

**DILSON LUCAS PEREIRA**

**ESTUDO DO PSO NA CLUSTERIZAÇÃO DE DADOS**

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS  
MINAS GERAIS - BRASIL  
2007

**DILSON LUCAS PEREIRA**

**ESTUDO DO PSO NA CLUSTERIZAÇÃO DE DADOS**

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:  
Otimização

Orientador:  
Ahmed Ali Abdalla Esmin

LAVRAS  
MINAS GERAIS - BRASIL  
2007

**DILSON LUCAS PEREIRA**

**ESTUDO DO PSO NA CLUSTERIZAÇÃO DE DADOS**

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 19/12/2007

---

Prof. Dr. Wilian Soares Lacerda

---

Prof. Dr. André Vital Saúde

---

Prof. Dr. Ahmed Ali Abdalla Esmin  
(Orientador)

LAVRAS  
MINAS GERAIS - BRASIL

*Dedico esse trabalho aos meus pais, Catarina e Dilson; à minha namorada, Juliana; e meu irmão, Armando.*

## **Agradecimentos**

Agradeço aos meus pais, Dilson e Catarina, por todo o apoio que me deram.

Agradeço à minha namorada, Juliana, pelas horas de descontração e carinho.

Agradeço a todos os professores por todo o conhecimento que me passaram.

Agradeço especialmente o professor Ahmed, pela orientação nesse trabalho.

# Estudo do PSO na Clusterização de Dados

## RESUMO

A quantidade de informação coletada atualmente é muito maior do que a capacidade humana de processá-la. Esse problema vem motivando o desenvolvimento de diversas técnicas computacionais capazes de extrair conhecimento valioso desses dados de modo eficiente. O problema de *Clustering*, ou Clusterização, consiste em encontrar grupos em certo conjunto de dados e é uma das principais tarefas de descoberta de conhecimento a partir de bancos de dados. O *Particle Swarm Optimization* (PSO) é uma técnica baseada no comportamento social bastante nova que vem sendo aplicada com sucesso a diversos tipos de problemas. Recentemente, foram propostas algumas abordagens que modificam o PSO para o problema de Clusterização, esses métodos são recentes e ainda podem ser trabalhados. Este trabalho tem como objetivo a análise do PSO na Clusterização de Dados, implementação e a melhoria de um dos métodos existentes.

**Palavras-Chave:** Clusterização, Particle Swarm Optimization, Otimização, Mineração de dados

## Study of the PSO on Data Clustering

## ABSTRACT

The amount of information collected nowadays is much greater than the human capacity of processing it. This problem has motivated the development of several computer methods able to extract valuable knowledge from these data in an efficient way. The Clustering problem consists on finding groups on a certain data set and is one of the main tasks on knowledge discovery from data bases. The Particle Swarm Optimization is a rather new method based on social behavior that has been successfully applied to many kinds of problems. Recently, some approaches were proposed modifying the PSO to the Clustering problem, these methods are recent and still can be worked. This work aims to analyze the PSO on data clustering, implement and improve one of the methods.

**Keywords:** Clustering, Particle Swarm Optimization, Optimization, Data Mining

# SUMÁRIO

1 – INTRODUÇÃO.....	1
1.1 – Contextualização e Motivação .....	1
1.2 – Objetivos.....	2
1.3 – Tipo de Pesquisa.....	2
1.4 – Procedimentos Metodológicos .....	3
2 – CLUSTERIZAÇÃO .....	4
2.1 – Introdução.....	4
2.2 – Clusterização .....	4
2.3 – Definição Formal.....	5
2.4 – Aplicações .....	5
2.5 – Medidas de Similaridade .....	6
2.6 – Métodos de Clusterização.....	7
2.7 – Conclusão .....	11
3 – COMPUTAÇÃO EVOLUTIVA .....	13
3.1 – Introdução.....	13
3.2 – Algoritmos Evolutivos.....	13
3.3 – Algoritmo Genético .....	15
3.4 – Estratégias de Evolução.....	17
3.5 – Programação Evolutiva.....	18
3.6 – Programação Genética.....	19
3.7 – Conclusão .....	21
4 – PARTICLE SWARM OPTIMIZATION (PSO) .....	22
4.1 – Introdução.....	22
4.2 – Origem.....	22
4.3 – Particle Swarm Optimization .....	23
4.4 – Classificação.....	24
4.5 – Conclusão .....	25
5 – MÉTODOS DE CLUSTERIZAÇÃO USANDO O PSO.....	26
5.1 – Introdução.....	26
5.2 – Clusterização com o PSO .....	26
5.3 – O método proposto por Merwe e Engelbrecht.....	28
5.3.1 – Introdução.....	28
5.3.2 – O Algoritmo.....	28
5.3.3 – A Função de Avaliação e modificações propostas .....	29
5.4 – Conclusão .....	30
6 – RESULTADOS .....	31
6.1 – A Implementação.....	31
6.2 – Resultados.....	34
6.3 – Conclusão .....	39
7 – CONCLUSÃO E TRABALHOS FUTUROS .....	40
8 – REFERÊNCIAS BIBLIOGRÁFICAS .....	41

## Lista de Figuras

Figura 3-1 – Exemplo de representação em árvore .....	20
Figura 6-1 – Parâmetros do Algoritmo.....	31
Figura 6-2 – Visualização dos grupos criados pelo algoritmo .....	32
Figura 6-3 – <i>Fitness</i> da melhor partícula ao longo das iterações .....	32
Figura 6-4 – Resultados do Algoritmo .....	33
Figura 6-5 – Interface do programa.....	33
Figura 6-6 – Agrupamentos encontrados pelas 3 funções de avaliação para o <i>benchmark</i> <i>Iris</i> .....	37
Figura 6-7 – Agrupamentos encontrados pelas 3 funções de avaliação para o <i>benchmark</i> <i>Wine</i> .....	37
Figura 6-8 – Agrupamentos encontrados pelas 3 funções de avaliação para o <i>benchmark</i> <i>Glass</i> .....	38
Figura 6-9 – Convergência do Algoritmo utilizando as 3 Funções.....	38



## Lista de Tabelas

Tabela 6-1 – Características dos <i>benchmarks</i> .....	34
Tabela 6-2 – Comparação entre o acerto médio das execuções utilizando as funções de <i>fitness</i> F1, F2 e F3 .....	36
Tabela 6-3 – Comparação entre a distância <i>intra-cluster</i> das execuções utilizando as funções de <i>fitness</i> F1, F2 e F3 .....	36
Tabela 6-4 – Comparação entre a distância <i>inter-cluster</i> das execuções utilizando as funções de <i>fitness</i> F1, F2 e F3 .....	36

# 1 – INTRODUÇÃO

## 1.1 – Contextualização e Motivação

A quantidade de informação disponível e coletada hoje em dia é maior do que a capacidade humana de analisar e extrair conhecimento a partir dela. As grandes corporações possuem uma quantidade imensa de dados inutilizados armazenados em seus bancos de dados, o conhecimento escondido nesses dados poderia ser muito útil aos objetivos da empresa. O aumento dramático na quantidade de dados necessitando de análise levou ao desenvolvimento de novas técnicas capazes de extrair conhecimento de modo automático e eficiente (COHEN e CASTRO 2006).

*Data Mining* é o nome usado para descrever o esforço computacional para se processar grandes quantidades de dados armazenadas em bancos de dados, com a finalidade de se obter conhecimento valioso e em alto nível (SOUZA et al., 2003).

Segundo Cohen e Castro (2006), *Clustering*, também conhecido como Agrupamento ou Clusterização, é uma das principais tarefas de descoberta de conhecimento a partir de bancos de dados, e consiste em encontrar grupos em um certo conjunto de dados. Cada grupo contém objetos similares entre si e diferentes daqueles em outros grupos. Para Merwe e Engelbrecht (2003), *Data Clustering* é o processo de reunir em grupos, vetores de dados multidimensionais similares.

Desde o início da infância, aprende-se a distinguir entre gatos e cachorros, ou plantas e animais, melhorando continuamente esquemas de classificação subconscientes (HAN e KAMBER, 2001). *Data Clustering* tem aplicações em diversas áreas como Biologia, Pesquisa de Mercado, Processamento de Imagens, Reconhecimento de Padrões, Geografia, e muitas outras.

O *Particle Swarm Optimization* (PSO) é um método de otimização inicialmente proposto por Kennedy e Eberhart (1995) baseado no comportamento social de populações. O método foi descoberto através da simulação de um modelo social simplificado, onde a intenção original era simular graficamente a graciosa e imprevisível coreografia de um bando de pássaros. O método de otimização consiste basicamente em várias partículas, ou seja, soluções potenciais, que se movimentam no espaço de busca com base em sua melhor posição

e na melhor posição em sua vizinhança. O PSO pode ser usado para resolver uma vasta faixa de diferentes problemas de otimização, incluindo a maioria dos problemas que podem ser resolvidos usando o Algoritmo Genético (BERGH, 2001).

Para Kennedy e Eberhart (1995), o algoritmo está obviamente relacionado à Computação Evolutiva e posicionado em algum lugar entre os Algoritmos Genéticos e a Programação Evolutiva.

O algoritmo é bastante novo e vem sendo modificado e aplicado com sucesso a diversos problemas de otimização. Recentemente surgiram na literatura algumas abordagens do PSO na Clusterização de Dados com resultados bastante satisfatórios. Sua aplicação na Clusterização de Dados é recente, assim suas abordagens ainda podem ser trabalhadas, contribuindo para melhores resultados.

## 1.2 – Objetivos

Este trabalho tem como objetivo a análise o PSO na Clusterização de Dados, dando uma descrição das principais técnicas existentes para a solução do problema. Além disso, uma das técnicas é analisada e algumas melhorias são propostas.

Para que seja possível uma análise e comparação da técnica implementada e das melhorias, são feitos vários testes e experimentos utilizando vários *benchmarks* tradicionais da área como *Iris*, *Glass* e *Wine*. Espera-se que esse trabalho possa contribuir a consolidação do PSO como uma importante ferramenta na otimização e Clusterização de Dados.

## 1.3 – Tipo de Pesquisa

Trata-se de uma pesquisa Básica ou Fundamental, pois visa à geração de novos conhecimentos, e para tal são estudadas e analisadas as técnicas de Clusterização existentes como meio de inspiração, para que se possa melhorar os métodos baseados no PSO, e após isso, a divulgação dos conhecimentos e resultados obtidos. É também uma pesquisa Exploratória, pois visa o aprimoramento de idéias já existentes e descoberta de novas informações/conhecimentos. Por fim, pode ser classificada também como uma pesquisa Operacional, pois utiliza métodos de otimização para que se possa encontrar um meio mais adequado de se obter melhores resultados, ou seja, maximização das distâncias *inter-cluster* e minimização das distâncias *intra-cluster*.

## 1.4 – Procedimentos Metodológicos

Os métodos de Clusterização baseados no PSO foram estudados e um dos métodos foi implementado entre julho e outubro de 2007 utilizando a linguagem de programação C++ e compilador Microsoft Visual C++ 6. O método foi profundamente estudado e analisado juntamente com as principais técnicas de Clusterização existentes para que fosse possível encontrar deficiências e melhorias pudessem ser sugeridas, experimentadas e testadas e os resultados pudessem ser gerados, analisados e comparados utilizando os principais *benchmarks* da área.

## 2 – CLUSTERIZAÇÃO

### 2.1 – Introdução

Nesse capítulo, será abordado o problema de *Clustering*, ou Clusterização, no que consiste o problema, quais são suas aplicações, como avaliar a semelhança entre os dados e quais as principais técnicas para a solução do problema.

### 2.2 – Clusterização

De um modo mais grosseiro, clusterizar corresponde ao processo de agrupar os elementos de um conjunto de modo que os elementos de um determinado grupo, ou *cluster*, sejam mais parecidos entre si do que com elementos dos outros grupos.

Cole (1998) define Clusterização como um procedimento exploratório que busca uma estrutura “natural” dentro de um conjunto de dados. Os dados são arranjados com base na similaridade entre eles, sem nenhuma suposição sobre a estrutura dos dados.

A Clusterização é uma tarefa prévia à Classificação, pois antes de classificar, é preciso ter as classes às quais os dados devem ser associados.

Após o agrupamento, é possível analisar os elementos de cada *cluster*, identificando assim as características semelhantes aos elementos do *cluster*, os *clusters* podem então ser nomeados, dando origem à classes. Com a existência dessas classes somos capazes de classificar um novo elemento pertencente ao universo considerado ao recebê-lo (CARLANTONIO, 2001).

Clusterização de Dados é uma disciplina científica recente sob um vigoroso desenvolvimento. Existe uma grande quantidade de artigos espalhados em periódicos, a maioria na área de *Data Mining*, Estatística, Aprendizagem de Máquina, Bancos de Dados Espaciais, Biologia, Marketing, etc. Devido a grande quantidade de dados coletados e armazenados em bancos de dados, a Análise de Cluster recentemente se tornou um tópico bastante ativo dentro da pesquisa em Mineração de Dados (HAN e KAMBER, 2001).

