**Universidade Estadual de Campinas**
**Faculdade de Engenharia Elétrica e de Computação**
**Departamento de Engenharia de Computação e Automação Industrial**

# Busca na Web e Agrupamento de Textos Usando Computação Inspirada na Biologia

por

André Luiz Vizine Pereira

Orientador: Prof. Dr. Ricardo Ribeiro Gudwin
DCA-FEEC-UNICAMP
Co-orientador: Prof. Dr. Leandro Nunes de Castro
Mestrado em Informática UNISANTOS

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da UNICAMP como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: Engenharia de Computação

Campinas, 18 de dezembro de 2007

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA E ARQUITETURA - BAE - UNICAMP

Pereira, André Luiz Vizine
P414b Busca na Web e agrupamento de textos usando computação inspirada na biologia / André Luiz Vizine Pereira. --Campinas, SP: [s.n.], 2007.

Orientadores: Ricardo Ribeiro Gudwin, Leandro Nunes de Castro
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Tecnologia da informação. 2. Algoritmos genéticos. 3. Ferramentas de busca na Web. 4. algoritmos de formiga. I. Gudwin, Ricardo Ribeiro. II. Castro, Leandro Nunes de. III. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. IV. Título.

Título em Inglês: Search in the Web and text clustering using computing inspired by biology
Palavras-chave em Inglês: Filtering information, Virtual communities, Swarm intelligence, Text clustering, Genetic algorithm, Search in the Web, Computing inspired by biology
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Fernando José Von Zuben, Ivan Luiz Marques Ricarte e Eduardo Raul Hruschka
Data da defesa: 18/12/2007
Programa de Pós-Graduação: Engenharia Elétrica

## COMISSÃO JULGADORA - TESE DE MESTRADO

**Candidato:** André Luiz Vizine Pereira

**Data da Defesa:** 18 de dezembro de 2007

**Título da Tese:** "Busca na Web e Agrupamento de Textos Usando Computação Inspirada na Biologia"

Prof. Dr. Ricardo Ribeiro Gudwin (Presidente): Ricardo R. Gudwin

Dr. Eduardo Raul Hruschka: Eduardo R. Hruschka

Prof. Dr. Fernando José Von Zuben: Fernando José Von Zuben

Prof. Dr. Ivan Luiz Marques Ricarte:

## Resumo

A Internet tornou-se um dos principais meios de comunicação da atualidade, reduzindo custos, disponibilizando recursos e informação para pessoas das mais diversas áreas e interesses. Esta dissertação desenvolve e aplica duas abordagens de computação inspirada na biologia aos problemas de otimização do processo de busca e recuperação de informação na *web* e agrupamento de textos. Os algoritmos investigados e modificados são o algoritmo genético e o algoritmo de agrupamento por colônia de formigas. O objetivo final do trabalho é desenvolver parte do conjunto de ferramentas que será usado para compor o núcleo de uma comunidade virtual acadêmica adaptativa. Os resultados obtidos mostraram que o algoritmo genético é uma ferramenta adequada para otimizar a busca de informação na *web*, mas o algoritmo de agrupamento por colônia de formigas ainda apresenta limitações quanto a sua aplicabilidade para agrupamento de textos.



## Abstract

The Internet became one of the main sources of information and means of communication, reducing costs and providing resources and information to the people all over the world. This dissertation develops and applies two biologically-inspired computing approaches, namely a genetic algorithm and the ant-clustering algorithm, to the problems of optimizing the information search and retrieval over the web, and to perform text clustering. The final goal of this project is to design and develop some of the tools to be used to construct an adaptive academic virtual community. The results obtained showed that the genetic algorithm can be feasibly applied to the optimizing information search and retrieval, whilest the ant-clustering algorithm needs further investigation in order to be efficiently applied to text clustering.

## Agradecimentos

Aos meus orientadores Ricardo Ribeiro Gudwin e Leandro Nunes de Castro pela paciência, pelas conversas, por acreditarem em minhas propostas, e por muito me ajudarem com conhecimentos e orientações que foram essenciais ao desenvolvimento do trabalho.

Aos professores Ciro Cirne Trindade e Santo Scuderi pelo estímulo e apoio incondicional.

Aos meus pais que ensinaram-me o valor da vida.

A minha esposa Andrea e aos meus filhos Amanda e Mário, pela paciência, força e compreensão nos momentos de minha ausência.

*"Confia em Deus como se só dele e não de ti dependesse o sucesso e, contudo, para realizar aja como se só tu e não Deus tivesse de fazer tudo"*

Santo Inácio de Loyola

# Índice

## Lista de Figuras

## Lista de Tabelas

# 1. Introdução

A cada dia que passa não só a quantidade de páginas *web* aumenta de forma exponencial, como também a quantidade de pessoas que necessitam se conectar e utilizar a Internet. A Internet tornou-se um dos principais meios de comunicação da atualidade, reduzindo custos, disponibilizando recursos e informação para pessoas das mais diversas áreas e interesses. Ao longo dos anos, uma das áreas que vem se beneficiando da Internet é a comunidade científica (Lawrence et al. 1999a,b; Lawrence, 2001).

Há algumas décadas atrás, reunir um grupo de pesquisadores de regiões distantes para uma conferência era uma tarefa árdua. Agrupar tais pessoas para participar de um mesmo projeto tinha um custo muito elevado. Como exemplo prático podemos citar a Inteligência Artificial, que nasceu da reunião de um pequeno grupo de pesquisadores no Campus do Dartmouth College em Hanover no ano de 1956 e atualmente reúne milhares de pesquisadores pelo mundo.

Há quem diga que a Internet é a maior biblioteca do mundo. Seu maior problema, entretanto, é que seu conteúdo (os 'livros') estão espalhados pela rede e sem indexação, cabendo ao leitor interessado fazer a busca por todas as partes vasculhando ('volume por volume') para encontrar o que procura (Barrie, 1996). Além disso, ao encontrar um documento de seu interesse o leitor deve se certificar se tal informação é de qualidade, visto que muitas vezes é difícil encontrar um responsável direto por tal obra. Em parte, as máquinas de busca vieram auxiliar o usuário a lidar com este excesso e distributividade da informação, embora seja praticamente impossível fazer uma busca exaustiva na *web*. Basicamente, as máquinas de busca procuram indexadores de assuntos na *web*, o que significa que a informação precisa estar indexada para que seja encontrada por uma máquina de busca.

A utilização de máquinas de busca, no entanto, soluciona apenas parte do problema, pois na maioria das vezes as máquinas de busca atuam respondendo a uma *consulta* (*query*) colocada pelo usuário. Essa consulta (composta por palavras-chave e operadores

booleanos) é confrontada com a base de dados, sendo retornadas as URLs que mais similaridade possuem com a consulta em questão. Mesmo assim, ainda cabe ao usuário analisar os resultados obtidos o que, dependendo da formulação da consulta, pode consumir bastante tempo.

Diante deste cenário, a pesquisa para o desenvolvimento de ferramentas que auxiliem a comunidade acadêmica no processo de busca e acesso a informação relevante é de suma importância. Iniciativas como o *site* CiteSeer[1], por exemplo, promoveram a criação de fontes mais seguras de informação para acadêmicos interessados na área de Computação. O CiteSeer mantém uma coleção de documentos científicos e recursos para facilitar a busca, seja ela por área, autor ou título do documento, além de apresentar dados relevantes à citação de cada documento, permitindo que o usuário possa avaliar a quantidade de pessoas que leram o referido documento.

O Google, empresa que criou uma das principais máquinas de busca utilizadas na Internet, recentemente lançou o Google Acadêmico[2], uma máquina de busca cujo foco é pesquisar a literatura acadêmica de forma simples e abrangente. A pesquisa pode ser feita em várias disciplinas e fontes em um só lugar, indexando artigos revisados por especialistas (*peer-rewiewed*), teses, livros, resumos e artigos de editoras acadêmicas, organizações profissionais, bibliotecas de pré-publicações, universidades e outras entidades acadêmicas. O Google Acadêmico classifica os resultados da pesquisa segundo a *relevância*, sendo que as referências mais úteis são exibidas no começo da página. A tecnologia de classificação considera o texto integral de cada artigo, o autor, o local onde o artigo foi publicado e a freqüência com que foi citado em outras publicações acadêmicas (Google, 2006).

Segundo Rheingold (2004), *comunidades virtuais* são agregações sociais que surgem na *web* quando um determinado grupo de pessoas realiza discussões públicas, formando uma teia de relacionamentos pessoais no ambiente *web*. As comunidades virtuais

[1] http://citeseer.ist.psu.edu/
[2] http://scholar.google.com.br/

possibilitam a integração de diferentes sociedades, a convergência de diversidades e a produção de interesses comuns, aproximando pessoas.

Uma *comunidade* (*virtual*) *acadêmica adaptativa* pode ser caracterizada por um ambiente computacional que procura e recupera documentos da Internet formando uma coleção de artigos científicos (arquivos no formato PDF) devidamente classificados e armazenados em estruturas de diretórios num sistema de arquivos de um servidor (Vizine *et al*., 2005b). Para uma pessoa ter acesso a essas fontes de informação, ela deve estar cadastrada em uma ou mais áreas de seu interesse, tornando-se um habitante ativo da comunidade, visto que as variáveis que mantêm uma comunidade virtual são as relações obtidas entre seus habitantes: a interação, colaboração e/ou cooperação. Através dessas relações, cada habitante da comunidade fica responsável pela qualidade da informação que a comunidade mantém. Quando algum habitante encontra um artigo interessante que ainda não foi indexado na comunidade, ele pode referenciar esse conteúdo. Assim, esse arquivo será colocado numa espécie de mural eletrônico e todos os outros habitantes dessa comunidade serão notificados da existência de um novo documento disponível para a leitura. O documento só será classificado na comunidade quando um certo número de membros der um voto favorável para o artigo em questão. Com o passar do tempo, essas estruturas de diretório começam a aumentar de maneira significativa, tanto em quantidade de arquivos como também na qualidade do conteúdo, pois os artigos são selecionados com base em opiniões que os habitantes da comunidade associam a eles. Os membros da comunidade podem trocar informações a respeito do que cada um achou sobre determinado artigo, atribuindo uma pontuação que poderá servir como referência a um outro membro que ainda não tenha lido o documento.

## 1.1 Comunidades Virtuais e a Pesquisa Científica

Assim como acontece em grandes conferências de âmbito nacional ou internacional que oferecem oportunidade de encontro com pessoas de lugares geograficamente distantes, as comunidades virtuais também permitem encontros virtuais entre alunos e/ou pesquisadores distantes e, conseqüentemente, facilitar o envolvimento em atividades de ensino e pesquisa através do uso de recursos tecnológicos. Ao considerar as comunidades

virtuais no contexto da pesquisa é possível identificar, dentre outras atividades, as seguintes:

- Troca de informações sobre um determinado assunto;
- Discussão geradora de críticas às propostas da pesquisa;
- Orientações de pesquisadores experientes decorrentes desse processo interativo.

Sendo possíveis essas atividades em um ambiente virtual como a Comunidade Virtual Acadêmica Adaptativa, os indivíduos podem interagir de forma semelhante à de um seminário, onde há divulgação e discussão sobre pesquisas realizadas pelas análises e críticas dos conteúdos apresentados (Matuzawa & Fonseca, 2001). Como resultado espera-se reduzir a duplicação de esforços e aumentar a eficiência do processo de busca, organização e compartilhamento de informações obtidas via web por comunidades virtuais acadêmicas.

### 1.1.1. Problemas a Serem Investigados no Desenvolvimento de Comunidades Virtuais Adaptativas

Comunidades Virtuais só existem e funcionam à medida que são frutos da participação ativa de seus habitantes, entendidos esses como indivíduos, grupos formais ou informais, empresas, organizações, etc. Basicamente, o desenvolvimento de uma comunidade virtual consite na criação de um ambiente virtual, dotado de uma série de características, como:

- **Informação personalizada:** são os usuários que decidem qual informação vão armazenar, mostrar e intercambiar. Portanto, cada usuário decide por onde começar a ver a comunidade, para que e com quem;
- **Acesso universal e simultâneo:** basta ter acesso a um computador que esteja conectado a Internet para ter acesso a comunidade ou visualizar seu conteúdo. Vários usuários podem estar conectados simultaneamente;

- **Independência de tempo, espaço e alta disponiblidade:** espaço aberto permanentemente para atividades independentemente da localização física do usuário; e
- **Ambiente eminentemente participativo:** o conteúdo de uma comunidade virtual é fruto das publicações, contribuições e interações seus habitantes.

Tal qual a internet, esses espaços virtuais propiciam um crescimento constante da informação e do conhecimento. A evolução, portanto, é um fator incorporado à própria estrutura da comunidade. Crescimento não significa só acrescentar mais informação, mas também tudo o que isto implica: sistemas de busca, classificação, síntese, participação e interação, organização e visualização da informação.

Segundo Palazzo (2002) podem ser identificados três grandes problemas na implementação de comunidades virtuais:

1. Elas não empregam técnica alguma para adaptar seu conteúdo, recursos, serviços e a navegação que disponibilizam a seus usuários, os quais podem possuir variados interesses, objetivos e expectativas com relação a comunidade. Essa simplificação acarreta diversas implicações que podem desmotivar, desorientar ou mesmo impedir a participação de um determinado usuário.
2. O segundo problema envolve a gestão do conhecimento produzido e compartilhado. Tal conhecimento está associado ao propósito/finalidade da comunidade e é representado por um domínio não estruturado, que pode ser visto como uma ontologia, envolvendo conceitos, relacionamentos, descrições e sua evolução. O conhecimento coletivamente produzido em comunidades virtuais pode ser de difícil recuperação e muitas vezes se apresenta com pouca ou nenhuma organização. Além disso, mesmo que o conhecimento esteja organizado, em muitos casos isto é feito de forma estática, não refletindo o caráter dinâmico e evolutivo da comunidade. Por fim, mesmo atendendo a todos os requisitos citados, a representação adotada pode não ser reutilizável, isto é, o conhecimento gerado precisa adotar uma forma de

representação que permita sua reutilização em outras comunidades com objetivos semelhantes ou relacionados.

3. O terceiro problema refere-se às condições de colaboração oferecidas pelo ambiente da comunidade, que idealmente deveria incentivar a comunicação síncrona e assíncrona, o intercâmbio de informações e o desenvolvimento de uma "cultura" de colaboração na comunidade. Tais objetivos envolvem a construção e utilização de ferramentas capazes de catalisar a ação colaborativa.

### 1.1.2. Uma Proposta para uma Comunidade Virtual Acadêmica Adaptativa

Esta seção apresenta a estrutura necessária para suportar o desenvolvimento de uma comunidade virtual acadêmica adaptativa. As funcionalidades discutidas nesta seção dão ênfase aos problemas levantados por Palazzo (2002). As demais funcionalidades presentes na maioria das comunidades virtuais (tais como: autenticação de usuário, bate-papo, fóruns, e-mail, mural, etc.) não serão objeto de estudo tendo em vista uma gama significativa de componentes de software que podem ser customizados ao ambiente sem requerer grandes esforços. A Figura 1.1 destaca os principais módulos a serem desenvolvidos para a implementação de uma comunidade virtual acadêmica adaptativa.

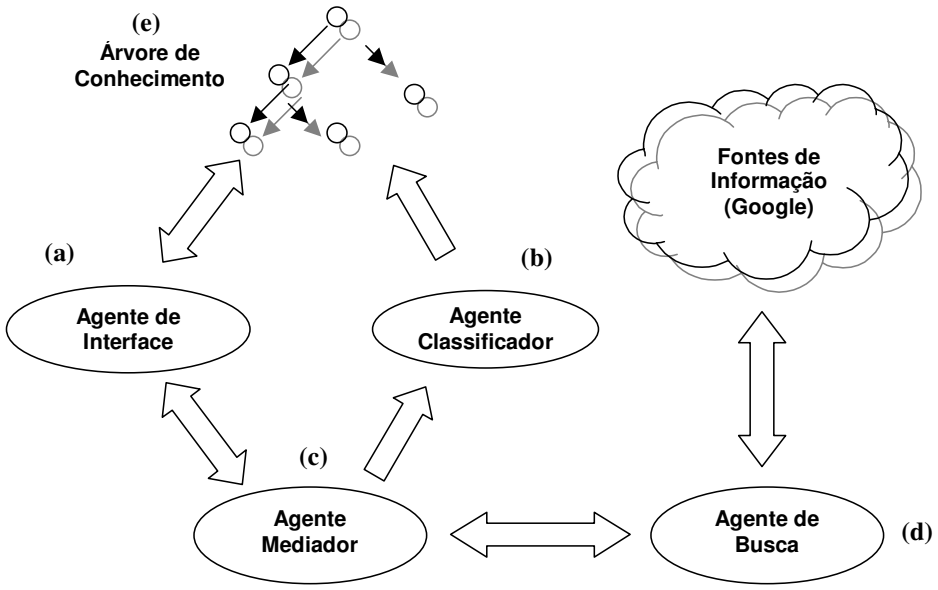



Figura 1.1: Principais módulos a serem desenvolvidos para a implementação de uma Comunidade Virtual Acadêmica Adaptativa.

**(a) Agente de Interface**

O *agente de interface* é responsável por representar os interesses do usuário na comunidade, ou seja, através dele o usuário manifesta seus interesses e preferências. Ele permite que o sistema se adapte (aprenda) aos interesses do usuário de duas formas:

**1) Através da observação do usuário:** o agente monitora quais documentos foram acessados pelo usuário e extrai as informações relevantes para futuras sugestões.

**2) Através de exemplos:** o usuário pode fornecer ao agente de interface conjuntos de documentos constituindo bons e maus exemplos para que o agente recupere a informação desejada e evite informações desnecessárias.

**(b) Agente classificador**

O *agente classificador* é responsável pela classificação automática de documentos relevantes na árvore de conhecimento. Agindo de forma autônoma, de tempos em tempos, esse agente procura reestruturar a ramificação da árvore de conhecimento no intuito de melhor classificação dos documentos nela inseridos. A classificação será feita levando em

consideração um conjunto de palavras-chaves extraídas automaticamente de documentos recuperados pelo agente de busca. A extração automática das palavras-chaves está detalhada na seção 3.4.1 desta dissertação.

Uma proposta de agente classificador foi implementada tomando por base o algoritmo de agrupamento inspirado no comportamento de formigas, cujo o detalhamento da implementação bem como os resultados obtidos podem ser encontrados no Capítulo 4.

**(c) Agente Mediador**

O *agente mediador* atua como ponte entre o agente de interface e o agente de busca e é também responsável pela apresentação e organização de conteúdo. Quando o agente mediador recebe os resultados do agente de busca ele procura identificar quais documentos estão estruturados visando melhorar a apresentação do mesmo ao usuário. Por exemplo: se um documento estiver na forma de artigo, o agente mediador apresentará ao usuário o resumo do mesmo junto com sua referência bibliográfica.

O agente mediador tem contato direto com o agente classificador. Quando um documento é referenciado de forma positiva pelo agente de interface (o usuário atribui ao documento uma nota o referenciando com útil) o agente mediador notifica o agente classificador do ocorrido para que o mesmo faça a classificação do documento na árvore de conhecimento.

**(d) Agente de busca**

O agente de busca interage com o motor de busca do Google com o objetivo de otimizar o resultado da busca fornecido pelo Google. Cabe, portanto, ao agente de busca apresentar os melhores resultados encontrados. O Capítulo 3 detalha a implementação desse agente de busca bem como os resultados obtidos com a sua implementação.

**(e) Árvore de conhecimento**

A árvore de conhecimento permite o armazenamento, e posterior compartilhamento via agente de interface, das informações obtidas pela comunidade ao longo de sua existência. É importante ressaltar que com o passar do tempo, a quantidade de documentos armazenados numa dada área de conhecimento tende a crescer de maneira significativa. Dessa forma, seria interessante que a árvore de conhecimento passasse por um processo automático de reclassificação de conteúdo a fim de apresentar aos usuários uma estrutura hierárquica construída de forma dinâmica. Um algoritmo de agrupamento poderia ser utilizado para encontrar novos subgrupos dentro de cada nó da árvore, ou seja, uma determinada área de interesse com o passar do tempo, poderia ser segmentada em subáreas de acordo com os documentos armazenados.

A comunidade virtual acadêmica apresentada envolve a integração de programas e componentes de software livre com métodos e técnicas da computação natural visando o desenvolvimento de um ambiente virtual colaborativo, dinâmico, de qualidade onde grande parte dos componentes necessários para a sua implementação podem ser obtidos em licença GPL (software livre) e sua integração, e distribuição podem ocorrer a um custo relativamente pequeno.

## 1.2. Objetivos do Trabalho

Há séculos o homem observa a natureza estudando seus fenômenos físicos, químicos e biológicos, procurando entender e explicar os mecanismos nela envolvidos. Devido a esses estudos, nossa civilização encontra-se num estágio científico-tecnológico que nos permite realizar tarefas desafiadoras, tais como viagens espaciais, romper a barreira do som, descobrir medicamentos para doenças antes consideradas incuráveis e até mesmo *clonar* seres vivos.

Pesquisadores de áreas como engenharia e computação estudam fenômenos biológicos para o desenvolvimento de sistemas computacionais com grande potencial de resolver problemas. Essa linha de pesquisa, conhecida por *computação inspirada na biologia*, compreende técnicas já bastante difundidas como *redes neurais artificiais* e *computação evolutiva*, e outras técnicas mais recentes como *sistemas imunológicos artificiais* e *inteligência de enxame* (de Castro & Von Zuben, 2004; de Castro, 2006; de Castro, 2007).

Esta dissertação investiga, desenvolve e aplica dois paradigmas de computação inspirada na biologia: algoritmos evolutivos e inteligência de enxame. A computação evolutiva, em particular os *algoritmos genéticos*, são empregados como ferramenta para otimizar o processo de busca de informação na *web*, avaliando sua capacidade de adaptação em um ambiente real e extremamente dinâmico, como a comunidade virtual acadêmica adaptativa discutida anteriormente. Tal ferramenta poderia servir como um agente de busca para fomentar as áreas de interesse da comunidade virtual acadêmica, trazendo constantemente novos conteúdos para seus membros.

O outro paradigma estudado é a inteligência de enxame, em particular um algoritmo de agrupamento inspirado no comportamento de insetos sociais (no caso desta dissertação, formigas), sendo proposta uma variação desse algoritmo com desempenho avaliado quando submetido a diversos problemas de agrupamento de dados numéricos e a dois conjuntos de bases textuais. Por possuir uma interface gráfica que permite acompanhar o processo de agrupamento e não necessitar da definição do número de *clusters* a priori, esse algoritmo poderia servir como um sistema de recomendação de leitura da comunidade, permitindo aos seus membros acessarem artigos que ainda não foram lidos e que se encontram numa região de vizinhança próxima aos documentos já lidos.

Portanto, esta dissertação investiga e modifica técnicas de computação inspirada na biologia para que elas possam futuramente compor o núcleo básico de algoritmos a serem empregados na comunidade acadêmica adaptativa.

## 1.3. Estrutura da Dissertação

O Capítulo 2 apresenta sucintamente de que forma a biologia vem sendo utilizada como fonte de inspiração para o desenvolvimento de algoritmos voltados a resolução de problemas complexos. O enfoque é dado aos algoritmos bio-inspirados estudados nesta dissertação.

No Capítulo 3 é apresentado um sistema evolutivo para a otimização de busca na *web*. É avaliado o comportamento de um algoritmo genético num espaço de busca instável e dinâmico como a Internet.

O Capítulo 4 apresenta o algoritmo de agrupamento baseado em formigas (*Ant Clustering Algorithm*, ACA), bem como algumas contribuições para sua melhoria de desempenho e critério de convergência.

No Capítulo 5 são feitas considerações gerais sobre este trabalho e apresentadas sugestões para trabalhos futuros.

# 2. Computação Inspirada na Biologia

## 2.1. Introdução

Há tempos o homem observa a natureza estudando seus fenômenos físicos, químicos e biológicos, procurando entender e explicar os mecanismos nela envolvidos. Através dessas observações alguns modelos e sistemas foram abstraídos, servindo de base para a construção de diversos produtos, como aeronaves, submarinos, sistemas de navegação autônoma, alimentos modificados, drogas para o tratamento de doenças e outros.

Os computadores contribuíram de maneira significativa na forma com que o homem interage com a natureza. Atualmente, a natureza tem sido fonte de inspiração, ou em alguns casos usada como metáfora, para o desenvolvimento de novas técnicas para a resolução de problemas complexos em vários domínios, tais como engenharia, filosofia, biologia e saúde.

A *computação natural* é a versão computacional do processo de extração de idéias da natureza para o desenvolvimento de sistemas computacionais com o objetivo de resolver problemas ou simular fenômenos naturais. Esta denominação também é empregada para descrever o uso de matéria-prima natural no desenvolvimento de novas tecnologias para computadores. O termo computação natural é utilizado, portanto, para representar três tipos de abordagens (de Castro & Von Zuben, 2004; de Castro, 2006): *i*) a computação inspirada na natureza; *ii*) a simulação e emulação computacional de fenômenos naturais; e *iii*) a computação com matéria-prima natural.

A simulação e emulação da natureza por meios computacionais é basicamente um processo de síntese de padrões, formas, comportamentos e organismos que (não necessariamente) assemelham-se à "vida como ela é". Neste segmento são estudados fenômenos como a geometria fractal da natureza e a vida artificial.

A computação com materiais naturais corresponde ao uso de matéria-prima natural, por exemplo, moléculas, para o desenvolvimento de novas tecnologias computacionais, como novos processadores. Portanto, ela constitui um novo paradigma computacional que visa substituir ou suplementar os tradicionais computadores baseados em silício. A computação quântica e a computação de DNA são exemplos de paradigmas estudados nessa área.

A computação inspirada na natureza, em particular na biologia, faz uso da natureza como fonte de inspiração para o desenvolvimento de novas técnicas para a resolução de problemas. A idéia principal desta linha de pesquisa é o desenvolvimento de ferramentas computacionais (algoritmos) inspirados por processos naturais ou seus respectivos modelos teóricos que são utilizados na resolução de problemas complexos.

O objetivo desse capítulo é apresentar, de forma resumida, as principais técnicas que compõem a computação inspirada na biologia que serão empregadas na dissertação. Este capítulo está organizado da seguinte maneira. A teoria da evolução proposta por Darwin é a fonte de inspiração utilizada na pesquisa e desenvolvimento da computação evolutiva apresentada na Seção 2.2, que enfatiza os algoritmos genéticos devido a suas aplicações no contexto desta dissertação. Um algoritmo genético foi empregado na implementação de um sistema evolutivo de busca de informação na *web*, que será apresentado de maneira detalhada no Capítulo 3. Os algoritmos baseados no comportamento de insetos sociais são discutidos na Seção 2.3. O algoritmo conhecido por ACA (*Ant Clustering Algorithm*) também é discutido de forma detalhada, pois uma nova versão desse algoritmo é proposta no Capítulo 4 desta dissertação com aplicações tanto em análise de dados numéricos quanto em análise de dados textuais. As considerações finais do capítulo são apresentadas na Seção 2.4.

## 2.2. Computação Evolutiva

*Computação evolutiva* é um termo utilizado para nomear técnicas computacionais que são baseadas na evolução das espécies (Eiben & Smith, 2003). A evolução é, na maioria das vezes, determinada pela seleção natural de diferentes indivíduos que competem

por recursos vitais para sobrevivência. De maneira geral, os indivíduos mais aptos sobrevivem, garantindo a continuidade da espécie através da troca de material genético durante a reprodução. A carga genética dos pais é recombinada (misturada) e herdada pelos seus descendentes.

A *teoria da evolução* de Darwin (1859) propõe uma explicação parcial da diversidade biológica hoje encontrada e dos mecanismos nela envolvidos. A seleção natural ocupa um papel essencial na evolução das espécies. Se analisarmos um dado ambiente que esteja restrito a hospedar um número limitado de indivíduos e sendo o instinto básico desses indivíduos a reprodução, então a *seleção natural* é o principal mecanismo para evitar que a população cresça de forma descontrolada. A seleção natural favorece aqueles indivíduos que competem pelos recursos disponíveis de forma mais eficiente, ou seja, aqueles que conseguem se adaptar melhor às condições impostas pelo ambiente.

Um outro fator importante identificado por Darwin refere-se às pequenas variações presentes entre indivíduos de uma mesma população. Estas particularidades podem ser físicas ou comportamentais e diretamente afetam a interação do indivíduo com o ambiente, determinando assim seu grau de adaptabilidade ao meio, conhecido como *fitness*. Cada indivíduo representa uma combinação única dessas particularidades (variações *fenotípicas*) que é avaliada pelo ambiente, determinando assim seu *fitness*. Em caso de uma avaliação positiva essas características serão propagadas para os filhos deste indivíduo. Em caso de uma avaliação negativa, o indivíduo provavelmente morrerá ou não será capaz de se reproduzir devido a não conseguir se adaptar ao ambiente, o que impede que suas características (genéticas) sejam propagadas.

De acordo com Darwin, essas variações fenotípicas ocorrem durante a reprodução dos indivíduos, de geração em geração. Através dessas variações, ocorrem novas combinações que são 'avaliadas' pelo ambiente, fazendo com que os mais aptos sobrevivam e se reproduzam. A ação conjunta destas etapas (reprodução, variação e seleção natural) caracteriza os chamados *processos evolutivos* .

A evolução, portanto, pode ser vista como um processo de busca capaz de localizar soluções para problemas encontrados em um ambiente. Do estudo do processo evolutivo surgiram modelos simplificados implementados na forma de algoritmos. Aos algoritmos desenvolvidos com base no processo biológico da evolução dá-se o nome de *algoritmos evolutivos* (de Castro, 2006). Atualmente, é crescente o número de algoritmos evolutivos empregados com sucesso em problemas de diferentes domínios, incluindo otimização, planejamento, controle, processamento de sinais, bioinformática e sistemas sociais.

A idéia de fazer do processo evolutivo um paradigma de resolução de problemas é antiga. Durante a década de 1960 três diferentes implementações surgiram em regiões distintas do mundo. Nos Estados Unidos, L.J. Fogel, A.J.Owens e M.J.Walsh introduziram a Programação Evolutiva (Fogel *et al*., 1965; 1966), enquanto J. Holland batizou seu método de Algoritmo Genético (De Jong, 1975; Holland, 1973; 1975). Na Europa, mais especificamente na Alemanha, I. Rechenberg e H. P. Schwefel apresentaram uma técnica conhecida como Estratégia Evolutiva (Rechenberg, 1973; Schwefel, 1995). No início dos anos 90 surge uma quarta abordagem conhecida como Programação Genética, formalizada por J. Koza (Koza, 1992; 1994). O termo Computação Evolutiva foi proposto em 1990 para englobar as quatro abordagens acima, mais os sistemas classificadores (Holland, 1975) que, a despeito de algumas diferenças, propõem algoritmos inspirados na biologia evolutiva (Eiben & Smith, 2003).

