ele pára com a saída correta. Dizemos que um algoritmo correto resolve o problema computacional dado.

Existem vários algoritmos para a ordenação de n números [Cormen et al. 2002]: *insertion-sort*, *bubble-sort*, *quicksort*, *merge-sort*, *heapsort*, etc. Aqui analisaremos o tempo de execução de dois deles: ordenação por inserção (*insertion-sort*) e ordenação por intercalação (*merge-sort*). Os algoritmos *bubble-sort*, *quicksort*, *heapsort* e suas correspondentes análises de complexidade podem ser vistas em [Cormen et al. 2002].

5.1.1 *Insertion-sort*

A ordenação por inserção (*insertion-sort*) funciona da mesma maneira como muitas pessoas ordenam as cartas em um jogo de baralho. Inicia-se com a mão esquerda vazia e as cartas viradas com a face para baixo na mesa. Em seguida, remove uma carta de cada vez da mesa e insere-a na posição correta na mão esquerda. Para encontrar a posição correta de uma carta, compara-se esta com cada uma das cartas que já estão na mão, da direita para a esquerda, como ilustra a Figura 5.1. Em cada instante, as cartas na mão esquerda estão ordenadas.

Nosso pseudocódigo para ordenação por inserção (*insertion-sort*) é descrito pelo Algoritmo 5.1, que toma como parâmetro de entrada o arranjo $A[1..n]$ de comprimento n que deverá ser ordenado. Os números são reorganizados dentro do arranjo A , com no máximo um número constante deles armazenados fora do arranjo a qualquer instante. O arranjo de entrada A conterá a seqüência de números ordenada quando o *insertion-sort* terminar.



Figura 5.1: Ordenando cartas manualmente conforme o algoritmo *insertion-sort*.

<i>Insertion-sort</i> (<i>A</i>)	custo	vezes
1 para $j \leftarrow 2$ até comprimento[<i>A</i>] faça	c_1	n
2 $chave \leftarrow A[j]$	c_2	$n-1$
3 $i \leftarrow j-1$	c_3	$n-1$
4 enquanto $i > 0$ e $A[i] < chave$ faça	c_4	$\sum_{j=2}^n t_j$
5 $A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^n (t_j - 1)$
6 $i \leftarrow i-1$	c_6	$\sum_{j=2}^n (t_j - 1)$
7 $A[i+1] \leftarrow chave$	c_7	$n-1$

Algoritmo 5.1: *Insertion-sort*.

Agora vamos analisar o tempo de execução do algoritmo *insertion-sort*. Suponha que o custo para executar as instruções 1–7 sejam dados por $c_1, \dots, c_7$. A coluna *vezes* indica o número de vezes que cada instrução é executada. Assim o tempo de execução do algoritmo para um arranjo de n elementos é dado por:

$$T(n) = c_1 n + (c_2 + c_3)(n - 1) + c_4 \sum_{j=2}^n t_j + (c_5 + c_6) \left(\sum_{j=2}^n (t_j - 1) \right) + c_7(n - 1)$$

onde t_j é o número de vezes que a condição da linha 4 é verificada. Note que $1 \leq t_j \leq j$. Desta forma vamos analisar o tempo no melhor e no pior caso.

Melhor caso: a sequência já está ordenada, isto é, $t_j = 1, \forall j$. Daí,

$$\sum_{j=2}^n t_j = \sum_{j=2}^n 1 = n - 1$$

e

$$\sum_{j=2}^n (t_j - 1) = 0$$

Assim,

$$\begin{aligned} T(n) &= c_1 n + (c_2 + c_3 + c_4 + c_7)(n - 1) \\ T(n) &= (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7) \\ T(n) &= an + b, \end{aligned}$$

onde a e b são constantes. Logo, no melhor caso, o algoritmo de ordenação por inserção é linear.

Pior caso: a sequência está ordenada em ordem contrária, isto é, $t_j = j, \forall j$. Daí,

$$\sum_{j=2}^n t_j = \sum_{j=2}^n j = \frac{n(n+1)}{2} - 1$$

e

$$\sum_{j=2}^n (t_j - 1) = \sum_{j=2}^n (j - 1) = \sum_{j=2}^n j - \sum_{j=2}^n 1 = \frac{n(n+1)}{2} - 1 - (n - 1) = \frac{n(n+1)}{2} - n$$

Assim,

$$\begin{aligned} T(n) &= c_1 n + (c_2 + c_3)(n - 1) + c_4 \left[\frac{n(n+1)}{2} - 1 \right] + (c_5 + c_6) \left[\frac{n(n+1)}{2} - n \right] \\ &\quad + c_7(n - 1) \\ T(n) &= \left(\frac{c_4 + c_5 + c_6}{2} \right) n^2 + \left(c_1 + c_2 + c_3 + \frac{c_4 + c_5 + c_6}{2} + c_7 \right) n \\ &\quad - (c_2 + c_3 + c_4 + c_7) \\ T(n) &= an^2 + bn + c, \end{aligned}$$

onde a , b e c são constantes. Logo, no pior caso, o algoritmo de ordenação por inserção é quadrático.

5.1.2 Ordem de Crescimento de Funções e Notação Assintótica

A ordem de crescimento do tempo de execução de um algoritmo, fornece uma caracterização simples da eficiência do algoritmo, e também nos permite comparar o desempenho de algoritmos alternativos. Embora seja possível determinar o tempo exato da execução de um algoritmo, como fizemos no caso da ordenação por inserção (*insertion-sort*), a precisão extra geralmente não vale o esforço de calculá-la. Para entradas grandes o bastante, as constantes multiplicativas e os termos de mais baixa ordem de um tempo de execução exato são dominados pelos efeitos do próprio tamanho da entrada.

Quando consideramos entradas grandes o suficiente para tornarem relevantes apenas a ordem de crescimento das funções dos tempos de execução, estamos estudando a **eficiência assintótica** dos algoritmos [Cormen et al. 2002] e [Ziviani 2004].

A **notação assintótica** é usualmente definida através de funções no conjunto dos $\mathbb{N}$ [Cormen et al. 2002]. Seja f a função de complexidade de um dado algoritmo F .

- Dizemos que $f(n) \in O(g(n))$ se existem constantes positivas c e n_0 tais que

$$0 \leq f(n) \leq cg(n) \quad \forall n \geq n_0.$$

- Dizemos que $f(n) \in \Omega(g(n))$ se existem constantes positivas c , n_0 tais que

$$0 \leq cg(n) \leq f(n) \quad \forall n \geq n_0.$$

- Dizemos que $f(n) \in \Theta(g(n))$ se

$$f(n) \in O(g(n)) \text{ e } f(n) \in \Omega(g(n))$$

Por abuso de linguagem dizemos que $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$ e $f(n) = \Theta(g(n))$. As intuições por trás das notações O , Ω e Θ são mostradas na Figura 5.2.

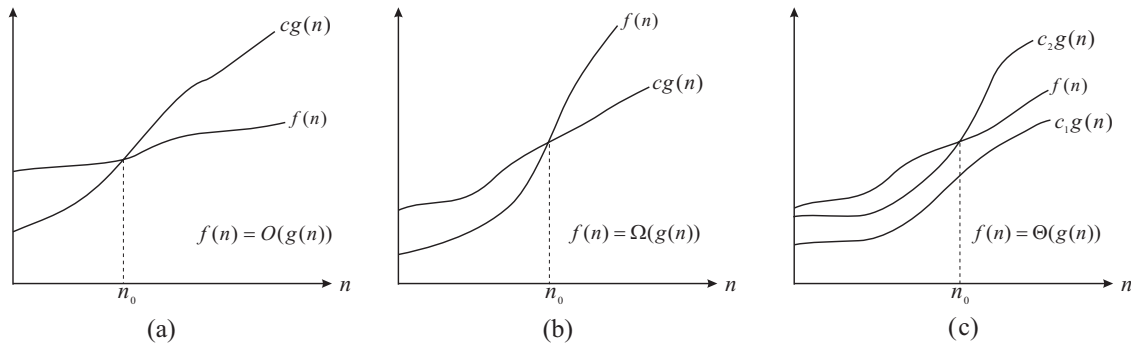


Figura 5.2: Ilustração gráfica das notações O , Ω e Θ .

5.2 Classes de Comportamento Assintótico

Se f é a função de complexidade para um algoritmo F , então $O(f)$ é considerada a **complexidade assintótica** ou a **comportamento assintótico** do algoritmo F . Igualmente, se g é a função de complexidade para um algoritmo G , então $O(g)$ é considerada a complexidade assintótica do algoritmo G . A relação de dominação assintótica permite comparar funções de complexidade.

As principais classes de problemas possuem as funções de complexidade descritas abaixo [Ziviani 2004].

- $f(n) = O(1)$
 - conhecidos como algoritmos de **complexidade constante**;
 - o tempo de execução do algoritmo não depende do tamanho da entrada;

- as instruções do algoritmo são executadas um número fixo de vezes.
- $f(n) = O(\log n)$
 - conhecidos como algoritmos de **complexidade logarítmica**;
 - o problema é resolvido através da decomposição em subproblemas, por exemplo, busca binária.
- $f(n) = O(n)$
 - conhecidos como algoritmos de **complexidade linear**;
 - é realizado um pequeno trabalho sobre cada elemento da entrada;
- $f(n) = O(n \log n)$
 - típico de algoritmos que quebram um problema em outros menores, resolvem cada um deles independentemente e depois combinam as soluções.
- $f(n) = O(n^2)$
 - conhecidos como algoritmos de **complexidade quadrática**
 - ocorrem quando os itens de dados são processados aos pares. Típico de dois laços aninhados.
- $f(n) = O(n^3)$
 - conhecidos como algoritmos de **complexidade cúbica**
 - típico de algoritmos com três laços aninhados.
- $f(n) = O(2^n)$
 - conhecidos como algoritmos de **complexidade exponencial**
 - ocorre na solução de problemas complexos por **força bruta**. Geralmente não são úteis sob o ponto de vista prático.
- $f(n) = O(n!)$
 - também conhecidos como algoritmos de **complexidade fatorial**
 - ocorre na solução de alguns problemas complexos por **força bruta**. Têm comportamento pior que os algoritmos de ordem $O(2^n)$.

5.2.1 Merge-sort

O algoritmo *merge-sort* é um exemplo típico de um algoritmo que utiliza a técnica de **dividir e conquistar**. A técnica de dividir e conquistar consiste em:

Dividir o problema em um determinado número de subproblemas.

Conquistar os subproblemas, resolvendo-os recursivamente. Porém, se o tamanho de um subproblema for pequeno o bastante, basta resolvê-lo de maneira direta.

Combinar as soluções dos subproblemas a fim de formar a solução do problema original.

A ordenação *merge-sort* é dada pelo Algoritmo 5.2.

<i>Merge-sort</i> (<i>A</i> , <i>início</i> , <i>fim</i>)	
1	se <i>início</i> < <i>fim</i> então
2	<i>meio</i> ← $\left\lfloor \frac{\textit{início} + \textit{fim}}{2} \right\rfloor$
3	<i>Merge-sort</i> (<i>A</i> , <i>início</i> , <i>meio</i>)
4	<i>Merge-sort</i> (<i>A</i> , <i>meio</i> +1, <i>fim</i>)
5	<i>Intercala</i> (<i>A</i> , <i>início</i> , <i>meio</i> , <i>fim</i>)

Algoritmo 5.2: Algoritmo *merge-sort*.

O algoritmo *merge-sort* opera da seguinte forma:

- Dividir a seqüência de n elementos a serem ordenados em duas subsequências de $n/2$ elementos cada uma;
- Conquistar as duas subsequências recursivamente, utilizando *Merge-sort*;
- Combinar (intercalar) as duas seqüências ordenadas, de modo a produzir a resposta ordenada.

O tempo de execução de algoritmos que seguem a abordagem de dividir e conquistar ou algoritmos recursivos é expresso por equações de recorrências.

Seja $T(n)$ o tempo necessário para resolver um dado problema de tamanho n . Se o tamanho do problema for pequeno o bastante, digamos $n \leq c$ para alguma constante c , a solução direta demorará um tempo constante, que consideraremos $\Theta(1)$. Vamos supor que o problema seja dividido em a subproblemas, cada um dos quais de tamanho n/b . (No caso do algoritmo *Merge-sort* tanto a quanto b são iguais a 2, mas existem muitos algoritmos de dividir e conquistar nos quais $a \neq b$). Considerando que é gasto o tempo $D(n)$ para dividir o problema em subproblemas e o tempo $C(n)$ para combinar as soluções dadas aos subproblemas na solução para o problema original, temos a recorrência

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq c, \\ aT(n/b) + D(n) + C(n) & \text{caso contrário.} \end{cases}$$

Embora o pseudocódigo de *Merge-sort* funcione corretamente para quaisquer números de elementos, simplificaremos nossa análise por supor que o tamanho da entrada é uma potência de dois. Então, cada passo de dividir produzirá duas subsequências de tamanho exatamente $n/2$. Tal premissa não afeta a ordem de crescimento da solução para a recorrência.

Apresentaremos em seguida as razões para configurar a recorrência correspondente a $T(n)$, considerando o tempo de execução do pior caso de *Merge-sort* sobre n números, conforme a equação 5.1. A ordenação *Merge-sort* sobre um único elemento demora um tempo constante. Quando temos $n > 1$ elementos, desmembramos o tempo de execução do modo explicado a seguir:

Dividir – a etapa de dividir simplesmente calcula o ponto médio da (sub)sequência, o que demora um tempo constante. Portanto $D(n) = \Theta(1)$.

Conquistar – resolvemos recursivamente dois subproblemas, cada um com tamanho $n/2$ e contribui com $2T(n/2)$ para o tempo de execução.

Combinar – o procedimento *Intercala* uma sequência de n elementos e leva o tempo $\Theta(n)$. Este pode ser visto em [Cormen et al. 2002]. Assim, $C(n) = \Theta(n)$.

A soma de $\Theta(n)$ com $\Theta(1)$ resulta em $\Theta(n)$, ou seja, a soma de uma função linear com uma constante resulta em uma função linear. Assim a função de recorrência para o pior caso de *Merge-sort*, $T(n)$, é dada por:

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1, \\ 2T(n/2) + \Theta(n) & \text{se } n > 1 \end{cases} \quad (5.1)$$

Mostraremos que $T(n)$ é $O(n \log n)$, onde $\log n$ significa $\log_2 n$. Vamos reescrever a recorrência 5.1 como

$$T(n) = \begin{cases} c & \text{se } n = 1, \\ 2T(n/2) + cn & \text{se } n > 1 \end{cases} \quad (5.2)$$

onde a constante c representa o tempo exigido para resolver problemas de tamanho 1, como também o tempo por elemento da sequência para as etapas de dividir e combinar. A Figura 5.3 mostra como podemos desmembrar a recorrência 5.2. Na parte (a) da Figura 5.3 temos $T(n)$, em (b) $T(n)$ foi expandido em uma árvore equivalente à recorrência. O termo cn é a raiz (o custo no nível superior da recursão), e as duas subárvores da raiz são as duas recorrências menores $T(n/2)$. A parte (c) mostra este processo levado uma etapa adiante pela expansão de $T(n/2)$. Continuamos a expandir cada nó na árvore, desmembrando-o em suas partes constituintes como determina a recorrência, até o tamanho dos problemas se reduzirem a 1, cada qual com o custo c . (d) mostra a árvore resultante.

Em seguida, somamos os custos através de cada nível da árvore. O nível superior tem custo cn , o próximo nível tem custo $cn/2 + cn/2 = cn$, o nível após este tem custo $cn/4 + cn/4 + cn/4 + cn/4 = cn$ e assim por diante. Em geral, o nível i abaixo do topo (raiz) tem 2^i nós, cada qual com custo $cn/2^i$, de forma que o i -ésimo nível abaixo do topo tem custo $2^i c(n/2^i) = cn$. No nível inferior existem n nós, cada um com custo c , resultando em um custo total de cn .

O número total de níveis da árvore de recursão da Figura 5.3 é $\log n + 1$. Podemos verificar isto através de uma prova por indução:

Hipótese de Indução: $A(n)$

$$A(n) \equiv n \text{ nós} \Rightarrow \log n + 1 \text{ níveis}$$

Supondo que $n = 2^i \Rightarrow \log 2^i + 1 = i + 1$ níveis

Base de Indução: $A(n)$ é verdadeira para $n = 1$?

$$\log 1 = 0 \Rightarrow \log n + 1 = 1 \text{ ok!}$$

Isto é, para um nó, o número de níveis na árvore é um.

Passo de Indução: $A(n) \Rightarrow A(n + 1)$?

Supondo que 2^i nós $\Rightarrow \log 2^i + 1 = i + 1$ níveis, devemos provar que $2^{(i+1)}$ nós $\Rightarrow (i + 1) + 1$ níveis.

$$2^{(i+1)} \Rightarrow \log 2^{(i+1)} + 1 = (\log 2^i + \log 2^1) + 1 = (i + 1) + 1$$

Portanto, uma árvore de recursão de n nós no nível inferior tem $\log n + 1$ níveis. Como cada nível tem custo cn o custo é de $cn \log n + cn$.

Além do **método da árvore de recursão**, existem outros métodos para resolver recorrências tais como: o **método da substituição** e o *método mestre* [Cormen et al. 2002].

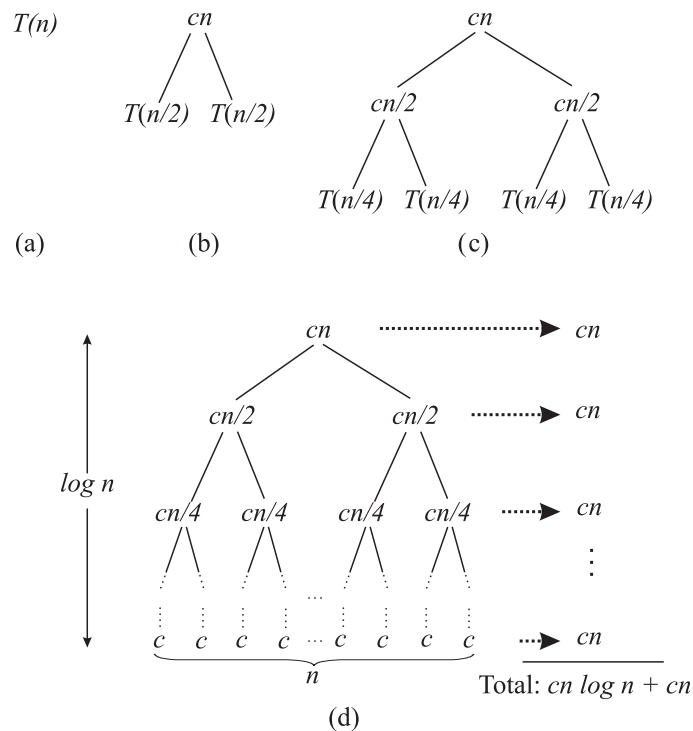


Figura 5.3: Árvore de recursão do algoritmo *Merge-sort*.

5.3 Algoritmos do Sistema Proposto

Os algoritmos do sistema CBIR proposto podem ser divididos em duas partes: a busca inicial e a realimentação de relevância.

A busca inicial consiste basicamente na computação da equação 4.8 para todas as imagens da coleção de referência e na ordenação dos valores de similaridade obtidos para formar o *ranking* que será mostrado ao usuário. Tal procedimento é realizado conforme o Algoritmo 5.6, considerando todos os pesos iguais a 1 (um).

Já a descrição do processo de realimentação de relevância é um pouco mais complexa vista que este é feito por um algoritmo genético, tornando-se necessária a especificação de série de operações como seleção, *crossover*, mutação, medida de aptidão, etc. Assim, a partir deste instante detalharemos o algoritmo genético (AG) responsável pelo ajuste da medida de similaridade conforme o critério do usuário expresso na realimentação do sistema.

O fluxo geral do AG é mostrado no Algoritmo 5.3. A partir desta descrição geral do algoritmo derivaremos uma série de sub-algoritmos correspondentes às operações do AG.

Dado que já especificamos anteriormente a codificação de indivíduo (equação 4.9), o primeiro passo é gerar aleatoriamente a população inicial. Para tal consideramos o Algoritmo 5.4.

-
- 1 Gerar a população inicial de T_p indivíduos de forma randômica (Ver Capítulo 4)
 - 2 Avaliar a aptidão dos indivíduos da população inicial
 - 3 Executar os seguintes passos enquanto o critério de parada não for atingido (isto é, enquanto $N_g \leq 250$ e a aptidão do melhor indivíduo for menor que a aptidão máxima):
 - 3.1 Selecionar $T_p \times P_c$ indivíduos para reprodução
 - a) Submeter os indivíduos selecionados aos pares à operações de *crossover*
 - b) Submeter os descendentes (filhos) à mutação com probabilidade P_m
 - 3.2 Avaliar a aptidão dos novos indivíduos formados por *crossover* e mutação
 - 3.3 Selecionar os T_p melhores indivíduos entre pais e filhos para a próxima geração
-

Algoritmo 5.3: Passos principais do algoritmo genético proposto

Em seguida são avaliadas as aptidões dos indivíduos conforme o Algoritmo 5.5. A avaliação da aptidão de um indivíduo consiste em atribuir uma nota ao *ranking* gerado pelo indivíduo. Na geração de um *ranking* consideramos o Algoritmo 5.6. O processo de atribuição de um valor de aptidão depende da função de aptidão utilizada, e como experimentamos um total de dez funções de aptidão, apenas trataremos sobre estas na análise de tempo do algoritmo.

