OSIRES PIRES COELHO FILHO

Uma nova abordagem híbrida do algoritmo de Otimização por Enxame de Partículas com Busca Local Iterada para o problema de Clusterização de Dados

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Algoritmos e Otimização.

Orientador:
Carlos Alberto Martinhon

Coorientador:
Lucídio dos Anjos Formiga Cabral

UNIVERSIDADE FEDERAL FLUMINENSE

NITEROI-RJ
2013

Ao meu pai Osires Pires Coelho *(in memoriam).*

Agradecimentos

A Deus.

Ao meu pai, que faleceu no transcorrer deste trabalho, mas que antes de falecer disse-me palavras motivadoras que fizeram toda a diferença nos momentos mais difíceis e que me deram forças pra continuar a jornada.

À minha mãe, por tudo, e pela vida de sacrifícios por mim e por meu irmão, e que possibilitou que eu tivesse a educação necessária para pleitear este mestrado.

À minha querida filha Isadora, que nasceu praticamente junto com o mestrado e que só em pensar na sua existência já conseguia forças para superar o que parecia ser insuperável.

Ao meu filho Osires Neto, que me incentivou bastante, mesmo sem querer, me fazendo perguntas, sempre curioso para saber do que se tratavam os Enxames de Partículas.

À Delma, minha esposa, que com sua serenidade sempre lembrava da importância deste mestrado para mim e para nossa família.

Ao professor Carlos Alberto Martinhon, pela sua orientação, sabedoria, paciência e principalmente por me oferecer um voto de confiança para que eu pudesse prosseguir com este trabalho.

Ao professor Lucídio Formiga Cabral, pela coorientação, ensinamentos, sugestões, incentivos e especialmente por usar sua larga experiência e inteligência de forma a contribuir decisivamente para o êxito desta pesquisa.

Enfim, a todos os amigos, parentes e funcionários que me incentivaram e contribuíram de alguma forma para a finalização deste trabalho.

Resumo

Clusterização ou Agrupamento de Dados é uma das técnicas mais conhecidas em Mineração de Dados. Um problema de Clusterização consiste em agrupar objetos de uma base de dados em grupos de objetos similares, de acordo com um conjunto de características, com o objetivo de reconhecer padrões existentes nos dados. Esse processo apresenta uma complexidade de ordem exponencial e exige heurísticas eficientes que permitam resolvê-lo nem um tempo computacionalmente aceitável, fornecendo soluções próximas à uma solução ótima. Para resolver esse problema, tem-se usado com bastante êxito algoritmos baseados em princípios de Inteligência de Enxames como a meta-heurística *Particle Swarm Optimization* (PSO) e suas variações e hibridizações. Neste trabalho é proposta uma nova abordagem híbrida do uso das meta-heurísticas PSO e *Iterated Local Search* (ILS) para resolver o problema de Clusterização de Dados. Nesta estratégia, a convergência do PSO é melhorada através da chamada ao ILS que usa o *K-means* como busca local. Este algoritmo híbrido ainda inclui um número caótico para determinar o valor do peso inercial do componente velocidade inercial da partícula. Esta versão denominada EPSO-ILS (*Enhanced* PSO-ILS) apresentou excelentes resultados quando testada em várias bases de dados de referência da área, obtendo melhores valores para todas as instâncias avaliadas.

Abstract

Clustering or Data Group is one of the most well-known techniques in Data Mining. A Clustering problem consists on getting objects together of a database in similar objects groups, according to a set of features with the goal of recognizing existing patterns in data. This process shows a complexity of exponential order and demands efficient heuristic that allows to solve it in an acceptable time, providing close solutions to a great solution. To solve this problem, it has been successfully used algorithms based on Swarm Intelligence principles as the metaheuristics *Particle Swarm Optimization* (PSO) and its variations and hybridizations. It is proposed in this paper a new hybrid approach of the use of the PSO metaheuristic and *Iterated Local Search* (ILS) to solve the Data Clustering problem. In this strategy, the convergence of PSO is improved by the so-called ILS using the *K-means* as local search. This hybrid algorithm still includes a chaotic number to determine the value of the inertial weight of the initial velocity component of the particle. This version entitled EPSO-ILS (*Enhanced* PSO-ILS) showed excellent results when tested in several database of the area reference, achieving better values to all the assessed instances.

Palavras chave

1. Otimização por Enxame de Partículas
2. *Particle Swarm Optimization* - PSO
3. Clusterização de Dados
4. Busca Local Iterada
5. *Iterated Local Search* - ILS
6. Números Caóticos

Glossário

ACPSO - *Acelerated Chaotic Particle Swarm Optimization*

CPSO - *Chaotic Particle Swarm Optimization*

DCBD - Descoberta de Conhecimento em Bases de Dados

EPSO-ILS - *Enhanced Particle Swarm Optimization- Iterated Local Search*

EPSO - *Enhanced Particle Swarm Optimization*

GA - *Genetic Algorithm*

GBest - *Global Best*

ILS - *Iterated Local Search*

ILS-KM - *Iterated Local Search-K-Means*

KDD - *Knowledge Discovery in Databases*

KGA - K-Means Genetic algorithms

LBest - *Local Best*

MatLab - *Matrix Laboratory*

PBest - *Personal Best*

PCA - Problema de Clusterização Automática

PSO - *Particle Swarm Optimization*

SA - *Simulated Annealing*

Lista de Figuras

Lista de Tabelas

Lista de Gráficos

Lista de Algoritmos

Sumário

Capítulo 1

INTRODUÇÃO

O presente trabalho visa a pesquisa e o desenvolvimento de um algoritmo baseado na hibridização de duas meta-heurísticas denominadas Otimização por Enxame de Partículas - *Particle Swarm Optimization* (PSO) e Busca Local Iterada - *Iterated Local Search* (ILS), aplicado à resolução do problema de Clusterização de Dados.

Nesta estratégia, a convergência do PSO é melhorada através da chamada do ILS usando o *K-means* como busca local. Este algoritmo híbrido ainda inclui um número caótico para determinar o valor do peso inercial a ser aplicado no componente velocidade inercial da partícula. Esta versão denominada EPSO-ILS (*Enhanced* PSO-ILS) apresentou excelentes resultados quando testada em várias bases de dados de referência da área, obtendo, na maioria das vezes, melhores valores para todas as instâncias avaliadas.

1.1 O problema de Clusterização de Dados

Analisar e extrair conhecimento existente na imensa quantidade de dados armazenada nos bancos de dados atuais é uma tarefa que extrapola a capacidade humana e torna-se um grande desafio inerente a quase todas as áreas do conhecimento. Este desafio impõe a criação de ferramentas, técnicas e métodos eficientes de forma a automatizar a tarefa de extrair e analisar este grande volume de dados.

A Clusterização de Dados é uma das técnicas automatizadas mais conhecidas e utilizadas para extrair conhecimento em grandes bases de dados. Ela consiste na classificação não-supervisionada de padrões (observações, itens de dados ou vetores de características) em grupos (*clusters),* ou seja, um problema de clusterização consiste em agrupar objetos de uma base de dados em grupos de objetos similares (HAN e KAMBER, 2001; COHEN e CASTRO, 2006) de acordo com um conjunto de características com o objetivo de reconhecer padrões existentes nos dados.

1.2 Motivação

O problema de Clusterização de Dados é abordado em muitos contextos, por pesquisadores das mais variadas áreas, incluindo por exemplo: computação visual e gráfica, computação médica, biologia computacional, redes de comunicação, engenharia de transportes, redes de computadores, sistemas de manufatura, dentre outras. Isto reflete o seu amplo apelo e utilidade como uma das etapas da Análise Exploratória de Dados em Mineração de Dados. Todavia, a resolução deste problema exige o processamento de muitas combinações de grupos e objetos até que seja encontrada uma disposição ideal e, portanto, apresenta uma complexidade de ordem exponencial. Isto significa que métodos computacionais que utilizam a "força bruta", para enumerar todos os possíveis grupos e escolher a melhor configuração exigiriam um altíssimo custo computacional sendo, portanto, inviáveis na prática. É necessário, então, buscar uma heurística eficiente que permita resolver o problema num tempo aceitável.

Algoritmos baseados nos paradigmas de Inteligência de Enxames e Computação Evolucionária têm sido propostos (MILLONAS, 1994) para resolver muitos problemas de otimização, em especial problemas de Clusterização de Dados. O representante mais popular destas técnicas, e que vem obtendo mais êxito, é a meta-heurística Otimização por Enxame de Partículas ou *Particle Swarm Optimization* (PSO) (MERWE; ENGELBRECHT, 2003).

Inúmeras pesquisas (RANA *et al*, 2011) demonstraram que o PSO padrão aplicado à Clusterização de Dados possui um ótimo desempenho quando comparado aos demais algoritmos usados para o mesmo fim. No entanto, a taxa de convergência na busca pelo ótimo global ainda é passível de melhora.

Várias hibridizações e modificações no PSO básico foram introduzidas para melhorar a velocidade de convergência e a qualidade das soluções encontradas (KAO *et al*, 2008). Neste sentido, buscou-se desenvolver um algoritmo capaz de melhorar ainda mais o poder do PSO através de uma

hibridização com outra meta-heurística, a ILS, com o objetivo de melhorar a eficiência do processo de Clusterização de Dados.

1.3 Objetivos

Nesta seção descreve-se o objetivo geral e objetivos específicos que nortearam o desenvolvimento deste trabalho.

1.3.1 Objetivo Geral

O principal objetivo deste trabalho é desenvolver um algoritmo baseado nas meta-heurísticas Otimização por Enxame de Partículas *(Particle Swarm Optimization* - PSO) e Busca Local Iterada (*Iterated Local Search* - ILS), denominado EPSO-ILS para melhorar o processo de Clusterização de Dados.

1.3.2 Objetivos específicos

O trabalho tem como objetivos específicos os seguintes:

a) Realizar estudos sobre Clusterização de Dados, PSO e ILS;

b) Realizar estudos sobre o estado da arte de algoritmos de PSO para Clusterização de Dados;

c) Avaliar os efeitos e influência dos parâmetros utilizados no PSO;

d) Desenvolver algoritmos utilizando a heurística *K-means* e as meta-heurísticas PSO e ILS para o problema de Clusterização de Dados;

e) Desenvolver o algoritmo EPSO-ILS;

f) Verificar e comparar o desempenho do algoritmo proposto, EPSO-ILS, com os algoritmos existentes na literatura para Clusterização de Dados.

1.4 Estrutura do trabalho

Este trabalho se divide em cinco capítulos, incluindo o Capítulo 1, referente à Introdução. No Capítulo 2 encontra-se a Fundamentação Teórica, onde é apresentado um estudo sobre o problema de Clusterização de Dados e as meta-heurísticas PSO e ILS, usadas no algoritmo proposto neste trabalho. No Capítulo 3 encontra-se toda a estratégia utilizada para o desenvolvimento do algoritmo proposto para se melhorar o processo de Clusterização de Dados e, em seguida, no Capítulo 4, são apresentados os Resultados Computacionais obtidos com a execução do algoritmo proposto, para Clusterizar cinco bases de dados de referência da literatura. Em seguida, faz-se uma análise e discussão de tais resultados. No Capítulo 5, são apresentadas as conclusões e as sugestões para trabalhos futuros a serem desenvolvidos com a utilização do algoritmo proposto.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

Este capítulo se divide em quatro seções. Na primeira seção apresenta-se um estudo detalhado sobre o problema de Clusterização de Dados. Na segunda seção faz-se um levantamento sobre o estado da arte do algoritmo PSO e suas variações. Na terceira seção faz-se uma breve apresentação da meta-heurística ILS que será usada no algoritmo proposto e, por fim, na quarta seção, são apresentados os trabalhos relacionados na literatura sobre uso do PSO, suas hibridizações e variações para resolução de problemas de Clusterização de Dados. Este capítulo servirá como base para o entendimento do algoritmo proposto, visto que no referido algoritmo utiliza-se as meta-heurísticas PSO e ILS para a resolução de problemas de Clusterização de Dados.

2.1 Clusterização de Dados

O principal objetivo desta seção é apresentar uma visão geral do processo de Clusterização de Dados e a forma como a literatura o aborda. Descreve-se a técnica de clusterização como uma das etapas da Mineração de Dados, determinando de forma concisa o problema de clusterização, suas aplicações, tipos de clusterização, métodos utilizados para clusterizar dados, classes de problemas de clusterização, medidas de similaridade, metodologias e principais técnicas ou algoritmos determinísticos utilizados para resolver o problema atualmente.

2.1.1 Contextualização

Clusterização de Dados é a classificação não-supervisionada de padrões (observações, itens de dados ou vetores de características) em grupos (*clusters),* ou seja, um problema de clusterização consiste em agrupar objetos de uma base de dados em grupos de objetos similares (HAN e KAMBER, 2001) de acordo com um conjunto de características com o objetivo de reconhecer padrões existentes nos dados.

A Clusterização de Dados é uma das técnicas automatizadas que fazem parte da etapa de Mineração de Dados, no processo maior conhecido como Descoberta de Conhecimento em Bases de Dados (DCBD) ou *Knowledge Discovery in Databases* (KDD). Segundo Fayyad *et al*. (1996), KDD é o processo não trivial de identificação, a partir de dados, de padrões que sejam válidos, novos, potencialmente úteis e compreensíveis. O processo de descoberta de conhecimento em bases de dados tem como objetivo a utilização de mecanismos automáticos de extração de conhecimento que auxilie na tomada de decisões (ROMÃO, 2002). O KDD focaliza o processo global de descoberta de conhecimento através dos dados, e é composto das etapas de Seleção, Pré-Processamento, Formatação, Mineração de Dados e Interpretação/Avaliação (pós-processamento), conforme ilustrado na Figura 1 e descritas a seguir.

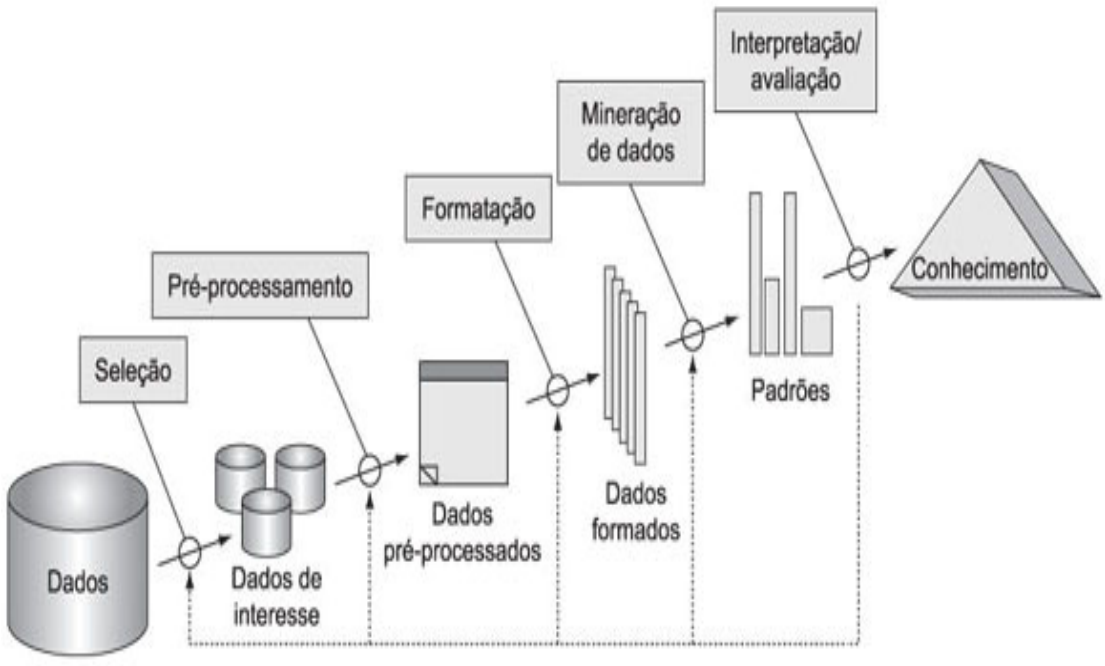

Figura 1 – Etapas que compõem o processo de KDD segundo Fayyad *et al* (1996).

- **Seleção:** nesta etapa ocorre a seleção dos dados que serão utilizados em todo o processo de KDD. Os dados podem ter origem em várias fontes de dados, tais como documentos, planilhas, bancos de dados, tabelas, mapas, etc.

- **Pré-processamento:** nesta etapa ocorre a preparação dos dados para o processamento e o controle de sua qualidade, para evitar possíveis inconsistências na fase de mineração.

- **Formatação:** etapa na qual são criadas as estruturas de dados a serem utilizadas na fase de mineração.

- **Mineração de Dados:** principal etapa do KDD, na qual são usadas técnicas automatizadas, algoritmos, com o objetivo de extrair os padrões de interesse.

- **Interpretação/Avaliação (pós-processamento):** fase onde os especialistas avaliam os resultados obtidos na fase de Mineração e identificam os padrões extraídos para sua posterior utilização prática.

Dentre as etapas do KDD descritas acima, está a Mineração de Dados, que consiste na gestão de algoritmos utilizados para extrair e interpretar padrões nos dados. Segundo Berry e Linoff (1997), Mineração de Dados é a exploração e a análise de dados, por meios automáticos ou semi automáticos, de grandes quantidades de dados, com o objetivo de descobrir regras ou padrões interessantes. As técnicas automatizadas de Mineração de Dados utilizam uma diversidade de algoritmos para identificar relacionamentos e padrões que estão implícitos nos dados. Dentre elas estão as técnicas de agrupamento ou Clusterização de Dados.

2.1.2 Motivação

Nas últimas décadas, a Clusterização de Dados, também chamada de Análise de *Clusters,* tem desempenhado um papel central em uma variedade de campos de diversas ciências, e pesquisadores de todo o mundo estão desenvolvendo novos algoritmos para atender à crescente complexidade dos grandes conjuntos de dados existentes em aplicações do mundo real.

Segundo Oliveira e Carvalho (2008) as aplicações das tarefas de agrupamento são as mais variadas possíveis: pesquisa de mercado, reconhecimento de padrões, processamento de imagens, análise de dados, segmentação de mercado, taxonomia de plantas e animais, pesquisas geográficas, classificação de documentos da *web*, detecção de comportamentos atípicos (fraudes), dentre outras.

2.1.3 Visão Geral

A palavra clusterização vem do inglês, do verbo *to cluster* que em português significa agrupar. Desta forma, Clusterização de Dados nesse contexto, ou seja, no contexto de Mineração de Dados, significa Agrupamento de Dados e consiste em reunir dados ou objetos com características similares em grupos ou *clusters.* Segundo Han e Kamber (2001), *clusterização* ou agrupamento é o processo de agrupar os dados em classes ou *clusters* (grupos) de forma que os objetos dentro de um *cluster* tenham alta similaridade em comparação uns com os outros, mas sejam bem dissimilares para objetos em outros *clusters*. Outra definição, agora segundo Jain *et al*. (1999) diz que clusterização é a classificação não supervisionada de dados, formando agrupamentos ou *clusters*. Ela representa uma das principais etapas de processos de análise de dados, denominada análise de *clusters*. É necessário distinguir aqui classificação supervisionada de classificação não supervisionada: na primeira, são fornecidos padrões rotulados (pré-classificados) e o problema é rotular novos padrões ainda não rotulados, na segunda, o problema é agrupar um conjunto de padrões não rotulados em *clusters* que possuam algum significado, de tal modo que os padrões apresentem alguma propriedade comum. Desta forma, uma vez definidos os *clusters* , os padrões também estarão “rotulados”, e neste caso os rótulos são determinados pelos padrões que compõem cada cluster. Clusterização é uma forma de aprendizado por observação, em lugar de aprendizado por exemplos. Ao agrupar dados pode-se descrever de forma mais eficiente e eficaz as características dos diversos grupos, o que permite um maior entendimento do conjunto de dados original, além de possibilitar o desenvolvimento de esquemas de classificação para novos dados. Com a utilização das técnicas de

clusterização, pode-se descobrir os padrões de distribuição global e correlações interessantes entre os atributos dos dados que não seriam facilmente visualizadas sem o emprego de tais técnicas.

Na prática, a Clusterização de Dados pode ser utilizada como uma ferramenta autônoma para obter pistas sobre a distribuição de dados, reconhecimento de padrões, ou como uma etapa de pré-processamento para outros algoritmos, como por exemplo no pré-processamento de imagens.

2.1.4 Etapas da clusterização

A atividade de Clusterização de Dados típica envolve a execução das seguintes etapas ou estágios segundo Jain e Dubes (1988):

- **Representação dos padrões:** envolve definição do número, tipo e modo de apresentação dos atributos que descrevem cada padrão;

- **Seleção de características:** processo de identificação do subconjunto mais efetivo dos atributos disponíveis para descrever cada padrão;

- **Extração de características:** uso de uma ou mais transformações junto aos atributos de entrada de modo a salientar uma ou mais características dentre aquelas que estão presentes nos dados;

- **Medida de similaridade:** é fornecida por uma função de distância definida entre pares de dados ou padrões. É possível incluir na medida de distância aspectos conceituais (qualitativos) ou numéricos (quantitativos);

- **Agregação ou Agrupamento:** os grupos ou *clusters* podem ser definidos como conjuntos *crisp* (puro) onde um padrão pertence ou não-pertence a um dado grupo) ou *fuzzy* (difuso) onde um padrão pode apresentar graus de pertinência a vários grupos. Conforme será visto na Seção 2.1.8, o processo de agrupamento pode ser hierárquico, com um processo recursivo de junções ou separações de grupos, ou não-

hierárquico, com o emprego direto de técnicas de discriminação de *clusters;*

- **Apresentação do resultado:** deve permitir que um computador possa utilizar o resultado de forma direta ou então deve ser orientada ao usuário, permitindo a visualização gráfica dos *clusters* e a compreensão de suas inter-relações, através da proposição de protótipos ou outras descrições compactas para os *clusters*. A realimentação do resultado do processo de clusterização pode levar à redefinição dos módulos de "extração ou seleção de características" e "medida de similaridade".

A Figura 2 a seguir ilustra bem as etapas acima de Clusterização de Dados:

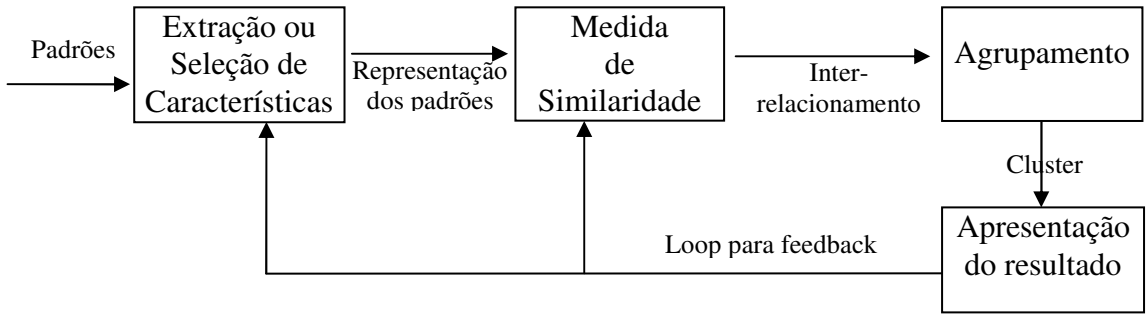

Figura 2 - Etapas da Clusterização de Dados segundo Jain e Dubes (1988)

2.1.5 Definição do problema de clusterização

Problemas de Clusterização podem ser definidos da seguinte forma: Dado um conjunto E com *m* elementos ou padrões, $E = \{x_1, x_2, \dots, x_m\}$, o problema de clusterização consiste na obtenção de um conjunto de *k clusters*, $C = \{C_1, C_2, \dots, C_k\}$, tal que os elementos contidos em um *cluster* C_i possuam uma maior similaridade entre si do que com os elementos de qualquer um dos demais *clusters* do conjunto C. O conjunto C é considerado uma clusterização com *k clusters* caso as seguintes condições sejam satisfeitas:

$$\bigcup_{i=1}^{k} C_i = E \tag{2.1.1}$$

$$C_i \neq \emptyset, para\ 1 \leq i \leq k$$

$$C_i \cap C_j = \emptyset,\ para\ 1 \leq i,j \leq k\ e\ i \neq j$$

Cada elemento ou objeto do conjunto E representa um padrão e é formado por um vetor que contem os valores que representam os atributos ou características do objeto, ou seja, $x_i = \{x_1, x_2, \ldots, x_d\}$, onde *i* = 1..m e *d* indica a quantidade de dimensões ou características do i-ésimo objeto do conjunto E. Um conjunto de padrões ou objetos é representado na maioria das vezes por uma matriz *m x d.*

Para determinar o quanto um objeto é similar a outro ou se eles pertencem a um determinado cluster, usa-se uma "medida de similaridade", que é específica para cada problema de clusterização a ser tratado. Esta medida geralmente é dada pela distância entre os objetos e quanto menor for a distância entre um par de objetos, maior é a similaridade entre eles. As medidas de similaridade são melhor explicadas na seção seguinte.