De maneira geral os algoritmos evolutivos partem de uma população de indivíduos, ou soluções candidatas, escolhidos inicialmente segundo algum critério, por exemplo, inicialização aleatória. Novas soluções são criadas aplicando operadores de recombinação (cruzamento ou, do inglês, *crossover*) e/ou mutação. Após a recombinação e mutação, cada indivíduo (solução) é avaliado e um critério de seleção é então aplicado determinando quais indivíduos serão mantidos na próxima geração. A execução do algoritmo é feita de maneira iterativa até que uma condição de parada tenha sido satisfeita. Na maioria das vezes a condição de parada estabelecida é um determinado número de iterações, denominadas de *gerações* pela analogia com o processo biológico. A Figura 2.1 ilustra os principais passos de um algoritmo evolutivo. Os pontos principais de um algoritmo evolutivo são: *i*) a etapa

de recombinação, que é responsável pelo compartilhamento de características entre os indivíduos; *ii*) a etapa de mutação, que permite inserir diversidade e, conseqüentemente, explorar mais amplamente o espaço de possíveis soluções; e *iii*) a etapa de seleção dos indivíduos para a próxima geração, que atua como uma espécie de 'controle de qualidade' dos indivíduos gerados. A atuação conjunta destas etapas proporciona um aumento qualitativo médio sucessivo nos indivíduos das populações a cada iteração, direcionando-os cada vez para mais perto de pontos de ótimo da superfície de adaptação.

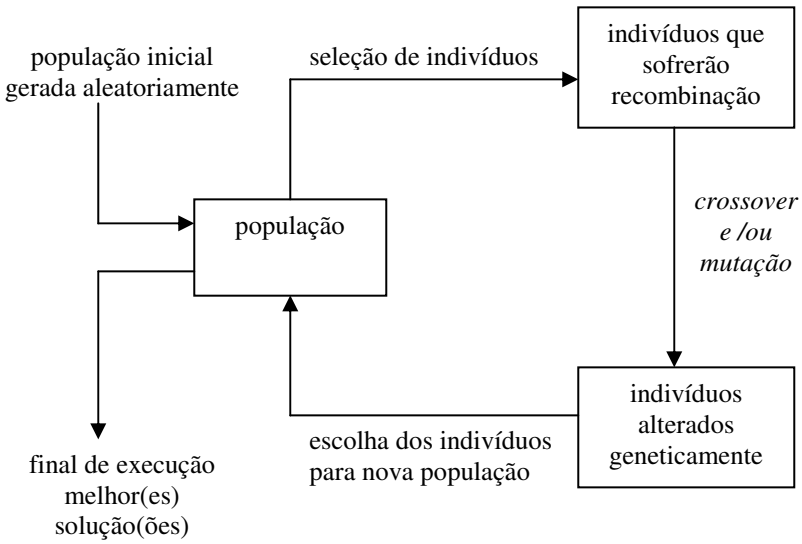



Figura 2.1 : Esquema genérico de um algoritmo evolutivo (adaptado de Eiben & Smith, 2003).

Como os algoritmos genéticos constituem uma das principais abordagens empregadas nesta dissertação, eles serão explicados em mais detalhes na seqüência.

### 2.2.1. Algoritmos Genéticos

*Algoritmos genéticos* (AGs) são algoritmos de busca e otimização baseados nos mecanismos de seleção natural e da genética (Holland, 1975). O comportamento do algoritmo corresponde a um procedimento de busca em um espaço de soluções candidatas para a resolução de um dado problema. Cada solução candidata é representada por um *indivíduo*, sendo o conjunto de soluções correspondente a uma *população* de indivíduos.

Cada indivíduo da população é representado por um único *cromossomo* constituído por um conjunto de *genes* que, computacionalmente, são implementados na forma de vetores ou cadeias de atributos, nas quais cada atributo corresponde a um gene.

O processo de busca é normalmente multi-direcional e multi-dimensional com etapas aleatórias. A busca reforça indivíduos bem adaptados ao meio, ou seja, indivíduos que correspondem a locais onde a função objetivo a ser otimizada assume valores relativamente altos (no caso de maximização). A cada iteração do algoritmo (geração) os indivíduos mais adaptados se reproduzem, ao passo que os menos adaptados tendem a serem eliminados. A seleção dos indivíduos que irão constituir a próxima geração é feita tomando por base o seu *fitness*. A cada geração, cada indivíduo é submetido a uma função de avaliação ou de adaptabilidade (*fitness*) que simula o papel da pressão seletiva exercida pelo ambiente sobre o indivíduo. Dessa forma, os indivíduos com maior adaptação (*fitness*) relativa têm maiores chances de sobreviver e se reproduzir.

Após um número de gerações espera-se que a população inicial tenha sofrido alterações genéticas substanciais como conseqüência da ação conjunta da população de indivíduos sujeita a recombinação, mutação e seleção, gerando, portanto, novos indivíduos mais adaptados ao ambiente. A maioria dos métodos de seleção é projetada para escolher preferencialmente indivíduos com maiores valores de *fitness*, embora não exclusivamente, a fim de manter a diversidade da população.

A estrutura básica de um algoritmo genético clássico, como proposto por Holland (1975), é constituída dos seguintes passos:

2. Inicialmente uma população de indivíduos aleatórios é criada;

3. A população de indivíduos é avaliada obedecendo algum critério determinado por uma função de avaliação, conhecida por *fitness*, cuja finalidade é avaliar a qualidade relativa do indivíduo;

4. Por intermédio de um mecanismo probabilístico de seleção são escolhidos prioritariamente os indivíduos de melhor valor (*fitness*). Estes servirão de base para a criação de um novo conjunto de possíveis soluções candidatas;

5. A nova geração é obtida aplicando-se sobre os indivíduos selecionados operações que misturem suas características, usando para isso os operadores genéticos de cruzamento e mutação;

6. Os Passos de 1 a 4 acima são repetidos até que uma solução aceitável seja encontrada, ou até que um número predeterminado de passos seja atingido.

**Representação dos Indivíduos**

Como dito anteriormente, cada indivíduo corresponde a uma possível solução para o problema. Dessa forma, a representação apropriada dos indivíduos torna-se um dos principais pré-requisitos para o bom desempenho do algoritmo. No algoritmo genético clássico proposto por Holland (1975), cada indivíduo é representado por um vetor de atributos binários de tamanho fixo. Embora a representação binária seja amplamente utilizada, existem determinadas situações onde o problema a ser resolvido exige uma representação mais especializada. Algoritmos genéticos aplicados ao problema do caixeiro viajante, por exemplo, normalmente são implementados empregando-se uma permutação de números inteiros (Braun, 1990). Indivíduos representados por uma cadeia de números de ponto flutuante são comumente utilizados em problemas de otimização contínua (Michalewicz, 1998). Cadeias de caracteres e até mesmo cadeias de sentenças (*strings*) também são utilizadas na representação dos indivíduos, como será visto no Capítulo 3. Diversas outras formas são possíveis; normalmente a forma mais apropriada está ligada ao problema em questão.

**Seleção de Indivíduos**

O algoritmo genético clássico utiliza um esquema de seleção de indivíduos para a próxima geração conhecido como *seleção pela roleta* (do inglês, *roulette wheel selection*). Nesse método, cada indivíduo da população possui uma probabilidade de seleção em uma roleta proporcionalmente ao seu *fitness.* Dessa forma, indivíduos com alto fitness obtêm uma porção maior da roleta, enquanto indivíduos com baixo fitness ocupam uma porção relativamente menor da roleta. Assim, quanto maior o *fitness* de um indivíduo, maior a probabilidade dele ser selecionado para a próxima geração. A Tabela 2.1 apresenta um exemplo de uma população constituída por quatro indivíduos representados por uma cadeia binária de tamanho igual a seis, na qual é apresentado o *fitness* de cada indivíduo, bem como o seu percentual em relação ao *fitness* global, informação que é utilizada na seleção pela roleta mostrada graficamente na Figura 2.2.

Tabela 2.1: Exemplo de uma população em um algoritmo genético.

| **Indivíduo** | **Representação** | ***fitness*** | **% do total** |
|---|---|---|---|
| 1 | 000001 | 50 | 10 |
| 2 | 110101 | 200 | 40 |
| 3 | 100101 | 150 | 30 |
| 4 | 010001 | 100 | 20 |
| Total | | 500 | 100 |

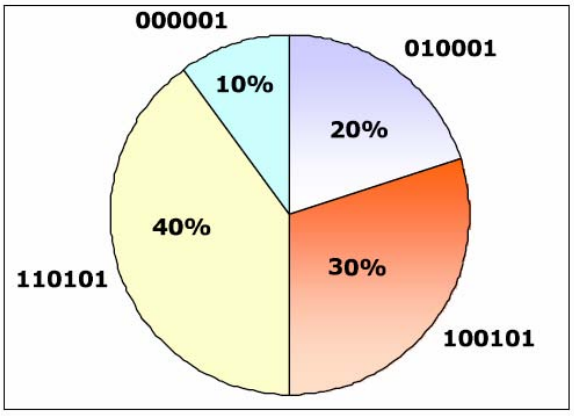



Figura 2.2: Representação pictórica do método de seleção pela roleta.

Além de apresentar uma demanda de tempo de processamento considerável, visto que o método da roleta exige duas passagens por todos os indivíduos da população, por ser uma seleção probabilística, o melhor indivíduo da população pode ser perdido, ou seja, pode não ser selecionado para fazer parte da nova geração.

**Operadores Genéticos**

O objetivo principal dos operadores genéticos é garantir a herança e mistura das características que levam a altos valores de *fitness*, além de manter a diversidade na população. Os operadores mais freqüentemente utilizados nos algoritmos genéticos são o cruzamento (*crossover*) e a mutação.

Através do *crossover* são gerados novos indivíduos misturando características de dois indivíduos "pais". A idéia intuitiva que embasa o operador de *crossover* é a troca de informação entre diferentes soluções candidatas. O resultado desta operação é um indivíduo que potencialmente combine características importantes dos seus pais. O operador de *crossover* mais comumente empregado é o *crossover de um ponto*: são selecionados dois indivíduos (pais) e a partir de seus cromossomos são gerados dois novos indivíduos (filhos), sendo que cada filho contém uma parte do código genético de cada pai (Holland, 1975).

O operador de mutação é utilizado para promover uma melhor exploração do espaço de busca e, assim, diminuir a incidência de convergência prematura para ótimos locais. Ele pode ser aplicado a cada descendente individualmente e consiste na mudança aleatória, dada uma probabilidade definida *a priori*, de cada um dos genes que constituem o indivíduo.

## 2.3. Inteligência de Enxame

No início dos anos 1990, o fascínio exercido por tais insetos nos biólogos começou a atrair a atenção de pesquisadores de outras áreas como computação e engenharia.

Segundo Bonabeau *et al*. (1999) existem boas razões que justificam o aumento de interesse pelo estudo dessas espécies, tais como: *i*) a cada dia o mundo em que vivemos torna-se mais complexo, ficando difícil para que uma pessoa possa entendê-lo; *ii*) o excesso de informação (e não a falta dela) está ameaçando nossas vidas; e *iii*) os sistemas computacionais tornaram-se tão complexos que em alguns casos são de difícil controle. Aliado a essas razões, o estudo dos insetos sociais pode oferecer novos caminhos para o desenvolvimento de projetos de sistemas inteligentes nos quais características como autonomia, comportamento emergente e funcionamento distribuído, substituam as necessidades atuais como: controle, pré-programação e centralização. O termo *inteligência de enxame* (do inglês, *swarm intelligence*) foi primeiramente usado no trabalho de G. Beni e J. Wang (1989), o qual descrevia o comportamento de um grupo de robôs que interagiam num determinado ambiente respeitando um conjunto de regras locais. Apesar da variedade de definições existentes para esse termo na literatura, particularmente a definição dada por T. White e B. Pagurek (1998) é bem interessante: "Inteligência de Enxame é a propriedade de sistemas de agentes não inteligentes com capacidades individuais limitadas exibirem um comportamento coletivo inteligente." (White & Pagurek, 1998).

### 2.3.1. Exemplos de Comportamentos Sociais de Insetos

Insetos, como formigas, abelhas e cupins, se beneficiam da vida em grupo em vários aspectos. Por exemplo, a vida em grupo facilita a busca por alimentos, reduz a probabilidade de ataques por predadores, facilita a caça e permite a divisão de trabalho. Um fato extremamente interessante é que apesar de que cada inseto pertencente a uma colônia tenha suas próprias obrigações e tarefas definidas, não existe uma hierarquia na colônia que supervisione o andamento dos trabalhos e, mesmo assim, a colônia como um todo possui um comportamento extremamente organizado. Esta seção descreve, sucintamente, três exemplos de comportamentos sociais de insetos.

Como primeiro exemplo, tome o caso das formigas da espécie *Atta*, conhecidas como cortadoras de folhas. Elas são predominantes nas florestas da América Central, América do Sul e nos estados do sul dos Estados Unidos e são altamente organizadas. Elas

mantêm uma relação mutualística com os fungos. As formigas ganham comida dos fungos e os fungos ganham um local para viverem livres, protegidos pelas formigas de predadores e parasitas. As tarefas são divididas de tal forma que, enquanto algumas formigas percorrem extensas trilhas em busca de folhas (algumas delas com cerca de 200 metros), outras são responsáveis pela alimentação e crescimento dos fungos (Ant Colony, 2005).

Estratégias para tirar vantagem competitiva não acontecem somente no mundo dos negócios. Uma espécie de abelha sem ferrão chamada *Trigona Hyalinata* faz uso de algumas estratégias na busca de alimento para evitar concorrentes. Essas abelhas, conhecidas popularmente no Brasil por *guaxupé*, ou apenas *xupé*, montam uma trilha química de cheiro característico (*feromônio*) apenas parcial entre a fonte de alimento e o caminho até a colméia e, além disso, zelam para que só suas conhecidas cheguem até lá. Um estudo feito por pesquisadores brasileiros e americanos (Contrera *et al*., 2001) na análise das trilhas e do comportamento das abelhas não deixou dúvidas: a trilha de feromônio é polarizada, isto é, as abelhas deixam o odor mais concentrado em uma das pontas do caminho, justamente onde fica a comida. Inicialmente esparsos, os pontos de cheiro vão aparecendo em maior quantidade conforme as operárias se aproximam do líquido açucarado. Mas ela não se estende por todo o caminho, ela vai, no máximo, a uma distância de 27 metros do recipiente, sinal de que as guaxupés estão 'escondendo o jogo' para evitar competidores. No entanto, como isso poderia evitar que as próprias colegas de colônia chegassem à fonte de alimento? A solução é ir até o ninho e recrutar abelhas "de confiança". Segundo F. Contrera (Contrera *et al.,* 2001), as abelhas chegavam à fonte de alimento em grandes grupos de uma vez. Essa estratégia adotada pelas abelhas guaxupés é diferente da de outras abelhas, que marcam todo o caminho entre a colméia e a fonte de comida.

Cupins são insetos de hábito social que, como as abelhas e formigas, formam uma colônia. A colônia de cupins é formada por indivíduos que exercem funções especializadas, tais como busca de alimento, reprodução, postura de ovos e defesa do ninho, dentre outras. Para manter todos os indivíduos desta sociedade, o ninho desempenha um papel importante, oferecendo condições microclimáticas (temperatura e umidade) adequadas e seguras a

todos os indivíduos desta comunidade, protegendo-a contra inimigos naturais (predadores e parasitas). Em relação à longevidade dos cupins, o rei e a rainha podem viver até 30 anos. A formação de uma nova colônia pode ocorrer por motivo de isolamento. Isso acontece quando uma colônia identifica uma nova fonte de alimento e uma sub-colônia é formada para explorar esta nova fonte alimentar. Se o caminho entre a colônia principal e a colônia secundária for quebrado a população isolada pode então dar origem às formas reprodutoras, geradas através dos operários funcionais e ninfas (Sentricon, 2005).

### 2.3.2. Algoritmos de Inteligência de Enxame

Ao contrário das redes neurais artificiais, não há muitos algoritmos de inteligência de enxame na literatura. Pode-se dizer que as três principais abordagens são: *i*) os algoritmos de otimização por colônias de formigas (Dorigo & Stützle, 2004); *ii*) o método de otimização por enxame de partículas (Kennedy *et al*., 2000); (de Castro & Von Zuben, 2004); e *iii*) o algoritmo de agrupamento baseado em formigas (Bonabeau *et al*., 1999).

O algoritmo de otimização por colônias de formigas (do inglês, *ant colony optimization* – ACO) é inspirado na observação de que as formigas em busca de alimentos liberam uma substância química conhecida por *feromônio* no caminho que vai da fonte de alimentos ao ninho. Quando alguma formiga encontra alimento, ela volta para o ninho despejando feromônio pelo caminho de volta. O feromônio serve como orientação para que outras formigas encontrem a fonte de alimento. A cada nova formiga que utiliza o caminho, o mesmo procedimento é adotado, reforçando assim a trilha de feromônio, que serve como uma espécie de recrutamento para que um número cada vez maior de formigas seja atraído para a fonte de alimento. A Figura 2.3 ilustra um experimento que mostra o recrutamento, através de feromônio, de formigas do ninho para a fonte de alimentos. Como pode ser observado na Figura 6, entre o ninho e a fonte de comida existe um obstáculo que faz uma bifurcação no meio do caminho. Inicialmente, as formigas escolhem de forma aleatória o segmento A ou B. Com o passar do tempo, o segmento mais utilizado será aquele que possuir uma presença maior de feromônio, fazendo com que grande parte das formigas adotem esse caminho.

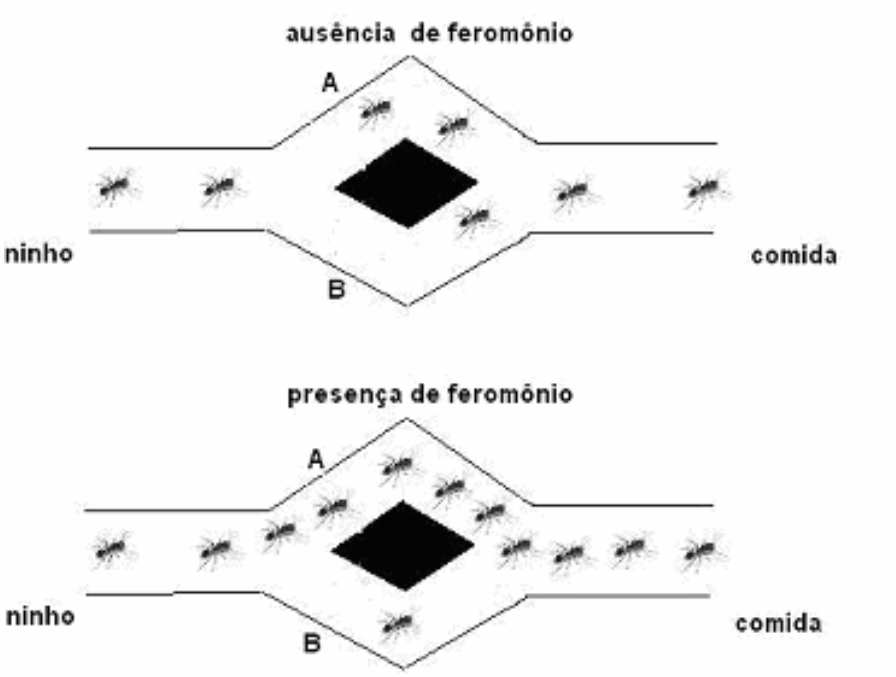



Figura 2.3: Adaptação do experimento feito por Deneubourg *et al*. (1987).

**Algoritmo de Agrupamento por Formigas**

A terceira principal técnica que compõe a inteligência de enxame é denominada de *algoritmo de agrupamento por formigas* (do inglês, *ant clustering algorithm* – ACA). Ela será descrita em detalhes nesta dissertação, pois constitui a principal técnica empregada no contexto de agrupamento automático de informação (Capítulo 4). Além disso, é importante salientar que esta técnica não tem recebido a mesma atenção das outras duas, que já possuem livros escritos exclusivamente para suas descrições (Kennedy *et al*., 2000; Dorigo & Stützle, 2004).

Algumas espécies de formigas manifestam um comportamento interessante no tocante à organização e limpeza do ninho. Chrétien (1996) comprovou esse comportamento estudando a espécie *Lasius niger*, na qual as operárias são responsáveis por agrupar formigas mortas e partes de corpos de formigas em regiões ao redor do ninho, mantendo as entradas de acesso desbloqueadas e o ninho limpo. Comportamento semelhante foi

observado na espécie *Pheidole pallidula* nos estudos feitos por Deneubourg *et al.* (1991). A Figura 2.4 mostra o resultado de experimentos obtidos com o estudo da espécie *Messor sancta* (Bonabeau, 1999).

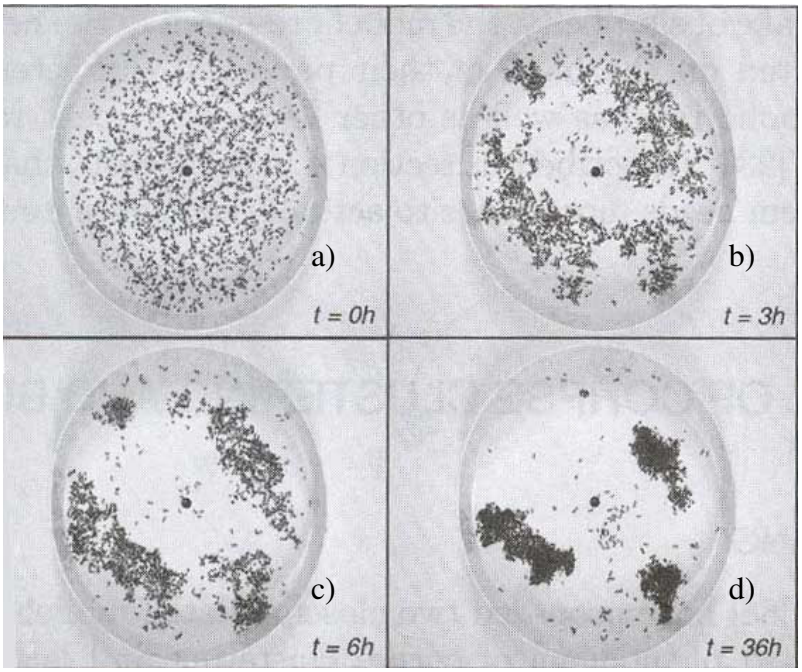



Figura 2.4: Agrupamentos de corpos de formigas da espécie *Messor sancta*. Situação do ninho em diferentes instantes de tempo. (Fonte: Bonabeau, 1999).

Inicialmente, são depositadas, numa arena circular cujo diâmetro possui 25cm, 1.500 formigas mortas de maneira aleatória, onde encontra-se um grupo de operárias da espécie *Messor sancta* (Fig. 2.4a). O trabalho das operárias pode ser analisado graficamente após 3 horas (Fig. 2.4b), 6 horas (Fig. 2.4c) e 36 horas (Fig. 2.4d). O mecanismo básico por trás desse comportamento é a atração exercida entre as formigas mortas, mediada pela ação das operárias. Pequenos grupos de itens (formigas mortas e partes de corpos de formigas) aumentam de tamanho, pois a operária é atraída a colocar mais corpos naquela região. Esse é um tipo de realimentação positiva que tende a formar grupos cada vez maiores. Entretanto, o comportamento individual que leva a esse mecanismo adaptativo de realimentação positiva ainda é pouco compreendido.

Um outro fenômeno relacionado com a organização de ninhos é observado nas formigas da espécie *Leptothorax unifasciatus*. As operárias dessa espécie classificam sua

prole de acordo com o seu tamanho. Os ovos e larvas menores são agrupados no centro do ninho. Já as larvas maiores e as pupas são agrupadas ao redor do grupo formado pelos ovos e larvas menores (Franks & Sendova-Franks, 1992). Uma possível explicação para tal comportamento é o fato de que, tendo os itens (larvas) de necessidades similares (mesmas características) reunidos num mesmo local, pode-se aumentar a eficiência no trato desses indivíduos. Por exemplo, durante a fase de pupa ou pré-pupa, os indivíduos não necessitam de alimentação, sendo necessário somente acondicionar esses indivíduos em um local adequado. Franks & Sendova-Franks mostraram que à medida em que os indivíduos se transformam eles vão sendo reagrupados de acordo com seu tamanho (Franks & Sendova-Franks, 1992).

Inspirados na forma pela qual algumas espécies de formigas mantêm a organização e limpeza do ninho, Deneubourg *et al.* (1991) propuseram dois modelos fortemente relacionados. Ambos os modelos se apóiam nos mesmos princípios, mas o primeiro (agrupamento de formigas mortas) é um caso especial do segundo (classificação das larvas). Enquanto o modelo de agrupamento reproduz com maior fidelidade o comportamento das formigas observado nos experimentos realizados em laboratório, o segundo possibilita sua adequação a um número maior de aplicações (Bonabeau *et al.*, 1999).

No trabalho de Deneubourg *et al*. (1991), um grupo de robôs representando uma colônia de formigas move-se aleatoriamente dentro de uma arena bidimensional na qual as "formigas-robô" são capazes de movimentar objetos básicos no intuito de agrupá-los. A idéia geral é que os objetos que estejam isolados na arena sejam pegos e colocados em outras localizações na vizinhança das quais existam mais itens do mesmo tipo. Assumindo que exista somente um tipo de objeto na arena, a probabilidade $p_p$ de uma formiga escolhida aleatoriamente, que não esteja carregando item algum, pegar um item da arena é dada pela seguinte equação:

$$p_p = \left( \frac{k_1}{k_1 + f} \right)^2 , \qquad (2.1)$$

onde $f$ é a *fração de itens percebidos* pela formiga em sua vizinhança e $k_1$ é um valor constante. Quando $f << k_1$, $p_p$ é próximo de 1, isto é, a probabilidade de pegar um item é alta quando não houver muitos itens na vizinhança da formiga. Por outro lado, $p_p$ é próximo de zero quando $f >> k_1$, isto é, torna-se improvável a remoção de itens de agrupamentos com alta densidade (grande número de objetos).

A probabilidade $p_d$ de uma formiga escolhida aleatoriamente, que esteja carregando um item, depositar o item na arena é dada pela seguinte equação:

$$p_d = \left( \frac{f}{k_2 + f} \right)^2 , \qquad (2.2)$$

onde $f$ é a *fração de itens percebidos* pela formiga em sua vizinhança e $k_2$ é um outro valor constante. Para $f << k_2$, $p_d$ é próximo de 0, ou seja, a probabilidade de uma formiga depositar um item na arena é baixa quando não existirem itens na sua vizinhança, enquanto que para $f >> k_2$, $p_d$ é próximo de 1, ou seja, a probabilidade de um item ser depositado num grupo de objetos é alta. Isto indica uma tendência no depósito de itens em regiões da arena onde existam grandes concentrações de objetos.

No intuito de empregar o modelo proposto por Deneubourg *et al.* (1991) para a implementação de um algoritmo genérico de agrupamento de dados inspirado em formigas, ACA (do inglês, *ant clustering algorithm*), duas questões importantes ainda precisam ser resolvidas (de Castro, 2006): *i*) Como projetar o espaço de atributos que constituem os dados de tal forma que seja possível a visualização dos grupos (*clusters*)?; e *ii*) De que forma a função $f$ (a fração de itens percebidos) deve ser definida?

Num dos primeiros algoritmos do tipo ACA, proposto por Lumer & Faieta (1994), as formigas movem-se sobre uma região discreta de duas dimensões caracterizada por uma grade (*grid*) $B$ de $s \times s$ células. Cada formiga pode viajar de um lado a outro do *grid*, assumindo dessa forma que o *grid* é toroidal. As formigas percebem uma região ao redor de

sua localização demarcada por uma matriz quadrada de vizinhança $\text{Neigh}_{(s\times s)}$ de $s\times s$ células, como ilustrado na Figura 2.5.

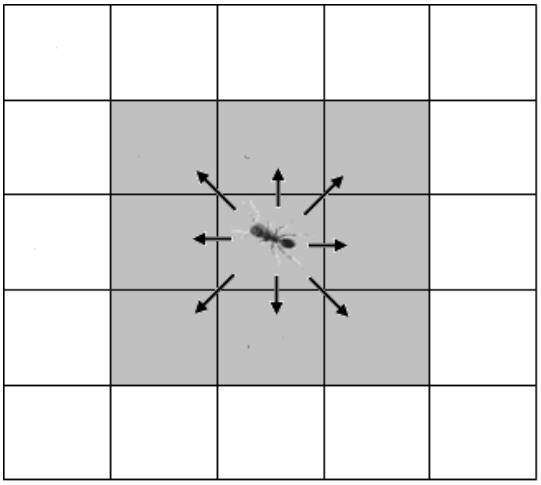


Figura 2.5: Representação de um *grid* bidimensional com 5×5 células. Uma formiga localizada no centro do *grid* e uma área cinza representando a vizinhança da formiga constituída por oito células referenciadas por setas.

O *grid* bidimensional é usado como um espaço de dimensão reduzida no qual serão projetados os objetos. No exemplo da Figura 2.5, a dimensão é $z = 2$ de forma que, após o término do processo de agrupamento feito pelas formigas, os grupos (*clusters*) apresentem a seguinte propriedade: as distâncias intra-clusters devem ser menores do que as distâncias inter-clusters. Além disso, o mapeamento obtido ao final do processo deve preservar a vizinhança inerente aos dados de entrada, evitando a criação de novos vizinhos no espaço $z$-dimensional que não existam no espaço $L$-dimensional correspondente ao espaço original do conjunto de dados.

No início da operação do algoritmo os dados de entrada devem ser projetados aleatoriamente sobre o *grid* bidimensional, como ilustrado na Figura 2.6. Formigas artificiais também são espalhadas pelo *grid* com as seguintes tarefas: andar pelo grid aleatoriamente; pegar e depositar objetos de acordo com algumas medidas de similaridade e respeitando as probabilidades $p_p$ e $p_d$, respectivamente.

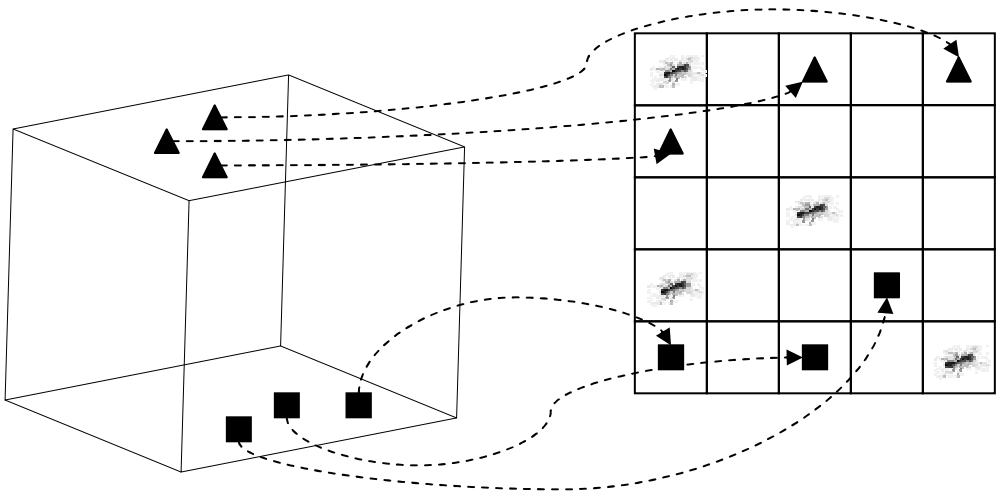


Figura 2.6: Projeção dos dados de entrada (representados num espaço tridimensional) dentro de um *grid* bidimensional. (Adaptado de: de Castro, 2006.)