O Algoritmo 5.7 ilustra o fluxo geral dos processos de seleção, *crossover* e mutação. A seleção é feita pelo método da roleta onde é necessária a construção da roleta a cada geração. Após construída a roleta através do Algoritmo 5.8 é sorteado um par de indivíduos através do Algoritmo 5.9¹ sendo estes submetidos a operação de *crossover* (Algoritmo 5.10). Os indivíduos gerados após o *crossover* (filhos) são submetidos a operação de mutação (Algoritmo 5.11). Este fluxo de seleção, *crossover* e mutação é repetido $(T_p P_c)/2$ vezes. A cada geração sobrevivem os melhores indivíduos entre pais e filhos (Algoritmo 5.12.)

¹A cada giro da roleta é selecionado um indivíduo

Função GeraPopulaçãoInicial()

```

1  para  $i \leftarrow 1$  até  $T_p$ 
2      para  $r \leftarrow 1$  até  $n_R$ 
3           $C_i(r, \text{região}) \leftarrow \text{Randômico}([-1, 1])$ 
4           $C_i(r, \text{cor}) \leftarrow \text{Randômico}([-1, 1])$ 
5           $C_i(r, \text{forma}) \leftarrow \text{Randômico}([-1, 1])$ 
6           $C_i(r, \text{textura}) \leftarrow \text{Randômico}([-1, 1])$ 
7  retorne  $C_{1..T_p}$ 

```

Algoritmo 5.4: Função de geração da população inicial. A função $\text{Randômico}([-1, 1])$ gera um valor real randômico no intervalo $[-1, 1]$.

Função Aptidão(C)

```

1   $C.\text{aptidão} \leftarrow \text{Avalia}(\text{Ranking}(q, C, I))$ 
2  retorne  $C$ 

```

Algoritmo 5.5: Função de avaliação da aptidão de um indivíduo. A função avalia depende da função de aptidão. A função Ranking (Algoritmo 5.6) gera um *ranking* conforme a similaridade entre a consulta q e as imagens da coleção de referência I , considerando a ponderação de um indivíduo C .

Função $\text{Ranking}(q, C, I)$

```

1  para  $i \leftarrow 1$  até  $n_I$ 
2       $S_I(q, i) \leftarrow 0$ 
3      para  $r \leftarrow 1$  até  $n_R$ 
4           $S_R(q, i, r) \leftarrow C(r, \text{cor}) * S_F(q, i, r, \text{cor}) + C(r, \text{forma}) * S_F(q, i, r, \text{forma})$ 
5               $+ C(r, \text{textura}) * S_F(q, i, r, \text{textura})$ 
6           $S_I(q, i) \leftarrow S_I(q, i) + C(r, \text{região}) * S_R(q, i, r)$ 
7   $\text{Ordena}(S_I)$ 
8  retorne ranking ordenado

```

Algoritmo 5.6: Função para a geração de um *ranking* considerando uma consulta q , um indivíduo C e a coleção de referência I

Fluxo dos procedimentos de seleção, *crossover* e mutação

```

1   $k \leftarrow T_p$ 
2   $Roleta \leftarrow \text{MontarRoleta}(C)$ 
3  para  $i \leftarrow 1$  até  $(P_c T_p)/2$ 
4       $P1 \leftarrow \text{GirarRoleta}(Roleta, C)$ 
5       $P2 \leftarrow \text{GirarRoleta}(Roleta, C)$ 
6       $[F1, F2] \leftarrow \text{Crossover}(P1, P2)$ 
7       $F1' \leftarrow \text{Mutação}(F1)$ 
8       $F2' \leftarrow \text{Mutação}(F2)$ 
9       $k \leftarrow k + 1$ 
10      $C_k \leftarrow F1'$ 
11      $k \leftarrow k + 1$ 
12      $C_k \leftarrow F2'$ 

```

Algoritmo 5.7: Fluxo do processo de reprodução: seleção, *crossover* e mutação, correspondente ao passo 3.2 do Algoritmo 5.3.

Função MontarRoleta($C1..T_p$)

```

1   $Roleta(1) \leftarrow C1.\text{aptidão}$ 
2  para  $i \leftarrow 2$  até  $T_p$ 
3       $Roleta(i) \leftarrow Roleta(i-1) + C_i.\text{aptidão}$ 
4  retorne  $Roleta$ 

```

Algoritmo 5.8: Função de montagem de uma roleta.

Função GirarRoleta($Roleta, C1..T_p$)

```

1   $n \leftarrow \text{Randômico}([0, Roleta(T_p)])$ 
2  para  $i \leftarrow 1$  até  $T_p$ 
3      se  $n \leq Roleta(i)$ 
4  retorne  $C_i$ 

```

Algoritmo 5.9: Função de sorteio pelo método da roleta. A função $\text{Randômico}([0, Roleta[T_p]])$ gera um valor real randômico entre 0 e $Roleta[T_p]$.

Função Crossover(Pai1, Pai2)

```

1  para  $r \leftarrow 1$  até  $T_p$ 
2       $\text{sorteio} \leftarrow \text{RandômicoBinário}()$ 
3      se  $\text{sorteio} = 0$ 
4           $F1(r, :) \leftarrow P1(r, :)$ 
5           $F2(r, :) \leftarrow P2(r, :)$ 
6      senão
7           $F1(r, :) \leftarrow P2(r, :)$ 
8           $F2(r, :) \leftarrow P1(r, :)$ 
9  retorne  $F1$  e  $F2$ 

```

Algoritmo 5.10: Função de *crossover*. A função *RandômicoBinário()* gera 0 ou 1 com iguais probabilidades.

Função Mutação(Filho)

```

1   $F' \leftarrow \text{Filho}$ 
2  para  $r \leftarrow 1$  até  $n_R$ 
3       $n \leftarrow \text{Randômico}([0, 1])$ 
4      se  $n \leq P_m$ 
5           $F'(r, \text{região}) \leftarrow \text{Randômico}([-1, 1])$ 
6       $n \leftarrow \text{Randômico}([0, 1])$ 
7      se  $n \leq P_m$ 
8           $F'(r, \text{cor}) \leftarrow \text{Randômico}([-1, 1])$ 
9       $n \leftarrow \text{Randômico}([0, 1])$ 
10     se  $n \leq P_m$ 
11          $F'(r, \text{forma}) \leftarrow \text{Randômico}([-1, 1])$ 
12      $n \leftarrow \text{Randômico}([0, 1])$ 
13     se  $n \leq P_m$ 
14          $F'(r, \text{textura}) \leftarrow \text{Randômico}([-1, 1])$ 
15 retorne  $F'$ 

```

Algoritmo 5.11: Função de Mutação. A função *Randômico*([0, 1]) gera um valor real randômico no intervalo [0,1].

Função Reinservação($C_{1..(T_p+P_cT_p)}$)

```

1   $\text{Ordena}(C_{1..(T_p+P_cT_p)}, \text{'aptidão'})$ 
2  retorne  $C_{1..T_p}$ 

```

Algoritmo 5.12: Função de reinservação. Responsável por eleger os sobreviventes de uma geração para outra. A função *Ordena*($C_{1..(T_p+T_pP_c)}$, *aptidão*) ordena os indivíduos utilizando como chave suas aptidões.

5.4 Análise de Complexidade

Dado que nosso sistema consiste do processo de busca inicial seguido do processo de realimentação de relevância, analisaremos separadamente a complexidade de tempo de cada um destes.

5.4.1 Busca Inicial

O processo de busca inicial é dado pelo Algoritmo 5.6. Considerando que este algoritmo consiste de um laço que é executado n vezes, sendo n o número de imagens da coleção e também considerando que a cada execução do laço é gasto um tempo constante, pois n_R é uma constante, tem-se que o código da linha 1 à 5 tem complexidade $O(n)$. A ordenação dos valores de similaridade (linha 6) é de complexidade $O(n \log n)$ considerando que esta é feita pelo Algoritmo *merge-sort*. Assim a complexidade do algoritmo de busca inicial é de $O(n) + O(n \log n) = O(n \log n)$ dado que na notação assintótica eliminamos os termos de mais baixa ordem.

5.4.2 Realimentação de Relevantes

A análise da complexidade do algoritmo de realimentação de relevância via AG se concentrará na análise de seu fluxo descrito pelo Algoritmo 5.3. Assim analisaremos o tempo de cada uma de suas operações para posteriormente compor o tempo total do algoritmo. Consideraremos como o tamanho do problema (tamanho de entrada) n o número de imagens da coleção de referência. Assim temos que n é igual a n_I . Os demais itens considerados no problema: número de regiões n_R , número de características de imagem n_F , número de indivíduos T_p são constantes e por isso não foram incorporados no tamanho do problema, desta forma não afetando o comportamento assintótico do algoritmo.

- Geração da população inicial (Algoritmo 5.4) – é de complexidade $O(1)$ pois o tamanho da população T_p e o número de regiões n_R não dependem do número de imagens do banco de dados.
- Avaliação da população (Algoritmo 5.5) – a qualidade de um indivíduo corresponde a qualidade do *ranking* gerado por aquele indivíduo. A análise da complexidade deste algoritmo depende da medida de aptidão utilizada, entretanto no pior caso, uma medida analisa a relevância de todas as imagens do *ranking* sendo portanto de ordem $O(n)$. O processo de geração de um *ranking* correspondente a um indivíduo é dado pelo Algoritmo 5.6 que é de ordem $O(n \log n)$, justamente a complexidade do algoritmo de ordenação, pelos seguintes motivos: o código da linha 1 à linha 5 tem complexidade linear $O(n)$; a ordenação, em seguida de complexidade $O(n \log n)$. Logo $O(n) + O(n \log n) = O(n \log n)$. Portanto a função de avaliação da população (Algoritmo 5.5) é de complexidade $O(n \log n)$.
- Fluxo seleção, *crossover* e mutação (Reprodução) – é de complexidade $O(1)$ pela seguinte razão. A função de montagem da roleta (Algoritmo 5.8 é de complexidade $O(1)$ visto que o tamanho da população é uma constante). As funções girar roleta (Algoritmo 5.9)

e crossover (Algoritmo 5.10) são de complexidade $O(1)$ pelo mesmo motivo da anterior. A função mutação (Algoritmo 5.11) é também de complexidade $O(1)$ devido ao número de genes n_R considerados ser constante.

- Reinserção (Algoritmo 5.12) – devido ao número de indivíduos a serem ordenados ser constante, tal função leva um tempo constante sendo portanto de complexidade $O(1)$.

Agora que analisamos a complexidade de todas as operações envolvidas no algoritmo genético estamos aptos a calcular a complexidade de tempo deste. Vale lembrar que após a reprodução os indivíduos recém gerados são avaliados onde é gasto um tempo de ordem $O(n \log n)$. Dado que o passo 3 do Algoritmo 5.3 é executado por no máximo 250 gerações seu tempo é de:

$$250 \left(\begin{array}{ll} O(1) & \text{– tempo do processo de reprodução (Algoritmo 5.7)} \\ + O(n \log n) & \text{– tempo para avaliação dos indivíduos recém gerados} \\ + \frac{O(1)}{O(n \log n)} & \text{– tempo do algoritmo de reinserção} \end{array} \right)$$

Logo o tempo do passo 3 do Algoritmo 5.3 é de $250O(n \log n) = O(n \log n)$.

Assim o tempo do processo de realimentação de relevância (Algoritmo 5.3) é de:

$$\begin{array}{ll} O(1) & \text{– tempo do algoritmo de geração da população inicial (passo 1)} \\ + O(n \log n) & \text{– tempo para avaliação da população inicial (passo 2)} \\ + \frac{O(n \log n)}{O(n \log n)} & \text{– tempo para a execução do passo 3} \\ \hline O(n \log n) & \text{– tempo total do Algoritmo 5.3} \end{array}$$

Portanto a complexidade de tempo do processo de realimentação de relevância (Algoritmo 5.3) é de $O(n \log n)$.

5.5 Considerações Finais

Verificamos que a complexidade de tempo do algoritmo de busca inicial e do algoritmo de realimentação de relevância ser de complexidade relativamente baixa para os padrões normais, a carga computacional exigida é significativamente alta devido ao algoritmo genético (AG) ser de natureza paralela, ou seja, trabalhar com uma população de indivíduos que evoluem por várias gerações. Se o AG for executado por processamento seqüencial certamente o tempo de processamento será multiplicado por uma constante consideravelmente alta. Por outro lado se o AG for executado de forma paralela esta constante será reduzida consideravelmente.

Também ambos os algoritmos podem exigir alta carga computacional quando é aumentada o número de imagens da coleção de referência, devido ao fato que o tempo necessário para as características de duas imagens apesar de ser constante é significativo para os computadores atuais. Assim o desenvolvimento de técnicas de indexação está entre nossas pretensões de trabalhos futuros. A comparação de imagens de forma paralela apesar de possível não resolve definitivamente o problema.

Resultados Experimentais

Este capítulo é dedicado às avaliações experimentais do sistema CBIR. Os experimentos aqui realizados têm três objetivos principais: comparar os nossos resultados com aqueles obtidos em [Stejić et al. 2003a]; comparar as várias funções de aptidão testadas neste trabalho; e apresentar alguns resultados obtidos por nossa abordagem. Para efeito de comparação dos resultados obtidos foram utilizados os mesmos bancos de dados de teste (*Vistex-60*, *Vistex-167*, *Brodatz-208* e *Corel-1000*), e a mesma medida de performance de recuperação (Precisão-*R*) usados em [Stejić et al. 2003a]. Para comparação das várias funções de aptidão experimentadas utilizamos a medida de Precisão-*R*, a curva Precisão-Revocação, ilustração de resultados obtidos, velocidade de convergência do AG e custo computacional. Adicionalmente vários experimentos em dois bancos de dados de porte médio, de 10000 e de 12750 imagens, ilustram a boa performance da nossa abordagem.

6.1 Bancos de Dados de Teste

Na condução dos experimentos foram utilizados seis bancos de dados de teste. Uma descrição seguida de uma amostra de cada um destes bancos de dados é apresentada a seguir:

Vistex-60: é um banco de dados de 60 imagens coloridas, com resolução de 128×128 *pixels*, de cenas do mundo real. O banco de dados é composto por 10 categorias, de 2 até 12 imagens cada. A fonte é [Vistex Database], diretório `pub/FLAT/scene128x128/`. A Figura 6.1 contém uma amostra deste banco de dados.

Vistex-167: é um banco de dados de 167 imagens coloridas, com resolução de 128×128 *pixels*, de texturas homogêneas. O banco de dados é composto de 19 categorias, com 3 até 20 imagens cada. A fonte é [Vistex Database], diretório `pub/FLAT/128x128/`. A Figura 2.9 (no Capítulo 2) contém uma amostra deste banco de dados.

Brodatz-208: é um banco de dados de 208 imagens em níveis de cinza, com resolução de 128×128 *pixels*, contendo texturas homogêneas. O banco de dados é dividido em 13 categorias, cada uma com 16 imagens. Cada categoria corresponde a uma imagem de 512×512 *pixels*, dividida em 16 sub-imagens não sobrepostas de 128×128 *pixels*. A

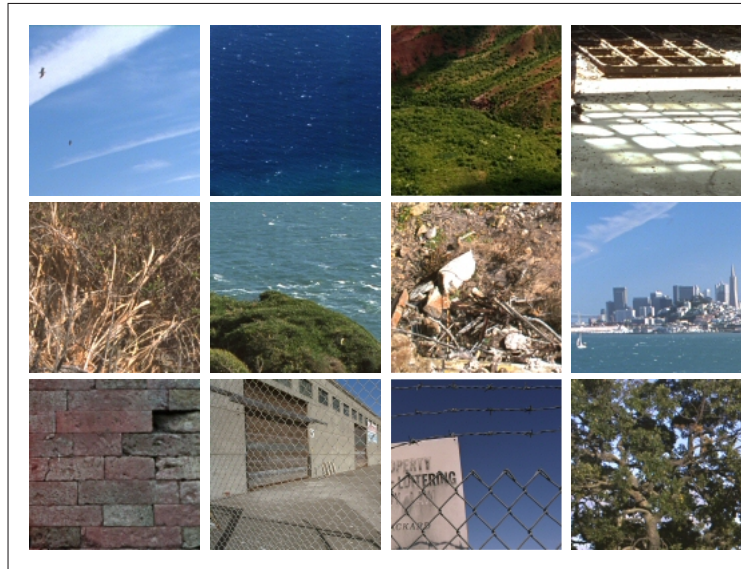


Figura 6.1: Amostra do banco de dados *Vistex-60*

fonte das 13 imagens originais da coleção *Brodatz-208* é [Brodatz Database]. A Figura 6.2 contém uma amostra deste banco de dados.

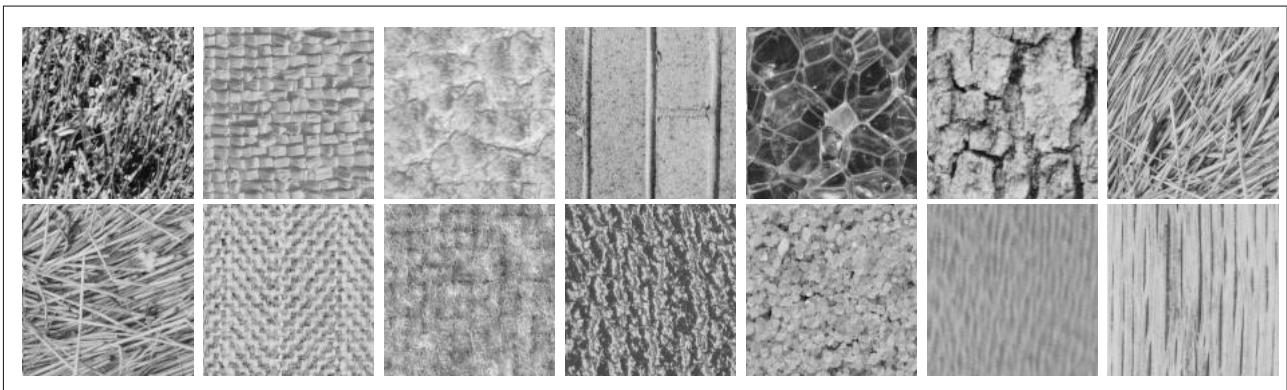
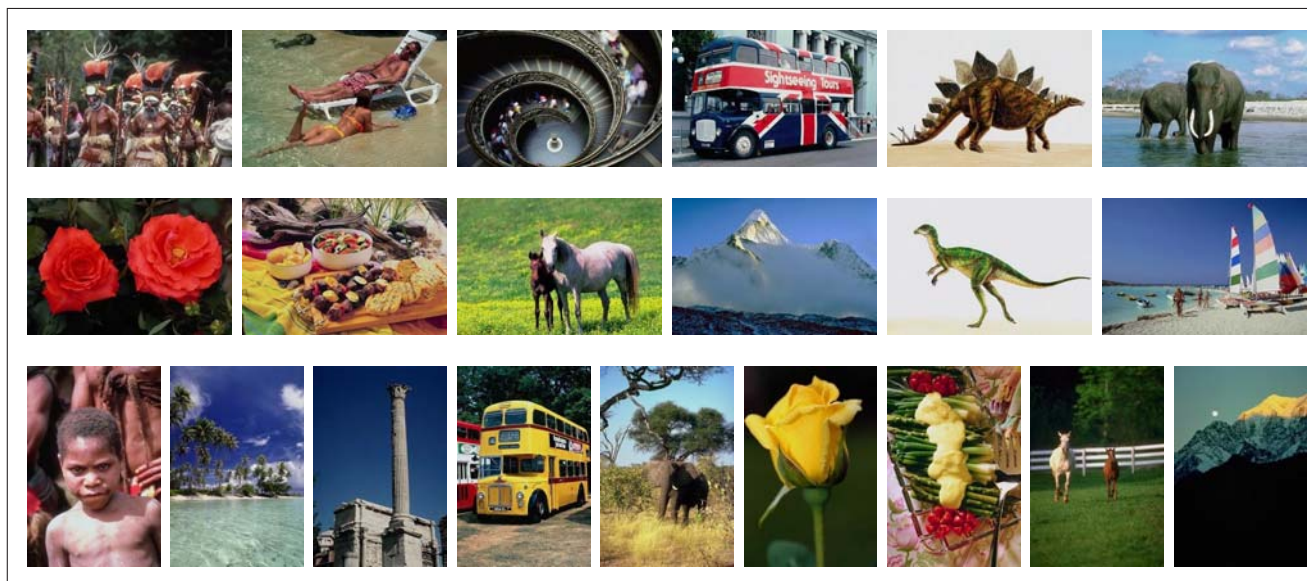


Figura 6.2: Amostra do banco de dados *Brodatz-208*

Corel-1000: é um subconjunto do banco de dados Corel, contendo 1000 imagens com resolução de 384×256 ou de 256×384 *pixels*. O banco de dados é composto por 10 categorias (África, praia, edifícios, ônibus, dinossauros, elefantes, flores, comidas, cavalos e montanhas), com 100 imagens em cada categoria. A fonte é [Corel Database]. A Figura 6.3 contém uma amostra deste banco de dados.