2.1.6 Medidas de similaridade

Agrupamentos ou *clusters* são formados por grupos de elementos que possuem maior similaridade entre si. Sendo assim precisa-se usar uma medida de similaridade para determinar se um elemento faz parte de um determinado grupo. A medida de similaridade mais utilizada é a distância, valor numérico que exprime as diferenças entre os atributos de cada elemento a ser clusterizado. Dois elementos com distância pequena entre eles possuem grande similaridade e portanto devem estar num mesmo grupo, enquanto que uma grande distância entre eles significa que são muito diferentes ou dissimilares e portanto devem estar em grupos diferentes (COLE, 1998). As principais medidas de distância usadas são: Distância Euclidiana, Distância de Minkowsk e Distância de Manhattan (quarteirões) (HAN e KAMBER, 2001).

A medida de similaridade mais utilizada é a Distância Euclidiana. Considerando que x_i e x_j são dois elementos do conjunto de dados E, *d* é a quantidade de atributos que cada elemento possui, x_{it} e x_{jt} são os valores do

t-ésimo atributo dos vetores x_i e x_j, a distância euclidiana é dada pela seguinte fórmula:

$$d(x_i, x_j) = \sqrt{\sum_{t=1}^{d} (x_{it} - x_{jt})^2}$$

(2.1.2)

Segundo Han e Kamber (2001), existem alguns atributos que podem, além de aumentar o custo computacional do algoritmo, prejudicar a precisão do resultado final do algoritmo. Por isso, nem todos os atributos devem ser, necessariamente, incluídos na Clusterização. Além disso, pesos podem ser atribuídos aos atributos de acordo com a sua relevância. A distância Euclidiana ponderada pode ser calculada da seguinte forma:

$$d(x_i, x_j) = \sqrt{\sum_{t=1}^{d} w_t (x_{it} - x_{jt})^2}$$

(2.1.3)

onde w_t é um peso atribuído à variável *t*. É importante observar que a unidade de medida utilizada pode afetar conclusivamente o resultado na clusterização. Uma das situações que podem prejudicar o processo de clusterização é quando se usa medidas de grandeza diferentes para a mesma característica. Por exemplo, a utilização de metros para quilômetros ou de gramas e quilos pode afetar o desempenho do processo. Para evitar esse problema, os dados devem ser normalizados.

Há casos de clusterização em que a distância não pode ser utilizada, ou não é conveniente que seja utilizada como medida de similaridade, tendo em vista que os valores dos atributos não são variáveis escalares. Por exemplo, ao tratar um problema de clusterização que envolve atributos como sexo ou endereço, são necessárias outras medidas que demonstrem o grau de similaridade entre as instâncias da base de dados. Outros tipos de variáveis

com as quais algoritmos de *cluster* podem lidar são: variáveis binárias, variáveis nominais, variáveis ordinais, variáveis escaladas em proporção, ou ainda combinações desses tipos de variáveis. Para estes tipos de variáveis outras medidas de similaridade têm que ser utilizadas. Neste trabalho é usado como medida de similaridade a distância euclidiana, pois as bases de dados utilizadas para executar o algoritmo proposto possuem somente valores numéricos ou atributos escalares.

2.1.7 Classes de problemas de clusterização

Existem basicamente duas classes de Problemas de Clusterização: problemas nos quais o número de *clusters* é previamente conhecido, denominados de Problemas de k-Clusterização, e problemas onde este valor não é informado ou conhecido e precisa ser determinado pelo algoritmo, denominado de Problema de Clusterização Automática (PCA). A primeira classe de problemas de clusterização é a classe mais estudada e é para esta classe de problemas que é proposta uma solução neste trabalho.

Para o Problema de *k*-Clusterização ou simplesmente Problema de Clusterização, o número total de diferentes formas de agrupamento de *n* elementos de um conjunto em *k clusters*, ou seja, o número de soluções possíveis, cresce exponencialmente e é dado por:

$$N(n,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^i \binom{k}{i} (k-i)^n \qquad (2.1.4)$$

Para ilustrar o crescimento exponencial do número de soluções possíveis para um problema de k-clusterização, considerando a Equação 2.1.4, foram combinados 10 elementos em 2 *clusters* , 100 elementos em 2 *clusters* , 100 elementos em 5 *clusters* e 1000 elementos em 2 *clusters*. Tem-se respectivamente os seguintes números de soluções possíveis:
$\mathrm{N}(10,2) = 511, \mathrm{N}(100,2) = 6{,}33825 \text{ x } 10^{29}, \mathrm{N}(100,5) =$
$6{,}57384 \text{ x } 10^{67}$ e $\mathrm{N}(1000,2) = 5.3575 \text{ x } 10^{300}$.

Para o problema de clusterização automática o número total de combinações aumenta ainda mais e pode ser definido de acordo com a

fórmula seguir:

$$N(n) = \sum_{k=1}^{n} \frac{1}{k!} \sum_{i=0}^{k} (-1)^i \binom{k}{i} (k-i)^n \qquad (2.1.5)$$

Dessa forma, para um conjunto com 10 elementos, a clusterização automática considera 115.975 diferentes maneiras de se combinar os elementos em um número de *clusters* que pode variar de 1 a 10.

2.1.8 Métodos para Clusterização de Dados

O processo de se agrupar objetos é normalmente bastante trabalhoso, conforme visto nas Equações 2.1.4 e 2.1.5, pois exige o processamento de muitas combinações de grupos e objetos até que seja encontrada uma disposição ideal. Este número de combinações cresce exponencialmente a medida que aumentam o número de objetos e *clusters*. Isto significa que métodos computacionais que utilizam a "força bruta", ou seja, métodos exatos, para enumerar todos os possíveis grupos e escolher a melhor configuração ou solução ótima exigem um altíssimo custo computacional e, portanto, não são viáveis. É necessário, então, buscar heurísticas eficientes que permitam resolver o problema em um tempo aceitável, fornecendo soluções próximas da ótima. Contudo, devido à grande diversidade das aplicações de problemas de clusterização, as heurísticas são normalmente desenvolvidas para determinadas classes de problemas, ou seja, não existe uma heurística que seja genérica a tal ponto que possa obter bons resultados em todas as aplicações de clusterização. Os métodos ou heurísticas utilizadas para Clusterização de Dados são os seguintes, segundo Han e Kamber (2001):

1. Métodos de particionamento: baseiam-se na construção de uma partição de um banco de dados em *k* grupos representados pelo valor médio dos objetos do grupo, ou por um objeto representativo do grupo que esteja localizado perto do seu centro, denominado centroide;

2. Métodos hieráquicos: esses métodos cria uma decomposição hierárquica de um dado conjunto de objetos e podem ser classificados como sendo aglomerativos *(bottom-up)* ou divisivos *(top-down)* de acordo com a maneira como a decomposição hierárquica é realizada;

3. Métodos baseados em densidade: a ideia geral desses métodos consiste no crescimento contínuo de um dado grupo até que a densidade (número de objetos) na vizinhança exceda um determinado limiar (*threshold*);

4. Métodos baseados em grade ou retícula: esses métodos quantificam o espaço do objeto em um número finito de células que formam uma estrutura de grade, onde todas as operações de agrupamento (clusterização) são realizadas;

5. Métodos baseados em modelos: criam um modelo hipotético para cada grupo, e a ideia geral é encontrar os objetos que mais se adaptem a esse modelo.

Os principais métodos utilizados para Clusterização de Dados são os métodos hierárquicos e de particionamento.

No método hierárquico os *clusters* vão sendo formados gradativamente através de aglomerações ou divisões de elementos/*clusters*, gerando uma hierarquia de *clusters* que são representados através de uma estrutura em árvore. Neste grupo de algoritmos, um *cluster* pode ser considerado como sendo composto por *clusters* menores.

No método de particionamento, que foi usado na solução proposta neste trabalho, um conjunto de elementos de dados ou objetos é dividido em *k* subconjuntos, onde *k* representa o número de *clusters* que é previamente definido pelo usuário. A divisão dos elementos em *k* sub-conjuntos se dá sem sobreposição, ou seja, cada objeto está exatamente em um grupo. Neste método cada grupo ou *cluster* é representado por um elemento denominado centro do *cluster* para que se possa definir a pertinência de um objeto ao *cluster*.

Existem duas formas de escolher o objeto que vai representar o centro

do *cluster*, e que será a referência para o cálculo da medida de similaridade. Na primeira forma, pode-se utilizar a média dos objetos que pertencem ao *cluster* em questão, este objeto é também chamado de centroide, esta é a abordagem conhecida como *K-means,* a ser detalhada na Seção 2.1.9.1. Na segunda forma de determinar o objeto representativo do *cluster,* pode-se escolher o objeto que se encontra mais próximo ao centro do cluster. Esta abordagem é conhecida como *K-medoids*, e o objeto mais próximo ao centro do *cluster* é chamado de *medoid* (medoide) e é vista na Seção 2.1.9.2. Nestes dois métodos os objetos migram entre os *clusters* até que nenhum objeto mude de *cluster* e, em consequência disto, os *clusters* melhoram gradativamente. Estes métodos também são referenciados na literatura como Otimização Iterativa. (BERKIN, 2002).

2.1.9 Algoritmos clássicos para Clusterização de Dados

Nesta seção serão mostrados os algoritmos determinísticos mais conhecidos da literatura para Clusterização de Dados utilizando-se o método de particionamento, são eles *K-means e K-medoids*.

2.1.9.1 O algoritmo *K-means*

Um dos métodos de particionamento mais conhecidos e utilizados para Clusterização de Dados é o *K-means* (MITRA e ACHARYA, 2004). O *K-means* é simples, fácil de implementar e muito eficiente para tratar uma grande quantidade de dados com complexidade de tempo linear.

O *K-means* utiliza um valor para representar cada *cluster*. Este valor é chamado de centroide e possui um valor médio para os atributos, relativos a todos os elementos do *cluster* considerado. Ele classifica os dados em um número de *clusters* previamente conhecido (*k clusters*). A ideia principal do algoritmo é estabelecer *k* centroides iniciais aleatoriamente e em seguida verificar de que centroide o objeto é mais similar, ou seja, está mais próximo, formando *clusters* ou agrupamentos destes objetos. Este processo é iterativo e a cada iteração os centroides são reposicionados calculando-se a média dos

valores dos elementos ou objetos mais próximos de cada centroide de acordo com a equação abaixo:

$$m^i = \frac{1}{\sigma_i} \sum_{\forall\, x^p \in C_i} x^p \tag{2.1.6}$$

Onde m^i equivale ao valor do centroide que representa o *cluster* C_i, *i = 1 .. k,* e tem a mesma estrutura de um objeto x^p. O valor σ_i é o número de elementos pertencentes ao *cluster* C_i, e x^p representa os atributos ou características de um objeto, ou seja, $x^p = \{x_1^p, x_2^p, \dots, x_d^p\}$, onde *d* indica a quantidade de dimensões ou características do p-ésimo objeto do *cluster i.*

Ao reposicionar os centroides, alguns pontos podem ficar mais próximos de outro centroide, e na próxima iteração farão parte de outro *cluster*.

Este método possui algumas desvantagens. A principal delas, segundo Metz (2006), é a necessidade de informar com antecedência o número de grupos ou *clusters* desejados. Outra desvantagem é quando a seleção aleatória inicial de centroides é ruim, ou seja, quando eles ficam mal posicionados, isso afeta significativamente o resultado da clusterização, e pode fazer com que a solução vá em direção a um máximo ou mínimo local de uma função de avaliação associada ao problema, como pode ser visto na Figura 3.

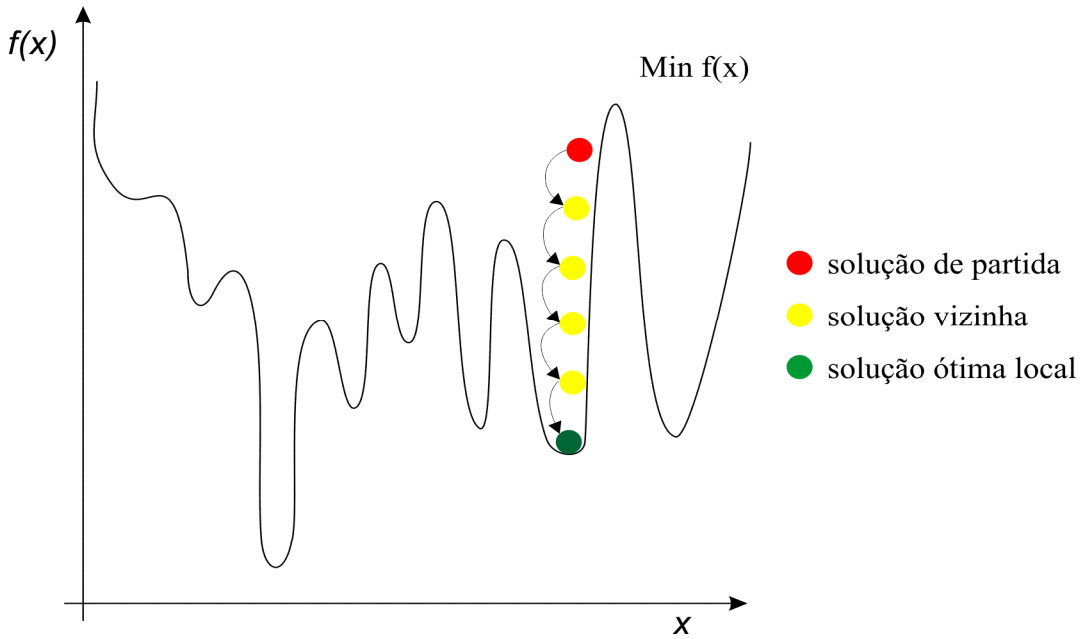

Figura 3 - Busca local

Para minimizar esse efeito negativo, usualmente os algoritmos são

executados diversas vezes com centroides iniciais diferentes, e, então, a melhor solução é atribuída ao resultado do processo de clusterização.

Abaixo é apresentado o algoritmo *K-means:*

Algoritmo *K_means*
Início

1 Receber a matriz de atributos dos pontos *n* x *d*;
2 Receber o valor *k* (número de *clusters);*
3 Inicializar aleatoriamente os *k* centroides;
4 **Repetir**
5 Atribuir todos os pontos ao centroide mais próximo de acordo com a distância euclidiana dada na Equação 2.1.2;
6 Recalcular/reposicionar centroides de acordo com Equação 2.1.6.
7 **Até** que os pontos não mudem de centroides;

Fim.

No Passo 1 é gerada uma tabela ou matriz onde cada linha representa um ponto ou objeto a clusterizar e as colunas representam os valores dos atributos ou dimensões da tabela. No Passo 2, o algoritmo recebe o número de clusters *k* em que se deseja agrupar os objetos da tabela. Em seguida, no Passo 3, os *k* centroides devem receber valores iniciais aleatórios, mas pode-se escolher os *k* primeiros pontos da tabela de objetos. Também é importante colocar todos os pontos em um centroide qualquer para que o algoritmo possa iniciar seu processamento. No Passo 5 é calculada a distância de todos os pontos a todos os centroides e então determinado a que centroide o ponto pertence, ou seja, está mais próximo. Esta é a parte mais pesada do algoritmo. No Passo 6 é feito o recálculo ou refinamento da posição dos centroides. Isso é feito calculando-se a média dos valores dos atributos dos pontos do *cluster* em cada dimensão, obtendo-se assim um novo valor médio para cada dimensão, o que resulta na mudança de posição do centroide. Caso o centroide não mude de posição, ou seja, nenhum ponto mudou de centroide, então o algoritmo encerra.

A seguir, na Figura 4, é apresentada uma ilustração de uma execução

de clusterização usando o algoritmo *K-means*.

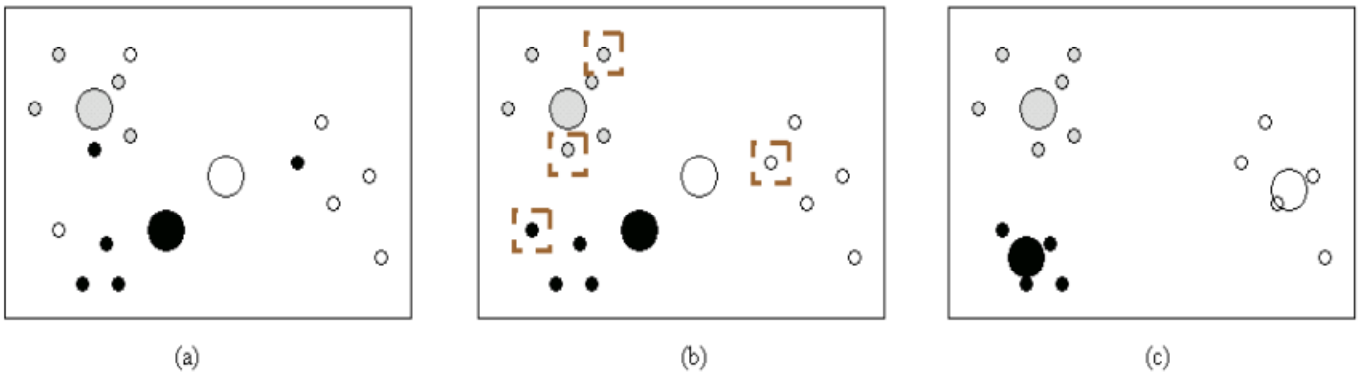

Figura 4 - Clusterização usando o *K-means*

Na Figura 4 cada círculo grande representa um centroide e cada círculo pequeno representa um objeto a clusterizar. Na fase (a), os centroides são posicionados de forma aleatória e cada ponto é atribuído a cada centroide também de forma aleatória. Na fase (b), primeira iteração, cada objeto recebe a identificação do centroide mais próximo e em seguida as posições dos centroides são recalculadas. Na fase (c), segunda iteração, os *clusters* já estão em sua forma final, pois nenhum ponto muda de *cluster* após o reposicionamento dos centroides.

2.1.9.2 O algoritmo *K-medoids*

O algoritmo *K-medoids* foi desenvolvido com o objetivo de diminuir a desvantagem da seleção de centroides iniciais ruins que pode ocorrer no *K-means.* Nele, ao invés de se utilizar o valor médio dos elementos em um *cluster* como centroide, usa-se o elemento melhor localizado no *cluster* denominado *medoid*. A estratégia do algoritmo *K-medoids* é encontrar *k clusters* em *n* elementos ou objetos primeiramente determinando o objeto representativo (*medoid*) para cada *cluster*. Cada objeto remanescente é atribuído ao *medoid* ao qual ele é mais similar. O algoritmo, então, iterativamente, troca um dos *medoids* por um dos não *medoids* caso a qualidade da *Clusterização* resultante seja melhorada com o uso do novo *medoid*. Esta qualidade é estimada usando

uma função custo que mede a dissimilaridade média entre um objeto e o *medoid* de seu *cluster*. A seguir o algoritmo *K-medoids* é apresentado.

Algoritmo_*Kmedoids*
Início

1 Gerar a Tabela de atributos dos pontos;
2 Receber o valor *k* (número de *clusters);*
3 Selecionar aleatoriamente, *k* objetos como os *medoids* iniciais;
4 **Repetir**
5 Atribuir cada objeto não *medoid* ao *cluster* com *medoid* mais próximo de acordo com Equação 2.1.2;
6 Selecionar aleatoriamente um objeto que não seja um *medoid;*
7 Recalcular o custo da função objetivo associada à clusterização;
8 Trocar o *medoid* caso a função objetivo melhore;
9 **Até** que os pontos não mudem de um *cluster* para outro;

Fim_algoritmo.

2.1.10 Outros algoritmos utilizados para clusterização

Como já foi dito anteriormente, o *K-means* possui uma série de desvantagens. Na tentativa de superar estas desvantagens outros algoritmos utilizando outras meta-heurísticas e hibridizações foram utilizados para resolver o problema de clusterização.

Genetic Algorithm (GA) ou Algoritmo Genético, desenvolvido por Holland (1975) e outros pesquisadores, e depois modificado por Goldberg (1986), foi utilizado para resolver problemas de clusterização. Entre estes algoritmos estão os métodos desenvolvidos por Maulik e Bandyopadhyay (2002) e também por Krishna e Murty (1999).

Existem também abordagens baseadas em *Simulated Annealing (SA)* para Clusterização de Dados, uma delas foi proposta por Selim e Alsultan (1991).

Em outros artigos encontram-se Khan e Ahmad (2004) que também propuseram um algoritmo para melhorar o *K-means*, criando uma rotina para calcular o centroide inicial. Steinley e Brusco (2007) propuseram vinte

procedimentos para inicializar o *K-means* e fizeram recomendações para melhores práticas. Ahalt *et al*. (1990) propuseram um algoritmo denominado FSCL para solucionar o problema de prisão em ótimos locais do *K-means*. Outro algoritmo nomeado de RPCL foi introduzido por Xu *et al*. (1993) para determinar com precisão o número de *clusters* para o *K-means*.

2.2 Otimização por Enxame de Partículas - *Particle Swarm Optimization* (PSO)

Nesta seção faz-se uma breve introdução à Inteligência de Enxames, seguida de uma visão geral da meta-heurística PSO, principal representante dos algoritmos baseados em Inteligência de Enxames, e os principais aspectos e variações encontrados no PSO básico, bem como os parâmetros utilizados para sua configuração.

2.2.1 Fundamentação teórica

Como já foi dito anteriormente, a resolução de problemas de Clusterização de Dados envolve o processamento de muitas combinações de grupos e objetos até que seja encontrada uma disposição ideal. Este processo apresenta uma complexidade de ordem exponencial. Isto significa que métodos computacionais que utilizam a "força bruta", para enumerar todos os possíveis grupos e escolher a melhor configuração exigiriam um altíssimo custo computacional e, portanto, não são viáveis. Torna-se necessário, então, buscar uma heurística eficiente que permita resolver o problema num tempo aceitável. Algoritmos baseados nos paradigmas de Inteligência de Enxames e Computação Evolucionária tem sido propostos (MILLONAS, 1994) para resolver muitos problemas de otimização, em especial problemas de Clusterização de Dados. O representante mais popular destas técnicas e que vem obtendo mais êxito é a meta-heurística Otimização por Enxame de Partículas ou *Particle Swarm Optimization* (PSO) (MERWE e ENGELBRECHT, 2003).