Resolvida a questão de projeção dos itens, deve-se definir uma equação para $f$ que irá representar a fração de itens percebidos pela formiga em sua vizinhança. Dependendo da área de aplicação, $f$ pode ser obtida de várias formas, de acordo com o problema em estudo. Por exemplo, no contexto de agentes robóticos usado por Deneubourg *et al.* (1991), $f$ foi definida como sendo o quociente entre o número $M$ de itens encontrados durante as últimas $T$ unidades de tempo e o maior número possível de itens que possa ser encontrado durante essas $T$ unidades de tempo. Na área de análise exploratória de dados, Lumer & Faieta (1994) definiram $f$ como sendo uma função que calcula a densidade local de itens presentes na vizinhança de um objeto $x_i$ situado na posição $r$ do grid:

$$f(x_i) = \begin{cases} \dfrac{1}{s^2} \displaystyle\sum_{x_i \in Neig_{(s\times s)}(r)} \left[1 - \frac{d(x_i, x_j)}{\alpha}\right] & se\ f > 0 \\ 0 \quad caso\ contrário \end{cases} \qquad (2.3)$$

onde $f(x_i)$ é a medida de similaridade média do objeto $x_i$ com um outro objeto $x_j$ na vizinhança de $x_i$; $s^2$ define o número de células que constituem a vizinhança ao redor de $r$; e $\alpha$ é o fator que define a escala de dissimilaridade entre objetos, ou seja, $\alpha$ determina quando dois itens devem ou não estar localizados próximos um do outro. Por exemplo, se o valor de $\alpha$ for grande, não irá existir muita discriminação entre os itens, tendendo a formação de agrupamentos compostos por itens que não deveriam pertencer ao mesmo cluster e vice-versa. O valor $d(x_i,x_j)$ é a métrica de distância, por exemplo Euclidiana, entre

dois dados no seu espaço de origem $L$-dimensional. Um fato importante que deve ser ressaltado é que essa distância $d(\cdot,\cdot)$ entre os dados é calculada levando-se em conta o espaço original dos dados e não o espaço de projeção (bidimensional).

Levando-se em conta que tanto $f$ como $d(\cdot,\cdot)$ podem ser definidas de acordo com o problema, o ACA pode ser resumido conforme mostra o Algoritmo 2.1.

```
ACA (max_it, N, k1, k2) {
    //fazer a projeção aleatória de todos os dados sobre células vazias do grid
    //colocar as N formigas no grid de forma aleatória
    t←0;
    enquanto (t<max_it){
            para i←1 até N {
                calcule f(x_i);          //por exemplo, Equação (2.3)
                se a formiga estiver descarregada e a célula estiver ocupada pelo
                        item x_i, então calcule p_p(x_i);   //Equação (2.1)
                        pegue x_i com probabilidade p_p(x_i)
                senão se a formiga estiver carregando o item x_i e célula vazia então
                        calcule p_d(x_i);  //Equação (2.2)
                        deposite x_i com probabilidade p_d(x_i)
                fim se
                movimentar formiga aleatoriamente;
            } //fim para
            t←t+1;
      }//fim enquanto
      imprimir a disposição final dos dados no grid;
}//fim ACA
```

Algoritmo 2.1: Resumo dos passos existentes no ACA.

O Algoritmo 2.1 representa os passos utilizados durante o processo de agrupamento de dados inspirado nas tarefas de organização e limpeza do ninho no caso específico do comportamento de agrupar formigas mortas em regiões próximas ao ninho. O ACA tem sido aplicado em várias áreas, desde análise de dados até sistemas robóticos. Entretanto, verifica-se na literatura que a versão original apresentada acima possui como principal inconveniente a determinação de um número de grupos maior no espaço projetado do que no espaço original. No Capítulo 4 serão dados mais detalhes sobre este algoritmo, bem como será feita uma breve revisão bibliográfica, em ordem cronológica, da pesquisa na área. Por fim, será proposta uma versão aperfeiçoada do ACA para amenizar o problema de determinação de múltiplos grupos por classes apresentado pelo algoritmo original.

## 2.4. Sumário do Capítulo

Esse capítulo apresentou sucintamente de que forma a biologia vem sendo utilizada como inspiração para o desenvolvimento de algoritmos voltados à resolução de problemas complexos, com ênfase nas ferramentas (algoritmos genéticos e algoritmo de agrupamento por colônia de formigas) a serem exploradas nesta dissertação. Ciente de que cada abordagem apresentada faz parte de uma linha de pesquisa que envolve uma extensa fonte bibliográfica composta de livros e artigos científicos, cada seção deste capítulo procurou descrever de maneira sucinta as principais abordagens existentes e suas respectivas fontes de inspiração biológica.

# 3. Filtragem Evolutiva de Informação para Otimização de Busca na Web

## 3.1. Introdução

A Internet pode ser vista como um repositório global de recursos e informação. Na maioria dos casos, estes recursos estão disponíveis de forma imediata para uso, cobrindo os mais variados domínios, desde suporte a atividades científicas e educativas, até recreação e entretenimento. Um estudo feito pelo centro de comunicação UCLA destaca as três razões principais que fazem com que novas pessoas acessem a Internet (Lebo, 2004): *i*) obter e recuperar informação rapidamente; *ii*) necessidades profissionais; e *iii*) acesso à comunicação via correio eletrônico (*e-mail*). Além disso, a Internet vem reduzindo os custos de produção e distribuição de informação. Como resultado, uma avalanche de material, em muitos casos de baixa qualidade, torna-se disponível diariamente na *web*.

Dessa forma, o sucesso na obtenção de informações relevantes na *web* fica a cargo do usuário, tarefa que mesmo para os mais experientes nesse ambiente, tais como professores, alunos e pesquisadores, demanda um tempo considerável na busca e filtragem de informações relevantes.

Se tomarmos como exemplo a comunidade científica, fica evidenciado que o volume de literatura científica disponibilizada na Internet excede a capacidade dos pesquisadores em identificar e utilizar todas as informações relevantes existentes na sua área de pesquisa. Mesmo levando em consideração que esse material varia de acordo com a área pesquisada, mais de um milhão de artigos científicos estão disponíveis na *web* para livre acesso, seja através de sítios de conferências que disponibilizam acesso gratuito *on-line* ou por pesquisadores e seus grupos de pesquisa que disponibilizam seus trabalhos em formato eletrônico (Lawrence, 2001).

Ao mesmo tempo em que o avanço tecnológico possibilita uma infra-estrutura de rede que suporta os mais variados recursos de informação (tais como objetos multimídia, documentos estruturados e bancos de dados especializados), há também uma necessidade de desenvolvimento de aplicações voltadas para o cliente e que auxiliem o usuário na busca e acesso a informação.

As *máquinas de busca* estão entre as primeiras aplicações voltadas a auxiliar o usuário na busca de informação na *web*. De maneira geral, as consultas são feitas por palavras-chave e os resultados são apresentados de forma organizada, tendo como proposta fazer isto de uma maneira rápida e eficiente. Com o passar dos anos, um grande número de máquinas de busca surgiram na *web*, sendo o Google a máquina de busca mais utilizada no mundo (Nielsen, 2004).

Apesar de ser incontestável a contribuição das máquinas de busca no acesso a informação na Internet, cabe ainda ao usuário a responsabilidade de analisar e filtrar os resultados obtidos por tais máquinas, o que ainda demanda um tempo considerável. Além disso, as máquinas de busca encontradas hoje na *web* carecem seriamente de funções que permitam buscas repetidas através do tempo por um usuário. Elas não retêm informações sobre o usuário, como suas preferências, para uma utilização posterior e auxílio nas buscas, melhorando-as e/ou refinando-as, por exemplo. Os *sistemas de filtragem de informação* podem atuar como mediadores entre as fontes de informação existentes na *web* e os interesses de pesquisa do usuário (Vizine, 2000).

Sistemas de filtragem de informação são projetados para separar a informação requisitada pelo usuário de uma quantidade enorme de informação que é gerada dinamicamente. O termo *fonte de informação* é usado para representar lugares nos quais existam conteúdos que sejam de interesse do usuário (Belkin & Croft, 2002). Na maioria das vezes, as fontes de informação são relacionadas a locais onde existe uma coleção de documentos na forma de texto. Nesta dissertação, fontes de informação correspondem a sítios na Internet (*web sites*) contendo documentos no formato PDF (*Portable Document Format*).

Nesse capítulo será apresentado um sistema evolutivo utilizado para a otimização da busca de informação na *web*. A Seção 3.2 faz um breve retrospecto sobre *filtragem de informação* e os modelos mais utilizados no processo de filtragem de informação. A Seção 3.3 faz uma pequena revisão bibliográfica sobre algoritmos evolutivos utilizados na busca de informação na *web*. A Seção 3.4 descreve a arquitetura do sistema proposto e a Seção 3.5 apresenta o algoritmo genético utilizado no processo de busca. O capítulo termina apresentando os resultados obtidos na avaliação de desempenho do sistema proposto.

## 3.2. Filtragem de Informação

O termo *filtragem de informação* apareceu pela primeira vez em um artigo escrito por P. J. Denning (1982). Esse artigo descreve a necessidade de filtragem de informação nas mensagens eletrônicas, no intuito de separar mensagens urgentes das rotineiras, além de restringir a apresentação de mensagens rotineiras de forma a se adaptar à capacidade de leitura do usuário. Na década seguinte, pesquisas ocasionais sobre filtragem de informação surgiram na literatura. Enquanto as mensagens eletrônicas eram o foco principal, pesquisas subseqüentes apontavam para textos digitalizados, artigos da *Internet News* e a ampliação dos recursos da rede.

O propósito da filtragem de informação é disponibilizar a informação que é relevante ao usuário de maneira rápida e eficiente. Entretanto, as necessidades de informação variam de usuário para usuário. Logo, os sistemas de filtragem de informação devem ser personalizados de forma a servir aos interesses individuais de cada usuário, assumindo o papel de uma espécie de assistente pessoal. Um sistema personalizado de filtragem de informação deve satisfazer três requisitos básicos (Sheth, 1994):

- **Especialização:** Um sistema de filtragem personalizado deve servir aos interesses específicos do usuário. O sistema seleciona os textos interessantes ao usuário e elimina o restante. Entretanto, um sistema de filtragem pode não estar apto a diferenciar, de maneira apropriada, textos que são realmente relevantes para o usuário daqueles que

não são. A quantidade de textos irrelevantes entregues ao usuário deve ser a menor possível e a proporção de textos relevantes eliminados também deve ser pequena. Visto que a filtragem envolve uma seqüência interativa com o usuário, o sistema deve estar apto a identificar padrões de comportamento do usuário. O sistema deve inferir os hábitos do usuário e especializar-se, ou seja, fazer a recomendação de textos relevantes e procurar minimizar o número de recomendações irrelevantes.

- **Adaptação:** Uma vez que a filtragem se faz num processo iterativo por longos períodos de tempo, é natural que os interesses do usuário não permaneçam constantes. Quando mudanças ocorrerem, como, por exemplo, surgir o interesse por um novo assunto, o sistema deverá estar preparado para perceber que ocorreu uma mudança nos interesses do usuário e se adaptar a estas mudanças.

- **Exploração:** Um sistema de filtragem deve também ser capaz de explorar novos domínios a fim de encontrar algo de potencial interesse ao usuário.

Um sistema de filtragem de informação deve prontamente atender as necessidades do usuário. O sistema deverá ter a capacidade de perceber ou deduzir as necessidades do usuário através de um processo interativo. Assumindo que grande parte das ações tomadas pelo usuário são relevantes, o sistema deverá aumentar o acerto das sugestões feitas à medida que o tempo passa. Ou seja, o sistema deverá atuar de forma que as necessidades do usuário sejam atendidas consistentemente.

Por outro lado, é de se esperar que os interesses dos usuários não permaneçam constantes. Essa mudança de interesse pode ser uma simples diversificação num dado assunto, como também o interesse por um assunto completamente novo ou até a perda de interesse por um assunto até então muito relevante. O sistema deve, portanto, estar apto a detectar ou permitir ao usuário indicar uma mudança de interesse e adaptar-se de forma a responder a essa mudança.

Finalmente, levando-se em conta a motivação do usuário na busca de informação, deveríamos considerar a hipótese do sistema ser capaz de recomendar novas e interessantes fontes de informação baseado no conhecimento que ele possui sobre o comportamento do usuário.

As necessidades de informação do usuário devem ser representadas através de uma *expressão de busca*, que deve resultar na recuperação de um número de documentos que possibilite a verificação de cada um deles a fim de selecionar os que são úteis. A principal dificuldade do usuário está em predizer, por meio de uma expressão de busca, as palavras ou expressões que devem ser usadas para representar os documentos que são de seu interesse. A maneira pela qual essa *expressão de busca* é construída varia de acordo com o *modelo* utilizado. O modelo, por sua vez, influencia diretamente o comportamento do sistema. Apesar de alguns desses modelos terem sido criados nos anos 60 e 70 e aperfeiçoados nos anos 80, as suas principais idéias ainda estão presentes nos sistemas atuais de filtragem e recuperação de informação e nas máquinas de busca da *web* (Ferneda, 2003).

### 3.2.1. Modelo Booleano

O modelo booleano adota a representação de um documento como sendo formado por um conjunto de termos de indexação que são gerados pelo usuário, ou automaticamente por algum algoritmo. As expressões de busca são construídas através de expressões booleanas formuladas pela interligação dos termos com os operadores lógicos E, OU e NÃO. Como resultado, são apresentados os documentos que satisfazem as restrições lógicas da expressão de busca.

Uma expressão de busca conjuntiva de enunciado $t_1$ **E** $t_2$ trará como resultado todos os documentos indexados por ambos os termos ($t_1$ e $t_2$). Essa operação equivale à interseção do conjunto de documentos indexados pelo termo $t_1$ com o conjunto de documentos indexados pelo termo $t_2$.

Uma expressão de busca disjuntiva $t_1$ **OU** $t_2$ retorna o conjunto de documentos indexados pelo termo $t_1$ ou pelo termo $t_2$. Essa operação é equivalente à união do conjunto de documentos indexados pelo termo $t_1$ com o conjunto de documentos indexados pelo termo $t_2$.

A expressão de busca formulada com o operador **NÃO** como, por exemplo, $t_1$ **NÃO** $t_2$ retornará o conjunto de documentos indexados pelo termo $t_1$ e que não sejam indexados pelo termo $t_2$.

O modelo Booleano é empregado na maior parte dos sistemas de filtragem de informação, seja como a principal maneira de formular as expressões de busca, seja como recurso alternativo (Vizine, 2000). A Tabela 3.1 sumariza suas principais vantagens e desvantagens.

Tabela 3.1: Vantagens e desvantagens do Modelo Booleano.

| | |
|---|---|
| **Vantagens** | • Facilidade de implementação.<br>• Eficiência computacional.<br>• Presente na maioria dos sistemas on-line de recuperação e filtragem de informação, inclusive em máquinas de busca.<br>• A combinação de termos e operadores permite a formulação de expressões de busca mais detalhadas e restritivas. |
| **Desvantagens** | • Sem treinamento apropriado um usuário leigo fica restrito a formular consultas simples, pois consultas mais complexas exigem um conhecimento mínimo de lógica Booleana.<br>• Existe pouco controle sobre a quantidade de documentos resultantes da busca, problema conhecido como *null output* ou *overload output* (Cooper, 1997).<br>• Todos os termos possuem a mesma importância (o mesmo peso).<br>• Ausência de ordenação dos documentos resultantes da busca (*ranking*). |

### 3.2.2. Modelo de Espaço Vetorial

No modelo de espaço vetorial cada documento é representado por um vetor de termos de indexação no qual cada elemento do vetor indica o peso ou a relevância do termo para o documento. Os pesos de cada termo devem ser normalizados para assumirem valores

entre zero e um. Para o cálculo dos pesos utiliza-se uma técnica que faz o balanceamento entre as características do documento utilizando o conceito da freqüência de um termo no documento. Valores próximos ou iguais a um indicam que o termo possui um alto valor descritivo para o documento, ou seja, aparece com alta freqüência. A mesma representação é utilizada para as expressões de busca.

Uma expressão de busca $q$ pode ser representada por $q = \{qt_1, qt_2,...,qt_n\}$ onde $qt_i$, $\forall i$, representa o peso ou relevância do *i-ésimo* termo da expressão de busca $q$. Dessa forma, a similaridade entre dois vetores quaisquer (por exemplo, uma expressão de busca e um documento) pode ser representada como uma função inversamente proporcional ao ângulo entre eles. Isto é, quando dois vetores são exatamente iguais, o ângulo entre eles é zero. Uma das medidas de similaridade mais utilizadas no modelo de espaço vetorial é a *medida do cosseno* (Salton & McGill, 1997).

Muito embora a maioria dos modelos clássicos foi proposta na década de 60, atualmente eles ainda são amplamente utilizados em ferramentas de busca para *web* e em mineração de dados que envolvam recuperação de bases textuais. Além disso, eles também contribuem para o desenvolvimento de novas técnicas de representação de conhecimento ligadas à área de inteligência artificial.

Como abordado anteriormente, características como aprendizado e adaptação são fundamentais num sistema de filtragem de informação. Sheth (1994) demonstrou que os modelos clássicos e os algoritmos genéticos podem ser utilizados em conjunto para a construção de sistemas de filtragem de informação. Os resultados experimentais do seu trabalho sugerem que os algoritmos genéticos constituem uma abordagem promissora para representar a adaptação e a exploração de novos ambientes.

## 3.3. Algoritmos Evolutivos e Busca de Informação na *web*: Revisão Bibliográfica

A maioria das ferramentas de busca de informação na *web* não leva em consideração os interesses do usuário. Toda interação feita pelo usuário durante o refinamento da busca, até que a informação necessária seja encontrada, não fica armazenada. Esse histórico de busca é essencial, pois representa as áreas de interesse de pesquisa de um determinado usuário. Uma das abordagens para construir esse histórico é a extração de termos (palavras) de documentos que o usuário julga relevante. Um perfil de interesse de usuário pode ser construído com esses termos e com o passar do tempo esse perfil vai se adaptando aos interesses específicos de cada usuário.

Procurando não só extrair termos de documentos relevantes, Bautista *et al* (1999) desenvolveram um sistema para a construção de perfis de usuário utilizando um classificador genético-nebuloso (*fuzzy-genetic*) de termos. O sistema é realimentado pelo usuário, o qual atribui um conceito $C \in [0,1]$ para cada documento recuperado. Conceitos $C < 0,5$ indicam documentos pouco relevantes, $C = 0,5$ representa documentos neutros, $C > 0,5$ indica bons documentos e $C = 1,0$ representa um documento muito relevante. O peso de cada termo é um número nebuloso (*fuzzy*) calculado por um algoritmo classificador que leva em consideração o conceito atribuído $C$ e a freqüência relativa de cada termo. Terminada a etapa de classificação dos termos, uma seleção genética é realizada, procurando evoluir termos mais significativos que irão constituir o perfil do usuário.

O processo de busca de informação na *web* muito se assemelha a problemas de otimização. O espaço de busca pode ser representado por um grafo, no qual as páginas *web* representam os vértices e os *links* correspondem às arestas. Baseado nessa representação e sabedores de que algoritmos genéticos são amplamente utilizados em problemas de busca e otimização, Picarougne *et al.* (2002) utilizaram um algoritmo genético para explorar novas páginas relacionadas à pagina que está sendo consultada pelo usuário. Cada indivíduo da população é uma página *web* e a função de avaliação (*fitness*) a ser maximizada é baseada na realimentação do usuário ao analisar o seu conteúdo.

Um agente pessoal que busca fontes de informação na *web* e recupera documentos de acordo com os interesses do usuário foi desenvolvido por Valim & Coello (2003). Este sistema utiliza técnicas clássicas de recuperação de informação e um algoritmo genético para aprender e se adaptar às mudanças de interesses do usuário. O agente utiliza como parâmetro de busca uma coleção de documentos relevantes fornecidos pelo usuário. A cada documento recuperado pelo agente o modelo de espaço vetorial é utilizado para calcular a similaridade do documento recuperado com os documentos relevantes fornecidos pelo usuário. Se o documento recuperado possuir uma similaridade superior a um determinado limiar, o mesmo é apresentado ao usuário que manifestará ou não seu interesse pelo novo documento através de uma realimentação positiva ou negativa. Cada indivíduo da população representa uma consulta (*query*) formulada por termos extraídos dos documentos apresentados como exemplos. O *fitness* de cada indivíduo é calculado de acordo com a realimentação obtida do usuário e com o grau de similaridade do documento recuperado pela consulta com os demais apresentados como relevantes.

Uma ferramenta capaz de aprender os interesses e preferências do usuário através da observação de seu comportamento durante sua navegação na *web*, chamada de SmartSeek, é apresentada como uma nova estratégia de mineração de dados na *web* por Joshi & Todwal (2003). O SmartSeek constrói um perfil do usuário através da extração de palavras e frases de páginas *web* que foram visitadas pelo usuário. Um algoritmo genético é então utilizado, no qual cada indivíduo da população é uma expressão de busca formulada pelos termos encontrados no perfil do usuário e operadores lógicos (E, OU e NÃO). Os operadores lógicos são gerados aleatoriamente, juntamente com um conjunto de palavras específicas, chamadas pelos autores de categorias, que tentam captar visões diferentes de interesse para um mesmo assunto. Essas diferentes visões podem ser retratadas por palavras do tipo INTRODUÇÃO, HISTÓRIA, APLICAÇÃO, CONCEITO, etc. Dessa forma, uma expressão de busca é formulada da seguinte maneira: CONSULTA = CATEGORIA + TERMOS + OPERADORES LÓGICOS.

O número de ambientes virtuais de aprendizagem encontrados na Internet aumenta a cada dia. Acompanhando esse crescimento, estão os cursos ministrados nesses ambientes e

a quantidade de informação disseminada e produzida em cada um deles. Controlar o fluxo de informação torna-se uma tarefa árdua. No trabalho de Romero *et al.* (2004), um algoritmo genético é utilizado em um curso de medicina com o objetivo de extrair regras que possam auxiliar o professor na produção do conteúdo a ser ministrado no curso. Essas regras têm como propósito avaliar o desempenho obtido pelos alunos nos estudos de caso propostos durante o transcorrer do curso. Com este tipo de realimentação obtido pelas regras, o professor pode alterar a trajetória do curso caso haja necessidade.

A tecnologia de agentes também vem sendo empregada na busca de informação na *web*. Um sistema baseado em agentes inteligentes, chamado Webnaut's (Nick & Themis, 2001), utiliza um algoritmo genético para coletar e recomendar páginas *web*. O sistema possui um agente que analisa o comportamento do usuário, procurando identificar suas preferências e áreas de interesse. O Webnaut's utiliza o modelo de espaço vetorial para comparar as páginas visitadas pelo usuário com um perfil de usuário criado com termos extraídos das páginas relevantes já visitadas. Um algoritmo genético é utilizado para explorar a *web*, procurando conteúdos relevantes ao perfil do usuário. Diferente das abordagens tradicionais, o algoritmo genético empregado utiliza duas populações de indivíduos que co-evoluem. Uma população é constituída por cromossomos formados pelos termos presentes no perfil do usuário e a outra população é composta por indivíduos nos quais cada gene representa um operador lógico (E, OU e AND) escolhido aleatoriamente. A expressão de busca é formulada intercalando-se os genes de um indivíduo de cada população. O *fitness* é calculado mediante a realimentação dada pelo usuário.

## 3.4. Sistema de Busca Proposto

Assim como acontece em congressos e conferências maiores de âmbito nacional ou internacional que oferecem oportunidade de encontro com pessoas de lugares geograficamente distantes, o desenvolvimento de comunidades virtuais acadêmicas pode viabilizar encontros virtuais entre alunos e pesquisadores distantes, e conseqüentemente, facilitar o envolvimento em atividades de pesquisa através do uso de recursos tecnológicos. No projeto desses ambientes virtuais um fator importante a ser considerado é o crescimento

constante da quantidade de informação e conhecimento armazenado pela comunidade. A evolução, portanto, é um fator incorporado à própria estrutura da comunidade. É importante salientar também que crescimento não significa apenas acrescentar mais informação, mas também tudo o que isto implica: sistemas de busca, classificação, síntese, participação e interação, organização e visualização da informação.

Nesta parte da dissertação é apresentado um sistema de busca de artigos científicos que poderá ser utilizado para prover uma comunidade virtual acadêmica adaptativa com documentos relevantes aos interesses dos membros da comunidade, de forma dinâmica e autônoma. O sistema é composto por duas ferramentas que foram desenvolvidas utilizando um conjunto de técnicas bem sucedidas empregadas em filtragem de informação na *web*, procurando explorar o potencial de cada uma delas: *i*) um algoritmo genético; e *ii*) um extrator automático de palavras-chave. O algoritmo genético é responsável por efetuar a busca empregando uma representação dos interesses do usuário durante a filtragem de informação. O algoritmo extrator de termos (palavras-chave), por sua vez, encarrega-se de extrair automaticamente palavras-chave (termos) dos documentos recuperados pelo algoritmo genético para compor o perfil de interesse do usuário. As *expressões de busca* (*queries*) são formuladas a partir de palavras existentes no perfil do usuário utilizando o modelo Booleano. Depois de formuladas, estas expressões são submetidas ao Google, uma das principais máquinas de busca da atualidade. A avaliação do resultado obtido na busca é feita utilizando-se o modelo de espaço vetorial. As sub-seções a seguir detalham a funcionalidade de cada componente do sistema.

### 3.4.1. Algoritmo Extrator de Termos

O algoritmo extrator de termos tem como objetivo principal identificar termos dentro de cada documento que o algoritmo de busca tenha baixado (*download*) da *web*. Cada documento passa pelas seguintes etapas:

- Transformação do documento PDF em arquivo texto;
- Identificação de palavras;
- Remoção de palavras irrelevantes;

- Cálculo da freqüência relativa dos termos; e
- Atualização do dicionário de termos.

**Transformação de um Documento PDF em Arquivo Texto**

Como discutido anteriormente, a Internet congrega uma vasta quantidade de fontes de informação dos mais variados formatos e com diversas peculiaridades de acesso. O sistema proposto nesse trabalho restringe-se a fontes de informação no formato PDF, pois grande parte dos trabalhos publicados no âmbito científico e educacional encontra-se nesse formato. Arquivos PDF não são tão comuns como arquivos HTML, mas freqüentemente contêm informação de alta qualidade que não pode ser encontrada em outros locais ou tipos de documento.

Entretanto, a escolha por trabalhar com documentos do tipo PDF gerou algumas dificuldades adicionais, pois embora exista uma variedade de ferramentas gratuitas para converter um arquivo texto para formato PDF são poucas as ferramentas gratuitas que permitem converter arquivos PDF para texto. Tal conversão se faz necessária para possibilitar a identificação das palavras relevantes no documento que irão servir como descritores do mesmo.

A ferramenta escolhida foi a PDFBox, um pacote de código livre de classes JAVA que acessam documentos PDF (PDFBox, 2004). Dentre as classes existentes, uma possibilita a conversão de arquivos PDF em arquivos texto fazendo ressalva apenas a arquivos PDF que foram gerados por meio de documentos digitalizados, ou seja, obtidos através de um *scanner*.

**Identificação de Palavras no Documento**

Nessa etapa, o documento PDF convertido para texto passa por um processo de padronização, *tokenização* e limpeza. Todos os caracteres do texto são convertidos para caracteres minúsculos, fazendo com que palavras como, por exemplo, ‘ALGORITMO’ e

‘algoritmo’, sejam interpretadas como idênticas pelo algoritmo extrator. Os *tokens* são gerados obedecendo a seqüências de caracteres (*strings*) existentes entre espaços em branco no documento. Em seguida, são eliminados todos os caracteres do início de cada *string* que não fazem parte do alfabeto (caracteres de ‘a’ a ‘z’). As cadeias de caracteres remanescentes e com tamanho maior ou igual a três são consideradas palavras.

**Remoção de Palavras Irrelevantes**

Palavras cujo comprimento seja menor que três caracteres são eliminadas, pois na maior parte dos testes realizados estas palavras referiam-se a artigos e preposições, os quais estão presentes na maioria dos documentos e, portanto, não possuem força descritiva. A retirada dessas seqüências de caracteres aumenta o desempenho do sistema de filtragem e reduz o tempo de processamento. As palavras remanescentes são então confrontadas com um arquivo de palavras irrelevantes para efeito de identificação de termos conhecidos na literatura como *stopwords* (Fox, 1992). Em geral, a lista de *stopwords* é composta por palavras sem valor informativo, como os artigos, pronomes, adjetivos, advérbios e preposições (dos, das, muito, até, etc.).

Um outro fator a ser considerado é o idioma em que o documento foi redigido. O sistema está preparado para identificar e eliminar palavras irrelevantes de documentos escritos em inglês, com um arquivo de *stopwords* com 420 palavras. Este arquivo pode ser editado e atualizado pelo usuário com um editor de texto comum. No Anexo I estão reunidas, em ordem alfabética, todas as palavras que constituem o arquivo de *stopwords* empregado nesta dissertação.

**Cálculo da Freqüência dos Termos**

Para uma palavra ser indexada no dicionário de termos, ou seja, ser classificada como uma palavra-chave, ela precisa ser um bom descritor de um conjunto de documentos

existentes em uma das áreas de interesse do usuário. Por exemplo, seja uma palavra $w$ e uma área de interesse $D$. A palavra $w$ deve ter as seguintes propriedades:

1. Ser predominante em $D$ quando comparada a outras palavras em $D$,ou seja, possuir $G(w) \geq \theta$.
2. Ser predominante em $D$ quando comparada sua ocorrência em todas as demais áreas de interesse.

O método de extração de palavras-chave adotado foi proposto por Lagus & Kaski, 1999 e opera da seguinte maneira. Seja $G(w)$ a classificação obtida para a palavra $w$:

$$G(w) = F^{cluster}(w) \times F^{coll}(w) \tag{3.3}$$

onde $F^{cluster}$ relaciona a palavra $w$ com todas as outras palavras existentes nos documentos de uma dada área de interesse, e o segundo termo da Equação (3.3), $F^{coll}$, relaciona a palavra $w$ com todas as outras palavras existentes em todas as áreas de interesse.

Dessa forma, se $f_j(w)$ corresponde ao número de vezes em que a palavra $w$ aparece na área de interesse $j$, isto é, a freqüência da palavra $w$ em $j$, então $F_j(w)$ representa a freqüência relativa da palavra $w$ definida como:

$$F_j(w) = \frac{f_j(w)}{\sum_v f_j(v)}; \quad v \neq w \tag{3.4}$$

É importante notar que $0 < F_j(w) < 1$ e $\Sigma_w F_j(w) = 1$. Esta normalização tem como propósito desconsiderar a quantidade de palavras por área de interesse e, ao invés disso, calcular a importância relativa de uma palavra com outras palavras existentes na mesma área. A freqüência relativa $F_j(w)$ assume o papel de $F^{cluster}$ na Equação (3.3). Para determinar a representatividade da palavra $w$ em todas as áreas de interesse existentes, $F^{coll}(w)$, é utilizada a seguinte equação:

$$F^{coll}(w) = \frac{F_j(w)}{\sum_i F_i(w)}; \; i \neq j. \tag{3.5}$$

Dessa forma, torna-se possível determinar a classificação *G* da palavra *w* que aparece na área de interesse *j* como:

$$G(w,j) = F_j(w)\frac{F_j(w)}{\sum_i F_i(w)}. \qquad (3.6)$$

Palavras com uma classificação (*G*) superior a um limiar θ previamente especificado serão inseridas como termos no dicionário de termos.

