UNIVERSIDADE PRESBITERIANA MACKENZIE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Ana Karina Fontes Prior

ENXAME DE PARTÍCULAS APLICADO AO AGRUPAMENTO DE TEXTOS

São Paulo
2010

UNIVERSIDADE PRESBITERIANA MACKENZIE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Ana Karina Fontes Prior

ENXAME DE PARTÍCULAS APLICADO AO AGRUPAMENTO DE TEXTOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Presbiteriana Mackenzie, como requisito para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Leandro Nunes de Castro

São Paulo
2010

P958e Prior, Ana Karina Fontes.
Enxame de partículas aplicado ao agrupamento de textos / Ana Karina Fontes Prior – 2010.
60 f. : il. ; 30 cm

Dissertação (Mestrado em Engenharia Elétrica) – Universidade Presbiteriana Mackenzie, São Paulo, 2010.
Bibliografia: f. 56-60.

1. Enxame de partículas. 2. Mineração de textos. 3. Mineração de dados. 4. Agrupamento de textos. 5. Agrupamento de dados. I. Título.

CDD 006.3

UNIVERSIDADE PRESBITERIANA MACKENZIE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Ana Karina Fontes Prior

ENXAME DE PARTÍCULAS APLICADO AO AGRUPAMENTO DE TEXTOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Presbiteriana Mackenzie, como requisito para a obtenção do título de Mestre em Engenharia Elétrica.

BANCA EXAMINADORA

Prof. Dr. Leandro Nunes de Castro
Universidade Presbiteriana Mackenzie

Prof. Dr. Leandro Augusto da Silva
Universidade Presbiteriana Mackenzie

Prof. Dr. Marco Antônio Garcia de Carvalho
UNICAMP - Limeira

São Paulo
2010

AGRADECIMENTOS

Agradeço ao meu orientador e amigo, Prof. Dr. Leandro Nunes de Castro, que conduziu minha formação acadêmica, por todas as suas contribuições, pelo incentivo a pesquisa, pela sua confiança e oportunidades oferecidas ao longo desses anos.

Aos meus pais Maria Virgina e Manuel, por todos os ensinamentos, amor, apoio, incentivo e por tudo que me proporcionaram durante toda minha vida.

Ao meu namorado Caio, que de forma especial e carinhosa, esteve sempre presente me incentivando e apoiando nos momentos mais difíceis.

Ao amigo Alexandre Szabo, por todas as suas contribuições, discussões, pela amizade e companhia durante o desenvolvimento deste trabalho.

Aos amigos da NatComp pela amizade, paciência e incentivo durante este último ano.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela concessão da bolsa e ao Mackpesquisa pelo apoio financeiro que possibilitaram o desenvolvimento deste trabalho.

À Universidade Presbiteriana Mackenzie (UPM) e ao Programa de Pós Graduação em Engenharia Elétrica pela infraestrutura e suporte.

A todos os professores e colegas do Programa de Pós Graduação em Engenharia Elétrica que contribuíram para minha formação e para este trabalho.

A todos aqueles que contribuíram direta ou indiretamente para a conclusão deste trabalho, meus sinceros agradecimentos.

RESUMO

A grande quantidade de dados gerados por pessoas e organizações tem estimulado a pesquisa sobre métodos efetivos e automáticos de extração de conhecimentos a partir de bases de dados. Essa dissertação propõe duas novas técnicas bioinspiradas, denominadas cPSC e oPSC, baseadas no algoritmo de otimização por enxame de partículas (PSO - *Particle Swarm Optimization*) para resolver problemas de agrupamento de dados. Os algoritmos propostos são aplicados a problemas de agrupamento de dados e textos, e seus desempenhos são comparados com outros propostos na literatura específica. Os resultados obtidos nos permitem concluir que os algoritmos propostos são competitivos com aqueles já disponíveis na literatura, porém trazem outros benefícios como a determinação automática do número de grupos nas bases e a efetuação de uma busca pelo melhor particionamento possível da base considerando uma função de custo explícita.

Palavras-chave: *Enxame de partículas, mineração de textos, mineração de dados, agrupamento de textos, agrupamento de dados.*

ABSTRACT

The large number of data generated by people and organizations has stimulated the research on effective and automatic methods of knowledge extraction from databases. This dissertation proposes two new bioinspired techniques, named cPSC and oPSC, based on the Particle Swarm Optimization Algorithm (PSO) to solve data clustering problems. The proposed algorithms are applied to data and text clustering problems and their performances are compared with a standard algorithm from the literature. The results allow us to conclude that the proposed algorithms are competitive with those already available in literature, but bring benefits such as automatic determination of the number of groups on the dataset and a search for the best partitioning of the dataset considering an explicit cost function.

Keywords*:* *Particle swarms, text mining, data mining, clustering.*

Lista de Figuras

Lista de Tabelas

SUMÁRIO

1 INTRODUÇÃO

Grande parte da informação armazenada na *web* encontra-se em forma de documentos textuais. Como a *web* tem crescido em dimensão e diversidade, ela adquiriu um imenso valor como um repositório de conhecimento ativo e em evolução (Chakrabarti, 2002). Frequentemente usuários da Internet lidam com o problema da "sobrecarga de informações", que ocorre quando estes têm muitas informações ao seu alcance, mas não têm condições de tratá-las ou de encontrar o que realmente desejam ou lhes interessam (Morais, 2007). Considerando a grande quantidade de informação disponível torna-se necessário o desenvolvimento de técnicas eficientes e eficazes que possibilitem a extração automática de informações úteis da *web*.

Mineração de Textos (Weiss et al., 2005; Hotho et al., 2005) é um segmento dentro da área de *Mineração de Dados* (Han e Kamber, 2000) que vem ganhando um grande destaque ultimamente pela sua relevância teórica e prática. A *Mineração de Textos* é um processo que surgiu a partir da necessidade de se extrair, de forma automática, conhecimento a partir de textos. O uso dessas técnicas permite recuperar informações, extrair dados, resumir documentos, descobrir padrões, associações e regras, e realizar análises qualitativas ou quantitativas em documentos textuais (Aranha e Passos, 2006). O processo de *Mineração de Textos* tem como objetivo a busca de informações relevantes e a descoberta de conhecimentos significativos a partir de documentos textuais (Monteiro et al., 2006). Este processo envolve um grau de dificuldade significativo considerando que as informações normalmente estão disponíveis em linguagem natural, sem a preocupação com a padronização ou com a estruturação dos dados (Monteiro et al., 2006).

A *Mineração de Textos* serve como solução para diversos problemas, como, por exemplo, classificação de documentos, recuperação de informação, recomendação e agrupamento de documentos, sendo este o foco desta dissertação. Esta dissertação propõe duas adaptações de um algoritmo de otimização, denominado algoritmo de *otimização por enxame de partículas* (PSO – do inglês *Particle Swarm Optimization*) (Kennedy e Eberhart, 1995), para aplicação em tarefas de agrupamento de textos.

1.1 MOTIVAÇÃO

O presente projeto de pesquisa investiga o uso de algoritmos bioinspirados no contexto de agrupamento de textos. Textos fazem parte da categoria de dados conhecidos como não estruturados. Eles são considerados não estruturados, pois não possuem uma forma de composição padronizada, podendo estar disponíveis em diversos formatos, como arquivos *pdf*, *doc*, *xml*, *html*, planilhas, bancos de dados, e até mesmo em imagens. Textos podem estar formatados de maneiras diferentes, como, por exemplo, texto, texto com imagens, texto com tags, etc.

Sob o ponto de vista da aplicação, sabe-se que trabalhar com dados não estruturados, não é uma tarefa trivial. O crescente volume de dados na *web* tem motivado o desenvolvimento de ferramentas computacionais capazes de analisar tal volume e extrair conhecimento dos mesmos, de modo automático e inteligente. Métodos de agrupamento são muito importantes, pois eles encontram e segmentam, dentro de uma base de dados, documentos que possuem características comuns.

Considerando o desenvolvimento das ferramentas de solução do problema, o estudo de algoritmos bioinspirados é motivado pela atualidade da área, pela possibilidade de aplicar os mesmos a problemas complexos e também pela possibilidade de determinar resultados diferenciados. A bioinspiração tem mostrado bons resultados no tratamento de problemas complexos e é uma área em franca expansão (Freitas, 2008, de Castro, 2007).

Portanto, esta dissertação é relevante tanto sob o ponto de vista do desenvolvimento de algoritmos bioinspirados, como sob a perspectiva do problema a ser abordado.

1.2 OBJETIVOS

Considerando a pesquisa e desenvolvimento de ferramentas bioinspiradas para agrupamento de texto, esta dissertação tem três objetivos principais:

1. Propor duas versões de um algoritmo bioinspirado usualmente aplicado em tarefas de otimização, de forma que ele possa trabalhar com agrupamento de textos;
2. Avaliar o desempenho destas ferramentas em relação à qualidade dos

agrupamentos gerados; e

3. Comparar o desempenho destas ferramentas com o algoritmo das *K-Médias*.

1.3 CONTRIBUIÇÕES DA DISSERTAÇÃO

Essa dissertação propõe uma versão construtiva do algoritmo de agrupamento por enxame de partículas, denominada cPSC, que apresenta a vantagem de determinar automaticamente o número de partículas no enxame e, consequentemente, a quantidade de grupos naturais em uma base de dados.

Além disso, essa dissertação também propõe uma versão do algoritmo PSO para agrupamento capaz de propor agrupamentos ótimos de uma base de dados considerando, para isso, funções de custo explícitas que determinam a distância intra- e/ou intergrupo de cada solução candidata.

1.4 ORGANIZAÇÃO DA DISSERTAÇÃO

Essa dissertação está organizada da seguinte forma. O Capítulo 2 contem todo o referencial teórico desta dissertação, introduzindo os conceitos de *Mineração de Dados*, *Mineração de Textos* e Inteligência de Enxame; também são introduzidos os algoritmos PSO (Otimização por Enxame de Partículas) e PSC (Agrupamento por Enxame de Partículas) utilizados como base para a proposta dos novos algoritmos. No Capítulo 3 são introduzidas as contribuições da dissertação em termos de algoritmos bioinspirados: o algoritmo cPSC que é um algoritmo de enxame construtivo para agrupamento de textos e o algoritmo oPSC que é um algoritmo para agrupamento ótimo de dados. No Capítulo 4 são descritos os materiais e métodos utilizados nos experimentos e são detalhados os experimentos realizados, seguidos por discussões sobre os mesmos. O Capítulo 5 contem as conclusões e trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 MINERAÇÃO DE DADOS

Com o avanço da tecnologia, do poder de processamento e da capacidade de armazenamento de dados, as organizações têm gerado uma grande quantidade de dados, obtidos continuamente. Apesar disso, algumas organizações não sabem como extrair informações úteis e compreensíveis dessas bases de dados. É necessária a utilização de técnicas que transformem essa grande quantidade de dados em informações úteis que representem conhecimento. Com este intuito, no início da década de 1990, surgiu a *Mineração de Dados* (Han e Kamber, 2000).

O termo *Mineração de Dados* (do inglês *data mining*) surgiu do interesse em utilizar grandes bancos de dados de uma maneira inteligente. A ideia é descobrir conhecimento em grandes conjuntos de dados, que corresponde a uma tarefa significativamente difícil e desafiadora, requerendo ativa participação de engenheiros de conhecimento, analistas de sistemas, analistas de dados, especialistas do domínio, usuários do sistema, estatísticos, etc. (Witten e Frank, 2005; Han e Kamber, 2000).

Dentre os problemas do dia a dia aos quais a *Mineração de Dados* pode ser aplicada destacam-se: análise de crédito, predição de pagamento de empréstimos, classificação e agrupamento de clientes de uma empresa, prevenção a lavagem de dinheiro, *Mineração de Dados* da *web*, dentre outros.

2.1.1 Agrupamento de Dados

Técnicas de agrupamento de dados têm como objetivo particionar um conjunto de dados em grupos de acordo com alguma similaridade, de modo que objetos de um mesmo grupo sejam similares entre si e dissimilares com relação aos objetos dos demais grupos.

Diferente da classificação, o agrupamento considera dados não rotulados, ou seja, não se conhece a priori as classes dos dados que estão sendo apresentados. O processo de agrupamento ou *clusterização* é normalmente usado para identificar tais classes (Everitt et al., 2001).

Existem diversas técnicas de agrupamento de dados disponíveis na literatura. Uma das técnicas mais conhecidas e utilizadas é o algoritmo das *K-Médias*, que é utilizado nessa dissertação para comparação com os algoritmos propostos.

2.1.2 K-Médias

O algoritmo das *K-Médias* é uma das mais simples, porém efetivas, técnicas de aprendizado não supervisionado para a resolução de problemas de agrupamento (Macqueen, 1967; Witten e Frank, 2005). Esse algoritmo é muito utilizado na área de *Mineração de Dados*, telecomunicações e também na área de processamento de imagens.

No *K-Médias* é escolhido previamente um número *k* de centroides (que corresponde ao número de representantes de grupos que se deseja encontrar), que são recalculados diversas vezes até que a geração dos grupos em torno dos centroides não apresente mudanças.

Para avaliar a similaridade entre os centroides e os vetores no espaço normalmente é utilizada a distância Euclidiana, uma das mais populares na literatura, dada por:

$$\mathbf{d}(\boldsymbol{i},\boldsymbol{j}) = \sqrt{|\mathbf{x}_{\boldsymbol{i1}} - \mathbf{x}_{\boldsymbol{j1}}|^{\mathbf{2}} + |\mathbf{x}_{\boldsymbol{i2}} - \mathbf{x}_{\boldsymbol{j2}}|^{\mathbf{2}} + \cdots + |\mathbf{x}_{\boldsymbol{ip}} - \mathbf{x}_{\boldsymbol{jp}}|^{\mathbf{2}}}, \quad \textbf{(2.1)}$$

na qual $\mathbf{x}_i$ e $\mathbf{x}_j$ são dois vetores (objetos ou partículas) quaisquer de dimensão *p*.