BD-10000: é um banco de dados de 10000 imagens reais cobrindo uma ampla variedade de categorias semânticas (textura uniforme, pôr-do-sol, nuvens, oceanos, animais, mamíferos, peixes, pássaros, répteis, plantas, flores, paisagens naturais, edifícios, motocicletas, carros, aviões, etc) sendo que 1000 dessas imagens são transportadas da coleção *Corel-1000*. As outras 9000 imagens foram coletadas de bases de dados públicas disponíveis na *Web*,

Figura 6.3: Amostra do banco de dados *Corel-1000*

principalmente de [CalPhotos], e pertencem a categorias diferentes daquelas da coleção *Corel-1000*. A Figura 6.4 contém uma amostra deste banco de dados.

Figura 6.4: Amostra do banco de dados *BD-10000*.

BD-12750: é um banco de dados de 12750 imagens reais cobrindo uma ampla variedade de categorias, tais como, texturas uniformes, nuvens, pôr-do-sol, oceanos, paisagens, edifícios, mamíferos, peixes, aves, répteis, árvores, flores, motocicletas, carros, aviões, faces humanas, etc. O banco de dados está em construção e até o momento não é agrupado em categorias. A Figura 6.5 contém uma amostra desta coleção.



Figura 6.5: Amostra do banco de dados *BD-12750*.

6.2 Avaliação Experimental

Para ilustrar a eficiência do nosso sistema de recuperação de imagens com realimentação de relevantes via algoritmo genético e para comparar as várias funções de aptidão uma série de experimentos serão reportados. Assim, a avaliação experimental é dividida em três partes. Na primeira, a nossa abordagem utilizando as várias funções de aptidão testadas é avaliada através da medida de Precisão- R , sendo os resultados obtidos comparados aos melhores resultados de [Stejić et al. 2003a]. Em seguida apresentamos uma análise de performance das várias funções de aptidão testadas através da curva Precisão-Revocação, velocidade de convergência e custo computacional que estas impõem ao AG. Ilustramos a boa performance do sistema através de alguns experimentos.

Antes de apresentar os resultados obtidos, gostaríamos de enfatizar que neste trabalho não foi feito nenhum estudo ou experimento acerca dos parâmetros de ajuste, k_j , $j = 1 \dots 9$ presentes em algumas medidas de aptidão. Nós consideramos os ajustes obtidos nos trabalhos de referência. Os valores usados são: $k_1 = 6$, $k_2 = 1.2$, $k_3 = 2$, $k_4 = 3.65$, $k_5 = 0.1$, $k_6 = 4$, $k_7 = 27.32$, $k_8 = 7$, $k_9 = 0.982$ e $A = 10$.

6.2.1 A Abordagem Proposta *versus* a de Sjeti  et al.

Como nosso sistema CBIR   baseado nos trabalhos de Steji , Takama e Hirota, nosso primeiro experimento consiste em comparar os nossos resultados com os melhores resultados obtidos pelos autores. Como pode ser visto em [Steji  et al. 2003a] os melhores resultados obtidos pelos autores s o atrav s dos modelos: *WLSP-C * e *LSP-C *.

Para tal compara  o usamos a mesma medida de performance – Preci  o-*R*, e os mesmos bancos de dados usados em [Steji  et al. 2003a]: *Vistex-60*, *Vistex-167*, *Brodatz-218* e *Corel-1000*. Adicionalmente testamos a t cnica proposta no banco de dados *BD-10000*.

O modelo proposto e o modelo *WLSP-C * t m o mesmo espa o de busca (pesos de regi es e de caracter sticas cont nuos, positivos e negativos). J  o modelo *LSP-C * tem um espa o de busca um pouco menor, uma vez que n o s o considerados os pesos de regi es.

Os resultados nos bancos de dados *Vistex-60*, *Vistex-167*, *Brodatz-208* e *Corel-1000* foram obtidos usando todas as imagens das cole  es de refer ncia como consulta. Os resultados no banco de dados *DB-10000* foram obtidos usando 1000 imagens de consultas, que pertencem   cole   o *Corel-1000*. Para todas as cole  es de refer ncia, todas as imagens que pertencem   mesma categoria (pr -definida) da consulta s o consideradas relevantes, enquanto que as demais imagens s o consideradas irrelevantes.

A Tabela 6.1 mostra os resultados obtidos pelo m todo proposto utilizando as dez diferentes fun  es de aptid o, assim como os resultados obtidos pelos modelos *WLSP-C * e *LSP-C * reportados em [Steji  et al. 2003a]. Nota-se que os resultados obtidos por [Steji  et al. 2003a] utilizando o modelo *LSP-C * s o superiores  queles obtidos com o uso de modelo *WLSP-C *. Os autores atribuem este resultado ao fato do espa o de busca do modelo *LSP-C * ser menor.

Podemos notar que em todos os bancos de dados obtivemos resultados superiores   ambos os modelos: *WLSP-C * e *LSP-C *. O m todo proposto utilizando as fun  es de aptid o $F1_{Lopez-Pujate \text{ et al.}}$, $F2_{Proposta}$, $F2_{Fan \text{ et al.}}$, $F3_{Fan \text{ et al.}}$ e $F4_{Fan \text{ et al.}}$ superam os resultados anteriores em todos os bancos de dados de teste.

Utilizando a mesma fun  o de aptid o utilizada por Steji  et al. foram obtidos resultados inferiores em dois bancos de dados (*Vistex-60* e *Vistex-167*), por m foram obtidos resultados superiores em outros dois (*Brodatz-208* e *Corel-1000*). Como em ambos os AGs s o considerados os mesmos cr terios de p rada, e como   conhecido da literatura que o modo como s o feitas as opera  es gen ticas influencia na performance do AG,   natural que os resultados sejam diferentes, uma vez que n o temos grandes detalhes do AG de [Steji  et al. 2003a].

No momento podemos destacar algumas fun  es de aptid o como $F4_{Fan \text{ et al.}}$, $F2_{Proposta}$, $F3_{Fan \text{ et al.}}$ e $F_{Steji  \text{ et al.}}$, entretanto n o temos um consenso sobre qual   a melhor fun  o de aptid o. Uma conclus o neste momento seria precipitada uma vez que Preci  o-*R* esconde v rios aspectos acerca da efici ncia de recupera  o, uma vez que   tomada a precisi o em apenas um ponto do *ranking*.

Método	Precisão-R (%)				
<i>WLSP-C±</i>	95	84	95	75	—
<i>LSP-C±</i>	98	86	96	76	—
<i>P-F_{Stejić et al.}</i>	90	79	97.8	79.6	58.1
<i>P-F1_{Proposta}</i>	90.5	76.2	97.8	79.6	58.6
<i>P-F1_{Lopez-Pujalte et al.}</i>	98.4	89	99.6	77.5	56.1
<i>P-F2_{Lopez-Pujalte et al.}</i>	97.4	87.5	99.1	70	45
<i>P-F2_{Proposta}</i>	98.4	91	99.5	78	56.4
<i>P-F1_{Fan et al.}</i>	97.6	84	99.6	77.1	56
<i>P-F2_{Fan et al.}</i>	98.4	89.6	99.6	77.3	56
<i>P-F3_{Fan et al.}</i>	98.4	90	99.7	77.5	56.4
<i>P-F4_{Fan et al.}</i>	98.8	89.9	99.5	80.2	57.5
<i>P-F_{P_{AVG}}</i>	89.7	73	98	77.8	37
Banco de Dados	<i>Vistex- 60</i>	<i>Vistex-167</i>	<i>Brodatz-208</i>	<i>Corel-1000</i>	<i>DB-10000</i>

Tabela 6.1: Precisão-*R* da técnica proposta para as várias funções de aptidão experimentadas, em contraste com os modelos *WLSP-C±* e *LSP-C±* de [Stejić et al. 2003a]. Os valores em **negrito** correspondem aos melhores resultados obtidos.

6.2.2 Análise das Funções de Aptidão

Antes de tomar a decisão sobre quais as melhores funções de aptidão alguns experimentos foram conduzidos. O primeiro deles consiste na análise das curvas Precisão-Revocação.

Usando a curva Precisão-Revocação podemos analisar a performance de recuperação em vários pontos do *ranking*. Em cada ponto (r, p) da curva Precisão-Revocação tem-se a precisão para a correspondente taxa de revocação. Por exemplo, o ponto $(0.3, 0.98)$ significa 98% de precisão quando 30% dos documentos relevantes foram recuperados, ou seja, 30% dos documentos relevantes existentes foram recuperados praticamente no topo do *ranking*. Já o ponto $(0.3, 0.6)$, 60% de precisão quando 30% dos documentos relevantes foram recuperados, significa que quando foi recuperado 30% dos documentos relevantes existentes, 40% daqueles retornados eram “lixo”.

A Figura 6.6 mostra as curvas Precisão-Revocação das funções de aptidão sob análise nos bancos de dados *Vistex-167*, *Corel-1000* e *BD-10000*, respectivamente. Novamente, os resultados para os bancos de dados *Vistex-167* e *Corel-1000* foram obtidos usando todas as imagens destas coleções como consulta; já os resultados obtidos para o banco de dados *BD-10000* foram obtidos usando 1000 consultas que pertencem ao banco de dados *Corel-1000*. Em cada consulta o sistema é realimentado com todas as imagens da coleção que pertencem à sua categoria semântica.

Através da Figura 6.6-(a), correspondente ao banco de dados *Vistex-167*, podemos notar que as funções de aptidão baseadas em ordem ($F1$ - $F2_{Lopez-Pujalte et al.}$, $F2_{Proposta}$, $F1$ - $F4_{Fan et al.}$) superam em precisão as não baseadas em ordem ($F_{Stejić et al.}$ e $F1_{Proposta}$) para taxas de revocação abaixo de 0.6. Na Figura 6.6-(b), correspondente ao banco de dados *Corel-1000*, as funções de aptidão baseadas em ordem superam as não baseadas em ordem para níveis de revocação abaixo de 0.4. Este mesmo fato ocorre na Figura 6.6-(c), relativa ao experimento no banco de dados *BD-10000*, exceto para a função de aptidão $F4_{Fan et al.}$ onde a precisão cai ra-

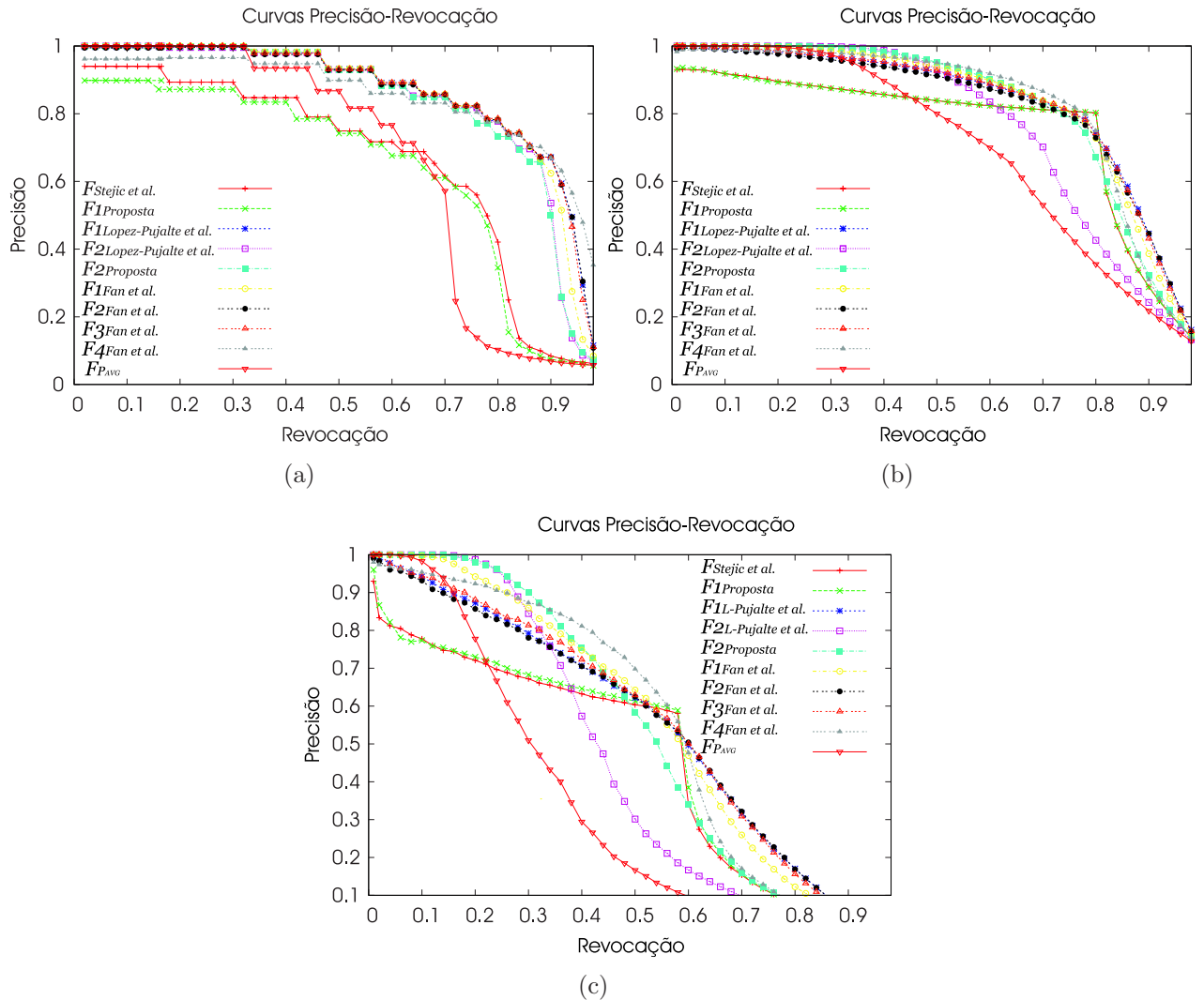


Figura 6.6: Curvas Precisão-Revocação para as várias funções de aptidão nos bancos de dados, *Vistex-167*, *Corel-1000* e *BD-10000*, respectivamente.

pidamente quando prosseguimos no *ranking*. Desta forma, um fato em comum pode ser notado nas três coleções de referências: para baixos níveis de revocação, as funções de aptidão baseadas em ordem apresentam melhores resultados de recuperação que as não baseadas em ordem. Tal fato é de grande importância visto que os usuários dão maior ênfase às primeiras posições do *ranking*. Este comportamento do usuário é motivado pelo seu próprio conceito de *ranking* onde, “os melhores” ocupam as primeiras posições. Ou seja, dado que os documentos são ordenados por similaridade à consulta, os mais similares devem ser os melhores classificados.

As funções de aptidão baseadas em ordem obtêm resultados mais precisos em baixos níveis de revocação devido à posição no *ranking* dos documentos relevantes ser levada em conta, de forma que quanto mais próximo do topo do *ranking* um documento relevante aparecer, melhor será a sua avaliação. Tal critério de avaliação realizado por várias gerações faz com que aqueles indivíduos (soluções) que produzem uma alta concentração de imagens relevantes nas primeiras posições do *ranking* sobrevivam e reproduzam, dando origem a melhores indivíduos (soluções)

à cada geração.

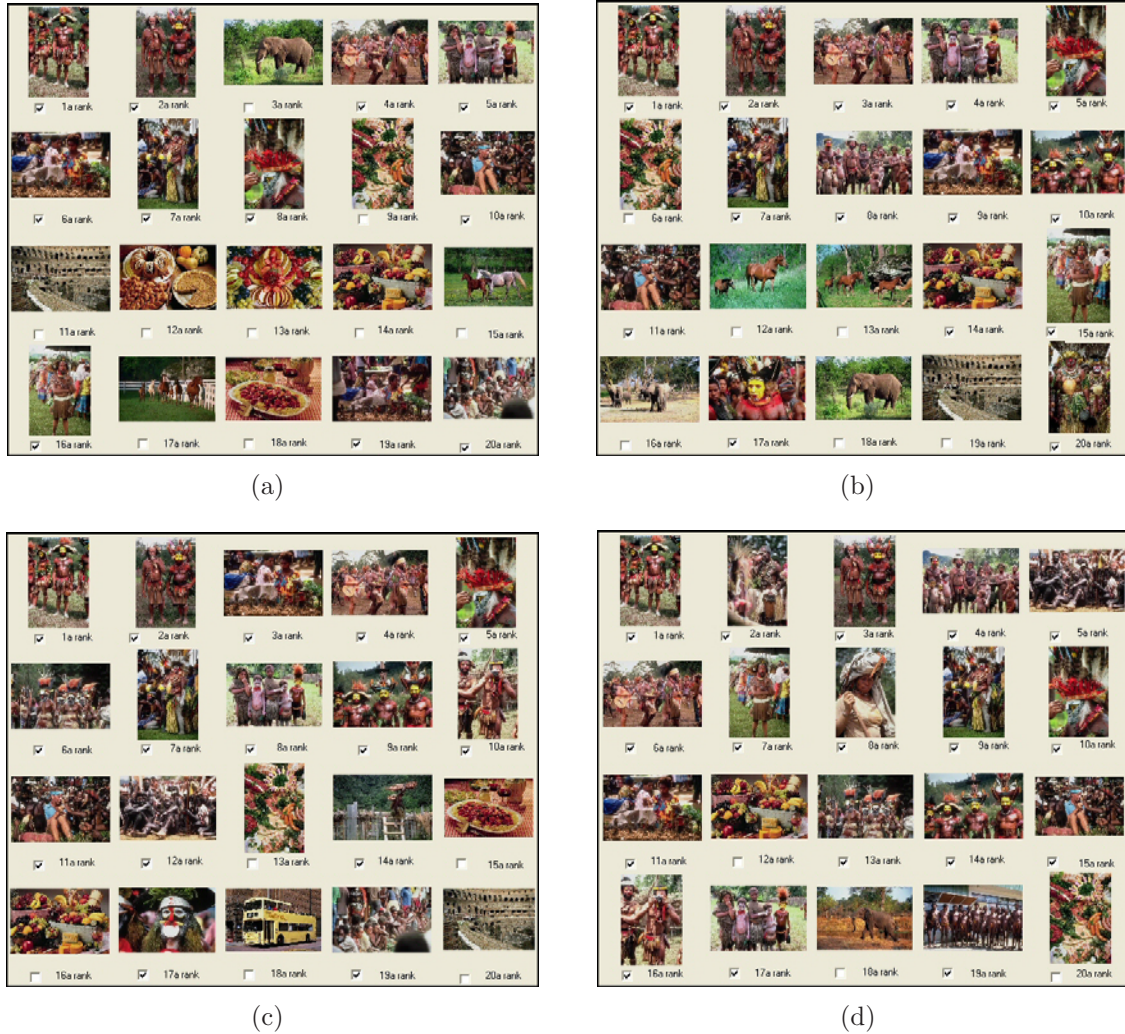


Figura 6.7: Resultados de busca obtidos para a categoria África no banco de dados *Corel-1000*, através das funções de aptidão $F_{Stejic et al.}$, $F1_{Proposta}$, $F2_{Proposta}$ e $F3_{Fan et al.}$, respectivamente

As Figuras 6.7(a)-(d) ilustram resultados de recuperação para a categoria África usando as funções de aptidão $F_{Stejic et al.}$, $F1_{Proposta}$, $F2_{Proposta}$ e $F3_{Fan et al.}$, respectivamente. O experimento foi realizado no banco de dados *Corel-1000*, sendo a consulta a primeira imagem nos *rankings*. O sistema foi realimentado com todas as imagens relevantes entre as vinte primeiras no *ranking* inicial. Podemos notar que as funções de aptidão baseadas em ordem ($F2_{Proposta}$ e $F3_{Fan et al.}$) obteve maior precisão entre as vinte primeiras imagens do *ranking*, do que as não baseadas em ordem ($F_{Stejic et al.}$ e $F1_{Proposta}$) onde: usando as funções de aptidão $F2_{Proposta}$ e $F3_{Fan et al.}$ foram recuperadas cinco e três imagens, respectivamente, que não pertencem à categoria África, enquanto que usando as funções de aptidão $F_{Stejic et al.}$ e $F1_{Proposta}$ foram recuperadas nove e sete imagens, respectivamente, que não pertencem à categoria África.