**Atualização do Dicionário de Termos**

À medida em que o número de documentos vai aumentando, o algoritmo extrator precisa repetir todas as etapas anteriores no intuito de recalcular e atualizar a classificação das palavras-chave extraídas de todos os documentos. Isso ocorre, pois poderão ocorrer alterações substanciais na classificação das palavras (Equação 3.6), sendo algumas delas inseridas no dicionário de termos e outras já existentes podendo ser removidas.

Por ser um processo demorado, esse procedimento pode ser parametrizado das seguintes maneiras:

- Executar esse algoritmo toda vez que o número de documentos ultrapasse um percentual previamente estipulado.
- Executar esse algoritmo em intervalos de tempo, por exemplo, dias, previamente estipulados.

### 3.4.2. Algoritmo de Busca

O método de busca utilizado faz uso de um algoritmo evolutivo para explorar e sugerir novos documentos para cada área de interesse do usuário. Utilizando o dicionário de termos como o conjunto de palavras que irão gerar o espaço de busca para o problema de sugerir novos documentos, um algoritmo genético é usado para descobrir quais termos

constituem um conceito e, a partir desses conceitos, quais os que melhor representam uma dada área de interesse.

Um dicionário de termos da área de interesse chamada *computação evolutiva*, por exemplo, poderia conter as seguintes palavras: genéticos, algoritmos, sistemas, inteligentes, redes, artificiais e neurais. Digamos que o interesse de pesquisa do usuário seja *algoritmos genéticos* e *sistemas inteligentes*, mas não *redes neurais artificiais*. Nesse caso, os termos podem ser combinados com operadores lógicos para descrever o interesse do usuário da seguinte forma:

```
algoritmos E genéticos E sistemas E inteligentes NÃO redes NÃO neurais
NÃO artificiais.
```

Diferente das abordagens tradicionais, o algoritmo genético implementado trabalha com duas populações de indivíduos que co-evoluem (Nick & Themis, 2001). O Algoritmo 3.1 descreve os passos que compõem o algoritmo genético empregado, onde: *tot_ger* corresponde ao total de gerações estabelecido previamente; *p_termos* é a população que é constituída por termos do dicionário de termos de cada área de interesse do usuário; *p_oplog* é a população de operadores lógicos; e *pc* é a probabilidade de *crossover*, valor parametrizado. As principais funções que compõem o algoritmo serão detalhadas na seqüência.

```
AG_BUSCA( ) {
        t ← 1;
        inicializa(p_termos);
        inicializa(p_oplog);
        merge(p_termos, p_oplog);
        avalia p_termos( );
        avalia p_oplog( );
        repita{
                crossover_p_termos(pc);
                crossover p_oplog(pc);
                merge(p_termos, p_oplog);
                avalia p_termos( );
                avalia p_oplog( );
                seleciona p_termos( );
                seleciona p_oplog( );
                t ← t+1;
           }enquanto(t<tot_ger)
}
```

Algoritmo 3.1: Pseudo-código do algoritmo genético empregado.

**Inicialização das Populações**

Na população denominada 'população de termos', cada cromossomo é constituído por um conjunto de termos extraídos de forma aleatória do dicionário de termos. Os cromossomos da 'população de operadores lógicos' contêm o mesmo número de genes dos cromossomos da população de termos, assim como o mesmo número de indivíduos. Cada gene da segunda população pode assumir um dos três operadores lógicos E, OU e NÃO, escolhidos aleatoriamente.

**Função Merge**

Após a inicialização das duas populações, a função merge tem como objetivo concatenar os genes das duas populações, intercalando-os para formar uma *consulta* que será submetida à máquina de busca do Google. A Figura 3.1 ilustra um exemplo de operação da função merge.

| results | Set | genetic | selection | generation | evolution | individual |
|---|---|---|---|---|---|---|

(a)

| E | OU | OU | OU | NÃO | NÃO | NÃO |
|---|---|---|---|---|---|---|

(b)

```
E results OU set OU genetic OU selection NÃO generation NÃO evolution NÃO
individual
```

(c)

Figura 3.1: Exemplo de operação da função merge. (a) Cromossomo da população de termos. (b) Cromossomo da população de operadores lógicos. (c) Resultado obtido pela função merge que irá ser transformado numa consulta.

**Função de Avaliação**

O resultado obtido a partir da função merge é transformado numa sintaxe padrão do Google. O Google foi escolhido, pois além de ser uma das principais máquinas de busca (seu índice contém mais de 1 bilhão de URLs), ele provê uma API (Google Web API) que

possibilita o desenvolvimento de *softwares* que acessem diretamente sua máquina de busca sem a necessidade de acessar a sua URL para fazer uma *consulta*. O resultado do merge da seção anterior seria submetido ao Google da seguinte forma:

```
results OR set OR genetic OR selection –generation –evolution –
individual – crossover filetype:PDF
```

Uma pequena adaptação deve ser feita, entretanto, pois é necessário substituir o operador E por um espaço em branco e o operador NÃO pelo hífen. Ao final da expressão de busca deve-se colocar a declarativa `filetype:PDF` restringindo a busca a documentos no formato PDF.

O modelo de espaço vetorial foi adotado para gerar uma representação que tornasse fácil o problema de determinação do *fitness* de cada indivíduo da população. Nesse modelo cada documento é representado por um vetor, no qual cada elemento representa o peso ou a relevância do respectivo termo no documento. Cada elemento (peso) do vetor que representa o documento recuperado na *consulta* deve ser normalizado de forma a assumir valores entre zero e um; onde valores próximos a um indicam um termo com alto poder descritivo. A freqüência relativa do termo no documento (Equação 3.4) foi utilizada para fazer essa normalização. Dessa forma, documentos e consultas são representados utilizando o mesmo modelo, o que facilita a adoção de uma medida de similaridade entre ambos. A Figura 3.2 ilustra a representação de uma consulta $q$ e um documento recuperado $d$ pelo Google em um espaço vetorial formado pelos termos $t_1$ e $t_2$.

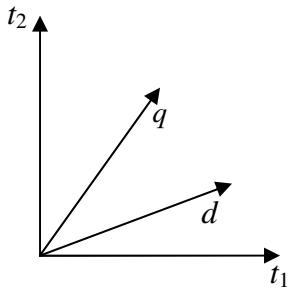



Figura 3.2: Representação da consulta $q$ e do documento $d$ num espaço bidimensional.

Em um espaço vetorial contendo *n* dimensões a similaridade *S* entre dois vetores *q* e *d* pode ser calculada através do coseno do ângulo formado por esses vetores utilizando a Equação 3.7.

$$S(q,d) = \frac{\sum_{i=1}^{n}(w_{iq} \times w_{id})}{\sqrt{\sum_{i=1}^{n}(w_{iq})^2} \times \sqrt{\sum_{i=1}^{n}(w_{id})^2}} \qquad (3.7)$$

onde $w_{iq}$ é o peso do *i-ésimo* elemento do vetor *q*, e $w_{id}$ é o peso do *i-ésimo* elemento do vetor *d.*

**Seleção de Indivíduos**

Depois de determinado o *fitness* de todos os indivíduos, um torneio binário é executado para selecionar os indivíduos que irão compor a próxima geração (Bäck *et al.*, 2000). O melhor indivíduo de cada população é preservado e não é submetido ao *crossover.*

**Crossover**

O operador de *crossover* escolhido foi o de um ponto com probabilidade *pc*. A única restrição deste operador é que uma mesma palavra não pode ocorrer duas vezes no mesmo cromossomo. Nesses casos, a operação de *crossover* não é aplicada e os cromossomos-pai permanecem inalterados

## 3.5. Avaliação de Desempenho do Algoritmo Evolutivo

No intuito de avaliar o desempenho do sistema, uma base de testes da área de Inteligência Computacional foi utilizada. O sistema foi testado num ambiente composto por três áreas de interesse: Computação Evolutiva (CE), Redes Neurais Artificiais (RNA) e Sistemas Fuzzy (SF). Os arquivos PDF utilizados nos testes foram copiados dos Anais do WCCI – *IEEE World Congress on Computational Intelligence* 2002: Proceedings do

IJCNN 2002, FUZZ-IEEE 2002 e CEC 2002. A Tabela 3.2 sumariza as informações pertinentes a cada área de interesse.

Tabela 3.2: Informações sobre os documentos armazenados em cada área de interesse.

| Área de interesse | Quantidade de artigos | Total de palavras | Total de palavras selecionadas para o dicionário de termos |
|---|---|---|---|
| **CE** | 347 | 60,348 | 36 |
| **RNA** | 519 | 75,761 | 37 |
| **SF** | 285 | 37,876 | 92 |

### 3.5.1. Análise de Desempenho da Extração de Termos

O algoritmo extrator de termos foi utilizado para determinar os termos a serem armazenadas no dicionário de termos de cada área. O valor adotado como limiar foi $\theta = 5\times10^{-5}$. Este valor foi escolhido empiricamente, mas existe uma relação entre o valor de $\theta$, o número e a qualidade dos termos selecionados. Valores altos de $\theta$ resultam em poucos termos selecionados com altos valores de classificação, enquanto valores baixos de $\theta$ resultam num aumento de palavras selecionadas para serem armazenadas no dicionário de termos com um baixo valor de classificação de cada uma delas. Isto pode ser observado na Tabela 3.3, na qual para $\theta = 10^{-3}$ 36 termos foram selecionados para representar a área de computação evolutiva, 37 para a área de redes neurais artificiais e 92 para a área de sistemas *fuzzy*.

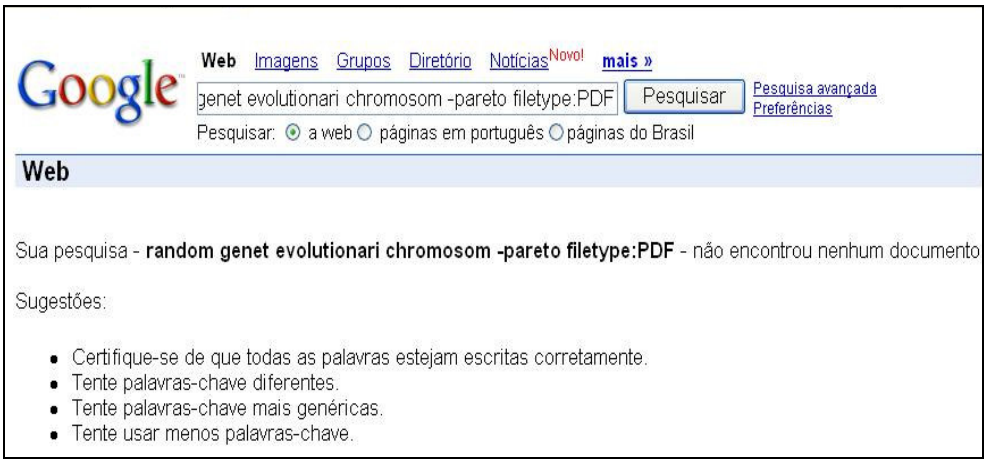
Tabela 3.3: Palavras selecionadas nos documentos pelo algoritmo extrator de termos usando um limiar $\theta = 10^{-3}$.

| | |
|---|---|
| **CE** | algorithm, algorithms, chromosome, crossover, diversity, dna, evolution, evolutionary, fitness, function, gene, generation, generations, genes, genetic, immune, individual, individuals, mutation, optimization, pareto, particle, population, pso, random, results, search, selection, set, size, solution, solutions, swarm, time, using, value |
| **RNA** | algorithm, approach, based, control, data, error, fig, figure, function, hidden, image, input, layer, learning, method, model, neural, neurons, output, paper, performance, process, proposed, results, set, shown, system, systems, table, time, training, using, value, values, vector, vol, weights |
| **SF** | agent, algorithm, algorithms, analysis, applications, applied, approach, attribute, attributes, base, vol, based, class, classification, cluster, clustering, clusters, control, controller, data, decision, using, defined, definition, degree, design, distance, error, example, feature, fig, figure, following, follows, form, function, functions, fuzzy, ieee, image, value, inference, input, knowledge, learning, level, linear, linguistic, logic, matrix, max, measure, membership, method, methods, min, mining, model, models, neural, obtained, output, paper, parameters, pattern, values, performance, process, proposed, relation, vector, result, results, rule, rules, section, set, sets, web, shown, similarity, space, step, structure, wkh, system, systems, table, theory, time, training, user |

Todas as palavras selecionadas pelo algoritmo extrator de termos estão relacionadas a suas respectivas áreas de interesse, como mostra a Tabela 3.3. Por exemplo, palavras como *evolution*, *dna* e *swarm* foram armazenadas no dicionário de termos da área de computação evolutiva. Palavras que são comuns a todas as áreas (p. ex. *value*, *algorithm* e *results*) foram armazenadas em todas as áreas. Pôde ser constatado também que palavras que possuem o mesmo radical (p. ex. *algorithm* e *algorithms*) foram selecionadas de forma distinta, segmentando de certa maneira o seu peso (freqüência relativa).

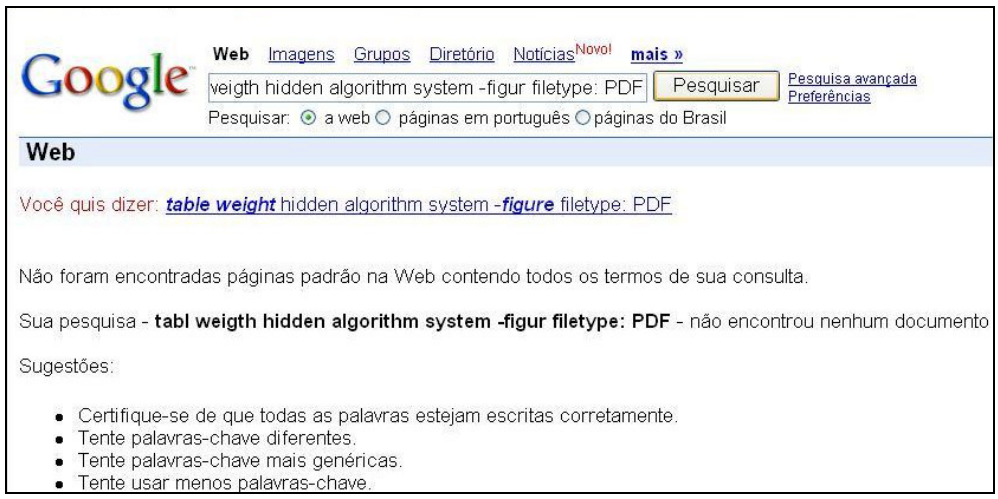
No intuito de diminuir a redundância e, conseqüentemente, aumentar a qualidade das palavras selecionadas o algoritmo de extração de radicais proposto por Porter (1980) foi testado. Entretanto, apesar do algoritmo de Porter contribuir para a diminuição de palavras redundantes e propiciar um pequeno aumento na freqüência relativa das palavras, ele compromete de maneira significativa a qualidade dos indivíduos gerados para a população do algoritmo genético de busca, pois ele generaliza termos que deveriam ser especializados. A Figura 3.3 mostra um resultado obtido a partir das consultas submetidas ao Google

utilizando o algoritmo de Porter nos termos. A Tabela 3.4 mostra os documentos recuperados quando o algoritmo de Porter não é utilizado.




(a)


(b)




(c)

Figura 3.3: Resultados obtidos a partir das consultas submetidas ao Google utilizando o algoritmo de Porter. (a) Computação evolutiva; (b) Sistemas fuzzy; e (c) Redes neurais artificiais.

Para se ter uma idéia da relevância dos termos selecionados pelo algoritmo extrator, com uma consulta de somente seis palavras e seis operadores lógicos, o algoritmo de busca foi capaz de recuperar um documento contendo 22 dos 36 termos encontrados no dicionário da área de computação evolutiva.

### 3.5.2. Análise de Desempenho do Processo de Busca de Documentos

O algoritmo genético de busca foi parametrizado da seguinte forma:

- Número de gerações: 20
- Tamanho da população: 20 (ambas as populações: termos e operadores lógicos)
- Tamanho do cromossomo: 6 (ambas as populações: termos e operadores lógicos)
- Probabilidade de *crossover* (*pc*): 0.6
- Probabilidade de mutação (*pm*): 0

A Figura 3.4 apresenta a evolução do *fitness* do melhor indivíduo (*query*) da população e o *fitness* médio da população para uma execução do algoritmo. O exemplo apresentado na Figura 3.4 foi especialmente escolhido dentre os demais, pois retrata o comportamento do algoritmo genético num espaço de busca instável que é a Internet.

Apesar de utilizar a estratégia que preserva o melhor indivíduo de cada geração, é possível constatar, nesse exemplo em especial, que o melhor indivíduo foi perdido nas gerações 2, 10 e 15. Perdas como essas ocorreram algumas vezes durante os testes realizados devido a problemas como falha de comunicação e servidores *web* temporariamente fora de uso, impedindo assim o acesso ao arquivo PDF. Visto que o *fitness* é calculado com base na relação entre as palavras existentes no arquivo e as palavras que constituem o indivíduo e que todos os indivíduos devem ser avaliados, um novo indivíduo é gerado para substituir o indivíduo que não pôde ser avaliado, o que vem justificar o comportamento do algoritmo genético apresentado na Figura 3.4, bem como a não utilização do operador de mutação, pois a cada inserção de um novo indivíduo na população ocorre um aumento na diversidade da mesma.

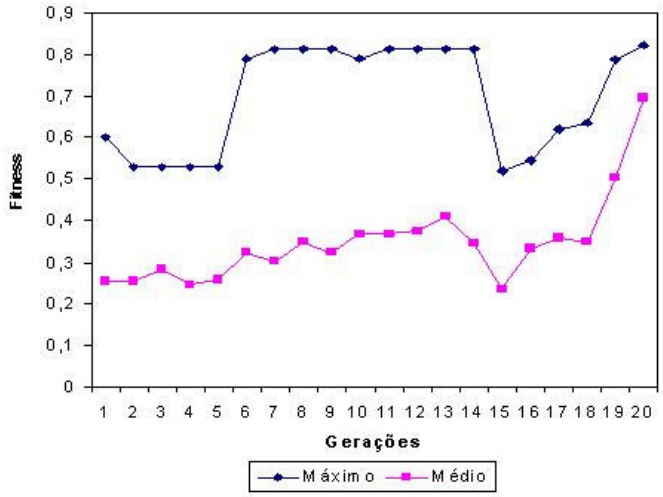



Figura 3.4: Evolução do melhor indivíduo da população (curva superior) e o *fitness* médio da população (curva inferior).

Para se ter uma idéia dos documentos sugeridos pelo algoritmo de busca, a Tabela 3.4 relaciona alguns exemplos de documentos recuperados durante os testes que foram realizados.

Tabela 3.4: Exemplos de documentos sugeridos pelo algoritmo de busca.

| | |
|---|---|
| **CE** | Conjunto de Slides de David Hales intitulado *Introduction to Genetic Algorithms*. Disponível em: http://cfpm.org/~david/talks/ga2005/ga-pres.pdf |
| **RNA** | *A Feed Forward Neural Network for Determining a User's Location* Disponível em: http://www.edgelab.ca/publications/feed_forward.pdf |
| **SF** | *Fuzzy Logic: An Interface Between Logic and Human Reasoning* Disponível em: http://www.cosy.informatik.uni-bremen.de/staff/freksa/publications/IEEE94Freksa.pdf |

As consultas realizadas para a busca dos documentos da Tabela 3.4 foram as seguintes:

- Computação Evolutiva**:** `random genetic evolutionary chromosome solutions –pareto filetype:PDF`
- Sistemas Fuzzy: `sets fuzzy logic step models –functions filetype:PDF`
- Redes Neurais Artificiais: `table weights hidden algorithm system –approach filetype:PDF`

#### 3.5.2.1 Critérios de Avaliação para Sistemas de Recuperação de Informação

Os Sistemas de Recuperação de Informação têm sido comparados e avaliados há décadas. Em meados dos anos 60, Cleverdon (1997) propôs seis quesitos que podem ser utilizados como critérios de avaliação para os sistemas de recuperação de informação:

1. *Coverage* (cobertura): a extensão na qual o sistema atua;
2. *Time lag* (tempo de resposta): o intervalo de tempo entre a requisição do usuário e a resposta do sistema;
3. *Presentation* (forma de apresentação): a apresentação final ao usuário;
4. *Effort* (esforço): esforço envolvido pelo usuário para conseguir obter a resposta desejada;
5. *Recall*: proporção dos documentos relevantes de uma coleção que foram recuperados; e

6. *Precision*: proporção relevante dos documentos recuperados na busca.

Dentre esses critérios, *precision* e *recall* são as medidas mais utilizadas para a avaliação de sistemas de recuperação de informação (Rijsbergen, 1979). Ao longo dos anos, muitos têm questionado se *precision* e *recall* são de fato critérios apropriados para serem utilizados como medida de eficácia, motivando o estudo e o desenvolvimento de alternativas (Cooper, 1968; Hersh & Molnar, 1995; Tague-Sutcliffe, 1996; Ellis, 1996; Hersh *et al*., 1996). Entretanto, apesar de suas deficiências, *precision* e *recall* continuam a ser amplamente utilizadas em parte, por não existir um consenso entre os pesquisadores sobre quais medidas alternativas possam ser superiores (Gwizdka & Chignell, 2006).

O cálculo de *recall* pode ser obtido conforme a Equação 3.8:

$$recall = \frac{A}{A+B} \tag{3.8}$$

onde $A$ representa o número de documentos relevantes recuperados pelo sistema durante a busca e $B$ representa o número de documentos relevantes existentes na base e que não foram recuperados pelo sistema. Dessa forma $A+B$ representa o número total de documentos relevantes existentes na base de dados. A Equação 3.9 apresenta o cálculo utilizado para *precision*:

$$precision = \frac{A}{A+C} \tag{3.9}$$

onde $A$ representa o número de documentos relevantes recuperados pelo sistema durante a busca e $C$ representa o número de documentos irrelevantes para essa busca e que foram recuperados pelo sistema.

O desenvolvimento de sistemas de recuperação de informação (RI), cuja área de atuação (*coverage*) é a *web*, é muito diferente dos sistemas de RI tradicionais que atuam em bases de dados indexadas. A *web* possui características específicas como dinamismo, natureza hipertextual, ausência de vocabulário controlado e indexado, heterogeneidade de tipos de documentos, liberdade de estilos e facilidade de acesso a usuários de diversos tipos.

Dos critérios de avaliação propostos por Cleverdon, os itens de 1 a 4 não foram analisados, visto que: a área de atuação (1 – *coverage*) é definida (*web*); o tempo de resposta (2 – *time lag*) não é relevante, pois as buscas são autônomas, ou seja, o usuário não necessita ficar esperando por uma resposta do sistema; a apresentação (3 –*presentation*) envolve sempre documentos no formato PDF e não existe a interação do usuário durante o processo de busca, não existindo dessa forma o *effort* (4).

Devido à natureza dinâmica da *web* e a quantidade imensa de informação disponível, tornou-se inviável também a adoção de *recall*, pois conforme observado na Equação 3.8 seria necessário ter o conhecimento *a priori* do número total de documentos relevantes existentes na *web* (Gwizdka & Chienell, 2006; Chu & Rosenthal, 1996; Shafi & Rather, 2005).

A eficácia do processo de busca de informação na *web* foi avaliada tendo *precision* (ou precisão) como principal medida de avaliação. A precisão foi avaliada para 32 documentos recuperados para a área de *Computação Evolutiva* (CE), 26 documentos recuperados para a área de *Redes Neurais Artificiais* (RNA) e 22 documentos para a área de *Sistemas Fuzzy* (SF). Para determinar a relevância dos documentos recuperados pelo agente de busca, foi feita uma leitura nos resumos dos documentos (para os documentos que não possuíam resumo, uma leitura superficial foi feita), no intuito de analisar a relevância do conteúdo. Os resultados obtidos são apresentados na Tabela 3.5. No Anexo II são apresentados todos os documentos recuperados pelas três áreas de busca.

Tabela 3.5: Eficácia do sistema de busca.

| Nro de docs recuperados | Área de Busca | *Precision* |
|---|---|---|
| 32 | CE | 81,25 |
| 26 | RNA | 84,62 |
| 22 | SF | 95,45 |
| Média | | 87,11 |

Conforme pode ser observado na tabela, os resultados atingiram um percentual acima de 81% nas três áreas de busca, destacando-se a área de Sistemas Fuzzy (SF) com uma precisão de 95,45%. Analisando os documentos recuperados, pôde-se constatar que os valores inferiores obtidos pelas áreas de Computação Evolutiva (CE) e Redes Neurais Artificiais ocorreram devido à própria inspiração biológica dessas áreas, nas quais palavras-chave, tais como neurônios, sinapse e neural, levaram a recuperação de documentos tanto da área de RNAs, quanto da área de Neurociências. Da mesma forma, palavras-chave tais como cromossomo, gene, alelo, *crossover* e mutação recuperaram documentos da área de CE e da área de Biologia e Genética.

## 3.6. Sumário do Capítulo

Este capítulo apresentou um sistema evolutivo para automatização da busca na *web*. O sistema é composto por dois algoritmos: um deles responsável pela extração de palavras-chave dos documentos e a classificação destas num dicionário de termos; e o outro, um algoritmo genético que faz uso das palavras existentes no dicionário de termos, tem o objetivo de buscar novos documentos na *web* que sejam relevantes para o usuário. Como base de testes para o algoritmo extrator de termos foram escolhidas três áreas de interesse: 1) computação evolutiva; 2) sistemas nebulosos; e 3) redes neurais artificiais. O conjunto de documentos de cada área foi copiado com amostragem aleatória dos Anais do IEEE WCCI 2002 (*World Congress on Computational Intelligence* 2002). O algoritmo evolutivo de buscas utilizou o dicionário de termos construído pelo algoritmo extrator de termos como um perfil de usuário para efetuar buscas na *web*. Os resultados mostraram que o sistema foi capaz de criar um dicionário representativo e parcimonioso de termos, com um número reduzido de palavras capaz de representar uma determinada área de interesse.

Diferente das abordagens tradicionais, o algoritmo genético implementado trabalha com duas populações de indivíduos que co-evoluem. Na população denominada 'população de termos', cada cromossomo é constituído por um conjunto de termos extraídos de forma

aleatória do dicionário de termos. A segunda população denominada a 'população de operadores lógicos' contém o mesmo número de genes dos cromossomos da população de termos, assim como o mesmo número de indivíduos. Cada gene da segunda população pode assumir um dos três operadores lógicos E, OU e NÃO, escolhidos aleatoriamente. Após a inicialização das duas populações e após cada etapa de crossover, uma função *merge* concatena os genes das duas populações produzindo uma consulta que é submetida à máquina de busca do Google. O *fitness* de cada indivíduo é calculado através da medida do coseno entre a expressão de busca (consulta) e o documento recuperado. Apesar do espaço de busca (ambiente *web*) ser altamente dinâmico e em alguns momentos instável, o algoritmo genético mostrou-se robusto apresentando resultados satisfatórios (Tabela 3.5). A medida de avaliação utilizada para analisar a o desempenho do sistema evolutivo de busca foi a proporção de documentos relevantes recuperados pelo sistema, conhecida como *precision*. O sistema foi avaliado na recuperação de documentos para três áreas distintas, apresentando um desempenho médio de 87%. Acredita-se que esse percentual poderia ser ainda maior caso as áreas envolvidas na busca não fossem tão correlacionadas.

# 4. A$^2$CA: Algoritmo Adaptativo de Agrupamento Inspirado em Formigas

## 4.1. Introdução

As formigas são um dos principais insetos estudados por várias áreas de pesquisa devido a seu comportamento social. Alguns pesquisadores têm documentado como muitas espécies de formigas trabalham de forma colaborativa na tarefa de organização e limpeza do ninho. Esta tarefa envolve, dentre outras coisas, o agrupamento de corpos, ou partes de corpos, de formigas mortas e o agrupamento de lixo, sempre com o objetivo de manter o ninho organizado e limpo (Chrétien, 1996; Deneubourg *et al.*, 1991). Ao encontrar estes *objetos* (corpos e/ou lixo) dispersos de forma aleatória pelo ninho, as formigas operárias começam o processo de limpeza e organização do ninho, agrupando os objetos em áreas comuns, formando pilhas (agrupamentos) de objetos nessas áreas. Independentemente da área de aplicação, em termos gerais, a idéia é que objetos que estejam isolados no ambiente sejam pegos e depositados em outras regiões que contenham objetos do mesmo tipo.

O algoritmo padrão de agrupamento baseado em formigas (*Standard Ant Clustering Algorithm* – SACA) foi brevemente apresentado no Capítulo 2. Neste algoritmo, as formigas se movimentam sobre uma grade (grid) bi-dimensional toroidal de dimensão $m \times m$. O movimento de uma formiga é caracterizado pelo seu deslocamento em uma unidade do grid, conhecida como *célula*, em qualquer direção adjacente a sua posição atual. O algoritmo começa distribuindo todos os objetos de forma aleatória no grid toroidal. O mesmo procedimento é utilizado com as formigas. Após a distribuição dos objetos e formigas sobre o grid, dá-se início a um processo iterativo, no qual é checado o estado de cada formiga, ou seja, se uma formiga está ou não carregando um objeto e se a posição no grid que ela ocupa contém um objeto. A probabilidade de pegar esse objeto aumenta de maneira inversamente proporcional à densidade de objetos na região e diminui proporcionalmente à similaridade desse objeto em relação aos objetos dispostos no raio de visão da formiga. O raio de visão da formiga é definido como um quadrado de $s \times s$ unidades do grid.

Nesse capítulo será proposta uma variação do algoritmo SACA, denominado $A^2CA$ (*Adaptive Ant Clustering Algorithm*). Na Seção 4.2 é feita uma breve revisão bibliográfica, em ordem cronológica, da pesquisa em métodos de agrupamento baseados no comportamento de formigas. Além do resfriamento gradativo no parâmetro que regula a probabilidade de uma formiga pegar um objeto do grid, as Seções 4.5 e 4.6 detalham as duas principais modificações que diferem o $A^2CA$ do SACA, são elas: 1) um esquema de visão adaptativa; e 2) a distribuição de feromônio pelas células do grid. O capítulo termina fazendo uma análise comparativa de desempenho entre o $A^2CA$ e o SACA utilizando um conjunto de três bases de dados sintéticas e duas bases de dados reais.

## 4.2. Algoritmos de Agrupamento Inspirados em Formigas

Em 1991, Deneubourg *et al.* (1991) apresentou um modelo no qual formigas simples eram capazes de organizar pilhas de objetos que inicialmente estavam distribuídas aleatoriamente sobre um plano. Essas formigas possuem uma espécie de comportamento baseado num conjunto de regras locais. Portanto, a capacidade de percepção de uma formiga restringe-se ao local (vizinhança) onde a mesma se encontra. Gutowitz (1993) chamou esses agentes de formigas básicas, as quais possuem: 1) uma memória limitada que atua como um registrador de tamanho *n* para gravar a presença ou ausência de objetos nos *n* locais anteriores por onde a formiga tenha passado; 2) uma capacidade de manipulação de objetos; 3) uma função que dê a probabilidade de manipular um objeto proporcionalmente aos valores armazenados na memória e uma variável aleatória; e 4) a capacidade de executar movimento Browniano. Além disso, como observado anteriormente no modelo de Deneubourg, dois objetos podem somente ser idênticos ou diferentes. Obviamente, essa mesma idéia pode ser facilmente estendida para atuar com outras métricas de distância como, por exemplo, a distância Euclidiana.