O algoritmo das *K-Médias* segue os seguintes passos:

1. Escolher os k centroides ou pontos do espaço.
2. Calcular a similaridade entre os centroides e cada ponto do espaço.
3. Recalcular a posição dos centroides no espaço.
4. Repetir os Passos 2 e 3 até que os centroides não tenham seu valor alterado.

Algoritmo 2.1: Algoritmo *K-Médias*

2.2 MINERAÇÃO DE TEXTOS

Mineração de Textos (Weiss et al., 2005; Hotho et al., 2005) é um segmento dentro da área de *Mineração de Dados* que vem ganhando um grande destaque ultimamente pela sua relevância teórica e prática. O processo de *Mineração de Textos* tem como objetivo a busca de informações relevantes e a descoberta de conhecimentos significativos a partir de documentos textuais (Monteiro et al., 2006). Este processo envolve um grau de dificuldade significativo, pois as informações normalmente estão disponíveis em linguagem natural, sem a preocupação com a padronização ou com a estruturação dos dados (Monteiro et al., 2006).

A *Mineração de Textos* surge da necessidade de descobrir, de forma automática e efetiva, informações em dados textuais. O uso dessa técnica permite recuperar informações, extrair dados, resumir documentos, descobrir padrões, associações e regras e realizar análises qualitativas ou quantitativas em documentos de texto (Aranha e Passos, 2006).

Técnicas de *Mineração de Textos* estão presentes em diversos sistemas conhecidos mundialmente, como no mecanismo de busca do Google (www.google.com), nos sistemas de recomendação dedicados como os disponíveis na Amazon (www.amazon.com) e também naqueles oferecidos como serviço (p. ex., www.tuilux.com.br), em sistemas de classificação de *spam* disponíveis na grande maioria de servidores de *e-mail*, dentre outros.

A *Mineração de Textos* serve como solução para diversos problemas, como, por exemplo:

- Classificação de documentos (Berry e Castellanos, 2007);
- Recuperação de informação (Baeza-Yates e Ribeiro-Neto, 1999);
- Agrupamento e organização de documentos (Witten e Frank, 2005); e
- Recomendação (Resnick e Varian, 1997).

O problema de classificação de documentos visa associar documentos a uma classe pré defindа, sendo um exemplo dessa tarefa a distribuição de notícias em uma categoria associada. Considerando as categorias de “Esportes” e “Política”, a tarefa de classificação seria inserir notícias referentes a *futebol* na categoria de “Esportes” e notícias referentes as *eleições* na categoria “Política”.

Recuperação de informação trabalha com o armazenamento e recuperação automática de documentos textuais. Um exemplo de sistema de Recuperação de informação é o Google, que é utilizado diariamente pela grande maioria de usuários da Internet. De acordo com o site *Worldometers* (http://www.worldometers.info/) já existem mais de 1.983.588.450 usuários de Internet no mundo todo e em um único dia foram feitas mais de 2.724.500.000 buscas no Google. Um sistema de busca funciona basicamente da seguinte maneira: algumas palavras são apresentadas através de uma expressão de busca (*query*) e as mesmas são comparadas com todos os documentos disponíveis no banco de dados do sistema. Ao final do processo as melhores "combinações", ou seja, as páginas da *web* mais relevantes são apresentadas como resposta.

No problema de agrupamento e organização de documentos a ideia é particionar um conjunto de documentos em grupos de maneira que documentos do mesmo grupo sejam similares entre si e dissimilares com relação a documentos de outros grupos. A principal diferença entre as tarefas de classificação e agrupamento é que na tarefa de classificação sabe-se a priori as classes em que os documentos são classificados e esse conhecimento é usado para classificar novos objetos com classes desconhecidas, enquanto na tarefa de agrupamento não se sabe a priori o número de grupos nos quais os documentos podem ser agrupados e nem a pertinência deles a cada grupo.

Um sistema de recomendação (Reategui e Cazella, 2003), por sua vez, tem o objetivo de auxiliar um usuário em um processo de escolha, por exemplo, de informação, conteúdo, itens a serem adquiridos ou vistos etc. Um sistema eficiente pode ser um diferencial não só para atrair usuários, mas também para fazer com que estes continuem acessando um *site* continuamente. Os *websites* de comércio eletrônico são atualmente o maior foco de utilização dos sistemas de recomendação, empregando diferentes técnicas para encontrar os produtos mais adequados para seus clientes e aumentar deste modo sua lucratividade (Reategui e Cazella, 2003).

Essa dissertação foca o desenvolvimento de algoritmos de agrupamento de dados numéricos e textuais.

2.2.1 Pré Processamento de Textos

O primeiro passo da *Mineração de Textos* é coletar os documentos em formato adequado. É comum achar documentos em vários formatos diferentes, que podem ser arquivos do Word (.doc), arquivos feitos no bloco de notas (.txt), tabela de banco de dados e até imagens.

Documentos armazenados em um banco de dados, no qual o formato dos elementos da tabela é texto simples, permitem uma recuperação direta. Caso a fonte dos dados seja a Internet e o formato PDF, é necessário utilizar um software de leitura de arquivos PDF e conversão para formato texto. Portanto, cada recuperação é dependente da fonte do documento e de seu formato.

A partir do momento em que os documentos estão disponíveis, é necessário seguir alguns passos para poder lidar com os mesmos.

Com o objetivo de preparar de forma apropriada um conjunto de dados textuais não estruturados devem-se utilizar algumas técnicas de pré-processamento de textos, de forma que ao final do processo obtenha-se um conjunto estruturado de dados que possa ser utilizado posteriormente por algoritmos de agrupamento, classificação, etc. Uma forma de tornar esses dados textuais estruturados é transformá-los em bases numéricas.

A partir da estruturação dos textos é realizada a etapa de análise, especificamente agrupamento no caso dessa dissertação, que tem como objetivo formar grupos de acordo com alguma similaridade entre os documentos (Kauffman e Rousseeuw, 1990; Everitt et al., 2001; Witten e Frank, 2005).

A seguir são listados os principais passos de pré-processamento de textos que serão utilizados neste trabalho:

1. Análise léxica ou *tokenização* do texto;
2. Geração do dicionário, incluindo a eliminação de palavras frequentes, como artigos e preposições (*stopwords*), redução de palavras através da retirada de derivações e plurais (*Stemming*);
3. Seleção de características; e
4. Comparação de documentos.

2.2.1.1 Tokenização

O processo de tokenização quebra uma sequência de caracteres em palavras (*tokens*), para que os processos posteriores como o de retirada de *stopwords*, *stemming*, etc., sejam executados com precisão e qualidade.

A separação do documento em tokens ocorre através da escolha de algumas delimitações como: pontuação (e.g. “;” (ponto e vírgula), “,” (vírgula), “.” (ponto)), caractere especial (e.g. “$”, “%”, “#”) e tags de linguagens de marcação (e.g. “<html>”, “</xml”).

2.2.1.2 Retirada das Stopwords

O processo de retirada de palavras frequentes e/ou irrelevantes é um processo importante na *Mineração de Textos* (Weiss et al., 2005), pois este diminui a quantidade de palavras da matriz e faz com que a frequência das palavras aumente tendo uma melhoria significativa na qualidade da geração do dicionário, visto que as palavras mais usuais e desnecessárias não serão calculadas.

Esse processo é bastante simples, mas requer uma boa lista de palavras frequentes (*stopwords*). A lista é formada basicamente por artigos, preposições e palavras irrelevantes e pode ser escrita para contemplar uma determinada língua ou ainda ser multilíngue, mas sua concepção vai depender integralmente das características e necessidades do processo em que será usada. O mais importante é ressaltar que a lista contenha sempre palavras ou caracteres que não sejam importantes para a comunicação (Weiss et al., 2005) e para o contexto do processo.

Várias listas de palavras frequentes estão disponíveis na Internet e em vários idiomas, como pode ser visto através do seguinte link http://www.ranks.nl/tools/stopwords.html. O melhor caminho é obter uma lista e alterá-la de acordo com a característica do contexto.

2.2.1.3 Stemming

Stemming é o processo de redução de palavras através da retirada de derivações e plurais e unificação dos sinônimos (Chaves, 2004), ou seja, tornar um conjunto de palavras cuja raiz é a mesma, seja por significado ou gramaticalmente, um centro de referência. Este processo aumenta a frequência de ocorrência de algumas palavras e

reduz o número de palavras distintas. Para uma melhor visualização desse processo pode-se citar a palavra “mulheres”. Facilmente identifica-se que a palavra “mulheres” é o plural da palavra “mulher”. Portanto, pode-se unificar a palavra “mulheres” a sua respectiva raiz “mulher”.

O *Stemming*, apesar de parecer um processo fácil, é um processo que requer um estudo gramatical minucioso da língua a ser utilizada e também é dependente da aplicação. A grande importância de seu uso é o fornecimento de padrões para cada palavra, o que favorece a contagem de frequência das palavras no documento.

2.2.1.4 Frequência das palavras

Após a execução dos processos descritos anteriormente, obtem-se uma matriz de documentos, na qual cada linha representa o documento e cada coluna um *token* (palavra), e esta matriz está preparada para o cálculo de frequência das palavras nos documentos.

Na etapa em que calcula-se a frequência das palavras são estipuladas as palavras com maior relevância para cada documento e também em relação ao dicionário como um todo.

Para contabilizar a frequência das palavras, é necessária a geração de uma nova matriz, na qual cada linha continua a representar um documento e cada coluna passa a representar uma palavra do dicionário e não mais somente do documento. Portanto, a reformulação para a criação da nova matriz consiste em:

1. Selecionar todas as palavras da matriz;
2. Retirar todas as repetições;
3. Ordenar as palavras não repetidas;
4. Representá-las de forma abstrata para cada coluna da matriz.

Existem diversas técnicas para o preenchimento da matriz com a ocorrência das palavras para cada documento. No modelo mais básico usa-se 0 (zero) para representar a ausência da palavra no documento ou 1 (um) para representar a presença da palavra no documento.

O cálculo de frequência *tf-idf* (*term frequency - inverse document frequency*) que além de ser uma das técnicas mais conhecidas na literatura, é amplamente utilizada no

cálculo de pesos ou contagem de palavras. A frequência relativa do termo no documento é dada por:

$$tf(i) = \frac{n_i}{\sum_k n_k} \quad \textbf{(2.2)}$$

onde n_i é o número de ocorrências do termo e o denominador é o número de ocorrência de todos os termos.

A frequência inversa é dada pelo logaritmo do número de todos os documentos dividido pelo número de documentos contendo o termo, ou seja:

$$idf(i) = \log\left(\frac{N}{df(i)}\right) \quad \textbf{(2.3)}$$

Obtendo:

$$tf - idf(i) = tf(i) * idf(i) \quad \textbf{(2.4)}$$

que é uma proporção da frequência do termo no documento e da frequência com que ele aparece no conjunto total de documentos.

O cálculo *tf-idf* sintetiza a relevância da palavra em relação ao documento e a toda base de dados. Portanto, uma palavra que aparece em vários documentos tem um valor bem menor (próximo a zero) do que uma palavra que aparece em poucos documentos, pois uma palavra que aparece em poucos documentos pode ser considerada uma palavra mais específica.

O cálculo de frequência é comumente utilizado *tf-idf* devido a maior precisão da frequência das palavras que é calculada, o que permite ter uma qualidade maior no processo de agrupamento. No caso de agrupamento de textos é comum várias colunas apresentarem valores zero, constando que determinada palavra não existe em determinado documento.

2.2.1.5 Armazenamento em Vetores

Para trabalhar com mineração e agrupamento de textos, cada objeto é considerado como um documento textual e todos os objetos são representados através do *Modelo de Espaço Vetorial* ou *Vector Space Model* (VSM) (Salton et al., 1975).

Nesse modelo, o conteúdo de um documento é formalizado como um ponto em um *espaço multidimensional* e representado por um vetor **v**, onde **v** = {v_1 , $v_{2,}$ ..., v_n}, sendo que cada v_i (i = 1, 2, ..., n) é o peso do termo i no documento. O peso do termo é determinado de acordo com a medida *tf-idf*, que calcula a significância ou *frequência* do termo dentro do documento em relação ao conjunto de documentos.

Após a execução de todos estes passos, o dicionário de frequência de palavras está pronto para utilização em técnicas de agrupamento.

2.2.2 Agrupamento de Textos

O processo de agrupamento de textos tem como objetivo particionar um conjunto de textos em subconjuntos de forma que os textos de cada subconjunto tenham características em comum.

Existem diversas técnicas de agrupamento de textos disponíveis na literatura, entre elas *K-Médias* e suas variantes, algoritmos bioinspirados (Redes Neurais, Inteligência de Enxame), etc.

2.3 ENXAME DE PARTÍCULAS

Enxame de Partículas é uma das principais subáreas da *Inteligência de Enxame* e constitui o foco dessa dissertação. O termo *Inteligência de Enxame* (do inglês *Swarm Intelligence*) apareceu pela primeira vez em 1998 em um trabalho de Beni e Wang se referindo a sistemas robóticos interagindo em um ambiente (Beni e Wang, 1989).