2.2.2 Inteligência de Enxames - *Swarm Intelligence (SI)*

Inteligência de Enxames é um campo relativamente novo de pesquisa interdisciplinar, que ganhou enorme popularidade nos últimos anos. Algoritmos pertencentes a este domínio, inspiram-se na inteligência coletiva que emerge do comportamento de um grupo de indivíduos socialmente organizados tais como insetos, pássaros, peixes, lobos, etc.

Muitas espécies de animais se organizam em grupos (ANDERSON e FRANKS, 2001) nos quais geralmente há a presença de um líder que controla ou orienta o restante do grupo, como por exemplo o caso de um bando de leões ou uma matilha de lobos, etc. Neste tipo de sociedade organizada, o comportamento dos indivíduos é fortemente ditado pela hierarquia social. Porém, e mais extraordinário ainda, é o comportamento das espécies que vivem em grupos em que nenhum líder pode ser identificado, por exemplo, um bando de aves ou um cardume de peixes. Nestes grupos sociais os individuos não têm nenhum conhecimento do comportamento global de todo o grupo e nem têm qualquer informação sobre o meio ambiente. Apesar disso, eles têm a capacidade de se juntar e de se mover juntos, com base em interações locais entre indivíduos, ou seja, um comportamento social complexo surge a partir de interações simples entre indivíduos de um mesmo grupo. Tal comportamento complexo permite que o grupo encontre mais facilmente alimentos ou a direção certa a seguir.

Um grande número de estudos sobre o comportamento coletivo dos animais têm sido feito, por exemplo:

- Comportamento de aves em revoada (REYNOLDS, 1987);
- Comportamento de um cardume de peixes (PARTRIDGE, 1982);
- Comportamento das baleias jubarte (SHARPE, 2000);
- Comportamento de macacos selvagens (JANSON, 1998) e
- Comportamento do tubarão-frade (Cetorhinus maximus) (SIMS e QUAYLE, 1998).

Para auxiliar na compreenção das dinâmicas de grupo, presentes no comportamento coletivo de espécies animais socialmente organizadas, têm

sido feitos uma série de estudos e simulações. O estudo mais notável é o de Reynolds (1987). Ele mostrou através de simulações que a reunião de pássaros em bando é um comportamento que surge a partir da interação entre os indivíduos onde cada um obedece a regras simples e locais, tais como:

- **Evitar colisões:** onde o objetivo é evitar colisões com os indivíduos vizinhos baseado na posição física;

- **Harmonizar a velocidade**: onde o objetivo de cada indivíduo é igualar a velocidade com a de seus companheiros mais próximos;

- **Centralizar com o bando:** onde cada indivíduo tenta ficar próximo aos companheiros do bando.

Heppner e Grenander (1990) usam um modelo similar ao de Reynolds, mas adicionam um “rooster” para servir como atrator para todos os pássaros do bando, com o objetivo de descobrir as regras subjacentes que permitem que as aves se reunam de forma síncrona. Muitos modelos de simulação do comportamento de animais reunindo-se em bandos ou enxames têm sido usados em várias aplicações, como na área de entretenimento (jogos, filmes animados) e também em otimização de processos computacionais. Estes animais ou indivíduos seguem uma regra ou comportamento muito simples que é o de imitar o sucesso dos indivíduos vizinhos. O comportamento coletivo resultante é a descoberta de regiões ideais num extenso espaço de busca, praticamente impossíveis de serem realizadas por um único indivíduo. Tais descobertas teriam muito menos chances de lograrem êxito apenas por um indivíduo.

Os princípios básicos que norteiam a Inteligência de Enxames são:

- **Proximidade:** os indivíduos devem ser capazes de interagir;

- **Qualidade:** os indivíduos devem ser capazes de avaliar seus comportamentos;

- **Diversidade:** permite ao enxame reagir a situações inesperadas;

- **Estabilidade:** nem todas as variações ambientais devem afetar o comportamento de um indivíduo;

- **Adaptabilidade:** capacidade de adequação a variações ambientais.

2.2.3 PSO Básico

Nesta seção apresenta-se uma visão geral do algoritmo PSO básico e seus principais aspectos tais como: velocidade, posição, movimentação das partículas e estruturas de vizinhança.

2.2.3.1 Visão geral

*C*om base nos princípios de Inteligência de Enxames, apresentados na Seção 2.2.2, Kennedy e Eberhart (1995) desenvolveram um algoritmo denominado *Particle Swarm Optimization (PSO)* ou Otimização por Enxame de Partículas.

O algoritmo PSO segue um método estocástico de otimização e explora um espaço de busca com uma população de indivíduos que sondam regiões de busca promissoras. Em analogia com os paradigmas de Computação Evolucionária, um enxame é semelhante a uma população, enquanto uma partícula é semelhante a um indivíduo. Simplificando, as partículas “voam” através de um espaço de busca multidimensional, em que a posição de cada partícula em cada dimensão é ajustada a cada iteração do algoritmo, de acordo com a sua própria experiência *(personal best)* e a melhor posição de seus vizinhos *(global best)*. Cada partícula é uma solução viável para o problema e tem sua qualidade ou valor dado por uma função de aptidão (*fitness)*, também chamada de função objetivo, conforme Figura 5, abaixo:

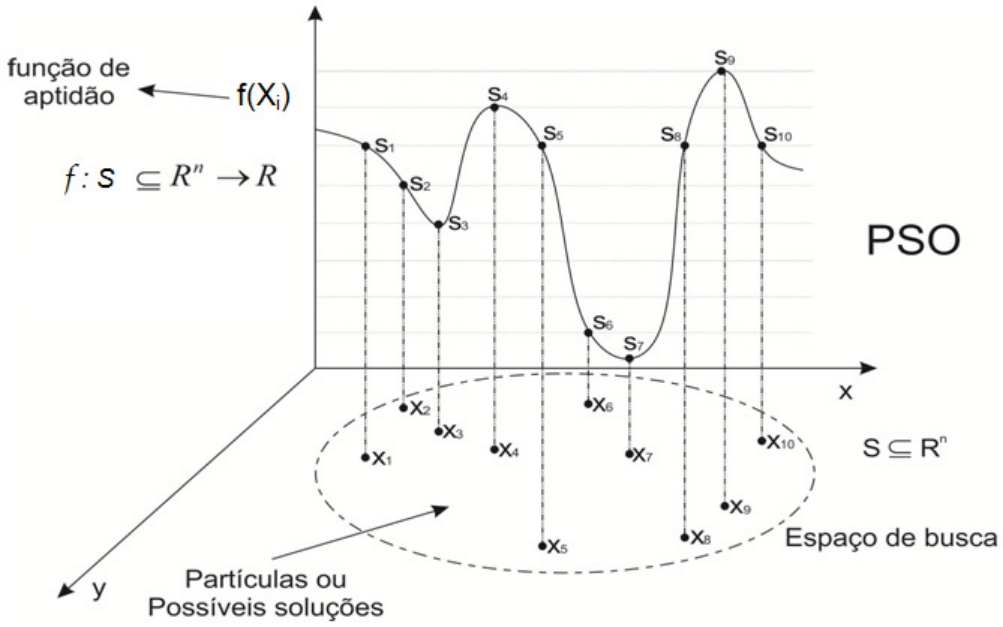

Figura 5 - Partículas e soluções

Como introduzido por Kennedy e Eberhart (1995), para um enxame de *N* partículas, onde cada partícula possui dimensão *D*, a velocidade (tamanho do passo) da *i-ésima* partícula na posição $X_{ij}(t)$ é calculada pela equação:

$$V_{ij}(t+1) = w \cdot \underbrace{V_{ij}(t)}_{\text{Velocidade prévia}} + c_1 \cdot r_{1j} \cdot \overbrace{\left(Pbest_{ij}(t) - X_{ij}(t)\right)}^{\text{Componente cognitivo}} + c_2 \cdot r_{2j} \cdot \overbrace{\left(Gbest_j(t) - X_{ij}(t)\right)}^{\text{Componente social}} \quad (2.2.1)$$

Onde $V_{ij}(t)$ é a velocidade da partícula *i* = 1 .. *N* na dimensão *j* = 1 .. *D*, no intervalo de tempo *t,* com *N* sendo a quantidade de partículas e *D* a dimensão da partícula. $X_{ij}(t)$ é a posição da partícula *i* na dimensão *j* no intervalo de tempo *t,* c_1 *e* c_2 são constantes de aceleração positivas usadas para balancear a contribuição dos componentes cognitivo e social

respectivamente, $\mathrm{r_{1j}(t)}$ e $\mathrm{r_{2j}(t)} \sim U(0,1)$ são valores randômicos na faixa [0,1] distribuídos uniformemente.

A posição $\mathrm{Pbest_i}$ é dada por:

$$\mathrm{Pbest_i(t+1)} = \begin{cases} \mathrm{Pbest_i(t)} & \text{se } \mathrm{f(x_i(t+1)) \geq f(Pbest_i(t))} \\ & \text{e } \mathrm{x_i(t+1)} \in [\mathrm{x_{min}, x_{max}}]^{\mathrm{N}} \\ \mathrm{x_i(t+1)} & \text{se } \mathrm{f(x_i(t+1)) < f(Pbest_i(t))}, \end{cases} \quad (2.2.2)$$

onde $f : R^N \rightarrow R$, é a função objetivo do problema em questão, para o caso de um problema de minimização.

A posição Gbest é calculada como sendo a melhor posição já encontrada por qualquer partícula do enxame.

O peso inercial *w* serve como uma espécie de "freio" que diminui gradativamente a velocidade da partícula a cada iteração. *w* pode ser um valor fixo ou um vetor de valores que decrescem a cada iteração. Na fórmula acima *w* aparece como um valor constante que deve ser inicializado no início do algoritmo. Vale ressaltar que na versão original do PSO, a equação da velocidade não utilizava o peso inercial *w,* que foi incluído mais tarde por Shi e Eberhart (1998).

Uma vez atualizada a velocidade da partícula, sua nova posição é dada por:

$$\underbrace{X_{ij}(t+1)}_{\text{Posição atual}} = \underbrace{X_{ij}(t)}_{\text{Posição anterior}} + \underbrace{V_{ij}(t+1)}_{\text{Velocidade atual}} \quad (2.2.3)$$

Com $X_{ij}(0) \sim U(X_{i_{min},j_{min}}, X_{i_{max},j_{max}})$.

2.2.3.2 Algoritmo PSO Básico

No algoritmo do PSO a equação da velocidade dirige todo o processo de otimização e reflete ambos os conhecimentos da partícula: o conhecimento advindo da sua própria experiência e o adquirido com a troca de informações com sua vizinhança. O conhecimento individual adquirido da sua própria experiência é referenciado como o componente cognitivo da partícula, e

proporciona a informação sobre a menor distância da solução ideal que aquela partícula já esteve. Esta posição é denominada de posição *personal best (Pbest)* da partícula, e influencia somente no seu próprio movimento. A informação obtida da troca de informações com a vizinhança é referenciada como o componente social na equação de velocidade. O componente social é também conhecido como *global best (Gbest)* e influencia no movimento de todas as partículas do enxame. Originalmente o algoritmo PSO tem duas versões que diferem somente em relação à estrutura de vizinhança utilizada. Os dois algoritmos foram batizados de PSO *Gbest*, onde a vizinhança de cada partícula é todo o enxame e o PSO *Lbest,* onde uma topologia de vizinhança e seu tamanho são determinados previamente, conforme será visto na Seção 2.2.6.

A seguir é apresentada a descrição do algoritmo PSO modelo *GBest*, onde a vizinhança da partícula é todo o enxame. Este é o modelo utilizado na versão hibridizada do PSO proposta neste trabalho, como será detalhado no Capítulo 3. No algoritmo abaixo também já está introduzido o componente inercial *w*.

Algoritmo PSO_Global_Best

Início

Inicializar aleatoriamente posição das partículas $\mathrm{X_{ij}}$;

Inicializar velocidade das partículas $\mathrm{V_{ij}}$ com valor nulo;

Inicializar constantes $\mathrm{c_1}$ e $\mathrm{c_2}$

Inicializar número máximo de iterações *iter_max_pso;*

Inicializar valor da variável *w*;

Determinar função objetivo do problema;

Determinar $\mathrm{Pbest_i}$ e Gbest iniciais

iter = 1;

Para iter = *1* **até** *iter_max_pso* **faça**

Para *i = 1 até N* **faça**

Para *j = 1 até D* **faça**

Atualizar $V_{ij}(t+1)$ de acordo com Equação 2.2.1;

Atualizar $X_{ij}(t+1)$ de acordo com Equação 2.2.3;

Fim_para;

Fim_para

Calcular nova aptidão da partícula de acordo com função objetivo do problema;

Atualizar partículas *Pbest conforme Equação 2.2.2;*

Atribuir partícula com melhor *fitness* à *Gbest* caso esta seja melhor que *GBest* atual;

Fim_para;

Fim_Algoritmo.

2.2.4 Movimentação das partículas

Uma das principais características do PSO é a capacidade das partículas, que representam as possíveis soluções para um problema, de se moverem num espaço de pesquisa à procura da melhor solução. Nesta seção são apresentados os principais aspectos que regem o movimento das partículas em busca da melhor solução, bem como as principais formas de interação entre elas e como é medida a aptidão ou qualidade das mesmas.

2.2.4.1 Velocidade das partículas

A velocidade, como já foi dito, dirige todo o processo de otimização em busca da melhor solução. Portanto nesta seção são descritos com mais detalhes os três componentes da equação da velocidade, a saber: velocidade prévia ou anterior, componente cognitivo e componente social.

- **Velocidade anterior: $V_{ij}(t)$,** serve como uma espécie de memória que indica a direção anterior da velocidade da partícula. Este componente vetorial impede que a partícula mude drasticamente de direção e tenha a tendência de manter a direção corrente. Este componente é também referenciado como componente inercial da velocidade.

- **Componente cognitivo:** O componente cognitivo ou *personal best (pbest)* referenciado como $\text{Pbest}_i(t)$, quantifica o desempenho da partícula *i* em relação

ao seu desempenho passado, ou seja, contêm o valor de aptidão (função fitness) da melhor posição já ocupada pela própria partícula no espaço de busca. Desse modo, o componente cognitivo assemelha-se a uma memória individual da melhor posição que a partícula já ocupou no espaço de busca. O efeito deste componente faz com que as partículas sejam atraídas de volta para sua melhor posição anterior, assemelhando-se ao fato de que indivíduos tendem a retornar aos locais mais preferidos por eles no passado.

- **Componente social:** O componente social ou *global best (gbest)* referenciado como *Gbest*, quantifica o desempenho da partícula *i* em relação a um grupo de partículas vizinhas, ou seja, contêm o valor de aptidão (função de *fitness*) da melhor posição que qualquer partícula já ocupou no espaço de busca. Conceitualmente, o componente social se assemelha a um objetivo comum que as partículas buscam atingir a cada iteração. O efeito do componente social é que cada partícula é também atraída para a melhor posição existente na sua vizinhança. No *PSO Gbest*, utilizado neste trabalho, a vizinhança é todo o enxame de partículas.

2.2.4.2 Movimentação da partícula: ilustração geométrica

Abaixo, na Figura 6, segue a ilustração sobre a movimentação de uma partícula X_i em um instante *t* num espaço bidimensional.

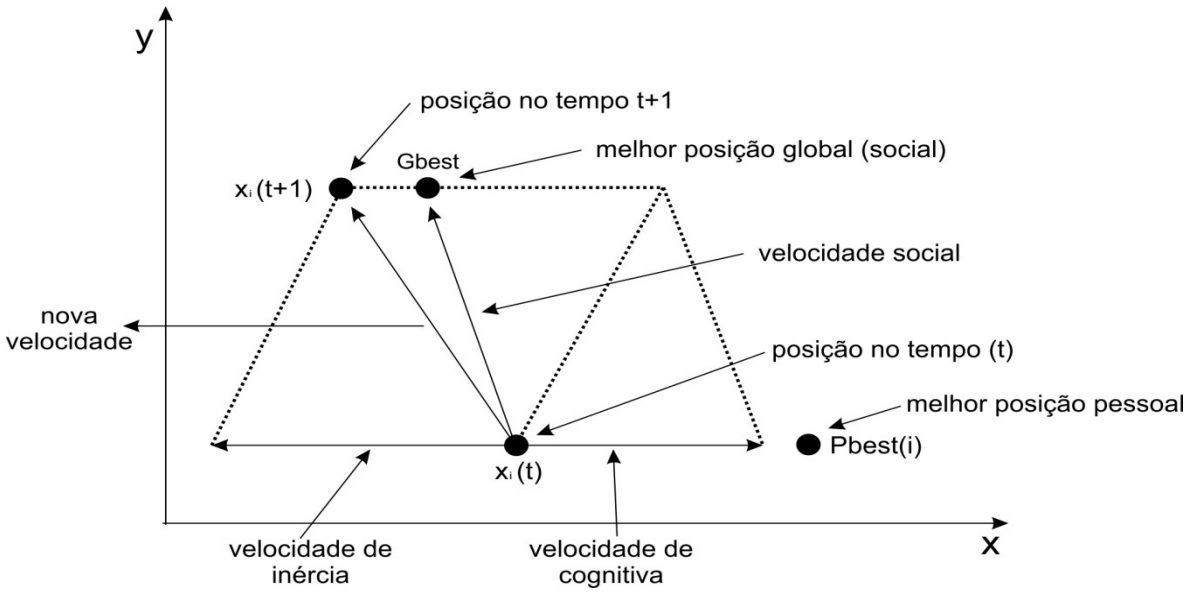

Figura 6 - Movimentação de uma partícula

Observa-se que a posição da partícula é mostrada em dois momentos, no tempo *t* e no tempo *t + 1*. A figura ilustra o comportamento da partícula sob o efeito das velocidades cognitiva, social e inercial, bem como sua nova posição.

A tendência da partícula como é visto, é mover-se em direção à posição da partícula mais bem posicionada no espaço de busca e também em direção à melhor posição já ocupada por ela própria. A equação da velocidade leva em consideração também a velocidade atual da partícula, componente inercial, para assim calcular o curso ou trajetória da partícula no próximo momento do tempo, $t + 1$.

Pode acontecer nesse cenário duas situações distintas, na primeira situação a partícula pode superar a melhor posição global, neste caso a posição da partícula torna-se a melhor posição global. Na segunda situação, a posição da partícula pode ser ainda pior que a posição *global best, (componente social)* neste caso em momentos posteriores a partícula *global best* e a *personal best* atrairão a partícula de volta.

2.2.5 Função de avaliação

O PSO realiza um processo de otimização iterativo. A cada iteração a velocidade e posição de cada partícula em cada dimensão é atualizada e depois disso é calculado o valor da aptidão de cada partícula, de acordo com a função de avaliação ou função objetivo da otimização. Com base na aptidão ou *fitness* da partícula são ajustadas as posições das partículas *Gbest e Pbest.* Portanto a cada iteração um total de *N* funções de avaliação é calculado onde *N* é o número de partículas do enxame. A função de aptidão é dependente do problema de otimização, que pode ser para maximização ou minimização do valor da função objetivo.

2.2.6 Estruturas de vizinhança

O desempenho do PSO depende fortemente da estrutura da rede social utilizada e, portanto, pode ser melhorada pela escolha da estrutura de vizinhança mais adequada ao problema. Várias topologias de estruturas de

rede sociais têm sido desenvolvidas para melhorar a interconectividade entre os componentes do enxame. As mais conhecidas são citadas por Engelbrecht (2005) e são descritas a seguir:

Estrutura social em estrela: nesta estrutura todas as partículas são interconectadas, portanto cada partícula pode se comunicar com todas as outras partículas, conforme Figura 7. Esta é a estrutura existente no PSO *Gbest* e possui a convergência mais rápida entre as estruturas de rede, porém permite que o PSO fique preso mais facilmente em ótimos locais. O uso desta estrutura é mais recomendado para problemas unimodais.

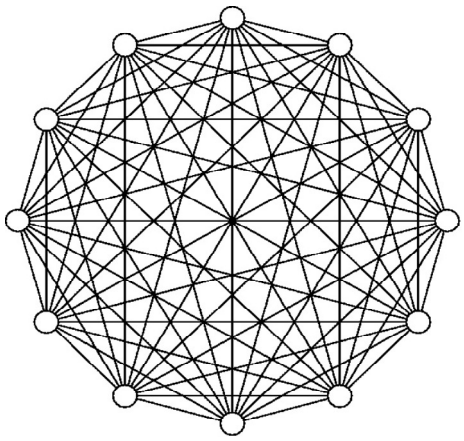

Figura 7 - Estrutura social em estrela

Estrutura social em anel: na estrutura social em anel básica, mostrada na Figura 8, cada partícula se comunica com somente duas partículas: a imediatamente posterior e a imediatamente anterior. Nesse caso, as vizinhanças se sobrepõem, e a taxa de convergência é mais lenta que na estrutura em estrela. Entretanto, isto permite que um maior espaço de busca seja pesquisado, proporcionando um desempenho melhor do algoritmo, especialmente quando se trata de problemas multi-objetivos. Algoritmos de PSO que utilizam esta estrutura são frequentemente classificados na categoria PSO *LBest.*

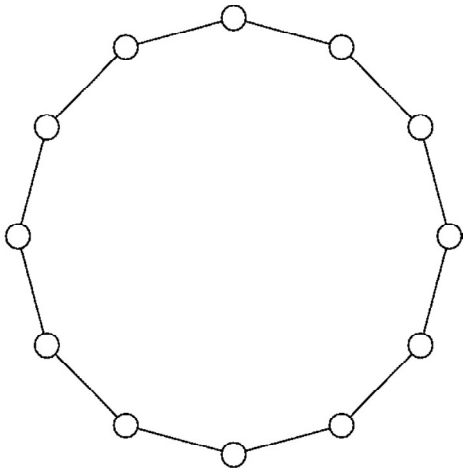

Figura 8 - Estrutura social em anel

Ainda existem outras estruturas de redes sociais menos utilizadas, como as estruturas em Roda ou Volante, Pirâmide, Quatro *Clusters* e Von Newmann.

Existem duas abordagens principais do PSO que diferem entre si pelo uso de um determinado tipo de estrutura de vizinhança: o PSO *Gbest* (*Global Best*) que utiliza a estrura de vizinhança do tipo estrela e o PSO *Lbest* (*Local Best*), que utiliza a estrutura em anel. As diferenças principais entre estas duas abordagens do PSO, com relaçao às características de convergência a um ótimo global, são as seguintes:

- Devido a grande interconectividade do PSO *Gbest*, estrutura em estrela, ele converge mais rápido que o PSO *LBest*, contudo esta convergência rápida implica numa menor diversidade que o PSO *LBest*.

- Como consequência da alta diversidade, que resulta num grande espaço de pesquisa sendo coberto pela estrutura em anel, o PSO *LBest* é menos susceptível a ficar preso em ótimos locais, contudo converge mais lentamente.

2.2.7 Variações Básicas

Embora o PSO básico (*LBest* ou *Gbest)* se mostre eficiente para resolver diversos problemas de otimização (ANGELINE, 1998) ele ainda sofreu algumas modificações no sentido de melhorar ainda mais a velocidade de convergência e a qualidade das soluções encontradas. As principais modificações sofridas pelo PSO básico são relatadas nesta seção, e são: introdução de peso inercial (SHI e EBERHART, 1998), fixação de limites de velocidade, introdução de um coeficiente de constrição (CLERC e KENNEDY, 2002), atualização síncrona versus assíncrona da posição da partícula e diferentes topologias de redes sociais (KENNEDY, 2000).