Embora as formigas básicas possuam somente a capacidade de percepção local, elas são capazes de trabalhar de maneira organizada em nível global. O mecanismo que dá sustentação para este fenômeno foi cuidadosamente investigado por Gutowitz (1993). Ele

propôs um novo tipo de formiga capaz de analisar a complexidade do ambiente a sua volta procurando exercer suas atividades nas vizinhanças onde existe maior complexidade. A complexidade é medida pelo número de lados que separam as células de diferentes tipos, contendo ou não objetos. Dessa forma, vizinhanças totalmente vazias ou totalmente ocupadas possuem complexidade igual a zero (baixa entropia), ao passo que um grid contrastando itens das duas classes possui complexidade máxima. Essas formigas mais especializadas podem executar suas tarefas de maneira mais eficiente do que as formigas básicas, principalmente porque elas tendem a manipular objetos em regiões de alta complexidade, isto é, regiões com densidade intermediária, onde a entropia é alta.

Conforme abordado no Capítulo 2, Lumer & Faieta (1994) apresentaram um método capaz de estruturar conjuntos de dados complexos dentro de grupos (*clusters*). O método utilizado foi inspirado no modelo de Deneubourg *et al.* (1991), no qual formigas movimentam-se randomicamente num grid bidimensional, onde os objetos estão inicialmente espalhados aleatoriamente. Inspirados pelo fenômeno biológico de agrupamento de objetos, as formigas não se comunicam entre si e somente percebem o ambiente local em sua vizinhança. Nesse contexto, cada formiga limita-se a pegar um objeto do grid ou largar um objeto sobre o grid. A probabilidade de pegar um objeto diminui de acordo com a densidade de outros objetos na sua vizinhança e a similaridade entre objetos nesta vizinhança. Já a probabilidade de largar um objeto aumenta com a similaridade e a densidade de objetos na vizinhança. Embora o trabalho de Deneubourg *et al.* (1991) estivesse restrito a objetos idênticos ou a dois tipos distintos de objetos, Lumer e Faieta (1994) generalizaram esse modelo para trabalhar com objetos que se diferenciassem em vários grupos, tomando por base uma medida de similaridade entre os objetos. Dessa generalização surgiu o *Algoritmo Padrão de Agrupamento Baseado em Formigas* (*Standard Ant Clustering Algorithm*), chamado nesse trabalho de SACA.

Monmarché *et al.* (1999) combinou os princípios estocásticos e exploratórios do SACA com os princípios determinísticos e heurísticos do popular algoritmo *k*-means no intuito de melhorar a convergência do SACA. O método híbrido proposto é chamado AntClass e foi baseado no trabalho de Lumer & Faieta (1994). O algoritmo AntClass

permite uma formiga depositar mais de um objeto numa mesma célula, formando pilhas de objetos. Esse método é constituído por quatro passos principais: 1) agrupamento baseado em formigas; 2) o emprego do algoritmo *k*-means na partição inicial disponibilizada pelas formigas; 3) agrupamento em pilhas de objetos; e 4) a utilização do algoritmo *k*-means mais uma vez para refinar os grupos determinados. Uma outra contribuição importante do algoritmo AntClass é que ele também faz uso de agrupamento hierárquico, permitindo as formigas carregarem pilhas inteiras de objetos.

Ramos & Merelo (2002) desenvolveram um algoritmo de agrupamento baseado em formigas chamado ACLUSTER, o qual foi utilizado no agrupamento de documentos no formato de arquivos texto. Os autores propuseram o uso de probabilidades de transição espacial inspiradas biologicamente, evitando que as formigas movimentem-se aleatoriamente e explorem regiões de pouco interesse. Diferente do SACA onde o movimento da formiga é aleatório, a movimentação de um agente no ACLUSTER é feita tomando por base em probabilidades de transição que dependem da distribuição espacial de feromônio pelo ambiente. Se um cluster em particular desaparece do grid, o feromônio tende a evaporar daquela região. Esta abordagem é interessante, pois o feromônio representa uma espécie de memória coletiva e todas as formigas podem se beneficiar desta memória. O ACLUSTER também foi empregado no problema de recuperação de imagens digitais. Mais detalhes sobre um estudo de caso utilizando uma base de dados para a identificação de granito podem ser encontrados em (Ramos *et al.*, 2002). Num trabalho posterior, Abraham e Ramos (2003) aplicaram o ACLUSTER para descobrir padrões de comportamento de visitantes de páginas *web*.

Handl & Meyer (2002) utilizaram um algoritmo de agrupamento baseado em formigas como parte principal de um sistema de filtragem de informação de documentos disponíveis na *web*. O objetivo básico é a classificação on-line de documentos baseada na similaridade de seus conteúdos. Os autores dotaram as formigas com memória de curto prazo e empregaram diferentes velocidades, permitindo inclusive que elas saltem para regiões distantes do grid. Os autores também adotaram um controle de estagnação baseado no trabalho de Monmarché *et al.* (1999), no qual depois de um número pré-definido de

tentativas sem sucesso de largar um objeto, a formiga o deposita no grid independentemente da vizinhança existente.

Labroche *et al.* (2002) propuseram um algoritmo de agrupamento chamado ANTCLUST, baseado em um sistema de reconhecimento químico das formigas. O sistema permite que as formigas que pertençam a um mesmo ninho se reconheçam mutuamente através de um odor característico. Dessa forma, elas conseguem diferenciar quais as formigas que fazem parte do ninho de eventuais intrusos. No ANTCLUST, cada objeto é assinalado como sendo uma formiga artificial possuindo um determinado odor. Durante a execução do algoritmo acontecem encontros aleatórios entre formigas e elas vão sendo rotuladas (pertencentes a um determinado grupo) de acordo com algumas regras que levam em consideração a similaridade entre os dados. O procedimento de rotulação evolui com o passar do tempo até que cada formiga tenha encontrado seu grupo.

Kanade & Hall (2003) combinaram o algoritmo proposto por Monmarché *et al.* (1999) com o algoritmo *Fuzzy C-Means* (FCM) (Bezdek,1981). O algoritmo baseado em formigas é empregado inicialmente no processo de criação dos primeiros grupos, os quais são então redefinidos pelo algoritmo FCM. Dessa forma, os centróides correspondentes a cada grupo inicial tornam-se os protótipos para o FCM.

## 4.3. $A^2CA$: Algoritmo Adaptativo de Agrupamento Baseado em Formigas

O algoritmo $A^2CA$ (*Adaptive Ant Clustering Algorithm*) foi desenvolvido tendo em mente o uso de feromônio durante o processo de construção e organização dos ninhos em formigas e cupins (Vizine *et al*., 2005a). Em particular, o $A^2CA$ foi inspirado no comportamento dos cupins durante o processo de construção do ninho, onde cada cupim deposita feromônio sobre partículas de terra recrutando outros cupins a fazerem o mesmo em regiões nas quais a concentração do feromônio seja maior. Na medida em que essas partículas de terra vão se juntando, a quantidade de feromônio acumulada na região aumenta, atraindo mais cupins (Camazine *et al.*, 2001). Dessa forma, eles constroem o ninho de maneira colaborativa, organizada e sem que haja hierarquia de comando. Uma

outra observação biológica incorporada no $A^2CA$ foi o fato de que a visão das formigas não se restringe à vizinhança local onde elas estão atuando, tendo a capacidade de percepção de uma área maior, que pode variar de formiga para formiga com o passar do tempo. Dessa maneira, o $A^2CA$ incorpora duas modificações importantes que o diferem do SACA: 1) um esquema de visão adaptativa; e 2) a inclusão de feromônio sobre as células do grid. Adicionalmente, foi adotado um resfriamento gradativo no parâmetro $k_p$ que regula a probabilidade de pegar um item do grid.

### 4.3.1. Resfriamento Gradativo de $k_p$

Além das modificações que motivaram o desenvolvimento do $A^2CA$, foi adotada também uma pequena alteração proposta para o algoritmo SACA (Vizine *et al.*, 2005). Esta alteração tem o objetivo de minimizar o problema da instabilidade do algoritmo em manter os grupos formados passados um certo número de iterações. O procedimento é simples: após um ciclo de execução, que corresponde a 10.000 passos de formiga, ter sido executado, o valor do parâmetro $k_p$ começa a ser geometricamente reduzido a cada ciclo, até que seja atingido um valor mínimo $k_{p\min}$, que corresponde a um dos critérios de parada do algoritmo. No $A^2CA$, o parâmetro $k_p$ é atualizado com base na Equação 4.1. É importante salientar que o SACA implementado nesse capítulo, utilizado para comparação de resultados com o $A^2CA$, também incorpora o resfriamento gradativo de $k_p$, tornando a análise de desempenho de ambos mais justa e focada principalmente nas modificações propostas para o SACA.

$$\begin{aligned} k_p &\leftarrow k_p \times 0.98, \\ k_{p\min} &= 0.001. \end{aligned} \tag{4.1}$$

### 4.3.2. Visão Adaptativa

No SACA a probabilidade de uma formiga pegar um item no grid aumenta conforme a baixa densidade de itens na vizinhança e diminui dependendo da similaridade entre itens presentes na área de percepção da formiga. Por outro lado, a probabilidade de uma formiga carregando um item depositá-lo em alguma posição $i$ aumenta quando altas densidades de itens similares são detectadas na área de percepção da formiga. A função utilizada para o cálculo de densidade é definida pela seguinte expressão:

$$f(i) = \begin{cases} \frac{1}{s^2} \sum_j (1 - d(i,j)/\alpha) & \text{se } f(i) > 0 \\ 0 & \text{caso contrário.} \end{cases}, \tag{4.2}$$

onde $s^2$ é o número de células na vizinhança de $i$, $d(i,j)$ é a distância Euclidiana entre os objetos $i$ e $j$ num espaço $N$-dimensional e $\alpha$ é uma constante que regula a dissimilaridade entre itens. O valor máximo para $f(i)$ é obtido se, e somente se, todas as células que são percebidas no raio de visão da formiga contiverem itens idênticos.

A definição de um valor fixo para $s$ pode, algumas vezes, causar comportamentos inapropriados, pois um tamanho fixo do campo de visão não permite a formiga fazer uma distinção entre grupos de diferentes tamanhos. Um campo de visão pequeno implica numa percepção reduzida do grupo em relação ao *grid* como um todo. Dessa forma, pequenos grupos e grandes grupos passam a ser percebidos como sendo de mesmo tamanho, acarretando sérios problemas, pois todo o processo de tomada de decisão de uma formiga em pegar ou largar um item está condicionado ao seu campo de visão. A Figura 4.1 ilustra esse problema, no qual uma formiga, cujo campo de visão está delimitado pela cor cinza, só consegue perceber dois *grupos* de mesmo tamanho dentro do *grid*, o que não corresponde à realidade. Entretanto, a adoção de um campo de visão mais amplo pode não ser conveniente nas primeiras iterações do algoritmo, onde os elementos a serem agrupados estão dispersos pelo *grid* de forma aleatória. Neste caso, uma visão ampla do *grid* pode dar a falsa impressão de que existe um grande número de pequenos grupos formados.

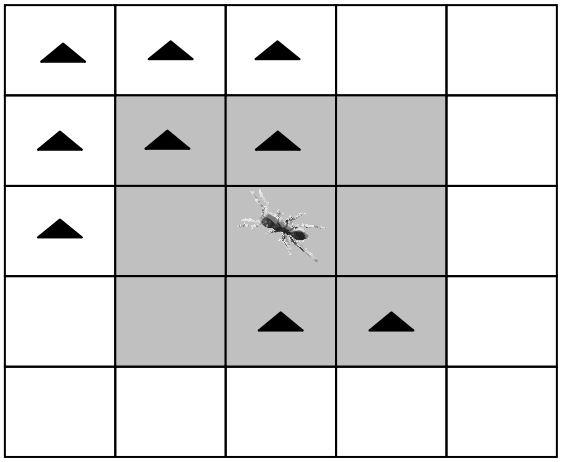


Figura 4.1: Campo de visão da formiga demarcado pela área cinza.

No intuito de minimizar esse problema, um esquema de visão adaptativa foi proposto para o SACA por Sherafat *et al.* (2004a). Quando uma formiga percebe um grupo 'grande', ela incrementa o seu campo de visão ($s_i^2$) até um tamanho máximo. Nessa proposta, $s_i^2$ é um parâmetro específico para cada formiga, podendo ser inicializado com o mesmo valor para todas. Durante a execução do algoritmo este valor vai sendo atualizado de forma dinâmica e independente. Essa modificação resolve parcialmente o problema, mas uma questão ainda permanece: "Como fazer uma formiga estimar o tamanho de um grupo sendo orientada apenas pelo tamanho de seu campo de visão?".

Procurando responder essa questão, o $A^2CA$ utiliza a função de densidade $f(i)$ como um parâmetro de controle. Existe uma relação entre o tamanho de um grupo e sua densidade: o valor médio de $f(i)$ aumenta durante o processo de agrupamento e isto acontece porque o número de grupos formados tende a aumentar. Quando $f(i)$ atinge um valor maior do que um limiar $\lambda$ pré-especificado, o parâmetro $s$ é incrementado de $n_s$ unidades até atingir seu valor máximo (Equação 4.3).

$$\begin{aligned} &\text{se } f(i) > \lambda \quad \text{e} \quad s \leq s_{\max}, \\ &\qquad \text{então } s \leftarrow s + n_s. \end{aligned} \tag{4.3}$$

onde os valores $s^2_{\max} = 7 \times 7$, $\lambda = 0.6$ e $n_s = 1$ foram utilizados durante os testes realizados com o $A^2CA$ nesta dissertação. Estes valores foram escolhidos com base em alguns experimentos preliminares.

### 4.3.3. Heurísticas de Feromônio

O processo de agrupamento do SACA consiste em analisar a distância relativa entre todos os itens presentes no campo de visão de uma formiga. O problema encontrado nessa abordagem é que ela é local, ou seja, ela não leva em consideração o processo de agrupamento em nível global. Uma forma de amenizar essa limitação foi apresentada por Sherafat *et al.* (2004a,b). A proposta baseia-se na introdução de uma variável local $\phi(i)$, associada a cada célula $i$ do grid, que irá representar a presença ou ausência de feromônio (indiretamente objetos) na célula $i$. Utilizando a inspiração biológica de como os cupins

usam o feromônio para orientar a construção dos seus ninhos, as formigas artificiais que atuam no $A^2CA$ irão adicionar uma quantidade de feromônio nos itens que elas carregam e este feromônio será transferido para a célula do grid quando um item é depositado. A cada iteração, esse feromônio artificial $\phi(i)$ de cada posição $i$ do grid evapora a uma taxa fixa.

Sherafat *et al.* (2004a,b) introduziram uma função de feromônio $Phe(\cdot)$, representada pela Equação 4.4, que influencia a probabilidade de pegar e largar itens do grid. A função de feromônio proposta varia linearmente conforme o nível de feromônio $\phi(i)$ em cada posição do grid e depende de alguns parâmetros definidos pelo usuário, como $\phi_{max}$ e $\phi_{min}$ (valores do feromônio percebidos pela formiga) e a influência máxima de feromônio permitida, $P$.

$$Phe(\phi_{\min}, \phi_{\max}, P, \phi(i)) = \frac{2.P}{\phi_{\max} - \phi_{\min}} \phi(i) - \frac{2.P.\phi_{\max}}{\phi_{\max} - \phi_{\min}} + P, \tag{4.4}$$

Para acomodar a adição do feromônio no grid, algumas variações nas funções de probabilidade de pegar e largar itens do SACA foram também propostas e estão representadas nas Equações (4.5) e (4.6), respectivamente:

$$P_{pick}(i) = (1 - Phe(\phi_{\min}, \phi_{\max}, P, \phi(i))) \times \left( \frac{k_p}{k_p + f(i)} \right)^2. \tag{4.5}$$

$$P_{drop}(i) = (1 + Phe(\phi_{\min}, \phi_{\max}, P, \phi(i))) \times \left( \frac{f(i)}{k_d + f(i)} \right)^2. \tag{4.6}$$

onde *max* representa a maior quantidade de feromônio percebida pelo agente num dado instante; *min* corresponde a menor quantidade de feromônio percebida pelo agente no mesmo instante da percepção de *max*; $P$ é a influência máxima do feromônio nas probabilidades de pegar e largar objetos; e $\phi(i)$ é a quantidade de feromônio na posição corrente $i$.

É importante ressaltar que, na Equação (4.6), a probabilidade de largar um item foi originariamente derivada do modelo proposto por Deneubourg *et al.* (1991). Tal escolha deve-se ao fato de que o algoritmo apresentou desempenho superior usando a Equação

(4.6). Dessa forma, tanto o $A^2CA$ quanto o SACA implementado para os testes realizados adotam a Equação (4.7), a qual é uma função inversa do parâmetro $k_d$:

$$P_{drop}(i) = \left( \frac{f(i)}{k_d + f(i)} \right)^2 . \tag{4.7}$$

Baseado na análise de sensibilidade descrita em Sherafat *et al.* (2004a,b) e em alguns experimentos preliminares, percebeu-se que a configuração dos parâmetros *max*, *min* e *P* pode ser uma tarefa difícil e trabalhosa. No intuito de facilitar o uso do algoritmo, reduzindo o número de parâmetros definidos pelo usuário, as Equações (4.5) e (4.6) foram substituídas pelas Equações (4.8) e (4.9), respectivamente:

$$P_{pick}(i) = \frac{1}{f(i)\phi(i)} \left( \frac{k_p}{k_p + f(i)} \right)^2 . \tag{4.8}$$

$$P_{drop}(i) = f(i)\phi(i) \left( \frac{f(i)}{k_d + f(i)} \right)^2 . \tag{4.9}$$

onde $f(i)$ é a função de densidade, $\phi(i)$ é a quantidade de feromônio na posição $i$ e $k_p$ e $k_d$ são as constantes das probabilidades de pegar e largar, respectivamente. Essa proposta apresenta uma nova variável em relação ao SACA, que é o nível de feromônio associado a cada posição do grid.

De acordo com a Equação 4.8, a probabilidade de uma formiga pegar um item do *grid* é inversamente proporcional à quantidade do feromônio existente na posição $i$ onde o objeto encontra-se como também à densidade de objetos existentes ao redor de $i$. Essa equação leva em consideração a distribuição de feromônio em regiões do grid nas quais existam objetos similares. No entanto, se a região está preenchida com objetos distintos, a incorporação de $f(i)$ multiplicando $\phi(i)$ contrabalanceia um eventual aumento na concentração de feromônio. Da mesma forma, a Equação 4.9 prioriza regiões com alta concentração de feromônio para o depósito de objetos de mesmo tipo.

É importante ressaltar que uma região com uma quantidade alta de feromônio tende a ser não só um grande grupo, como também um grupo em fase de construção. Isso ocorre porque o feromônio evapora com o passar do tempo. Cada posição *i* do grid tem uma variável independente $\phi(i)$ correspondente à quantidade de feromônio na qual procedimentos de *evaporação* e *difusão* são implementados. A taxa pela qual o feromônio evapora é pré-configurada, de acordo com a Equação 4.10. Cada posição *i* do grid também tem uma conexão com seus vizinhos, possibilitando que um percentual de $\phi(i)$ seja difundido entre eles. Isto é executado de tal maneira que a porcentagem do feromônio para os dois vizinhos mais próximos em todas as direções decai geometricamente na razão de 1/2, ao passo que para os demais é configurada igual a zero. Na implementação do $A^2CA$ a quantidade máxima de feromônio a ser adicionada é igual a 0.01. Essa abordagem aumenta a probabilidade de desarranjar clusters pequenos e aumenta a probabilidade de depósito de itens de dados em clusters maiores, o que torna essa proposta uma espécie de procedimento de clusterização baseado em densidade (Everitt *et al*., 2001).

$$\phi(i) \leftarrow \phi(i) \times 0.99. \tag{4.10}$$

## 4.4. Avaliação de Desempenho

No intuito de avaliar o desempenho do algoritmo $A^2CA$ em relação ao algoritmo tradicional SACA, o qual foi modificado para incorporar o resfriamento gradativo e uma nova função de probabilidade de largar itens dada pela Equação 4.7, ambos os algoritmos foram submetidos a três conjuntos de dados sintéticos de teste e dois problemas de agrupamento de dados reais, um envolvendo dados de bioinformática e outro considerando dados textuais, mais especificamente artigos científicos digitalizados no formato PDF (*Portable Document Format*). Os parâmetros utilizados para a execução dos algoritmos foram baseados na análise de sensibilidade feita por Sherafat *et al.* (2004a) e em alguns experimentos preliminares realizados para esta dissertação. As bases de teste utilizadas e os respectivos parâmetros de adaptação para os algoritmos estão resumidos abaixo e mais detalhes serão apresentados nas seções seguintes:

- Base de dados sintética **4Gauss**: 100 objetos divididos em 4 classes (*grupos*). $\theta = 0.6$, $k_p = 0.20$, $k_d = 0.05$, $n_{ants} = 10$, grid = 25×25, e $\alpha = 0.35$.

- Base de dados sintética **Animais**: 16 objetos com 13 atributos (sem rótulos de classes definidos previamente). $\theta = 0.6$, $k_p = 0.20$, $k_d = 0.05$, $n_{ants} = 1$, grid = 15×15, and $\alpha = 2.10$.
- Base de dados sintética **Ruspini**: 75 objetos divididos em 4 classes. $\theta = 0.6$, $k_p = 0.20$, $k_d = 0.05$, $n_{ants} = 10$, grid = 25×25, and $\alpha = 0.35$.
- Base de dados de bioinformática **Yeast galactose**: 205 objetos divididos em 4 classes. $\theta = 0.6$, $k_p = 0.20$, $k_d = 0.05$, $n_{ants} = 10$, grid = 35×35, and $\alpha = 1.05$.

É importante ressaltar que os parâmetros utilizados na execução dos algoritmos são praticamente os mesmos para todas as bases de dados, sendo modificados apenas os valores de $\alpha$, o tamanho do grid e o número de formigas $n_{ants}$. O tamanho do grid varia de acordo com o número de objetos a serem agrupados, ou seja, quanto maior o número de objetos, maior será o tamanho do grid. O parâmetro $\alpha$ foi o que sofreu maior variação, pois ele é responsável por regular a medida de distância entre objetos para a determinação dos grupos. Na base de dados Animais uma única formiga foi utilizada devido ao baixo número de objetos a serem agrupados.

### 4.4.1. Quatro Distribuições Gaussianas

A primeira base de dados, 4Gauss, usada para ilustrar o desempenho do algoritmo foi uma versão modificada do conjunto de dados de quatro classes proposto por Lumer & Faieta (1994) para analisar o comportamento do SACA. O conjunto de dados usado nesse experimento corresponde a quatro distribuições de 25 pontos cada, definidas por funções densidade de probabilidade Gaussianas com várias médias $\mu$ e desvio padrão fixo $\sigma = 1.5$, $G(\mu,\sigma)$, como mostra a Figura 4.2:

$A = [x \propto G(0,1.5), y \propto G(0,1.5)]$;
$B = [x \propto G(0,1.5), y \propto G(8,1.5)]$;
$C = [x \propto G(8,1.5), y \propto G(0,1.5)]$;
$D = [x \propto G(8,1.5), y \propto G(8,1.5)]$.

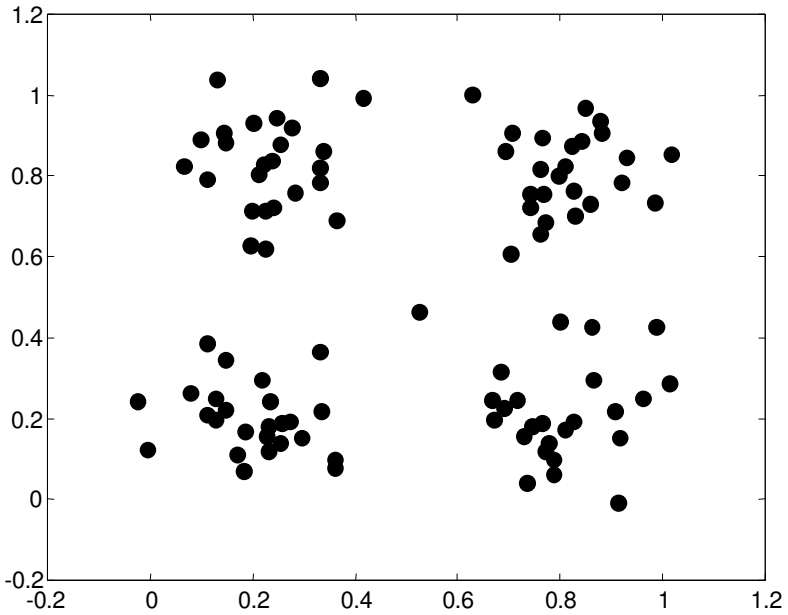
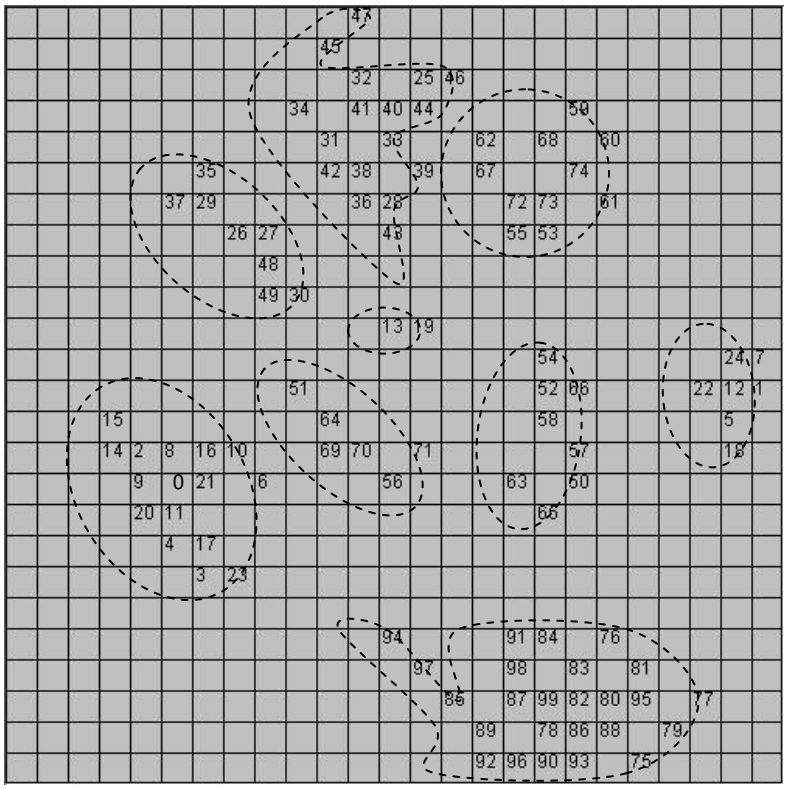


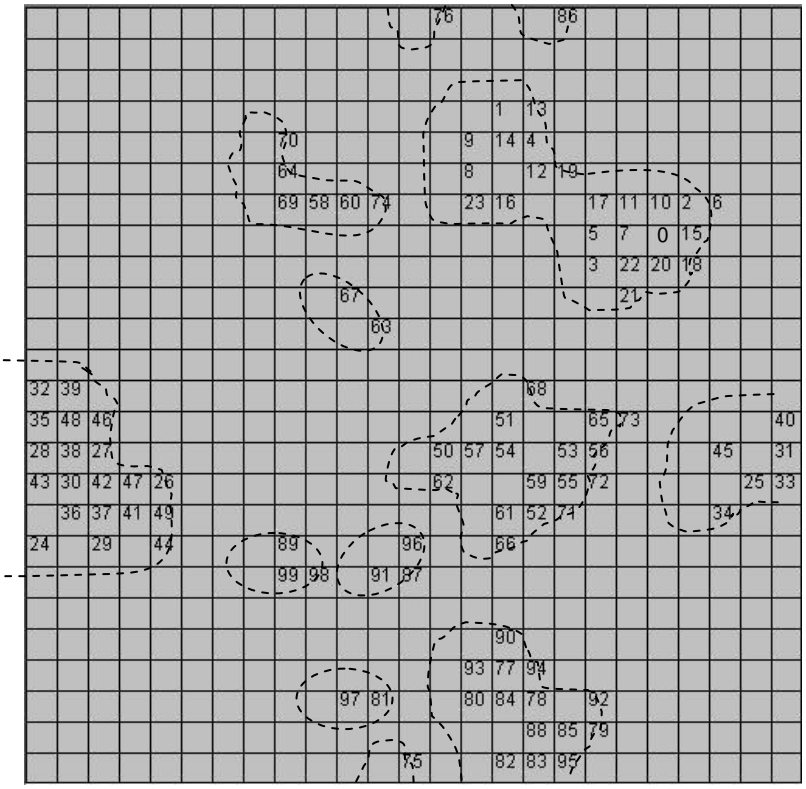
Figura 4.2: Conjunto de dados de quatro distribuições Gaussianas.

A Figura 4.3(a) mostra alguns dos resultados obtidos durante as simulações realizadas com o SACA implementado com o resfriamento gradativo de $k_p$ descrito anteriormente. Os grids apresentados correspondem ao estado final dos objetos depositados no grid após 273.000 passos de formiga (27,3 ciclos de execução). Cada objeto foi numerado de 0 a 99, sendo que os primeiros 25 (de 0 à 24) pertencem ao primeiro grupo e assim sucessivamente. É importante salientar que esta quantidade de iterações é definida automaticamente pelo critério de parada $k_{p\min} = 0.001$.

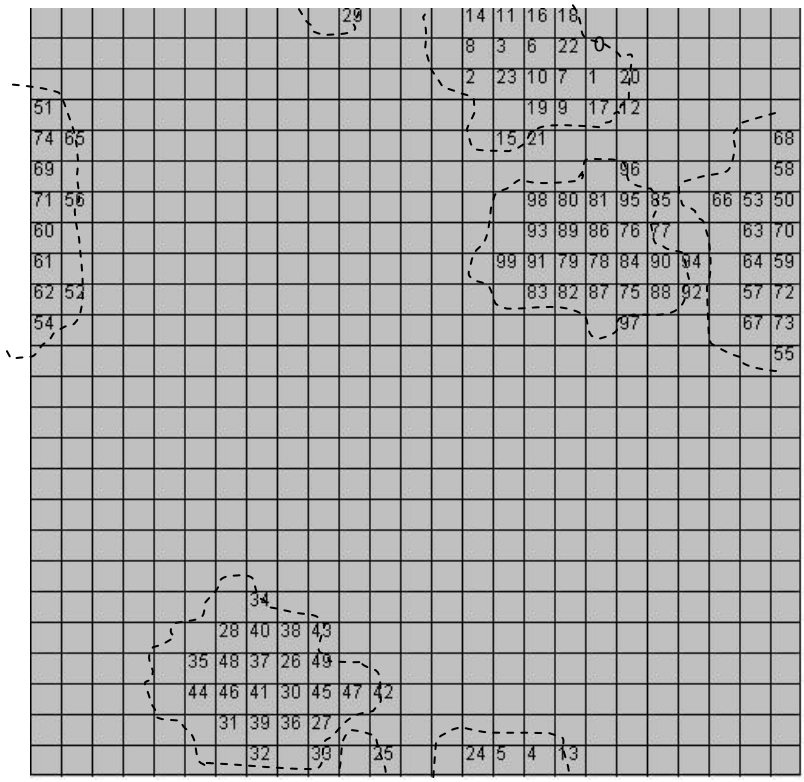
Como pode ser observado, o algoritmo SACA é capaz de agrupar os objetos corretamente, apesar de gerar um número de clusters bem maior do que os quatro grupos esperados. O comportamento do algoritmo $A^2CA$ pode ser observado na Figura 4.3(b), na qual percebe-se que o algoritmo adaptativo gerou um número muito menor de subgrupos; na maioria dos casos, somente quatro ou cinco grupos de objetos foram determinados.