A Inteligência de Enxame é a subárea da Computação Natural (de Castro, 2006; de Castro, 2007) que tem como inspiração o comportamento de insetos que vivem em comunidades, como formigas, abelhas e cupins e também no comportamento sociocognitivo humano. Os sistemas de enxame se baseiam no comportamento coletivo de indivíduos que interagem entre si e com o ambiente promovendo comportamentos coletivos inteligentes (Kennedy et al., 2001).

Insetos que vivem em comunidades, ou insetos sociais, resolvem diversos problemas, como busca de alimentos, construção de ninhos, defesa contra predadores, divisão de trabalho e etc (Freitas, 2008).

O comportamento de colônias de formigas tem sido um dos modelos mais populares de Inteligência de Enxame. Ao contrário do que acontece nos filmes, que mostram uma organização hierárquica em colônias de formigas, na qual a formiga rainha dá as ordens, não existe hierarquia nessas sociedades. Esses insetos, na maioria das vezes, se comunicam indiretamente através de uma interação com o ambiente, gerando um processo auto-organizado.

É importante ressaltar que o termo enxame (*swarm*), apesar de ser específico para um coletivo de abelhas, é um termo generalizado para indicar qualquer coleção estruturada de indivíduos (ou agentes) capazes de interagir (Kennedy et al., 2001).

Hoje em dia existem diversas aplicações de Inteligência de Enxame que resolvem problemas de robótica coletiva (Bonabeau et al., 1999), otimização (Kennedy e Eberhart, 1995), agrupamento de dados (Cohen e de Castro, 2006), problema do caixeiro viajante (Stutzle et al., 1999), navegação de robôs (Parhi et al., 2009), dentre outros.

Existem duas principais linhas de pesquisa na Inteligência de Enxame: a primeira se inspira no comportamento social de insetos, como formigas, cupins, abelhas e cardumes de peixes. Como exemplos, podemos citar trabalhos baseados no

comportamento social de formigas, como a habilidade de encontrar caminhos mais curtos para fontes de alimentos e a habilidade de organizar o ninho (Bonabeau et al, 1999; Dorigo e Stutzle, 2004). A segunda tem como inspiração a sociocognição (Kennedy, 2004).

Algumas das técnicas pioneiras de Inteligência de Enxame são:

- Algoritmo de Otimização por Colônias de Formigas (ACO) (Dorigo, 1992) que se inspira no comportamento das formigas buscando alimento, aplicado originalmente a problemas de otimização discreta;
- Algoritmo de Agrupamento baseado em Formigas (ACA) (Deneubourg et al., 1991) que se inspira no comportamento de formigas separando partes de formigas mortas em regiões do formigueiro, originalmente aplicado a problemas de agrupamento de dados;
- Otimização por Enxame de Partículas (PSO) (Eberhart e Kennedy, 1995), que se inspira na adaptação social do conhecimento de seres humanos, originalmente aplicado em problemas de otimização em ambientes contínuos;

Nesta dissertação serão utilizados os algoritmos de Otimização por Enxame de Partículas (PSO) e Agrupamento por Enxame de Partículas (PSC).

O foco desta dissertação está ligado com a frente de pesquisa de Inteligência de Enxame que se baseia na sociocognição. Em particular, o objetivo é adaptar um algoritmo de otimização por enxame de partículas para aplicação à tarefa de agrupamento e agrupamento ótimo de dados textuais.

2.3.1 Otimização por Enxame de Partículas (PSO)

O ser humano possui uma grande capacidade de processar informações, o que faz com que ele tenha facilidade para se adaptar a diversas situações. Em uma determinada situação na qual duas pessoas que tenham opiniões distintas estão conversando e tentam convencer uma a outra de que sua opinião é a “correta”, é possível se observar um processo de persuasão mútua. Se este processo for estendido para um grande número de indivíduos através do tempo, podemos observar uma auto-organização de normas e culturas (Kennedy, 2004). Com inspiração nesse processo e

com o objetivo de resolver problemas de otimização em ambientes contínuos, surgiu o algoritmo PSO (Kennedy e Eberhart, 1995).

O algoritmo PSO utiliza a seguinte teoria sociocognitiva: cada indivíduo da população possui sua própria experiência e também possui conhecimento da experiência dos seus vizinhos. Esse tipo de situação está presente no dia a dia da maioria das pessoas. Um exemplo seria o de utilizar o seu conhecimento mais o conhecimento de outras pessoas para chegar a um determinado destino. Por exemplo, João acabou de se formar em um curso de engenharia mecânica e gostaria de construir um carro. Para isso ele utiliza informações adquiridas no curso (aspecto cognitivo) mais o conhecimento de alguns colegas (aspecto social) engenheiros e técnicos que conhecem outras áreas do processo de construção de carros que João não conhece bem.

No algoritmo PSO, cada indivíduo é representado por uma *partícula*, ou seja, um *ponto no espaço* (solução candidata a um determinado problema), e o conjunto de partículas representa o *enxame*. As partículas se movimentam no espaço de busca e a movimentação das mesmas é influenciada pela sua melhor posição (termo cognitivo) e pela posição da melhor partícula do conjunto de partículas com as quais ela interage até o momento (termo social).

As partículas têm suas posições atualizadas de acordo com a Equação (2.5):

$$\mathbf{x}_i(\boldsymbol{t}+\mathbf{1}) = \mathbf{x}_i(\boldsymbol{t}) + \Delta\mathbf{x}(\boldsymbol{t}+\mathbf{1}) \qquad \textbf{(2.5)}$$

na qual $\mathbf{x}_i$ representa a posição da partícula e $\Delta\mathbf{x}$ um deslocamento.

Para calcular a velocidade das partículas são utilizados os seguintes termos:

- $\boldsymbol{\omega}$ - termo de inércia, que controla a convergência da partícula. Esse termo normalmente decai de 0.9 a 0.4 em uma execução (Eberhart e Shi, 2000). O trabalho feito por Clerc (1999) indica que o uso de um fator de constrição é necessário para garantir a convergência do algoritmo. O Coeficiente de constrição controla a convergência da partícula e é um método simples, porém elegante, para prevenir a explosão das partículas, assegurando a convergência e eliminando o parâmetro v_{max} (Poli et al., 2007).

- $\boldsymbol{\varphi_1}$ e $\boldsymbol{\varphi_2}$ - vetores de pesos aleatórios positivos que exercem influência nos termos cognitivo (melhor posição individual) e social (melhor posição global), respectivamente;
- $\mathbf{p}_j^i$ - vetor que contem a melhor posição na história da partícula *i* em relação ao objeto de entrada *j* (termo cognitivo);
- $\mathbf{x}_j$ - posição atual da partícula;
- $\mathbf{g}^i$ - vetor que contem a posição da melhor partícula em relação ao objeto de entrada até o momento (termo social).

O deslocamento das partículas é atualizado de acordo com a Equação (2.6):

$$\Delta\mathbf{x} = \boldsymbol{\omega}.\mathbf{v}_i(\boldsymbol{t}) + \boldsymbol{\varphi_1} \otimes \left(\mathbf{p}_i(\boldsymbol{t}) - \mathbf{x}_i(\boldsymbol{t})\right) + \boldsymbol{\varphi_2} \otimes \left(\mathbf{g}^i(\boldsymbol{t}) - \mathbf{x}_i(\mathbf{t})\right), \quad \textbf{(2.6)}$$

na qual o símbolo ⊗ representa produto elemento a elemento.

O deslocamento das partículas deve ser limitado a v_{max} e $-v_{max}$, segundo as regras:

Se $v_{id} > v_{max}$ **então**

$v_{id} = v_{max}$

Se $v_{id} < -v_{max}$ **então**

$v_{id} = -v_{max}$

onde *d* é uma dimensão da partícula *i*.

O pseudocódigo do algoritmo PSO para o problema de minimização de função é descrito a seguir:

```
1.    procedure [X] = PSO(max_it, v_max, n_part, ω)
2.    initialize X //usually every particle x_i is initialized at random
3.    initialize V //at random, v_i ∈ [-v_max, v_max]
4.    t = 1
5.    while t < max_it do
6.       for i = 1 to n_part do
7.          if f(x_i) < f(p_i) then  //f(p_i) is lBest
8.             p_i = x_i
9.          end if
10.         if f(x_i) < f(p_g) then  //f(g^i) is gBest
11.            g^i = x_i
12.         end if
13.         update v_i (Equação 2.7)
14.         v_i(t+1) ∈ [-v_max, v_max]
15.         update x_i (Equação 2.6)
16.      end for
17.      ω = 0.95*ω
18.      t = t+1
19.   end while
20.   end procedure
```

Pseudocódigo 2.1: Pseudocódigo PSO

2.3.2 Agrupamento por Enxame de Partículas (PSC)

O algoritmo de Agrupamento por Enxame de Partículas, também conhecido como PSC (*Particle Swarm Clustering*) foi proposto por Cohen e de Castro (2006). É um método inspirado no algoritmo PSO (*Particle Swarm Optimization*) (Eberhart e Kennedy, 1995).

Com o intuito de modificar o PSO para trabalhar com problemas de agrupamento de dados, Cohen e de Castro (2006) propuseram as seguintes modificações:

1. Não utilizar uma função explícita para avaliar a qualidade das soluções.
2. Todas as partículas configuram uma solução para o problema, diferente do PSO, no qual cada partícula representa uma solução para o problema.
3. Adição de um novo termo, denominado termo auto-organizado, que move as partículas na direção dos objetos de entrada.

No algoritmo PSC, as partículas se movem no espaço de maneira que elas se tornem protótipos (representantes) dos grupos da base de dados. A movimentação de uma partícula no espaço é influenciada pela sua melhor posição (termo cognitivo), pela posição da melhor partícula do enxame até o momento, ou seja, a partícula que chegou

mais próxima ao objeto (termo social), e pelo novo termo auto-organizado que move a partícula na direção do objeto.

As partículas têm suas posições atualizadas de acordo com a Equação (2.7):

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \Delta\mathbf{x}(t+1) \quad \textbf{(2.7)}$$

na qual $\mathbf{x}_i$ representa a posição da partícula e $\Delta\mathbf{x}$ um deslocamento.

Para calcular o deslocamento das partículas são utilizados os seguintes termos:

- ω - termo de inércia, que controla a convergência da partícula.
- φ_1, φ_2 e φ_3 - vetores de pesos aleatórios positivos que exercem influência nos termos cognitivo (melhor posição individual), social (melhor posição global) e de auto-organização (distância da partícula em relação ao objeto), respectivamente.
- $\mathbf{p}_j^i$ - vetor que contem a melhor posição na história da partícula *j* em relação ao objeto de entrada *i* (termo cognitivo).
- $\mathbf{x}_j$ - posição atual da partícula *j*.
- $\mathbf{g}^i$ - vetor que contem a posição da melhor partícula em relação ao objeto *i* de entrada até o momento (termo social).
- $\mathbf{y}^i$ - posição do objeto *i* (termo auto organizado).

O deslocamento das partículas é atualizado de acordo com a Equação (2.8):

$$\Delta\mathbf{x} = \omega.\Delta\mathbf{x}(t) + \varphi_1 \otimes \left(\mathbf{p}_j^i(t) - \mathbf{x}_j(t)\right) + \varphi_2 \otimes \left(\mathbf{g}^i(t) - \mathbf{x}_j(t)\right) + \varphi_3 \otimes (\mathbf{y}^i - \mathbf{x}_j(t)) \quad \textbf{(2.8)}$$

na qual o símbolo $\otimes$ representa o produto elemento a elemento.

Para evitar que as partículas fiquem estagnadas, Cohen e de Castro (2006) propuseram atualizar a posição das partículas que não se moveram na direção da partícula vencedora durante a iteração.

A primeira versão do algoritmo proposta por Cohen e de Castro (2006) utiliza a distância Euclidiana para calcular a similaridade entre os objetos e as partículas.

O pseudocódigo do algoritmo PSC é descrito a seguir (Cohen e de Castro, 2006):

```
1.    procedure [X] = PSC(data_set,max_it,vmax,n_part,ω)
2.    Y = data_set
3.    initialize X //aleatoriamente
4.    initialize V //aleatoriamente, v_j ∈[-v_max,v_max]
5.    initialize dist
6.    t ←1
7.    while t < max_it do,
8.    for i = 1 to N do, //para cada objeto
9.          for j = 1 to n_part do, //para cada partícula
10.              dist(j) = distance(y^i,x_j)
11.         end for
12.         I = min(dist)
13.         if distance(x_I, y^i) < distance(p_I^i, y^i),
14.              then p_I^i = x_I,
15.         end if
16.         if distance(x_I, y^i) < distance(g^i, y^i),
17.              then g^i = x_I,
18.         end if
19.         update Δx Equação (2.9)
20.         v_I ∈[-v_max,v_max]
21.         update x_i Equação (2.8)
22.   end for
23.   for i = 1 to n_part do,
24.       if(x_i != win)
25.           v_i(t+1) = ω v_i(t) + φ_4⊗(x_most_win - x_i(t))
26.           v_i ∈[-v_max,v_max]
27.           x_i(t+1) = x_i(t) + v_i(t+1)
28.       end if
29.   end for
30.   ω = 0.95*ω
31.   t ←t + 1
32.   end while
33.   end procedure
```

Pseudocódigo.2.2: Pseudocódigo PSC

2.3.2.1 PSC para Agrupamento de Textos

No problema de agrupamento de textos trabalha-se com um conjunto de dados com um grande número de atributos. Em Prior et al. (2009) os autores propuseram a aplicação do PSC à tarefa de agrupamento de textos transformando o conjunto de textos em vetores através do modelo de Espaço Vetorial (Salton et al., 1975) e utilizando a Medida do Cosseno como medida de similaridade.

