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

Osires Pires Coelho Filho
Departamento de Informática
Instituto Federal de Educação, Ciência e Tecnologia do Piaui- IFPI
Teresina-PI-Brasil
Osires123@gmail.com

Carlos Alberto Martinhon
Instituto de Computação
Universidade Federal Fluminense -UFF
Niteroi-RJ-Brasil
martinhon@ic.uff.br

Lucídio dos Anjos Formiga Cabral
Departamento de Computação Científica
Universidade Federal da Paraíba - UFPB
João Pessoa-PB-Brasil
lucidio@ci.ufpb.br

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. Para resolver este problema propomos uma nova abordagem híbrida do uso das metaheurísticas PSO (*Particle Swarm Optimization)* e ILS (*Iterated Local Search*). 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 do componente velocidade inicial da partícula. Esta versão denominada ECPSO-ILS (*Enhanced Chaotic PSO-ILS*) apresentou excelentes resultados quando testada em várias bases de dados de referência da área, obtendo novos melhores valores para todas as instâncias avaliadas.

Palavras chave: Otimização por Enxame de Partículas, Clusterização, Busca Local Iterada, Mapa Caótico

I. INTRODUÇÃO

Clusterização de Dados é uma técnica de Mineração de Dados, usada no processo de análise de dados para reconhecimento de padrões. Este processo consiste em reunir em grupos objetos com características similares. Os objetos de um mesmo grupo ou cluster devem ser o mais parecidos entre si e o mais diferente possível de objetos de outros grupos. [1]. Este processo de agrupar objetos em grupos 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 e, portanto, não são viáveis. 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 tem sido propostos [2] para resolver muitos problemas de otimização, em especial problemas de Clusterização de Dados. Um dos representantes mais populares destas técnicas e que vem obtendo bastante êxito é a meta-heurística Otimização por Enxame de Partículas ou *Particle Swarm Optimization* (PSO) [3].

Inúmeras pesquisas [4] demonstraram que o PSO padrão aplicado à Clusterização de Dados possui um desempenho superior aos demais algoritmos usados para o mesmo fim. No entanto, a taxa de convergência na busca do 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[5].

Uma das versões principais e mais exitosas variações do PSO é a variação conhecida na literatura como CPSO [8]. Nesta variação, os coeficientes de aceleração c_1 e c_2 da Equação 2, que combinados com as variáveis randômicas r_1 e r_2 controlam a influência estocástica dos componentes social e cognitivo na velocidade da partícula, são substituídos por sequências de números caóticos ou mapa caótico [9], conforme Equação 5.

Este trabalho propõe a criação de uma nova abordagem para o uso do algoritmo CPSO na Clusterização de Dados. Nesta versão, utiliza-se um número caótico para gerar o índice do vetor que guarda os valores do componente inercial *w*. A principal inovação desta variação do CPSO é a utilização de uma estratégia de aceleração utilizando outra meta-heurística, o ILS, para acelerar a convergência do CPSO. A técnica de busca local utilizada no ILS é o algoritmo *K-Means*.

Os resultados de testes feitos com a utilização de cinco bases de dados reais com diferentes problemas de Clusterização demonstram que a versão proposta é superior às encontradas na literatura, tais como *K-Means*, PSO, K-PSO, CPSO, ACPSO [8; 10; 11] e ILS-KM.

II. CLUSTERIZAÇÃO DE DADOS

Clusterização de Dados é o processo que visa reunir vetores multidimensionais em grupos, tendo em vista obter a maior similaridade dentro de cada grupo. Esta técnica também conhecida como Agrupamento de Dados é uma das mais conhecidas em Mineração de Dados. Segundo [12; 13], 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*. Um Problema de Clusterização consiste em dado uma base de dados X, agrupar (*clusterizar*) os *n* elementos ou objetos de X de modo que objetos mais similares fiquem no mesmo cluster e objetos menos similares sejam alocados para *clusters* distintos. Para medir a similaridade entre elementos de um mesmo cluster a métrica mais utilizada é a distância entre eles, quanto mais próximo mais similar e vice-versa. As

principais medidas de distância são: Distância Euclidiana, Distância de Minkowsk e Distância de Manhattan (quarteirões). Existem basicamente duas classes de Problemas de Clusterização: o caso onde o número de *clusters* não é previamente conhecido e o caso mais estudado onde o número de *clusters* já é previamente conhecido, chamado problema de k-clusterização. Num problema de k-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 equivale à função $N(n, k)$ apresentada a seguir:

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

A. Algoritmo K-Means

Uma das técnicas mais conhecidas e utilizadas para Clusterização de Dados é o *K-Means* [11; 13]. O *K-Means* usa um elemento para representar cada *cluster*. Ele é chamado de centroide e possui um valor médio para os atributos considerados, relativos a todos os elementos do *cluster*. 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 pontos mais próximos de cada centroide. Ao reposicionar os centroides os pontos então podem ficar mais próximos de outro centroide na próxima iteração e então farão parte de outro cluster.