2.2.7.1 Peso inercial

O peso inercial w foi introduzido por Shi e Eberhart (1998) como um mecanismo para controlar a intensificação e diversificação da busca e também como um mecanismo para eliminar a necessidade de fixação da velocidade descrito na Seção 2.2.7.2. O primeiro objetivo foi bem tratado pela inclusão do peso de inércia na equação de velocidade, mas não conseguiu eliminar completamente a necessidade de fixação da velocidade.

O peso de inércia é responsável pela desaceleração gradativa da partícula até a convergência ou parada final. Isto serve para que a partícula não passe "voando" por cima de uma possível solução ótima. Este componente só foi introduzido na equação de velocidade mais tarde, visto que no PSO original este componente ainda não aparece. O valor de w é muito importante para garantir um comportamento de convergência e para fazer o balanceamento ideal entre diversificação e intensificação da busca. Para $w > 1$, a velocidade cresce com o tempo, acelerando a partícula até a velocidade máxima (assumindo que existe a fixação de velocidade) e afasta (diverge) o enxame. Deste modo as partículas falham ao mudar de *direção* e não conseguem mais retornar a uma área de busca promissora. Para $w < 1$, as partículas desaceleram até chegar à velocidade zero (dependendo também do coeficiente de aceleração). Valores de w grandes favorecem a diversificação e pequenos favorecem a intensificação da busca.

No início da introdução do peso de inércia se usou um valor estático para w. Depois passou-se a usar um w dinâmico diminuindo ao longo das iterações, o que favorece uma diversificação da busca no início e uma intensificação no final. É importante mencionar ainda a estreita relação entre w e os coeficientes de aceleração c_1 e c_2 e que, portanto, estes valores devem ser definidos em conjunto.

2.2.7.2 Limites de velocidade

Como já vimos, a equação da velocidade orienta todo o processo de pesquisa no espaço de busca. Ela determina não somente o tamanho do "passo" da partícula como também sua direção. Passos grandes favorecem a diversificação enquanto passos pequenos facilitam a intensificação da busca. Desta forma é preciso que o algoritmo de PSO fixe os limites inferior e superior de velocidade de modo que as partículas não deem passos muito grandes e saiam do espaço de busca ou deem passos muito pequenos e se movimentem muito pouco. É preciso então determinar com cuidado as variáveis *Vmax* e *Vmin* de forma que a exploração seja bem balanceada. Geralmente se escolhe uma fração do valor de cada dimensão do espaço de busca para *Vmax* e *Vmin*, ou seja, uma fração entre o máximo e o mínimo valor da posição da partícula em cada dimensão.

2.2.7.3 Coeficiente de constrição

Uma abordagem parecida com o peso de inércia foi desenvolvida por Clerc e Kennedy (2002) para balancear a diversificação e intensificação da busca, na qual a velocidade é limitada por uma constante, referida como coeficiente de constrição. O uso deste coeficiente de constrição pode evitar o uso de limites máximo e mínimo para a velocidade e melhora a taxa de convergência.

2.2.7.4 Atualização síncrona x assíncrona

PSO é um processo iterativo, onde as posições das partículas e as melhores posições são atualizadas a cada iteração. A ordem em que as posições das partículas e as melhores posições são atualizadas é denominada estratégia de iteração. Existem duas principais estratégias de iteração para PSO, ou seja, atualizações síncronas e atualizações assíncronas. Um certo número de estudos têm discutido as vantagens e desvantagens destas estratégias de iteração, e a maioria destes estudos indicam que atualizações assíncronas são melhores do que as atualizações síncronas no que dizem respeito à precisão das soluções obtidas e à velocidade em que o enxame converge. Em contrapartida, Engelbrecht (2013b) fornece evidências através de uma análise empírica extensa que as opiniões atuais de que interações assíncronas resultam em convergência mais rápida e precisa não são verdadeiras.

Segundo Carlisle e Dozier (2001) a atualização síncrona é mais indicada para o PSO *Gbest* enquanto que a atualização assíncrona é mais adequada ao PSO *LBest*.

Neste modelo proposto optou-se por usar uma estratégia de atualização síncrona das posições da partícula tendo em vista que se utilizou o modelo de PSO *Gbest*.

2.2.8 Parâmetros de controle

O PSO básico é influenciado por um certo número de parâmetros de controle, a saber, o número de partículas, coeficientes de aceleração, o peso inercial, o tamanho da vizinhança, o número de iterações, e os valores dos coeficientes de aceleração. Adicionalmente, caso se esteja usando limitação de velocidade ou coeficiente de constrição isto também influencia na performance do PSO.

Escolhas erradas de valores dos parâmetros podem conduzir à divergência ou comportamento cíclico do algoritmo. Valores padrão para alguns parâmetros devem ser usados com cuidado, visto que testes empíricos podem

funcionar somente para algumas amostras. Isto mostra que a especificação dos parâmetros de controle são muito dependentes do problema.

2.2.8.1 Tamanho do enxame

O tamanho do enxame refere-se ao número de partículas que compõem o enxame. Um enxame grande permite que mais partículas ocupem o espaço de busca por iteração, permitindo a descoberta de soluções ótimas com menos iterações. Por outro lado grandes enxames podem aumentar muito a complexidade computacional e degradar a performance do algoritmo. Um espaço de busca liso (com poucos ótimos locais) pode precisar de menos partículas que um espaço de busca rugoso (com muitos ótimos locais) para a descoberta de soluções ótimas, portanto o tamanho do enxame é dependente do problema, contudo pesquisas mostram (BRITS *et al* ,2001; VAN DEN BERGH e ENGELBRECHT, 2001) que tamanhos de enxame entre 10 e 30 partículas são bons para descobrir soluções ótimas.

2.2.8.2 Tamanho da vizinhança

O tamanho da vizinhança define a extensão da interação social com o enxame. Quanto menor for a vizinhança menos interações ocorrem. Enquanto uma vizinhança pequena causa uma convergência lenta, elas têm uma convergência mais confiável para soluções ótimas. Tamanhos de vizinhanças pequenos são menos susceptíveis a mínimos locais.

Para quantificar as vantagens com tamanhos de vizinhança grandes e pequenos, deve-se iniciar a busca com tamanho pequeno e incrementar gradativamente conforme o número de iterações. Esta abordagem garante uma rápida convergência e diversidade alta do enxame. No PSO *Gbest* o tamanho da vizinhança é o tamanho do enxame.

2.2.8.3 Coeficientes de aceleração

Os coeficientes de aceleração, constantes c_1 e c_2 combinados com as variáveis randômicas r_1 e r_2, controlam a influência estocástica dos

componentes social e cognitivo na velocidade da partícula. As constantes c_1 e c_2 são também conhecidas como os parâmetros de confiança, onde c_1 expressa quanta confiança a partícula tem em si, enquanto c_2 expressa quanta confiança a partícula tem na vizinhança. Com $c_1 = c_2 = 0$, as partículas continuam voando com sua velocidade corrente até que atinjam o limite do espaço de busca (assumindo que não tem inércia). Se $c_1 > 0$ *e* $c_2 = 0$ todas as partículas são independentes. Neste caso cada partícula procura a melhor posição na vizinhança trocando a posição corrente se a nova posição é melhor, realizando uma busca local somente. Por outro lado se $c_1 = 0$ *e* $c_2 > 0$, o enxame inteiro é atraído para um único ponto *global best*.

As partículas são mais "fortes" quando cooperam entre si, portanto quando os parâmetros c_1 e c_2 são bem balanceados a busca é mais promissora. Quando $c_1 = c_2$, as partículas são atraídas com a mesma força em direção ao *Pbest* e ao *Gbest*. Muitas aplicações usam esta igualdade, mas a relação entre estas constantes é dependente do problema. Se $c_1 > c_2$ cada partícula é mais atraída em direção a sua melhor posição pessoal, o que resulta num número excessivo de partículas errantes. Por outro lado, se $c_1 < c_2$ as partículas são mais atraídas em direção à melhor posição global, o que leva as partículas a se apressarem prematuramente para um ótimo local. Para espaços de busca mais suaves, um coeficiente social maior pode ser mais eficiente, enquanto para um espaço de busca mais acidentado, um grande componente cognitivo pode ser mais vantajoso.

Comumente os parâmetros c_1 e c_2 são estáticos com seus valores ótimos sendo determinados empiricamente. Inicializações erradas de c_1 e c_2 podem causar divergências ou comportamento cíclico da busca.

2.2.8.4 Inicialização das posições das partículas

Um dos primeiros passos do algoritmo deve ser a inicialização das posições das partículas do enxame em todas as suas dimensões. A dimensão da partícula refere-se à quantidade de características ou atributos que cada partícula carrega, necessárias para calcular a solução ótima para o problema a ser otimizado.

As partículas devem ser inicializadas de forma a cobrir uniformemente o espaço de busca, visto que a performance do PSO é influenciada pela distribuição inicial do enxame no espaço de busca, pois se alguma região do espaço não for preenchida por partículas o PSO terá mais dificuldades em encontrar uma possível posição promissora nesta região. O valor inicial da posição das partículas para cada dimensão, $X_{ij}(0) \in R^n$, com $i = 1 \dots N$ e $j = 1 \dots D$, onde *N* é igual ao número de partículas e *D* é igual ao número de dimensões da partícula, deve estar entre os valores máximos e mínimos encontrados em cada dimensão.

Esquemas de inicialização diferentes têm sido usados pelos pesquisadores para inicializar as posições das partículas. Estes esquemas de inicialização têm sido utilizados principalmente para assegurar que o espaço de busca é coberto uniformemente, segundo Engelbrecht (2005).

2.2.8.5 Inicialização da velocidade das partículas

A velocidade de cada partícula em cada dimensão pode ser inicializada com o valor nulo, ou seja, $V_{ij}(t) \equiv 0$, com $i = 1 \dots N$ e $j = 1 \dots D$, onde *N* é igual ao número de partículas e *D* é igual ao número de dimensões da partícula. É possível ainda inicializar as velocidades com valores aleatórios, mas isso deve ser feito com bastante cuidado visto que isto pode colocar a partícula para fora do espaço de busca e também viola a regra natural, pois as partículas inicialmente são objetos estáticos e portanto devem ter velocidade igual a zero.

2.2.8.6 Condições de parada do algoritmo PSO

Outro aspecto importante do PSO é a condição de parada, que diz respeito ao critério usado para terminar o processo de busca. Muitos critérios têm sido alvo de pesquisas. Para selecionar um determinado critério dois aspectos precisam ser levados em consideração, são eles:

- Uma condição de parada não deve causar uma parada prematura do PSO, visto que o algoritmo ainda não pode ter atingido a convergência desejada;

- A condição de parada não deve requerer cálculo da função de fitness, pois isso pode aumentar consideravelmente a complexidade computacional do processo de pesquisa.

As seguintes condições de parada tem sido utilizadas, a saber:

- Terminar quando um número máximo de iterações for atingido;
- Terminar quando uma solução aceitável for encontrada;
- Terminar quando não houver melhora observada durante um determinado número de iterações;
- Terminar quando o valor da função objetivo se aproximar do valor ótimo conhecido.

2.3 Metaheurística ILS (*Iterated Local Search*)

Nesta seção faz-se um breve levantamento sobre a meta-heurística ILS, que é utilizada juntamente com o PSO neste trabalho, para resolver problemas de Clusterização de Dados.

2.3.1 *Iterated Local Search* (ILS) - Visão geral

A metaheurística ILS *(Iterated Local Search*) é baseada na ideia de que um procedimento de busca local pode ser melhorado iterativamente, gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local (LOURENÇO, 2010). Na Figura 9 pode-se observar melhor o processo de pesquisa com ILS.

Para aplicar um algoritmo ILS, quatro componentes têm que ser especificados: um procedimento para gerar a solução inicial; um procedimento para fazer uma busca local, que retorna uma solução possivelmente melhorada; um procedimento de perturbação da solução, que modifica a solução corrente guiando a uma solução intermediária; e um procedimento que avalia de qual solução a próxima perturbação será aplicada.

2.3.2 Algoritmo ILS

Abaixo é mostrado o algoritmo ILS básico.

Algoritmo ILS
Início

1 Gerar solução inicial;
2 Realizar busca local na solução inicial;
3 **Enquanto** critério de parada não for satisfeito **Faça**
4 Realizar perturbação na solução;
5 Fazer busca local na solução perturbada;
6 Aplicar critério de aceitação;
7 **Fim_Enquanto**;

Fim_Algoritmo_ILS;

O bom desempenho do ILS deve-se principalmente à escolha correta do método de busca local, das perturbações e do critério de aceitação. Em princípio, qualquer método de busca local pode ser usado, mas o desempenho do ILS com respeito à qualidade da solução final e a velocidade de convergência depende fortemente do método escolhido. Normalmente um método de descida ou subida é usado, mas também é possível aplicar algoritmos mais sofisticados ou outras meta-heurísticas.

Quanto à intensidade da perturbação, ela deve ser forte o suficiente para permitir escapar de ótimos locais e explorar diferentes regiões, mas também deve ser fraca o suficiente para guardar as características do ótimo local corrente.

Com relação ao critério de aceitação, ele é usado para decidir de qual solução se continuará a exploração, bem como qual será a perturbação a ser aplicada. Um aspecto importante do critério de aceitação e da perturbação é que eles induzem aos procedimentos de intensificação e diversificação. A intensificação consiste em permanecer na região do espaço onde a busca se encontra, procurando explorá-la de forma mais efetiva, enquanto a diversificação consiste em se deslocar para outras regiões do espaço de soluções. A intensificação da busca no entorno da melhor solução encontrada é obtida, por exemplo, pela aplicação de "pequenas" perturbações sobre ela. A diversificação, por sua vez, pode ser realizada aceitando-se quaisquer soluções s'' e aplicando-se "grandes" perturbações na solução ótima local. Um critério de aceitação comumente utilizado é o critério determinístico, onde se move para o ótimo local s'' somente se ele for melhor que o ótimo local corrente s , isto é, somente se $f(s'') < f(s)$, em um problema de minimização, ou se $f(s'') > f(s)$ em um problema de maximização.

A Figura 9, abaixo, ilustra o funcionamento do método ILS em um problema de minimização. A partir de um ótimo local s, é feita uma perturbação que guia a uma solução intermediária s'. Após a aplicação de um método de busca local a s' é produzido um novo ótimo local s''. Considerando como critério de aceitação o fato de $f(s'')$ ser melhor que $f(s)$, então a busca prossegue a partir de s''.

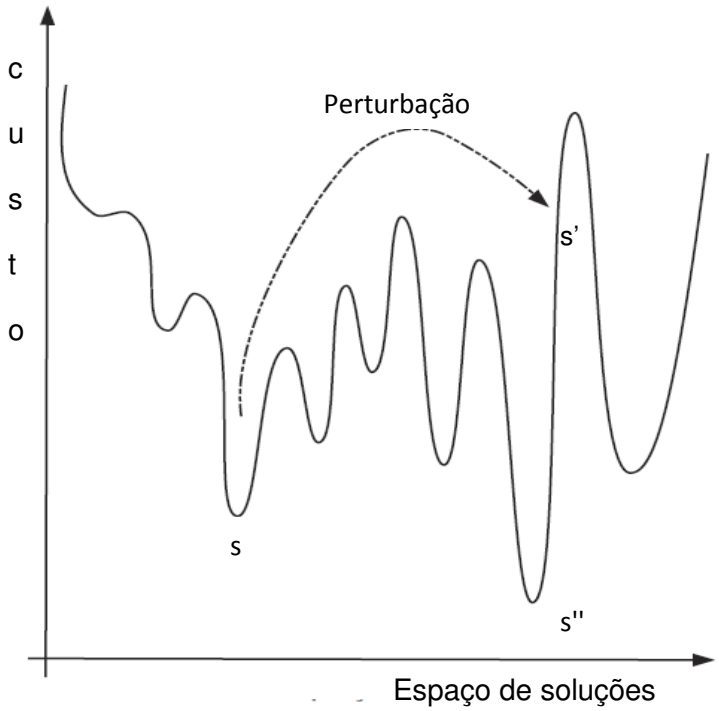

Figura 9 - Ilustração do ILS

Na abordagem híbrida do algoritmo proposto neste trabalho, utilizou-se o algoritmo ILS com a busca a local sendo o algoritmo *K-means* e usou-se como perturbação uma pequena movimentação nos centroides da solução ótima local para tentar obter uma melhor solução. O ILS é usado como estratégia de aceleração da convergência do algoritmo EPSO-ILS proposto, conforme será visto com mais detalhes no Capítulo 3.

2.4 Trabalhos relacionados

Nesta seção faz-se uma breve justificativa da utilização de hibridizações de meta-heurísticas e também um levantamento de alguns trabalhos relacionados na literatura que incluem a hibridização do PSO com outras heurísticas e meta-heurísticas com o objetivo de melhorar o processo de Clusterização de Dados.

2.4.1 Motivação para a hibridização de meta-heuríticas

Nos últimos anos tem se usado muito a hibridização de uma ou mais meta-heurísticas baseadas em Computação Evolucionária e Inteligência de Enxames com outras técnicas de otimização para resolver problemas

complexos (OLIVEIRA, 2007). O uso de uma só meta-heurística pode tornar muito restritiva a solução. A combinação de uma meta-heurística com outras técnicas de otimização, chamada meta-heurística híbrida, pode proporcionar um comportamento mais eficiente e uma maior flexibilidade, principalmente quando se trata de problemas de otimização combinatória do mundo real e em larga escala como o problema de Clusterização de Dados.

Uma meta-heurística pode ser combinada, por exemplo, com métodos exatos, com outras meta-heurísticas ou com heurísticas de busca local, com o objetivo de obter sistemas com desempenho melhor que explorem e unam as vantagens das estratégias puras aplicadas individualmente.

Embora as técnicas baseadas nos paradigmas de Inteligência de Enxames e Computação Evolucionária obtenham, na maioria das vezes, bons resultados para os problemas de Clusterização de Dados, estudos apontam que o uso prático destas técnicas ainda possuem algumas falhas na solução de problemas complexos de otimização, pois são sensivelmente limitadas pelo alto custo computacional e taxa de convergência lenta para uma solução global do problema.

O algoritmo PSO possui algumas deficiências tais como: dá bons resultados para otimização de problemas unimodais, mas fica preso em ótimos locais (JUNLIANG e XINPING, 2008) para problemas multi-objetivos, as partículas sofrem uma espécie de estagnação ao chegarem próximas ao limite do espaço de busca e pode ter uma rápida e prematura convergência próximo a ótimos locais.

2.4.2 Trabalhos relacionados

Rana *et al* (2011) e Kao *et al* (2008) demonstraram que o PSO padrão aplicado à Clusterização de Dados possui um desempenho superior aos demais algoritmos usados para o mesmo fim. Porém, a taxa de convergência na busca do ótimo global ainda é passível de melhora.

Van der Merwe e Engelhrecht (2003) introduziram um método com o uso de PSO para resolver os problemas de Clusterização de Dados. Os resultados do algoritmo foram comparados com o algoritmo *K-means*. A abordagem

proposta deu melhores resultados em comparação com o algoritmo *K-means* puro.

Chen e Ye (2004) afirmaram que a qualidade do algoritmo *K-means* é altamente dependente da sua natureza de seleção dos centroides iniciais, e não fornece a solução global ótima na maioria das vezes. Para superar esta desvantagem do *K-means,* um algoritmo baseado em PSO para a Clusterização de Dados foi proposto por eles.

Kao *et al*. (2008) afirmaram que PSO dá melhores resultados em problemas de clusterização quando é aplicado em objetos de dados de poucas dimensões ou características e para um pequeno número de objetos, mas quando é aplicado a um grande conjunto de objetos não dá bons resultados. A razão disto é que as partículas ficam estagnadas quando chegam no limites do espaço de busca. Para superar esta deficiência eles propuseram uma espécie de "turbulência" aplicada nas partículas estagnadas.

Em Rana *et al*. (2011), é proposta uma hibridização do algoritmo PSO com o algoritmo *K-means* denominada *Hybrid Sequential Clustering Algorihtm* para o problema de Clusterização de Dados. Nesta abordagem o processo de clustrização começa com o PSO que executa até o final das iterações. Após o término do PSO o algoritmo *K-means* inicia a execução recebendo do PSO a partícula *Gbest* como centróides iniciais. Esta abordagem traz resultados melhores para a clusterização do que a execução do *K-means* ou do PSO em separado, pois ela supera as desvantagens de ambos os algoritmos, ou seja, evita a escolha inicial de centroides ruins no algoritmo *K-means* puro e intensifica a busca da melhor partícula do PSO para finalizar a clusterização.

Uma das muitas versões do PSO conhecida como CPSO (*Chaotic Particle Swarm Optimization*), encontrada em Chuang *et al*. (2011), usa um mapa caótico ou sequência de números caóticos para substituir os componentes randômicos $\mathrm{r}_1 \mathrm{e}\, \mathrm{r}_2$ da Equação 2.2.1. No campo da engenharia, é bem reconhecido que a Teoria do Caos pode ser aplicada como uma técnica muito útil em aplicações práticas (COELHO e MARIANE, 2009). O Sistema Caótico pode ser descrito como um sistema não linear limitado com comportamento dinâmico determinista que tem propriedades ergódicas e estocásticas (SCHUSTER e JUST, 2005). É muito sensível às condições iniciais e parâmetros utilizados. Em outras palavras, a causa e efeito de caos

não são proporcionais às pequenas diferenças entre os valores iniciais. Pequenas variações de uma variável inicial podem resultar em enormes diferenças nas soluções após algumas iteração. Matematicamente, uma variável caótica é aleatória e imprevisível, mas também possui um elemento de regularidade (ALATAS *et al*, 2009). Devido a estas características, a teoria do caos pode ser aplicada na Otimização, em especial nesta variação do PSO para acelerar a convergência em busca da solução global.

Os números caóticos são semelhantes aos números randômicos e são gerados por fórmulas de iteração determinísticas, como o Mapa Logístico, um dos mapas caóticos mais simples, introduzido por May (1976), que gera números caóticos usando a seguinte fórmula:

$$\mathrm{Cr(t+1)} = \mathrm{k} * \mathrm{Cr(t)} * (1 - \mathrm{Cr(t)}) \qquad (2.4.1)$$

Na Equação 2.4.1, o $Cr(t)$ inicial é um número randômico entre $0\ e\ 1$ e diferente dos valores do conjunto $\{0.00\ 0.25\ 0.50\ 0.75\ 1.00\}$. O valor de k controla o comportamento do Mapa Logístico. Para $k\ = 4$ o valor de Cr tem um comportamento caótico.

Os gráficos 1 e 2 abaixo mostram o comportamento da sequência de 100 números caóticos gerados pelo Mapa Logístico e uma sequência de 100 números gerados aleatoriamente, respectivamente.