Ankerst et al. (1999) identificam três motivos que fazem com que a efetividade dos algoritmos de Clusterização seja um problema. Primeiro, quase todos os algoritmos de Clusterização precisam de valores de entrada que são difíceis de se determinar, especialmente para conjunto de dados do mundo real que contém muitos atributos. Segundo, os algoritmos são muito sensíveis a esses parâmetros de entrada, e

frequentemente produzem particionamentos muito diferentes dos dados, mesmo para pequenos ajustes dos parâmetros de entrada. Terceiro, conjuntos de dados multidimensionais do mundo real têm uma distribuição muito complexa que pode não ser descoberta usando apenas um único ajuste de parâmetro global.

## 2.3 – Definição Formal

Cole (1998), define o problema de Clusterização formalmente da seguinte maneira:

O conjunto de  $n$  objetos  $X = \{X_1, X_2, X_3, \dots, X_n\}$  deve ser clusterizado. Cada  $X_i \in \mathfrak{R}^p$  é um vetor de  $p$  medidas reais que descrevem o objeto. Esses objetos serão agrupados em grupos disjuntos  $C = \{C_1, C_2, C_3, \dots, C_k\}$  ( $C$  é conhecido como *Clustering*), onde  $k$  é número de *clusters*,  $C_1 \cup C_2 \cup \dots \cup C_k = X$ ,  $C_i \neq \emptyset$ , e  $C_i \cap C_j = \emptyset$  para  $i \neq j$ . Os objetos dentro de cada grupo deveriam ser mais similares entre si do que com objetos de outros grupos, e o valor de  $k$  pode ser desconhecido. Se  $k$  é conhecido o problema é chamado de problema de  $k$ -Clusterização.

## 2.4 – Aplicações

A Clusterização é útil nas mais diversas áreas e com as mais diversas finalidades.

Por Exemplo:

Cole (1998) cita vários trabalhos em que a Clusterização já foi usada. Na Psiquiatria, a Clusterização foi usada para o desenvolvimento de uma classificação para a depressão. Em Pesquisa de Mercado, para a identificação de conjuntos homogêneos de mercados de teste. Na Arqueologia, Clusterização foi aplicada ao problema de classificar machados de mão ingleses. No Reconhecimento de Padrões, para segmentação de imagens. Na Medicina, como método de aquisição de conhecimento para sistemas de diagnóstico.

Ng e Han (1994), dizem que a Clusterização tem sido aplicada nos últimos trinta anos em áreas como Medicina, para classificação de doenças; na Química, para agrupamento de compostos; Estudos Sociais, para classificação de estatísticas.

Ankerst et al. (1999), citam como exemplo a criação de mapas temáticos em sistemas de informações geográficas; Clusterização de um banco de dados de web-logs para descobrir grupos de padrão de acesso similares que podem corresponder a diferentes perfis de usuários.

Han e Kamber (2001), dizem que a Clusterização pode ser usada na área de negócios para ajudar empresários a descobrir grupos distintos em suas bases de clientes e caracterizar grupos de clientes baseado em padrões de compra. Na Biologia, pode ser usado para derivar taxonomias de plantas e animais, categorizar genes com funcionalidades semelhantes e analisar melhor estruturas inerentes às populações. Na Geografia, para a identificação de áreas semelhantes em bancos de dados de observação da Terra e identificação de grupos de casas em uma cidade de acordo com tipo, valor e localização geográfica. Pode ajudar também na classificação de documentos na internet. Além disso, também ressaltam que a Clusterização pode servir como uma etapa de pré-processamento para outros algoritmos, como os de classificação e caracterização, operando nos clusters identificados.

## 2.5 – Medidas de Similaridade

Para clusterizar objetos de acordo com sua similaridade, é preciso que se defina a proximidade entre dois objetos ou como seus valores se comparam. Uma pequena distância deve identificar alta similaridade, portanto a medida da distância pode ser usada para quantificar a dissimilaridade (COLE, 1999).

Cole (1999) identifica a distância Euclidiana como a mais comum entre as medidas utilizadas,

$$d(X_i, X_j) = \sqrt{\sum_{t=1}^p (X_{it} - X_{jt})^2},$$

a distância Euclidiana mede a distância entre os pontos através de uma linha reta.

Outro método bastante utilizado é a distância *Manhattan* (ou *City Block*) definida por

$$d(X_i, X_j) = \sum_{t=1}^p |X_{it} - X_{jt}|.$$

Ambas os métodos são generalizações da distância *Minkowski*,

$$d(X_i, X_j) = \left( \sum_{t=1}^p (X_{it} - X_{jt})^n \right)^{1/n}.$$

Han e Kamber (2001) ainda ressaltam que nem todos os atributos devem ser incluídos na Clusterização. Um atributo sem significância é pior do que inútil, pois, pode

prejudicar o resultado final. Além disso, podem ser atribuídos pesos às variáveis de acordo com sua importância. A distância Euclidiana ponderada pode ser calculada como,

$$d(X_i, X_j) = \sqrt{\sum_{t=1}^p w_t (X_{it} - X_{jt})^2},$$

onde  $w_t$  é um peso atribuído à variável  $t$ . O mesmo pode ser feito com *Manhattan* e *Minkowski*.

É importante salientar que a unidade de medida pode afetar no *Clustering*. Por exemplo, a alteração de metros para quilômetros ou de gramas para quilos pode afetar o *Clustering*. Para evitar esse problema com as unidades de medida, os dados deveriam ser padronizados de modo que as variáveis tivessem pesos iguais.

Uma das soluções é transformar as medidas em variáveis sem escala, o que pode ser feito da seguinte maneira:

1. Calcular o desvio absoluto médio,  $s_f$ :

$$s_f = \frac{1}{n} (|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|),$$

onde  $x_{1f}, x_{2f}, \dots, x_{nf}$  são medidas do atributo  $f$  e  $m_f$  é o valor médio de  $f$ ,

$$m_f = \frac{1}{n} (x_{1f} + x_{2f} + \dots + x_{nf}).$$

2. Calcular a medida padronizada, conhecida como *z-score*, da seguinte maneira:

$$z_{if} = \frac{x_{if} - m_f}{s_f}.$$

A padronização pode ou não ser realizada, dependendo da aplicação.

## 2.6 – Métodos de Clusterização

De acordo com Han e Kamber (2001), os principais métodos de Clusterização podem ser classificados nas seguintes categorias:

### 1. Métodos de Particionamento:

Dado um conjunto de dados de tamanho  $n$  e o número de *clusters*  $k$ , um algoritmo de Particionamento constrói  $k$  partições de dados, onde  $k \leq n$ . O método consiste em criar um particionamento inicial que vai sendo refinado através de uma técnica de recolocação iterativa que tenta melhorar o particionamento movendo



dados de uma partição para outra. O critério geral de um bom particionamento é que dados do mesmo cluster estejam próximos e dados de clusters diferentes estejam longe.

Métodos por particionamento têm sido bastante focados devido à boa qualidade dos clusters produzidos (CARLANTONIO, 2001).

Percorrer todas as partições possíveis buscando pelo ótimo local é inviável mesmo para bancos de dados de tamanho razoável, por isso métodos heurísticos são mais indicados. Os dois métodos mais populares são os algoritmos *k-means* e o algoritmo *k-medoids*.

Merwe e Engelbrecht (2003) explicam que o algoritmo *k-means* agrupa dados em um número de *clusters* pré-definido, e que os *clusters* são representados por um vetor centróide, que representa o ponto central daquele *cluster*. O algoritmo *k-means* é eficiente em grandes conjuntos de dados, porém não consegue trabalhar com dados categóricos, apenas dados numéricos (SAN et al. 2004).

O método *k-medoids* ao invés de centróides usa objetos (dados), do próprio conjunto de dados para representar os clusters. Dado um conjunto  $X$  de dados e um número  $k$  de *clusters*, o algoritmo busca por  $k$  *medoids* que possam minimizar a distância média dos objetos aos seus *medoids* mais próximos (HUANG, 1998).

Os métodos de particionamento não trabalham bem em *clusters* de formas complexas e tamanhos diferentes (HAN e KAMBER, 2001; CARLANTONIO, 2001)

## **2. Métodos Hierárquicos:**

Karapys et al. (1999) explicam que algoritmos de Clusterização Hierárquicos produzem uma seqüência aninhada de *clusters*, com um *cluster* que inclui todos os outros no topo e pequenos *clusters* de apenas um objeto no piso. Criando uma decomposição hierárquica do conjunto de dados. Essa decomposição hierárquica pode ser representada através de uma estrutura em árvore chamada Dendograma (CARLANTONIO, 2001).

Baseado em como a estrutura é criada, os métodos de Clusterização Hierárquicos podem ser classificados como Aglomerativos ou Divisivos (HAN e KAMBER, 2001).

Os métodos Aglomerativos, também chamados de *bottom-up*, iniciam com cada objeto do conjunto de dados representando um *cluster*. A cada passo o algoritmo funde dois *clusters* que são mais similares. O algoritmo prossegue até que todos os *clusters* tenham sido fundidos em um *cluster* final, o nível mais alto da hierarquia, ou até que um número total *clusters* desejado tenha sido alcançado ou a distância entre os dois clusters mais próximos esteja acima de um limite estipulado (HAN e KAMBER, 2001; KARAPYS et al., 1999).

Os *clusters* são fundidos baseados na similaridade, distância entre eles, que geralmente é calculada através da distância entre os centros dos clusters, ou através da distância do par de dados mais próximo, ou ainda através da média da distância entre todos os pares de dados dos dois clusters.

Os métodos Divisivos, também chamados de *top-down*, partem do nível mais alto da hierarquia, ou seja, um único *cluster* que contém todos os objetos e vai dividindo esse *cluster* em *clusters* menores até que cada cluster contenha apenas um objeto ou algum outro critério seja alcançado (HAN e KAMBER, 2001; CARLANTONIO, 1999).

Han e Kamber (2001) afirmam ainda, que pode ser vantajoso combinar métodos hierárquicos com recolocação iterativa. Primeiro o método hierárquico é usado e depois refinado utilizando a recolocação iterativa. Como algoritmos desse tipo citam BIRCH e CURE.

### **3. Métodos Baseados em Densidade**

Como mencionado por Carlantonio (2001) e também por Han e Kamber (2001), a maioria dos métodos de Clusterização se baseia na distância entre os objetos, esses métodos conseguem encontrar apenas clusters de forma esférica e têm dificuldades em encontrar clusters de forma arbitrária.

De acordo com Ankerst et al. (1999), as abordagens baseadas em densidade aplicam um critério de *cluster* local e são muito populares em *Data Mining*. *Clusters* são considerados regiões do espaço de dados nas quais a distribuição de objetos é densa, e são separados por regiões de baixa densidade de objetos (conhecidas como *noise*). Essas regiões podem ter forma arbitrária e os dados em uma região podem estar distribuídos arbitrariamente.

Ankerst et al. (1999), Han e Kamber (2001) citam o método DBSCAN onde a idéia básica é que para cada ponto em cluster sua vizinhança de raio ( $\epsilon$ ) deve conter um número mínimo de pontos (*MinPts*), onde  $\epsilon$  e *MinPts* são parâmetros de entrada.

#### 4. Métodos Baseados em *Grid*

Um método baseado em *Grid* divide o espaço de dados em uma estrutura de dados em *grid* (grade) multidimensional. Após isso todas as operações de Clusterização são realizadas sobre a estrutura em *grid*. A principal vantagem dessa abordagem é a velocidade que é independente da quantidade de dados e dependente apenas do número de células em cada dimensão do espaço dividido.

O método STING, S**T**atistical **I**nformation **G**rid-based method, é um método criado por Wang et al. (1997) com o objetivo de solucionar problemas relacionados a *clustering* e consultas orientadas a regiões. Trata-se de uma abordagem multi-resolução baseada em *grid* onde o espaço é dividido em células. Essas células formam uma estrutura hierárquica onde cada célula, em um alto nível, é particionada de modo a formar um número de outras células no próximo nível mais baixo. Dados estatísticos importantes associados com os valores dos atributos em cada célula como média, máximo, mínimo, desvio padrão, etc, são pré-computados e armazenados antes de uma consulta ser realizada no sistema (HAN e KAMBER, 2001).

A idéia é capturar informações estatísticas associadas com as células de maneira que classes inteiras de consultas e problemas de *clustering* possam ser respondidos sem recorrer a objetos individuais.