O cálculo de similaridade entre os vetores é um passo muito importante, pois ele define o grau de similaridade entre os dados. Dois dos métodos mais utilizados para calcular a similaridade no campo de *Mineração de Dados* são o cálculo pela *distância Euclidiana* e o cálculo pela *distância do Ângulo do Cosseno*, mais conhecida como *Medida do Cosseno.* A Medida do Cosseno é a medida de similaridade mais utilizada na literatura para trabalhar com agrupamento de textos, pois ela é invariante a dimensão do

vetor, considerando apenas a distância angular entre dois vetores. O cálculo pela *distância do cosseno*, mais conhecido como *Medida do Cosseno*, essencialmente calcula o produto interno dos vetores normalizados:

$$\text{similaridade}(\mathbf{z}_a,\mathbf{z}) = \cos(\mathbf{z}_a,\mathbf{z}) = (\mathbf{z}_a \cdot \mathbf{z})/(||\mathbf{z}_a||_2 * ||\mathbf{z}||_2) \quad \textbf{(2.9)}$$

onde $\mathbf{z}_a$ e $\mathbf{z}$ são vetores cuja similaridade quer se medir, $||\mathbf{z}||_2$ é a norma Euclidiana, e $\mathbf{z}_a \cdot \mathbf{z}$ corresponde ao produto interno dos vetores $\mathbf{z}_a$ e $\mathbf{z}$. Note que quanto menor a similaridade, maior a distância e vice-versa.

3 PSC CONSTRUTIVO E PARA AGRUPAMENTO ÓTIMO

Sob a prespectiva da engenharia de algoritmos bioinspirados, essa dissertação tem como objetivo adaptar um algoritmo de otimização, denominado algoritmo de *otimização por enxame de partículas*, para aplicação à tarefa de agrupamento (cPSC) e agrupamento ótimo (oPSC) de dados textuais. Este capítulo descreve as duas contribuições do trabalho.

3.1 ALGORITMO CONSTRUTIVO DE AGRUPAMENTO BASEADO EM ENXAME DE PARTÍCULAS (cPSC)

O Algoritmo Construtivo de Agrupamento baseado em Enxame de Partículas, cPSC (*Constructive Particle Swarm Clustering*), tem como diferencial detectar automaticamente o número de protótipos do enxame e, consequentemente, a quantidade de grupos existentes em uma base de dados, ao contrário do PSC que recebe esse valor como parâmetro de entrada (*n_part*). Para que isso seja possível, o cPSC emprega uma estratégia de duplicação e poda de partículas que segue os mesmos princípios do PSC, com a adição de três novos processos inspirados na rede neuroimune ABNET (AntiBody NETwork) (de Castro et al., 2003):

1. Crescimento do número de partículas através de uma política de clonagem (duplicação);
2. Remoção ou poda de partículas;
3. Critério automático de parada.

Além das três funções também são introduzidos dois novos parâmetros, o parâmetro β e o limiar de afinidade ε, que exercem influência no crescimento do número de partículas. O limiar de afinidade ε é uma constante obtida empiricamente que controla a especificidade do enxame, influenciando na quantidade de partículas. Um limiar de afinidade alto resultará em partículas mais específicas e, consequentemente, um enxame com um número maior de partículas, enquanto valores pequenos de

afinidade implicam em partículas generalistas, resultando, assim, em um enxame com um número menor de partículas (Knidel, 2006). O parâmetro β é uma janela de controle populacional. Ele controla o crescimento do enxame e exerce uma grande influência no número de iterações para convergência. Ambos são utilizados como critérios para determinar o crescimento do enxame.

Os três novos processos do algoritmo são descritos a seguir.

3.1.1 Crescimento do Número de Partículas

Durante a execução do algoritmo é determinado o nível de concentração de cada partícula, ou seja, o número de objetos da base que essa partícula reconhece. Quando o nível de concentração da partícula for maior que um, ou seja, quando ela reconhecer mais do que um objeto, ela se torna candidata a ser clonada. A partícula que possuir o maior nível de concentração será escolhida como única candidata.

Se a afinidade da partícula candidata em relação ao objeto que possui maior afinidade a ela for maior do que o limiar de afinidade ε, então a partícula é considerada a partícula mais estimulada e é clonada; senão a quantidade de partículas permanece inalterada (de Castro et al., 2003). O processo de clonagem das partículas é verificado a cada β iterações, sendo β = 2 por definição.

Se a partícula candidata é clonada, a nova partícula gerada recebe os valores da média entre a posição da partícula e do objeto que possui maior afinidade a ela.

3.1.2 Poda das Partículas

O processo de poda das partículas verifica o nível de concentração das partículas do enxame. Se uma partícula *p* possui seu nível de concentração igual a zero, ou seja, se a partícula não reconhece objeto algum por uma quantidade de iterações maior do β, então esta partícula pode ser eliminada.

3.1.3 Critério de Parada

O critério de parada verifica se em uma janela de β iterações a diferença entre a posição atual das partículas e a memória da posição das partículas é menor ou igual a 0,01%; nesses casos o algoritmo encerra a execução.

3.1.4 Algoritmo

As partículas têm suas posições atualizadas de acordo com a Equação (3.1):

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \Delta\mathbf{x}(t+1) \quad \textbf{(3.1)}$$

onde $\mathbf{x}_i$ representa a posição da partícula, $\Delta\mathbf{x}$ um deslocamento e *t* um instante de tempo

Para calcular o deslocamento das partículas são utilizados os seguintes termos:

- $\boldsymbol{\omega}$ - termo de inércia, que controla a convergência da partícula.
- $\boldsymbol{\varphi_1}$, $\boldsymbol{\varphi_2}$ e $\boldsymbol{\varphi_3}$ - vetores de pesos aleatórios que exercem influência nos termos cognitivo (melhor posição individual), social (melhor posição global) e de auto-organização (distância da partícula em relação ao objeto), respectivamente.
- $\mathbf{p}_j^i$ - vetor que contem a melhor posição na história da partícula *j* em relação ao objeto de entrada *i* (termo cognitivo).
- $\mathbf{x}_j$ - posição atual da partícula *j*.
- $\mathbf{g}^i$ - vetor que contem a posição da melhor partícula em relação ao objeto *i* de entrada até o momento (termo social).
- $\mathbf{y}^i$ - posição do objeto *i* (termo auto-organizado).

O deslocamento das partículas é atualizado de acordo com a Equação (3.2):

$$\Delta\mathbf{x} = \boldsymbol{\omega}.\Delta\mathbf{x}(t) + \boldsymbol{\varphi_1} \otimes \left(\mathbf{p}_j^i(t) - \mathbf{x}_j(t)\right) + \boldsymbol{\varphi_2} \otimes \left(\mathbf{g}^i(t) - \mathbf{x}_j(t)\right) + \boldsymbol{\varphi_3} \otimes \left(\mathbf{y}^i - \mathbf{x}_j(t)\right) \quad \textbf{(3.2)}$$

onde o símbolo ⊗ representa produto de vetores elemento a elemento.

O pseudocódigo do algoritmo cPSC é descrito a seguir:

```
1.    procedure [X] = cPSC(data_set, n_part, ω, β, ε)
2.    Y = data_set
3.    initialize X //aleatoriamente
4.    initialize V //aleatoriamente, v_j ∈[-v_max,v_max]
5.    initialize dist
6.    t ←1
7.    while t < max_it do,
8.    for i = 1 to N do, //para cada objeto
9.          for j = 1 to n_part do, //para cada partícula
10.              dist(j) = distance(y^i,x_j)
11.         end for
12.         I = min(dist)
13          ind = index(I)
14          nwin(ind) = nwin(ind) + 1
15.         if distance(x_I, y^i) < distance(p_I^i, y^i),
16.              then p_I^i = x_I,
17.         end if
18.         if distance(x_I, y^i) < distance(g^i, y^i),
19.              then g^i = x_I,
20.         end if
21.         update Δx (Equação 3.2)
22.         v_I ∈[-v_max,v_max]
23.         update x_i (Equação 3.1)
24.   end for
25.   if mod(t,β) == 0,
26.       X=cresce_poda(X, nwin(X)),
27.   end if
28.   ω = 0.95*ω
29.   t ←t + 1
30.   end while
31.   end procedure
32.
33.   procedure [X] =  cresce_poda(X, nwin(X))
34.   for i = 1 to n_part do,
35.         if nwin(x_i) == 0,
36.              X = X – x_i,
37.         end if
38.   end for
39.   C = seleciona(nwin(X) > 1),
40.   c = arg max_∀i C_i,
41.   X = X ∪ c,
```

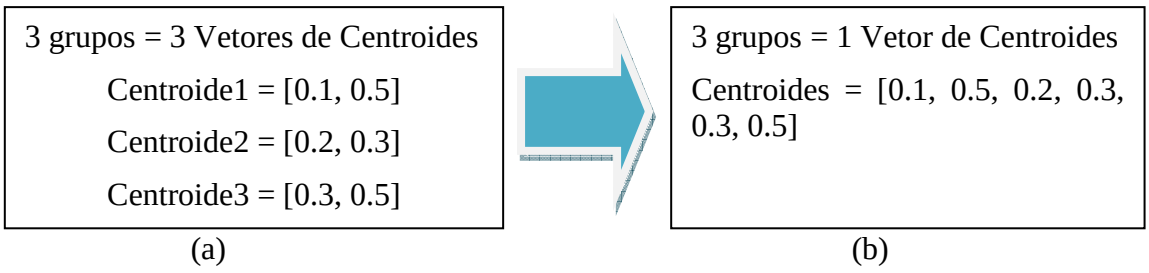
Pseudocódigo 3.1: Pseudocódigo cPSC

O vetor nwin representa o número de objetos que cada partícula reconhece.

3.2 AGRUPAMENTO ÓTIMO POR ENXAME DE PARTÍCULAS (oPSC)

Uma das contribuições dessa dissertação é o algoritmo oPSC, um algoritmo de agrupamento que utiliza uma função objetivo explícita para avaliar as soluções de agrupamento. Diferentemente do algoritmo cPSC, o oPSC considera cada partícula como uma possível solução para o problema de agrupamento, ou seja, cada partícula representa um conjunto de centroides (protótipos).

A Figura 3.1 (a) ilustra a representação dos centroides no algoritmo cPSC, onde cada centroide é representado por um vetor (partícula), e o conjunto desses centroides (enxame de partículas) representa uma possível solução para o problema. A Figura 3.2 (b) ilustra a representação dos centroides no algoritmo oPSC, onde todas os centroides são representados no mesmo vetor (partícula), ou seja, cada vetor representa um conjunto de centroides que é uma possível solução para o problema. Portanto, enquanto o cPSC adapta uma solução para o problema a cada iteração, o oPSC adapta múltiplas soluções candidatas simultaneamente a cada iteração.



(a) (b)

Figura 3.1: Representação dos centroides: (a) No algoritmo cPSC; (b) No algoritmo oPSC.

O primeiro passo do algoritmo é calcular o valor de aptidão (fitness) das partículas. Depois de calcular o valor de aptidão, determina-se a partícula vencedora. Esta então é comparada com a memória da partícula e a memória global. Caso a partícula vencedora possua uma melhor solução de agrupamento do que as memórias utilizadas nas comparações, estas são atualizadas. Por último, todas as partículas do enxame são atualizadas considerando sua melhor configuração até o momento e a melhor partícula do enxame.

No oPSC, assim como no PSO, a cada iteração avalia-se a aptidão de todas as partículas para verificar qual partícula possui a melhor solução de agrupamento. Porém, no cPSC um dado de entrada é apresentado de cada vez para verificar qual é a partícula mais similar ao mesmo. Como o oPSC não verifica um dado de cada vez para avaliar a solução, não há mais a necessidade de utilizar o termo $\boldsymbol{\varphi_3}$ que move a partícula em direção ao dado de entrada.

Para atualizar a posição das partículas são considerados os seguintes termos:

- $\boldsymbol{\varphi_1}$ e $\boldsymbol{\varphi_2}$ - vetores de pesos aleatórios que exercem influência nos termos cognitivo (melhor posição individual) e social (melhor posição global), respectivamente.

- $\mathbf{p}_j$ - vetor que contem a melhor posição na história da partícula (termo cognitivo).
- $\mathbf{x}_j$ - posição atual da partícula.
- $\mathbf{g}^i$ - vetor que contém a posição da melhor partícula até o momento (termo social).

A posição da partícula é atualizada de acordo com a Equação (3.3):

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \boldsymbol{\varphi}_1 \otimes \left(\mathbf{p}_j\ (t) - \mathbf{x}_j(t)\right) + \boldsymbol{\varphi}_2 \otimes \left(\mathbf{g}^i(t) - \mathbf{x}_j(t)\right) \quad \textbf{(3.3)}$$

onde o símbolo ⊗ representa produto elemento a elemento.

O pseudocódigo do algoritmo oPSC utilizando é descrito a seguir:

```
1.    procedure [X] = oPSC(data_set, max_it, n_part)
2.    Y = data_set
3.    initialize X //aleatoriamente
4.    t ←1
5.    while t < max_it do,
6.          dist(j) = fitness(X, Y)
7.          I = max(dist)
8.          if I > distance(p_I),
9.          then p_I = x_I,
10.         end if
11.         if I > distance(g^i),
12.               then g^i = x_I,
13.         end if
14.         update x_I (Equação 3.3)
15.         t ←t + 1
16.    end while
17.    end procedure
```

Pseudocódigo 3.2: Pseudocódigo oPSC

Muitos critérios de validação de agrupamentos podem ser usados para avaliar partições contendo um número *k* de grupos. Dentre os diversos critérios de avaliação disponíveis na literatura, dois foram utilizados nesta dissertação: a *silhueta* e a *soma das distâncias intragrupo*.