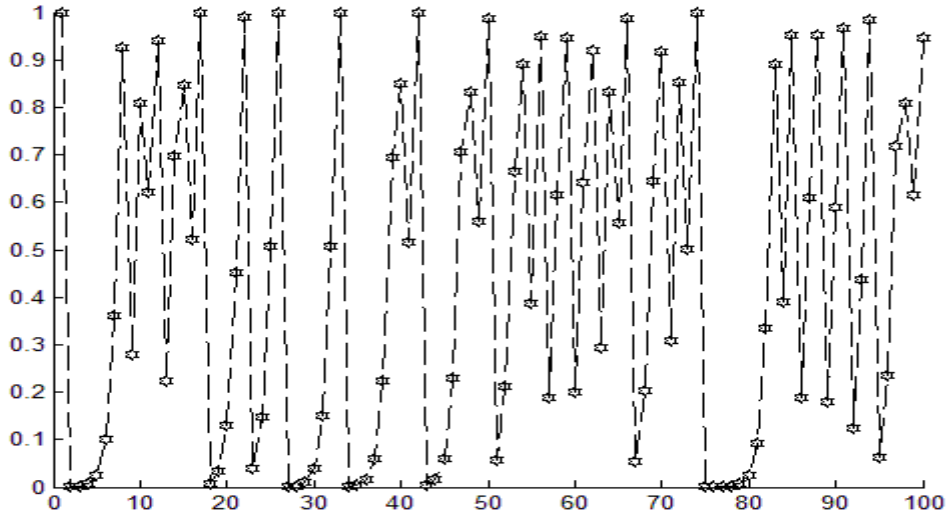

Gráfico 01 – Sequência de números caóticos

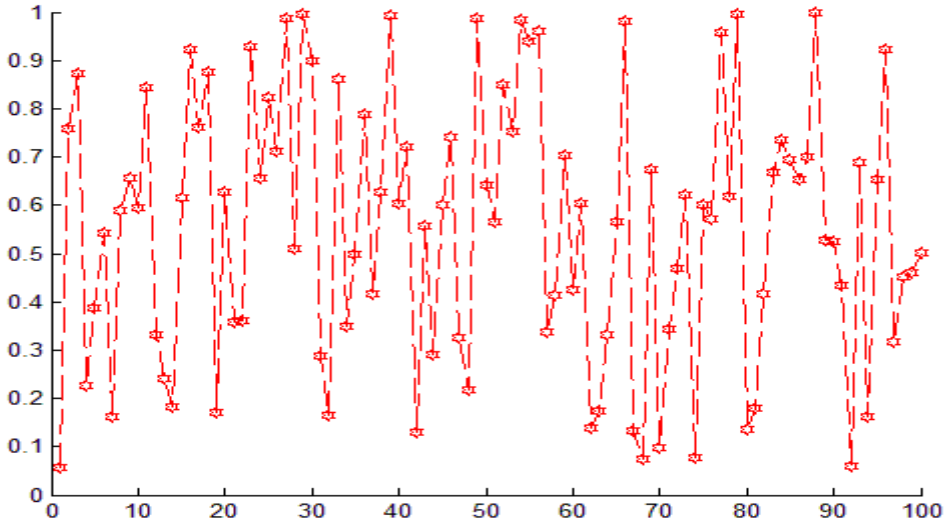

Gráfico 02 – Sequência de números randômicos

Pode-se observar que no Gráfico 1 mais números são gerados nos extremos dos valores entre 0 e 1. Foram computados, na geração dos 100 números caóticos apresentados no Gráfico 1 uma quantidade de 48 números caóticos maiores que 0,9 e menores que 0,1, enquanto que no Gráfico 2 apenas 21 números randômicos foram gerados nestes dois intervalos.

Em Mendel (2011) é feita uma análise estatística que justifica o uso do mapa logístico no PSO. Eles mostram que o mapa logístico, com $k = 4$, gera mais números aleatórios próximos aos dois extremos do intervalo $[0,1]$. Esta propriedade do mapa logístico pode ser vista no histograma da Figura 10, que mostra o grau de dispersão dos valores gerados por um mapa logístico com $k = 4$, num intervalo $z[0,1]$. Desta forma o uso desta característica permite que partículas possam dar "saltos" maiores para escapar de ótimos locais ou sair de uma situação de estagnação mais facilmente, como também dar "saltos" ou "passos" pequenos possibilitando um maior refinamento da pesquisa.

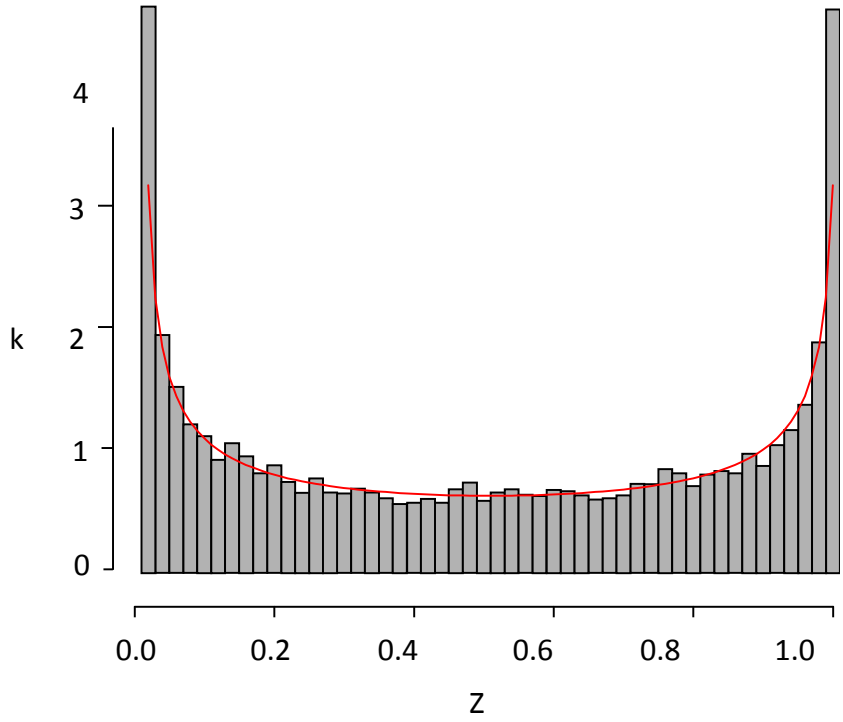

Figura 10 - Histograma de 10.000 iterações do mapa logístico

A equação da velocidade para o CPSO com a inclusão do número caótico Cr gerado pelo Mapa Logístico com valor de $k = 4$, em substituição à Equação 2.2.1 fica então da seguinte forma:

$$\mathrm{V_{ij}(t+1)} = \mathrm{w} * \mathrm{V_{ij}(t)} + c_1 * \mathrm{Cr(t)} * \left(\mathrm{Pbest_{ij}} - \mathrm{X_{ij}(t)}\right) + \mathrm{c_2} * \left(1 - \mathrm{Cr(t)}\right) * \left(\mathrm{Gbest_j} - \mathrm{X_{ij}(t)}\right) \quad (2.4.2)$$

Em Chuang *et al*. (2011) eles usam uma estratégia de aceleração para aumentar ainda mais a velocidade de convergência do CPSO. Esta estratégia consiste em aplicar o *K-means* em um terço das partículas para melhorar a posição dos centroides antes de iniciar o CPSO. Esta estratégia supera os algoritmos com os quais foi comparada, inclusive o próprio CPSO descrito na seção acima.

Em Coelho Filho et al. (2013b) é usada uma abordagem similar ao ACPSO descrito no parágrafo anterior mas com uma estratégia de aceleração que difere um pouco. Nesta abordagem, denominada ECPSO, o autor propõe o uso do K-means em um décimo das iterações do CPSO, ou seja, nas primeiras iterações o K-means tenta melhorar uma das partículas, escolhidas

aleatoriamente a cada iteração, e caso o fitness da partícula seja melhorado pelo K-means ela é substituída no enxame. Este método obteve melhores resultados do que o ACPSO em algumas bases de dados, e igual em outras, conforme pode ser observado na Tabela 2, seção de Resultados computacionais.

Em Coelho Filho *et al*. (2013a) foi proposto um algoritmo híbrido para Clusterização de Dados baseado na meta-heurística PSO e na meta-heurística ILS, denominado EPSO-ILS. Nesta abordagem, somente a melhor solução ou partícula da primeira iteração do PSO é passada como parâmetro para a meta-heurística ILS. O ILS então melhora gradativamente esta partícula ao longo das iterações através da aplicação de perturbações representadas por pequenas movimentações nos centroides representados pela partícula, fazendo uma busca local na solução perturbada. Este método provou ser eficiente e rápido conforme resultados exibidos na Tabela 2 do Capítulo 4.

Capítulo 3

METODOLOGIA

Neste capítulo descreve-se a metodologia utilizada para o desenvolvimento do algoritmo híbrido proposto para melhorar a resolução de problemas de Clusterização de Dados. Este algoritmo utiliza as meta-heurísticas PSO e ILS hibridizadas, cujo embasamento teórico de cada uma delas foi visto no Capítulo 2. Apresenta-se aqui todos os aspectos envolvidos na criação do algoritmo proposto, que foi denominado de EPSO-ILS (*Enhanced Particle Swarm Optimization - Iterated Local Search*), bem como a configuração dos parâmetros usados no algoritmo e a estratégia de aceleração utilizada.

3.1 Considerações gerais

Como foi apresentado na Seção 2.4, muitas variações, adaptações e hibridizações do algoritmo PSO têm sido propostos para superar as deficiências ainda encontradas no PSO, com relação à baixa taxa de convergência, estagnação das partículas e prisão em ótimos locais.

Neste sentido buscou-se desenvolver um algoritmo capaz de melhorar ainda mais o poder do PSO, através de uma hibridização com outra meta-heurística, o ILS e uma heurística de busca local, o *K-means*. O objetivo desta hibridização é melhorar a eficiência do processo de Clusterização de Dados, quando se conhece, a *priori*, o número de *clusters (problema de k-clusterização).*

A combinação das duas meta-heurísticas mencionadas se justifica porque o uso de uma meta-heurística populacional, como é o caso do PSO (que otimiza várias soluções ou partículas e com isso tem uma convergência mais lenta), hibridizada com uma meta-heurística que otimiza uma única solução (por isso tem um esforço computacional menor), como é o caso do ILS, permite superar as desvantagens de ambas as meta-heurísticas e obter um algoritmo com performance melhor, que explore e una as vantagens das duas estratégias puras aplicadas individualmente.

Vários testes foram feitos utilizando-se outras meta-heurísticas para hibridizar com o PSO, porém, ao se implementar e testar o ILS para Clusterização de Dados, verificou-se que este possui uma excelente performance e uma taxa de convergência muito rápida para soluções próximas à ótima.

Além da chamada do ILS, foi aplicada outra estratégia para melhorar ainda mais a eficiência do algoritmo proposto. Baseado na versão exitosa do algoritmo CPSO, proposto por Chuang *et al*. (2011), descrito na Seção 2.4.2, ao invés de se multiplicar os componentes *Pbest* e *Gbest* por um número caótico, utilizou-se um componente caótico para gerar o peso inercial. Isto foi feito com o objetivo de possibilitar que a partícula possa fugir mais facilmente de ótimos locais mesmo ao fim das iterações, pois no PSO original o valor do peso inercial diminui gradativamente ao longo das iterações e, próximo ao final, a partícula dá passos muito pequenos, que não permitem que ela saia de ótimos locais.

3.2 Características do EPSO-ILS

Nesta seção são vistos os principais aspectos do algoritmo proposto neste trabalho, tais como a forma de representar uma partícula do PSO, a função de avaliação que se utilizou para medir a qualidade da partícula para o problema em questão, o modelo de PSO utilizado, a estrutura de vizinhança ideal, a medida de similaridade utilizada, a nova equação da velocidade e por fim os parâmetros utilizados e a estratégia de aceleração com o uso do ILS.

3.2.1 Representação da partícula

Como visto Seção 2.1.6, problemas de clusterização podem ser definidos da seguinte forma: dado um conjunto E com *m* elementos ou padrões, $E = \{x_1, x_2, \dots, x_m\}$, o problema de clusterização consiste na obtenção de um conjunto de *k clusters*, $C = \{C_1, C_2, \dots, C_k\}$, tal que os elementos contidos em um *cluster* C_i possuam uma maior similaridade entre si do que com os elementos de qualquer um dos demais *clusters* do conjunto C. O conjunto C é considerado

uma clusterização com *k clusters* caso sejam satisfeitas as condições da Equação 2.1.1.

Cada elemento ou objeto do conjunto E representa um padrão e é formado por um vetor que contem os valores que representam os atributos ou características do objeto, ou seja, $x_i = \{x_{i1}, \dots, x_{ij}, \dots, x_{id}\}$, com *i* = 1..m e *j = 1..d,* onde *m* indica a quantidade de objetos a clusterizar e *d* indica a quantidade de dimensões de cada objeto do conjunto E. Um conjunto de padrões ou objetos é representado na maioria das vezes por uma matriz m *x d.*

Como cada partícula do PSO representa uma solução completa para um problema em questão, no caso do problema de clusterização cada partícula representa os k-centroides necessários para resolver o problema. Portanto, neste algoritmo proposto, cada partícula $\mathrm{X_i}$ é construída como: $\mathrm{X_i} = (m^1, \dots, m^j, \dots, m^k)$, onde m^j refere-se ao j-ésimo centroide que representa cada cluster do conjunto C da i-ésima partícula. Cada centroide m^j possui o mesmo número de dimensões dos objetos que ele representa, ou seja, $m^j = (m_1^j, m_2^j, \dots, m_d^j)$, portanto, a dimensão de cada partícula, *D,* é dada pela multiplicação do número de dimensões, *d*, de cada objeto, pelo número de centroides ou *clusters*, *k*, ou seja:

$$D = k * d \tag{3.1}$$

Cada centroide m^j , onde *j = 1 .. k*, representa um cluster C_j composto por σ_j objetos. O valor de cada centroide m^j da partícula X_i, é determinado calculando-se a média dos valores dos objetos mais próximos de cada centroide conforme Equação 2.1.6 readequada para este método.

$$m_i^j = \frac{1}{\sigma_j} \sum_{\forall\, x^p \in C_j} x_i^p \quad , onde\ i = 1..d \tag{3.2}$$

Onde σ_j é o número de objetos pertencentes ao *cluster* C_j, e x^p é um objeto pertencente ao *cluster* C_j, com *p = 1 .. σ_j*.

Para determinar se um objeto x^p pertence a um centroide m^j, utiliza-se a medida de proximidade dada pela distância euclidiana da Equação 2.1.2 readequada para o problema de clusterização e apresentada abaixo:

$$D(x^p, m^j) = \sqrt{\sum_{i=1}^{d}(x_i^p - m_i^j)^2} \tag{3.3}$$

Assim, um objeto x^p pertencerá a um centroide m^j que possuir o mínimo valor na Equação 3.3, para *j = 1 .. k*.

Para um melhor entendimento, abaixo, na Figura 11, é ilustrada a formação das partículas que representam as soluções para a clusterização de uma base de dados com objetos de duas dimensões em três *clusters*.

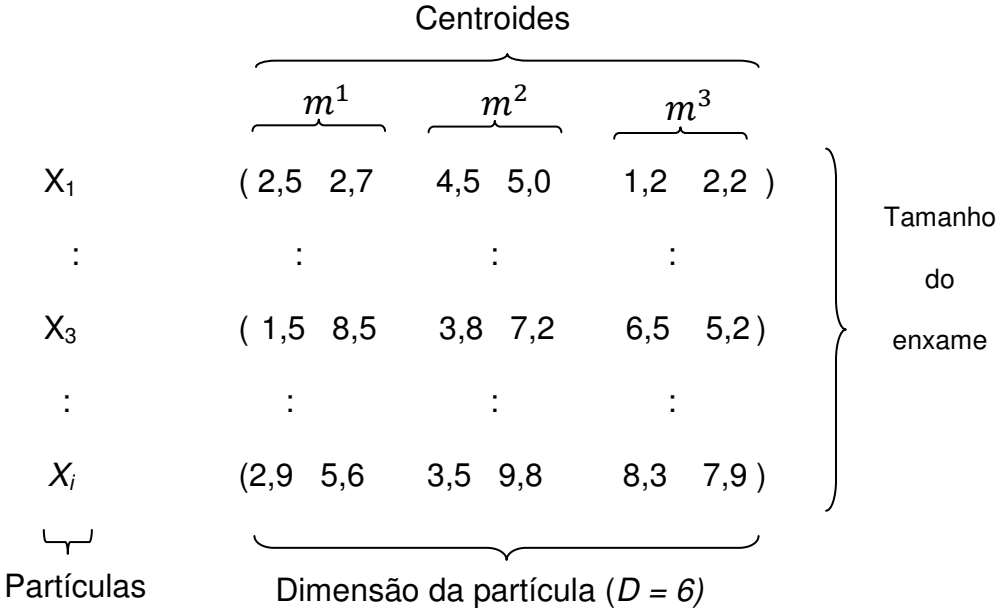

Figura 11 - Estrutura da partícula

Na Figura 11, acima, *i = 1 .. N*, onde *N* representa o número de partículas do enxame. Cada partícula representa uma solução para o problema de k-clusterização apresentado, onde o número de *clusters* é *k* = 3 e o

centroide que representa os objetos de cada cluster tem dimensão *d = 2*. Cada partícula então possui dimensão *D = k * d = 6.*

3.2.2 Função de Avaliação

Conforme visto na Seção 2.2.5, a função de avaliação é dependente do problema de otimização. No caso do algoritmo proposto para resolver um problema de k-clusterização, o objetivo é minimizar a soma das distâncias entre todos os objetos e seus respectivos *clusters,* ou seja, a partícula de melhor qualidade *(fitness)* será aquela que tiver os centroides cuja distância euclidiana entre eles e seus objetos seja a menor, ou seja, a menor distância *intracluster*.

Considerando que cada partícula é representada como sendo: $X_i = (m^1, \ldots, m^j, \ldots, m^k)$, onde m^j representa o *j-ésimo* centroide da partícula X_i, tem-se que o valor da aptidão ou função de *fitness* de cada partícula é dado por:

$$f(X_i) = \sum_{\forall x^p \in C_j} \left\| x^p - m^j \right\|, \quad com\, j = 1..k. \tag{3.4}$$

Com $i = 1..N$, $p = 1..\sigma_j$, $j = 1..k$. Onde d é o número de dimensões dos objetos e centroides, *N* é o número de partículas do enxame, σ_j é o número de objetos de dados pertencentes ao cluster C_j e *k* é o número de *clusters,* m^j é o j-ésimo centroide da i-ésima partícula e x^p é o p-ésimo vetor de dados pertencente ao *cluster* C_j.

3.2.3 Estrutura de vizinhança

Para o algoritmo proposto neste trabalho, optou-se pela estrutura social de vizinhança em estrela**,** onde todas as partículas são interconectadas, conforme Figura 7, encontrada na Seção 2.2.6. Tal estrutura de vizinhança é também mais recomendada quando se usa o PSO *Gbest* (ENGELBRECHT,

2005), cujo modelo foi o escolhido neste algoritmo, onde o tamanho da vizinhança é o tamanho do enxame. Esta estrutura é a que permite uma convergência mais rápida entre as estruturas de rede sociais e também é a mais recomendada para problemas unimodais. Pelo fato de se utilizar uma estratégia de aceleração onde o ILS devolve ao PSO uma partícula de alta qualidade esta escolha também se justifica, pois todas as partículas irão convergir mais rapidamente em direção à referida partícula melhorada.

3.2.4 Equação da velocidade e posição

Na variação do PSO, proposta neste trabalho, a equação da velocidade se mantêm praticamente igual à equação da velocidade usada no PSO padrão, Equação 2.2.1.

No EPSO-ILS a equação da velocidade, que rege toda a movimentação da partícula fica da seguinte forma:

$$V_{ij}(t+1) = p(t)\,.V_{ij}(t) + c_1\,.\,r_{1j}\,.\left(Pbest_{ij}(t) - X_{ij}(t)\right) + c_2\,.\,r_{2j}\,.\left(Gbest(t) - X_{ij}(t)\right) \quad (3.5)$$

Na Equação 3.5, acima, *p(t)* representa um vetor de valores de pesos inerciais a serem aplicados sobre a velocidade inicial da partícula a cada intervalo de tempo *t*. Nesta abordagem os valores contidos no vetor *p(t)* são selecionados utilizando-se um número caótico, conforme será descrito na Seção 3.3.4.

A atualização da posição da partícula continua sendo dada pela Equação 2.2.3.

3.2.5 Inicialização da posição e fixação da velocidade das partículas

Para a inicialização correta da posição das partículas, elas devem ser posicionadas de maneira uniforme de modo a cobrir todo o espaço de busca. Desta forma, considerando a estrutura da partícula mostrada na Figura 11, no

algoritmo proposto cada dimensão da partícula é inicializada com um valor aleatório entre o valor máximo e o valor mínimo alcançado por todos os objetos naquela dimensão.

Com respeito à velocidade, conforme visto na Seção 2.2.8.5, embora as velocidades das partículas possam ser inicializadas com valores diferentes de 0, isto viola a regra natural de que as partículas estão inicialmente paradas e sendo assim devem ter velocidade igual a zero, desta forma neste método proposto inicializou-se a velocidade inicial de todas as partículas e em todas as dimensões com o valor zero.

Ainda com respeito à velocidade, no método proposto utiliza-se a fixação de velocidade máxima (*Vmax*) e velocidade mínima (*Vmin*). Bons resultados foram obtidos fazendo-se a fixação da velocidade máxima e mínima em cada dimensão em um quarto do valor de cada dimensão do espaço de busca, ou seja: $\mathrm{Vmax_{ij}} <= 1/4 * (\mathrm{X_{ij}})$ e $\mathrm{Vmin_{ij}} >= -1/4 * (\mathrm{X_{ij}})$.

3.2.6 Algoritmo EPSO-ILS

Abaixo encontra-se o algoritmo proposto EPSO-ILS para a Clusterização de Dados.

Algoritmo EPSO-ILS

Início

Selecionar base de dados a clusterizar;

Receber número de objetos da base (*n*);

Receber dimensão dos objetos (*d);*

Receber número de centroides (*k);*

Determinar dimensão das partículas (*D = k * d);*

Determinar número de partículas de acordo com Equação 3.6;

Determinar limites inferior e superior de cada dimensão;

Inicializar posição das partículas $\mathrm{X_i}$ dentro dos limites de cada dimensão;

Inicializar com zero velocidade das partículas em cada dimensão $\mathrm{V_{ij}}$;

Inicializar constantes $c_1 = 1{,}5$ e $c_2 = 2{,}0$;

Inicializar vetor de peso inercial *p* com valores de 0,6 a 0,2*;*

Determinar valor da aptidão inicial de cada partícula ($Pbest_i$);

Determinar valor inicial da partícula $Gbest$;

Determinar número máximo de iterações de acordo com Equação 3.6;

Para *iter_pso = 1* **Até** *iter_max_pso* **Faça**

Calcular o índice caótico do vetor *p*;

Para *i = 1* **Até** *N* **Faça**

Para *j = 1* **Até** *D* **Faça**

Atualizar velocidade da partícula usando Equação 3.5;

Atualizar posição da partícula Equação 2.2.3;

Fim_para;

Calcular nova aptidão da partícula de acordo com Equação 3.4;

Fim_para;

Se *iter_pso = 1* **Então**

Aplicar estratégia de aceleração (chamada do ILS);

Atualizar partícula do PSO caso tenha tido melhora no ILS;

Fim_se;

Atualizar partículas $Pbest_i$;

Atualizar partícula $Gbest$;

Fim_Enquanto;

Fim_Algoritmo.

3.2.7 Fluxograma do EPSO-ILS

Abaixo é apresentado o fluxograma do algoritmo proposto.