#### 5. Métodos Baseados em Modelos

Para Han e Kamber (2001), um Método Baseado em Modelos hipotetiza um modelo para cada um dos clusters e encontra o melhor ajuste dos dados para aquele modelo. Um algoritmo baseado em modelos pode encontrar *clusters* construindo uma função de densidade que reflita a distribuição espacial dos dados. Também leva a um modo de determinar automaticamente o número de *clusters*, levando ruído (*noise*) em conta, rendendo assim métodos de Clusterização robustos.

Segundo Yeung et al. (2001), algoritmos de Clusterização Baseados em Modelos assumem que os dados são gerados através de uma mistura finita de distribuições de probabilidade subjacentes.

Han e Kamber (2001) afirmam que os Métodos Baseados em Modelos possuem duas abordagens principais: abordagem Estatística e abordagem Rede Neural.

- Abordagem Estatística:

*Clustering* em aprendizagem de máquina é frequentemente chamado de Aprendizado Não Supervisionado ou Formação de Conceitos (*clusters*). A maioria dos trabalhos em Formação de Conceitos adota uma abordagem baseada em probabilidade que usa medidas de probabilidade para Clusterização e representa os *clusters* com descrições de probabilidade.

Carlantonio (2001) identifica o método COBWEB como sendo um método de clustering desse tipo bastante popular.

- Abordagem Rede Neural:

Carlantonio (2001) afirma que métodos de *Clustering* utilizando a abordagem por Rede Neural geralmente representam *clusters* como exemplares, que servem de protótipo para o *cluster* e que novos objetos podem ser distribuídos aos *clusters* baseado em sua similaridade (distância) com o exemplar do *cluster*.

## 2.7 – Conclusão

Nesse capítulo, foi apresentado o problema de Clusterização, ou agrupamento, que consiste em encontrar grupos em um conjunto de dados. Na seção 2.2 o problema foi discutido, juntamente com sua importância e dificuldades dos algoritmos de Clusterização, como por exemplo encontrar o número correto de *clusters*. Na seção 2.3 foi apresentada a definição formal do problema. Foi visto, na seção 2.4, que o problema de Clusterização tem aplicações em diversas áreas como Medicina, Biologia, Reconhecimento de Padrões, geografia, etc. Na seção 2.5, foi demonstrado como medir a similaridade entre dois dados e como normalizar os dados para que a unidade de medida não tenha efeito durante o agrupamento. Finalmente, na seção 2.6, foi visto que os métodos de Clusterização podem ser divididos em: Métodos de Particionamento, Hierárquicos, Baseados em Densidade,

Baseados em Grid e Baseados em Modelos. Alguns métodos se encaixam em mais de um dos tipos.

## 3 – COMPUTAÇÃO EVOLUTIVA

### 3.1 – Introdução

Computação Evolutiva é uma área de pesquisa dentro da computação que se inspira no processo de evolução natural.

Bergh (2001) afirma que a Computação Evolutiva define vários métodos projetados para simular a evolução, esses métodos são todos baseados em população e usam de uma combinação de variação aleatória e seleção para resolver problemas.

Eiben e Smith (2003) explicam que não é surpresa que cientistas da computação se inspirem no processo de evolução natural, já que o poder da evolução natural é evidente nas diversas espécies que existem em nosso mundo.

De modo simplificado, evolução natural pode ser descrita da seguinte maneira: Um ambiente é ocupado por uma população de indivíduos que lutam por sobrevivência e reprodução. A aptidão, ou *fitness*, desses indivíduos está relacionada com sua capacidade de atingir seus objetivos, quanto melhor o indivíduo estiver adaptado ao ambiente, mais chances terá de sobreviver e se multiplicar. No contexto de um processo estocástico de solução de problemas por tentativa e erro, temos uma coleção de soluções candidatas. Seu *fitness* (ou seja, o quão bem resolve o problema) determina suas chances de serem mantidas e usadas como base para a construção de novas soluções candidatas (EIBEN e SMITH, 2003).

O princípio fundamental por trás da evolução é o da otimização, com o objetivo de sobrevivência das espécies. Isso não significa porém, que os métodos Evolutivos só possam ser aplicados a problemas de otimização. Várias categorias foram identificadas como prosperamente tratáveis pelos métodos evolutivos: Planejamento, Projeto, Simulação e Identificação, Controle, Classificação (BERGH, 2001).

### 3.2 – Algoritmos Evolutivos

De acordo com Eiben e Smith (2003), existem diferentes tipos de Algoritmos Evolutivos, porém a idéia básica por trás deles é a mesma: Dada uma população de indivíduos, a pressão do ambiente provoca uma seleção natural (sobrevivência do mais apto), o que causa um aumento no *fitness*, ou aptidão, da população. O *fitness* é calculado como uma função de qualidade que desejamos maximizar. Os melhores indivíduos são

escolhidos para dar origem a próxima geração através da recombinação e/ou mutação. O processo é repetido até que um candidato com a qualidade desejada seja encontrado ou um limite computacional pré-estabelecido seja alcançado. Nesse processo duas forças formam a base dos sistemas evolutivos:

- Operadores de Variação (recombinação e mutação) criam a diversidade necessária.
- Seleção age como uma força em direção a qualidade.

A aplicação combinada de variação e seleção geralmente leva ao melhoramento do *fitness* nas populações consecutivas.

Muitos componentes desse processo são estocásticos, durante a seleção, por exemplo, indivíduos mais aptos têm uma maior chance de serem selecionados do que indivíduos menos aptos, mas mesmo indivíduos mais fracos possuem chances de se tornar pais ou sobreviverem.

Abaixo podemos encontrar um pseudo-código para os algoritmos evolutivos (EIBEN e SMITH, 2003):

**Inicialize** a população com soluções candidatas aleatórias

**Avalie** cada candidato

**Repita** até que a condição de término seja satisfeita

**Selecione** os pais

**Recombine** os pais

**Mute** a descendência gerada

**Avalie** os novos candidatos

**Selecione** os indivíduos para a próxima geração

Eiben e Smith (2003) explicam que possíveis soluções no contexto original do problema são chamados de fenótipo, enquanto a sua codificação, sua representação dentro dos Algoritmos Genéticos é chamada de genótipo. A definição da representação dos indivíduos, a estrutura de dados dos indivíduos, é o primeiro passo na definição de um Algoritmo Evolutivo.

O papel da Função de Avaliação, também conhecida com Função *Fitness*, é avaliar a qualidade dos indivíduos. Tipicamente essa função mede a qualidade dos genótipos baseada em seu fenótipo, ou seja, mede a qualidade da representação do indivíduo baseado em quão boa é a solução que ele representa.

Os Operadores de Variação, ou seja, Operadores de Mutação e Recombinação são os responsáveis pela criação de novas soluções a partir das existentes. O Operador de Mutação é um operador que é aplicado a somente um indivíduo e seu papel é o de introduzir diversidade à população, aumentando a área efetiva do espaço de busca que o algoritmo considera (BERGH, 2001). O Operador de Recombinação, ou *Crossover*, junta a informação contida em dois genótipos pais em um ou dois genótipos filhos. Através da Recombinação de duas soluções com boas características podemos criar uma nova solução que combine as boas características de suas soluções pai.

A Seleção é responsável por elevar a qualidade da população. A Seleção dos Pais permite que melhores indivíduos se tornem pais da próxima geração, porém, indivíduos de menor qualidade também possuem chance de se tornarem pais. A Seleção dos Sobreviventes, também conhecida como Substituição, é responsável por selecionar quais os indivíduos estarão na próxima geração. Ao contrário da Seleção dos Pais, que é estocástica, a Seleção dos Sobreviventes geralmente é determinística, por exemplo, selecionando os indivíduos de maior qualidade entre os pais e filhos ou substituindo os pais pela nova geração.

O tipo de recombinação e mutação, tipo de seleção de pais e sobreviventes, representação dos indivíduos determinam o tipo de Algoritmo Evolutivo: Algoritmo Genético, Estratégias de Evolução, Programação Evolutiva e Programação Genética.

### 3.3 – Algoritmo Genético

O Algoritmo Genético é o mais conhecido dentre os Algoritmos Evolutivos (EIBEN e SMITH, 2003). Mitchell e Forrest (1993) afirmam ainda que o Algoritmo Genético tem sido usado tanto como ferramenta para a solução de problemas práticos quanto como modelo científico de processos evolutivos.

O Algoritmo Genético foi inicialmente descrito por John Holland nos anos 60 e posteriormente, nos anos 60 e 70, desenvolvido juntamente com seus alunos e colegas da Universidade de Michigan (MITCHEL e FORREST, 1993).

Bäck (1991) afirma que uma definição geral do Algoritmo Genético não existe. Pode ser esboçado aquilo que pode ser considerado o Algoritmo Genético Clássico ou Algoritmo Genético Canônico: Os indivíduos são representados por uma *string* binária, a seleção é proporcional ao *fitness*, a probabilidade de mutação é baixa e o algoritmo



ênfatiza uma recombinação geneticamente inspirada como meio de gerar novas soluções candidatas (BERGH, 2001; EIBEN e SMITH, 2003).

Embora o Algoritmo Genético Clássico represente os indivíduos através de *strings* de *bits*, elas podem não ser a que se encaixa melhor ao problema. Dependendo do problema podem também ser usadas representações com inteiros, reais, permutações, árvores e outras. Bergh (2001) aponta que a desvantagem das representações não binárias é que elas requerem operadores de recombinação e mutação específicos à representação.

Os operadores de recombinação aplicados às *strings* binárias mais populares são os operadores: “*Crossover* de um Ponto” e “*Crossover* de  $n$  Pontos”. Dois pais são selecionados para a recombinação e segmentos de suas *strings* de *bits* são trocados para que sejam formados dois novos filhos. No *Crossover* de um Ponto, um ponto das *strings* é escolhido aleatoriamente e os *bits* após esse ponto são trocados. No *Crossover* de  $n$  pontos,  $n$  pontos são escolhidos e os filhos são formados pegando segmentos alternativos dos pais.

O operador de mutação mais utilizado em *strings* binárias é o *Bit-Flip* onde cada gene (*bit*) é considerado separadamente e é invertido com uma pequena probabilidade (BERGH, 2001; EIBEN e SMITH, 2003).

A Seleção dos pais, geralmente é feita baseada no *fitness* do indivíduo, cada indivíduo tem probabilidade  $f_i / \sum_{j=1}^{\mu} f_j$  de ser escolhido, onde  $f_i$  é o *fitness* do indivíduo  $i$  e  $\mu$  é o tamanho da população. Eiben e Smith (2003) identificam que usando esse mecanismo de seleção, indivíduos muito bons tomam conta da população rapidamente, problema conhecido como Convergência Prematura. Além disso, quando os valores de *fitness* são muito próximos, quase não existe uma pressão de seleção, pois as probabilidades são muito parecidas e os indivíduos são escolhidos quase aleatoriamente. O método baseado em *rank* busca eliminar as desvantagens do método baseado no *fitness*. Nesse método a população é ordenada baseada no *fitness* e então a probabilidade de seleção dos indivíduos é dada de acordo com seu *rank*.

Segundo Bergh (2001), a maioria dos Algoritmos Genéticos usa um operador de Seleção dos Sobreviventes onde a próxima geração é selecionada a partir da descendência. Um Algoritmo Genético usando essa abordagem é chamado de Algoritmo Genético Geracional. Essa técnica aumenta a diversidade da população e ajuda a prevenir a convergência prematura, mas diminui a taxa de convergência, pois boas soluções da atual geração podem ser jogadas fora. Existem também os métodos baseados no *fitness*, onde os

indivíduos da próxima geração são escolhidos a partir dos pais e dos filhos, dentre esses métodos temos o Elitismo, que consiste em inserir o melhor indivíduo da atual geração na nova geração caso não exista nenhum indivíduo na descendência com um *fitness* maior (EIBEN e SMITH, 2003).

### 3.4 – Estratégias de Evolução

As Estratégias de Evolução foram criadas nos anos 60 por Biernertk, Rechenberg e Schwefel que trabalhavam na Universidade de Berlim buscando otimizar corpos finos e tridimensionais em fluxos turbulentos (RUDOLPH, 1997; EIBEN e SMITH, 2003).