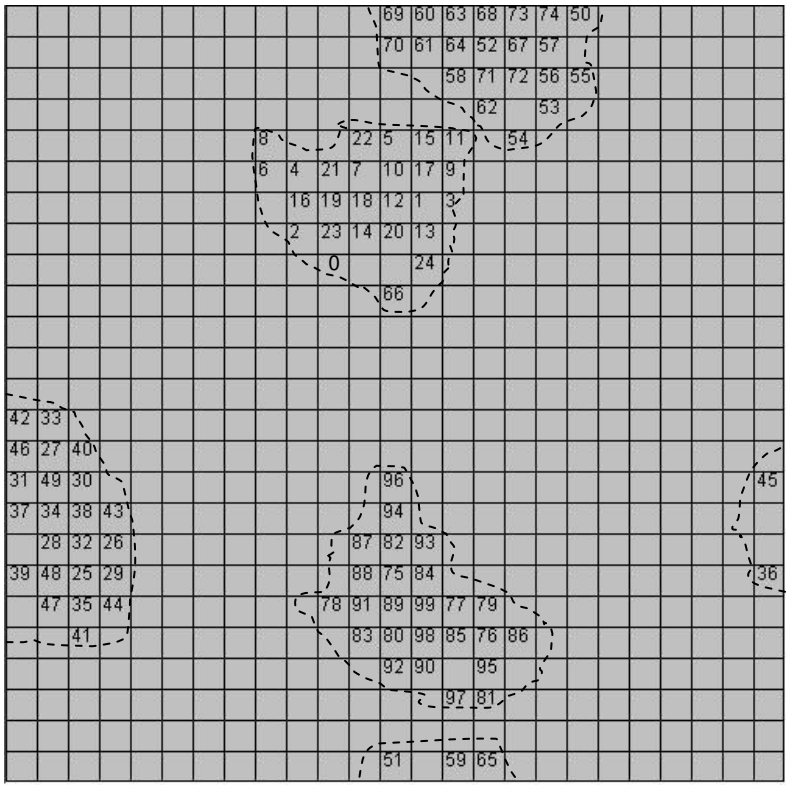
(a-1)



(a-2)



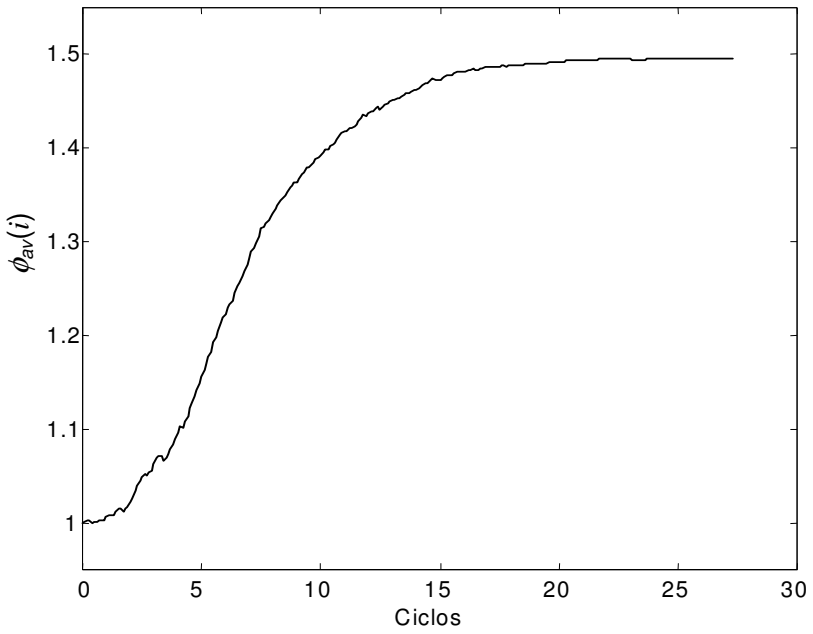
(b-1)



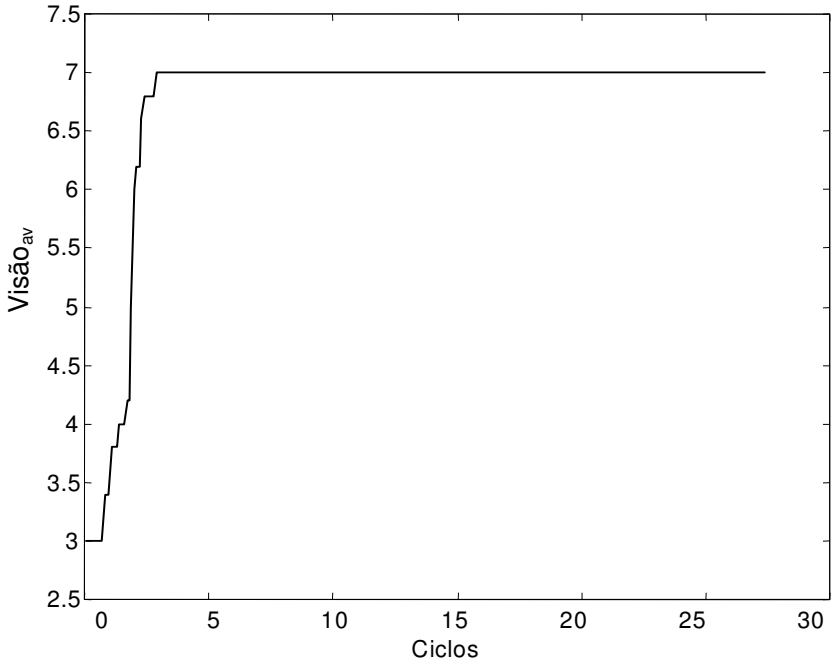
(b-2)

Figura 4.3: Dois diferentes resultados obtidos com o SACA (a) e o $A^2CA$ (b).

As Figuras 4.4(a) e (b) mostram, respectivamente, o comportamento do nível de feromônio médio sobre o grid e a visão média de todas as formigas durante as simulações apresentadas na Figura 4.3(b-1). A Figura 4.5(a) reproduz a Figura 4.3(b-1) com os grupos identificados. Uma visão tridimensional (Figura 4.5(b)) e bidimensional (Figura 4.5(c)) da distribuição de feromônio no grid após o término de execução do algoritmo é ilustrada a título de contraste. Por intermédio dessas figuras é fácil observar a alta concentração de feromônio no grid onde existem vários objetos depositados. Também pode ser observado nessas figuras que tanto o nível de feromônio médio sob o grid quanto o campo de visão das formigas tendem a estabilizar após um número de iterações. No caso particular da visão, todas as formigas estabilizam seu campo de visão numa dimensão $7 \times 7$.

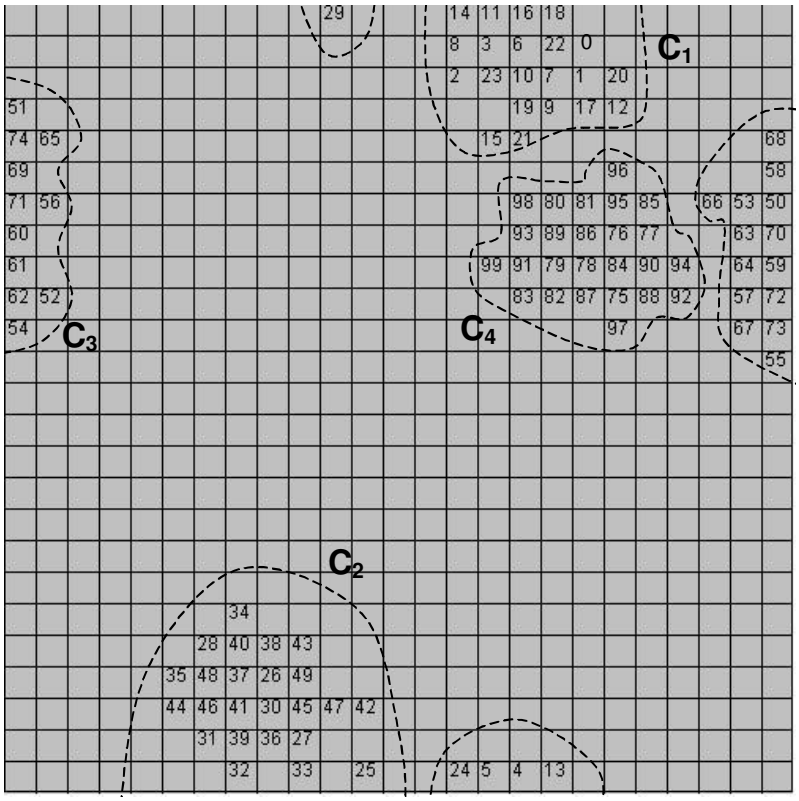



(a)

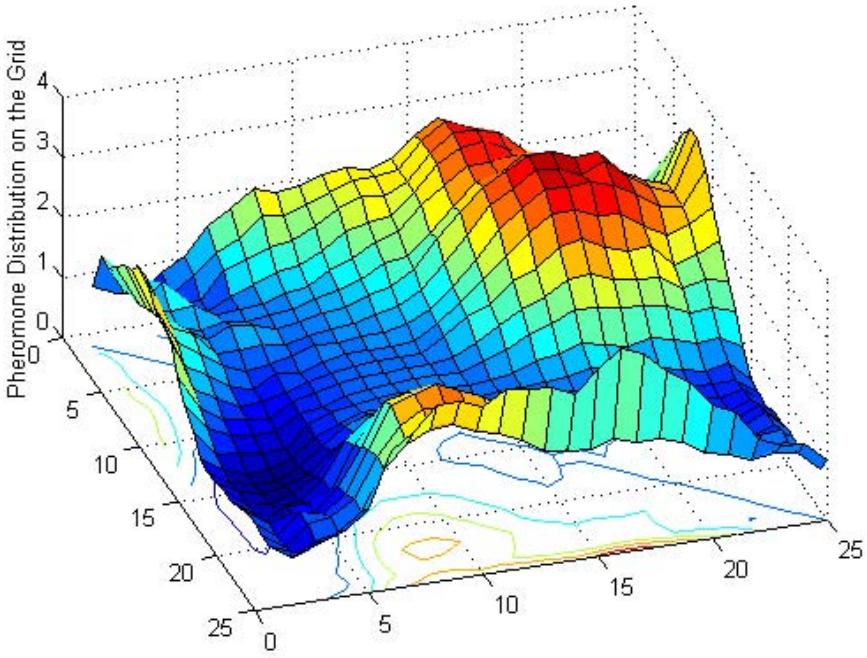



(b)

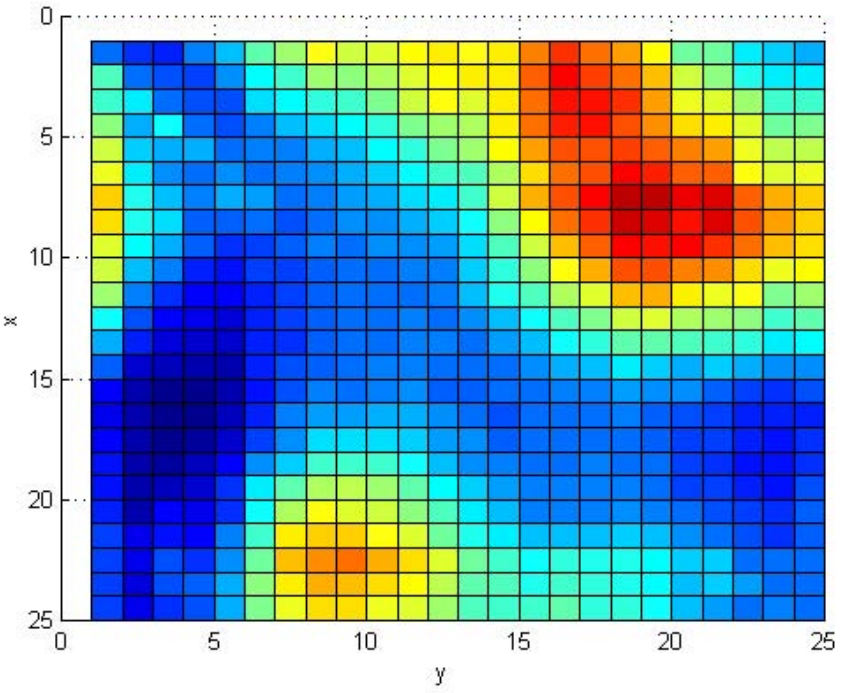
Figura 4.4: Evolução do nível do feromônio médio sobre o grid (a), e o campo de visão médio das formigas (b) durante o experimento representado na Figura 4.3(b-1).




(a)




(b)




(c)

Figura 4.5: Objetos e distribuição de feromônio sobre o grid. (a) Distribuição de objetos sobre o grid após o término de execução (Figura 4.3(b-1)). Perspectiva tridimensional (b) e bidimensional (c) da distribuição de feromônio no grid.

### 4.4.2. Base de Dados Animais

Esta seção compara o desempenho do $A^2CA$ com o SACA quando submetidos ao conjunto de dados Animais. Este é um conjunto de dados de alta dimensionalidade proposto por Ritter & Kohonen (1989) para verificar a capacidade de um mapa auto-organizado agrupar animais com características semelhantes. O conjunto de dados é composto por 16 vetores de entrada, cada um representando um animal cujos atributos assumem valores binários, como mostra a Tabela 4.1. Um valor igual a 1 nessa tabela corresponde à presença de um atributo, enquanto um valor igual a 0 indica a ausência deste atributo.

Tabela 4.1: Base de dados Animais com os nomes dos animais e seus respectivos atributos binários (Fonte: Haykin, 2001).

| | | pomba | galinha | pato | ganso | coruja | falcão | águia | raposa | cachorro | lobo | gato | tigre | leão | cavalo | zebra | vaca |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| é | pequeno | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | médio | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | grande | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| tem | 2 pernas | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 4 pernas | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | pêlo | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | cascos | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| | crina | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| | penas | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gosta | caçar | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | correr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| | voar | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | nadar | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Os resultados obtidos com a execução de 10 experimentos para o SACA e o $A^2CA$ utilizando a base de dados Animais são apresentados na Tabela 4.2. Como pode ser observado, o algoritmo $A^2CA$ consistentemente determinou dois grupos de dados, um correspondendo aos pássaros e um outro representando os mamíferos. Na maioria das vezes o algoritmo SACA apresentou os mesmos resultados, mas em alguns casos os mamíferos foram separados em dois grupos aparentemente sem sentido. Por exemplo, no experimento de número 5, o SACA agrupou Leão(12) com Cavalo(13) e Zebra(14). Em (Haykin, 2001;

p.476), um mapa auto-organizável de Kohonen agrupa estes dados em três principais grupos: pássaros, mamíferos pacíficos e caçadores. Entretanto, o agrupamento desse conjunto de dados pode ser aceito como a separação em apenas dois diferentes grupos, como os resultados obtidos pelos algoritmos SACA e $A^2CA$ (Haykin, 2001; p.476).

Tabela 4.2: Grupos encontrados pelo SACA e $A^2CA$ no conjunto de dados Animais. $N_c$ é o número de grupos encontrados.

| | **SACA*** | | **$A^2CA$** | |
|---|---|---|---|---|
| Execução | $N_c$ | **Grupos** | $N_c$ | **Grupos** |
| 1 | 2 | (0-6) (7-15) | 2 | (0-6) (7-15) |
| 2 | 2 | (0-6) (7-15) | 2 | (0-6) (7-15) |
| 3 | 2 | (0-6) (7-15) | 2 | (0-6) (7-15) |
| 4 | 3 | (0-6) (10) (7-9,11-15) | 2 | (0-6) (7-15) |
| 5 | 3 | (0,6) (7-11,15) (12-14) | 2 | (0-6) (7-15) |
| 6 | 2 | (0-6) (7-15) | 2 | (0-6) (7-15) |
| 7 | 3 | (0-6) (7-12,15) (13,14) | 2 | (0-6) (7-15) |
| 8 | 2 | (0-6) (7-15) | 2 | (0-6) (7-15) |
| 9 | 2 | (0-6) (7-15) | 2 | (0-6) (7-15) |
| 10 | 2 | (0-6) (7-15) | 2 | (0-6) (7-15) |
| **Média ± desvio padrão** | $2 \pm 0{,}48$ | | $2 \pm 0$ | |

### 4.4.3. Base de Dados Ruspini

Esse conjunto de dados é comumente utilizado na análise de desempenho de algoritmos de agrupamentos (Kaufman & Rousseeuw, 1990). Ele é composto por 75 objetos agrupados em quatro classes, como mostra a Figura 4.6, onde $N_c$ representa o número de classes encontradas e $P_{ic}$ o percentual de classificação incorreta.

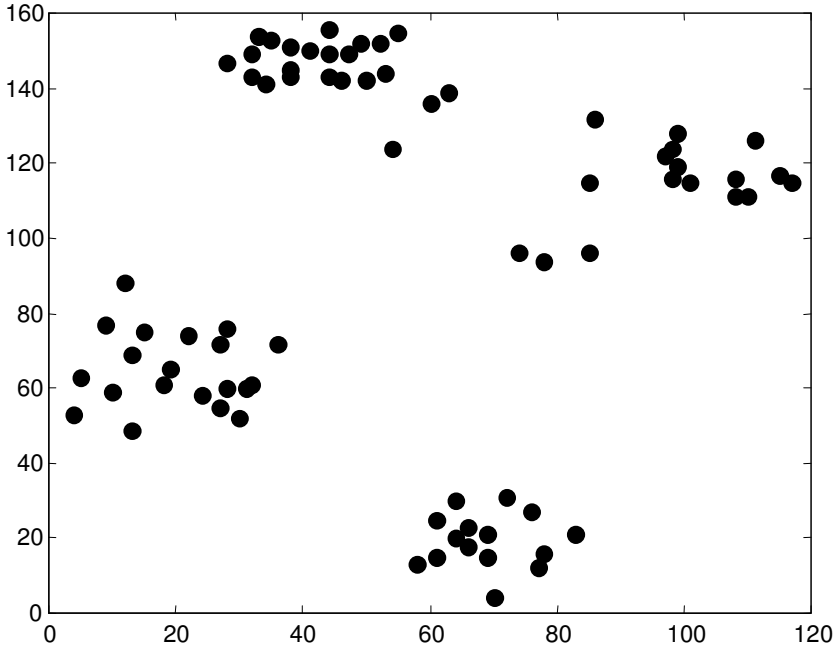


Figura 4.6: Representação gráfica da base de dados Ruspini.

A Tabela 4.3 sumariza os resultados obtidos pelos dois algoritmos quando aplicados ao conjunto Ruspini. Os resultados apresentados são a média ± o desvio padrão obtidos a partir de 10 execuções de cada algoritmo. Note que o algoritmo $A^2CA$ consegue encontrar o número correto de classes, agrupando todos os objetos corretamente.

Tabela 4.3: Desempenho do SACA e $A^2CA$ aplicados à base de dados Ruspini.

| | **SACA** | | **$A^2CA$** | |
|---|---|---|---|---|
| | $N_c$ | $P_{ic}$ (%) | $N_c$ | $P_{ic}$ (%) |
| Ruspini | $7.4 \pm 1.46$ | $1.5 \pm 2.72$ | $4.0 \pm 0.0$ | $0 \pm 0.0$ |

### 4.4.4. Base de Dados Yeast Galactose

O conjunto de dados *yeast galactose* (Yeung *et al*., 2003) é uma base real utilizada em bioinformática. Cada objeto é constituído por 20 atributos – nove deleções de único gene, um atributo anormal (*wild-type*) com *galactose* e *rafinose*, nove deleções e um atributo anormal sem *galactose* e *rafinose*. Para a realização dos experimentos, foi utilizado um subconjunto de 205 genes (objetos), os quais refletem padrões de quatro categorias

funcionais (grupos) formadas por 83, 15, 93 e 14 genes, respectivamente. Para esse conjunto de dados o SACA demonstrou-se incapaz de agrupar os objetos corretamente na maioria dos experimentos. O $A^2CA$, entretanto, demonstrou ser capaz de agrupar de forma apropriada os objetos em todos os experimentos, com pequenas variações no número de classes encontradas a cada execução. Em 10 execuções, o $A^2CA$ apresentou os seguintes resultados: $n_c = 6.9 \pm 1.0$ e $P_{ic} = 3.17\% \pm 0.93\%$. A Figura 4.7 ilustra uma das soluções encontradas pelo $A^2CA$ quando aplicado ao conjunto *yeast galactose*. Esta figura também mostra as classes encontradas (demarcadas pelas linhas pontilhadas) e os objetos agrupados de forma incorreta (demarcados por linhas contínuas).

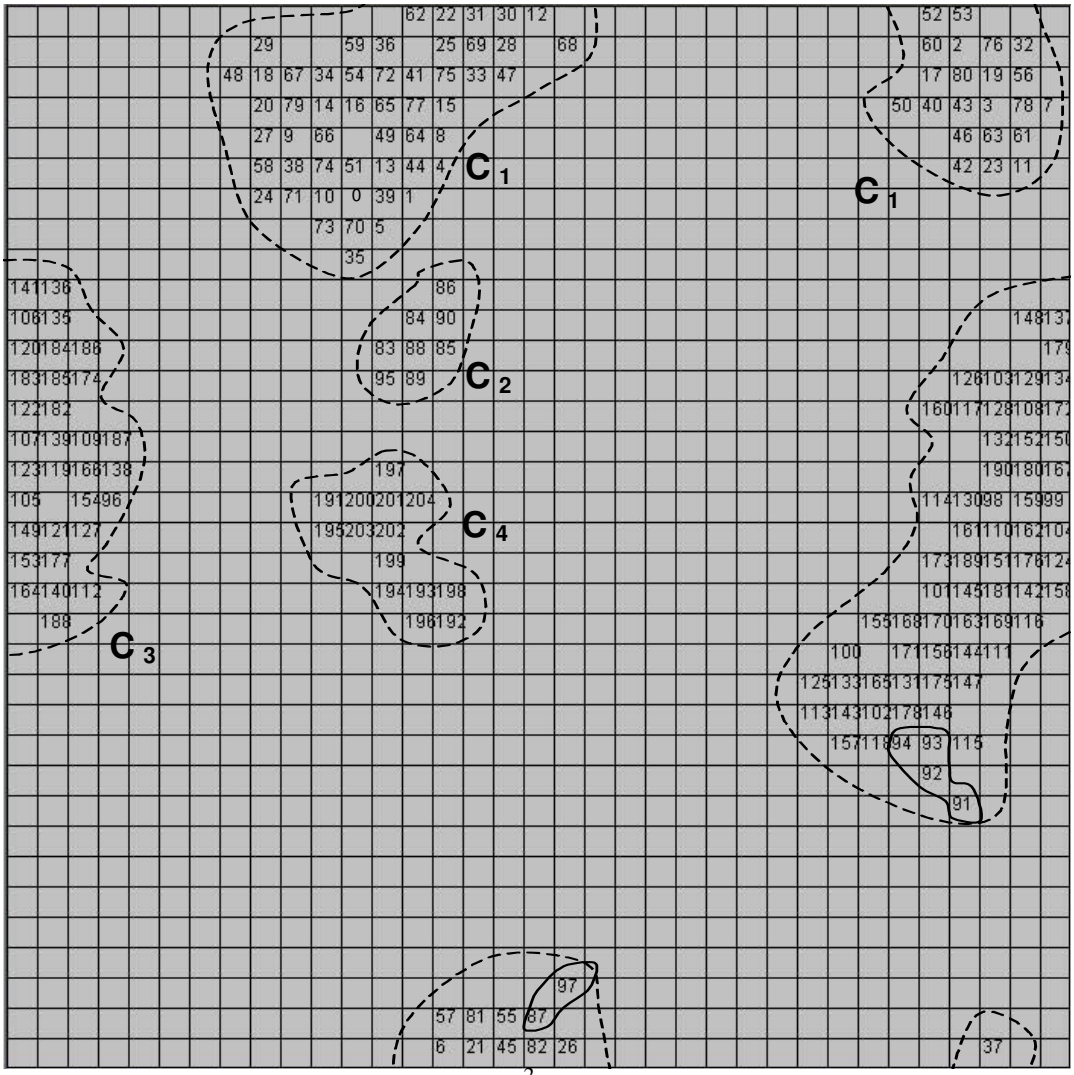



Figura 4.7: Solução encontrada pelo $A^2CA$ quando aplicado a base de dados de bioinformática *Yeast Galactose*.

## 4.5. Agrupamento de Documentos

O agrupamento de documentos tem como objetivo encontrar *clusters* de documentos baseado nas características que esses objetos apresentam, por intermédio de uma medida de dissimilaridade, geralmente expressa em termos de distância, de tal forma que seja minimizada a distância entre documentos de um mesmo grupo e maximizada a distância entre grupos distintos.

O agrupamento de documentos vem sendo utilizado em sistemas de recuperação de informação (Hearst & Pedersen, 1996), na geração automática de clusters hierárquicos (Zhao & Kharypis, 2002), e na organização dos resultados de máquinas de busca (Zamir & Etzioni, 1998) podendo, inclusive, ser empregado na organização de documentos que possuam características semelhantes. Assim, ele pode servir, por exemplo, como um sistema de recomendação de artigos científicos da comunidade virtual acadêmica, apresentando aos usuários artigos ainda não lidos que compartilhem características similares de acordo com o perfil de interesse de cada usuário. Essa recomendação poderia ser feita de forma visual, sendo que o usuário teria uma representação gráfica dos grupos de documentos formados, facilitando assim o processo de busca por assuntos correlatos.

Com base nos resultados obtidos apresentados na seção anterior e por ser um algoritmo de agrupamento que possui uma interface gráfica que possibilita o acompanhamento visual do processo de clusterização, tanto o SACA como o $A^2CA$ poderiam ser adaptados para servirem como um sistema de recomendação de artigos cuja interface gráfica seria interativa. Esta interface permitiria, além da visualização dos documentos, o acesso aos artigos que ainda não foram lidos e que se encontram numa região de vizinhança próxima aos documentos já lidos. Essa diferenciação entre os documentos lidos e não lidos poderia ser obtida pela adoção de cores distintas.

Esta seção investiga o desempenho dos algoritmos SACA e $A^2CA$ quando submetidos ao agrupamento de bases textuais. Para os experimentos foram utilizados dois conjuntos de dados sendo o conjunto *resumos* composto por 80 documentos extraídos de uma combinação de bases freqüentemente utilizadas na avaliação de sistemas de recuperação de

informação[3]. São elas: ADI (recuperação de informação), CRANFIELD (aerodinâmica), MEDLARS (medicina), e NPL (eletrônica), constituindo um conjunto de quatro classes com vinte documentos cada. O segundo conjunto, denominado *artigos*, possui três classes de 20 artigos cada, totalizando 60 documentos PDF que foram copiados dos Anais do IEEE WCCI 2002 (*World Congress on Computational Intelligence* 2002). Os parâmetros de configuração, tanto do algoritmo SACA quanto do $A^2CA$, foram fixados em: $\theta = 0.6$, $k_p = 0.01$, $k_d = 0.06$, $n_{ants} = 10$, grid = 35×35, e $\alpha = 0.75$.

Os resultados obtidos pelos algoritmos SACA e $A^2CA$ foram comparados com um algoritmo da ferramenta CLUTO (Zhao & Kharypis, 2002). A comparação com o *toolkit* CLUTO foi motivada pelos estudos realizados por Melo & Lopes (2004) e Zhao & Kharypis (2005).

Melo & Lopes (2004) investigaram diversas técnicas de *clustering* para a escolha de uma que fosse apropriada para o agrupamento de documentos sem a intervenção humana e que pudesse ser utilizada como ferramenta inteligente de apoio à pesquisa. As ferramentas de *clustering* utilizadas foram: CLUTO (Zhao & Kharypis, 2002), gCLUTO (Rasmussen & Kharypis,2004), Gmeans (Guan, 2005), Som_Pak (Kohonen *et al,* 1996) e GHSOM (Rauber, 2006).

Os resultados obtidos por Melo & Lopes (2004) vieram de encontro às conclusões feitas por Zhao & Kharypis (2005) que avaliaram nove algoritmos de clusterização que utilizam o método aglomerativo (*bottom-up*) e seis algoritmos que utilizam o método particional (*top-down*), chegando a conclusão que os algoritmos que utilizam o método particional produzem melhores soluções de agrupamento, sobretudo quando os documentos possuem baixa similaridade entre si ou quando existem muitas semelhanças entre todos os documentos.

[3] Fontes disponíveis em: ftp://ftp.cs.cornell.edu/pub/smart/

O critério de avaliação adotado para medir a qualidade dos agrupamentos obtidos pelos algoritmos SACA, A$^2$CA e CLUTO observou dois quesitos: *i*) a entropia; e *ii*) a pureza do *cluster*. A entropia define a homogeneidade dos grupos formados, ou seja, de que forma as classes de documentos estão distribuídas em cada cluster, sendo que baixa entropia indica clusters mais homogêneos. Dado um determinado cluster $S_r$ de tamanho $n_r$, a entropia ($E$) deste cluster pode ser medida da seguinte forma:

$$E(S_r) = -\frac{1}{\log q}\sum_{i=1}^{q}\frac{n_r^i}{n_r}\log\frac{n_r^i}{n_r} \qquad (4.11)$$

onde $q$ é o número de classes no conjunto de documentos, e $n_r^i$ é o número de documentos da *i*-é*sima* classe presente no cluster $S_r$. A entropia global pode então ser calculada como a somatória das entropias obtidas em cada cluster, sendo ponderada pelo tamanho de cada cluster, da seguinte forma:

$$E_{global} = \sum_{r=1}^{k}\frac{n_r}{n}E(S_r) \qquad (4.12)$$

Uma solução de agrupamento ideal será aquela que apresenta documentos de uma única classe dentro de cada cluster fazendo com que a entropia seja igual a zero.

A pureza fornece a razão da classe dominante no cluster em relação ao tamanho do próprio cluster e pode ser calculada da seguinte maneira:

$$P(S_r) = \frac{1}{n_r}\max_i(n_r^i) \qquad (4.13)$$

onde $n_r^i$ e $n_r$ são definidos como anteriormente.