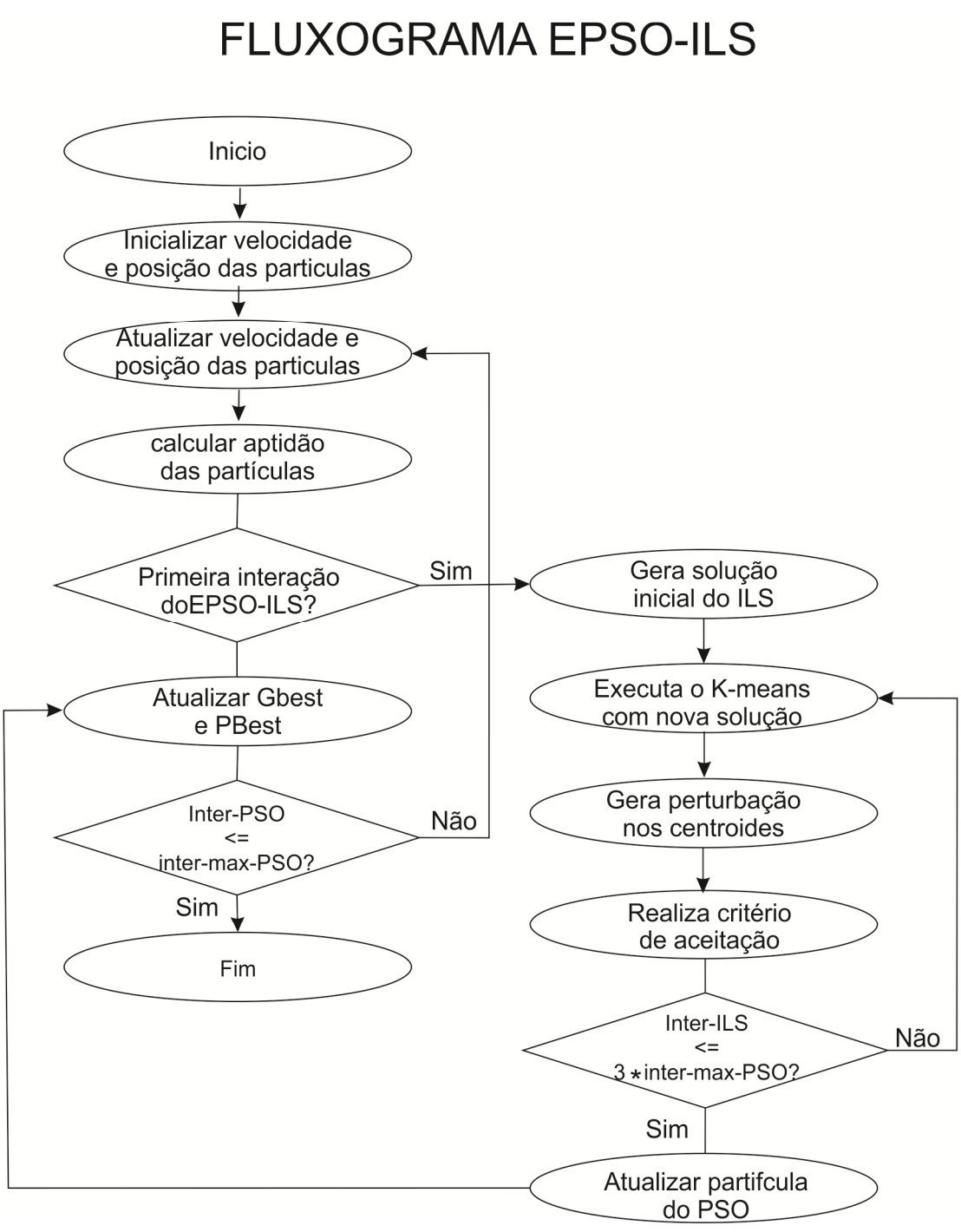

Figura 12 - Fluxograma do EPSO-ILS

3.3 Configuração dos parâmetros do EPSO-ILS

Nesta subseção detalham-se os valores utilizados para configurar os parâmetros usados no algoritmo proposto com base na fundamentação teórica da Seção 2.2.

3.3.1 Tamanho do enxame

Embora o número ideal de partículas gire em torno de 10 a 30, de acordo com Engelbrecht (2005), em nosso experimento, para efeitos de comparação, utilizou-se o tamanho do enxame dependente do problema, ou seja, é dependente do número de *clusters* que se deseja agrupar os objetos e do número de dimensões de cada objeto, ou seja, o número de partículas *n* é dado por:

$$N = 3 * k * d \tag{3.6}$$

Onde k é o número de *cluster*s e d é a dimensão ou número de características de cada objeto.

3.3.2 Número de iterações

O número máximo de iterações realizadas pelo EPSO-ILS, que também utilizou-se para efeitos de comparação com os outros algoritmos, é dado por:

$$iter\_\max\_pso = 10 * k * d \tag{3.7}$$

Onde k e d são o número de *clusters* e o número de dimensões dos objetos dos *clusters*, respectivamente.

3.3.3 Coeficientes de Aceleração

Como visto na Seção 2.2.8.3, os coeficientes de aceleração c_1 e c_1 determinam a natureza estocástica da busca e controlam o quanto as partículas vão se mover em direção ao *Pbest* ou *Gbest.* No caso do algoritmo proposto, em que sugerimos uma estratégia em que o ILS melhora uma partícula do PSO e devolve a ele para ser refinada, é mais lógico que se dê um maior valor ao coeficiente c_1 para que as partículas se movam com mais velocidade em direção ao componente *Gbest.* Porém, mesmo usando esta lógica, foram testados diversos valores para os coeficientes de aceleração, fazendo-se $c_1 < c_2$, $c_1 = c_2$ e $c_1 > c_2$. Os valores com os quais se obtiveram melhores resultados do algoritmo proposto foram: $c_1 = 1{,}5$ e $c_2 = 2{,}0$.

3.3.4 Peso Inercial

Esta seção merece especial atenção, pois a forma como o peso inercial é aplicado à velocidade inicial é modificada neste algoritmo proposto. Esta mudança é feita pelo uso de um componente caótico para determinar o valor do peso inercial usado em cada iteração para "frear" a partícula.

Enquanto no PSO padrão os valores do vetor *w*, que representam os pesos inerciais a serem aplicados à velocidade inercial a cada iteração, são selecionados de forma gradativa e vão diminuindo a cada iteração, no EPSO-ILS os valores dos pesos inerciais não decrescem gradativamente. Eles são distribuídos uniformemente no vetor, mas são selecionados utilizando-se um gerador de números caóticos. Neste método proposto, os números caóticos, que se usam para substituir os números randômicos, são gerados através da Equação 2.4.1 do Mapa Logístico introduzido por May (1976).

A seleção de forma caótica do valor do peso a ser aplicado em cada intervalo de tempo, faz com que ao longo das iterações pesos maiores ou menores sejam aplicados às partículas independentemente das iterações estarem no início ou no final. Entretanto, como a seleção é baseada não por um número randômico, e sim por um número caótico, os valores dos pesos selecionados tendem a ser mais frequentes nas extremidades do vetor *p*, ou seja, onde estes valores são os maiores ou os menores.

Propomos usar um vetor de pesos inerciais, que foi identificado por *p,* com um componente caótico para ter os mesmos benefícios obtidos com o comportamento caótico dos componentes *Gbest* e *Pbest* da equação de velocidade original do algoritmo CPSO vista na Equação 2.4.2.

O peso inercial armazenado no vetor *p*, é inicializado com valores dispostos linearmente de 0.6 a 0.2 nas posições do vetor. O tamanho do vetor é determinado pelo número de iterações do PSO, ou seja, o vetor contêm tantas posições quantas forem o número de iterações do PSO.

Desta forma os valores selecionados do índice do vetor *p* tem a característica apresentada na Figura 12, abaixo.

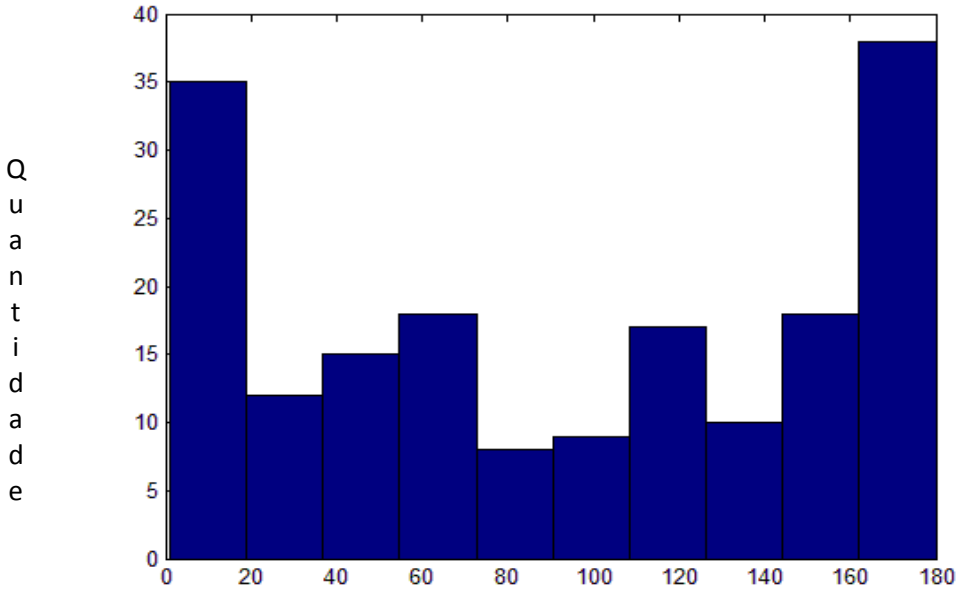

Figura 12 - Faixa de valores gerados pelo mapa logístico com k = 4

O histograma acima foi gerado com base em 180 valores gerados pela equação do mapa logístico 2.4.1 para a clusterização de uma base de dados com *k* = 6 e d = 3, o que dá um número de iterações = 180, segundo a Equação 3.7, encontrada na Seção 3.3.2. Pode-se notar que valores próximos às extremidades são mais frequentes, ou seja, a maioria dos números gerados são mais próximos a zero ou a 180. Esta característica, como mencionado em Mendel (2011), permite que uma velocidade inercial assuma valores mais altos ou mais baixos em muitas iterações, o que permite que a partícula possa

intensificar a busca, com valores pequenos, ou sair de ótimos locais quando são encontrados valores mais altos.

Abaixo é mostrado o algoritmo para gerar o valor de *ind_p*, índice do vetor de pesos inerciais *p,* cuja faixa de valores gerados é mostrada no histograma acima.

Algoritmo *Geração_de_índice_caótico_do_vetor_p*
Inicio
Gerar número caótico (*n_c*) inicial de forma aleatória (*n_c = rand*);
Para *iter_pso = 1* **Até** *iter_max_pso* **Faça**
Inicio
n_c = 4 * *n_c* * (1 - *n_c*);
ind_p = *n_c* * *iter_max_pso*;
Se round(*ind_p*) = 0 **Então**
ind_p = 1;
Senão Se round(*ind_p*) = *iter_max_pso* **Então**
ind_p = iter_max_pso;
Senão
ind_p = round(n_c * iter_max_pso);
Fim_se;
Fim_se;
Fim_para;
Fim_algoritmo.

Observa-se que o algoritmo acima gera o valor das posições do vetor ou índices do vetor e não os valores do vetor. Os valores do vetor já são atribuídos de forma linear ao longo do vetor no início do algoritmo, conforme ilustrado na Figura 13, que mostra o vetor *e* cujos valores na faixa de 0,60 a 0,20 são distribuídos linearmente ao longo de suas 180 posições.

p	0,6000	0,5978	0.5955	. . . . .	0,2045	0,2022	0,2000
	1	2	3	. . . . .	178	179	180

Figura 13 - Exemplo do vetor *p* de pesos inerciais

3.4 Estratégia de aceleração

A principal mudança proposta nesta versão híbrida é o uso da estratégia de aceleração através da chamada da meta-heurística ILS. Nesta estratégia utilizou-se uma chamada ao algoritmo ILS na primeira iteração do PSO somente. Após a avaliação da aptidão de todas as partículas da primeira iteração, o ILS é chamado e recebe os centroides da partícula com melhor *fitness*.

No algoritmo ILS utilizado nesta estratégia, que denominou-se de ILS-KM, cujos resultados são apresentados na Seção 4.8, é feita uma busca local com o algoritmo *K-means* padrão para refinar os centroides gerados pela perturbação do ILS. Neste caso o *K-means* realiza uma intensificação da busca em uma região considerada promissora, até que um ótimo local seja encontrado, conforme ilustrado na Figura 14, a seguir.

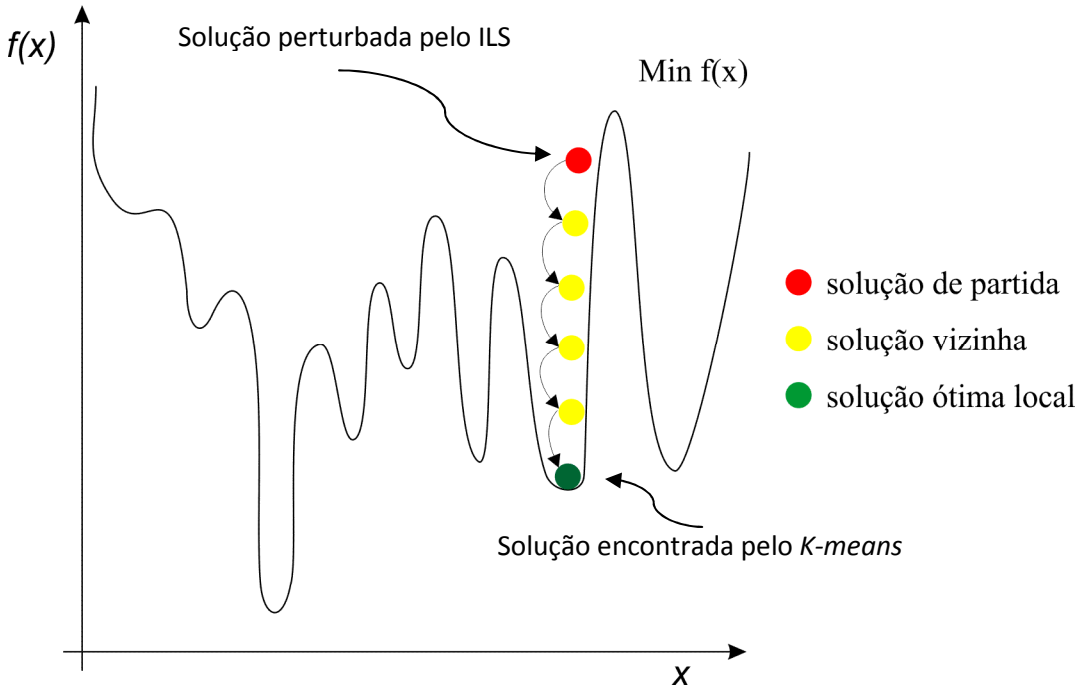

Figura 14 - Busca local do ILS

Três níveis de **perturbação** são aplicados nas soluções do ILS. As perturbações consistem na aplicação de uma pequena movimentação em todos os centroides da partícula num raio de até 10%, 5% e 1% respectivamente, da

magnitude de sua posição atual. A cada iteração do ILS todos os centroides são movidos aleatoriamente em cada dimensão de acordo com o nível de perturbação da iteração. Os testes empíricos mostraram que a aplicação de perturbações diferentes a cada dimensão obteve melhores resultados do que quando se aplicou a mesma perturbação em todas as dimensões dos centroides. A perturbação inicia em 10% e finaliza com 1%, ou seja, aplica-se uma perturbação maior no início das iterações do ILS, permitindo uma maior diversificação da busca no início das iterações e intensificando a busca gradativamente.

Após a aplicação da perturbação é chamado o *K-means* para fazer uma busca local com os novos centroides perturbados. O **critério de aceitação** usado possui um componente randômico. A solução obtida pelo *K-means* só é aceita, para ser perturbada novamente, caso seu valor seja inferior a (*rand* + 1) * (valor da melhor solução encontrada), ou seja, caso *f(s") <= (rand + 1) * f(s).*

A Figura 15, abaixo, ilustra o método de perturbação que utilizou-se no ILS proposto. É ilustrado apenas um nível de perturbação num problema de k-clusterização com *k=3* e objetos de duas dimensões, ou seja, *d = 2.*

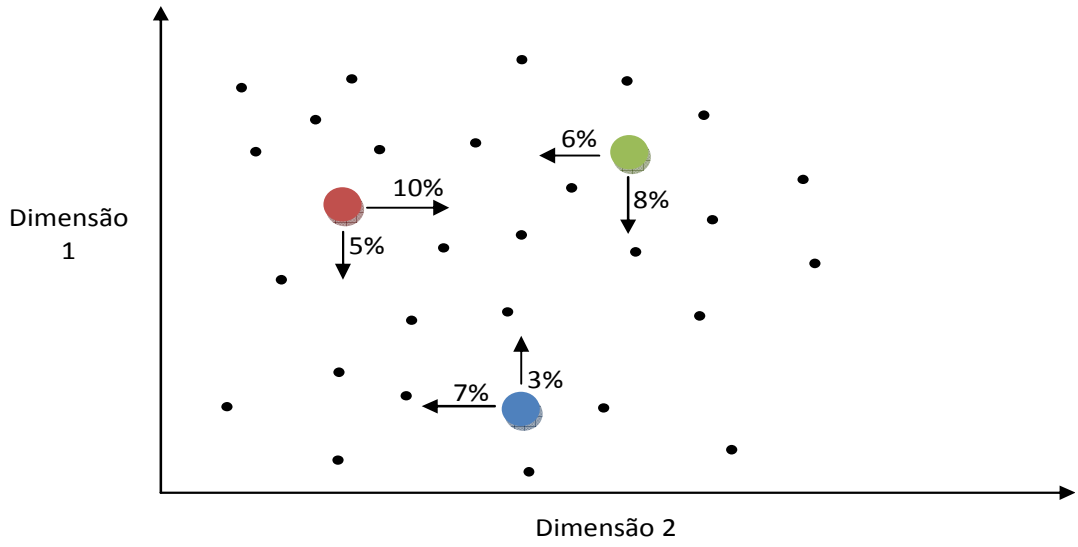

Figura 15 - Movimento dos centroides - perturbação nível 1: <= 10%

Na Figura 15, acima, os círculos pequenos representam os pontos a clusterizar e o círculos grandes representam os centroides. Observa-se que o percentual de movimento aplicado a cada dimensão de cada centroide é variável, porém é limitado ao nível da perturbação, ou seja, no primeiro nível o centroide pode mover-se em até 10% de sua posição em qualquer dimensão,

no segundo nível em até 5% e no menor e último nível de perturbação pode mover-se em no máximo 1% em qualquer dimensão. Observa-se ainda que todos os centroides se movimentam na iteração do ILS.

Para determinar o número de iterações do ILS usou-se a relação com o número de *clusters* e dimensões da base de dados que se deseja clusterizar, da seguinte forma:

$$iter\_\max\_ils = (10 * \mathrm{k} * \mathrm{d}) * 3 \qquad (3.8)$$

onde *k,* como visto, é o número de *clusters* e *d* a dimensão de cada centroide. Este valor equivale a três vezes o número de iterações executadas no PSO e não precisa ser um número muito grande, visto que o ILS recebe uma solução do PSO que está numa região promissora do espaço de busca e por isso vai convergir rapidamente para uma possível solução ótima sem precisar executar muitas iterações.

Capítulo 4

RESULTADOS COMPUTACIONAIS

Neste capítulo apresentam-se todos os aspectos relativos ao desenvolvimento do trabalho, tais como o ambiente de desenvolvimento, as características das bases de dados de referência utilizadas para os testes, os resultados computacionais, a análise dos resultados e da parametrização utilizada, e, ao final, uma discussão sobre os resultados computacionais.

4.1 Ambiente de Desenvolvimento

O algoritmo proposto foi implementado utilizando a linguagem de programação MatLab (Matrix Laboratory) versão 7.11.0 (R2010b), por ser um ambiente de programação de alto nível, possuindo características de aplicativo (facilidade para o usuário) e de linguagem de programação e ainda por permitir a fácil e rápida manipulação de matrizes e vetores bastante utilizadas para o desenvolvimento do algoritmo proposto e, principalmente, por possibilitar a visualização rápida de gráficos.

De modo a demonstrar a eficiência do EPSO-ILS, foram comparados os resultados obtidos com os seguintes algoritmos: *K-means*, GA, KGA, ILS-KM, PSO, K-PSO, ACPSO e CPSO.

O EPSO-ILS foi executado um total de 20 vezes para cada uma das cinco bases de dados de referência constantes na Tabela 1, a seguir.

As medidas reportadas são média aritmética, desvio padrão e melhor resultado, ou seja, a menor distância euclidiana *intracluster*, obtida em 20 execuções seguidas.

4.2 Bases de Dados

O experimento consiste em aplicar a versão do EPSO-ILS para a Clusterização de cinco bases de dados de referência: Iris, *Vowel, Wine*, CMC, e Câncer, que contêm diferentes números de *clusters* e dimensões, para

comparação dos resultados com outros algoritmos de clusterização, conforme Tabela 1. As bases foram baixadas de ftp://ftp.ics.uci.edu/pub/machine-learning-databases/.

A seguir as características das bases utilizadas e suas descrições.

Base de Dados	*Clusters*	Dimensões	Quantidade de Objetos
Vowel	6	3	871
Iris	3	4	150
CMC	3	9	1473
Câncer	2	9	683
Wine	3	13	178

Tabela 1: Características das bases de dados

1 - *Vowel*: a base de dados *Vowel* (vogal) é constituída de 871 sons vocálicos indígenas Telugu. Ele inclui as três características correspondentes às primeira, segunda e terceira frequências de vogal, e seis classes que se sobrepõem.

2 - Iris: a base de dados Iris é composta de três espécies diferentes: Iris setosa, iris virginica e versicolour íris. Para cada espécie, 50 amostras foram coletadas com quatro características cada, comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala.

3 - CMC: a base de dados Método de Escolha Contraceptiva, conhecida como CMC é composta por amostras que consistem de dados obtidos de mulheres casadas que estavam ou não grávidas ou não tinham certeza do seu estado de gravidez no momento em que as entrevistas foram realizadas. Ele prevê que a escolha do método contraceptivo de uma mulher se dá com base em suas características demográficas e sócio-econômicas.

4 - Câncer: a base de dados *Breast Cancer* (Cancer de mama) de Wisconsin, consiste de 683 objetos identificados por nove características: espessura, uniformidade de tamanho da célula, forma celular, adesão marginal, tamanho da célula epitelial, núcleos, cromatina, nucléolos normais e mitose.

5 - *Wine*: a base de dados *Wine* (vinho) consiste em 178 objetos identificados por 13 características: teor de álcool, quantidade de ácido málico, o teor de cinzas, a alcalinidade das cinzas, a concentração de magnésio, fenóis, flavonóides, fenóis não-flavonóides e proantocianinas e intensidade de cor, matiz e OD280/OD315 de vinhos diluídos. Estas características foram obtidas por análise química dos vinhos que são produzidos na mesma região, na Itália, mas são derivados de três cultivos diferentes.

É importante ressaltar que todas as bases de dados descritas acima possuem somente valores numéricos ou atributos escalares, permitindo assim que seja usada a distância euclidiana para definir a pertinência de um objeto a um determinado *cluster.*

4.3 Resultados computacionais

De modo a demonstrar a eficiência do EPSO-ILS, foram comparados os resultados obtidos com os seguintes algoritmos: *K-means*, GA, KGA, PSO, K-PSO, CPSO, ECPSO, ACPSO, conforme Tabela 2, a seguir.

O EPSO-ILS foi executado um total de 20 vezes para cada base de dados utilizada, ou seja, o mesmo número de execuções dos outros algoritmos.

As medidas reportadas são média aritmética, desvio padrão e melhor resultado obtido em 20 execuções seguidas.