As Estratégias de Evolução (EE) em seu modelo inicial tinham a seguinte forma: Um indivíduo  $a$ , consistindo de um elemento  $X \in \mathfrak{R}^n$  (um vetor de tamanho  $n$ ) é mutado através da adição de um vetor aleatório normalmente distribuído  $Z \sim N(0, I_n)$  multiplicado por um escalar  $\sigma > 0$ , onde  $I_n$  é uma matriz identidade de tamanho  $n$ . O valor  $\sigma$  é chamado de tamanho do passo de mutação, pois determina a magnitude da mutação nos elementos de  $X$ . A seleção é feita com base em uma comparação do valor função objetivo do indivíduo velho e do indivíduo novo. Dada uma função objetivo  $f : \mathfrak{R}^n \rightarrow \mathfrak{R}$  a ser minimizada, a EE simples, começando com um indivíduo  $X_0 \in \mathfrak{R}^n$ , tem o seguinte esquema de iteração:

$$X_{t+1} = \begin{cases} X_t + \sigma_t Z_t & \text{se } f(X_t + \sigma_t Z_t) > f(X_t) \\ X_t & \text{senão} \end{cases}$$

onde  $t$  denota a iteração e  $Z_t$  é uma seqüência de vetores aleatórios normalmente distribuídos. O parâmetro  $\sigma_t$ , que controla a força da mutação, é ajustado conforme a regra de 1/5 de sucesso que define que a taxa de mutações com sucesso deveria ser de 1/5. Assim, se a taxa for maior que 1/5 o valor deve ser aumentado para fazer uma busca maior no espaço, se o valor for menor que 1/5 o valor deve ser diminuído pra concentrar mais a busca em volta da solução atual (RUDOLPH, 1997; EIBEN e SMITH, 2003).

Segundo Eiben e Smith (2003), as estratégias evolutivas são usadas para otimização de parâmetros contínuos, com ênfase na mutação para a criação de filhos.

A representação padrão dos indivíduos através de um vetor  $X \in \mathfrak{R}^n$  de valores reais é bastante direta já que na otimização de parâmetros, o problema pode ser dado como uma função objetivo  $\mathfrak{R}^n \rightarrow \mathfrak{R}$ . Porém, nas Estratégias Evolutivas contemporâneas os

indivíduos contêm também vários parâmetros que controlam sua mutação, parâmetros de estratégia. Esses parâmetros podem ser divididos em dois conjuntos, o conjunto  $\sigma$ , que são os tamanhos dos passos de mutação e o conjunto  $\alpha$  de ângulos que permitem a rotação da elipse no espaço de busca. Os indivíduos são mutados então de acordo com uma distribuição normal com média zero e uma matriz de covariância C, construída a partir de  $\sigma$  e  $\alpha$  (RUDOLPH, 1997; EIBEN e SMITH, 2003)..

O esquema básico de recombinação nas estratégias de evolução envolve dois pais para a criação de um filho. Os pais são sorteados aleatoriamente para a geração de cada filho e o processo é feito  $\lambda$  vezes para a criação de  $\lambda$  filhos. Usando a recombinação discreta um dos alelos dos pais é escolhido aleatoriamente com chance igual para ambos os pais. Usando a recombinação intermediária, é feita uma média entre os alelos (RUDOLPH, 1997; EIBEN e SMITH, 2003)..

Após a criação dos novos indivíduos, os  $\mu$  melhores indivíduos são escolhidos a partir somente da descendência ou a partir da união da descendência e dos pais, o primeiro método é chamado  $(\mu, \lambda)$  e o segundo  $(\mu + \lambda)$  (RUDOLPH, 1997; EIBEN e SMITH, 2003).

### 3.5 – Programação Evolutiva

O surgimento da Programação Evolutiva foi motivado pelo desejo da geração de uma abordagem alternativa em relação à inteligência artificial. Fogel teve a idéia de utilizar a simulação da evolução para desenvolver inteligência artificial que não dependesse de heurísticas, mas ao invés disso gerasse organismos com intelecto crescente com o tempo (PORTO, 1997).

Originalmente a Programação Evolutiva foi definida da seguinte maneira. Uma população de máquinas de estados finitas é exposta à uma seqüência de símbolos. A cada símbolo de entrada apresentado à cada máquina, o símbolo de saída é observado e comparado ao próximo símbolo de entrada, ou seja, a máquina faz uma previsão do próximo símbolo de entrada. É definida uma função para medir o valor de cada predição. Após ser feita a última predição, uma função da seqüência de valores é usada para indicar o *fitness* geral da máquina. Máquinas descendentes são criadas através da mutação de máquinas pai. Existem cinco modos de mutação possíveis: Mudar um símbolo de saída,

mudar uma transição de estados, adicionar um estado, remover um estado existente ou mudar o estado inicial.

Operadores de mutação são escolhidos de acordo com uma distribuição de probabilidade que pode ser uniforme ou não. O número de operações de mutação aplicadas à cada filho pode ser de acordo com uma distribuição de probabilidade ou definido previamente.

Um filho é criado para cada máquina da população. As melhores máquinas são escolhidas entre os pais e filhos como membros da próxima geração (PORTO, 1997).

Por motivos históricos a Programação Evolutiva é associada com tarefas de predição e o uso de máquinas de estado finitas como representação. Porém, a partir dos anos 90 surgiram diversas variantes para otimização de parâmetros de valores reais. A representação então, deve ser derivada de acordo com o problema a ser tratado, e os operadores de mutação devem ser criados de acordo com a representação.

Por ser geralmente mais utilizada na otimização de funções da forma  $f : \mathfrak{R}^n \rightarrow \mathfrak{R}$ , os indivíduos assumem a forma  $\langle x_1, \dots, x_n \rangle \in \mathfrak{R}^n$ . Atualmente é freqüentemente usada a auto-adaptação dos parâmetros de mutação, por isso a representação padrão dos indivíduos da Programação Evolutiva tem a forma  $\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n \rangle$ .

Na variante com auto-adaptação dos parâmetros de estratégia, a mutação transforma um cromossomo  $\langle x_1, \dots, x_n, \sigma_1, \dots, \sigma_n \rangle$  em  $\langle x_1', \dots, x_n', \sigma_1', \dots, \sigma_n' \rangle$ , onde

$$\begin{aligned}\sigma_i' &= \sigma_i (1 + \alpha N(0,1)), \\ x_i' &= x_i + \sigma_i' N_i(0,1).\end{aligned}$$

$N(0,1)$  é o resultado de um valor aleatório retirado de uma distribuição normal com media 0 e desvio padrão 1, e  $\alpha \approx 2$ .

Na Programação Evolutiva não há recombinação, cada indivíduo cria um filho através da mutação e a seleção dos sobreviventes é  $(\mu + \mu)$  (PORTO, 1997; EIBEN e SMITH, 2003).

## 3.6 – Programação Genética

Programação Genética é implementada como um Algoritmo Evolutivo no qual as estruturas de dados que sofrem adaptação são programas de computador executáveis. O calculo do *fitness* envolve a execução do programa. A Programação Genética é uma busca

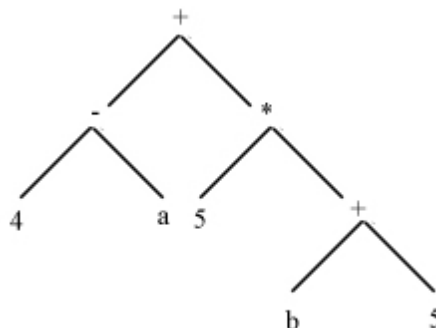
através da evolução por um programa, que quando executado produza o melhor *fitness*, no espaço de programas possíveis.

A população de indivíduos da Programação Genética é constituída de vários programas, os programas da população inicial são gerados aleatoriamente. Operadores genéticos são usados para criar novos programas a partir dos programas da geração atual selecionados como pais. Os indivíduos sobreviventes são selecionados e o processo se repete até que seja encontrado um programa correto ou algum outro critério de parada (KINNEAR, 1997).

Kinnear (1997) define que a Programação Genética, em um nível mais básico, é um Algoritmo Genético com escolhas incomuns para a representação dos indivíduos, operadores genéticos para essa representação e as técnicas de avaliação do *fitness*.

Na Programação Genética os indivíduos são representados por programas. Não existe uma forma única que seja usada por todas as implementações de Programação Genética, mas a maioria das implementações usam uma estrutura em árvore.

Na Figura 3.2 pode ser visto um exemplo de representação em árvore para a expressão  $(4 - a) + (5 * (b + 5))$



**Figura 3-1 – Exemplo de representação em árvore**

Os nós folhas são os Terminais, ou seja, variáveis e constantes, enquanto os nós internos são Funções, que recebem entradas e produzem saídas, mas não somente isso, podem também ser *loops* e instruções condicionais (KINNEAR, 1997; EIBEN e SMITH, 2003).

A representação do programa (Terminais e Funções) deve ser projetada juntamente com uma Máquina Virtual para interpretar os programas gerados.

A recombinação na Programação Genética cria duas árvores filhas a partir de duas árvores pais. Segundo Eiben e Smith (2003) a implementação mais comum é o

Cruzamento de Sub-árvore, onde uma sub-árvore é escolhida em cada árvore pai e as sub-árvores são trocadas uma pela outra para gerar os filhos.

A mutação típica substitui uma sub-árvore que começa em um nó aleatório por uma nova sub-árvore nova gerada aleatoriamente.

Eiben e Smith (2003) explicam que os pais geralmente são selecionados proporcionalmente ao *fitness*. Porém, em grandes populações é usado um método chamado *over-selection*, onde a população é ordenada pelo *fitness* e dividida em dois grupos, um contendo os  $x\%$  melhores indivíduos e outro contendo os restantes ( $100-x\%$ ) indivíduos, 80% dos pais são selecionados do primeiro grupo e 20% do segundo grupo.

A Programação Genética usa uma estratégia geracional para escolher os sobreviventes, o número de descendentes criados é o mesmo que o tamanho da população, que é totalmente substituída pelos novos indivíduos.

### **3.7 – Conclusão**

A Evolução natural talvez seja o mecanismo de solução de problemas mais eficiente que exista, por isso a computação busca se inspirar nesse processo. Esse capítulo tratou da Computação Evolutiva, área da computação que se inspira na evolução natural.

Na seção 2.2 é apresentado o processo básico dos Algoritmos Evolutivos: a partir de população de soluções potenciais, são selecionadas algumas para dar origem às novas soluções. Após isso, são selecionadas as soluções que continuarão a existir no algoritmo. O processo se repete até uma condição de término ser satisfeita. As novas soluções são criadas utilizando-se operadores de recombinação e mutação e as seleções são feitas com base na capacidade de cada solução em resolver o problema.

Os quatro tipos básicos de Algoritmos Evolutivos são: Algoritmo Genético, Estratégias de Evolução, Programação Evolutiva e Programação Genética, apresentados nas seções 3.3, 3.4, 3.5 e 3.6, respectivamente. Os algoritmos diferem nos tipos de recombinação, mutação, seleção, representação e finalidade.

# 4 – PARTICLE SWARM OPTIMIZATION (PSO)

## 4.1 – Introdução

*Particle Swarm Optimization* (PSO) é um método de otimização baseado em população inicialmente proposto por Kennedy e Eberhart (1995) e modelado de acordo com o comportamento social de bandos de pássaros. O PSO tem sido aplicado com sucesso a problemas de busca em muitos domínios, particularmente na otimização de funções numéricas. Além disso, é capaz de resolver maioria dos problemas de otimização que podem ser resolvidos usando Algoritmos Genéticos (KENNEDY e EBERHART, 1995; BERGH, 2001; COHEN e CASTRO., 2006).

No PSO, cada indivíduo na população busca pela solução levando em conta o melhor indivíduo em uma certa vizinhança e a melhor posição pessoal já encontrada.

Segundo Kennedy e Eberhart (1995), *Particle Swarm Optimization* tem raízes em duas metodologias principais. Vida Artificial (A-life) e Computação Evolutiva. A classificação do algoritmo será discutida mais adiante.

Suas principais características incluem facilidade de implementação, baixo custo de memória e velocidade (KENNEDY e EBERHART, 1995; BERGH, 2001).

## 4.2 – Origem

O PSO foi criado por Kennedy e Eberhart (1995), o algoritmo nasceu da simulação de um ambiente social simplificado onde a intenção era simular a movimentação de um bando de pássaros.

Kennedy e Eberhart (1995) notam que algumas simulações propostas anteriormente se baseiam na distância entre os indivíduos, onde se imagina que a sincronia do comportamento do grupo está relacionada aos esforços dos indivíduos em manter uma distância ótima entre si e seus vizinhos. Segundo os autores, é sensato supor, com certa abstração, que algumas das mesmas regras fundamentam o comportamento social animal, incluindo rebanhos, cardumes, bandos e até mesmo humanos. Além disso, se afirma que o compartilhamento social de informações oferece uma vantagem evolutiva, indivíduos podem se beneficiar das descobertas e experiências passadas de outros indivíduos. Ressalta-se também que os humanos tendem a ajustar crenças e atitudes conforme outros humanos e usam experiências passadas na tomada de decisões e comportamentos.

### 4.3 – Particle Swarm Optimization

De acordo com Merwe e Engelbrecht (2003), dado um problema, o algoritmo mantém uma população de partículas onde cada partícula representa uma solução potencial para o problema. Cada partícula representa uma posição em um espaço de busca multidimensional e mantém as seguintes informações:

- $x_i$  – Posição atual da partícula.
- $v_i$  – Velocidade Atual da partícula.
- $y_i$  – Melhor posição pessoal da partícula (Melhor posição em que a partícula já esteve).