3.2.1 Silhueta

Uma das funções de aptidão utilizadas nesta dissertação é uma versão simplificada da silhueta proposta por Kaufman e Rousseeuw (1990). A silhueta para cada objeto *i*, *s*(*i*), é calculada da seguinte maneira:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad \textbf{(3.4)}$$

onde $a(i)$ é a dissimilaridade média do objeto i pertencente a um grupo qualquer **A** em relação aos outros objetos do mesmo grupo; $b(i) = \min d(i,\mathbf{C})$, $\mathbf{C} \neq \mathbf{A}$, sendo que $d(i,\mathbf{C})$ é a dissimilaridade média do objeto i em relação a todos os outros objetos pertencentes a um grupo qualquer **C** diferente de **A**.

Na silhueta simplificada calcula-se a distância entre os objetos e os *centroides* dos grupos. A maximização do valor médio de $s(i)$ da silhueta simplificada tende a induzir *clusters* compactos e bem separados (Kaufman e Rousseeuw, 1990).

O valor da silhueta $s(i) \in [-1, +1]$, sendo que quanto maior o valor de $s(i)$ maior é a proximidade do objeto $\mathbf{x}_i$ com um dado grupo. Se o valor de $s(i)$ for zero, então não é possível definir claramente se este objeto $\mathbf{x}_i$ deveria estar no grupo atual ou em outro grupo próximo. Se um grupo **A** possui apenas um valor, então $s(i)$ não é definida e é considerada zero. O *critério da Silhueta* é dado pela média de $s(i)$, $i = 1, ..., N$, onde N é o número de objetos da base, e a melhor forma de agrupamento é encontrada maximizando esta função.

3.2.2 Soma das Distâncias Intragrupo

A função que minimiza a soma das distâncias intragrupo visa encontrar grupos em que a distancia entre os objetos que fazem parte desse grupo seja a menor possível. Ela considera os objetos da base $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ e os centroides dos grupos $\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_k\}$:

$$F(\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_k) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in \mathbf{C}_j} \| \mathbf{x}_i - \mathbf{c}_j \|^2 , \qquad \textbf{(3.5)}$$

onde $\{\mathbf{C}_1, ..., \mathbf{C}_k\}$ é o conjunto de k grupos codificados na partícula, $\mathbf{x}_i$ é um objeto da base e $\mathbf{c}_j$ é o centroide do grupo $\mathbf{C}_j$.

4 ANÁLISE DE DESEMPENHO

Neste capítulo são avaliados os algoritmos propostos na dissertação: o cPSC (Algoritmo Construtivo de Agrupamento baseado em Enxame de Partículas) e o oPSC (Agrupamento Ótimo por Enxame de Partículas). Além disso, são descritas as medidas de avaliação, bases de dados utilizadas para avaliar o desempenho de cada algoritmo e a comparação de desempenho com o *K-Médias.*

A implementação computacional das ferramentas propostas foi feita na linguagem de programação Java através da IDE Eclipse. Os experimentos do algoritmo cPSC foram realizados no Laboratório de Computação Natural (LCoN) vinculado ao PPGEE, em máquinas com as seguintes configurações: Intel Core 2 Duo 2,8 GHz com 4GB de Memória RAM e HD de 250GB. Os experimentos do algoritmo oPSC foram realizados em uma máquina Intel Atom CPU Z520 1.33GHz, com 2GB de Memória RAM e HD de 320GB.

4.1 MATERIAIS E MÉTODOS

4.1.1 Medidas de Avaliação

Para avaliar o desempenho das ferramentas propostas nesta dissertação, foram utilizadas as medidas de *Entropia*, *Pureza e Percentual de Classificação Incorreta*. A *Entropia* e a *Pureza* são medidas de avaliação amplamente utilizadas na literatura para avaliar o desempenho de algoritmos de agrupamento, desde que conheçamos a priori os grupos da base de dados (apenas para efeito de avaliação) (Crabtree et al., 2005), (Zhao e Karypis, 2004).

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 cluster S_r de tamanho n_r, a entropia $E(S_r)$ do cluster pode ser medida da seguinte maneira:

$$E(\mathbf{S}_r) = -\frac{1}{\log k}\sum_{i=1}^{k}\frac{n_r^i}{n_r}\,log\frac{n_r^i}{n_r} \qquad \textbf{(4.1)}$$

sendo *k* o numero de classes no conjunto de documentos e n_r^i o número de documentos da classe *i* no cluster S_r. A entropia global pode ser calculada como a soma das entropias de cada cluster, ponderadas pelo tamanho de cada cluster:

$$\boldsymbol{E_g = \sum_{r=1}^{k} \frac{n_r}{n} E_r} \quad \textbf{(4.2)}$$

A pureza fornece a razão da classe dominante no cluster em relação ao tamanho do mesmo. Valores de pureza próximos a 01 (um) indicam um subconjunto puro da classe dominante no grupo. A pureza pode ser calculada da seguinte maneira:

$$\boldsymbol{P(S_r) = \frac{\max(n_r^i)}{n_r}}, \quad \textbf{(4.3)}$$

A pureza global pode ser definida como:

$$\boldsymbol{P_g = \sum_{r=1}^{k} \frac{n_r}{n} P_r}, \quad \textbf{(4.4)}$$

sendo n_r definido acima.

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 e a pureza seja igual a 01 (um).

O percentual de classificação incorreta, *CI*, é o número de objetos classificados incorretamente, N_{ci}, dividido pelo número total de objetos, *n*, vezes 100%:

$$\boldsymbol{CI = {N_{ci}}/{n} * 100\%} \quad \textbf{(4.5)}$$

4.1.2 Bases de Dados

4.1.2.1 Algoritmo Construtivo de Agrupamento Baseado em Enxame de Partículas (cPSC)

Para avaliar o desempenho do algoritmo cPSC, três bases de dados públicas e rotuladas foram escolhidas. A primeira foi a base Reuters-21578 (Lewis et al., 2004)

compilada por David Lewis e originalmente construída pelo Grupo Carnegie de notícias da Reuters em 1987. Esses dados são normalmente empregados levando em conta os documentos das classes mais frequentes, como, por exemplo, das 115, 90, ou 10 categorias mais frequentes (Sebastiani, 2002; Joachims, 1997). Nos experimentos a serem relatados aqui, foram obtidos os textos das 10 categorias mais frequentes, conhecidas como: *earn*, *acquisition*, *money-fx*, *grain*, *crude*, *trade*, *interest*, *ship*, *wheat* e *corn*, totalizando 7.193 documentos.

Para avaliar o desempenho dos algoritmos em relação a um aumento na base de dados textuais, a base foi dividida em três subamostras com 500, 1000 e 2000 melhores atributos, selecionados como aqueles de maior ganho de informação, onde cada subamostra contém o número de categorias da coleção, isto é, 10 categorias. O cálculo do ganho de informação fornece um valor de significância às palavras relevantes (Joachims, 1997), permitindo a ordenação do dicionário de palavras pela sua relevância.

A segunda base de dados utilizada neste trabalho foi a Spambase (*UCI Repository of Machine Learning Databases*) recolhida por George Forman. Esta base contem 4601 instâncias de *e-mails* com 48 atributos, dos quais 1.813 são *spam* e 2.788 não são. A coleção de *e-mails* foi feita através de um *postmaster* e alguns indivíduos que contribuíram de forma independente; a coleção de *e-mails* que não eram *spam* veio de *e-mails* pessoais e de trabalho. Os anúncios de produtos e/ou *sites* da Internet, sistemas de fazer dinheiro rápido, correntes e pornografia foram considerados *spam*. Essa base de dados já estava pré-processada.

A terceira base de dados utilizada foi a 20 Newsgroups construída por Ken Lang (Lang, 1995). Esta base contém aproximadamente 20.000 emails de notícias particionados em 20 grupos. Estes dados foram empregados levando em conta os documentos das classes *rec.autos*, *rec.motorcycles*, *rec.sport.baseball*, *rec.sport.hockey*. Além disso, a base foi dividida em uma subamostra com os 500 melhores atributos, selecionados como aqueles de maior ganho de informação.

As Tabelas 4.1 e 4.2 apresentam as informações referentes ao número de *tokens* e documentos das bases Reuters e 20 Newsgroups.

Tabela 4.1: Número de documentos nas dez maiores classes da base Reuters (RE); número total de documentos (NTD) utilizados em cada subamostra; número de tokens em cada dicionário gerado (NT).

Categoria	500	1000	2000
1	1649	1649	1649
2	181	181	181
3	389	389	389
4	2877	2877	2877
5	433	433	433
6	347	347	347
7	538	538	538
8	197	197	197
9	369	369	369
10	212	212	212
NTD	7192	7192	7192
NT	500	1000	2000

Tabela 4.2: Número de documentos nas quatro classes da base 20 Newsgroups (20N) utilizadas nas simulações; número total de documentos (NTD) utilizados em cada subamostra; número de tokens em cada dicionário gerado (NT).

Categoria	500
1	977
2	989
3	985
4	996
NTD	**3947**
NT	**500**

4.1.2.2 Agrupamento Ótimo por Enxame de Partículas (oPSC)

As bases de dados usadas para avaliar o desempenho do algoritmo oPSC fazem parte do repositório de bases de dados da Universidade da Califórnia em Irvine (UCI), Estados Unidos (http://archive.ics.uci.edu/ml/datasets.html). As principais características dessas bases de dados são listadas na Tabela 4.3.

Ao contrário das bases de dados textuais descritas na seção anterior, as bases de dados utilizadas nas simulações do oPSC já estão disponíveis pré-processadas.

Tabela 4.3: Principais características das bases de dados utilizadas para validação do algoritmo proposto

Bases de Dados	Objetos	Atributos	Classes
Diabetes	768	8	2
Iris	150	4	3
Ruspini	75	2	4
Yeast	205	20	4

4.2 ANÁLISE DE SENSIBILIDADE DO ALGORITMO cPSC

A aplicação do cPSC ao problema de agrupamento de textos requer a definição dos seguintes parâmetros: ω, β, v_{max}, φ_1, φ_2, φ_3 e ε. Esta seção apresenta uma breve discussão de como os principais parâmetros do algoritmo cPSC foram escolhidos para realizar os experimentos.

Com o objetivo de estudar a influência desses parâmetros no desempenho do cPSC, uma análise de sensibilidade simplificada do algoritmo foi realizada, aplicando o mesmo à base Reuters com 500 atributos.

Foram realizadas dez simulações e os resultados apresentados nas tabelas a seguir mostram a média e o desvio padrão de dez execuções. As simulações foram executadas utilizando os seguintes parâmetros: $\varepsilon = 0.1$, $\omega = 0.09$, $\beta = 2$, $v_{max} = 0.1$, φ_1, $\varphi_2, \varphi_3 \in [0\ \ 1]$, variando de acordo com cada análise.

A Tabela 4.4 mostra que o parâmetro ε influência a especificidade das partículas e, assim, o número final de partículas no enxame: quanto maior o valor de ε, menor o número de partículas no enxame e vice-versa. Foi possível observar que um aumento no número de partículas no enxame (valores menores de ε) resultou em melhores valores de Entropia e Pureza.

Tabela 4.4: Número de partículas, Entropia, Pureza e Classificação Incorreta (CI) para diferentes valores de ε.

ε	0,1	0,3	0,5
Número de partículas	15	5	4
Entropia	0,22±0,02	0,46±0,06	0,64±0,06
Pureza	0,78±0,02	0,70±0,04	0,64±0,01
CI	22,07±2,17	30,21±3,71	35,40±1,37

A Figura 4.1 resume a influência do limiar de afinidade (ε) no número de partíulas do enxame.

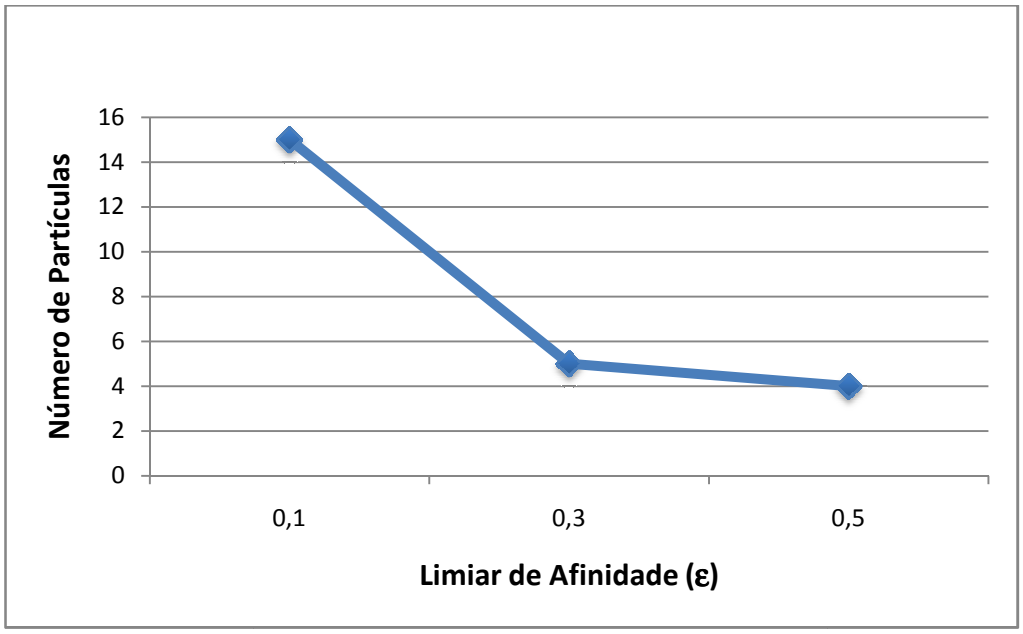


Figura 4.1: Influência do limiar de afinidade (ε) no número de partículas.