BASE	MEDIDA	*K-MEANS*	GA	KGA	PSO	K-PSO	CPSO	ACPSO	ECPSO	EPSO-ILS
VOWEL	MÉDIA	159242,87	390088.24	149368.45	168477,00	149375,70	151337,00	149051,84	149020,11	**148967,39**
	DESVIO	916	N/A	N/A	3715,73	155,56	3491,43	67,27	47,65	**0,34**
	MELHOR	149422,26	383484,15	149356,01	163882,00	149206,10	148996,50	148970,84	148967,27	**148967,24**
IRIS	MÉDIA	106,05	135,40	97,10	103,51	96,76	96,90	96,66	96,65	**96,65**
	DESVIO	14,11	N/A	N/A	9,69	0,07	0,303	0,001	0,00	**0,00**
	MELHOR	97,33	124,13	97,10	96,66	96,66	96,66	96,66	96,65	**96,65**
CMC	MÉDIA	5693,60	N/A	N/A	5734,20	5532,90	5532,23	5532,20	5532,18	**5532,18**
	DESVIO	473,14	N/A	N/A	289,00	0,09	0,04	0,01	0,00	**0,00**
	MELHOR	5542,20	N/A	N/A	5538,50	5532,88	5532,19	5532,19	5532,18	**5532,18**
CÂNCER	MÉDIA	2988,30	N/A	N/A	3334,60	2965,80	2964,49	2964,42	2964,38	**2964,38**
	DESVIO	0,46	N/A	N/A	357,66	1,63	0,12	0,03	0,00	**0,00**
	MELHOR	2987	N/A	N/A	2976,30	2964,50	2964,40	2964,39	2964,38	**2964,38**
WINE	MÉDIA	18061,00	N/A	N/A	16311,00	16294,00	16292,90	16292,31	16292,18	**16292,18**
	DESVIO	793,21	N/A	N/A	22,98	1,70	0,78	0,03	0,00	**0,00**
	MELHOR	16555,68	N/A	N/A	16294,00	16292,00	16292,19	16292,18	16292,18	**16292,18**

N/A significa dado não avaliado. Os melhores resultados são indicados em negrito.

Tabela 2 - Resultados computacionais

Na tabela acima, os resultados de GA podem ser encontrados em Murthy and Chowdhury (1996), os resultados de KGA podem ser encontrados em Bandyopadhyay and Maulik (2002). Os resultados de *K-means*, PSO, K-PSO, podem ser encontrados em Kao *et al*. (2008). CPSO e ACPSO podem ser encontrados em Chuang (2011), e o ECPSO em Coelho Filho *et al*. (2013b).

Os resultados do EPSO-ILS, obtidos neste trabalho, já foram publicados em Coelho Filho *et al*. (2013a).

4.4 Análise dos resultados

Verifica-se, com base na tabela acima, que o EPSO-ILS é bastante eficiente e obtêm melhores resultados tanto na média aritmética quanto no desvio padrão e no melhor resultado na maioria das bases de dados testadas. Observa-se, ainda, que o algoritmo proposto possui uma excelente robustez trazendo na maioria das execuções um desvio padrão insignificante para todas as bases de dados testadas.

A primeira comparação foi feita com o *K-means*, por ser este um dos algoritmos determinísticos mais utilizados para resolver o problema de clusterização. Esta comparação permitiu mostrar a melhor performance do algoritmo proposto em todas as comparações feitas, desvio padrão, média e melhor resultado. Verificou-se que o *K-means* é um algoritmo muito rápido e com poucas iterações já dá bons resultados, mas fica preso facilmente em ótimos locais.

Em seguida fez-se uma comparação com os trabalhos de Murthy and Chowdhury (1996) e Bandyopadhyay and Maulik (2002) com os algoritmos GA e KGA respectivamente, nos quais é usado um algoritmo da classe dos Algoritmos Evolucionários, o Algoritmo Genético. Este algoritmo, sozinho, não se mostrou muito eficiente para resolver o problema de Clusterização de Dados, mas quando usado com uma abordagem híbrida usando o *K-means* como estratégia de aceleração, obtém melhores resultados.

Depois foi feita uma comparação com o trabalho de Kao *et al*. (2008), com os algoritmos PSO e KPSO, onde se observou, de acordo com os resultados, que o algoritmo PSO sozinho ainda é passível de melhora e que quando usado com o *K-means*, para melhorar a qualidade de uma das partículas a serem clusterizadas por ele, apresenta resultados muitos bons.

O algoritmo CPSO, proposto por Chuang (2011), apresenta ótimos resultados para a clusterização, como se pode ver na Tabela 2. Ele faz uso de números caóticos gerados por uma equação de Mapa Logístico para substituir os números randômicos na multiplicação das partículas *Gbest* e *Pbest* da equação original do PSO. No mesmo trabalho é apresentada uma versão do CPSO utilizando uma estratégia de aceleração, denominado ACPSO, que usa o *K-means* para melhorar aleatoriamente um terço das partículas iniciais geradas para o PSO. Esta estratégia logra bastante êxito como se pode ver nos resultados apresentados na Tabela 2.

Em Coelho Filho *et al*. (2013b) também é utilizada uma estratégia de aceleração baseada em Chuang (2011), usando o *K-means* e números caóticos, cujo algoritmo foi chamado de ECPSO. Nesta abordagem foi utilizado, além dos números caóticos para multiplicar os componentes *Gbest* e *Pbest*, um componente caótico para selecionar o peso inercial a ser aplicado à velocidade da partícula. Como estratégia de aceleração, em um terço das iterações do

PSO selecionou-se uma partícula aleatoriamente para ser melhorada pelo *K-means*. Este método obteve melhores resultados que o ACPSO na maioria das bases testadas.

Ao se analisar os resultados apresentados na Tabela 2, quanto aos melhores resultados obtidos, verifica-se que o algoritmo proposto neste trabalho, EPSO-ILS, obteve melhores resultados, considerando a menor distância *intracluster*, tanto na média aritmética, desvio padrão e no melhor resultado em todas as bases de dados testadas se comparados com as melhores soluções obtidas de oito algoritmos da literatura.

4.5 Análise da parametrização

O desempenho do PSO é muito sensível à calibração dos parâmetros utilizados na sua configuração. Contudo, muitos dos valores dos parâmetros utilizados foram os mesmos de outros algoritmos, para efeitos de comparação. Os valores dos parâmetros que dizem respeito somente ao algoritmo proposto, como por exemplo faixa de valores do peso inercial e número de iterações do ILS foram definidos de forma empírica com o objetivo de obter a melhor combinação dos parâmetros que produzem um melhor desempenho do EPSO-ILS.

Abaixo, na Tabela 3, é apresentado um resumo dos parâmetros utilizados para configurar o algoritmo proposto. Estes valores foram apresentados no Capítulo 3, Metodologia.

Número de partículas	Número de iterações do PSO	Faixa do peso inercial	c1	c2	Vel_max	Vel_min	Num. terações do ILS	Níveis de perturbação do ILS	Critério de aceitação do ILS
3 * *k* * d	10 * *k* * d	0,6 a 0,2	1,5	2,0	1/4 * Xi	- 1/4 * Xi	10 * *k* * d)*3	10%, 5% e 1%	(*rand* + 1) * melhor solução encontrada

Tabela 3 - Valores dos parâmetros utilizados para o EPSO-ILS

Alguns valores de parâmetros que foram utilizados para efeitos de comparação se mostraram maiores que os ideais para o algoritmo proposto, pois, considerando a estratégia de aceleração utilizada, o EPSO-ILS pôde

chegar a praticamente os mesmos resultados, com um número menor de partículas e de iterações, diminuindo assim o custo computacional.

Em relação ao número de partículas utilizado, este mostrou-se mais do que suficiente para atingir uma boa convergência das partículas. Usou-se este número principalmente para efeitos de comparação, porém conforme sugerido por Engelbrecht (2005), foi verificado que se forem mantidos os mesmos parâmetros da Tabela 3, acima, o algoritmo proposto alcança praticamente os mesmos valores com o número de partículas n = 30, conforme verificado Tabela 4, abaixo.

BASE	**MEDIDA**	**EPSO-ILS N = 3*k*d iter_max_pso=10*k*d**	**EPSO-ILS N = 30 iter_max_pso=10*k*d**
Vowel **(k=6) (d=3)**	MÉDIA	148981,32	149045,14
	DESVIO	34,31	43,73
	MELHOR	148967,24	148967,24
	TEMPO (uma exec.)	65,35 segundos	42,43 segundos
Wine **(k=3) (d=13)**	MÉDIA	16292,18	16292,8383
	DESVIO	0,00	1,05
	MELHOR	16292.18	16292,205
	TEMPO (uma exec.)	35,10 segundos	17,80 segundos
CMC **(k=3) (d=9)**	MÉDIA	5532,18	5532,31
	DESVIO	0,00	0,23
	MELHOR	5532,18	5532,18
	TEMPO (uma exec.)	2min 47seg	1min 55seg

Tabela 4: Análise da parametrização 1

Utilizou-se nos testes acima as três bases de dados cujas partículas possuem a maior dimensão.

Pode-se observar ainda na tabela acima, com a inclusão da variável tempo, que a diminuição considerável no número de partículas diminui sensivelmente o custo computacional para executar o algoritmo proposto.

Com relação ao número máximo de iterações do EPSO-ILS, que foi usado para efeitos de comparação, verificou-se nos experimentos que um número menor de iterações já seria suficiente para que as partículas alcançassem a convergência para uma solução ótima local ou global. Para provar esta afirmação executou-se o mesmo algoritmo com somente um terço do número de iterações realizadas pelo PSO. Vale ressaltar que o número de iterações do ILS fica sendo $iter\_max\_ils = (10 * \mathrm{k} * \mathrm{d})/3$ e que o número de partículas continua sendo $N = 3 * \mathrm{k} * \mathrm{d}$. Os resultados são apresentados na Tabela 5, abaixo.

BASE	MEDIDA	EPSO-ILS iter_max_pso=10*k*d	EPSO-ILS iter_max_pso=(10*k*d)/3
Vowel **(k=6) (d=3)**	MÉDIA	148981,32	149016,96
	DESVIO	34,31	46,36
	MELHOR	148967,24	148967,53
	TEMPO (uma exec.)	50,26 segundos	16,77 segundos
Wine **(k=3) (d=13)**	MÉDIA	16292,18	16292,68
	DESVIO	0,00	0,69
	MELHOR	16292,18	16292,18
	TEMPO (uma exec.)	14,56 segundos	6,15 segundos
CMC **(k=3) (d=9)**	MÉDIA	5532,18	5532,29
	DESVIO	0,00	0,06
	MELHOR	5532,18	5532,20
	TEMPO (uma exec.)	167 segundos	59,79 segundos

Tabela 5: Análise da parametrização 2

Considerando a estratégia de aceleração utilizada, verificou-se com base nos resultados da Tabela 6, abaixo, que o valor ideal para o número de partículas é aproximadamente 30 e o número máximo de iterações do PSO seria um terço do número utilizado na comparação com os outros algoritmos.

Com esta parametrização o custo computacional cai consideravelmente e mantêm-se praticamente os mesmos resultados.

BASE	MEDIDA	EPSO-ILS iter_max_pso=10*k*d N=3*k*d	EPSO-ILS iter_max_pso=(10*k*d)/3 N=30
Vowel **(k=6) (d=3)**	MÉDIA	148912,32	149052,10
	DESVIO	34,31	40,96
	MELHOR	148967,24	148967,95
	TEMPO (uma exec.)	65,35 segundos	11,70
Wine **(k=3) (d=13)**	MÉDIA	16292,18	16292,83
	DESVIO	0,00	0.61
	MELHOR	16292,18	16292,21
	TEMPO (uma exec.)	14,56 segundos	6,41 segundos
CMC **(k=3) (d=9)**	MÉDIA	5532,18	5532,29
	DESVIO	0,00	0,07
	MELHOR	5532,18	5532,21
	TEMPO (uma exec.)	167 segundos	39,80 segundos

Tabela 6: Análise da parametrização 3

4.6 Análise da estrutura de vizinhança

Um dos principais aspectos que determinam o desempenho do PSO é a estrutura de vizinhança utilizada, como visto na Seção 2.2.6. Em Engelbrecht (2013a) é apresentado um estudo detalhado sobre o uso do modelo *Gbest*, onde a vizinhança de uma partícula é todo o enxame, e do modelo *Lbest,* onde a vizinhança de uma partícula são seus dois vizinhos adjacentes, na tentativa de determinar qual dos dois modelos é mais indicado, no entanto ele conclui que nenhum dos modelos pode ser considerado melhor até mesmo quando se considera problemas específicos. Contudo, com o objetivo de determinar o modelo a ser utilizado neste método para o problema de Clusterização de Dados, foram realizados alguns experimentos onde foram testadas estas duas

estruturas principais. Neste teste foi utilizado o PSO puro nos modelos *Gbest* e *Lbest,* utilizando-se os mesmos parâmetros da tabela acima, *não sendo introduzido por enquanto o peso inercial caótico e a chamada ao ILS*. Cada algoritmo foi executado 20 vezes em cada uma das três bases de dados abaixo, para determinar a melhor média, desvio padrão e melhor resultado alcançado por ambos os modelos, cujos resultados encontram-se na Tabela 7, a seguir.

BASE	**MEDIDA**	**PSO *(Lbest)***	**PSO *(Gbest)***
Vowel	MÉDIA	155380,64	**151802,51**
	DESVIO	3614,83	**5005,30**
	MELHOR	150923,20	**148967,24**
CMC	MÉDIA	5603,09	**5532,20**
	DESVIO	31,43	**0,045**
	MELHOR	5550,95	**5532,18**
Wine	MÉDIA	16300,50	**16292,45**
	DESVIO	4,65	**0,246**
	MELHOR	16294,83	**1629218**

Tabela 7: Comparação entre os modelos PSO-*Gbest* e PSO-*LBest* para Clusterização de Dados

Verificou-se, com base nos resultados acima, que para todas as bases de dados testadas o modelo PSO *Gbest* obteve, na maioria das vezes, melhor desempenho tanto na média como no desvio padrão e no menor resultado. Baseado nestes resultados optou-se por usar este modelo de PSO no método proposto neste trabalho.

4.7 Análise do uso do componente caótico

Ao se observar os resultados do CPSO constantes na Tabela 2 de Resultados Computacionais, verifica-se que ele possui excelentes resultados ocasionados pelo uso de componentes caóticos para multiplicar os valores de *Pbest* e *Gbest,* conforme justificado por Mendel (2011) na Seção 2.4.2. Com base nesta ideia, optou-se neste trabalho por utilizar um componente caótico na

escolha do valor do peso inercial que multiplica a velocidade anterior, deixando que a partícula dê passos maiores mesmo ao final das iterações. Foi criado então um algoritmo independente que denominou-se de EPSO (*Enhanced* PSO).

Com o objetivo de testar os resultados obtidos com o novo algoritmo, foram feitas 20 execuções do CPSO e do EPSO, ambos no modelo *GBest*. Vale ressaltar que neste experimento não foi utilizada a chamada ao ILS, pois o objetivo foi obter uma comparação mais efetiva e real entre os dois métodos quanto ao uso dos componentes caóticos multiplicando a velocidade (EPSO) e multiplicando os elementos *PBest* e *Gbest (CPSO).* O CPSO foi usado como comparação por não utilizar nenhuma estratégia de aceleração antes do PSO assim como é feito no EPSO.

Inicialmente foram usados os mesmos parâmetros para efeitos de comparação nos dois algoritmos. Os resultados são apresentados na Tabela 8, e em seguida , na Tabela 9, foram utilizados os parâmetros ideais quanto ao número de partículas, ou seja, $n = 30$, e número de iterações *itmax_pso = (3 * k * d) / 3*, nos dois algoritmos. Os resultados são apresentados nas tabelas abaixo.

BASE	MEDIDA	CPSO	EPSO *(sem ILS)*
Vowel	MÉDIA	151337,00	**150770,88**
	DESVIO	3491,43	**3205,12**
	MELHOR	148996,50	**148968,00**
CMC	MÉDIA	**5532,23**	5543.39
	DESVIO	**0,04**	13.49
	MELHOR	5532,19	**5532.18**
Wine	MÉDIA	**16292,90**	16293.24
	DESVIO	**0,78**	0.83
	MELHOR	16292,19	**16292.18**

Tabela 8 - CPSO X EPSO (N=3*k*d)

BASE	MEDIDA	CPSO	EPSO *(sem ILS)*
Vowel	MÉDIA	187010,70	**153202,25**
	DESVIO	15665,85	**4332,06**
	MELHOR	160935,76	**149173,94**
CMC	MÉDIA	6301.25	**5648,56**
	DESVIO	310,10	**49,76**
	MELHOR	5952,60	**5572,24**
Wine	MÉDIA	16680,30	**16300,87**
	DESVIO	286,43	**4,79**
	MELHOR	16355,18	**16294,70**

Tabela 9 - CPSO X EPSO (N=30)

Os resultados das tabelas acima mostram que o desempenho do EPSO é superior, na maioria das vezes, e especialmente quando são usados menos partículas e somente um terço de iterações, o que justifica o uso do

componente caótico na escolha do peso inercial que vai multiplicar a velocidade inicial.

4.8 Análise da configuração do ILS

O ILS utilizado no método proposto, que foi denominado de ILS-KM, por utilizar o *K-means* na busca local, foi implementado inicialmente separadamente, para se poder avaliar a melhor configuração a ser utilizada na hibridização com o PSO. O ILS-KM foi desenvolvido com o objetivo de acelerar a convergência do algoritmo PSO através do refinamento de apenas uma partícula gerada pelo PSO na sua primeira iteração.

Para determinar a melhor configuração dos parâmetros deste algoritmo foram feitos vários testes nos quais foram utilizados diferentes valores para o número de iterações, níveis de perturbação e critérios de aceitação utilizados. Os valores destes parâmetros e os resultados obtidos com o ILS-KM em 20 execuções para clusterizar a base de dados *Vowel,* são mostrados na Tabela 10, abaixo. A base de dados *Vowel* foi usada para testes por ser a que tem a maior quantidade de *clusters.* Os centroides iniciais foram escolhidos de forma aleatória.

Ordem	Número máximo de iterações	Níveis de perturbação do ILS	Critério de aceitação do ILS	Centroides movimentados por iteração	ILS-KM (Media)	ILS-KM (Desvio)	ILS-KM (Mínimo)
1	10 * *k* * d	10%, 5% e 1%	1,5*Melhor solução encontrada	Alguns	149379,63	5,96	149373,09
2	10 * *k* * d	10%, 5% e 1%	1,5*Melhor solução encontrada	Alguns	149384.44	10.96	149369,63
3	10 * *k* * d	10%, 5% e 1%	2*Melhor solução encontrada	Alguns	149378.20	6,53	149369,63
4	10 * *k* * d	10%, 5% e 1%	2*Melhor solução encontrada	Alguns	149373.92	6.34	149369,63
5	10 * *k* * d	10%	2*Melhor solução encontrada	Sempre todos	149373.89	5,90	149369,63
6	**10 * *k* * d**	10%, 5% e 1%	**(rand + 1)*Melhor solução encontrada**	**Sempre todos**	**149370,32**	**1,54**	**149369,63**
7	10 * *k* * d	10%	rand+1,5)*Melhor solução encontrada	Sempre todos	149373,79	5,90	149369,63
8	10 * *k* * d	10%	2*Melhor solução encontrada	Alguns	149378,25	7,86	149369,63
9	10 * *k* * d	5%	2*Melhor solução encontrada	Sempre todos	149371,71	1,81	149369,63
10	10 * *k* * d	15%, 10% e 5%	2*Melhor solução encontrada	Sempre todos	149371,08	6.35	149362,78

11	10 * *k* * d	3%	2*Melhor solução encontrada	Sempre todos	149378.20	7,81	149369,63
12	10 * *k* * d	15%, 10% e 5%	Melhor solução encontrada	Sempre todos	149388.45	7.27	149383.74

Tabela 10 - Configuração do ILS-KM

Verifica-se com base nos resultados apresentados na tabela acima que o melhor resultado obtido com o ILS-KM utiliza os valores encontrados na sequência 6 da tabela. Nesta sequência de 20 execuções foi usado três níveis de perturbação nos valores de 10%, 5% e 1% do valor da posição dos centroides. Observa-se ainda que a cada iteração todos os centroides são movimentados. Uma nova solução é aceita para ser perturbada quando seu valor é menor que até duas vezes o melhor valor já encontrado para uma solução do ILS.

Observa-se ainda o excelente desempenho do algoritmo em relação aos algoritmos comparados da Tabela 2, de resultados, o que faz do ILS uma boa opção para Clusterização de Dados, uma vez que o seu custo computacional é muito inferior ao do PSO por se tratar da melhora de uma única solução inicial. Nota-se também que o ILS chega rapidamente à convergência para uma boa solução e que, portanto, mesmo triplicando-se o número de iterações ainda se chega praticamente aos mesmos resultados para a clusterização.

4.9 Análise da hibridização com o ILS

A ideia de usar o ILS surgiu ao ser observado o funcionamento do algoritmo ACPSO, desenvolvido por Chuang (2011). Neste algoritmo é proposta uma estratégia de aceleração que melhora, através do uso do *K-means,* um terço das partículas geradas inicialmente para o PSO e então o PSO inicia as iterações com estas partículas melhoradas pelo *K-means.* Contudo, sabe-se que o *K-means* é uma heurística que sofre de algumas desvantagens como ficar preso em ótimos locais e depender da escolha dos centroides iniciais. Desta forma surgiu a ideia de utilizar-se uma meta-heurística para substituir a heurística *K-means* e superar suas desvantagens.

Com o uso do ILS apenas uma partícula é utilizada para ser melhorada. Esta partícula é selecionada na primeira iteração do PSO, após serem avaliadas as aptidões de todas as partículas iniciais, portanto é uma partícula que provavelmente vai estar numa região promissora do espaço de busca e consequentemente tem uma boa qualidade para servir como solução inicial do ILS.

Uma vez que o ILS produz resultados melhores que o *K-means*, e que, portanto, é capaz de devolver ao PSO uma partícula de melhor qualidade, optou-se por usar o ILS para hibridizar com o PSO.

A chamada ao ILS somente na primeira iteração do PSO se justifica pelo fato de que o ILS executado uma única vez já traz para o PSO uma solução próxima à ótima, conforme Tabela 10 de resultados do ILS. Esta solução vai ser usada pelo PSO nas próximas iterações para servir como *Gbest*, inicialmente. Todas as partículas vão convergir rapidamente em direção a esta partícula gerada pelo ILS e algumas podem superá-la em termos de qualidade, desta forma não se justifica a chamada ao ILS em mais iterações do PSO, o que somente aumentaria o custo computacional, bastando assim chamar o ILS somente uma vez com um número maior de iterações. Para fundamentar esta afirmação vários testes foram feitos, nos quais a chamada ao ILS foi feita apenas na primeira iteração ou em várias iterações do PSO. Os resultados estão apresentados na Tabela 11, a seguir.

BASE	**MEDIDA**	**Chamada nas 5 primeiras iterações do PSO**	**Chamada nas 10 primeiras iterações do PSO**	**Chamada nas 20 primeiras iterações do PSO**	**Chamada na primeira iteração do PSO**
VOWEL	MÉDIA	148968,79	148972,72	148971,44	**148967,39**
	DESVIO	5,94	11,84	7,40	**0,34**
	MELHOR	148967,24	148967.24	148967,24	**148967,24**

Tabela 11 - Resultados da hibridização com o ILS

A tabela acima mostra que os resultados obtidos com a chamada ao ILS em várias iterações confrontados com os obtidos somente em uma iteração,

são bem melhores, portanto, decidiu-se optar pela chamada ao ILS somente na primeira iteração, reduzindo assim drasticamente o custo computacional. Vale ressaltar que quando chama-se o ILS somente uma vez aumentou-se o número de iterações do ILS para 3 vezes o número de iterações realizadas no PSO.