A cada iteração a partícula tem sua velocidade ajustada de acordo com a seguinte equação:

$$v_{i,k}(t + 1) = wv_{ik}(t) + c_1r_{1,k}(t)(y_{i,k}(t) - x_{i,k}(t)) + c_2r_{2,k}(t)(\hat{y}_k(t) - x_{i,k}(t)) \quad (4.1)$$

Onde  $v_{i,k}$  denota a  $k$ -ésima dimensão do vetor velocidade associado à  $i$ -ésima partícula. A velocidade é atualizada separadamente para cada dimensão  $k \in i..n$ .  $r_1$  e  $r_2$  são valores aleatórios,  $r_1 \sim U(0,1)$  e  $r_2 \sim U(0,1)$  que contribuem para a natureza estocástica do algoritmo.  $c_1$  e  $c_2$ ,  $0 < c_1, c_2 \leq 2$ , são coeficientes de aceleração,  $c_1$  regula o passo máximo na direção da melhor posição pessoal e  $c_2$  na direção da posição global (gbest) ou da vizinhança (lbest) (BERGH, 2001). O termo  $c_1r_{1,k}(t)(y_{i,k}(t) - x_{i,k}(t))$  leva em conta as experiências passadas da partícula, é associado a cognição. O termo  $c_2r_{2,k}(t)(\hat{y}_k(t) - x_{i,k}(t))$  é o termo social, pois a partícula se inspira na melhor solução ao seu redor (BERGH, 2001; COHEN e CASTRO, 2006).

Na versão gbest do PSO,  $\hat{y}$  representa a melhor posição já encontrada por qualquer uma das partículas (*global best*), Bergh (2001) explica que essa versão oferece uma taxa de convergência mais rápida, porém é menos robusta. Na versão lbest, a população é dividida em vizinhanças, e  $\hat{y}_j$  representa a melhor posição encontrada na vizinhança da partícula  $j$  (*local best*), para uma população de tamanho  $s$  e vizinhança de tamanho  $l$  as equações são atualizadas da seguinte maneira:

$$N_i = \{y_{i-l}(t), y_{i-l+1}(t), \dots, y_i(t), \dots, y_{i+l-1}(t), y_{i+l}(t)\}$$

$$\hat{y}_i(t+1) \in N_i | f(\hat{y}_i(t+1)) = \min\{f(y)\}, \forall y \in N_i$$

$$v_{i,k}(t + 1) = wv_{ik}(t) + c_1r_{1,k}(t)(y_{i,k}(t) - x_{i,k}(t)) + c_2r_{2,k}(t)(\hat{y}_{j,k}(t) - x_{i,k}(t))$$

A posição da partícula é atualizada usando o novo vetor velocidade:

$$x_i(t + 1) = x_i(t) + v_i(t+1) \quad (4.3)$$



Usando o símbolo  $f$  para denotar a função objetivo, a melhor posição pessoal da partícula  $i$  é calculada como:

$$y_i(t + 1) = y_i(t) \text{ se } f(x_i(t + 1)) \geq f(y_i(t))$$

$$x_i(t + 1) \text{ se } f(x_i(t + 1)) < f(y_i(t))$$

O algoritmo consiste de repetidas aplicações das equações de atualização apresentadas:

**Criar** e Inicializar as partículas

**Repita**

**Para** cada partícula  $i \in [1..s]$

**Se**  $f(x_i) < f(y_i)$

**Então**  $x_i = y_i$

**Se**  $y_i < \hat{y}$

**Então**  $\hat{y} = y_i$

Cada atributo  $x_{i,j}$  é inicializado aleatoriamente com um valor no intervalo  $[-x_{max}, x_{max}]$ , para todo  $i \in 1..s$  e  $j \in 1..n$ . As velocidades  $v_{i,j}$  geralmente também são inicializadas aleatoriamente em um intervalo  $[-v_{max}, v_{max}]$ , onde  $v_{max} = k \times x_{max}$ ,  $0.1 \leq k \leq 1.0$ . O critério de parada pode ser um número determinado de iterações ou outro critério dependendo do problema (BERGH, 2001).

## 4.4 – Classificação

Para Kennedy e Eberhart (1995), criadores do PSO, o algoritmo parece repousar em algum lugar entre o Algoritmo Genético e a Programação Evolutiva. Assim como na Programação Evolutiva o algoritmo é dependente de processos estocásticos. O ajuste da posição das partículas em relação à sua melhor posição pessoal e à melhor da vizinhança é similar a operação de cruzamento no Algoritmo Genético. Assim como na Computação Evolutiva, o conceito de *fitness* também é usado. Único ao PSO é o vôo das soluções potenciais pelo espaço de busca acelerando em direção às melhores soluções.

Bergh (2001) em seu trabalho, explica que o PSO é claramente relacionado a alguns Algoritmos Evolutivos. Um dos motivos é que o PSO mantém uma população de indivíduos, assim como os Algoritmos Evolutivos. Se as melhores posições individuais ( $y_i$ )

forem tratadas como parte da população então existe um mecanismo fraco de seleção que se assemelha ao  $(\mu + \lambda)$  das Estratégias de Evolução, onde a descendência substitui os pais caso seja mais adaptada (maior *fitness*). Bergh (2001) ainda afirma que se o termo  $v_{i,j}(t)$  for removido da equação de atualização da velocidade, a mesma pode ser interpretada como um operador de mutação onde a força da mutação é controlada pelos seus dois pais, a melhor posição já encontrada pelo indivíduo e a posição do melhor indivíduo da vizinhança.

O PSO pode ser visto como um processo de adaptação ao invés de substituição das populações anteriores. Bergh (2001) diz que isso deixa a diferença entre o PSO os Algoritmos Evolutivos mais clara, o PSO mantém informações sobre a posição e velocidade, enquanto os Algoritmos Evolutivos tradicionais apenas rastreiam a posição.

## 4.5 – Conclusão

Esse capítulo apresentou o PSO, método de otimização inspirado no comportamento social de bandos de pássaros. Os indivíduos, ou partículas, buscam pela melhor solução levando em conta a melhor posição em que já esteve e o melhor indivíduo na vizinhança.

A seção 4.2 fala sobre como o algoritmo foi descoberto, durante a simulação da movimentação de bandos de pássaros. Adiante, a seção 4.3 apresenta a definição formal do método. Finalmente, a seção 4.4 diz que o PSO está relacionado aos Algoritmos Evolutivos e ocupa algum lugar entre o Algoritmo Genético e a Programação Evolutiva.

# 5 – MÉTODOS DE CLUSTERIZAÇÃO USANDO O PSO

## 5.1 – Introdução

Podem ser encontrados na literatura alguns trabalhos que modificam o PSO para a Clusterização de Dados, nessa seção esses trabalhos serão brevemente discutidos.

## 5.2 – Clusterização com o PSO

No método proposto por Merwe e Engelbrecht (2003), cada partícula do PSO é composta por um vetor de centróides de tamanho  $N_c$ , onde  $N_c$  é o número de grupos a serem criados, e representa uma solução completa para o problema. A cada iteração, os dados são atribuídos ao centróide ao qual estão mais próximos, as soluções são avaliadas e então atualizadas de acordo com sua melhor posição no espaço de busca e a melhor posição já encontrada por alguma partícula. Os autores propõem também um híbrido onde o resultado encontrado pelo algoritmo *k-means* é introduzido na população inicial do PSO como uma das partículas.

Cui et al. (2005) utilizam o PSO para a Clusterização de documentos. Os documentos são representados como vetores de pesos, onde cada posição do vetor corresponde ao peso de determinado termo no documento. Desse modo, os documentos tomam a forma de pontos em um espaço multidimensional, assim o PSO pode ser aplicado. O algoritmo de classificação usado é semelhante ao proposto por Merwe e Engelbrecht (2003). Segundo os autores, o comportamento do PSO tem dois estágios, um estágio de busca global e um estágio de busca local. Nas iterações iniciais, a equação de velocidade provê uma diversidade ao *swarm* através da mudança do momento das partículas, evitando a estagnação em ótimos locais. Múltiplas partículas buscando a solução paralelamente podem explorar melhor o espaço de busca. Essas iterações iniciais são classificadas como o estágio de busca global. Com o passar das iterações, a velocidade das partículas diminui com a aproximação da solução ótima e a partícula passa a explorar um espaço menor. O estágio de busca global gradualmente muda para o estágio de busca local. Quando se trata de grandes conjuntos de documentos, a mudança lenta do estágio de busca global para o de busca local faz com que o PSO demore mais que o *k-means* para convergir. Para solucionar

esse problema, os autores propõem um híbrido, onde o PSO é executado por um tempo e após isso o *k-means* é usado para refinar a solução encontrada. Através desse método os autores tentam combinar a eficiência do PSO com a velocidade do *k-means*.

Cohen e Castro (2006) propõem o PSC (*Particle Swarm Clustering*), onde, ao contrário do algoritmo PSO, em que cada partícula codifica uma solução completa do problema, cada partícula corresponde ao protótipo de um único *cluster*, representando apenas parte da solução. As partículas não são avaliadas, elas apenas se movem buscando se posicionar em regiões do espaço que representem os *clusters* naturais. Para cada dado que é apresentado ao *swarm*, a partícula mais próxima do dado se move em sua direção, sendo influenciada por sua melhor posição anterior em relação àquele dado e pela melhor posição de uma partícula em relação àquele dado. A cada iteração, todos os dados são apresentados ao *swarm*, caso alguma das partículas não tenha vencido nenhuma vez, ou seja, caso não tenha sido a mais próxima de nenhum dos dados, ela é ajustada em direção à partícula que mais venceu.

Xiao et al. (2003) utilizam um híbrido entre *Self-Organizing Maps* (SOM) e o PSO. No método proposto, o PSO é usado para melhorar os pesos do SOM. No primeiro estágio, SOM é utilizado para clusterizar o conjunto de dados e gera um grupo de pesos. No segundo estágio, o PSO é inicializado com os pesos produzidos pelo SOM e então é usado para refinar esses pesos.

Omran et al. (2005) Propõem um método híbrido, que utiliza o PSO binário e *k-means*, o método é chamado de *Dynamic Clustering using a Particle Swarm Optimization algorithm* (DBPSO) e é capaz de encontrar o número de *clusters* correto. O PSO binário é uma versão do PSO, adaptada para buscar no espaço binário. As posições das partículas são compostas de valores 0 ou 1 e a velocidade é interpretada como a probabilidade de mudar um bit de 0 para 1 ou vice-versa. O DBPSO funciona da seguinte maneira: dado um conjunto de dados, um conjunto de  $N_c$  centróides é escolhido dentre esses dados,  $N_c$  corresponde ao número máximo de centróides desejado. Cada partícula é formada por um vetor de tamanho  $N_c$ , se a posição  $i$  desse vetor for 0, isso significa que o centróide  $i$  não é parte da solução proposta pela partícula. Em cada iteração, O PSO binário é utilizado para encontrar o melhor subconjunto de centróides e então o *k-means* é utilizado para refinar os centróides escolhidos. Após isso, os centróides que não foram escolhidos são substituídos por novos centróides aleatórios, os indivíduos são novamente inicializados aleatoriamente e uma nova iteração começa.

O método proposto por Merwe e Engelbrecht (2003) é o mais simples e também aquele que mais se adequa as idéias do PSO, onde cada indivíduo representa uma solução ao problema. Por isso, decidiu-se trabalhar sobre esse método. Adiante, o método será discutido em mais detalhes.

## 5.3 – O método proposto por Merwe e Engelbrecht

### 5.3.1 – Introdução

Dentre as adaptações do PSO ao problema de Clusterização, a abordagem de Merwe e Enelbrecht (2003) é aquela que mais se adequa às idéias do PSO, onde cada indivíduo ou partícula representa uma solução completa para o problema.

### 5.3.2 – O Algoritmo

No método proposto por Merwe e Enelbrecht (2003), uma partícula  $x_i$  é construída da seguinte maneira:

$$x_i = (m_{i1}, m_{i2}, \dots, m_{ij}, \dots, m_{iN_c})$$

Onde  $N_c$  é o número de *clusters* a serem criados e  $m_{ij}$  corresponde ao  $j^{\text{ésimo}}$  centróide da  $i^{\text{ésima}}$  partícula, isto é, o centróide do *cluster*  $C_{ij}$ . Assim, uma única partícula representa uma solução candidata a uma dada instância do problema de Clusterização.

Cada partícula é avaliada através da seguinte equação:

$$J_e = \frac{\sum_{j=1}^{N_c} \left[ \sum_{\forall Z_p \in C_{ij}} d(Z_p, m_{ij}) / |C_{ij}| \right]}{N_c} \quad (F1)$$

Onde  $Z_p$  denota o  $p^{\text{ésimo}}$  dado,  $|C_{ij}|$  é o número de dados pertencentes ao *cluster*  $C_{ij}$  e  $d$  é a distância euclidiana entre  $Z_p$  e  $m_{ij}$ .