Para avaliar a influência do termo de inércia (ω) no tamanho do enxame e no número de iterações para convergência três valores foram escolhidos: $\omega \in \{0.09, 0.5, 0.95\}$. Os resultados são apresentados na Tabela 4.5. Foi possível observar que o termo de inércia (ω) exerce pouca influência no número de partículas do enxame, mas exerce uma grande influência no número de iterações para convergência.

Tabela 4.5: Número de partículas, Entropia, Pureza, Classificação Incorreta (CI) e número de iterações para diferentes valores de ω.

ω	**0,09**	**0,5**	**0,95**
Número de partículas	15	18	21
Entropia	0,22±0,02	0,20±0,01	0,18±0,02
Pureza	0,78±0,02	0,79±0,01	0,80±0,01
CI	22,07±2,17	20,76±0,92	20,02±1,46
Iterações	140	175	190

A Figura 4.2 resume a influência do termo de inércia (ω) no número de partículas do enxame.

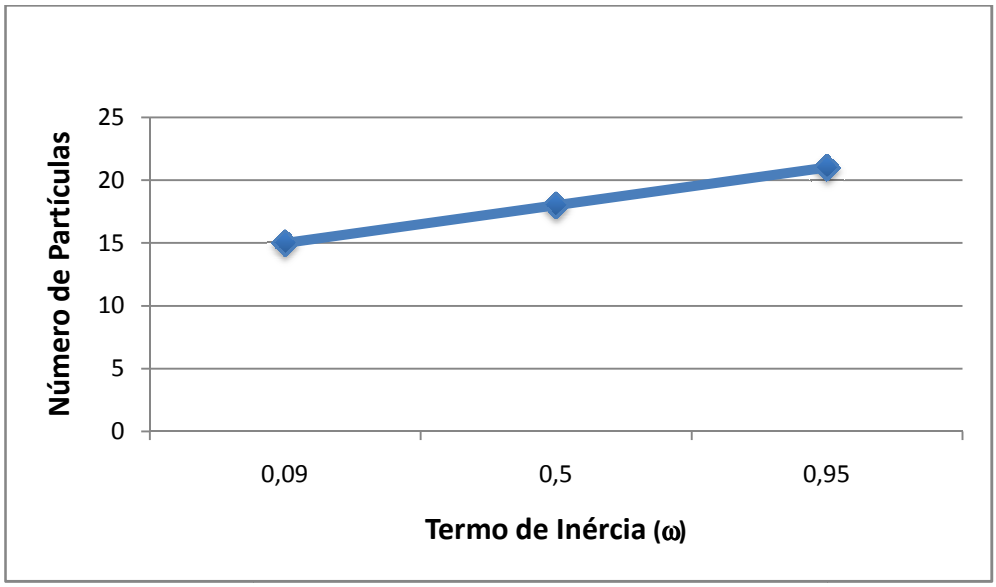


Figura 4.2: Influência do termo de inércia (ω) no número de partículas.

O parâmetro β controla o crescimento do enxame e exerce uma grande influência no número de iterações para convergência. Valores altos de β resultam em um maior tempo de convergência. Foi possível observar que o parâmetro β não exerce influência nos valores de Entropia e Pureza.

Esta análise permite sugerir a seguinte configuração de parâmetros para a aplicação genérica do algoritmo:

- Limiar de afinidade entre as partículas e os dados: $\varepsilon \approx 0{,}1$. Esse valor é sugerido, pois, apesar de promover um maior crescimento do enxame, resulta em desempenhos melhores de entropia e pureza;
- Termo de inércia: $\omega = 0{,}09$. Esse valor é sugerido, pois o termo de inércia tem maior influência no número de iterações para convergência do que no desempenho do algoritmo, sendo valores menores importantes para acelerar a convergência do cPSC;
- Parâmetro do crescimento do enxame: $\beta = 2$. Assim como o ω, influencia o tempo de convergência sem prejudicar o desempenho.

Os seguintes parâmetros foram utilizados para executar o algoritmo cPSC: $\omega = 0{,}09$, $\beta = 2$, $v_{max} = 0{,}1$, $\varphi_1, \varphi_2, \varphi_3 \in [0, 1]$.

Foi possível observar também que ε é sensível ao tamanho da base de dados. Para realizar as simulações o parâmetro ε foi ajustado com diferentes valores para cada

base, tal que o algoritmo conseguisse detectar um número ideal de grupos em cada base de dados que fosse próximo ao número real de grupos da base.

A análise de sensibilidade do algoritmo cPSC foi feita considerando os parâmetros ε, ω, e β que exercem influência no número de iterações para convergência do algoritmo e no desempenho de Entropia e Pureza. Diferentemente do algoritmo cPSC, o oPSC contem apenas o número de partículas como parâmetro de entrada do algoritmo, por essa razão não foi realizada uma análise de sensibilidade desse algoritmo.

4.3 EXPERIMENTOS E RESULTADOS

4.3.1 Algoritmo Construtivo de Agrupamento Baseado em Enxame de Partículas (cPSC)

Como o algoritmo cPSC define automaticamente o número de grupos da base de dados (*k* do *K-Médias* e o *número de partículas* do PSC), ele foi executado primeiro, depois calculou-se a média do número de iterações e do número de grupos encontrados pelo algoritmo e os mesmos foram adotados como o valor de *k* do *K-Médias*, o *número de partículas* do PSC, e o número de iterações do PSC executado posteriormente. Isso é feito para que a comparação entre os algoritmos considere os resultados de agrupamentos encontrados para o mesmo número de grupos. Para cada subamostra foram realizadas dez simulações para cada algoritmo. Os resultados apresentados nas tabelas a seguir mostram a média e o desvio padrão de dez execuções para cada base de dados e algoritmo e cada medida de avaliação considerada.

Tabela 4.6: Média ± desvio padrão das medidas de Entropia (E), Pureza (P) e Percentual de Classificação Incorreta (CI) dos algoritmos cPSC, PSC e *K-Médias* aplicados a base Spambase.

Reuters		**500**	**1000**	**2000**
cPSC	**E**	0,22±0,02	0,22±0,01	0,21±0,02
	P	0,78±0,02	0,78±0,02	0,79±0,03
	CI	22,07±2,17	22,01±1,39	21,22±2,54
PSC	**E**	0,22±0,01	0,22±0,01	0,22±0,01
	P	0,78±0,02	0,78±0,03	0,78±0,01
	CI	22,43±1,73	22,12±2,50	21,81±0,86
K-Médias	**E**	0,23±0,01	0,23±0,01	0,23±0,02
	P	0,76±0,02	0,77±0,02	0,77±0,03
	CI	23,39±2,23	23,47±2,08	23,37±2,58

Tabela 4.7: Média ± desvio padrão das medidas de Entropia (E), Pureza (P) e Percentual de Classificação Incorreta (CI) dos algoritmos cPSC, PSC e *K-Médias* aplicados a base Spambase.

Spambase		
cPSC	**E**	0,44±0,14
	P	0,74±0,05
	CI	25,45±4,68
PSC	**E**	0,47±0,02
	P	0,73±0,03
	CI	27,38±3,17
K-Médias	**E**	0,47±0,03
	P	0,70±0,04
	CI	29,71±4,15

Tabela 4.8: Média ± desvio padrão das medidas de Entropia (E), Pureza (P) e Percentual de Classificação Incorreta (CI) dos algoritmos cPSC, PSC e *K-Médias* aplicados a base 20 Newsgroups.

20 Newsgroups		**500**
cPSC	**E**	0,21±0,09
	P	0,91±0,07
	CI	8,63±6,95
PSC	**E**	0,22±0,08
	P	0,90±0,10
	CI	10,37±9,87
K-Médias	**E**	0,28±0,10
	P	0,84±0,11
	CI	16,42±11,63

É possível observar que todos os algoritmos tiveram um desempenho similar em termos de Entropia e Pureza. Observando os resultados apresentados nas Tabelas 4.6 e 4.8, o algoritmo cPSC e o algoritmo PSC tiveram um desempenho semelhante, em média, seguidos pelo *K-Médias* com resultados um pouco inferiores. Na Tabela 4.7 é possível observar que o algoritmo cPSC teve um desempenho melhor, em média, do que o PSC e o *K-Médias*.

As Figuras 4.3 e 4.4 resumem os resultados médios de Entropia e Pureza obtidos pelos algoritmos aplicados a base Reuters.

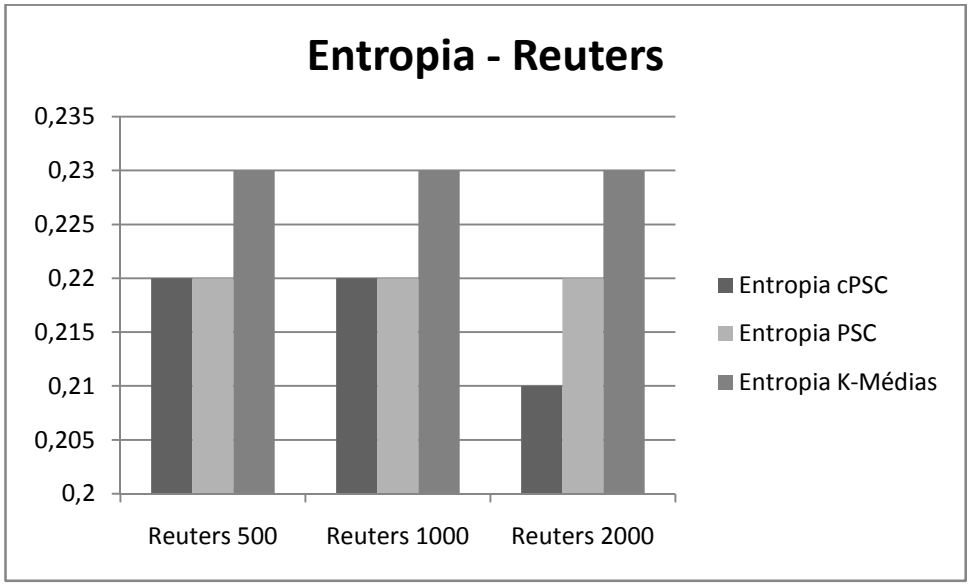


Figura 4.3: Valores de Entropia obtidos pelos algoritmos cPSC, PSC e *K-Médias*.

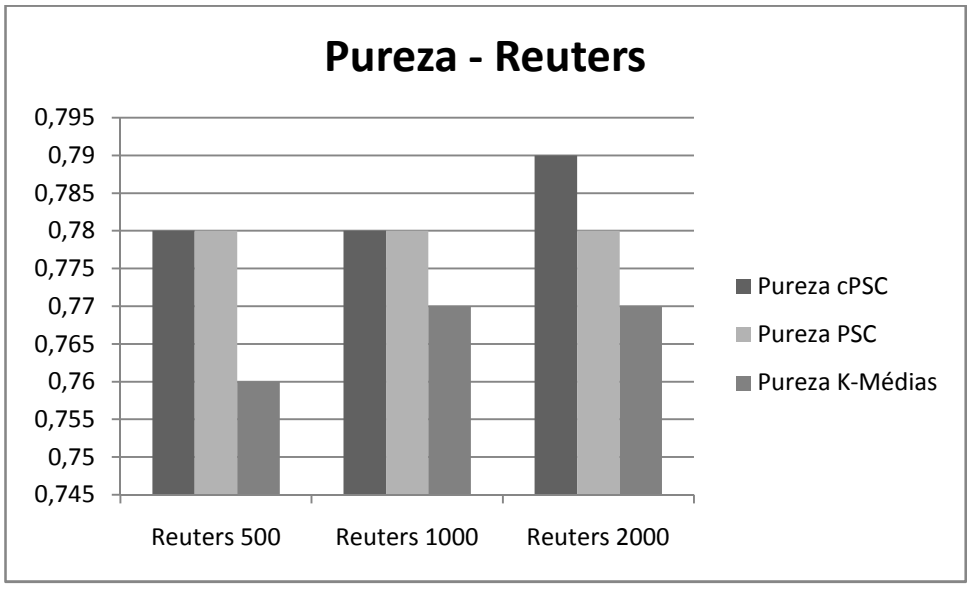


Figura 4.4: Valores de Pureza obtidos pelos algoritmos cPSC, PSC e *K-Médias*.

Em relação ao percentual de classificação incorreta, para a base Reuters, os algoritmos cPSC e PSC tiveram um desempenho semelhante, já o *K-Médias* teve um desempenho inferior. Para as bases Spambase e 20 Newsgroups o algoritmo cPSC encontrou as menores percentuais de classificação incorreta.

4.3.2 Agrupamento Ótimo por Enxame de Partículas (oPSC)

O algoritmo foi inicializado com os seguintes parâmetros: número de partículas = 50 e número de iterações = 100. Duas versões do algoritmo foram avaliadas, uma utilizando a função de *fitness* que maximiza a distância intracluster (oPSC MI), e outra que utiliza a função da silhuetta como função de *fitness* (oPSC SI). Para avaliar o desempenho do oPSC ele foi comparado ao algoritmo *K-Médias*. Para cada base de dados foram realizadas dez simulações para cada algoritmo. Os resultados apresentados nas tabelas a seguir mostram a média e o desvio padrão de dez execuções para cada base de dados e algoritmo e cada medida de avaliação considerada.