Todos os resultados obtidos até então indicam que as funções de aptidão baseadas em ordem são superiores às não baseadas em ordem, pois as mesmas produzem resultados superiores para

baixos níveis de revocação. Contudo, faz-se necessário verificar se estas exigem uma maior carga computacional. Para isto estabelecemos que o único critério de parada do algoritmo genético é atingir 100 gerações e então, o tempo médio gasto no processo evolutivo foi medido. Tal experimento foi conduzido no banco de dados *BD-10000* sendo o tempo gasto ao usar cada medida de aptidão dado pela média de três execuções. No experimento foi utilizado um processador Pentium 4 de 3.0 GHz e 512 MB de memória, com sistema operacional Windows XP, sendo que a implementação é feita em linguagem C. A Tabela 6.2 mostra os resultados obtidos, sendo o tempo dado em segundos. Podemos observar que as funções de aptidão não baseadas em ordem $F_{Stejic\ et\ al.}$ e $F1_{Proposta}$ gastam menos tempo em uma evolução por 100 gerações. Tal resultado é explicado pelo fato destas não analisarem todo o *ranking*, ao contrário das funções baseadas em ordem.

Função de Aptidão	Tempo de Execução (seg.)
$F_{Stejic\ et\ al.}$	45.390
$F1_{Proposta}$	45.274
$F1_{Lopez-Pujalte\ et\ al.}$	52.031
$F2_{Lopez-Pujalte\ et\ al.}$	55.437
$F2_{Proposta}$	55.442
$F1_{Fan\ et\ al.}$	55.547
$F2_{Fan\ et\ al.}$	55.828
$F3_{Fan\ et\ al.}$	55.687
$F4_{Fan\ et\ al.}$	55.672
$F_{P_{AVG}}$	55.875

Tabela 6.2: Tempo médio de execução usando as várias funções de aptidão experimentadas.

Uma outra questão importante é saber qual dos grupos de funções de aptidão impõe ao AG uma maior velocidade de convergência, ou seja, através de quais funções de aptidão o processo evolutivo se dá de forma mais rápida.

Na tentativa de responder a esta questão usamos a função de aptidão não baseada em ordem ($F_{Stejic\ et\ al.}$) e a função de aptidão baseada em ordem ($F2_{Proposta}$) para avaliar a precisão de recuperação e a velocidade de convergência da algoritmo genético. O experimento aqui exposto foi realizado no banco de dados *BD-10000* e o sistema foi realimentado com todas as imagens relevantes entre as vinte primeiras. As Figuras 6.8-(a) e 6.8-(b) mostram os resultados obtidos em uma busca por leopardos quando usamos as funções de aptidão $F_{Stejic\ et\ al.}$ e $F2_{Proposta}$, respectivamente. Podemos observar que através da função de aptidão $F2_{Proposta}$ obtivemos uma melhor performance de recuperação nas vinte primeiras imagens do *ranking*, com nenhuma imagem irrelevante, enquanto que através da função de aptidão $F_{Stejic\ et\ al.}$ obtivemos três imagens que não pertencem à categoria leopardo. Através da curva de evolução do algoritmo genético (Figura 6.8-(c)) podemos observar que com o uso da função de aptidão $F2_{Proposta}$ o algoritmo tem uma velocidade de convergência bem maior do que com o uso da função de aptidão $F_{Stejic\ et\ al.}$. Pode ser notado que com o uso da função de aptidão $F2_{Proposta}$ o algoritmo genético encontra o valor de aptidão máximo em noventa gerações, enquanto que com o uso da função de aptidão $F_{Stejic\ et\ al.}$ o algoritmo genético irá precisar de um número muito maior de gerações para encontrar o valor de aptidão máximo. Tal resultado pode ser explicado

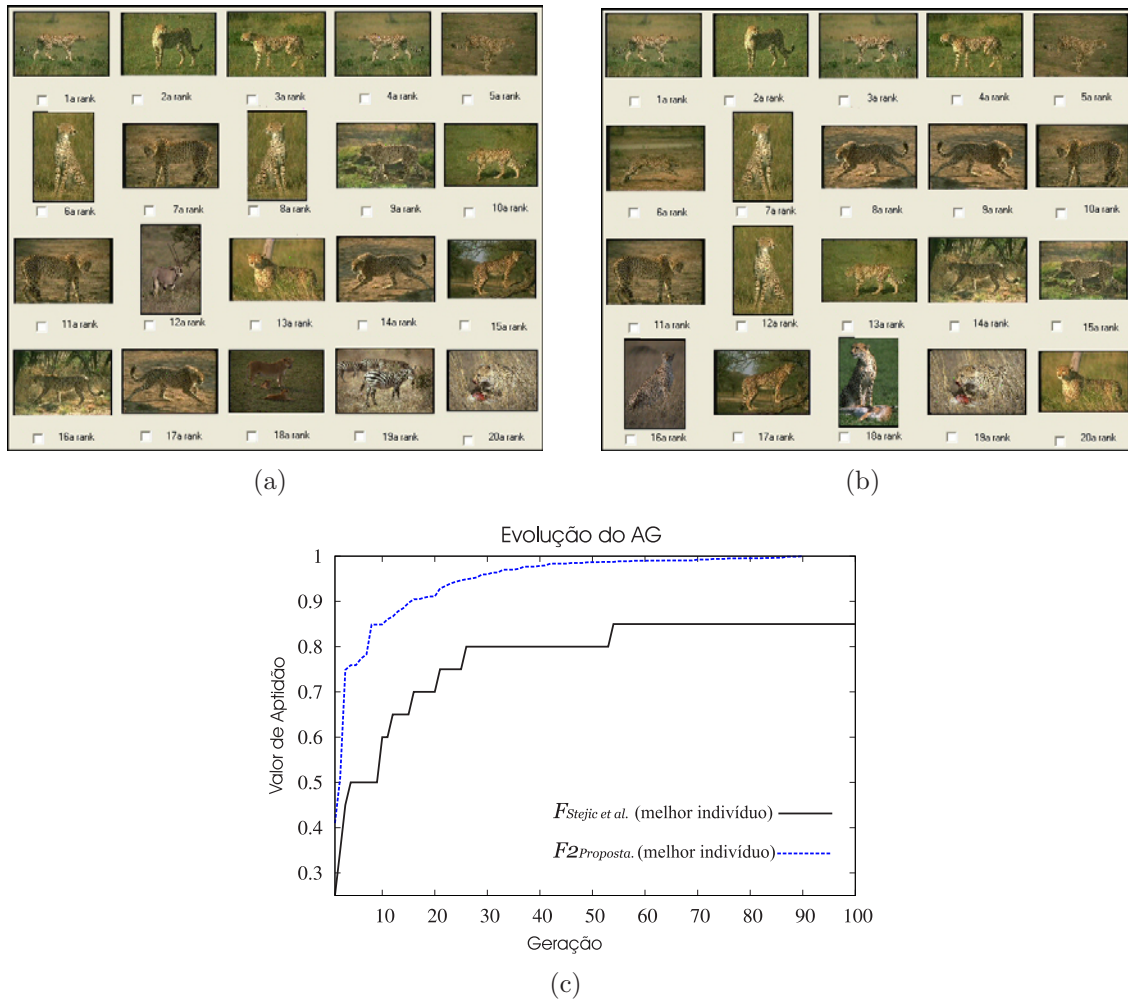


Figura 6.8: Resultado de busca obtido para a categoria leopardo, no banco de dados *BD-10000*. (a) usando a função de aptidão $F_{Stejic et al.}$. (b) usando a função de aptidão $F_{2Proposta}$. (c) evolução do algoritmo genético usando $F_{Stejic et al.}$ e $F_{2Proposta}$.

pela análise dos dois grupos de funções de aptidão, baseadas em ordem e não baseadas em ordem, feito na penúltima seção do Capítulo 4, de onde podemos concluir que devido à melhor diferenciação dos indivíduos feita pelas funções de aptidão baseadas em ordem a evolução ocorre de forma mais rápida.

Para reforçar a evidência de que as funções de aptidão baseadas em ordem faz com que o algoritmo genético tenha uma maior velocidade de convergência, foi realizado um segundo experimento. Novamente usamos a função de aptidão não baseada em ordem $F_{Stejic et al.}$ e a função de aptidão baseada em ordem $F_{2Proposta}$. Desta vez comparamos o número médio de gerações gasto pelo algoritmo genético na busca pelo valor de aptidão máximo. Neste experimento os indivíduos evoluem por até 350 gerações, caso o valor máximo de aptidão não seja encontrado. Em uma busca, se o valor máximo de aptidão não for encontrado ao longo de 350 gerações, pára-se o processo de busca e atribui à mesma um custo de 350 gerações. Pela formulação de ambas as funções de aptidão sabemos que a aptidão máxima possível é 1 e

Categoria	Função $F_{Stejjic\ et\ al.}$ (número de gerações)	Função $F2_{Proposta}$ (número de gerações)
África	324	82
Praia	303	51
Edifícios	174	39
Ônibus	95	25
Dinossauros	242	47
Elefantes	335	98
Flores	17	15
Comidas	135	87
Cavalos	21	16
Montanhas	331	117
Média	198	58

Tabela 6.3: Número de gerações médio gasto pelo AG na busca do valor de aptidão máximo usando as funções de aptidão $F2_{Proposta}$ e $F_{Stejjic\ et\ al.}$.

que a aptidão de um indivíduo pode variar no intervalo $[0,1]$. Na condução deste experimento foram realizadas 100 consultas no banco de dados *BD-10000* sendo estas obtidas através de uma seleção randômica de 10% para cada uma das dez categorias de imagem do banco de dados *Corel-1000*. O sistema é realimentado com as dez primeiras imagens relevantes no *ranking* inicial. A Tabela 6.3 mostra o número médio de gerações gasto pelo algoritmo genético, em cada categoria. Por esta tabela podemos verificar que a função de aptidão baseada em ordem $F2_{Proposta}$ é muito superior à função de aptidão não baseada em ordem $F_{Stejjic\ et\ al.}$ quanto ao número de gerações gasto para conduzir o AG na busca por uma solução ótima conforme a medida de aptidão.

Finalmente, através da fusão dos resultados das Tabelas 6.2 e 6.3 chegamos à seguinte conclusão: *as funções de aptidão baseadas em ordem produzem melhores resultados para baixa revocação e conduzem mais rapidamente o AG na busca por uma solução heurísticamente ótima, conforme a medida de aptidão.* Embora a Tabela 6.2 apresente que as funções não baseadas em ordem são mais rápidas considerando um número fixo de gerações, a diferença não é tão significativa quanto ao número de gerações médio. Ou seja, na prática as funções de aptidão baseadas em ordem são muito mais rápidas que as não baseadas em ordem, uma vez que com estas o AG gasta aproximadamente três vezes menos gerações. Entre as funções de aptidão baseadas em ordem podemos destacar as funções $F2_{Proposta}$, $F1_{Fan\ et\ al.}$ e $F4_{Fan\ et\ al.}$ com base principalmente nas curvas Precisão-Revocação da Figura 6.6.

6.2.3 Outros Resultados

Realizamos uma série de experimentos dos quais selecionamos alguns que serão reportados nesta seção. Em todos os experimentos reportados o *ranking* inicial considerando que todas as característica e regiões têm iguais importâncias, isto é, com todos os pesos, de características e de regiões, iguais a 1.

Os dois primeiros experimentos desta seção ainda têm como objetivo comparar as funções de aptidão baseadas em ordem com as não baseadas em ordem através das funções $F2_{Proposta}$ e

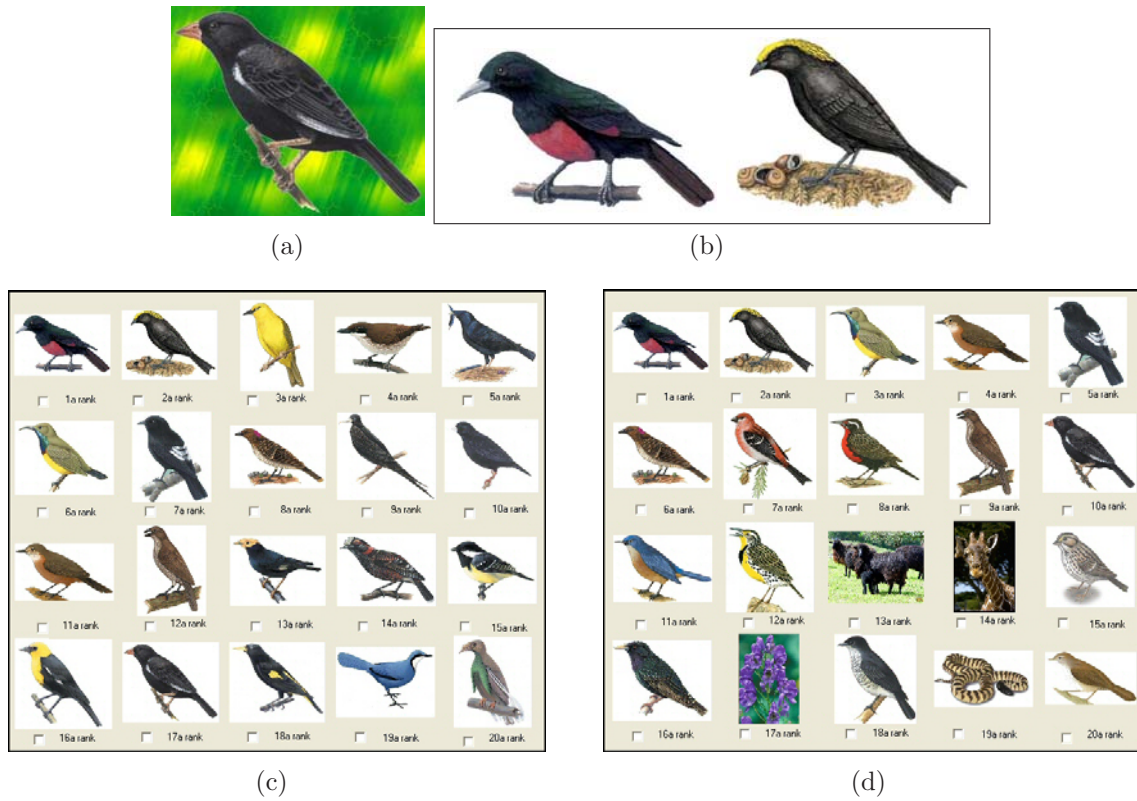


Figura 6.9: Resultado de uma busca por pássaros no banco de dados *BD-12750*; (a) imagem de consulta; (b) imagens usadas na realimentação do sistema; (c) e (d) resultados obtidos usando as funções de aptidão $F2_{Proposta}$ e $F_{Stejic et al.}$, respectivamente.



Figura 6.10: Resultado de uma busca por cavalos no banco de dados *BD-12750*. A imagem da primeira posição do *ranking* é a consulta; (a) resultado obtido usando a função de aptidão $F2_{Proposta}$; (b) resultado obtido usando a função de aptidão $F_{Stejic et al.}$.

$F_{Stejjic\ et\ al.}$. No primeiro destes experimentos pegamos uma imagem de um pássaro da coleção de referência (*BD-12750*), alteramos esta inserindo artificialmente um plano de fundo verde-amarelo e a utilizamos como consulta. Nossa intenção é avaliar o desempenho do método quando o plano de fundo não é importante, uma vez que nenhuma das imagens da categoria pássaros da coleção de referência possui plano de fundo verde-amarelo. Considerando que o usuário não estaria interessado na informação de fundo da imagem, mas sim nas informações internas de forma e textura, as imagens fornecidas como relevantes no *ranking* inicial, consistem de dois primeiros pássaros (Figura 6.9(b)). As Figuras 6.9-(c) e (d) apresentam os resultados obtidos usando as funções de aptidão $F_{2Proposta}$ e $F_{Stejjic\ et\ al.}$, respectivamente. Podemos observar que através da função de aptidão baseada em ordem $F_{2Proposta}$, todas as imagens, entre as vinte primeiras do *ranking*, pertencem à categoria pássaro. Por outro lado, usando a função de aptidão não-baseada em ordem $F_{Stejjic\ et\ al.}$ foram recuperadas quatro imagens entre as vinte primeiras do *ranking* que não pertencem à categoria pássaro. Tal resultado pode ser visto como uma confirmação de que as funções de aptidão baseadas em ordem são superiores àquelas não baseadas em ordem, para baixa revocação.

Nosso segundo experimento, ilustrado pela Figura 6.10, consiste de uma busca por imagens da categoria cavalo. A consulta é a primeira imagem dos *rankings* (Figura 6.10-(a) e Figura 6.10-(b)). Em ambos os resultados o sistema foi realimentado duas vezes com todas as imagens relevantes entre as 20 primeiras posições do *ranking*. A Figura 6.10-(a) mostra o resultado obtido pelo uso da função de aptidão baseada em ordem $F_{2Proposta}$, e a Figura 6.10-(b) mostra o resultado obtido pela função de aptidão não baseada em ordem $F_{Stejjic\ et\ al.}$. Novamente percebemos que a função de aptidão $F_{2Proposta}$ proporciona uma maior precisão entre as vinte primeiras imagens do *ranking*.

No restante desta seção todos os experimentos reportados utilizam a função de aptidão $F_{2Proposta}$.

O terceiro experimento consiste do uso de uma figura de um avião, rotacionada 90 graus e intencionalmente estragada com riscos artificiais como consulta (Figura 6.11-(a)). A Figura 6.11-(b) mostra o resultado inicial obtido juntamente com as imagens assinaladas como relevantes (✓), usadas para realimentar o sistema. Podemos notar neste resultado inicial a presença de oito imagens que não pertencem à categoria avião, entre as vinte primeiras imagens do *ranking*. Depois da realimentação de relevantes o sistema apresenta uma excelente adaptação às características comuns entre as imagens fornecidas como relevantes (conjunto de relevantes), como mostrado na Figura 6.11-(c). No resultado obtido, entre as 40 primeiras imagens do *ranking* somente duas são irrelevantes, contudo são figuras de oceano, onde há uma certa similaridade no plano de fundo. Este experimento mostra a robustez do sistema proposto quando a imagem de consulta contém estragos.

No quarto experimento foi utilizado como consulta uma imagem construída a partir da metade da imagem de um cavalo e metade da imagem de uma montanha (Figura 6.12-(a)). O experimento foi realizado no banco de dados *Corel-1000*, sendo que as classes cavalo e montanha pertencem à esta coleção. A Figura 6.12-(b) mostra o resultado obtido quando o sistema foi realimentado com cinco imagens de cavalo. A Figura 6.12-(c) mostra o resultado obtido quando o sistema foi realimentado com cinco imagens de montanha. E a Figura 6.12-(d) mostra o resultado obtido quando o sistema foi realimentado com três imagens de cavalo e três imagens de montanha.

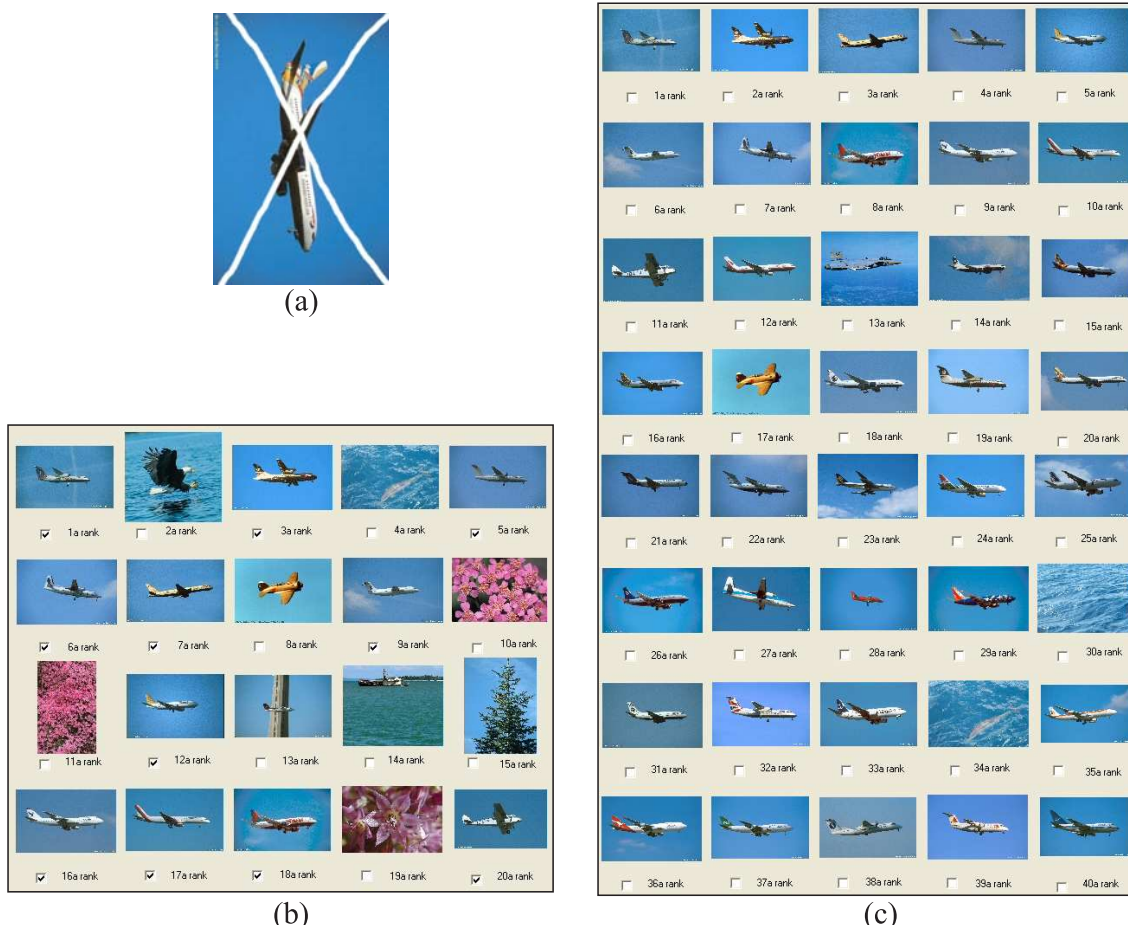


Figura 6.11: Resultado de uma busca por aviões no banco de dados *BD-12750*. (a) Imagem de consulta. (b) *Ranking* inicial com as imagens relevantes assinaladas. (c) Resultado obtido após a realimentação de relevantes.