Em seu trabalho, Merwe e Engelbrecht (2003), propõem dois métodos, ambos construídos sobre o PSO *gbest* padrão. Um dos métodos é um híbrido com o *k-means*, onde o resultado do *k-means* é usado como uma das partículas do PSO.

O algoritmo de Clusterização proposto pode ser descrito como:

**Inicialize** os centróides dos *clusters* de cada partícula aleatoriamente

**Para**  $t = 1$  até  $t_{\max}$

**Para** cada partícula  $i$

**Para** cada dado  $z_p$

Calcule  $d(z_p, m_{ij})$  para todos centróides  $m_{ij}$

Atribua  $z_p$  ao *cluster*  $C_{ij}$  tal que:

$$d(z_p, m_{ij}) = \min_{\forall k = 1, \dots, N_c} \{d(z_p, m_{ik})\}$$

**Calcule** o *fitness* usando a equação F1

**Atualize** o melhor global e os melhores locais

**Atualize** os centróides dos *clusters* usando as equações (4.1) e (4.3)

### 5.3.3 – A Função de Avaliação e modificações propostas

A função de avaliação, ou função objetivo, desempenha um papel fundamental em qualquer algoritmo evolutivo, ela determina o quão boa uma determinada solução é, quão bem ela resolve o problema.

Analisando mais profundamente a equação F1, usada por Merwe e Engelbrecht (2003) pode ser visto que primeiro, para cada *cluster*  $C_{ij}$ , a distância média dos dados pertencentes ao *cluster* ao seu centróide  $m_{ij}$  é calculada. Então, é calculada outra média, a média das distâncias médias aos centróides de todos os *clusters*  $C_{ij}$ , esse é o resultado da equação.

Pode ser visto que um *cluster*  $C_{ij}$  com apenas um dado influenciará no resultado final (valor de F1) tanto quanto um *cluster*  $C_{ik}$  com vários dados, e assim, uma partícula que não representa uma boa solução acabará sendo avaliada como se representasse. Por exemplo, suponhamos que um dos *clusters* da partícula tenha um dado que esteja bem próximo ao seu centróide, e que outro *cluster* tenha vários dados que não estejam muito perto de seu centróide, mas que também não estejam muito distantes. A distância média para o primeiro *cluster* é pequena, mas ele contém apenas um dado. A distância média para o segundo *cluster* não é pequena e nem grande, mas ele contém vários dados. A distância média ao centróide do segundo *cluster* deveria ter um peso maior no cálculo da função objetivo. Porém, isso não acontece na equação F1, pois a equação dá um peso igual para ambas as distâncias médias. Portanto, essa solução, que não é uma boa solução, pode acabar sendo interpretada como tal.

Além disso, essa equação não recompensa soluções homogêneas, isto é, soluções onde quantidade de dados nos *clusters* é homogênea não são recompensadas.

Para resolver esse problema, é proposta a equação F2, onde o número de dados em cada *cluster* é levado em conta no cálculo da qualidade. A distância média dos dados ao centróide de cada *cluster* é multiplicada pela porcentagem de dados que aquele *cluster* possui:

$$F = \left\{ \sum_{j=1}^{N_c} \left[ \left( \sum_{\forall Z_p \in C_{ij}} d(z_p.m_{ij}) / |C_{ij}| \right) \times (|C_{ij}| / N_o) \right] \right\} \quad (F2)$$

A equação F2 pode ser reescrita, da seguinte maneira:

$$F = \left( \sum_{j=1}^{N_c} \left( \sum_{\forall Z_p \in C_{ij}} d(z_p.m_{ij}) \right) \right) / N_o$$

Onde  $N_o$  é o número de dados a serem clusterizados.

Para levar em conta a distribuição de dados entre os *clusters*, a equação pode ser modificada:

$$F' = F \times (|C_{ik}| - |C_{il}| + 1) \quad (F3)$$

Tal que,  $|C_{ik}| = \max_{\forall j = 1, \dots, N_c} \{|C_{ij}|\}$  e  $|C_{il}| = \min_{\forall j = 1, \dots, N_c} \{|C_{ij}|\}$ .

Mais adiante, resultados usando ambas as equações e uma comparação com o método de Merwe e Engelbrecht serão mostrados.

## 5.4 – Conclusão

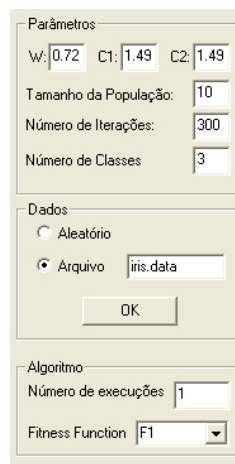
Nesse capítulo foram apresentados alguns métodos de Clusterização existentes que utilizam o PSO. Dentre esses métodos, o método proposto por Merwe e Engelbrecht, por ser aquele que mais se adequa as idéias do PSO, foi escolhido e discutido mais a fundo na seção 5.3. Nesse método, cada partícula representa uma solução completa para o problema. Para avaliar as partículas, é usada uma função que não leva em conta o número de dados em cada grupo, podendo gerar alguns problemas. Assim, a seção 5.3.3 propõe duas novas funções de avaliação, uma que leva em conta o número de dados em cada cluster e uma que busca criar clusters mais uniformes. Adiante, as funções propostas serão avaliadas utilizando *benchmarks* da área.

## 6 – RESULTADOS

### 6.1 – A Implementação

O algoritmo, juntamente com as três funções de avaliação, foi implementado utilizando a linguagem de programação c++, com o intuito de criar um programa capaz de executar mais rapidamente. O programa criado é orientado a objetos, dando assim uma melhor estrutura ao código, pois é interessante que seja de fácil manutenção, ou seja, que suporte a introdução de novas idéias e testes e que as mesmas classes possam ser reutilizadas em trabalhos futuros.

O programa permite que os parâmetros do algoritmo sejam alterados, permite a escolha da função de avaliação e conjunto de dados que se deseja usar e número de vezes que se deseja executar o algoritmo (Figura 8.1). É capaz de exibir, em duas dimensões que podem ser escolhidas, os dados e os grupos gerados pelo algoritmo, onde os dados pertencentes ao mesmo grupo possuem a mesma cor e forma (Figura 8.2). Também é capaz de criar um gráfico que mostra a evolução do *fitness* da melhor partícula ao longo das iterações (Figura 8.3). Ao final da execução, o programa exibe a taxa de acerto média do algoritmo para o número de execuções escolhido, juntamente com o desvio padrão, exibe também as distâncias *intra-cluster* e distância *inter-cluster* média, para a melhor partícula no número de execuções escolhido. Além disso, exibe, para cada grupo criado, quantos dados de cada classe do conjunto de dados existem naquele grupo (Figura 8.4). A interface completa pode ser vista na Figura 8.5.



The image shows a dialog box titled 'Parâmetros' (Parameters) with three sections: 'Parâmetros', 'Dados' (Data), and 'Algoritmo' (Algorithm). In the 'Parâmetros' section, there are input fields for 'w:' (0.72), 'C1:' (1.49), and 'C2:' (1.49). Below these are 'Tamanho da População:' (10), 'Número de Iterações:' (300), and 'Número de Classes' (3). The 'Dados' section has radio buttons for 'Aleatório' (unselected) and 'Arquivo' (selected), with a text field containing 'iris.data'. An 'OK' button is at the bottom. The 'Algoritmo' section has 'Número de execuções' (1) and a dropdown menu for 'Fitness Function' set to 'F1'.

Figura 6-1 – Parâmetros do Algoritmo



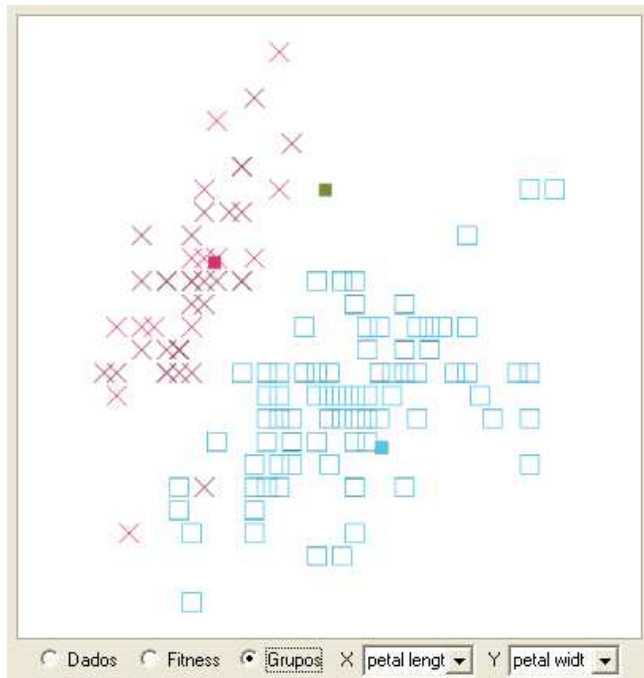


Figura 6-2 – Visualização dos grupos criados pelo algoritmo

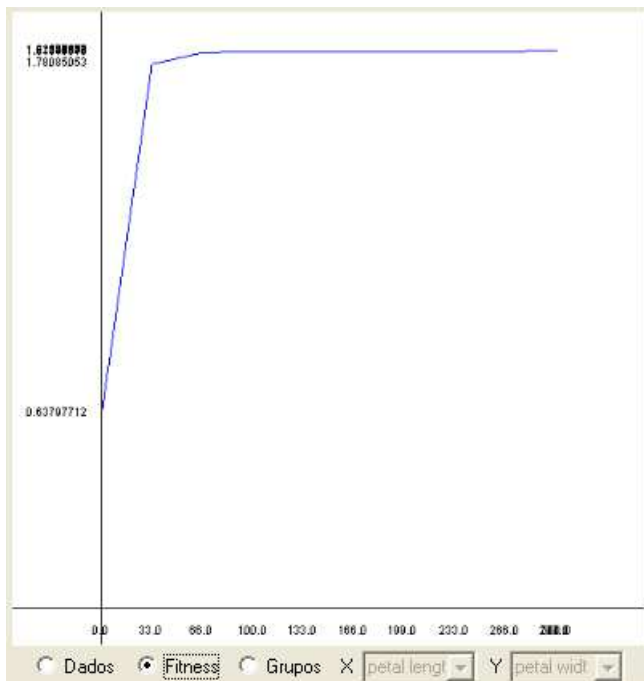


Figura 6-3 – *Fitness* da melhor partícula ao longo das iterações

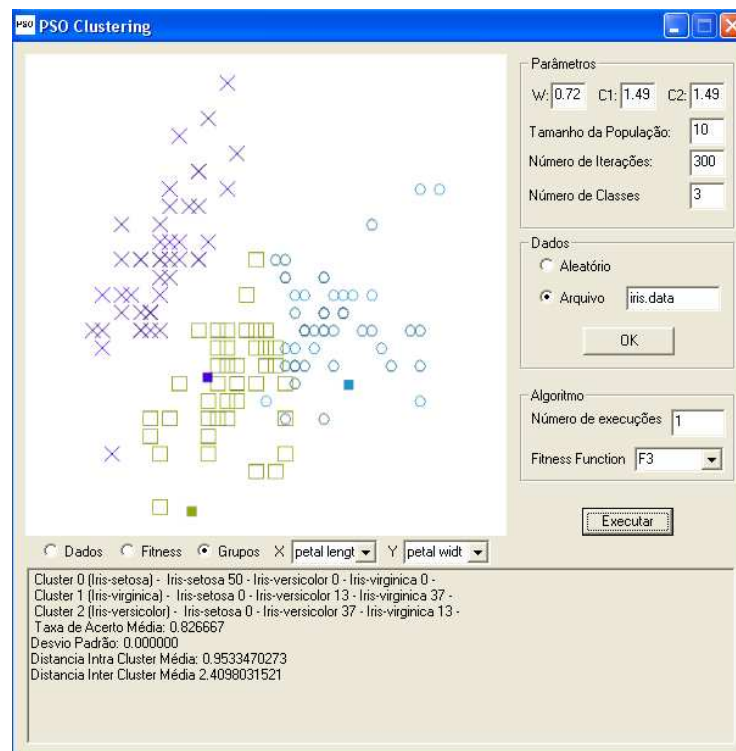
```

Cluster 0 (Iris-setosa) - Iris-setosa 50 - Iris-versicolor 1 - Iris-virginica 0 -
Cluster 1 (Iris-versicolor) - Iris-setosa 0 - Iris-versicolor 0 - Iris-virginica 0 -
Cluster 2 (Iris-virginica) - Iris-setosa 0 - Iris-versicolor 49 - Iris-virginica 50 -
Taxa de Acerto Média: 0.666667
Desvio Padrão: 0.000000
Distancia Intra Cluster Média: 0.7331130505
Distancia Inter Cluster Média 1.9792265892

```

**Figura 6-4 – Resultados do Algoritmo**

Foram criadas as classes: Particle, que representa uma partícula, juntamente com os métodos e atributos para cada partícula; Swarm, que representa a população de partículas como um todo e métodos e atributos da população em geral; Cluster, que representa o *cluster* formado por um centróide, e métodos e atributos para o *cluster*; Algorithm, responsável pela execução do algoritmo em si; DataSet, que representa o conjunto de dados e métodos e atributos para trabalhar com o conjunto de dados; Instance, que representa um único dado; File, para carregar os dados do arquivo; D3DX\_Control, PSO\_MFCDlg, responsáveis pela interface com o usuário e PSO\_MFC\_Business, responsável pela ligação entre a interface e as classes do algoritmo e de dados.