Tabela 4.9: Média ± Desvio Padrão de Entropia (E), Pureza (P), Percentual de Classificação Incorreta (CI) e Tempo de Execução (T) em segundos, para o oPSC utilizando a função que minimiza a soma das distâncias intra cluster (oPSC MI), oPSC utilizando a função da silhueta (oPSC SI) e *K-Médias*

Base de Dados		**oPSC MI**	**oPSC SI**	***K-médias***
Ruspini	**E**	0,02±0,02	0,04±0,08	0,30±0,5
	P	0,99±0,01	0,97±0,07	0,72±0,03
	CI	0,67±0,01	2,67±3,21	27,59±0,03
	T	3s±0s	5s±1s	13s±19s
Iris	**E**	0,40±0,06	0,42±0,00	0,42±0,03
	P	0,70±0,07	0,66±0,0	0,71±0,05
	CI	29,73±0,07	33,00±0,01	29,06±0,04
	T	9s±1s	9s±0s	22s±21s
Diabetes	**E**	0,5±0,11	0,93±0,0	0,0±0,0
	P	0,99±0,02	0,65±0,0	1,0±0,0
	CI	1,0±0,02	34,90±0,0	0,0±0,0
	T	1m05s±11s	36s±0s	1m13s±1m21s
Yeast	**E**	0,36±0,08	0,62±0,02	0,10±0,03
	P	0,83±0,05	0,51±0,01	0,94±0,03
	CI	16,73±0,05	48,83±0,01	5,56±0,03
	T	45s±3s	40s±6s	1m42s±27s

Observando os resultados apresentados na Tabela 4.9 o algoritmo oPSC e o algoritmo *K-Médias* tiveram um desempenho semelhante em termos de Entropia e Pureza para as bases Iris e Diabetes. As duas versões do algoritmo oPSC obtiveram

melhores resultados de Entropia e Pureza, em média, para a base Ruspini, já para a base Yeast o *K-Médias* obteve um melhor desempenho.

No caso do algoritmo que utiliza *critério da Silhueta* como função de fitness para a base Diabetes foi possível observar que o uso da informação intergrupo prejudicou o desempenho do algoritmo.

As Figuras 4.5 e 4.6 resumem os resultados de Entropia e Pureza obtidos pelos algoritmos para cada base de dados.

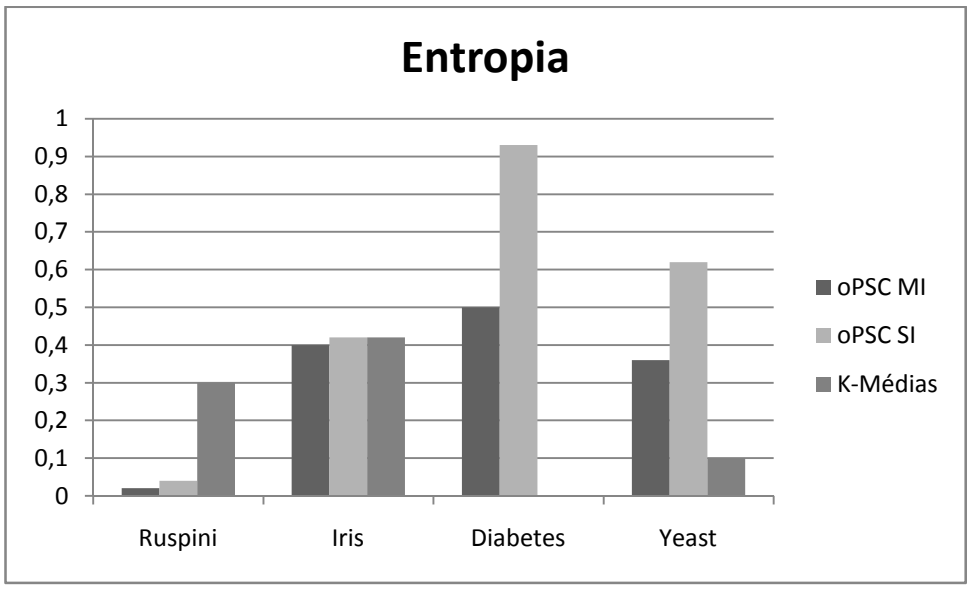


Figura 4.5: Valores de Entropia obtidos pelos algoritmos oPSC MI, oPSC SI e *K-Médias*.

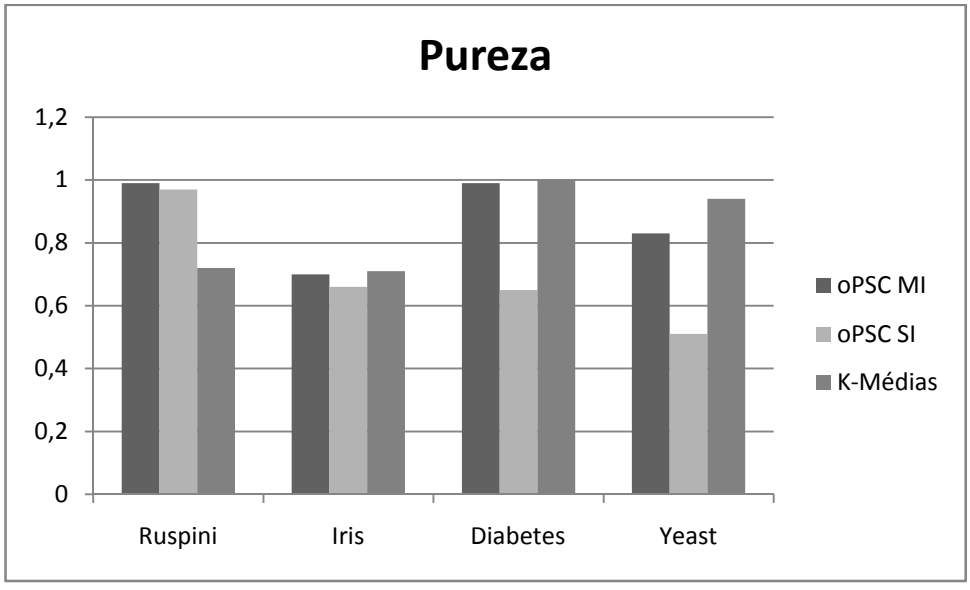


Figura 4.6: Valores de Pureza obtidos pelos algoritmos oPSC MI, oPSC SI e *K-Médias*.

Em relação ao percentual de classificação incorreta o *K-Médias* apresentou um menor percentual para as bases Iris, Diabetes e Yeast, já para a base Ruspini o algoritmo oPSC utilizando a função que maximiza a distância intracluster (oPSC MI) obteve um menor percentual.

4.4 DISCUSSÃO

4.4.1 Algoritmo Construtivo de Agrupamento Baseado em Enxame de Partículas (cPSC)

É importante ressaltar que, apesar de ter resultados similares ao PSC, o algoritmo cPSC detecta automaticamente o número de grupos da base de dados. No caso dos algoritmos PSC e *K-Médias* o número de grupos que devem ser encontrados na base de dados é definido pelo usuário. Em geral, os resultados mostram que os algoritmos são razoavelmente robustos em relação ao seu desempenho.

Para avaliar a significância estatística da diferença de desempenho dos algoritmos uma análise de variância (teste ANOVA) foi realizada com valor de confiança de 95%. Para todos os resultados (Reuters 500, Reuters 1000, Spambase e 20 Newsgroups), com exceção dos valores de Entropia encontrados para a base Reuters com 2000 atributos, a hipótese nula não pode ser rejeitada, o que significa que os valores de Entropia e Pureza são equivalentes, apesar de, na maioria dos casos, o desempenho absoluto do cPSC ter sido superior. Para os resultados de Entropia da base Reuters com 2000 atributos foi possível concluir que o algoritmo cPSC teve um melhor desempenho. Mesmo assim, ao analisarmos os desempenhos de melhor e pior caso, verificamos uma superioridade em termos de qualidade da solução quando comparado aos outros algoritmos. As Tabelas 4.10 e 4.11 mostram os melhores e piores valores absolutos de Entropia e Pureza encontrados para cada base de dados.

Tabela 4.10: Melhor resultado de Entropia e Pureza encontrado para cada base de dados.

Melhor Resultado				
	Entropia	**Algoritmo**	**Pureza**	**Algoritmo**
Reuters 500	0.19	cPSC	0.81	cPSC
Reuters 1000	0.19	cPSC	0.81	cPSC
Reuters 2000	0.18	cPSC	0.83	cPSC
Spambase	0.23	cPSC	0.83	cPSC
20Newsgroups	0.17	PSC	0.95	PSC

Tabela 4.11: Pior resultado de Entropia e Pureza encontrado para cada base de dados.

Pior Resultado				
	Entropia	**Algoritmo**	**Pureza**	**Algoritmo**
Reuters 500	0.25	PSC/ *K-Médias*	0.72	*K-Médias*
Reuters 1000	0.25	PSC/ *K-Médias*	0.73	PSC/ *K-Médias*
Reuters 2000	0.27	*K-Médias*	0.72	*K-Médias*
Spambase	0.74	cPSC	0.62	*K-Médias*
20Newsgroups	0.47	cPSC	0.70	*K-Médias*

O algoritmo cPSC obteve melhores resultados absolutos de Entropia e Pureza para as bases Reuters (todas as amostras) e Spambase. O algoritmo *K-Médias* obteve a maioria dos piores resultados. Para os resultados de Entropia e Pureza da base Reuters 1000 e Entropia da base Reuters 500 o algoritmo *K-Médias* e o PSC obtiveram os mesmos piores valores. Apenas para o pior valor de Entropia da base Spambase e 20Newsgroups o cPSC obteve o pior resultado.

É possível observar que o algoritmo cPSC obteve o melhor e o pior resultado de Entropia para a base Spambase. Isso pode ser explicado pelo fato de que como ele detecta automaticamente o número de grupos na base de dados, esse número pode variar de uma execução para outra. Por exemplo, em uma determinada execução ele pode encontrar dois grupos na base Spambase e na execução seguinte ele pode encontrar três grupos. Essa pequena variação no número de grupos encontrados pode alterar o valor de Entropia e Pureza. Encontrando um maior número de grupos na base de dados o resultado de Entropia pode melhorar já que os grupos encontrados podem ser mais homogêneos.

De acordo com a Tabela 4.12, o algoritmo cPSC encontrou um número razoável de partículas próximo ao número real de grupos de cada base de dados.

Tabela 4.12: Média, maior valor (max), menor valor (min) e desvio padrão das partículas encontradas pelo algoritmo cPSC encontrados para cada base de dados.

Quantidade de Partículas				
	Média	**Max**	**Min**	**Desvio**
Reuters 500	15	16	13	0,92
Reuters 1000	15	16	12	1,27
Reuters 2000	15	16	15	0,23
Spambase	3	6	2	1,17
20Newsgroups	4	4	3	0,32

Por fim, a Figura 4.7 ilustra o crescimento do enxame em uma execução típica do algoritmo cPSC aplicado a base de dados Reuters 500. O número de partículas vai aumentando de duas em duas iterações devido ao parâmetro β ter sido definido como β = 2. A partir da quadragésima iteração o algoritmo estabiliza até satisfazer o critério de parada.

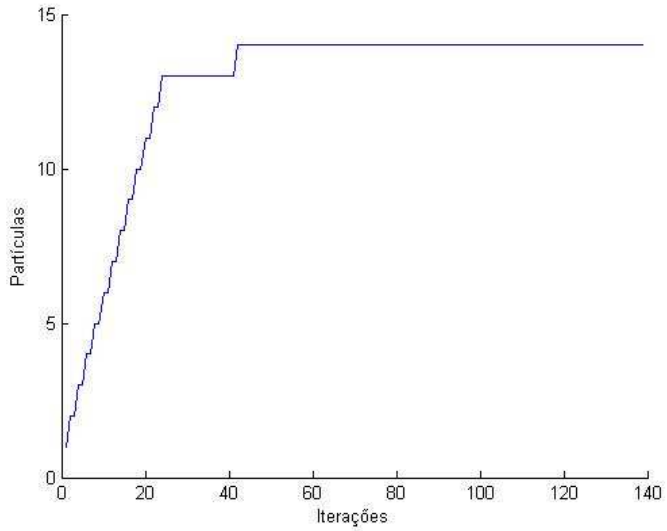


Figura 4.7: Crescimento do enxame em uma execução do algoritmo cPSC aplicado a base de dados Reuters 500.

4.4.2 Agrupamento Ótimo por Enxame de Partículas (oPSC)

Para avaliar a significância estatística da diferença de desempenho dos algoritmos uma análise de variância (teste ANOVA) foi realizada com valor de confiança de 95%. Para todos os resultados da base Iris, a hipótese nula não pode ser rejeitada, o que significa que os valores de Entropia e Pureza são equivalentes. Para os

resultados de Entropia e Pureza da base Yeast foi possível concluir que o algoritmo *K-Médias* teve um melhor desempenho, já nos resultados de Entropia e Pureza da base Ruspini foi possível concluir que o algoritmo oPSC utilizando a função que maximiza a soma das distâncias intracluster (oPSC MI) teve um melhor desempenho. Para os resultados de Entropia e Pureza da base Diabetes foi possível concluir que os valores do oPSC MI e do *K-Médias* são equivalentes. As Tabelas 4.13 e 4.14 mostram os melhores e piores valores absolutos de Entropia e Pureza encontrados para cada base de dados.

Tabela 4.13: Melhor resultado de Entropia e Pureza encontrado para cada base de dados.