No quinto experimento temos uma busca por árvores (Figura 6.13). Novamente, a primeira imagem dos *rankings* é a consulta. A Figura 6.13-(a) mostra o *ranking* inicial, sendo as imagens assinaladas (✓) aquelas usadas para realimentar o sistema. A Figura 6.13-(b) mostra o resultado obtido após a realimentação do sistema.

O sexto experimento ilustra a situação onde o usuário deseja fazer uma busca, porém não tem uma imagem exemplo para utilizar como consulta. Assim, uma possibilidade é o usuário fazer um esboço da imagem desejada e utilizar este como consulta. Para ilustrar o funcionamento do sistema nesta situação, realizamos os três testes a seguir.

No primeiro teste foi utilizado como consulta a imagem de um carro desenhado manualmente (Figura 6.14-(a)). A Figura 6.14-(b) mostra o *ranking* inicial obtido. Podemos observar que todas as imagens entre as vinte primeiras pertencem à categoria carro, contudo nem todos são da mesma cor da consulta. A Figura 6.14-(c) mostra o resultado obtido ao realimentar o sistema duas vezes com carros de cor amarela.

No segundo teste, realizado no banco de dados *Corel-1000*, fizemos o esboço de um ônibus e este foi utilizado como consulta. A Figura 6.15-(a) mostra este esboço, juntamente com o

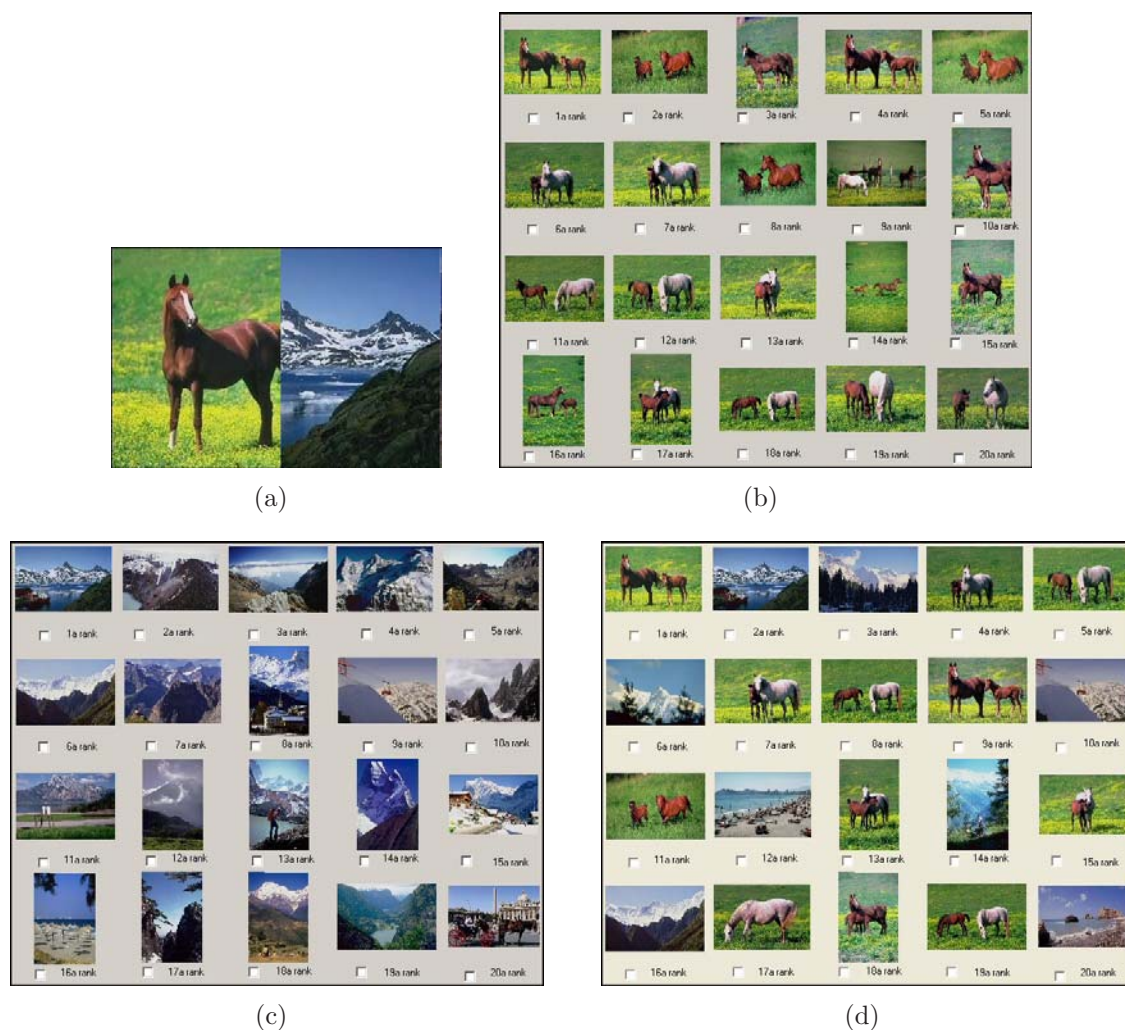


Figura 6.12: Resultado de busca para a imagem de consulta construída pela metade de uma imagem de cavalo e pela metade de uma imagem de montanha. (a) Imagem de consulta. (b) Resultado obtido após realimentar o sistema com cinco imagens de cavalo. (c) Resultado obtido após realimentar o sistema com cinco imagens de montanha. (d) Resultado obtido após realimentar o sistema com três imagens de cavalo e três imagens de montanha.

ranking inicial obtido e as imagens usadas na realimentação do sistema (✓). A figura 6.15-(b) mostra o resultado obtido após a realimentação do sistema.

No terceiro teste, supomos que o usuário deseja fazer uma busca por flores. Assim fizemos o esboço de uma flor e utilizamos este como consulta. A Figura 6.17-(a) mostra a interface do nosso sistema, a imagem de consulta à esquerda, e as imagens usadas na realimentação do sistema (✓). A Figura 6.17-(b) mostra o resultado obtido após a realimentação do sistema.

O último experimento deste capítulo consiste de busca pelo indivíduo indicado como consulta nas Figuras 6.16-(a) e 6.16-(b) no banco de dados *BD-12750*. A Figura 6.16-(a) mostra o resultado no *ranking* inicial e as imagens assinaladas (✓) utilizadas para realimentar o sistema. A Figura 6.16-(b) mostra o resultado obtido após a realimentação.

Outros resultados experimentais são expostos no Apêndice B.

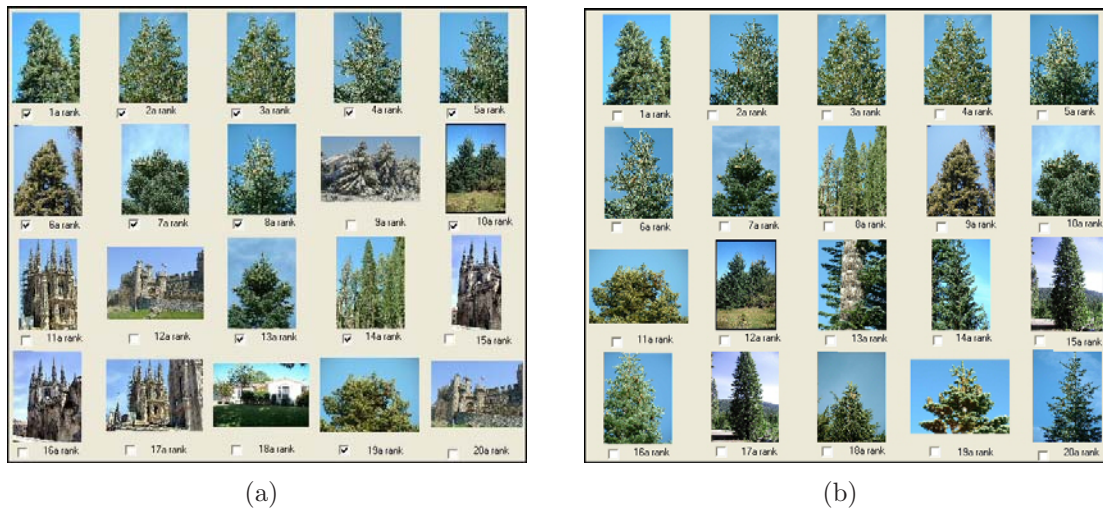
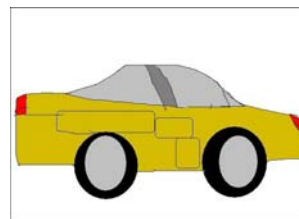
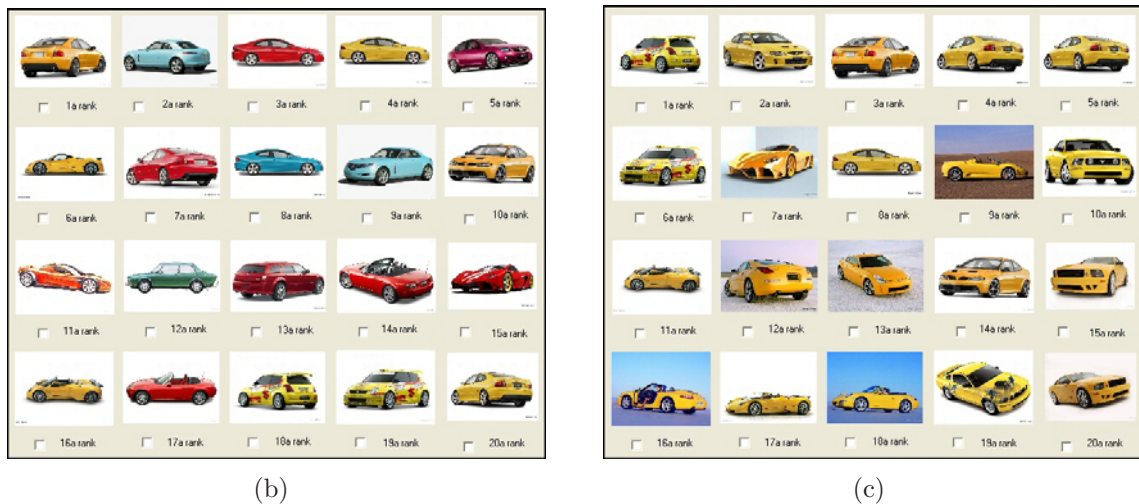


Figura 6.13: Resultado de uma busca por árvores no banco de dados *BD-12750*, usando a função de aptidão $F2_{Proposta}$. A imagem da primeira posição dos *rankings* é a consulta. (a) *ranking* inicial e realimentação do usuário (✓); (b) resultado obtido após a realimentação.



(a)



(b)

(c)

Figura 6.14: Resultado de uma busca por carros no banco de dados *BD-12750*; (a) imagem de consulta; (b) *ranking* inicial; (c) resultado após realimentação com carros de cor amarela.

Nas Figuras 6.15, 6.16 e 6.17, podemos visualizar a interface da nossa aplicação que é feita com a ferramenta MATLAB® 6.5. Todo o processo de computação de similaridade de imagens

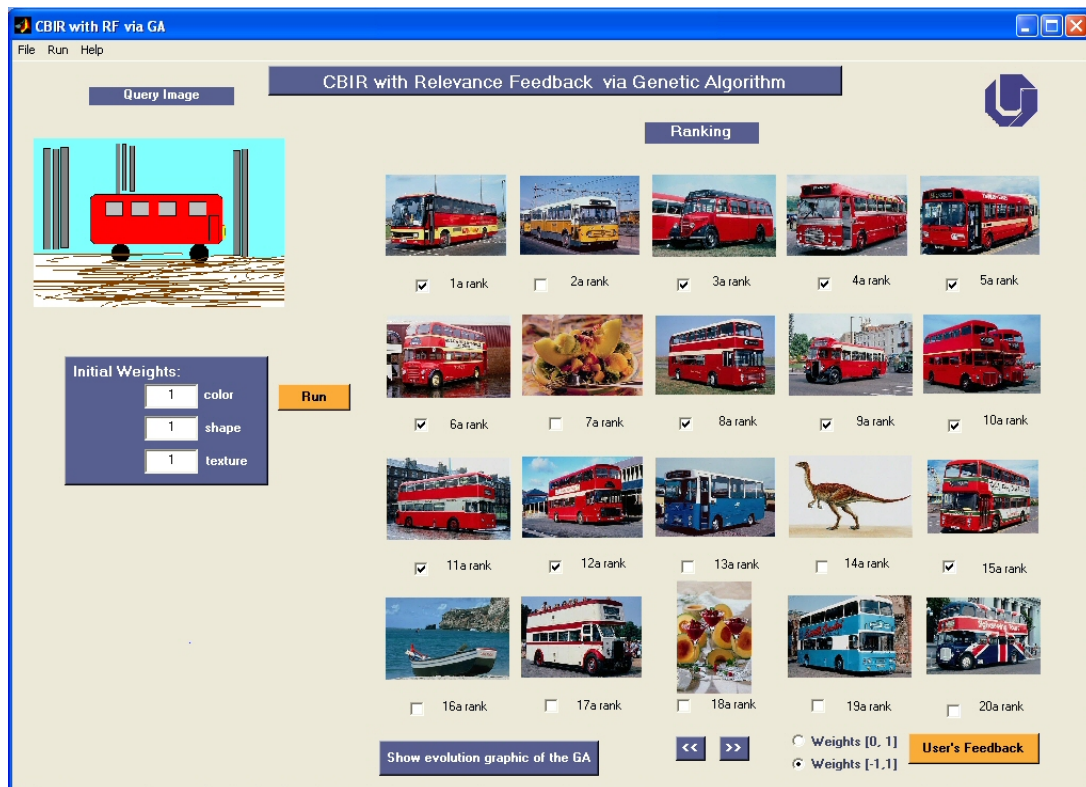
e realimentação de relevantes via algoritmo genético foi feito em Linguagem C. O MATLAB e a Linguagem C se comunicam via *DLL* (*Dynamic Link Library*). Para maiores instruções veja http://www.mathworks.com/access/helpdesk/help/techdoc/matlab_external/index.html?access/helpdesk/help/techdoc/matlab_external.

A interface da nossa aplicação conta com os seguintes recursos:

- (1) o usuário pode indexar (extração de características) um banco de dados, clicando na opção *Run » Index*, na barra de ferramentas, conforme mostrado na Figura 6.18-(a).
- (2) o usuário pode consultar uma determinada coleção de referência a partir de uma imagem de consulta localizada em uma mídia ou em uma *URL* qualquer, através das opções mostradas na Figura 6.18-(b), presentes na barra de ferramentas. Caso o usuário escolha a opção *Read Query* irá aparecer uma segunda janela que permite ao usuário selecionar a imagem de consulta a partir de alguma mídia. Se o usuário escolher a opção *Read Query URL* irá aparecer uma caixa de diálogo onde o usuário deverá inserir o endereço *Web* onde se encontra a imagem de consulta.
- (3) o usuário pode definir as importâncias das características de cor, forma e de textura no *ranking* inicial (Figura 6.18-(e)), caso contrário será considerado que as características de cor, forma e textura têm iguais importâncias. Em seguida o usuário deve executar a busca clicando no botão *Run*.
- (4) os botões ilustrados pela Figura 6.18 permitem ao usuário navegar pelo *ranking* de imagens recuperadas.
- (5) o usuário deve selecionar o domínio do conjunto de pesos utilizados na realimentação pela escolha de uma das opções mostradas na Figura 6.18-(d) e, após escolher as imagens relevantes ele deve realimentar o sistema clicando no botão *User's Feedback*.
- (6) o usuário pode visualizar o gráfico de evolução do AG ao clicar no botão *Show evolution graphic of the GA*. Na Figura 6.18-(f) temos um destes gráficos de evolução, que ilustra: a evolução do melhor indivíduo (curva *Best Individual*), a evolução do pior indivíduo (curva *Worst individual*) e a aptidão média da população (curva *Average of the Population*).

6.3 Considerações Finais

Neste capítulo verificamos que nosso método de recuperação de imagens com realimentação de relevantes via algoritmo genético produziu bons resultados, que quando comparados com aos resultados reportados em [Stejić et al. 2003a] mostraram-se superiores. Pelos resultados obtidos pelos experimentos, pela análise da curva precisão-revocação, e pela verificação do custo computacional por um período evolutivo e velocidade de convergência do AG foi detectado que as funções de aptidão baseadas em ordem produzem resultados mais precisos para baixa revocação e conduzem muito mais rapidamente o AG a uma solução heurísticamente ótima.

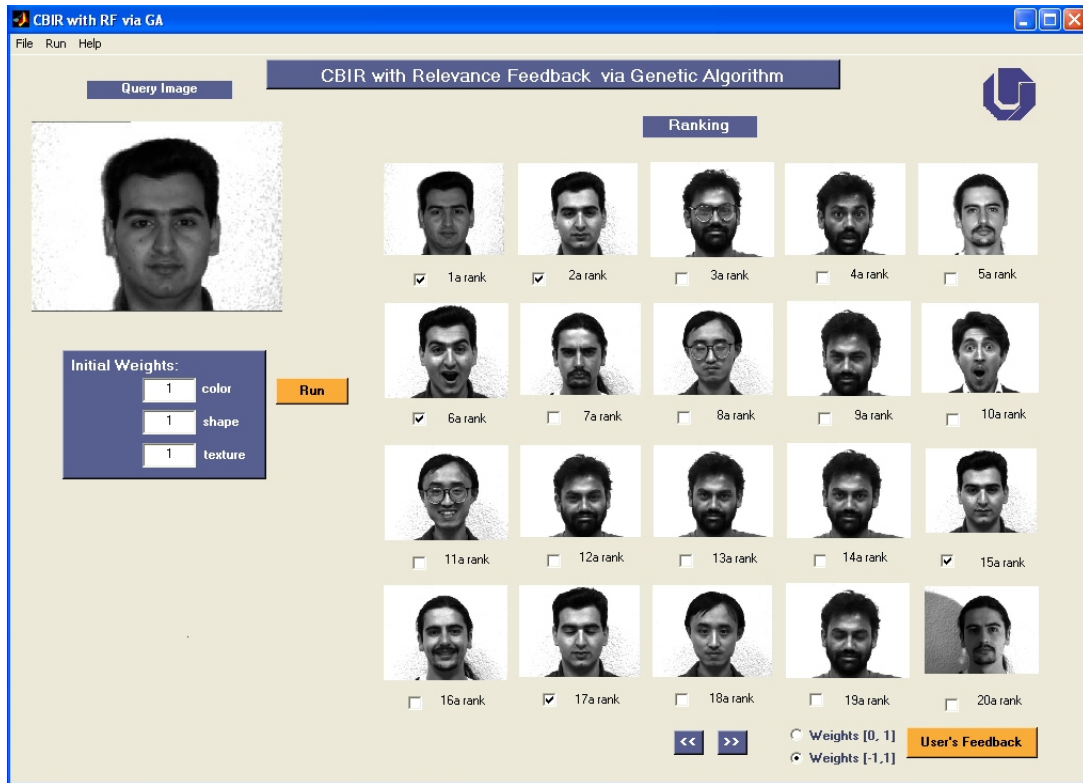


(a)

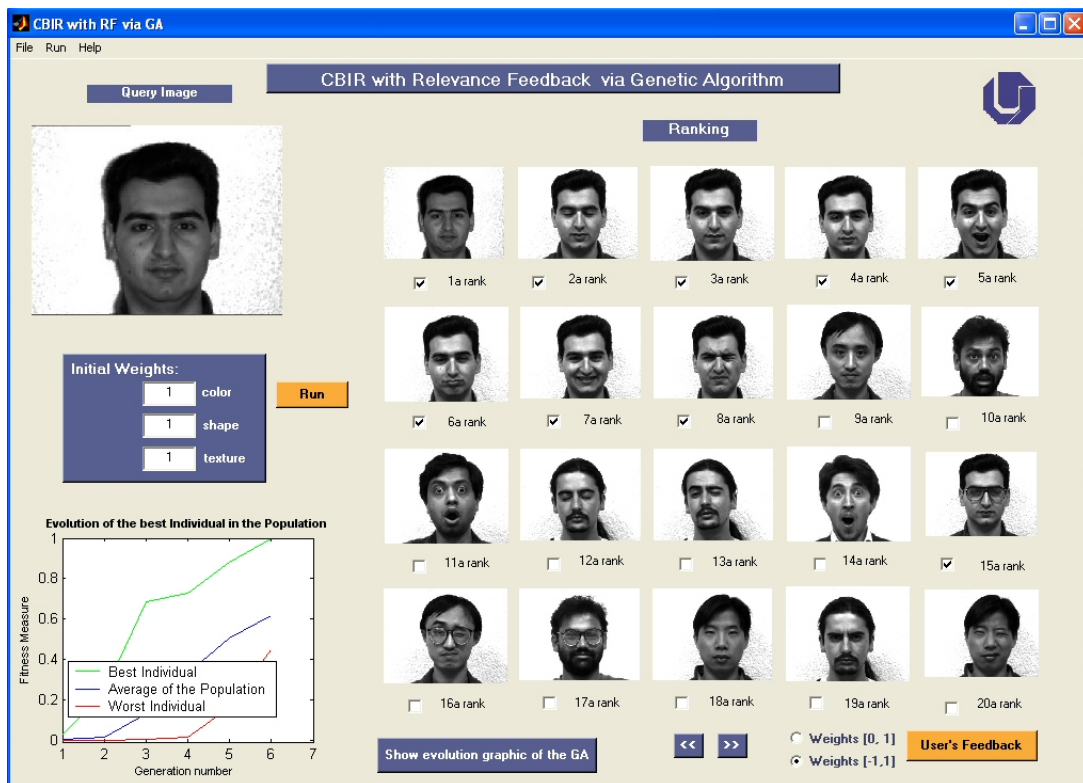


(b)

Figura 6.15: Resultado de uma busca por ônibus no banco de dados *Corel-1000*. A imagem de consulta à esquerda foi desenhada manualmente. (a) *Ranking* inicial obtido sendo que as imagens assinaladas foram usadas para realimentar o sistema. (b) Resultado obtido após a realimentação.