**Figura 6-5 – Interface do programa**

## 6.2 – Resultados

Para avaliar e comparar os métodos, foram usados três *benchmarks* amplamente abordados na literatura: *Iris*, *Glass* e *Wine*. Todos podem ser encontrados no *UCI Repository of Machine Learning Databases*.

O *benchmark Iris* apresenta cento e cinquenta instâncias da flor Iris, divididas em três classes com cinquenta instâncias cada. A primeira classe corresponde ao tipo *Setosa*, a segunda ao tipo *Versicolour* e a terceira *Virginica*. As instâncias possuem quatro atributos de valores reais, *sepal length* (comprimento da sépala), *sepal width* (largura da sépala), *petal length* (comprimento da pétala), *petal width* (largura da pétala). Uma das classes (*Setosa*) é linearmente separável das outras duas, que não são linearmente separáveis entre si.

O *benchmark Wine* possui cento e setenta e oito instâncias de vinho. Esses dados são resultados de uma análise química realizada em vinhos da mesma região da Itália, mas vindos de diferentes cultivares. Análise determinou as quantidades de treze constituintes encontrados em cada um dos três tipos de vinho. Os atributos são: *alcohol*, *malic acid*, *ash*, *alcalinity of ash*, *magnesium*, *total phenols*, *flavanoids*, *noflavanoid fenols*, *proanthocyanins*, *color intensity*, *hue*, *OD280/OD315 of diluted wines*, *praline*. A primeira classe contém cinquenta e nove instâncias, a segunda classe contém setenta e uma e a terceira quarenta e oito.

O *benchmark Glass* apresenta duzentas e quatorze instâncias de vidros divididas em sete tipos. Setenta instâncias em *building windows float processed*, dezessete em *vehicle windows float processed*, setenta e seis em *buiding windows non-float processed*, nenhuma em *vehicle windows non-float processed*, treze em *containers*, 9 em *tableware*, vinte e nove em *headlamps*. Os atributos dos dados são: *refrative index*, *sodium*, *magnesium*, *aluminium*, *silicon*, *potassium*, *calcium*, *barium*, *iron*.

A tabela 9.1 sumariza as características dos *benchmarks*.

<b>Benchmark</b>	<b>Número de Instâncias</b>	<b>Número de Atributos</b>	<b>Número de Classes</b>
<i>Iris</i>	150	4	3
<i>Wine</i>	178	13	3
<i>Glass</i>	214	9	7

**Tabela 6-1 – Características dos *benchmarks*.**

Para cada conjunto de dados, o programa foi executado por 30 vezes, com 200 iterações, 10 partículas,  $w = 0,72$ ;  $c1 = 1,49$ ;  $c2 = 1,49$ ; utilizando cada uma das funções de avaliação. Esses são os mesmos valores usados no trabalho de Merwe e Engelbrecht (2003), de acordo com eles, tais valores para  $w$ ,  $c1$  e  $c2$  asseguram uma boa convergência. O número de iterações relativamente pequeno foi escolhido devido à rápida convergência do PSO.

Cada execução é avaliada de acordo com sua taxa de acerto, ou seja, é calculada a porcentagem de acertos obtida pelo algoritmo, de acordo com a fórmula:

$$t = \frac{N_r}{N_o}$$

Onde  $N_o$  é o número de dados a serem clusterizados e  $N_r$  é o número de dados clusterizados corretamente.

Primeiramente, para saber se determinado dado foi agrupado corretamente, é preciso saber a que classe do *benchmark* o grupo encontrado pelo algoritmo corresponde. Dado que:

- $G = \{g_1, g_2, \dots, g_k\}$  é um conjunto com os grupos gerados pelo algoritmo.
- $B = \{b_1, b_2, \dots, b_k\}$  é o conjunto de classes do *benchmark*.
- $r_i$  é a classe representada pelo grupo  $i$ .
- $n_{ij}$  é o número de dados da classe  $j$  no grupo  $i$ .

Essa correspondência pode ser calculada através do seguinte algoritmo:

**Enquanto**  $G \neq \emptyset$

**Encontrar** maior  $n_{ij}$ , tal que  $g_i \in G$  e  $b_j \in B$

$r_i = j$

$G = G - g_i$

$B = B - b_j$

Dessa maneira, todos os dados que estiverem em  $c_i$  e também pertencerem a classe  $r_i$  estão agrupados corretamente, os outros estão agrupados incorretamente.

Outras medidas que podemos usar para a avaliação do algoritmo são a distância *intra-cluster* e distância *inter-cluster*.

A distância *intra-cluster* mede a densidade dos *clusters* criados, o quão compactos são esses *clusters*, já que dados no mesmo *cluster* devem ser similares. Esse dado foi medido através da média da distância média entre os dados pertencentes a cada grupo.

A distância *inter-cluster* mede a separação entre os *clusters* encontrados, os *clusters* devem estar o mais distante possível uns dos outros. Esse dado foi medido através da distância entre os centros de massa de cada grupo.

A taxa de acertos média das 30 execuções, para cada função de avaliação e conjunto de dados, juntamente com o desvio padrão  $\sigma$  é apresentada na tabela 9.2.

A distância *intra-cluster* e a distância *inter-cluster* média, nas 30 execuções, para cada conjunto de dados e função de avaliação, são dadas nas tabelas 9.3 e 9.4.

<i>Benchmark</i>	$t \pm \sigma$		
	F1	F2	F3
<i>Iris</i>	66.6444% $\pm$ 9.6156%	83.1333% $\pm$ 8.4837 %	88.3778% $\pm$ 10.6421 %
<i>Wine</i>	68.9139% $\pm$ 6.4636%	71.2172% $\pm$ 0.5254%	71.8726% $\pm$ 0.1425%
<i>Glass</i>	42.3053% $\pm$ 5.1697%	46.3396% $\pm$ 3.7626%	43,3178 $\pm$ 3,4833

Tabela 6-2 – Comparação entre o acerto médio das execuções utilizando as funções de *fitness* F1, F2 e F3

<i>Benchmark</i>	Distância <i>intra-cluster</i>		
	F1	F2	F3
<i>Iris</i>	0,7727	0,8954	1,0126
<i>Wine</i>	135,5002	132,6602	133,6435
<i>Glass</i>	1,1903	1,8265	1,8334

Tabela 6-3 – Comparação entre a distância *intra-cluster* das execuções utilizando as funções de *fitness* F1, F2 e F3

<i>Benchmark</i>	Distância <i>inter-cluster</i>		
	F1	F2	F3
<i>Iris</i>	1,8253	2,2899	2,3515
<i>Wine</i>	378,2947	365,7352	345,8403
<i>Glass</i>	3,4512	5,2463	3,8923

Tabela 6-4 – Comparação entre a distância *inter-cluster* das execuções utilizando as funções de *fitness* F1, F2 e F3

As mudanças na função de avaliação trouxeram boas melhoras aos resultados nos *benchmarks* avaliados.

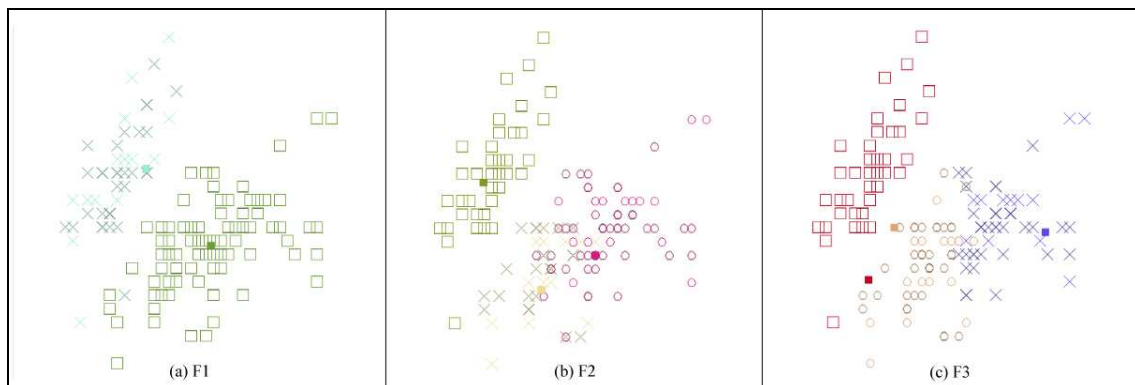
Em *Iris*, onde os *clusters* têm tamanhos uniformes, a equação F3 produziu resultados muito bons, a equação F2 também produziu um bom resultado e a equação F1 produziu um resultado razoável. A equações F1, F2 e F3 são capazes de encontrar

corretamente a classe *Setosa*, que é linearmente separável das outras duas. Porém, F1 tem mais dificuldades em separar as outras duas classes, já F2 e principalmente F3 conseguem chegar bem próximo da separação ideal.

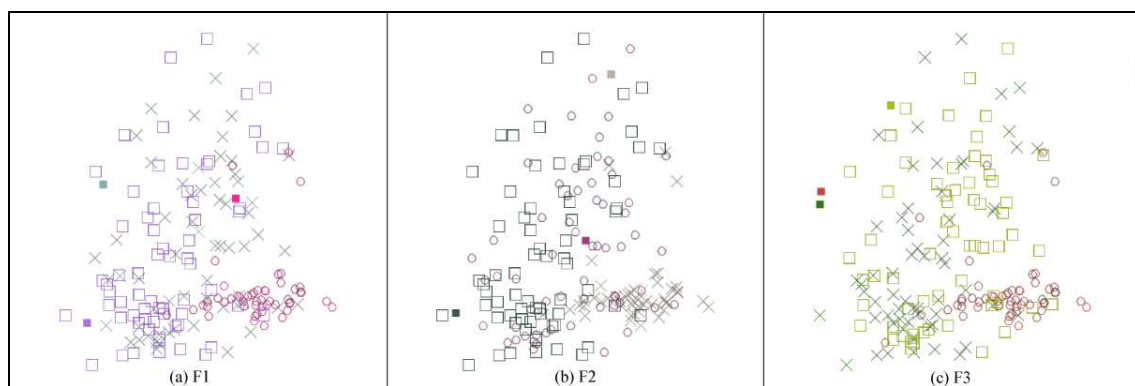
No conjunto de dados *Wine*, um conjunto um pouco mais difícil, as três equações produziram bons resultados. Porém, os resultados utilizando as equações F2 e F3 são ligeiramente superiores.

Já em *Glass*, os resultados encontrados, tanto para F1, F2 e F3 são apenas razoáveis, por se tratar de um *benchmark* onde as classes são difíceis de separar, com sobreposições entre si. Ainda assim, os resultados encontrados por F2 e F3 são superiores.

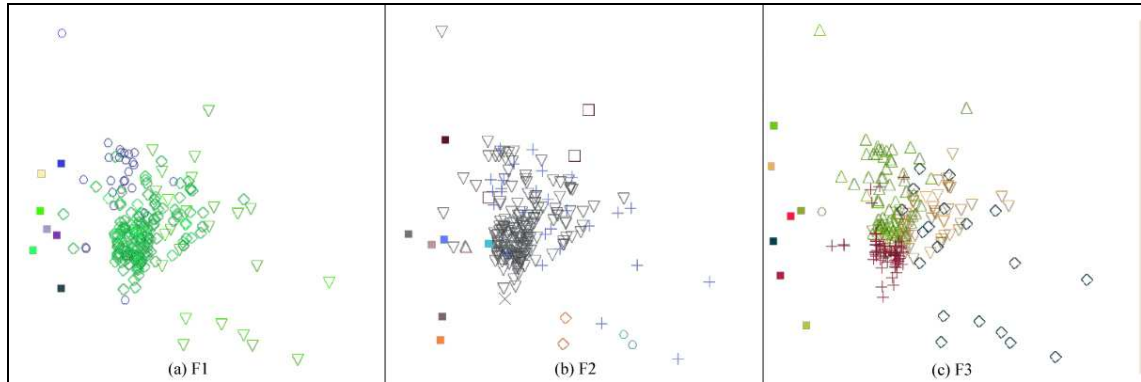
É importante perceber que a equação F3 leva as partículas em direção a *clusters* com dados distribuídos mais uniformemente, portanto, deveria ser usada em problemas onde se é conhecido que os dados são uniformemente distribuídos nos *clusters*. Caso contrário, a equação F2 deve ser usada.