4.10 Análise da convergência do EPSO-ILS

A estratégia de aceleração utilizada com o uso do ILS permite uma convergência rápida do PSO para uma solução ótima local ou global. Isto se deve ao fato de o algoritmo ILS, chamado de dentro do PSO, devolver uma partícula de excelente qualidade. Esta partícula passa então a funcionar como a partícula *Gbest* do PSO, pra onde todas as outras partículas do enxame passam a convergir. No Gráfico 3, abaixo, é apresentado a comparação da convergência para os algoritmos CPSO e EPSO-ILS, através do qual se pode verificar claramente a convergência do EPSO-ILS logo nas primeiras iterações.

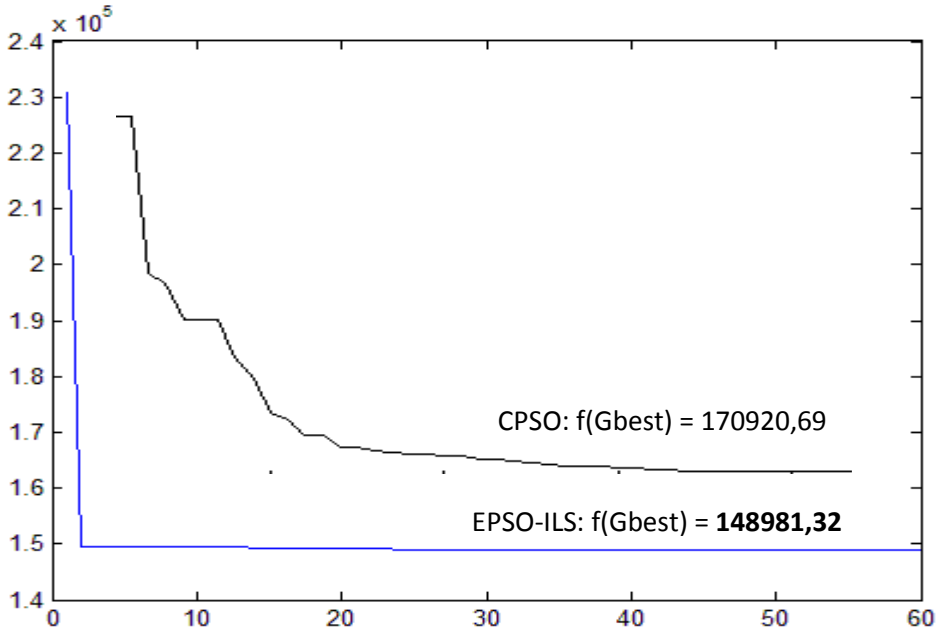

Gráfico 3 - Análise de convergência do EPSO-ILS

O gráfico acima trata do resultado da clusterização da base de dados *Vowel* constante na Tabela 1. O número de iterações utilizado foi de 60

iterações e o número de partículas *n* = 30, os outros parâmetros são os mencionados no Capítulo 3, Metodologia.

Para ilustrar melhor a convergência dos dois algoritmos, nos gráficos tridimensionais 4 e 5, abaixo, é mostrado o posicionamento das partículas após 60 iterações de ambos os algoritmos. Cada partícula, possui 6 centroides onde cada centroide é representado por um asterisco de cor diferente, ou seja, cada partícula possui 6 asteriscos de 6 cores diferentes como ilustrado no gráfico.

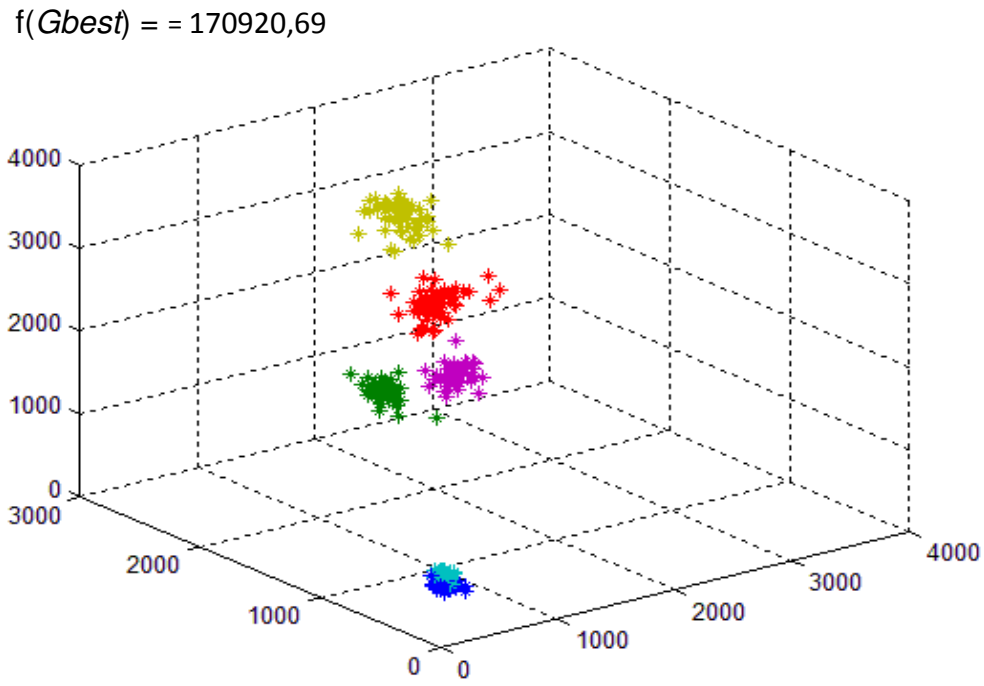

Gráfico 4 - Convergência das partículas do CPSO

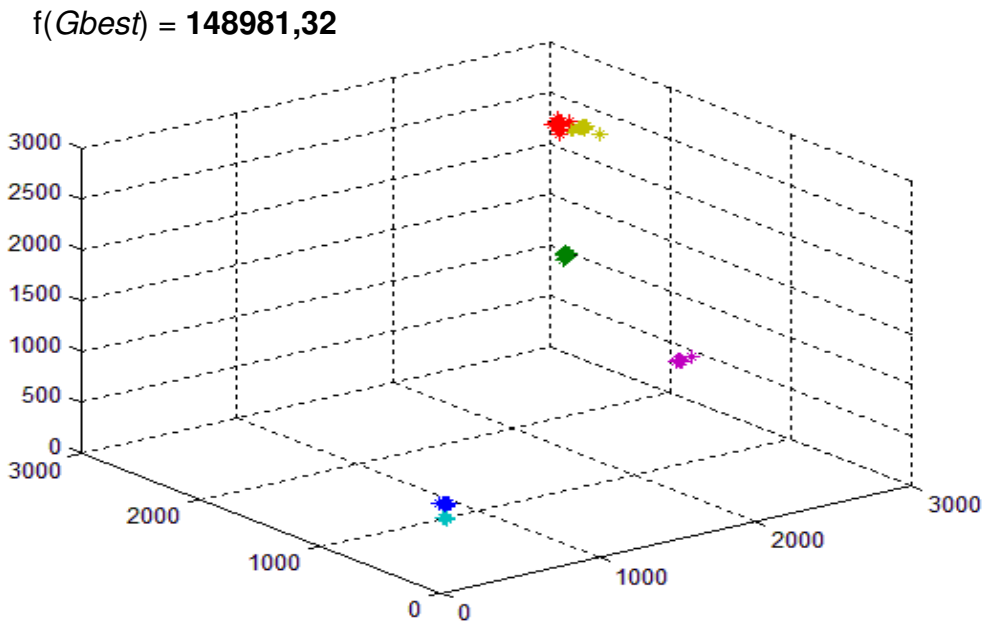

Gráfico 5 - Convergência das partículas do EPSO-ILS

No Gráfico 3, pode-se observar, claramente, o comportamento da convergência do EPSO-ILS para uma solução próximo à ótima logo na primeira iteração e ao final ao final das iterações é observado um valor muito inferior no resultado da clusterização.

Pode-se observar, ainda, claramente, nos Gráficos 4 e 5 que a coesão das partículas é maior no EPSO-ILS, indicando uma convergência maior para uma solução ótima local ou global. Ambos os gráficos demonstram o excelente desempenho do algoritmo proposto neste trabalho.

Capítulo 5

CONCLUSÃO

Neste trabalho foi apresentado um novo algoritmo composto pela hibridização de duas meta-heurísticas, PSO e ILS, para resolver o problema da Clusterização de Dados. Este capítulo descreve as conclusões relativas às pesquisas realizadas, citando suas contribuições bem como as limitações e dificuldades encontradas, além de sugestões para trabalhos futuros.

5.1 Conclusões

O problema de Clusterização de Dados é de natureza combinatória. O processo de se agrupar objetos similares em grupos ou *clusters* exige um número de combinações que crescem exponencialmente quando se aumentam o número de dimensões dos objetos e número de *clusters*. Devido a isto torna-se inviável a utilização de heurísticas determinísticas para resolver este problema, assim, para sua resolução foi proposto um algoritmo denominado EPSO-ILS que combina as meta-heurísticas PSO e ILS e a busca local *k-means.*

O algoritmo usa duas estratégias principais para melhorar o processo de clusterização: uma chamada à meta-heurística ILS e o uso de um número caótico para gerar o valor do índice do vetor de valores de peso inercial a ser aplicado na velocidade anterior das partículas a cada iteração.

Enquanto geralmente se usa o K-*means* como uma heurística de busca local pra se acelerar o processo de convergência, nesta versão apresentada procurou-se usar outra meta-heurística, a ILS, para acelerar este mesmo processo. O casamento de uma meta-heurística que tenta otimizar várias soluções ao mesmo tempo (populacional), com outra baseada na busca por entornos de uma única solução viável, proporcionou uma melhora significativa na convergência do PSO.

A ideia consiste em aproveitar o poder das duas meta-heurísticas para alcançar o mesmo objetivo, ou seja, diminuir a distância intracluster no

problema de clusterização. Para isto utilizou-se a partícula mais bem posicionada da primeira iteração do PSO para servir como solução inicial para o ILS. O ILS devolve ao PSO a solução melhorada e esta passa a servir como referência para as outras partículas do enxame, possibilitando assim que o PSO chegue a soluções melhores, visto que o modelo de PSO utilizado é o PSO *Gbest*, no qual a melhor partícula é seguida por todas as outras do enxame, podendo inclusive superá-la em termos de qualidade.

Para validar o algoritmo proposto neste trabalho, foram feitos testes utilizando-se cinco bases de dados de referência cujos resultados foram comparados com os resultados de oito algoritmos, utilizando-se sempre a mesma métrica, ou seja, minimização da distância *intracluster*, para pesquisar centroides ideais para a clusterização num espaço de busca euclidiano n-dimensional.

Os resultados computacionais mostram que a abordagem proposta é competitiva, visto que o algoritmo apresentou excelentes resultados quando testado nas várias bases de dados de referência da área, obtendo melhores valores para todas as instâncias avaliadas com um menor custo computacional.

Vale ressaltar que bons resultados foram obtidos com uma versão similar, mas ainda sem o uso do ILS. Estes resultados foram publicados em Coelho Filho et al. (2013b), fato que motivou a criação de uma segunda versão melhorada, com o uso do ILS, que está sendo apresentada neste trabalho e que também foi publicada em Coelho Filho et al. (2013a).

Este trabalho teve ainda como contribuição, além do algoritmo proposto, a criação de dois algoritmos independentes que foram denominados EPSO e ILS-KM. O EPSO é uma versão do PSO que se diferencia da original somente pelo uso de um componente caótico para selecionar o índice do vetor de pesos inerciais a serem aplicados à velocidade inercial a cada iteração. Tal algoritmo revelou uma excelente convergência quando analisados os resultados das Tabelas 8 e 9. O ILS-KM foi desenvolvido como estratégia para acelerar a convergência do PSO. Verificou-se ainda, com base nos resultados da Tabela 10, que este algoritmo apresenta uma boa performance, podendo ser utilizado sozinho para resolver problemas de clusterização de forma mais rápida que o PSO.

Ressalta-se ainda que na literatura pesquisada não foram encontradas versões do PSO hibridizadas com ILS, com abordagem parecida, usada para solucionar problemas de clusterização, assim como também não foram encontradas versões do PSO com o uso de um componente caótico na velocidade inercial.

5.2 Limitações e sugestões de modificações

Como limitações dos experimentos se pode citar as poucas bases de dados utilizadas. O uso de mais bases de dados, especialmente de bases maiores e com mais *clusters* e atributos possibilitaria uma melhor comparação entre os algoritmos testados. Bases com atributos do tipo categóricos também podem ser utilizadas. É reconhecido que o uso de bases somente com atributos escalares limita um pouco as comparações. Porém a principal limitação deste algoritmo é o fato dele não determinar automaticamente a quantidade de *clusters* ideal para a clusterização, ou seja, não faz a Clusterização Automática.

Embora se verifique o bom desempenho do EPSO-ILS, algumas modificações podem ser feitas no sentido de melhorar ainda mais sua performance. Uma das modificações sugeridas seria a atualização da velocidade de forma assíncrona para as partículas. Outra modificação que se pensou foi uma alteração na busca local e perturbação feita no ILS. Na perturbação, ao invés de se alterar a posição do centroide podia-se mover alguns objetos entre os centroides para verificar se a aptidão da partícula melhora, utilizando algum critério ou de forma aleatória. Outra modificação importante poderia ser o uso de uma espécie de "turbulência" nas partículas estagnadas. Esta "turbulência" poderia ser feita alterando apenas os centroides de pior qualidade de cada partícula estagnada a cada iteração.

A utilização da Clusterização Automática aliada à aplicação de todas estas modificações citadas, resultando numa espécie de PSO "turbinado", poderiam resultar num algoritmo ainda mais eficiente para ser utilizado em diversas aplicações práticas.

5.3 Trabalhos futuros

Sabe-se que atualmente uma das maiores causas de falecimento em todo o mundo entre as mulheres adultas é o câncer de mama, especialmente em países ocidentais, e que a análise de mamografias é uma tarefa muito difícil até para médicos bastante experientes. Para ajudar os médicos nesta tarefa de análise de imagens de mamografia, diversos estudos têm sido desenvolvidos atualmente na busca por algoritmos de segmentação de imagens que ajudem a encontrar evidências de aglomerações de células cancerígenas, especialmente quando elas estão no início, para que se possa fazer um diagnóstico precoce resultando em maiores chances de cura.

Pretende-se, portanto, como trabalhos futuros, realizar a implementação destas modificações para a Clusterização Automática de Dados e aplicá-la na segmentação de imagens de mamografia para a detecção de pequenos nódulos mamários conhecidos como microcalcificações.

Referências

Ahalt S. C., Krishnamurty A. K., Chen P, Melton D. E. *Competitive algorithms for vector quantization*. Neural Netw 3:277–291, 1990.

Alatas, B., Akin, E., Ozer, A. B. *Chaos embedded particle swarm optimization algorithms*. Chaos, Solitons & Fractals, 40, 1715–1734, 2009.

Anderson C., N. R. Franks. *Teams in Animal Societies*. *Behavioral Ecology*, 12(5):534-540, 2001.

Angeline P. J. *Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences*. In Proceeding of the Seventh Anual Conference of Evolutionary Programming, pages 601-610, 1998.

Berkhin, P. *Survey of Clustering Data Mining Techniques*. Accrue Software, 2002.

Berry, M. J. A, Linoff G. *Data Mining Techiques for Marketing, Sales and Customer Support*. John Wiley & Sons, Inc. 1997.

Brits R., Engelbrecht A. P and Van den Bergh F. *A Niching Particle Swarin Optimizer.* In Proceedings *of the Fourth Asia-Pacific* Conference 011 *Simulated Evolution and Learriing,* pages 692-696, 2002.

Carlisle J. A., Dozier G. *An Off-the-Shelf PSO*. In Proceedings of the workshop on Particle Swarm Optimization, pages 1-6, 2001.

Chuang, Li-Yeh ; Hsiao, Chih-Jen ; Yang, Cheng-Hong**.** *Chaotic particle swarm optimization for data clustering.* Expert Systems With Applications, 2011, Vol. 38(12), pp.14555-14563.

Clerc M., Kennedy J. *The Particle Swarm-Explosion, Stability and Convergence in a Multidimensional Complex Space*. IEEE Transactions on Evolutionary Computation, 6(1):58-73, 2002.

Coelho, L. d. S., Mariani, V. C. *A novel chaotic Particle Swarm Optimization approach using Hénon map and implicit filtering local search for economic load dispatch*. Chaos, Solitons & Fractals, 39, 510–518, 2005.

Coelho Filho, O. P. Martinhon C. A. Cabral, L. A. F *Uma nova abordagem híbrida do algoritmo de Otimização por Enxame de Partículas com Busca Local Iterada para o problema de Clusterização de Dados.* In 11th Brazilian Congress (CBIC) on Computational Intelligence, University of Pernambuco, BraziL, 2013. CD ROM.

Coelho Filho, O. P. Martinhon C. A. Cabral, L. A. F. *Uma nova abordagem melhorada do algoritmo de Otimização por Enxame de Partículas para o problema de Clusterização de Dados.* In XLV Simpósio Brasileiro de Pesquisa Operacional, Natal-RN, 2013.

Cohen, S. C. M. and Castro, L. N. (2006). *Data clustering with particle swarms*. Congress on Evolutionary Computation, Proceedings of IEEE Congress on Evolutionary Computation 2006:1792–1798.

Cole, R. M., 1998, *Clustering with Genetic Algorithms*, M. Sc., Department of Computer Science, University of Western Australia, Australia.

Engelbrecht, A. P., *Particle Swarm Optimization: Global Best or Local Best?.* In 1st BRICS Countries Congress (BRICS-CCI) and 11th Brazilian Congress (CBIC) on Computational Intelligence, Recife-PE, 2013.

Engelbrecht, A. P., *Particle Swarm Optimization: Iteration Strategies Revisted.* In 1st BRICS Countries Congress (BRICS-CCI) and 11th Brazilian Congress (CBIC) on Computational Intelligence, Recife-PE, 2013.

Engelbrecht, A. P., *Fundamentals of Computational Swarm Intelligence*. (John Wiley, 2005).

Fayyad U. M., G. Piatetsky-Shapiro, P. Smyth. *From data mining to knowledge discovery in database.* AI Magazine, p.37-54, 1996.

Goldberg D., *Genetic Algorithms in Search Optimization and Machine Learning,* Addison Wesley Publishing Company, Reading, Massa chutes, 1986.

Han, J., Kamber, M. *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufmann, 2001. 550 p.

Heppner F., Grenander U. *A Stochastic Non linear Model for Coordinated Bird Flocks.* In S. Krasner, editor, The Ubiquity of Chaos AAAS Publlications, 1990.

Holland John H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Oxford, England: U Michigan Press. (1975).

Jain, A. K., Murty, M. N., & Flynn, P. J. *Data clustering: A review*. ACM Computing Surveys, 31, 264–323, 1999.

Jain, A. K., Dubes, R. C. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series. Prentice-Hall, Inc., Upper Saddle River, NJ, 1988.

Janson C.H. *Experimental Evidence for Spatial Memory in Foraging Wild Capuch in Monkeys*, Cebus Apella. Animals Behaviour, 55:1229-1243, 1998.

Junliang L, Xinping X. *Multi-swarm and multi-best particle swarm optimization algorithm.* In: IEEE world congress on intelligent control and automation. pp 6281–6286, 2008.

Khan S., Ahmad A. *Cluster centre initialization algorithm for K-means clustering.* Pattern Recognit lett 25:1293–1302, 2004.

Kao, Y.-T., Zahara, E., Kao, I. W. *A hybridized approach to data clustering.* Expert Systems with Applications, 34, 1754–1762, 2008.

Kennedy, J. *Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance*. In Proceedings of the IEEE Congress on Evolutionary Computation, Volume 2, pages 1507-1512, July 2000.

Kennedy, J., Eberhart, R. *Particle Swarm Optimization.* In IEEE international joint c conference on neural network (Vol. 4, pp. 1942–1948), 1995.

Krishna K., Murty M. *Genetic K-means algorithm.* In: IEEE transactions on systems, man, and cybernetics, vol 29. pp 433–439, 1999.

Lourenço, H.R.; Martin O. and Stützle T. *"Iterated Local Search: Framework and Applications".* Handbook of Metaheuristics, 2nd. Edition. Kluwer Academic Publishers, International Series in Operations Research & Management Science 146: 363–397, 2010.

Maulik U, Bandyopadhyay S. *Genetic algorithm based data clustering techniques*. Pattern Recogn 33:1455–1465, 2002.

May, R. M. *Simple mathematical models with very complicated dynamics*. Nature, 261, 459–467, 1976.

Mendel, E.; Krohling, R. A. ; Campos, M.. *Swarm Algorithms with Chaotic Jumps Applied to Noisy Optimization Problems*. Information Sciences, v. 181, p. 4494-4514, 2011.

Merwe, D. W., Engelbrecht*, A. P. Data clustering using particle swarm optimization. Congress on Evolutionary computation, 2003. Proceedings of IEEE Congress on Evolutionary computation. pages 215-220.*

Metz J. *Interpretação de Clustering gerados por algoritmos de clustering hierárquico.* Dissertação de Mestrado, USP, São Carlos - SP, 2006.

Millonas, M. M. Swarms, phase transitions, and collective intelligence. In C.G. Langton (Ed.), Artificial Life III, pp. 417–445. Reading, MA: Addison-Wesley. 1994.

Mitra S., Acharya T. *Data Mining.* Wiley Publications, 2004.

Oliveira, A. C. M. e Lorena, L. A. N.; *Hybrid evolutionary algorithms and clustering search, Hybrid Evolutionary Systems - Studies in Computational Intelligence*. Springer SCI Series, 81-102, 2007.

Oliveira, R. R; Carvalho, C. L. *Algoritmos de agrupamento e suas aplicações*. Technical report, Universidade Federal de Goiás, 2008.

Partridge B. L..*The Structure and Function of Fish Schools*. Scientific American, 246:114-123, 1982.

Rana, S., Jasola, S., Kumar, R. *A review on particle swarm optimization algorithms and their applications to data clustering*. Artificial Intelligence Review 35, pp 211–222, 2011.

Reynolds C.W. Flocks, Herds and Schools: *A Distributed Behavioral Model*. Computer Graphics, 21(4):25-34, 1987.

Romão W. *Descoberta de Conhecimento Relevante em Banco de Dados sobre Ciência e Tecnologia*. Dissertação de Mestrado, UFSC – PR, 2002.

Schuster, H. G., & Just, W. *Deterministic chaos: An introduction*. Weinheim: Wiley-VCH Verlag GmbH, 2005.

Selim SZ, Alsultan K. *A simulated annealing algorithm for the clustering problem.* Pattern Recogn 24(10):1003–1008, 1991.

Sharpe F. A. *Social Foraging of Southeast Alaskan Humpback Whales*. PhD thesis, Simon Fraser University, Burnaby, British Columbia, 2000.

Shi, Y., and Eberhart, R. C. *A modified particle swarm optimizer.* In The 1998 IEEE international conference on evolutionary computation proceedings (pp. 69–73), 1998.

Sims D.W., V. A. Quayle. *Selective Foraging Behaviour of Basking Sharks on Zooplackton in a Small-scaleFront*. Nature, 393:460-464, 1998.

Steinley D., Brusco M. J. *Initialization K-means batch clustering: a critical evaluation of several techniques.* 2007.

Van den Bergh F. e Engelbrecht A. P. *Effects of Swarm Size on Cooperative Particle Swarm Optimisers*. In *Proceedings of the Genetic* and *Evolutionary Computation Conference,* pages 96-101, 2001.

Xu L, Krzyzak A, Oja E. *Rival penalized competitive learning for clustering analysis, RBF net and curve detection.* IEEE Trans Neural Netw 4:636–648, 1993.