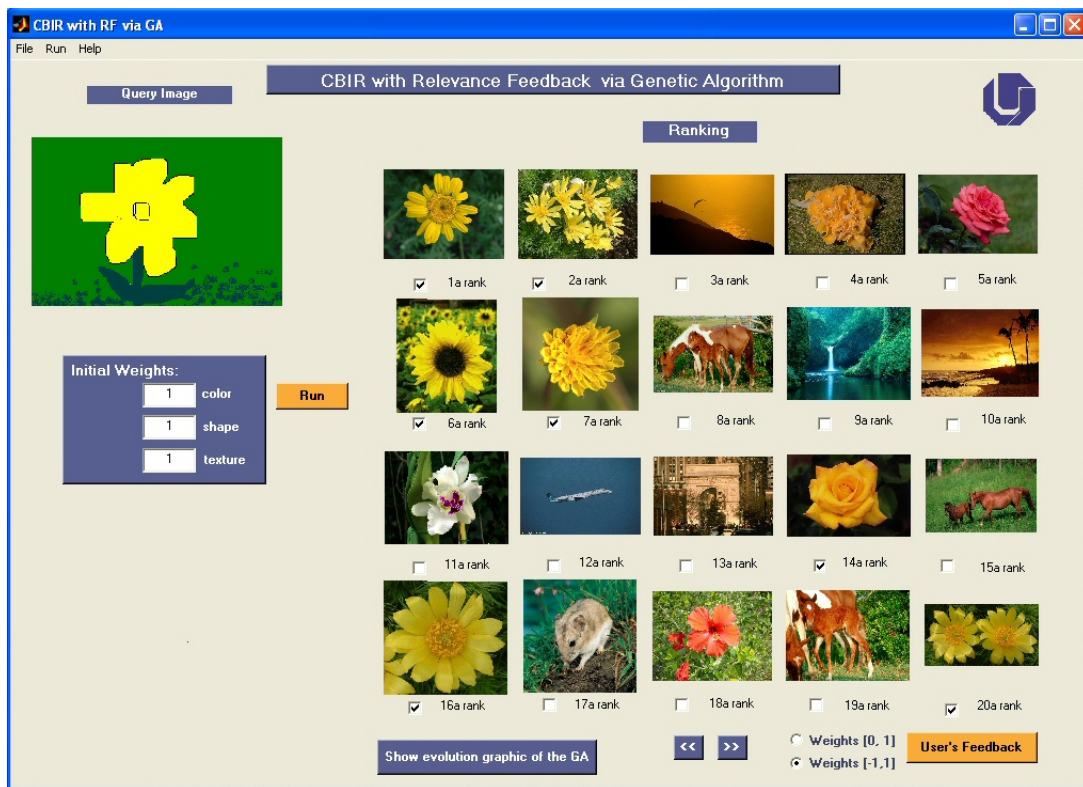


(a)

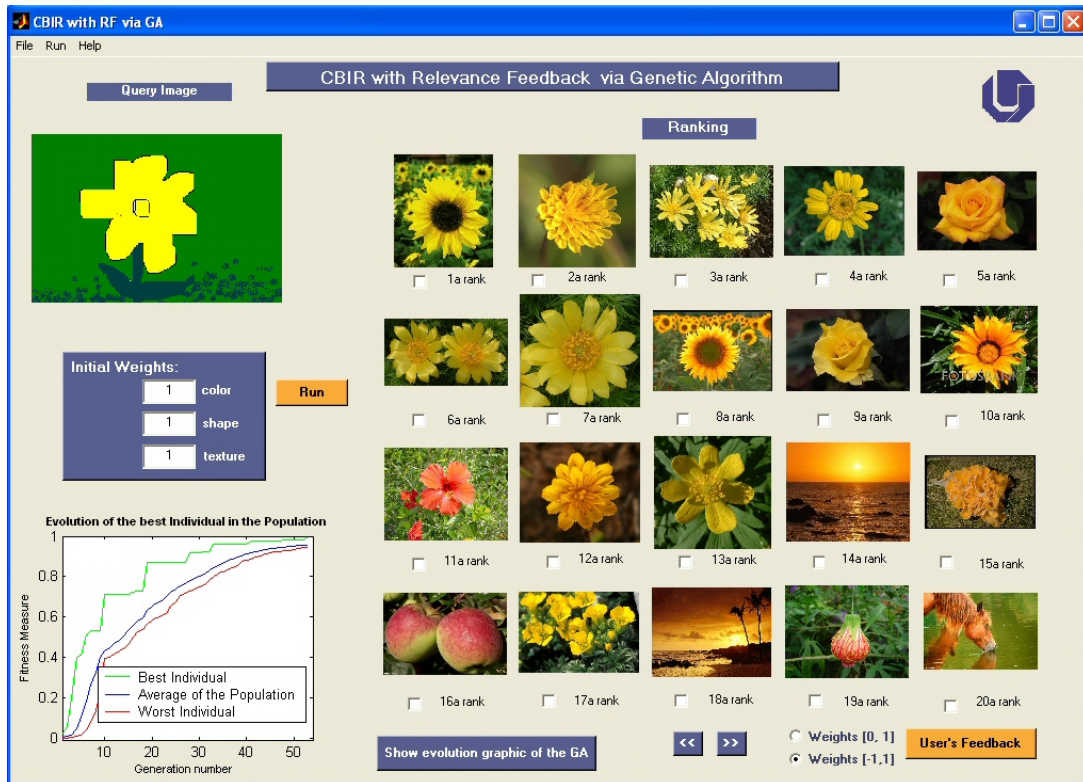


(b)

Figura 6.16: Consultando faces no banco de dados *BD-12750*, usando a função de aptidão $F2_{Proposta}$. (a) Imagem de consulta à esquerda e *ranking* inicial obtido, sendo que as imagens assinaladas (✓) foram as utilizadas para realimentar o sistema. (b) Resultado obtido após a realimentação.



(a)



(b)

Figura 6.17: Resultado de uma busca por flores no banco de dados *BD-12750*. A imagem de consulta à esquerda foi desenhada manualmente. (a) *Ranking* inicial obtido sendo que as imagens assinaladas foram usadas para realimentar o sistema. (b) Resultado obtido após a realimentação.

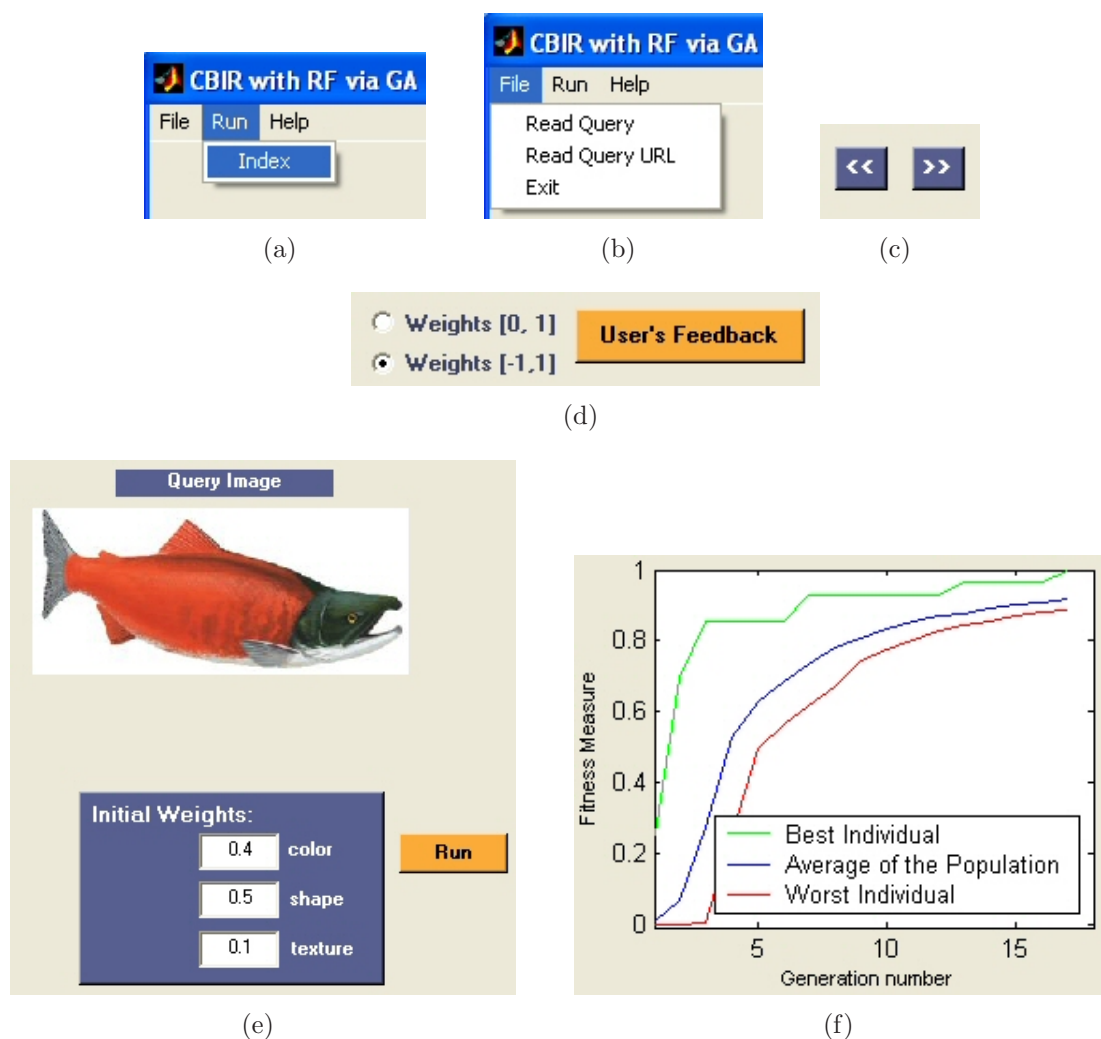


Figura 6.18: Recursos da interface da nossa aplicação; (a) Opções de consulta; (b) Indexação (extração de características) de um banco de dados; (c) Botões para navegação no *ranking*; (e) Pesos (importâncias) das características de cor forma e textura no *ranking* inicial. Após definidas o usuário deve executar a busca clicando no botão *Run*; (d) escolha do domínio de pesos usados no processo de realimentação de relevantes. Após escolhido uma das opções e marcado as imagens relevantes, o usuário deve realimentar o sistema clicando no botão *User's Feedback*; (f) gráfico de evolução do algoritmo genético. Será exibido assim que o usuário clicar no botão *Show evolution graphic of the GA*.

Trabalhos Futuros e Perspectivas

Neste capítulo são propostos uma série de trabalho futuros. A primeira proposta trata-se do uso de algoritmos genéticos com função de aptidão multiobjetivos na condução de um processo de realimentação com exemplos positivos e negativos. Tal abordagem encontra-se em desenvolvimento. Em seguida são apresentadas resumidamente nossas sugestões de trabalhos futuros e nossas expectativas acerca da contribuição destes.

7.1 Realimentação via Algoritmo Genético usando Função de Aptidão Multiobjetivos

O desenvolvimento desta abordagem parte da disponibilização ao usuário de uma interface onde é possível identificar as imagens previamente recuperadas como exemplos positivos (relevantes) e exemplos negativos (indesejáveis) (ver Figura 7.1). A meta é aproximar as imagens relevantes das primeiras posições do *ranking* e ao mesmo tempo, afastar as imagens indesejáveis das primeiras posições do *ranking*.

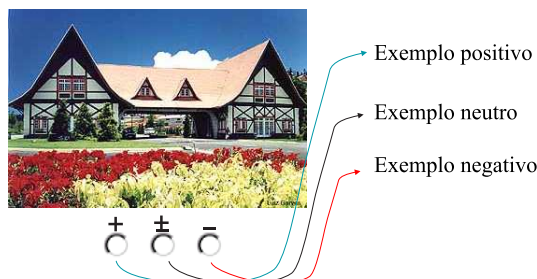


Figura 7.1: Interface básica para a classificação de exemplos em positivos e negativos.

Para cumprir a meta preestabelecida é proposto o uso de algoritmos genéticos com funções de aptidão multiobjetivos.

7.1.1 Algoritmos Genéticos com Função de Aptidão Multiobjetivo

A função de aptidão passa a ser composta de múltiplas funções objetivo que devem ser combinadas de forma a resultar em um único valor de aptidão. Nesta proposta são considerados dois casos clássicos de combinação de funções (avaliações) multiobjetivos:

Caso 1:

$$F = p_1 F_1 + (1 - p_1) F_2 \quad (7.1)$$

onde $p_1 \in \mathbb{R}$; $0 \leq p_1 \leq 1$ indica a importância de atender ao objetivo 1, $(1 - p_1)$ indica a importância de atender ao objetivo 2, F_1 é a função de avaliação do objetivo 1 e F_2 é a função de avaliação do objetivo 2.

Caso 2:

$$F = F_1 F_2 \quad (7.2)$$

Aqui é suposto que os valores de F_1 e F_2 estejam no intervalo real de $[0, 1]$.

Outras formulações matemáticas podem ser utilizadas para compor funções de aptidão envolvendo múltiplos objetivos.

7.1.2 Exemplos Positivos e Negativos: Otimização Multiobjetivo

O presente problema é tradicionalmente conhecido como realimentação de relevantes com exemplos positivos e negativos. Neste contexto, são definidos dois objetivos supostamente inversos: aproximar as imagens relevantes do topo *ranking* e ao mesmo tempo, afastar as imagens indesejáveis do início do *ranking*, ou seja, é definido um problema de otimização multiobjetivo.

Em termos de otimização, podemos pensar em duas funções objetivo inversas. Chamaremos de F_1 a função que avalia o quão próximo as imagens relevantes estão do topo do *ranking* e de F_2 a função que avalia o quão distante as imagens indesejáveis estão do topo do *ranking*.

Vamos supor que F_1 seja computada da mesma forma que $F1_{Proposta}$ descrita no Capítulo 3:

$$F_1(q, \omega) = \frac{Utilidade1(q, \omega)}{\sum_{j=1}^{|P|} \frac{1}{j}} \quad (7.3)$$

onde,

$$Utilidade1(q, \omega) = \sum_{\forall i \in P} \frac{1}{\text{Posição de } i \text{ no ranking}(q, \omega)} \quad (7.4)$$

onde $ranking(q, \omega)$ representa o *ranking* das imagens recuperadas para a consulta q , usando o indivíduo ω . P é o conjunto de imagens indicadas pelo usuário como exemplos positivos, $|P|$ indica o número de elementos do conjunto P , $\sum_{j=1}^{|P|} \frac{1}{j}$ representa a utilidade máxima possível do *ranking* para q com respeito ao conjunto P e $Utilidade(q, \omega)$ representa a estimacão de utilidade do *ranking*(q, ω).

Adicionalmente supomos que F_2 seja dada como a seguir:

$$F_2(q, \omega) = \frac{Utilidade2(q, \omega)}{|N|P_F^2} \quad (7.5)$$

onde

$$Utilidade2(q, \omega) = \sum_{\forall i \in N} U_i \quad (7.6)$$

sendo

$$U_i = \begin{cases} pos(i)^2, & \text{se } pos(i) \leq P_F \\ P_F^2, & \text{caso contrário} \end{cases} \quad (7.7)$$

onde N é o conjunto de imagens marcadas pelo usuário como indesejáveis, $|N|$ é o número de elementos deste conjunto, $pos(i)$ é a posição da imagem i no $ranking(q, \omega)$, $P_F \in \mathbb{N}^*$ é uma constante denominada ponto de fuga.

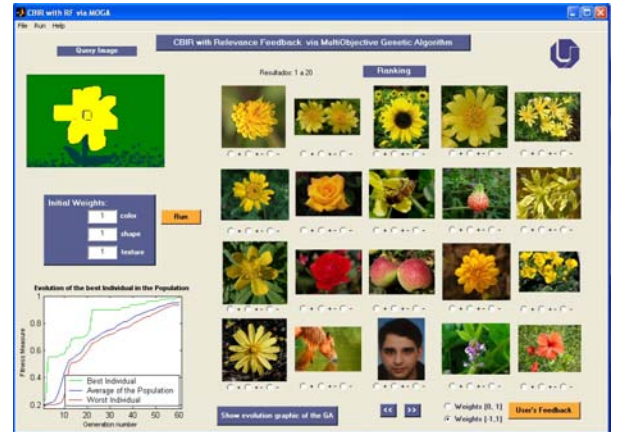
7.1.3 Resultados Preliminares

Devido à dificuldade de automaticamente identificar exemplos positivos e negativos não avaliamos quantitativamente a técnica proposta. Posteriormente pretendemos realizar experimentos com usuários reais como forma de validação da técnica proposta. Contudo testes iniciais no banco de dados *BD-12750* nos trazem boas expectativas.

A Figura 7.2 ilustra uma busca por flores utilizando a combinação das funções objetivo pelo caso 1, onde consideramos $p_1 = 0.7$. Na Figura 7.2-(a) temos a imagem consulta (desenhada manualmente), o *ranking* inicial (onde todos os atributos de imagem têm a mesma importância), e a realimentação do usuário; na Figura 7.2-(b) temos o resultado obtido após a realimentação do sistema.



(a)



(b)

Figura 7.2: Resultado de uma busca por flores no banco de dados *BD-12750*. (a) *Ranking* inicial e realimentação do usuário. (b) Resultado obtido após a realimentação.

A Figura 7.3 ilustra uma busca por interiores de igrejas, onde utilizado a combinação de

objetivos pelo caso 2 e realizamos duas realimentações consecutivas do sistema. Figura 7.3-(a) ilustra o *ranking* inicialmente obtido, juntamente com a realimentação do usuário, Figura 7.3-(b) ilustra o resultado obtido e a nova realimentação do sistema e, finalmente, a Figura 7.3-(c) mostra o resultado final obtido.

A Figura 7.4 ilustra um resultado de uma busca por zebra pela combinação dos objetivos através do caso 1. O sistema foi realimentado consecutivas vezes através da avaliação das vinte primeiras imagens.

7.1.4 Complemento da Proposta

Complementarmente propomos analisar a performance das várias funções de aptidão experimentadas ao longo deste trabalho, no contexto do presente problema de otimização multiobjetivo. Para tal aplicação, cada uma das funções de aptidão serão mapeadas em duas funções objetivo inversas entre si. A função que avaliará o objetivo 1 – aproximar as imagens relevantes do topo do *ranking* – será dada pelas várias funções de aptidão descritas no Capítulo 3; já a função que avaliará o objetivo 2 – afastar as imagens indesejáveis das primeiras posições do *ranking* – será dado pelas mesmas funções de aptidão, porém invertidas. Acreditamos que as melhores funções identificadas anteriormente, darão origem às melhores funções de aptidão no contexto multiobjetivo.