Melhor Resultado				
	Entropia	**Algoritmo**	**Pureza**	**Algoritmo**
Ruspini	0,0	oPSC SI / oPSC MI	1,0	oPSC SI / oPSC MI
Iris	0,27	oPSC MI	0,87	oPSC MI
Diabetes	0,0	*K-Médias* / oPSC MI	1,0	*K-Médias* / oPSC MI
Yeast	0,05	*K-Médias*	0,98	*K-Médias*

Tabela 4.14: Pior resultado de Entropia e Pureza encontrado para cada base de dados.

Pior Resultado				
	Entropia	**Algoritmo**	**Pureza**	**Algoritmo**
Ruspini	0,44	*K-Médias*	0,64	*K-Médias*
Iris	0,49	oPSC MI	0,66	*K-Médias* / oPSC MI
Diabetes	0,93	oPSC SI	0,65	oPSC SI
Yeast	0,68	oPSC SI	0,50	oPSC SI

O algoritmo oPSC obteve melhores resultados absolutos de Entropia e Pureza para as bases Ruspini e Iris. O oPSC também dividiu os melhores resultados com o *K-Médias* para a base Diabetes. Considerando a base Yeast o algoritmo *K-Médias* obteve o melhor resultado absoluto. O algoritmo oPSC utilizando a silhuetta como função (oPSC SI) obteve a maioria dos piores resultados. Para os resultados de Pureza da base Iris o algoritmo *K-Médias* e o oPSC MI obtiveram os mesmos piores valores. O oPSC MI obteve o pior resultado de entropia para a base Iris, e o *K-Médias* obteve os piores resultados de Entropia e Pureza para a base Ruspini.

5 Conclusões e Trabalhos Futuros

Nesta dissertação foram investigadas as tarefas de agrupamento de dados e agrupamento de textos. Algoritmos de inteligência de enxame, mais especificamente, algoritmos que usam enxames de partículas, foram utilizados para formalizar conceitos sobre tais tarefas. Dois novos algoritmos de agrupamento foram propostos:

1. cPSC: uma variação do algoritmo PSC para agrupamento de dados textuais que detecta automaticamente o número de grupos em uma base de dados sem a necessidade de passar esse valor como parâmetro de entrada; e
2. oPSC: uma variação do algoritmo PSC capaz de propor agrupamentos ótimos de uma base de dados considerando, para isso, funções de custo explícitas que determinam a distância intra- e/ou intergrupo de cada solução candidata;

Os algoritmos foram aplicados a bases da literatura e seus desempenhos foram comparados a um algoritmo clássico da literatura, o *K-Médias*. Os resultados mostraram que os algoritmos propostos (cPSC, oPSC) são competitivos, possuindo, portanto, potencial de aplicação em tarefas de agrupamento de dados.

Dentre as possíveis extensões e trabalhos futuros destacam-se:

1. Aplicação do algoritmo oPSC para *Mineração de Dados* textuais;
2. Utilização de outras funções de *fitness* no algoritmo oPSC;
3. Testes exaustivos das ferramentas utilizando outras bases de dados;
4. Comparação do desempenho das ferramentas com outros algoritmos da literatura;
5. Identificação das características dos problemas para os quais os algoritmos propostos desempenham melhor ou pior;
6. Adaptação dos algoritmos desenvolvidos para aplicação em tarefas de agrupamento nebuloso (*fuzzy*) de dados; e
7. Avaliação do potencial de aplicação dos algoritmos para dados variantes no tempo.

5.1 PUBLICAÇÕES ASSOCIADAS

- PRIOR, A. K. F.; DE CASTRO, L. N.; de FREITAS, L. R.; SZABO, A. (2009) The Proposal of two Bio-Ispired Algorithms for Text Clustering. Learning and Nonlinear Models, v. 6, pp. 29-43.
- SZABO, A.; PRIOR, A. K. F., DE CASTRO, L. N. (2010). The Behavior of Particles in the Particle Swarm Clustering Algorithm, In Proceedings of the IEEE World Congress on Computational Intelligence (FUZZY 2010), Barcelona, Spain, v. 1. pp. 3034-3040.
- SZABO, A.; PRIOR, A. K. F., DE CASTRO, L. N. (2010) The Proposal of a Velocity Memoryless Clustering Swarm, In Proceedings of the IEEE World Congress on Computational Intelligence (CEC 2010), Barcelona, Spain, v. 1. pp. 4342-4346.
- PRIOR, A. K. F., DE CASTRO, L. N. (2010) cPSC: Um Algoritmo de Enxame Construtivo para Agrupamento de Dados, Proceedings do XVIII Congresso Brasileiro de Automática 2010, Bonito, MS, pp 3300-3307.

REFERÊNCIAS BIBLIOGRÁFICAS

ARANHA, C.; PASSOS, E. (2006), "A Tecnologia de *Mineração de Textos*", *Revista Eletrônica de Sistemas de Informação*, Vol. 5, (2).

BAEZA-YATES, R.; RIBEIRO-NETO, B. (1999), *Modern Information Retrieval*, ACM Press.

BENI, G.; WANG, J. (1989), *Swarm Intelligence*, Proc. Of the 7th Annual Meeting of the Robotics Society of Japan, pp. 425-428.

BERRY, M. W.; CASTELLANOS, M. (2007) *Survey of Text Mining II: Clustering, Classification, and Retrieval*, V. 2, Springer.

BONABEAU, E.; DORIGO, M.; THERAULAZ, T. (1999), *Swarm Intelligence: From Natural to Artificial Systems*, New York: Oxford University Press.

CHAKRABARTI, S. (2002), *Mining the Web: Discovering Knowledge from Hypertext Data*, Morgan-Kauffman.

CHAVES, M. S. (2004), *Um estudo e apreciação sobre algoritmos de stemming para a língua portuguesa*, IX Jornadas Iberoamericanas de Informática, Cartagena de Indias, Colômbia.

CLERC, M. (1999). *The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization*. Proc. I999 ICEC, Washington, DC, pp 1951 - 1957.

CRABTREE, D.; GAO, X.; ANDREAE, P. (2005), *Standardized Evaluation Method for Web Clustering Results*, 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), pp.280-283.

DE CASTRO, L. N.; VON ZUBEN, F. J.; DE DEUS JR., G. A. (2003), "The construction of a boolean competitive neural network using ideas from immunology". *Neurocomputing*, 50, pp. 51–85.

COHEN, S. C. M.; DE CASTRO, L., N. (2006), *Data Clustering with Particle Swarms*. In: International Conference on Evolutionary Computation, Proceedings of World Congress on Computational Intelligence, pp. 6256-6262.

DE CASTRO, L. N. (2006), *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*, Chapman & Hall/CRC.

DE CASTRO, L., N. (2007), Fundamentals of Natural Computing: An Overview, *Physics of Life Reviews*, 4, pp. 1-36.

DENEUBOURG, J. L.; GOSS, S.; FRANKS, N.; SENDOVA-FRANKS, A.; DETRAIN, C.; CHRÉTIEN, L. (1991), The Dynamics of Collective Sorting: Robot-Like Ant and Ant-Like Robot, In J. A. Meyer and S. W. Wilson (eds.) Simulation of Adaptive Behavior: From Animals to Animals, pp. 356 – 365, MIT Press/Bradford Books, Cambridge, MA.

DORIGO, M. (1992), *Optimization, Learning and Natural Algorithms*, (in Italian), Ph.D. Thesis, Dipartimento di Elettronica, Politecnico di Milano, IT.

DORIGO, M.; STÜTZLE, T., (2004), *Ant Colony Optimization*, MIT Press.

EBERHART, R., SHI. Y. (2000), *Comparing Inertia Weights and Constriction Factors in Particle Swarm Algorithm*. In: Proceedings of the 2000 IEEE Congress on Evolutionary Computation, Piscataway, NJ, IEEE Press pp. 84-88.

EVERITT, B.S.; LANDAU, S.; LEESE, M. (2001), *Cluster Analysis*, Arnold Publishers, London.

FREITAS, L, R. (2008), *Algoritmos Bio-Inspirados para Agrupamento de Textos com Aplicação em Recomendação de Conteúdo*, Dissertação de Mestrado em Ciências da Computação, Universidade Católica de Santos.

HAN, J.; KAMBER, M. (2000), *Data Mining: Concepts and Techniques*, Morgan Kaufman.

HOTHO, A.; NÜRNBERGER, A.; PAAß G. (2005), "A Brief Survey of Text Mining", *GLDV-Journal for Comp. Linguistics and Language Technology*, 20(1), pp. 19-62.

JOACHIMS, T. (1997), Text Categorization with Support Vector Machines: Learning with Many Relevant Features. LS8-Report 23, Universität Dortmund, LS VIII-Report.

KAUFMAN, L.; ROUSSEEUW, P. J. (1990), *Finding Groups in Data – An Introduction to Cluster Analysis*, Wiley Series in Probability and Mathematical Statistics.

KENNEDY, J. (2004), *Particle Swarms: Optimization Based on Sociocognition*, In L. N. de Castro and F. J. Von Zuben, Recent Developments in Biologically Inspired Computing, Idea Group Publishing, Chapter 10, pp. 235-269.

KENNEDY, J.; EBERHART, R., SHI. Y. (2001), *Swarm Intelligence*, Morgan Kaufmann Publishers.

KENNEDY, J.; EBERHART, R. (1995), *Particle Swarm Optimization*, Proc. of the IEEE Int. Conf. on Neural Networks, Perth, Australia, 4, pp. 1942–1948.

KNIDEL, H. (2006), *Extensoes e Aplicações de Redes Neuro-Imunologicas*, Dissertação de Mestrado em Engenharia da Computação, Faculdade de Eng. Eletrica e Comp. da Universidade de Campinas.

LANG, K. (1995), *Newsweeder: Learning to filter netnews*. In Proceedings of the Twelfth International Conference on Machine Learning, pp 331–339.

LEWIS, D. (2004), Reuters-21578 Text Categorization Text Collection, 2004. Disponível em: < http://dit.unitn.it/~moschitt/corpora.htm />. Acesso em: 04 nov. 2010.

MACQUEEN, J. B. (1967), *Some Methods for classification and Analysis of Multivariate Observations*, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1, pp: 281-297.

MONTEIRO, L. O.; GOMES, I. R.; OLIVEIRA, T. (2006), *Etapas do Processo de Mineração de Textos – uma abordagem explícita a textos em Português do Brasil*. Proc I Workshop de Computação e Aplicações. Campo Grande, Brasil.

MORAIS, E. A. M. (2007), *Contextualização de Documentos em Domínios Representados por Ontologias Utilizando Mineração de Textos*, Universidade Federal de Goiás Instituto de Informática.

PARHI, D. R.; POTHAL, J. K.; SINGH, M. K. (2009), *Navigation of Multiple Mobile Robots Using Swarm Intelligence*, Nature & Biologically Inspired Computing, 2009. NaBIC 2009. pp. 1145-1149.

POLI, R.; KENNEDY, J.; BLACKWELL, T. (2007), "Particle Swarm Optimization: An overview". *Swarm Intelligence*, V. 1, pp. 33-57.

PRIOR, A. K. F.; de CASTRO, L. N.; FREITAS, L. R.; SZABO, A. (2009), "The proposal of two bio-inspired algorithms", *Learning and Nonlinear Models*, Vol 6, (1).

PRIOR, A. K. F., DE CASTRO, L. N. (2010) *cPSC: Um Algoritmo de Enxame Construtivo para Agrupamento de Dados*, Proceedings do XVIII Congresso Brasileiro de Automática 2010, Bonito, MS, pp 3300 – 3307.

REATEGUI, E. B.; CAZELLA, S. C. (2003), *Sistemas de Recomendação*, XXV Congresso da Sociedade Brasileira de Computação.

RESNICK, P.; VARIAN, H. R. (1997), *Recommender Systems*, Communications of the ACM, 40(3), pp.56-58.

SALTON, G.; WONG, A.; YANG, C. S. (1975), "A Vector Space Model for Information Retrieval", *Journal of the American Society for Information Science*, 18(11), pp. 613-620.

SEBASTIANI, F. (2002), "Machine learning in automated text categorization". *ACM Computing Surveys*, 34(1), pp. 1-47.

STÜTZLE, T.; DORIGO, M. (1999), *ACO Algorithms for the Traveling Salesman Problem*, Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications. John Wiley & Sons.

SZABO, A.; PRIOR, A.K.F.; de CASTRO, L.N. (2010a), *The Behavior of Particles in the Particle Swarm Clustering Algorithm*, In Proceedings of the IEEE World Congress on Computational Intelligence (FUZZY 2010), Barcelona, Spain, v. 1, pp 3034-3040.

SZABO, A.; PRIOR, A.K.F.; de CASTRO, L.N. (2010b), *The Proposal of a Velocity Memoryless Clustering Swarm*, In Proceedings of the IEEE World Congress on Computational Intelligence (CEC 2010), Barcelona, Spain, v. 1, pp 4342-4346.

UCI Repository of Machine Learning Databases, Neural Networks Benchmarks. Disponível em: < http://archive.ics.uci.edu/ml/datasets.html />. Acesso em: 04 nov. 2010.

WEISS, S.; INDURKHYA, N. (1999), *Predict Data Mining*, Morgan Kauffman Publishers, Inc.

WEISS, S.; INDURKHYA, N.; ZHANG, T.; DAMERAU, F. (2004), *Text Mining: Predictive Methods for Analyzing Unstructured Information*, Springer.

WITTEN, I. H.; FRANK, E. (2005), *Data Mining – Practical Machine Learning Tools and Techniques*, Morgan Kaufman Publishers, USA.

ZHAO, Y.; KARYPIS, G. (2004), *Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering*, Machine Learning, 55(3), pp. 311-331.