De forma análoga ao cálculo da entropia global, a pureza global pode ser obtida por:

$$P_{global} = \sum_{r=1}^{k} \frac{n_r}{n} P(S_r) \qquad (4.14)$$

Cada *cluster* pode conter documentos de diferentes classes. Valores de pureza próximos a um indicam um subconjunto puro da classe dominante no *cluster*.

O algoritmo do *toolkit* CLUTO utilizado nos experimentos foi o algoritmo particional RB (*repeated bisections*) (Zhao & Kharypis (2005). Neste método, a solução desejada é calculada executando uma seqüência de repetidas bi-seções de *k* (sendo *k* o número de *clusters* a ser encontrado). Inicialmente a tabela que representa os documentos é segmentada formando dois grupos; em seguida, um destes grupos é selecionado e segmentado novamente, e assim por diante, até que o número desejado (*k*) de grupos seja encontrado. Esse algoritmo faz uso de uma entrada de dados pré-definida, sendo necessária uma fase de pré-processamento na qual os documentos são transformados de texto para uma tabela conhecida por *bag-of-words* construída através do modelo de espaço vetorial (Seção 3.2.4). O peso atribuído a cada atributo foi calculado utilizando a freqüência relativa da palavra no documento (Equação 3.4). Essa tabela é inicialmente particionada em dois *clusters*, então um desses *clusters* é selecionado e partiocionado. Esse processo continua até se atingir o número desejado de *clusters* definido a priori. A função de otimização utilizada visa maximizar a similaridade entre cada documento e o centróide do cluster ao qual ele foi associado (Zhao & Kharypis, 2005). A medida de similaridade utilizada é a medida do coseno (Salton & McGill, 1997).

### 4.5.1. Avaliação de Desempenho

Todos os documentos dos dois conjuntos de dados foram submetidos a uma etapa de pré-processamento, sendo removidas as palavras irrelevantes (Seção 3.4.1). No intuito de minimizar a redundância e, consequentemente, aumentar a qualidade das palavras selecionadas, o algoritmo de Porter (1980) para *stemming* foi utilizado. Tanto o algoritmo SACA como o $A^2CA$ sofreram uma alteração na função responsável pelo cálculo de densidade (Equação 4.2). A distância Euclidiana, que era utilizada para avaliar a dissimilaridade entre os objetos, foi substituída pela medida do coseno, não só pelo fato do

algoritmo RB utilizá-la como medida de similaridade, mas devido a testes previamente realizados, os quais apresentaram melhores resultados quando a medida do coseno foi escolhida ao invés da distância Euclidiana. Os resultados obtidos com a execução de 10 experimentos para o SACA e o $A^2CA$ utilizando o conjunto de dados *resumos* são apresentados na Tabela 4.4, onde *k* representa o número de *clusters* encontrados no experimento.

Tabela 4.4: Comparativo de desempenho entre os algoritmos $A^2CA$ e SACA para 10 experimentos com a base *resumos*.

| | **Algoritmo $A^2CA$** | | | **Algoritmo SACA** | | |
|---|---|---|---|---|---|---|
| **nro exp.** | ***k*** | **Entropia** | **Pureza** | ***k*** | **Entropia** | **Pureza** |
| 1 | 11 | 0,15 | 0,77 | 15 | 0,18 | 0,75 |
| 2 | 11 | 0,26 | 0,77 | 11 | 0,28 | 0,68 |
| 3 | 10 | 0,24 | 0,75 | 14 | 0,34 | 0,65 |
| 4 | 12 | 0,21 | 0,77 | 15 | 0,24 | 0,67 |
| 5 | 12 | 0,17 | 0,78 | 12 | 0,30 | 0,65 |
| 6 | 13 | 0,25 | 0,78 | 15 | 0,16 | 0,71 |
| 7 | 12 | 0,21 | 0,77 | 13 | 0,22 | 0,77 |
| 8 | 13 | 0,18 | 0,77 | 14 | 0,29 | 0,67 |
| 9 | 11 | 0,07 | 0,73 | 15 | 0,24 | 0,70 |
| 10 | 13 | 0,1 | 0,81 | 19 | 0,14 | 0,77 |
| **Média** | **11,8** | **0,18** | **0,77** | **14,3** | **0,24** | **0,70** |

Observando os resultados da Tabela 4.4, numa primeira análise, podemos formular a hipótese de que o algoritmo $A^2CA$ obteve um desempenho superior em relação ao algoritmo SACA visto que em média, encontrou um menor número de *clusters,* uma menor entropia e uma maior pureza.

Levando-se em consideração essa hipótese, e tendo em vista que o algoritmo RB do *toolkit* CLUTO necessita do número de clusters definidos a priori, a Tabela 4.5 apresenta os resultados obtidos com o algoritmo RB do *toolkit* CLUTO para 10 experimentos com a base de dados *resumos* para um número de *k* igual a 12. A adoção do número de clusters tomou por base o número médio de clusters encontrado pelo algoritmo $A^2CA$ que foi igual a 11,8.

Tabela 4.5: Resultados do algoritmo RB – CLUTO para base *resumos*.

| | Algoritmo RB – CLUTO | | |
|---|---|---|---|
| nro. exp. | *k* | **Entropia** | **Pureza** |
| 1 | 12 | 0,15 | 0,91 |
| 2 | 12 | 0,15 | 0,91 |
| 3 | 12 | 0,15 | 0,91 |
| 4 | 12 | 0,15 | 0,91 |
| 5 | 12 | 0,18 | 0,9 |
| 6 | 12 | 0,18 | 0,9 |
| 7 | 12 | 0,18 | 0,9 |
| 8 | 12 | 0,15 | 0,91 |
| 9 | 12 | 0,15 | 0,91 |
| 10 | 12 | 0,15 | 0,91 |
| Média | **12** | **0,16** | **0,90** |

Os resultados obtidos com a execução de 10 experimentos para o SACA e o $A^2CA$ utilizando o conjunto *artigos* são apresentados na Tabela 4.6.

Tabela 4.6: Comparativo de desempenho entre os algoritmos $A^2CA$ e SACA para 10 experimentos com a base *artigos*.

| | **Algoritmo $A^2CA$** | | | **Algoritmo SACA** | | |
|---|---|---|---|---|---|---|
| **nro exp.** | ***k*** | **Entropia** | **Pureza** | ***K*** | **Entropia** | **Pureza** |
| 1 | 14 | 0,15 | 0,78 | 8 | 0,64 | 0,51 |
| 2 | 6 | 0,59 | 0,53 | 8 | 0,80 | 0,41 |
| 3 | 6 | 0,75 | 0,43 | 10 | 0,70 | 0,58 |
| 4 | 9 | 0,57 | 0,66 | 7 | 0,76 | 0,51 |
| 5 | 7 | 0,54 | 0,55 | 10 | 0,66 | 0,51 |
| 6 | 8 | 0,46 | 0,51 | 8 | 0,69 | 0,50 |
| 7 | 6 | 0,6 | 0,53 | 9 | 0,80 | 0,45 |
| 8 | 8 | 0,7 | 0,5 | 11 | 0,67 | 0,50 |
| 9 | 10 | 0,38 | 0,65 | 9 | 0,67 | 0,50 |
| 10 | 10 | 0,38 | 0,58 | 10 | 0,63 | 0,51 |
| **Média** | **8,4** | **0,51** | **0,57** | **9** | **0,70** | **0,49** |

Novamente pode-se observar que o algoritmo $A^2CA$ apresenta melhores resultados que o SACA em todos os quesitos avaliados, o que o levou a ser novamente comparado com os resultados obtidos pelo algoritmo RB – CLUTO para 10 experimentos com a base *artigos*, sendo o número de clusters igual a oito (ver Tabela 4.7).

Tabela 4.7: Resultados do algoritmo RB – CLUTO para base *artigos*.

| | Algoritmo RB - CLUTO | | |
|---|---|---|---|
| nro. exp. | *k* | **Entropia** | **Pureza** |
| 1 | 8 | 0,72 | 0,63 |
| 2 | 8 | 0,72 | 0,63 |
| 3 | 8 | **0,57** | **0,73** |
| 4 | 8 | 0,72 | 0,63 |
| 5 | 8 | 0,72 | 0,63 |
| 6 | 8 | 0,60 | 0,73 |
| 7 | 8 | 0,65 | 0,65 |
| 8 | 8 | 0,61 | 0,71 |
| 9 | 8 | 0,64 | 0,68 |
| 10 | 8 | 0,58 | 0,71 |
| Médias | **8** | **0,65** | **0,67** |

Os resultados obtidos com os experimentos realizados com os três algoritmos mostram que o RB-CLUTO foi o que obteve as melhores médias em todos os quesitos na base de dados *resumos*, sendo seguido de perto pelo algoritmo $A^2CA$. Nos testes realizados com a base de dados *artigos*, o algoritmo $A^2CA$ obteve um valor médio de entropia menor do que o obtido pelo algoritmo RB-CLUTO e um valor médio de pureza bem próximo ao obtido pelo algoritmo RB-CLUTO. O algoritmo SACA foi o algoritmo que obteve as piores médias.

Se considerarmos os experimentos realizados com os algoritmos como amostras de soluções de clusterização, obtidas a partir de uma população (conjunto de todas as possíveis soluções de clusterização) podemos fazer uso de ferramentas estatísticas para realizar um *teste de hipóteses*. Em estatística um teste de hipóteses pode ser utilizado para avaliar diferenças entre duas ou mais amostras. Um teste bastante conhecido é o teste da *hipótese nula*, o qual, de maneira simplificada, afirma terem duas amostras sido extraídas da mesma população. De acordo com a hipótese nula, qualquer diferença observada entre as amostras (no nosso caso experimentos) é considerada como uma ocorrência casual, mero resultado do erro amostral. Portanto, uma diferença entre duas médias amostrais não representa, à luz da hipótese nula, uma verdadeira diferença entre as médias populacionais (Levin, 1977).

No contexto dos experimentos realizados com os algoritmos a hipótese nula pode ser simbolizada das seguintes formas:

1. $M_{SACA} = M_{A^2CA}$
2. $M_{SACA} = M_{CLUTO}$
3. $M_{A2CA} = M_{CLUTO}$

onde, $M_{SACA}$ é a média da população do algoritmo SACA, $M_{A^2CA}$ é a média da população do algoritmo $A^2$CA e $M_{CLUTO}$ é a média da população do algoritmo CLUTO.

Examinando as hipóteses nulas relativas ao desempenho dos algoritmos para cada quesito analisado (*entropia* e *pureza*) temos:

1) Não existe variação de desempenho entre os algoritmos SACA e $A^2$CA;
2) Não existe variação de desempenho entre os algoritmos SACA e CLUTO;
3) Não existe variação de desempenho entre os algoritmos $A^2$CA e CLUTO;

É importante ressaltar que a hipótese nula não nega a possibilidade de ocorrerem diferenças entre médias amostrais. Ao contrário, ela procura explicar tais diferenças através da inevitável presença do erro amostral. De acordo com a hipótese nula, por exemplo, ao confrontarmos a média de entropia obtida pelo algoritmo $A^2$CA (Tabela 4.6) com a média de entropia obtida pelo algoritmo CLUTO (Tabela 4.7), não se deve concluir só por esses resultados (o valor obtido pelo algoritmo CLUTO foi menor do que o algoritmo $A^2$CA) que a população (possíveis soluções de clusterização) do algoritmo $A^2$CA produz uma entropia maior do que a população do algoritmo CLUTO. Em lugar disso, trata-se essa diferença amostral (0,18 – 0,16 = 0,02) como um resultado do erro amostral, ou seja, aquela diferença que inevitavelmente surge quando extraímos uma amostra de uma população.

A hipótese nula é, em geral (embora não necessariamente), estabelecida com esperança de que possa ser rejeitada. Se a hipótese nula é rejeitada, ou seja, se chega-se a conclusão que a hipótese que antecipa a inexistência de diferença entre as médias é frágil, automaticamente aceita-se a *hipótese experimental* que afirma existir uma verdadeira diferença populacional.

No contexto dos experimentos realizados com os algoritmos a hipótese experimental pode ser simbolizada das seguintes formas:

1. $M_{SACA} \neq M_{A^2CA}$
2. $M_{SACA} \neq M_{CLUTO}$
3. $M_{A2CA} \neq M_{CLUTO}$

Examinando as hipóteses experimentais acima para cada quesito analisado (entropia e pureza) temos:

1. Existe variação de desempenho entre os algoritmos SACA e $A^2CA$;
2. Existe variação de desempenho entre os algoritmos SACA e CLUTO;
3. Existe variação de desempenho entre os algoritmos $A^2CA$ e CLUTO;

*Para decidir se a diferença amostral obtida é estatisticamente significante, ou seja, resultado de uma real diferença entre as populações e não apenas resultado do erro amostral, é habitual estabelecer um nível de confiança que representa a probabilidade com que a hipótese nula pode ser rejeitada com confiança (segurança) ou, dizendo de outro modo, a probabilidade com que a hipótese experimental pode ser aceita (com confiança)* (Levin, 1977).

Objetivando comparar estatisticamente os resultados obtidos adotou-se o teste-*t* (*t-student*) com nível de confiança para a rejeição da hipótese nula igual a 0,05 (5%), o que significa dizer que a hipótese nula será rejeitada se a diferença amostral obtida ocorrer *por acaso* somente 5 vezes ou menos em 100.

O nome *t-student* deve-se a William Gosset, que escreveu com o pseudônimo Student, em 1908, o trabalho intitulado “The Probable Error of a Mean”. Neste trabalho Gosset especulou sobre a importância de se ver o valor médio de uma amostra de um experimento como o exemplo do valor médio de uma “população de experimentos realizados sob as mesmas condições”. Esta idéia de uma população de experimentos gerou

o que se denomina atualmente de distribuição de médias amostradas. Quando se amostra um experimento as seguintes observações são válidas:

- À medida em que se aumenta o tamanho da amostra sobre a qual a média é calculada, a distribuição obtida tende progressivamente a uma distribuição na forma de sino. Isto se deve ao teorema do limite central, que postula que a distribuição da média tende à normalidade (distribuição normal) à medida em que o número de amostras cresce;
- A distribuição das médias é centrada em torno da média da população. A razão disto é que o valor esperado da amostra é o valor médio da população.

A distribuição *t-student* é útil quando se deseja especificar a incerteza do valor médio da amostra de um experimento para um dado nível de confiança. Neste caso, não se conhece o desvio padrão da população de dados experimentais, sendo o intervalo de confiança a probabilidade de que a incerteza a ser obtida inclua a média.

O teste *t* pode ser resumido pela seguinte equação:

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sigma_{dif}}, \tag{4.15}$$

onde $X_1$ é a média da primeira amostra, $X_2$ é a média da segunda amostra e $\sigma_{dif}$ é o erro padrão da diferença.

O erro padrão da diferença pode ser obtido através do seguinte cálculo:

$$\sigma_{dif} = \sqrt{\sigma\overline{X}_1^2 + \sigma\overline{X}_2^2} \tag{4.16}$$

onde $\sigma\overline{X} = \frac{s}{\sqrt{N-1}}$ representa o erro padrão de cada média, sendo $s$ o desvio padrão da amostra e $N$ o número de objetos da amostra

Se tomarmos como exemplo a hipótese formulada no início dessa seção de que o desempenho do algoritmo $A^2CA$ foi superior ao do algoritmo SACA, podemos usar o teste *t* para testar se a diferença obtida entre eles nos quesitos *entropia* e *pureza* é estatisticamente significante. Neste caso obteve-se $t_{entropia} = 2{,}14$ e $t_{pureza} = 5$ o que rejeita a hipótese nula e se aceita a hipótese experimental (SACA$\neq A^2CA$).

Os resultados da análise estatística comparando os algoritmos estão sumarizados nas Tabelas 4.8 e 4.9:

Tabela 4.8: Teste *t* obtido para os quesitos *entropia* e *pureza* para a base de dados *resumos*.

| | $t_{entropia}$ | $t_{pureza}$ | $t_{crítico\ (tabelado)}$ |
|---|---|---|---|
| $A^2CA$ x SACA | 2,142 | 5 | 2,101 |
| $A^2CA$ x CLUTO | 1 | 14 | 2,101 |
| CLUTO x SACA | 4 | 21 | 2,101 |

Confrontando os dados da Tabela 4.8 com as médias de pureza e entropia obtidas pelos algoritmos (ver tabela 4.4), pode-se verificar que o algoritmo $A^2CA$ foi superior ao algoritmo SACA nos dois quesitos e quando comparado com o algoritmo CLUTO no quesito entropia foi equivalente, sendo inferior no quesito pureza.

Tabela 4.9: Teste *t* obtido para os quesitos *entropia* e *pureza* para a base de dados *artigos*

| | $t_{entropia}$ | $t_{pureza}$ | $t_{crítico\ (tabelado)}$ |
|---|---|---|---|
| $A^2CA$ x SACA | 3,166 | 2,187 | 2,101 |
| $A^2CA$ x CLUTO | 2,329 | 3,046 | 2,101 |
| CLUTO x SACA | 1,767 | 10,2 | 2,101 |

Nos experimentos realizados com a base de dados artigos, o algoritmo $A^2CA$ manteve sua superioridade em relação ao algoritmo SACA e apresentou um desempenho superior ao algoritmo CLUTO na avaliação do quesito *entropia*. O algoritmo CLUTO por sua vez, foi superior ao algoritmo $A^2CA$ no quesito *pureza* e teve uma queda de desempenho no quesito *entropia*, ficando equivalente ao algoritmo SACA neste quesito.

## 4.6. Sumário do Capítulo

O algoritmo de agrupamento inspirado em formigas é um sistema multi-agente auto-organizado tipicamente utilizado para agrupamento de dados não rotulados. Seu objetivo é

projetar os dados originais dentro de um *grid* bidimensional e posicionar aqueles itens que são similares entre si no seu espaço de origem em regiões vizinhas do *grid*. Fazendo isto o algoritmo é capaz de agrupar itens similares, apresentando o resultado desse agrupamento num ambiente gráfico bidimensional, o que ajuda o usuário no acompanhamento e visualização do processo. A vantagem do acompanhamento visual é que o usuário está diretamente envolvido no processo de mineração de dados (Keim, 2002).

O objetivo principal desse capítulo foi apresentar algumas contribuições para a melhoria de desempenho e critério de convergência para o SACA. Em particular, um resfriamento gradativo para o parâmetro que regula a probabilidade de pegar objetos do *grid* foi proposto. Isto faz com que o algoritmo estabilize depois de um certo número de iterações. Além disso, foi implementada a idéia de visão progressiva proposta por Sherafat *et al*. (2004a) e uma nova forma da distribuição de feromônio no *grid* foi proposta, permitindo que grupos de dados reforcem a atração daquelas regiões do *grid* onde existem maiores concentrações de dados.

O algoritmo adaptativo proposto, chamado $A^2CA$, foi aplicado a algumas bases de dados sintéticos de teste e a três bases de dados reais, uma de bioinformática e duas de dados textuais. Os resultados obtidos foram comparados com o SACA implementado com o resfriamento gradativo, o que ressaltou os benefícios das contribuições apresentadas. Com exceção das bases textuais, o $A^2CA$ mostrou-se robusto no que tange a encontrar o número correto de grupos e sempre estabilizou depois de um número fixo de iterações.

Na análise de resultados das bases textuais, os algoritmos SACA e $A^2CA$ foram comparados com o algoritmo RB-CLUTO. A adoção pelo comparativo com o *toolkit* CLUTO foi motivada pelo bom desempenho desse algoritmo na clusterização de textos Os critérios utilizados como mediadas de desempenho foram entropia e pureza do cluster. Objetivando comparar estatisticamente os resultados obtidos adotou-se o teste *t* (*t-student*). Assim como nos outros testes realizados, o algoritmo $A^2CA$ confirmou a sua superioridade em relação ao algoritmo SACA, além de ser equivalente ao algoritmo CLUTO no quesito entropia para um conjunto de dados e superior no mesmo quesito no segundo conjunto de documentos.

# 5. Conclusão e Perspectivas Futuras

Nesse trabalho foram investigadas duas ferramentas utilizando técnicas da computação inspirada na biologia: *i*) um sistema evolutivo para a otimização de busca de informações na *web*; e *ii*) um algoritmo adaptativo inspirado no comportamento de formigas para agrupamento de dados. As ferramentas desenvolvidas nesse trabalho podem ser utilizadas para a criação de um ambiente computacional que não só facilite a busca de artigos científicos como também viabilize a interação de pessoas (pesquisadores) possibilitando a troca de informação e, por sua vez, o compartilhamento de conhecimentos obtidos. Isso visa, dentre outras coisas, evitar a duplicação de esforços, aumentar a eficiência do processo de busca e organizar e compartilhar informações obtidas via *web* em *comunidades virtuais acadêmicas* (Vizine et al., 2005).

O sistema evolutivo para otimização de busca na *web* é composto por dois algoritmos: *i*) um deles responsável pela extração de palavras-chave dos documentos e a classificação destas num dicionário de termos; e *ii*) um algoritmo genético que faz uso das palavras existentes no dicionário de termos para buscar novos documentos na *web* que sejam relevantes para o usuário.

Durante os testes realizados com o sistema evolutivo, pôde-se constatar que mesmo com um espaço de busca altamente dinâmico e algumas vezes instável, como é a Internet, o sistema foi capaz de melhorar a qualidade da busca efetuada por uma máquina de busca amplamente usada pela comunidade, que é o Google. Entretanto, os seguintes problemas foram detectados:

- Apesar do algoritmo de Porter para *stemming* ser utilizado em grande parte das aplicações que envolvem mineração de textos, no caso específico da busca na *web* utilizando o Google o mesmo não é recomendado, pois ele reduz a especificidade da busca;

- A adoção do Google (*API Google Engine*) para a realização das buscas também trouxe algumas limitações no tamanho do cromossomo empregado no algoritmo evolutivo, no número de indivíduos da população e no número de gerações, visto que a API limita-se a 1.000 *hits* (pesquisas) diários;

- Em alguns testes, constatou-se a perda do melhor indivíduo (mesmo utilizando estratégia salvacionista) devido a problemas de comunicação na rede.

O algoritmo $A^2CA$ (*Adaptive Ant Clustering Algorithm*) foi desenvolvido tendo em mente o uso de feromônio durante o processo de construção e organização dos ninhos em formigas e cupins. Em particular, o $A^2CA$ foi inspirado no comportamento dos cupins durante o processo de construção do ninho, no qual cada cupim deposita feromônio sobre partículas de terra recrutando outros cupins a fazerem o mesmo em regiões nas quais a concentração do feromônio seja maior. Uma outra observação biológica incorporada ao $A^2CA$ foi o fato de que a visão das formigas não se restringe à vizinhança local onde elas atuam, tendo a capacidade de percepção de uma área maior, que pode variar de formiga para formiga com o passar do tempo. O $A^2CA$ incorpora um esquema de visão adaptativa, a inclusão de feromônio sobre as células do *grid* e implementar um resfriamento gradativo no parâmetro que regula a probabilidade de pegar um objeto do *grid*.

O $A^2CA$ foi submetido a um conjunto de testes envolvendo problemas de agrupamento de dados. Os resultados obtidos foram comparados com o SACA implementado com o resfriamento gradativo, o que ressaltou os benefícios das contribuições propostas. Com exceção das bases textuais, o $A^2CA$ mostrou-se robusto no que tange a encontrar o número correto de grupos e sempre estabilizou depois de um número fixo de iterações.

Além de certa dificuldade no ajuste dos parâmetros utilizados pelo algoritmo, um dos principais problemas encontrados foi o agrupamento de bases textuais, talvez devido à alta dimensionalidade destas bases. É importante ressaltar que muito embora o algoritmo $A^2CA$ não tenha encontrado o número correto de *clusters* quando aplicado ao agrupamento de textos, mesmo assim o seu desempenho foi superior quando comparado ao algoritmo

SACA, o que justifica as contribuições propostas. Quando comparado ao algoritmo CLUTO (amplamente utilizado em agrupamento de textos), o $A^2CA$ mostrou equivalência de desempenho no quesito entropia para um conjunto de dados e superioridade no mesmo quesito no segundo conjunto de documentos.

## 5.1. Contribuições

Esse trabalho procurou demonstrar a importância que o estudo dos fenômenos biológicos exerce no desenvolvimento de aplicações computacionais. Ciente da complexidade e da abrangência dessa área de pesquisa optou-se por atuar em duas frentes distintas: *i*) na análise do comportamento de técnicas consolidadas, como os algoritmos genéticos, aplicadas a problemas emergentes, como a busca de informação na Internet; e *ii*) no aperfeiçoamento e análise de desempenho de novas propostas, como é o caso da inteligência de enxame quando aplicada a problemas clássicos de agrupamento de dados. Mais especificamente, esta dissertação possui as seguintes contribuições:

- Um mecanismo autônomo e adaptativo para a filtragem colaborativa de informação, baseado em um processo evolutivo de busca e uma ferramenta para extração automática de palavras-chave de documentos do tipo texto;

- Uma forma de estabilizar o comportamento do algoritmo de agrupamento por formigas (SACA) através do resfriamento gradativo do parâmetro que regula a probabilidade de pegar objetos do *grid* e

- A proposta de uma nova forma de distribuição de feromônio no *grid* para o algoritmo SACA, de modo que o algoritmo promova uma distribuição mais precisa dos dados no *grid*, formando assim clusters mais consistentes com os clusters naturais dos dados de entrada.

## 5.2. Trabalhos Futuros

Existe um número considerável de trabalhos futuros a serem realizados de modo a dar continuidade na consolidação das ferramentas propostas nesse trabalho e na criação da comunidade virtual acadêmica, dentre os quais destacam-se:

- Alteração no cálculo da freqüência relativa do algoritmo extrator de termos, procurando dar um maior peso às palavras encontradas no título do documento e na seção de palavras-chave do artigo;

- Adaptar o algoritmo para fazer a filtragem baseada apenas no 'resumo' do artigo ao invés do documento inteiro, objetivando uma redução significativa do esforço computacional;

- Inserção de termos específicos na expressão de busca, no intuito de captar visões diferentes de interesse para um mesmo assunto;

- Ao invés de avaliar o *fitness* do melhor documento classificado pelo Google, adaptar o algoritmo genético para avaliar, por exemplo, os cinco melhores documentos classificados;

- Adaptar o sistema evolutivo a direcionar sua busca ao Google Acadêmico (Scholar Google), procurando maximizar a qualidade dos artigos recuperados;

- Alterar a medida de precisão (*precision*) dando pesos diferentes aos documentos recuperados de acordo, por exemplo, com a origem do documento (p. ex., bibliotecas do IEEE e ACM, CiteSeer, etc); tipo de documento (*proceedings*, *surveys*, capítulos de livro, transparências, etc.)

- Alterar o $A^2CA$ para que, ao término das iterações, o algoritmo seja capaz de fazer a segmentação e contagem automática de grupos;

- Usar pilhas de objetos ao invés de um único objeto numa célula do *grid* do $A^2CA$;

- Utilizar métodos de busca local (p. ex., *k-means*) para aumentar a precisão dos grupos encontrados pelas formigas;

- Adaptar o $A^2CA$ para a utilização do algoritmo extrator de termos quando utilizado no agrupamento de bases textuais, reduzindo assim a dimensionalidade de dados;

- Analisar outros algoritmos de agrupamento de textos e procurar introduzir no $A^2CA$, medidas que minimizem a distância intra-cluster e maximizem as distâncias inter-clusters; e

- Desenvolver efetivamente a comunidade virtual acadêmica adaptativa.

## Referências


Abbas, A.; Lichtman,H. *Imunologia Celular e Molecular*, 5ª edição, Elsevier, 2005.

Abraham, A.; Ramos, V. (2003). *Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming*. Proc. of the Congress on Evolutionary Computation (CEC 2003), Canberra, pp. 1384-1391, IEEE Press.

AntColony *Fungus-growing gants,* disponível em http://www.antcolony.org/leafcutter/leafcuttermain.htm. Acessado em 20/11/2005.

Banzhaf, W; Nordin, P.; Keller, R. E. & Francone, F. D. *Genetic Programming: An Introduction*. Morgan Kaufmann, 1998.

Back,T.; Fogel,D.B.; Michalewicz, Z.; *Evolutionary Computation 1- Basic Algorithm and Operator*, Institute of Physics Publishing, 2000.

Back, T.; Fogel,D.B.; Michalewicz, Z. *Evolutionary Computation 2: Basic Algorithms and Operators,* Institute of Physics Publishing, 2000.

Barrie ,J.M.; Presti,D.E.: *Collaborative Filtering*. Science, vol. 274, pp. 371-372, 1996.

Bautista, M. J. M.; Amparo, M.; Larsen, H.L.;Vila, M. A. *A genetic fuzzy classifier to adaptive user interest profiles with feature selection,* Proc. of the European Society for Fuzzy Logic and Technology,( ESTYLF-EUSFLAT'99) , pp. 327-330, Palma de Mallorca(, Spain 1999.

Belkin, N.J.; Croft W.B. *Information Filtering and Information Retrieval: Two Sides of the same coin?*, Communications of the ACM, vol 35, n. 12, pp. 29-38 1992.

Bezdek, J.C., (1981). *Pattern Recognition with Fuzzy Objective Function Algorithm*, Plenum Press.

Bonabeau, E.; Dorigo, M.; Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.

Braun, H. *On Solving Traveling Salesman Problems by Genetic Algorithms*, Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, pp. 129-133, 1990.

Burnet, F.M. *The Clonal Selection Theory of Acquired Immunity*, Cambridge University Press, 1959.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraulaz, G. and Bonabeau, E. *Self-Organization in Biological Systems,* Princeton University Press, 2001.

Chrétien, L. *Organization Spatiable du Matériel Provenant de l'excavation du nid chez Messor Barbarus et des Cadavres d'ouvrières chez Lasius niger (Hymenopterae: Formicidae)*, Tese de Doutorado, Univerité Libre de Bruxelles, 1996.

Chu, H.; Rosenthal, M.; *Search Engines for the World Wide Web: A Comparative Study and Evaluation Methodology*, ASIS Annual Conference Proceedings, disponível em http://www.asis.org/annual-96/ElectronicProceedings/chu.html

Cleverdon, C.W.; Mills, J.; and Keen, E.M; *Factors Determining the Performance of Indexing Systems, Volume I - Design, Volume II - Test Results, ASLIB Cranfield Project*, in Readings in Information Retrieval, K. Sparck Jones and P. Willett, Editors. 1997, Morgan Kaufmann: San Francisco.

Cooper, W.S.; *Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems* vol. 19, n. 1, pp. 30-41 Journal of the

American Society for Information Science, 2007 disponível em: http://www3.interscience.wiley.com/cgi-bin/jissue/114214048.

Cooper, W.S.; *Getting Beyond Boole*, Readings in Information Retrieval, Morgan Kaufmann Publishers, 1997.

Corne, D.; Dorigo, M. and Glover, F. *New Ideas in Optimization*, McGraw-Hill, 1999.

Darwin, C. *The Origin of* Species, Gramercy; New edition (1995)

de Castro, L.N. Fundamentals of Natural Computing: Basic Concepts, Algorithms and Applications, CRC Press LLC, 2006.

de Castro, L.N. *Fundamentals of Natural Computing: An Overview*, Physics of Life Reviews, vol. 3, n. 1, pp. 1-35, 2007.

De Jong, K.A. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* Tese de Doutorado, University of Michigan, 1975.