7.2 Trabalhos Futuros

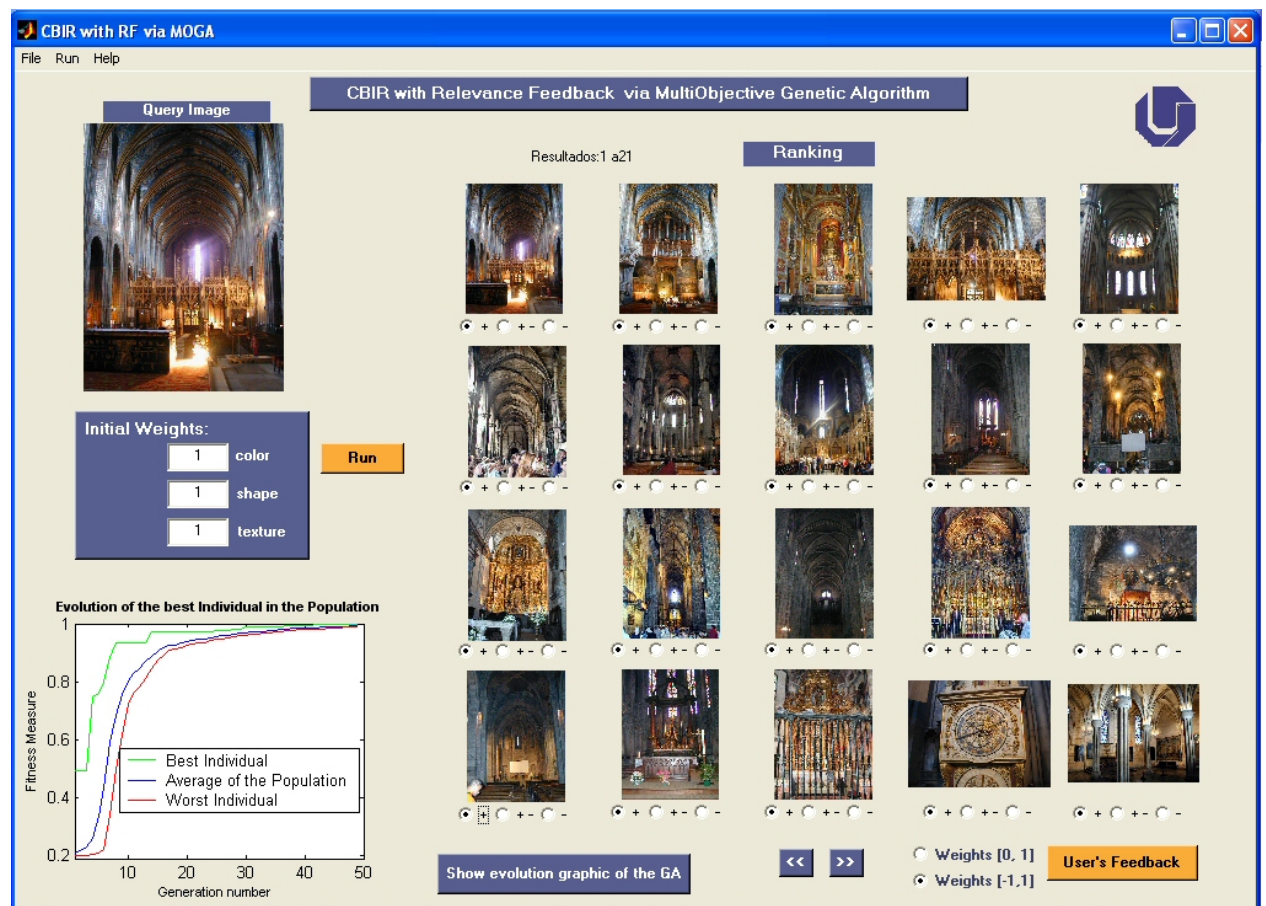
- Desenvolver um robô coletor de imagens na *Web*. Propomos coletar somente o *link* e as características das imagens de forma a reduzir o espaço de armazenamento.
- Implementar vários descritores de cor, forma e textura, e utilizar a abordagem de algoritmos genéticos para decidir os melhores ou a importância de cada um destes no contexto de uma busca particular do usuário.
- Comparar a performance entre as abordagens de Algoritmos Genéticos e de Programação Genética na combinação de descritores de imagens.
- Comparar a abordagem de realimentação de relevantes bioinspirada com outras abordagens.
- Estudar e analisar a performance de técnicas de indexação no sistema desenvolvido.
- Implementar o CBIR em uma arquitetura paralela, principalmente o Algoritmo Genético.
- Realizar uma série de experimentos com usuários reais, com o intuito de verificar e comparar a performance dos sistemas.



(a)



(b)



(c)

Figura 7.3: Resultado de uma busca por interiores de igrejas no banco de dados *BD-12750*; (a) *Ranking* inicial e realimentação do usuário; (b) Resultado obtido após a primeira realimentação; (c) Resultado obtido após a segunda realimentação

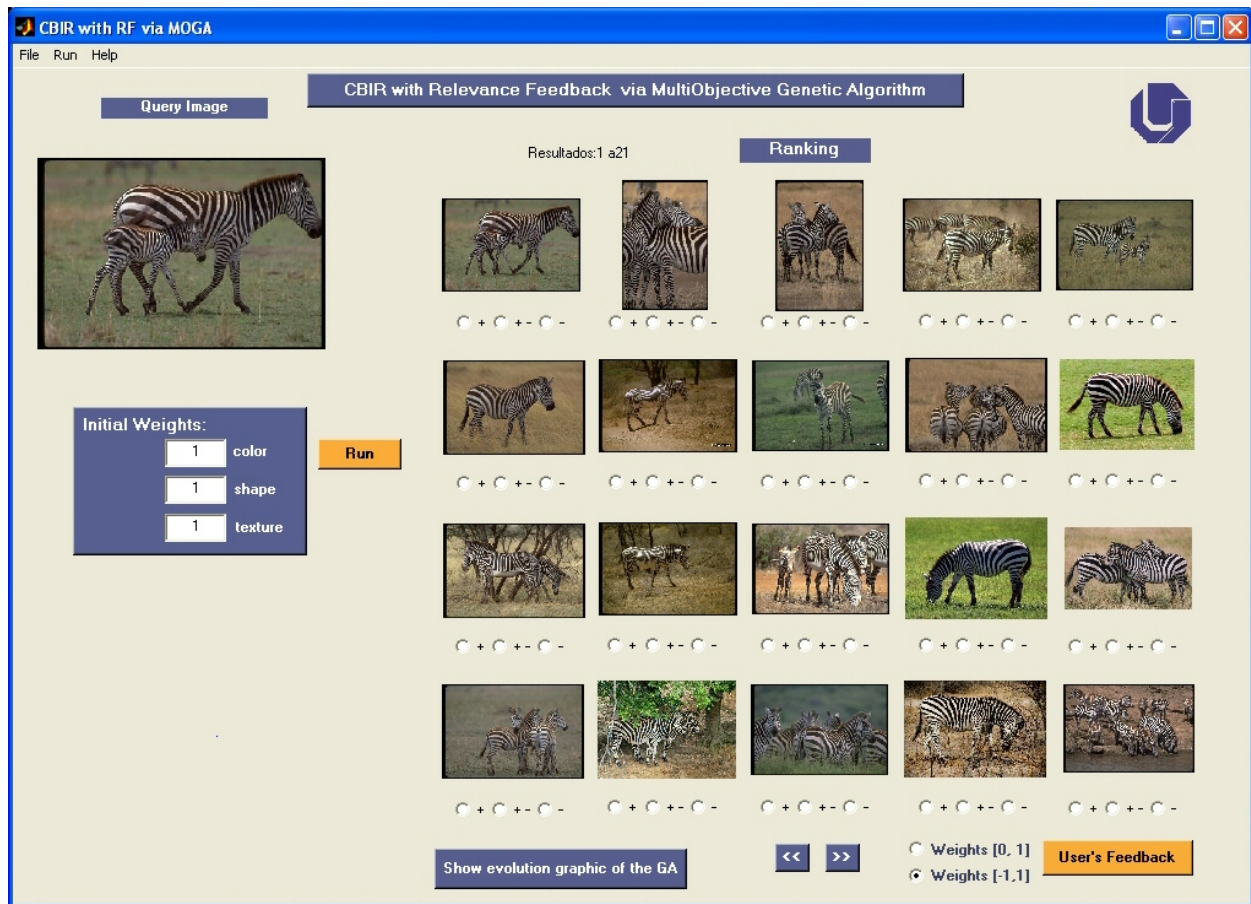


Figura 7.4: Resultado de uma busca por zebras no banco de dados *BD-12750*

7.3 Perspectivas

Acreditamos que:

- A continuação deste trabalho através das sugestões propostas é uma alternativa viável para a obtenção de melhores resultados tanto em eficácia, quanto em tempo de processamento.
- A aplicação do sistema para busca na *Web* será possível com o desenvolvimento de sofisticadas técnicas de indexação ou a implementação do CBIR de forma paralela, principalmente o AG.
- A implementação de vários descritores de imagens e o uso de algoritmos genéticos na ponderação destes, resultará em um sistema altamente preciso e adaptável às necessidades de busca dos usuários.

Conclusão

Neste trabalho foi apresentado um sistema de recuperação de imagens adaptável aos interesses do usuário pelo uso de realimentação de relevantes via algoritmo genético. O algoritmo genético foi modelado para inferir pesos de características e de regiões que melhor traduzem os conceitos de similaridade do usuário expresso na realimentação do sistema, repercutindo na qualidade do *ranking* final.

Devido ao problema de avaliação de *ranking* ser subjetivo nós propusemos duas e experimentamos um total de dez funções de aptidão, as quais são responsáveis por direcionar o AG durante a busca. As funções de aptidão foram classificadas em dois grupos: baseadas em ordem e não baseadas em ordem, conforme seu princípio de construção. Pela análise da curva precisão-revoação em três bancos de dados foi verificado que as funções de aptidão baseadas em ordem levam a resultados significativamente superiores para baixa revocação. Tal resultado é importante pois grande parte dos usuários preferem que os documentos relevantes apareçam no topo do *ranking*. Também verificamos experimentalmente que as funções de aptidão baseadas em ordem impõem ao AG uma maior carga computacional durante um determinado período evolutivo, porém faz com que a evolução do AG na busca de uma solução heurísticamente ótima se dê de forma muito mais rápida. Tais fatos são explicados pela melhor diferenciação da qualidade dos indivíduos feita pelas funções de aptidão baseadas em ordem.

Propusemos um sistema de realimentação de relevantes a partir de exemplos positivos e negativos através do uso de algoritmos genéticos com funções de aptidão multiobjetivos, cujos resultados iniciais mostram-se tão bons quando àqueles obtidos pela técnica apresentada ao longo deste trabalho. Pela substituição ou incremento de descritores e pelo uso da técnica de realimentação via AG para a inferência da importância de cada um destes no contexto de uma busca do usuário, acreditamos que é possível desenvolver sistemas de recuperação que permitem obter resultados de recuperação ainda melhores que os obtidos neste trabalho.

Referências Bibliográficas

- [Arbib 2003] Arbib, M. A. (2003). *The Handbook of Brain Theory and Neural Networks*. MIT Press, Massachusetts, England, 2nd edition.
- [Austin 1990] Austin, S. (1990). *An Introduction to Genetic Algorithms*. AI Expert. **apud** Oliveira, G. M. B.
- [Bäck 1992] Bäck, T. (1992). Self-Adaptation in Genetic Algorithms. In *I European Conference on Artificial Life, Cambridge: MIT Press*, pages 163–271, Paris.
- [Bäck et al. 2000] Bäck, T., Fogel, D. B., and Michalewicz, Z. (2000). *Evolutionary Computation 1: Basic Algorithms and Operators*, volume 1. Institute of Physics Publishing, Philadelphia, USA.
- [Baeza-Yates and Ribeiro-Neto 1999] Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison-Wesley, Essex, UK.
- [Bin and Jia-Xiong 2002] Bin, Y. and Jia-Xiong, P. (2002). Invariance analysis of improved Zernike moments. *Journal os optics A: Pure ans Applied Optics*, 4:606–614.
- [Brandt 1999] Brandt, S. (1999). Use of shape features in content-based image retrieval. Master’s thesis, Helsinki University of Technology, Department of Engineering Physics e Mathematics, Epoo, Finland.
- [Brodatz Database] Brodatz Database. University of Southern California, Signal and Image Processing Institute. USC-SIPI Image Database. Available in <http://sipi.usc.edu/services/database/Databases.html>.
- [CalPhotos] CalPhotos. University of California, Berkeley. Available: <http://calphotos.berkeley.edu/>.
- [Cantú-Paz 2001] Cantú-Paz, E. (2001). *Efficient and Accurate Palallel Genetic Algorithms*. Kluwer Academic Publishers.
- [Cardoso 1998] Cardoso, F. A. C. M. (1998). Algoritmos Genéticos para Intepolação Não-Linear de Imagem e Decodificação de Códigos Lineares. Tese de mestrado, Faculdade de Engenharia Elétrica e de Computação, UNICAMP.

- [Carson et al. 1999] Carson, C., Thomas, M., Belongie, S., Hellerstein, J. M., and Malik, J. (1999). Blobworld: A system for region-based image indexing and retrieval. In *Proceedings of the Third International Conference VISUAL, Lecture Notes in Computer Science, Spring*, Amsterdam, Netherlands.
- [Castelli and Bergman 2002] Castelli, C. and Bergman, L. D. (2002). *Image Databases: Search and Retrieval of Digital Imagery*. Wiley, New York, USA.
- [Chang and Hsu 1992] Chang, S. K. and Hsu, A. (1992). Image information systems: where do we go from here? *IEEE Trans. on Knowledge and Data Engineering*, 5(5):431–442.
- [Chow et al. 2006] Chow, T., Rahman, M., and Wu, S. (2006). Content-based image retrieval by using tre-structured features and multi-layer self organizing map. *Pattern Analysis and Applications*, 9:1–20.
- [Ciocca and Schettini 1999] Ciocca, G. and Schettini, R. (1999). A relevance feedback mechanism for content-based image retrieval. *Information Processing and Management*, 35:605–632.
- [Cordón et al. 2003] Cordón, O., Herrera-Viedma, E., López-Puljalte, C., Luque, M., and Zarco, C. (2003). A review on the application of evolutionary computation to information retrieval. *International Journal of Approximate Reasoning*, 34:241–264.
- [Corel Database] Corel Database. Corel Corporation, Corel Gallery 3.0. Available in James Z. Wang's Research Group: <http://wang.ist.psu.edu/jwang/test1.tar>.
- [Cormen et al. 2002] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2002). *Introduction to Algorithms*. McGraw-Hill, 2nd edition.
- [Crucianu et al. 2004] Crucianu, M., Ferecatu, M., and Boujemaa, N. (2004). Relevance Feedback for Image Retrieval: a Short Survey. *Le Chesnay Cedex*. France, Italy.
- [de Ves et al. 2006] de Ves, E., Domingo, J., Ayala, G., and Zuccarello, P. (2006). A novel Bayesian framework for relevance feedback in image content-based retrieval systems. *Pattern Recognition*, 39:1622–1632.
- [Dorigo and Caro 1999] Dorigo, M. and Caro, G. D. (1999). Ant Colony Optimization: A New Meta-Heuristic. *Proceedings of the Congress on Evolutionary Computation, IEEE Press*, 2:1470–1477.
- [Doulamis and Doulamis 2006] Doulamis, N. and Doulamis, A. (2006). Evaluation of relevance feedback esquemes in content based image retrieval systems. *Signal Processing: Image Communication*, 21:334–357.
- [Fan et al. 2004] Fan, W., Fox, E. A., Pathak, P., and Wu, H. (2004). The Effects of Fitness Functions on Genetic Programming-Based Ranking Discovery For Web Search. *Journal of the American Society for Information Science and Technology*, 55(7):628–636.
- [Fishburn 1998] Fishburn, P. (1998). *Non-Linear Preference and Utility Theory*. Johns Hopkins University Press.

- [Fisher 1930] Fisher, R. A. (1930). *The Genetical Theory of Natural Selection*. Clarendon.
- [Flickner et al. 1995] Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Petkovic, D., Steele, D., and Yanker, P. (1995). Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32.
- [French and Jin 2004] French, J. and Jin, X. (2004). An empirical investigation of the scalability of a multiple viewpoint CBIR system. In *Proceedings of the CIVR, Ireland*, pages 252–260.
- [Fu 1974] Fu, K. S. (1974). *Syntactic Methods in Pattern Recognition*. Academic Press, N.Y.
- [Gevers and Smeulders 1999] Gevers, T. and Smeulders, A. W. M. (1999). Content-based image retrieval by viewpoint-invariant image indexing. *Image and Vision Computing*, 17(7):475–488.
- [Gevers and Smeulders 2000] Gevers, T. and Smeulders, A. W. M. (2000). Pictoseek: Combining color and shape invariant features for image retrieval. *IEEE Trans. on image processing*, 9(1):102–119.
- [Goldberg 1989] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison Wesley.
- [Gonzalez and Woods 2002] Gonzalez, R. C. and Woods, R. E. (2002). *Digital Image Processing*. Prentice-Hall, 2nd edition.
- [Grefenstette 1986] Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1):122–128.
- [H. Tamura 1984] H. Tamura, N. Y. (1984). Image database systems: A survey. *Pattern Recognition*, 17(1):29–43.
- [Haralick 1979] Haralick, R. M. (1979). Statistical and Structural Approaches to Texture. In *Proc. IEEE*, volume 67, pages 786–804.
- [Haupt and Haupt 1998] Haupt, R. L. and Haupt, S. E. (1998). *Practical Genetic Algorithms*. Wiley-Interscience.
- [Holland 1975] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Michigan: MIT Press.
- [Horng and Yeh 2000] Horng, J. and Yeh, C. (2000). Applying genetic algorithms to query optimization in document retrieval. *Information Processing & Management*, 36:737–759.
- [Hu 1962] Hu, M. K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 12:179–187.

- [Huang et al. 1997] Huang, J., Kumar, S. R., Mitra, M., Zhu, W., and Zabih, R. (1997). Image indexing using color correlogram. In *IEEE International Conference on Computer Vision and Evolutionary Computation and Pattern Recognition*, Puerto Rico.
- [Iqbal and Aggarwal 2002] Iqbal, Q. and Aggarwal, J. K. (2002). CIRES: A System for Content-Based Retrieval in Digital Image Libraries. In *Seventh International Conference on Control, Automation, Robotics and Vision*, pages 205–210.
- [Laaksonen 2000] Laaksonen, J. (2000). Analyzing low-level visual features using content-based image retrieval. In *Proceedings of the 7th International Conference on Neural Information Processing*, Teajon, Korea.
- [Laaksonen et al. 2000] Laaksonen, J., Koskela, M., Laakso, S., and Oja, E. (2000). Pic-som: content-based image retrieval with self-organizing maps. *Pattern Recognition Letters*, 21:1199–1207.
- [Langdon 1998] Langdon, W. (1998). *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming*. Kluwer Publishing.
- [Lew et al. 1997] Lew, M., Lempinen, K., and Huijmans, N. (1997). Webcrawling using sketches. In *Proceedings of the 2nd International Conference on Visual Information Systems*, San Diego.
- [Li et al. 2000] Li, J., Wang, J. Z., and Wiederhold, G. (2000). IRM: integrated region matching for image retrieval. In *Proceedings of the 8th ACM Multimedia Conference*, pages 147–156, Los Angeles, USA.
- [Li et al. 2004] Li, Q., Hu, H., and Shi, Z. (2004). Semantic Feature Extraction Using Genetic Programming in Image Retrieval. In *Proceedings of the 17th International Conference on Pattern Recognition, IEEE*.
- [Liu and Picard 1996] Liu, F. and Picard, R. W. (1996). Periodicity, directionality, and randomness: Wold features for image modeling and retrieval. *IEEE Trans. on Pattern Analysis and Machine Learning*, 18(7):184–189.
- [Liu et al. 2007] Liu, Y., Zhang, D., Lu, G., and Ma, W. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40:262–282.
- [Long et al. 2003] Long, F., Zhang, H., and Feng, D. (2003). Fundamentals of Content-Based Image Retrieval. *Multimedia information retrieval and management – technological fundamentals and applications*, Springer-Verlag, pages 1–26.
- [López-Pujalte et al. 2002] López-Pujalte, C., Guerrero-Bote, V., and Moya-Anegón, F. (2002). A test of genetic algorithms in relevance feedback. *Information Processing & Management*, 38:793–805.
- [López-Pujalte et al. 2003a] López-Pujalte, C., Guerrero-Bote, V., and Moya-Anegón, F. (2003a). Genetic algorithms in relevance feedback: a second test and new contributions. *Information Processing & Management*, 39(5):669–807.