**Figura 6-6 – Agrupamentos encontrados pelas 3 funções de avaliação para o benchmark Iris.**



**Figura 6-7 – Agrupamentos encontrados pelas 3 funções de avaliação para o benchmark Wine.**

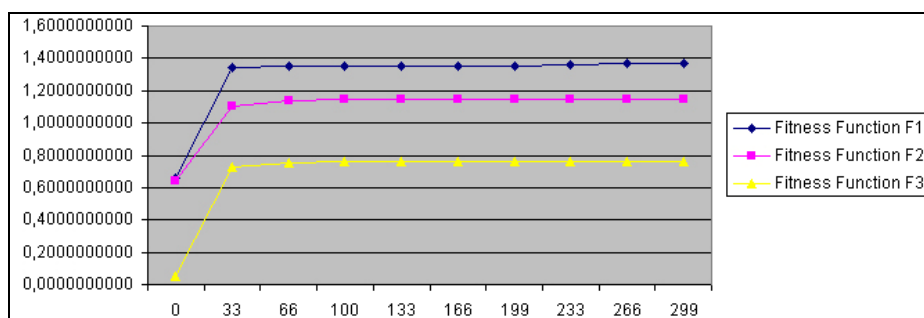


**Figura 6-8 – Agrupamentos encontrados pelas 3 funções de avaliação para o benchmark Glass.**

Nas figuras 9.1, 9.2 e 9.3 podemos ver alguns exemplos de agrupamentos encontrados. Na figura 9.1 temos exemplos de agrupamento pra o conjunto de dados Iris, em (a) o algoritmo utilizando a função F1 encontrou o grupo correto para 71,9% dos dados, em (b), F2 encontrou o grupo correto para 88,6% e em (c) F3 encontrou o grupo correto para 85,3%. Pode ser visto que F2 e F3 conseguiram distinguir totalmente a classe *setosa* (representada por quadradinhos) das outras classes. Na figura 9.2 temos exemplos de agrupamento para o conjunto de dados Wine, F1 acertou em 71,9% (a), F2 encontrou o grupo correto para 70,8% (b) e F3 para 71,3% (c). Na figura 9.3, com exemplos para Glass, F1 agrupou 41,8% dos dados corretamente (a), F2 48,5% (b) e F3 40,1% (c).

Na figura 9.4, a convergência para as três funções é exibida para o benchmark Iris. Podemos ver que, como uma característica do PSO, todas possuem uma convergência rápida.

Geralmente, como podemos ver nesse trabalho, algoritmos que utilizam centróides no agrupamento, como essa versão do PSO, são capazes de encontrar *clusters* de forma esférica com bastante facilidade e robustez, porém, não bons em *clusters* com formas complexas e sobreposições.



**Figura 6-9 – Convergência do Algoritmo utilizando as 3 Funções**

## 6.3 – Conclusão

No início desse capítulo, foi apresentada a solução implementada para a avaliação do algoritmo de Clusterização e das funções de avaliação propostas. O software foi programado em c++ e permite o ajuste de vários parâmetros do algoritmo. Adiante, na seção 6.2, primeiramente são apresentados os *benchmarks* utilizados para a avaliação: *Iris*, *Glass* e *Wine*. Após isso são apresentados os parâmetros utilizados para a avaliação e são definidas as medidas para a avaliação: Taxa de acerto média, distância *Intra-cluster* e distância *Inter-cluster*. Finalmente, os resultados foram apresentados e discutidos, mostrando que embora o algoritmo tenha dificuldades em clusters com formatos mais complexos, as funções propostas trouxeram boas melhoras aos resultados.



## 7 – CONCLUSÃO E TRABALHOS FUTUROS

Nesse trabalho, foi apresentado um estudo do problema de Clusterização, uma das principais tarefas de descoberta de conhecimento em bancos de dados e aplicado em diversas áreas. Foi visto que as técnicas para a resolução do problema de *Clustering* podem ser divididas em: Métodos por Particionamento, Métodos Hierárquicos, Métodos Baseados em Densidade, Métodos Baseados em Grid, Métodos Baseados em Modelos.

Adiante, foi apresentado o PSO, *Particle Swarm Optimization*, um Algoritmo Evolutivo baseado em comportamento social que tem sido aplicado com sucesso em diversos problemas. Foram apresentados alguns métodos de resolução do problema de Clusterização baseados no PSO que tem surgido na literatura nos últimos anos. Dentre eles, o método desenvolvido por Merwe e Engelbrecht (2003), um método simples e aderente às idéias do PSO, foi implementado e seu funcionamento estudado um pouco mais a fundo. Então, foram propostas algumas modificações a função de *fitness* utilizada pelo algoritmo. Foi proposta uma função na qual o número de dados dentro do *cluster* é levado em conta na influência que esse *cluster* terá no cálculo do *fitness* da partícula. Uma outra função, que leva as partículas em direção a soluções com *clusters* mais uniformes também foi proposta.

Três *benchmarks* conhecidos da área foram usados para comparar a eficiência desses três métodos. O PSO demonstrou-se capaz de encontrar boas soluções para o problema, principalmente em *clusters* com formato circular. Por ser um Método de Particionamento baseado em centróides, existe certa dificuldade em encontrar *clusters* com formas complexas. Os resultados mostraram que melhoras significativas foram alcançadas quando as funções de *fitness* propostas foram utilizadas.

Em trabalhos futuros, pretende-se o desenvolvimento de um método de Clusterização, baseado no PSO, que seja capaz de determinar o número ótimo de *clusters* e lidar com *clusters* de formas complexas.

## 8 – REFERÊNCIAS BIBLIOGRÁFICAS

ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P.; SANDER, J. OPTICS: Ordering Points to Identify the Clustering Structure. In: ACM SIGMOD International Conference on Management of Data, 1999, Philadelphia. **Proceedings of the ACM SIGMOD International Conference on Management of Data**. New York: ACM Press, 1999.

ASUNCION, A.; NEWMAN, D.J. (2007). **UCI Machine Learning Repository** [<http://www.ics.uci.edu/~mllearn/MLRepository.html>]. Irvine, CA: University of California, Department of Information and Computer Science.  
COHEN

BÄCK, T. Optimization by Means of Genetic Algorithms. In: **36<sup>th</sup> International Scientific Colloquium**, n. 36, 1991, Technical University of Ilmenau.

BERGH, F. van der. **An Analysis of Particle Swarm Optimizers**. 2001. 300 p. Tese (PhD in the Faculty of Natural and Agricultural Science) – University of Pretoria, Pretoria.

CARLANTONIO, L. M. di. **Novas Metodologias Para Clusterização de Dados**. 2001. 257 p. Tese (Mestrado em Ciências em Engenharia Civil) – Universidade Federal do Rio de Janeiro, Rio de Janeiro.

COHEN, S. C. M.; CASTRO, L. N. de. Data Clustering with Particle Swarms. In: Congress on Evolutionary Computation, 2006. **Proceedings of IEEE Congress on Evolutionary Computation 2006 (CEC 2006)**. Vancouver: IEEE Computer Society, 2006. p. 1792- 1798.

COLE, R. M. **Clustering with Genetic Algorithms**. 1998. 110 p. Tese (Master of Science). University of Western Australia , Perth.

CUI, X.; Potok, T. E.; Palathingal, P. Document Clustering using Particle Swarm Optimization. In: Swarm Intelligence Symposium, 2005. **Proceedings of the 2005 IEEE Swarm Intelligence Symposium**. Pasadena: IEEE Computer Society, 2005. p. 185-191.

EIBEN, A. E.; SMITH, A. E. **Introduction to Evolutionary Computing**. Amsterdam: Springer, 2003. 299 p. (Natural Computing).

ESMIN, A. A. A. ; LAMBERT-TORRES, G. ; SOUZA, A. C. Z. A Hybrid Particle Swarm Optimization Applied to Loss Power Minimization. **IEEE Transactions on Power Systems**, V. 20, n. 2, p. 859-866, 2005.

ESTER, M; KRIEGEL, H; SANDER, J; XU, X A Density-Based Algorithm for Discovering Cluster in Large Spatial Databases with Noise. In: The Second International Conference of Knowledge Discovery and Data Mining, 2., 1996, Portland. **Proceedings of the Second International Conference of Knowledge Discovery and Data Mining**. Menlo Park: AAI Press, 1996.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. San Francisco: Morgan Kaufmann, 2001. 550 p.

- HUANG, Z. Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. **Data Mining and Knowledge Discovery**, v. 2., n. 3, p. 283-304, 1998.
- KARYPIS, G.; HAN E. H.; KUMAR, V. Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling. **Computer**, v. 32, n. 8, p. 38-75, 1999.
- KENNEDY, J.; EBERHART, R. C. Particle Swarm Optimization. In: IEEE International Conference on Neural Networks, 1995, Perth. **Proceedings of IEEE International Conference on Neural Networks**. 1995, p. 1942- 1948.
- KINNEAR, K. E. J. Evolution Strategies. In: BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Handbook of Evolutionary Computation**. 1. ed. Bristol: IOP Publishing, 1997.Cap. B1.5, p. 1-5.
- KOZA, J. R. Genetic Programming. **Encyclopedia of Computer Science and Technology**, v. 38, p. 29-43, 1998.
- MERWE, D. W. van der; ENGELBRECHT, A. P. Data Clustering using Particle Swarm Optimization . In: Congress on Evolutionary Computation, 2003. **Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)**, Caribella: IEEE Computer Society, 2003. p. 215-220.
- MITCHELL, M.; FORREST, S. Genetic Algorithms and Artificial Life. **Artificial Life**, v. 1, n. 3, p. 267-289, 1994.
- NG, R. T.; HAN, J. Efficient and Effective Clustering Methods for Spatial Data Mining. In: International Conference on Very Large Data Bases, 20., 1994, Santiago. **Proceedings of 20th International Conference on Very Large Data Bases**. San Francisco: Morgan Kaufmann, 1994. p. 144-155.
- OMRAM, M. G. H.; Engelbrecht, A. P.; Salman, A. Dynamic Clustering using Particle Swarm Optimization with Application in Unsupervised Image Classification. **Pattern Analysis & Applications**, v. 8, n. 4, p.332-344, nov. 2005.
- PORTO, V. W. Evolutionary Programming. In: BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Handbook of Evolutionary Computation**. 1. ed. Bristol: IOP Publishing, 1997.Cap. B1.4, p. 1-10.
- RUDOLPH, G. On Correlated Mutations in Evolution Strategies. In: Parallel Problem Solving from Nature, 2., 1992, Brussels. **Proceedings of the 2<sup>nd</sup> PPSN Conference**. New York: Elsevier Science Inc., 1992.
- RUDOLPH, G. Evolution Strategies. In: BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Handbook of Evolutionary Computation**. 1. ed. Bristol: IOP Publishing, 1997.Cap. B1.3, p. 1-6.

SAN, O. M.; HUYNH V.; NAKAMORI, Y. **An Alternative Extension of the k-Means Algorithm for Clustering Categorical Data.** International Journal of Applied Mathematics and Computer Science, Zielona Góra, v. 14, n. 2, p. 241-247, 2004.

SCHWEFEL, H.-P.; RUDOLF, G.; RUDOLPH, G. Contemporary Evolution Strategies. In: Third International Conference on Artificial Life. 3., 1995, Granada. **Proceedings of the 3<sup>rd</sup> European Conference on Artificial Life.** Berlin: Springer, 1995, 893-907.

SOUZA, T.; NAVES, A.; SILVA, A. Swarm Optimisation as a new tool for data mining. In: Parallel and Distributed Processing Symposium, 17., 2003. **Proceedings of the 17th International Symposium on Parallel and Distributed Processing.** Nice: IEEE Computer Society, 2003.

XIANG XIAO; Dow, E. R.; Erberhart, R. Miled, Z. B. Oppelt, R. J. Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. In: Parallel and Distributed Processing Symposium, 17., 2003. **Proceedings of the 17th International Symposium on Parallel and Distributed Processing.** Nice: IEEE Computer Society, 2003.

WANG, W.; YANG, J.; MUNTZ, R; STING: A Statistical Information Grid Approach for Spatial Data Mining. In: 23<sup>rd</sup> International Conference on Very Large Data Bases, 23., 1997, Athens. **Proceedings of the 23<sup>rd</sup> VLDB Conference.** San Francisco: Morgan Kaufmann, 1997.p. 186-195.