Deneubourg, J.-L.; Aron,S.; Goss, S.; Franks,N.; Sendova-Franks,A.; Detrain,C.; Chrétien,L. *The Dynamic of Collective Sorting: Robot-Like Ant and Ant-Like Robot* In J.A.Meyer and S.W.Wilson (eds), Simulation of Adaptive Behavior: From Animal to Animats, pp. 356-365, Cambridge, MA, MIT Press/Bradford Books, 1991.

Denning, P.J. *Electronic Junk,* Communications of the ACM, vol. 25, n.3, pp.163-165, 1982.

Dorigo, M.; Maniezzo,V. and Colorni, A. *The Ant Colony System: Optimization by a Colony of a Cooperating Agents*, IEEE Trans. on Systems, Man, and Cybernetics - Part B, vol. 26, n. 1, pp.29-41, 1996.

Dorigo, M. and Gambardella, L.M. *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem,* IEEE Trans. on Evolutionary Computation, vol. 1, n. 1, pp. 53-66, 1997a.

Dorigo M. and Gambardella, L.M. *Ant Colonies for Traveling Salesman Problem,* BioSystems, 43, pp.73-81, 1997b.

Dorigo, M. Di Caro, G. and Sampels, M. *Ant Algorithms*, Third International Workshop on Ant Algorithms (ANTS 2002), Lecture Notes in Computer Science, 2463, Springer-Verlag, 2002.

Dorigo, M.; Stutzle, T. *Ant Colony Optimzation*, MIT Press, 2004.

Eiben, A.E.; Smith,J.E.; *Introduction to Evolutionary Computing*. Natural Computing Series Springer-Verlag Berlin Heidelberg, 2003.

Ellis, D.; *The Dilemma of Measurement in Information Retrieval*. Journal of the American Society for Information Science, vol. 7(1), pp.23-36 1996.

Ferneda, E *Recuperação de Informação: Análise sobre a contribuição da Ciência da Computação para a Ciência da Informação* Tese de doutorado defendida na Escola de Comunicação e Artes da Universidade de São Paulo USP 2003.

Fogel, L.J.; Owens,A.J.; Walsh,M.J. *Artificial intelligence through a simulation of evolution.* In: A. Callahan, M. Maxfield, L.J.Fogel, Eds., Biophysics and Cybernetic Systems. pp. 131-156 Spartan, Washington DC 1965.

Fogel, L.J.; Owens,A.J.; Walsh,M.J. *Artificial Intelligence through Simulated Evolution.* Wiley, Chichester, UK, 1966.

Forrest ,S.; Javornik,B., Smith, R.E. and Perelson,A.S. *Using Genetic Algorithms to Explore Pattern Recognition in the Immune System* Evolutionary Computation, 1(3),pp.191-121, 1992.

Fox, C.; *Lexical Analysis and Stoplists* in Information Retrieval Data Structures & Algorithms pp. 102-129, Prentice Hall New Jersey 1992.

Franks, N.R.; and Sendova-Franks, A.B.; *Brood Sorting by Ants: Distributing the Workload Over The Work Surface.* Behavioral Ecology and Sociobiology vol.30 pp.109-123 Springer Berlin / Heidelberg (1992).

Google *About Google Scholar* disponível em http://scholar.google.com/intl/en/scholar/about.html acessado em setembro/2006.

Gutowitz, H. . *Complexity-Seeking Ants*. In Deneubourg and Goss, editors, Proceedings of the Third European Conference on Artificial Life, 1993 disponível em: http://citeseer.ist.psu.edu/gutowitz94complexityseeking.html

Gwizdka, J.; and Chignell, M.; *Towards Information Retrieval Measures for Evaluation of Web Search Engines* Department of Mechanical and Industrial Engineering - University of Toronto disponível em http://anarch.ie.utoronto.ca/~jacekg/pubs/webIR_eval1_99.pdf acessado em agosto/2006.

Hart. E.; Ross, P. *An Immune System Approach to Scheduling in Changing Environments,* Proc. of the Genetic and Evolutionary Computation Conference, pp. 1559-1566 GECCO 1999.

Haykin, S. *Redes Neurais Princípios e Prática* Bookman – Porto Alegre, 2001

Hearst M.A. ; Pedersen,J.O. *Reexamining the cluster hypothesis*: *scatter/gather on retrieval results* In Proc. of SIGIR-96, 19th ACM Int. Conf. on Research and Development in Information Retrieval, pp. 76–84, Zürich, 1996. disponível em http://citeseer.ist.psu.edu/hearst96reexamining. html acessado em março/2005.

Hersh, W.R.; Molnar, A.; *Towards New Measures of Information Retrieval Evaluation.* In Proc. of SIGIR-95. 164-170 1995.

Hersh, W.R.; Pentecost, J.; Hickam, D.; *A Task-Oriented Approach to Information Retrieval Evaluation*. Journal of the American Society for Information Science, vol.47(1), pp.50-56 1996.

Holland, J.H. *Genetic Algorithms and the optimal allocation of trials.* SIAM J. of Computing, 2 pp. 88-105 1973.

Holland, J.H. *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

ISURP *O Sistema Nervoso* disponível em http://www.isurp.com.br/aula/ciencia/Marcio/osistema.htm acessado em 06/2005.

Jerne, N.K. *Towards a Network Theory of the Immune System* Annals of Immunology. (Inst. Pasteur) ,pp. 373-389 1974.

Joshi, A.; Todwal, S. *Evolutionary Machine Learning for Web Mining* TENCON 2003 Bangalore - India October 2003

Kanade, P., Hall, L.O. *Fuzzy ants as a clustering concept*. Proc. of the 22nd International Conference of the North American Fuzzy Information Processing Society (NAFIPS), pp. 227-232 (2003).

Kaufman, L., Rousseeuw, P.J., *Finding Groups in Data – An Introduction to Cluster Analysis*, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons Inc., 1990.

Keim,D.A. *Information Visualization and Visual Data Mining* IEEE Transactions on Visualization and Computer Graphics, vol. 7, number 1 (2002).

Kim,J.; Bentley, P. *The Artificial Immune Model for Network Intrusion Detection* Proc. of the 7th European Conference on Intelligent Techniques and Soft Computing, Aachen, Germany 1999.

Kohonen, T. *The self-organizing map* Proceedings of the Institute of Electrical and Electronics Engineers, n.9, vol.78, pp.1464-1480 1990.

Kohonen,T. ;Hynninen,J.; Kangas,J.; Laaksonen,J. *SOM_PAK: The Self-Organizing Map Program Package***.** Technical Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, FIN-02150 Espoo, Finland, 1996.

Koza, J.R. *Genetic Programming* MIT Press, Cambridge, MA, 1992.

Koza, J.R. *Genetic Programming II* MIT Press, Cambridge, MA, 1994.

Labroche, N., Monmarché, N., Venturini,G. *A new clustering algorithm based on the chemical recognition system of ants*. Proc. of the 15th European Conference on Artificial Intelligence, France, pp. 345-349, IOS Press. (2002).

Lagus, K.; Kaski,S. *Keyword selection method for characterizing text documents maps.* Artificial Neural Networks, vol 7, pp.371-376 1999.

Lawrence,S., *Online or invisible?,* Nature, vol 411, p.521, number 6837 2001.

Lawrence, S, Bollacker K, Giles C.L.: *Indexing and Retrieval of Scientific Literature*. In Proc of the CIKM99, Kansas City, Missouri, November 2-6, pp. 139-146 1999.

Lawrence, S,Giles C.L., Bollacker K: *Digital Libraries and Autonomous Citation* Indexing. *IEEE Computer*, vol. 32, pp. 67-71 1999.

Lebo,H. *The UCLA Internet Report – Surveying the Digital Future,* UCLA Center form Communication Policy http://www.digitalcenter.org/downloads/DigitalFutureReport-Year4-2004.pdf,2004 (acessado em maio/2005).

Lumer, E.D. and Faieta, B. *Diversity and Adaptation in Populations of Clustering Ants*. Proceedings of the Third International Conference On the Simulation of Adaptive Behavior: From Animals to Animats 3: MIT Press, 499-508 (1994).

Matzinger, P. *The Danger Model: A Renewed Sense of Self* Science, 296, pp. 301-305 2002.

Maes,P. *Agents of Change,* Wired Magazine, file 3.04 http://www.wired.com/wired/archieve/3.04/maes.html?pg=1&topic=, 1995. (acessado em maio/2005).

Melo V. V.; Lopes, A. A. *Clustering de Artigos Científicos em uma Ferramenta Inteligente de Apoio a Pesquisa. In: Workshop on Artificial Intelligence (WAI'04)*, Jornadas Chilenas de la Computacion, pp. 1-7, Arica – Chile (2004).

Michalewicz,Z *Genetic Algorithms + Data Structures = Evolution Programs* Springer; 3 edition (November 26, 1998).

Monmarché, N., Slimane, M., Venturini, G., *On Improving Clustering in Numerical Databases with Artificial Ants. Advances in Artificial Life*, D. Floreano, J.D. Nicoud, and F. Mondala Eds., Lecture Notes in Computer Science 1674, pp. 626-635, Springer-Verlag, Berlin(1999).

Nick, Z.Z.; Themis, P. *Web Search Using a Genetic Algorithm.* IEEE Internet Computing vol 5 pp. 18-26, 2001.

Nielsen, Netratings *One in three americans use a search engine,* http://www.nielsen-netratings.com/pr/pr_040223_us.pdf, acessado em julho de 2005.

Nossal, G.J.V. *Life, Death and Immune System* Scientific American, 269 (3), pp. 21-30 1993.

Palazzo, L.A.M. *Comunidades Virtuais Adaptativas: Ontologias, Metadados e Colaboração On-line*, Programa de P&D para a capacitação de pequenos grupos acadêmicos na área de tecnologia da informação - CT-INFO: CNPq 11/2002 – PDPG-TI

PDFBox, *Java PDF Library* www.pdfbox.org acessado em janeiro, 2004.

Porter, M.: *An Algorithm for Suffix Stripping*, Program. (1980), vol. 14, no. 3, pp. 130-138.

Ramos, V., Merelo, J.J.. *Self-Organized Stigmergic Document Maps: Environment as a Mechanism for Context Learning*. In E. Alba, F. Herrera, J.J. Merelo et al. Eds., AEB´2002, First Spanish Conference on Evolutionary and Bio-Inspired Algorithms, pp. 284-293, Spain(2002).

Ramos, V., Muge, F., Pina, P. *Self-Organized Data and Image Retrieval as a Consequence of Inter-Dynamic Synergistic Relationships in Artificial Ant Colonies*. In J. Ruiz-del-Solar, A. Abrahan and M. Köppen Eds., Soft-Computing Systems - Design, Management and Applications, Frontiers in Artificial Intelligence and Applications: IOS Press, v. 87, pp.500-509, Amsterdam(2002).

Rasmussen, M; Karypis, G. *gCLUTO: An Interactive Clustering, Visualization, and Analysis System*. . UMN-CS TR-04-021, 2004.

Rauber,A.; Merkl,D.; Dittenbach,M. *The Growing Hierarchical Self-Organizing Map: Exploratory Analysis of High-Dimensional Data* disponível em: http://www.ifs.tuwien.ac.at/~andi/ghsom/publications.html acessado em julho/2006.

Rechenberg, I. *Cybernetic Solution Path of an Experimental Problem* United Kingdom's Ministry of Aviation, Royal Aircraft Establishment.. Library Translation 1122 UK 1965.

Rheingold, H.: The Virtual Community. Electronic version
disponível em http://www.rheingold.com/vc/book/ acessado em 02/02/2004.

Ritter, H. & Kohonen, T. (1989). *Self-Organizing Semantic Maps*. Biol. Cybern., **61**, pp. 241-254.

Romero, C.; Ventura, S. ; Castro, C. ; Hall, W. *Using Genetic Algorithms for Data Mining in Web-based Educational Hypermedia Systems*
disponível em http://www.lcc.uma.es/~eva/WASWBE/romero.pdf acessado em agosto, 2004.

Salton, G. & Bucckley, C. *Term-weighting approaches in Automatic Retrieval*, Information Processing & Management, vol. 24, No 5,pp.513-523, 1988.

Salton, G.; McGill, M. J. *The SMART and SIRE Experimental Retrieval Systems*. In: Readings in Information Retrieval, pp.381-399 San Francisco –CA : Morgan Kaufmann Publishers, 1997**.**

Shafi,S.M.; Rather,R.A.; *Precision and Recall of Five Search Engines for Retrieval of Scholarly Information in the Field of Biotechnology* Webology , vol. 2, n. 2, August, 2005.

Schwefel, H.-P. *Evolution and Optimum Seeking.* Wiley, New York 1995.

Sherafat, V., de Castro, L. N. & Hruschka, E. R. *TermitAnt: An Ant Clustering Algorithm Improved by Ideas from Termite Colonies.* In Proc. of ICONIP 2004, Special Session on Ant Colony and Multi-Agent Systems (2004a).

Sherafat, V., de Castro, L. N. & Hruschka, E. R. *The Influence of Pheromone and Adaptive Vision on the Standard Ant Clustering Algorithm.* In: L. N. de Castro and F. J. Von Zuben, *Recent Developments in Biologically Inspired Computing*, Chapter IX, pp. 207-234. Idea Group Inc(2004b).

Sheth, B.D., *A Learning Approach to Personalized Information Filtering,* Master of Science in Computer Science and Engineering, MIT, 1994.

Storb, U. *Progress in Understanding the Mechanisms and Consequences of Somatic Hypermutation* Immun. Rev. 162, pp. 5-11 1998.

Tague-Sutcliffe, J.M; *Some Perspectives on the Evaluation of Information Retrieval*. Journal of the American Society for Information Science, 47(1), pp.1-3 1996.

Van Rijsbergen, C. J. *Information Retrieval*, Butterworths, 2nd edition, 1979.

Valim, M.S.; Coello, J.M.A. *An Agent for Web Information Dissemination Based on a Genetic Algorithm* In: IEEE International Conference on Systems, Man and Cybernetics- IEEE SMC'03, 2003, Washington, D.C., USA Proc. IEEE International Conference on Systems, Man and Cybernetics- IEEE SMC'03. IEEE Press, pp..3834 – 3839, 2003.

Vizine, A.L.P., *O Uso da Tecnologia de Agentes para a Filtragem e Classificação Automática de Mensagens Eletrônicas*, Dissertação de Mestrado defendida na Pontifícia Universidade Católica de Campinas, 2000.

Vizine, A. L., de Castro, L. N., Gudwin, R. R. *Text Document Classification using Swarm Intelligence.* In Proc. of KIMAS 2005a.

Vizine, A. L.; de Castro, L. N.; Gudwin, R ,R.. *A Proposal for an Intelligent Academic Virtual Community*. In: IADIS International Conference on Web Based Communities, Algarve. Proceedings do IADIS WBC , 2005b.

Von Zuben F. J.; de Castro, L.N.; *Regressão Paramétrica e Não-Paramétrica / Redes Neurais com Funções de Ativação de Base Radial* tópico 8 do curso de redes neurais artificiais ministrado em 2003 disponível em: ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_03/topico8_03.pdf

Wright, S. *The roles of mutation, inbreeding, crossbreeding, and selection.* In: Proc. of 6*th* Int. Congr. on Genetics, Vol. 1. Ithaca, NY, 1932, pp. 356-366.

Yeung, K.Y., Medvedovic, M., Bumgarner, R.E., *Clustering gene-expression data with repeated measurements*, Genome Biology, v.4, issue 5, article R34, 2003.

Yuqiang, G. *Software Gmeans* disponível em http://www.cs.utexas.edu/users/yguan acessado em julho 2006.

Zamir,O.; Etzioni,O. *Web document clustering: A feasibility demonstration*. In *Research and Development in Information Retrieval*, pages 46–54. 1998 disponível em http://citeseer.ist.psu.edu/zamir98web.html acessado em maio 2005.

Zhao, Y.; Karypis, G. *Hierarchical Clustering Algorithms for Documents Datasets* Data Mining and Knowledge Discovery, vol 10, n. 2 pp. 141 – 168 2005

Zhao, Y; Karypis, G. *Evaluation of Hierarchical Clustering Algorithms for Document Datasets* 11th Conference of Information and Knowledge Management (CIKM), pp. 515-524, 2002

Zhao, Y; Karypis, G. *Criterion Functions for Document Clustering: Experiments and Analysis* disponível em http://glaros.dtc.umn.edu/gkhome/node/165 acessado em julho/2006.

# Anexo I

## Palavras irrelevantes (*stopwords*)

about
above
across
after
again
against
all
almost
alone
along
already
also
although
always
among
an
and
another
any
anybody
anyone
anything
anywhere
are
area
areas
around
as
ask
asked
asking
asks
at
away
back
backed
backing
backs
be
because
became
become
becomes
been
before
began
behind
being
beings
best
better
between
big
both
but
by
came
can
cannot
case
cases
certain
certainly
clear
clearly
come
coming
could
did
differ
different
differently
do
does
domain
domains
done
down
downed
downing
downs
during
each
early
either
enc
end
ended
ending
ends
enough
even
envenly
ever
every
everybody
everyone
everything
everywhere
face
faces
fact
facts
far
felt
few
find
finds
first
for
four
from
free
full
fully
further
furthered
furthering
furthers
gave
general
generally
get
gets
give
given
gives
go
going
good
goods
got
great
greater
greatest
group
grouped
grouping
groups
had
has
have
having
he
her
herself
here
high
higher
highest
him
himself
his
how
however
if
important
in
information
informations

interest
interested
interesting
interests
into
is
it
its
itself
just
keep
keeps
kind
knew
know
known
knows
large
largely
last
later
latest
least
less
let
lets
like
likely
listing
long
longer
longest
made
make
making
man
many
may
me
member
members
men
might
more
most
mostly
much
must
my
myself
necessary
need
needed
needing
needs
never
new
newer
newest
next
net
networks
network
no
non
not
nobody
noone
nothing
now
nowhere
number
numbered
numbering
numbers
of
off
office
often
old
older
oldest
on
once
one
only
open
opened
opening
opens
or
order
ordered
ordering
orders
other
others
our
out
over
part
parted
parting
parts
per
perhaps
place
places
please
point
pointed
pointing
points
possible
present
presented
presenting
presents
problem
problems
program
put
puts
quite
rather
really
res
right
room
rooms
said
same
saw
say
says
second
seconds
see
seem
seemed
seeming
seems
sees
several
service
services
shall
she
should
show
showed
showing
shows
side
sides
since
small
smaller
smallest
so
some
somebody
someone
something
somewhere
state
states
still
such
sure
take
taken
than
that
the
their
them
then
there
therefore
these
they
thing
things
think
thinks
this
those

though
thought
thoughts
three
through
thus
to
today
together
too
took
toward
turn
turned
turning
turns
type
two
under
until
up
upon
us
use
uses
used

very
visit
want
wanted
wanting
wants
was
way
ways
we
well
wells
went
were
what
when
where
whether
which
while
who
whole
whose
why
will
with
within
without
work
worked
working
works
would
year
years
yet
you
young
younger
youngest
your
yours

Anexo II

Documentos recuperados para análise de precisão por área de busca.

**Computação Evolutiva**

*An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms*
url: http://www.cs.uwyo.edu/~wspears/papers/ppsn90.pdf

*Algorithm Evolution for Face Recognition: What Makes a Picture Difficult*
url: http://www.cs.cmu.edu/~mmv/papers/tellerICEC95.pdf

*Content Planner Construction via Evolutionary Algorithms and a Corpus-based Fitness Function*
url: http://www.cs.columbia.edu/~pablo/publications/INLG2002genetic-slides.pdf

*Correlation Based Search Algorithms for Motion Estimation*
url: http://amp.ece.cmu.edu/Publication/Deepak/PCS1999.pdf

*Effects of Intra-gene Fitness Interactions on the Benefit of Sexual Recombination (irrelevante)*
url: http://eprints.ecs.soton.ac.uk/12880/01/bio_chem_soc_trans_watson_2006_preprint.pdf

*Effects of Intra-gene Fitness Interactions  the Benefit of Sexual Recombination (irrelevante)*
url:
http://www.oeb.harvard.edu/faculty/wakeley/John/documents/bio_chem_soc_trans_watson_2006_preprint.pdf

*Evolutionary Computation*
url: http://www.sci.unich.it/~aroli/dida/iasc/lucidi0506/evocomp.4perpage.pdf

*Genetic Algorithms and their Application to Continuum Generation*
url: http://www.physics.ohio-state.edu/~reu/02reu/REU2001reports/tmoorepaperout.final.pdf

*Genetic Algorithms Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully Understand*
url: http://www.sci.unich.it/~aroli/dida/iasc/articoli/holland.GAIntro.pdf

*Genetic Algorithm Design of Two-Way Crossover Circuits*
url: http://www.cse.ucsc.edu/classes/cmps290a/Winter00/lilly_fp_pres.pdf

*Genetic Algorithm Evolution of Cellular Automata Rules for Complex Binary Sequence Prediction*
url: http://www.math.upatras.gr/~npav/papers/AdamopoulosVP_2005.pdf

*How the immune system generates diversity: Pathogen space coverage with random and evolved antibody libraries (irrelevante)*
url: http://www.cs.unm.edu/~forrest/publications/gecco-mihaela.pdf

*Hybrid Particle Swarm – Evolutionary Algorithm for Search and Optimization(*2 ocorrências*)*
url: http://www.gelbukh.com/lab/Publications/2005/MICAI-2005-PSO.pdf

*Intelligent Control - Presentation Part 7*
url: http://www.researchcentre.apsc.ubc.ca/MECH523/7-mech-523-presentation.pdf

*Machine Learning - Genetic Algorithms*
url:
http://users.wmin.ac.uk/~dracopd/DOCUM/courses/2ait608/genetic_algorithms_lecture_notes.pdf

*Model Generation for an Intrusion Detection System Using Genetic Algorithms*
url: http://www1.cs.columbia.edu/ids/publications/gaids-thesis01.pdf

*On Random Numbers and the Performance of Genetic Algorithms*
url: http://www.llnl.gov/CASC/sapphire/pubs/146850.pdf

*Optimal Design of Matched Load by Immune Micro Genetic Algorithm*
url: http://piers.mit.edu/piersonline/vol1/2k5hz_p101.pdf

*Parallel Global Optimization with the Particle Swarm Algorithm*
url: http://www.mat.univie.ac.at/~neum/glopt/mss/SchRF03.pdf

*Part II: Mechanisms of Evolutionary Change Case Studies in Evolution - SELECTION AND MUTATION AS MECHANISMS OF EVOLUTION (irrelevante)*
url: http://faculty.washington.edu/herronjc/SoftwareFolder/Files/Tutorial1.SelectionMutation.pdf

*Robot Gaits Evolved by Combining Genetic Algorithms and Binary Hill Climbing*
url: http://heim.ifi.uio.no/~matsh/591188/Henriette/Paper3.pdf

*Selection*
url: http://ww3.algorithmdesign.net/handouts/Selection.pdf

*Slides sobre Algoritmos Genéticos*
url: http://cs.wlu.edu/~levy/courses/cs397-ga-02/lectures/09_SEP_2002.pdf

*Spatial effects in discrete generation population models*
url: http://www.math.utah.edu/~fife/agg1.pdf

*Spread Spectrum Code Estimation by Particle Swarm Algorithm*
url: http://www.enformatika.org/data/v7/v7-40.pdf

*The Genetic Algorithm in Computer Science*
url: http://www-math.mit.edu/phase2/UJM/vol1/PREBYS-F.PDF

*The LifeCycle model: Combining Particle Swarm Optimisation, Genetic Algorithms and HillClimbers*
url: http://www.evalife.dk/publications/TK_PPSN2002_LifeCycle_PSO_HC_GA.pdf

*The Role of Selective Neutrality in Evolutionary Search*
url: http://www.cogs.susx.ac.uk/users/eji21/master/Neutrality.pdf

*Theory of Population and Evolutionary Ecology(irrelevante)*
url: http://rydberg.biology.colostate.edu/CourseCatalog/syllabus/FA06_syllabus_BZ348.pdf

*Understanding EA Dynamics via Population Fitness Distributions*
url: http://cs.gmu.edu/~epopovic/gecco03-poster-paper.pdf

*Y chromosome polymorphism is a strong determinant of male fitness in Drosophila melanogaster(irrelevante)*
url: http://www.pnas.org/cgi/reprint/98/10/5677.pdf

**Redes Neurais Artificiais**

*AI Handwriting Recognition*
url: http://www.seas.upenn.edu/~cse400/CSE401_2006/Ziller/06writeup_revision2.pdf

*A COMPARISON OF THREE NEURAL NETWORK ARCHITECTURES FOR AUTOMATIC SPEECH RECOGNITION*
url:
http://www.ece.odu.edu/~zahorian/pdf/A%20COMPARISON%20OF%20THREE%20NEURAL%20NETWORK%20ARCHITECTURES%20FOR.pdf

*A neural model of how the brain represents and compares multi-digit numbers: spatial and categorical processes*
url: http://www.cns.bu.edu/Profiles/Grossberg/GroRep2003NN.pdf

*A Support Vector Method for Multivariate Performance Measures (irrelevante)*
url: http://www.cs.cornell.edu/people/tj/publications/joachims_05a.pdf

*A Practical Guide to Support Vector Classification (irrelevante)*
url: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

*An Introduction to Kernel-Based Learning Algorithms*
url: http://www.ist.temple.edu/~vucetic/cis526fall2003/SVMintro.pdf

*An Unbiased Clustering Algorithm based on Self-organisation Processes in Spiking Networks*

url: http://www.ini.unizh.ch/~tott/ndes06tott.pdf

*Adaptive Training of Radial Basis Function Networks Based on Cooperative Evolution and Evolutionary Programming*
url: http://www.msu.edu/~topchyal/rbfnnz.pdf

*Create Classification Training Data*
url: http://www.microimages.com/documentation/cplates/59classtraindata.pdf

*Drift Data Modeling of Laser Gyro Based on Neural Networks*
url: http://ej.iop.org/links/r32afkefL/Jq1N4fOf2xG3urF4av5vpA/jpconf6_48_023.pdf

*Dynamic topology adjustment algorithm for MLP networks*
url: http://www.ire.pw.edu.pl/~rsulej/NetMaker/icaisc/icaisc_poster.pdf

*Efficient Neural Network Training Using Subsets of Very Large Datasets*
url: http://www.public.asu.edu/~svadrevu/UMD/ThesisTalk.pdf

*EE E6820: Speech & Audio Processing & Recognition*
url: http://www.ee.columbia.edu/~dpwe/e6820/lectures/E6820-L03-patrecog.pdf

*Evolutionary Training of Hybrid Systems of Recurrent Neural Networks and Hidden Markov Models*
url: http://www.enformatika.org/data/v15/v15-12.pdf

*IMAGE INTERPOLATION USING FEEDFORWARD NEURAL NETWORK*
url: http://emerald.is.tsukuba.ac.jp/kameyama/common/publications/aokage-aia05.pdf

*Large Mesh Simplification using Processing Sequences*
url: http://www.cc.gatech.edu/~lindstro/papers/vis2003/paper.pdf

*Machine Learning*
url: http://www.comp.lancs.ac.uk/computing/users/dixa/teaching/MScHCI/notes-2004/Machine-Learning.pdf

*Multi-layer Perceptron: nagdmc mlp*
url: http://www.nag-j.co.jp/pdf/nagdmc_mlp.pdf

*Neural Networks*
url: http://www.kscape.com/downloads/neural%20networks.pdf

*Neurotic About Hidden Neurons*
url: http://www.usc.edu/CSSF/History/2004/Projects/S1216.pdf

*5.3. On neural control of a manipulator with output feedback*
url: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/iel5/8393/26435/01177084.pdf

*PET-MRI IMAGE REGISTRATION AND FUSION USING ARTIFICIAL NEURAL NETWORKS*
url:
http://www.press.ntu.edu.tw/ejournal/Files/%C2%E5%BE%C7%A4u%B5%7B/200306/2.pdf

*Spatiotemporal activity patterns of rat cortical neurons predict responses in a conditioned task (irrelevante)*
url: http://www.pnas.org/cgi/reprint/96/3/1106.pdf

*The Minimum Number of Hidden Neurons Does Not Necessarily Provide the Best Generalization*
url: http://binf.gmu.edu/kinser/pubs/MinNum.pdf

*The Neural Network Objects*
url: http://www-root.fnal.gov/root2001/presentations/session6/RhoNNO.pdf

*Two important (and simple) concepts in computational neuroscience (irrelevante)*
url: http://pt.svt.ntnu.no/artikler/pt-biegler.pdf

**Sistemas Fuzzy**

*Data-Driven Linguistic Modeling Using Relational Fuzzy Rules*
url: http://ci.uofl.edu/zurada/publications/gaweda.tfs.2003.pdf

*Fuzzy Approximation as a Theory Stemming from Fuzzy IF-THEN Rules*

url: http://www-bisc.cs.berkeley.edu/BISCSE2005/Abstracts_Proceeding/Saturday/SM2/Berkeley05IP.pdf

*Fuzzy Control*
url: http://www.ece.osu.edu/~passino/FCbook.pdf

*Fuzzy Data Analysis: Challenges and Perspectives*
url: http://fuzzy.cs.uni-magdeburg.de/~borgelt/papers/fieee_99.pdf

*Fuzzy Logic Control with the Intel 8XC196 Embedded Microcontroller*
url: http://www.intel.com/design/mcs96/papers/esc_196.pdf

*Fuzzy Logic Introduction(4 ocorrências)*
url: http://www.fpk.tu-berlin.de/~anderl/epsilon/fuzzyintro4.pdf

*Fuzzy Logic Toolbox 2.1Design and simulate fuzzy logic systems*
url: http://www.mathworks.com/academia/student_version/r14sp1_products/fuzzylogic/fl.pdf

*Fuzzy sets and Fuzzy Techniques*
url: http://www.teknat.uu.se/internt/forskarutbildning/handbok/kurs/FuzzySetsTechniques.pdf

*Hierarchical Neuro-Fuzzy BSP Model – HNFB*
url: http://csdl2.computer.org/comp/proceedings/sbrn/2000/0856/00/08560286.pdf

*INDUSTRIAL FUZZY CONTROLLER*
url: http://free.bol.bg/ditchko/Fubest.PDF

*Introduction to Fuzzy Control(2 ocorrências)*
url: http://www.personal.reading.ac.uk/~sis01xh/teaching/ComputerControl/fcslide1.pdf

*Learning Methods for Fuzzy Systems*
url: http://adiret.cs.uni-magdeburg.de/publications/kruNuern98.pdf

*Machine Learning (irrelevante)*
url: http://www.comp.lancs.ac.uk/computing/users/dixa/teaching/MScHCI/notes-2004/Machine-Learning.pdf

*Modeling Fuzzy Rules with Description Logics*
url: http://www.mindswap.org/2005/OWLWorkshop/sub5.pdf

*On Generating FC Fuzzy Rule Systems from Data Using Evolution Strategies (2 ocorrências)*
url: http://www.soft-computing.de/IEEE-SMC99.pdf

*Optimization under fuzzy linguistic rule constraints*
url: http://www.abo.fi/~rfuller/afuse99.pdf

*Simulation of Traffic Flow Regulated by a Fuzzy Logic Controller*
url: http://devils.eng.fsu.edu/download/HSC03.abstract.pdf