- [López-Pujalte et al. 2003b] López-Pujalte, C., Guerrero-Bote, V., and Moya-Anegón, F. (2003b). Order-Based Fitness Functions for Genetic Algorithms Applied to Relevance Feedback. *Journal of the American Society for Information Science*, 54(2):152–160.
- [Ma and Manjunath 1999] Ma, W. Y. and Manjunath, B. S. (1999). Netra: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198.
- [Mallat 1992] Mallat, S. G. (1992). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11:674–693.
- [Michalewicz 1992] Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag.
- [Michalewicz and Fogel 2000] Michalewicz, Z. and Fogel, D. B. (2000). *How to solve it: modern heuristics*. Springer Verlag.
- [Mitchell 1997] Mitchell, M. (1997). *An introduction to genetic algorithms*. Cambridge: MIT Press.
- [Nakazato et al. 2002] Nakazato, M., Manola, L., and Huang, T. S. (2002). ImageGrouper: Search, Annotate and Organize Images by Groups. In *5th International Conference VISUAL*, pages 129–142, Taiwan.
- [Oliveira 1999] Oliveira, G. M. B. (1999). *Dinâmica e Evolução de Autômatos Celulares Unidimensionais*. Tese de doutorado, Instituto Tecnológico de Aeronautica, Curso de Engenharia Eletronica e Computação na Área de Informática.
- [Pass and Zabith 1996] Pass, G. and Zabith, R. (1996). Histogram refinement for content-based image retrieval. In *IEEE Workshop on Applications of Computer Vision*, pages 96–102.
- [Pentland et al. 1996] Pentland, A., Picard, R., and Sclaroff, S. (1996). Photobook: Content-based manipulation of image databases. *International Journal of Computer Vision*, 18(3):233–254.
- [Rui et al. 1997] Rui, Y., T.S.Huang, and Mehrotra, S. (1997). Content-based image retrieval with relevance feedback in MARS. In *Proceedings of International Conference on Image Processing*, volume 2, pages 815–818.
- [Salton 1989] Salton, G. (1989). *Automatic Text Processing*. Addison-Wesley Publishing.
- [Salton and Buckley 1990] Salton, G. and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297.
- [Schettini et al. 2001] Schettini, R., Ciocca, G., and Zuffi, S. (2001). A survey on methods for colour image indexing and retrieval in image databases. In *R. Luo and L. MacDonald (Eds.), Color Imaging Science: Exploiting Digital Media*, Wiley, New York.

- [Sciascio et al. 1999] Sciascio, E. D., Mingolla, G., and Mongiello, M. (1999). Content-Based Image Retrieval over the Web Using Query by Sketch and Relevance Feedback. In *Third International Conference on Visual Information and Information Systems, Lecture Notes In Computer Science, Vol. 1614*, pages 123–130. Springer-Verlag, London, UK.
- [Silva 2002] Silva, A. P. A. (2002). Tutorial: Genetic Algorithms. *Learning and Nonlinear Models - Revista da Sociedade Brasileira de Redes Neurais*, 1(1):45–60.
- [Smith and Fogarty 1996] Smith, J. and Fogarty, T. C. (1996). Self adaptation of mutation rates in a steady state genetic algorithm. In *IEEE International Conference on Evolutionary Computation*, pages 318–323, New York.
- [Stejić et al. 2002] Stejić, Z., Takama, Y., and Hirota, K. (2002). Genetic algorithm-based relevance feedback for image retrieval using local similarity patterns. *Information Processing and Management*, 39:1–23.
- [Stejić et al. 2003a] Stejić, Z., Takama, Y., and Hirota, K. (2003a). Genetic algorithms for a family of image similarity models incorporated in the relevance feedback mechanism. *Applied Soft Computing*, 2:306–327.
- [Stejić et al. 2003b] Stejić, Z., Takama, Y., and Hirota, K. (2003b). Weighted Local Similarity Pattern as image similarity model incorporated in GA-based relevance feedback mechanism. *Intelligent Data Analysis*, 7:443–467.
- [Stejić et al. 2004] Stejić, Z., Takama, Y., and Hirota, K. (2004). Modified hierarchical genetic algorithm for relevance feedback in image retrieval. *Intelligent Data Analysis*, 8:363–384.
- [Stejić et al. 2005] Stejić, Z., Takama, Y., and Hirota, K. (2005). Mathematical aggregation operators in image retrieval: effect on retrieval performance and role in relevance feedback. *Signal Processing*, 85:297–324.
- [Stricker and Orengo 1995] Stricker, M. and Orengo, M. (1995). Similarity of Color Images. In *Proceedings of IS&T and SPIE Storage and Retrieval of Image and Video Databases III*, San Jose, USA.
- [Swain and Ballard 1991] Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7(1):11–32.
- [Tamura et al. 1978] Tamura, H., Mori, S., and Yamawaki, T. (1978). Textural Features Corresponding to Visual Perception. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-8(6):460–473.
- [Tanomaru 1995] Tanomaru, J. (1995). Motivação, Fundamentos e Aplicações de Algoritmos Genéticos. In *Congresso Brasileiro de Redes Neurais*, Curitiba, PR.
- [Torres et al. 2005] Torres, R., Falcão, A., Gonçalves, M., Zhang, B., Fan, W., and Fox, E. (2005). A New Framework to Combine Descriptor for Content-based Image Retrieval. Technical Report IC-05-021, Instituto de Computação, Universidade Estadual de Campinas.

- [Torres and Falcão 2006] Torres, R. S. and Falcão, A. X. (2006). Contour Saliency Descriptors for Effective Image Retrieval and Analysis. *Image and Vision Computing*. To appear.
- [Veltkamp and Tanase 2002] Veltkamp, R. and Tanase, M. (2002). Content-Based Image Retrieval Systems: A Survey. Revised and Extended version of Technical Report UU-CS-2000-34, Department of Computing Science, Utrecht University.
- [Vistex Database] Vistex Database. Vision Texture Database. Massachusetts Institute of Technology. Media Laboratory. Available in <ftp://whitechapel.media.mit.edu/pub/Vis-Tex/>.
- [Voorhees and Poggio 1988] Voorhees, H. and Poggio, T. (1988). Computing texture boundaries from images. *Nature*, 333:364–367.
- [Wang et al. 2001] Wang, J. Z., Li, J., and Wiederhold, G. (2001). SIMPLIcity: Semantics-sensitive Integrated Matching for Picture Libraries. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(9):947–963.
- [Zhang 2002] Zhang, D. (2002). Image Retrieval Based on Shape. Phd thesis, Faculty of Information Technology, Monash University.
- [Zhang and Lu 2004] Zhang, D. and Lu, G. (2004). Review of shape representation and description techniques. *Pattern Recognition*, 37:1–19.
- [Ziviani 2004] Ziviani, N. (2004). *Projeto de Algoritmos com Implementações em Pascal e C*. Thomson, 2nd edition.
- [Zobel and Moffat 1998] Zobel, J. and Moffat, A. (1998). Exploring the similarity space. *SIGIR Forum*, 32(1):18–34.

Parte III

Apêndice

Funções de Aptidão Adaptadas do Contexto de Recuperação de Informação Textual

Neste apêndice as funções de aptidão descritas no Capítulo 3 e que antes eram utilizadas na recuperação de informações textuais serão adaptadas conforme o nosso contexto.

Nas medidas de aptidão, $Apt(q, C)$ denota a aptidão do cromossomo C para a consulta q , $|I|$ representa o número de imagens do banco de dados, D representa o conjunto de documentos conhecidos como relevantes para a consulta q , $|D|$ representa o número de elementos de D , $r(d)$ retorna a relevância do documento d , sendo 1 se o documento é relevante, isto é, se $d \in D$ e 0 caso contrário e k_i , $i = 1 \dots 9$ são constantes. Adicionalmente, consideramos que o *ranking* retornado (documentos recuperados) consiste de todas as imagens da coleção ordenadas conforme sua similaridade à consulta.

A seguir apresentamos as funções de aptidão adaptadas, sendo estas identificadas com o mesmo nome utilizado na descrição de suas versões originais no Capítulo 3.

$F1_{Fan \text{ et al.}}$:

$$Apt(q, C) = k_1 \sum_{i=1}^{|I|} r(d_i) \ln^{-1}(i + k_2) \quad (A.1)$$

$F2_{Fan \text{ et al.}}$:

$$Apt(q, C) = k_3 \sum_{i=1}^{|I|} r(d_i) \log_{10}^{-1}(|I|/i) \quad (A.2)$$

onde $|I|$ é o número de imagens da coleção de referência. Utilizamos $\log_{10}^{-1}(|I|/i)$ ao invés de $\log_{10}^{-1}(1000/i)$ devido ao fato de considerarmos que todas as imagens da coleção de referência são “ranqueadas” e recuperadas.

$F3_{Fan \text{ et al.}}$:

$$Apt(q, C) = k_4^{-1} \sum_{i=1}^{|I|} r(d_i) (e^{-k_5 \ln(i) + k_6} - k_7) \quad (A.3)$$

$F4_{Fan \text{ et al.}}$:

$$Apt(q, C) = k_8 \sum_{i=1}^{|I|} r(d_i) k_9^i \quad (A.4)$$

$F1_{Lopez-Pujalte \text{ et al.}}$:

$$Apt(q, C) = \frac{1}{|I|} \sum_{i=1}^{|I|} \left(r(d_i) \sum_{j=i}^{|I|} \frac{1}{j} \right) \quad (A.5)$$

$F2_{Lopez-Pujalte \text{ et al.}}$: nesta medida, o AG irá incluir um adicional parâmetro A , que determina o fator a ser utilizado na computação da utilidade dos documentos. Aqui, a medida de aptidão é dada por:

$$Apt(q, C) = \sum_{i=1}^{|I|} \left(r(d_i) \frac{1}{A} \left(\frac{(A-1)}{A} \right)^{(i-1)} \right) \quad (A.6)$$

Eliminamos a fator $\frac{\sum_{i=1}^{|D|} r(d_i)}{|D|}$ devido ao fato deste não fazer sentido a partir do instante que consideramos que todas as imagens da coleção de referências são “ranqueadas” e recuperadas.

$F_{P_{AVG}}$:

$$Apt(q, C) = \sum_{i=1}^{|I|} \frac{\left(r(d_i) \left(\frac{\sum_{j=1}^i r(d_j)}{i} \right) \right)}{|D|} \quad (A.7)$$

Outros Resultados

Neste apêndice são apresentados um total de 14 resultados obtidos pela abordagem proposta e usando a função de aptidão $F2_{Proposta}$. Todos os resultados aqui apresentados foram realizados usando *BD-12750* como coleção de referência. Em todos os resultados a primeira imagem do *ranking* é a consulta, e o *ranking* inicial é obtido com todos os pesos iguais a 1.

A Figura B.1-(a) mostra um resultado de busca pela categoria flores vermelhas. Para obter este resultado o sistema foi realimentado duas vezes com todas as imagens relevantes entre as vinte primeiras do *ranking*. Na Figura B.1-(b) temos o resultado de uma busca por mini-flores. Neste o sistema foi realimentado uma vez com todas as imagens relevantes entre as vinte primeiras no *ranking* inicial. A Figura B.1-(c) mostra um resultado de busca por texturas de casca de árvores. Neste o sistema é realimentado duas vezes com todas as imagens relevantes entre as vinte primeiras do *ranking*. Na Figura B.1-(d) temos um resultado de busca por árvores, obtido através de duas realimentações com todas as imagens relevantes entre as vinte primeiras do *ranking*. A Figura B.1-(e) corresponde a uma busca por árvores floridas (primavera) obtida com uma realimentação com as imagens entre as vinte primeiras no *ranking* inicial. Na Figura B.1-(f) temos uma busca por elefantes obtida à partir de três realimentações com todas as imagens relevantes entre as vinte primeiras do *ranking*.

As Figuras B.2-(a) e (b) correspondem à buscas por cobras e torres, respectivamente e foram obtidas com duas realimentações com todas as imagens relevantes entre as vinte primeiras dos *rankings*. As Figuras B.2-(c), (d), (e) e (f), correspondem à buscas por pássaros, textura (folhas), peixes e motocicletas, respectivamente, sendo estes resultados obtidos no *ranking* inicial.

As Figuras B.3 e B.4 mostram a interface da nossa aplicação e, os resultados de busca obtidos para montanhas polares e pôr-do-sol, respectivamente. No resultado para montanhas (Figura B.3) o sistema é realimentado duas vezes com todas as imagens relevantes entre as vinte primeiras do *ranking*; já no resultado para pôr-do-sol (Figura B.4) o sistema é realimentado uma única vez com todas as imagens relevantes entre as vinte primeiras do *ranking* inicial.

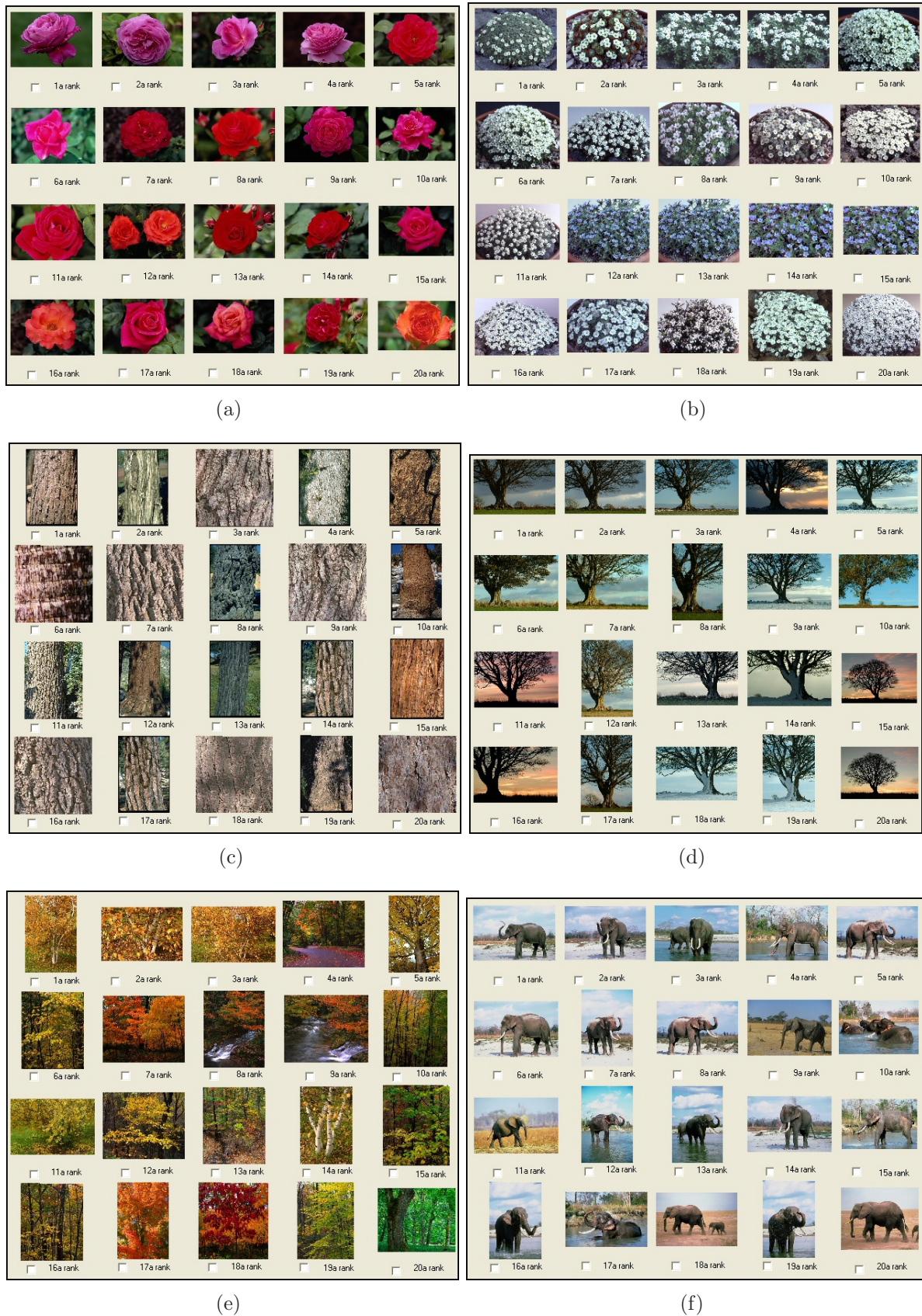


Figura B.1: Resultados de busca por rosas vermelhas, mini-flores, casca de árvores, árvores, primavera e elefantes, respectivamente, no banco de dados *BD-12750*.



Figura B.2: Resultados de busca por cobras, torres, pássaros, textura, peixes e motos, respectivamente, no banco de dados *BD-12750*.

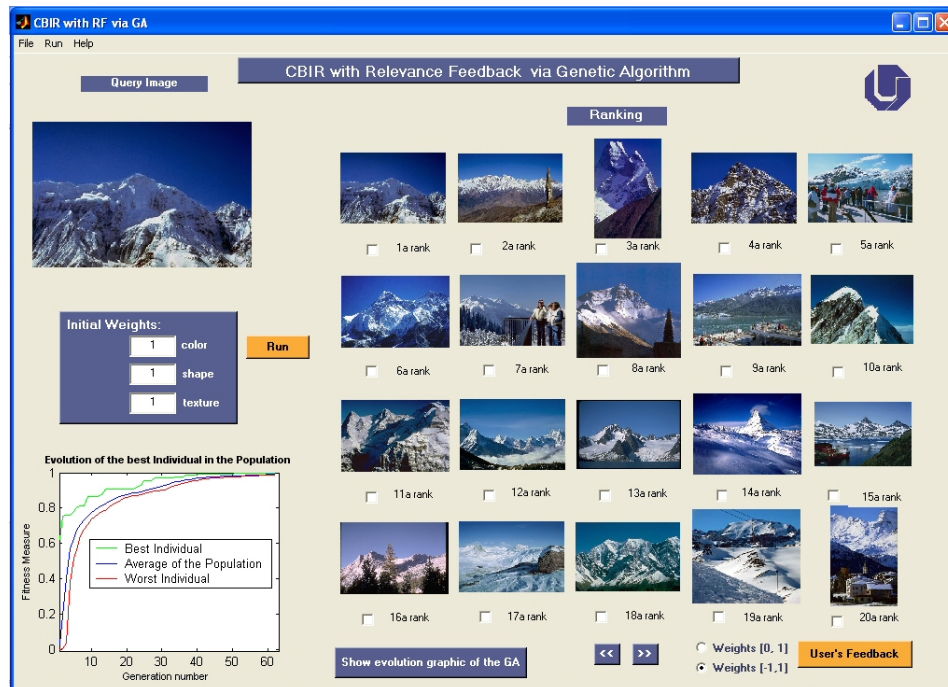


Figura B.3: Resultado de busca por imagens de montanhas polares no banco de dados *Corel-1000*.

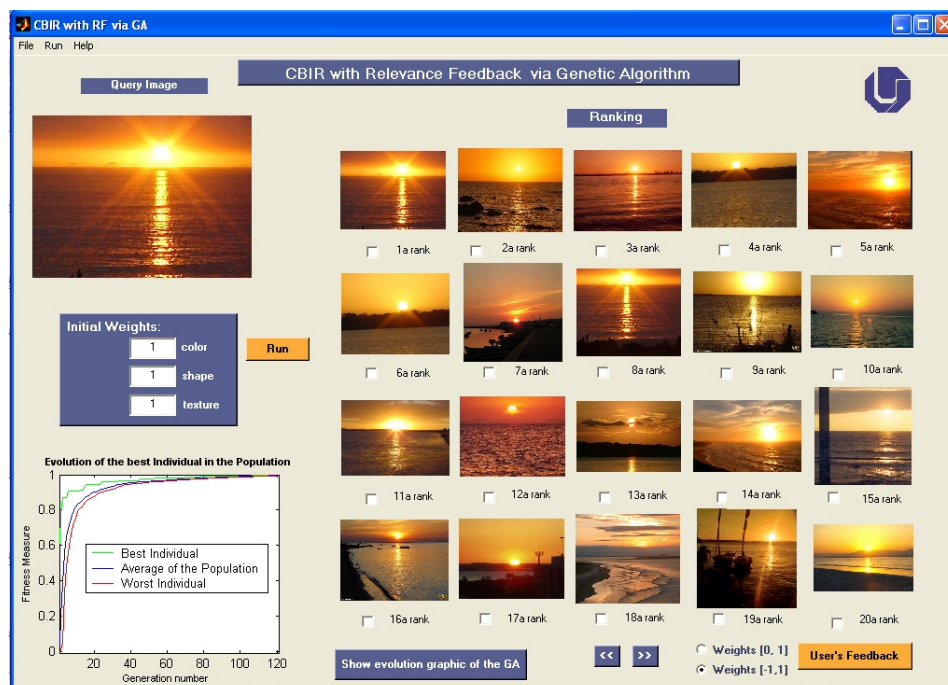


Figura B.4: Resultado de busca por imagens de pôr-do-sol, no banco de dados *BD-12750*.

© Sérgio Francisco da Silva & Celia Aparecida Zorzo Barcelos

Rua Feliciano de Moraes 256 - Bairro Aparecida
38.400-684 Uberlândia (MG) - Brasil
`sergio.f.silva@gmail.com`

Dissertação de Mestrado em Ciência da Computação
documento escrito em L^AT_EX 2_ε
UFU - Universidade Federal de Uberlândia - FACOM