Na versão mais básica do k-means, os centroides iniciais ou sementes, são definidos aleatoriamente, o que pode resultar numa Clusterização ruim quando as sementes ficam mal posicionadas, desvantagem esta que já foi superada em versões posteriores do k-means.

Algoritmo_*K-Means*
Início
1 Selecionar os pontos para clusterizar;
2 Inicializar aleatoriamente os centroides;
3 Repetir
4 Atribuir centroide aos pontos de acordo com a similaridade (proximidade)
5 Recalcular/reposicionar centroides;
6 Até que os pontos não mudem de centroides;
Fim_Algoritmo_kmeans;

B. Metaheurística ILS (Iterated Local Search)

O método *Iterated Local Search* (ILS) é baseado na ideia de que um procedimento de busca local pode ser melhorado gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local [10; 14]. 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. Abaixo é mostrado o algoritmo ILS básico.

Algoritmo_ILS
Início
1 Gerar solução inicial*;*
2 Fazer busca local na solução inicial;
3 Repetir enquanto os critérios de parada não forem satisfeitos
4 Realizar perturbação na solução;
5 Fazer busca local na solução perturbada;
6 Realizar critério de aceitação;
7 Fim_enquanto;
Fim_Algoritmo_ILS;

III. OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS (*PARTICLE SWARM OPTIMIZATION* - PSO)

Com base no comportamento de indivíduos que se organizam em grupos, conhecido como *Swarm Intelligence (SI)* [15; 16], Kennedy e Eberhart desenvolveram o algoritmo conhecido como *Particle Swarm Optimization - PSO* [17]. Este algoritmo segue um método estocástico de otimização. Em analogia com os paradigmas de Computação Evolucionária, um enxame é semelhante a uma população, enquanto que 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 é ajustada a cada iteração do algoritmo de acordo com a sua própria experiência e a de seus vizinhos. Cada partícula é uma solução viável para o problema e tem sua qualidade ou valor dado por uma função de *fitness* (aptidão), também chamada de função objetivo, conforme Figura 1.

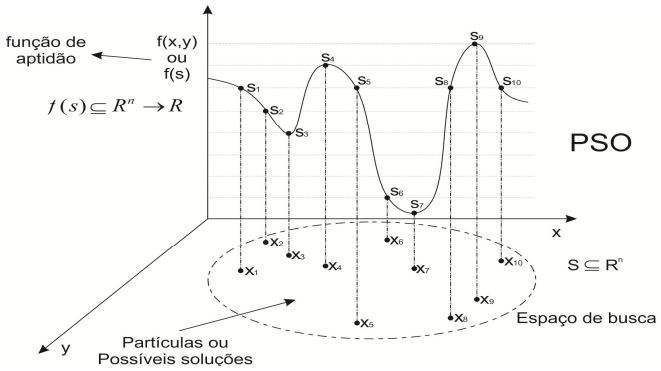

Figura 1 - Partículas = Soluções

Para um enxame de *N* partículas, a velocidade da i-ésima partícula na posição $X_i(t)$ é calculada pela equação:

$$V_i(t+1) = \underbrace{w \,.\, V_i(t)}_{\text{Velocidade prévia}} + c_1 \,.\, r_1 \,.\, \overbrace{\left(Pbest_i(t) - X_i(t)\right)}^{\text{Componente cognitivo}} + c_2 \,.\, r_2 \,.\, \overbrace{\left(Gbest - X_i(t)\right)}^{\text{Componente social}} \qquad (2)$$

onde c_1 e c_2 são constantes de aceleração, r_1 e r_2 são variáveis randômicas e $Pbest_i(t)$ e $Gbest$ são respectivamente a melhor posição que a partícula *i* já ocupou e a melhor posição já ocupada por qualquer das partículas até o momento. A variável *w*, incluída depois na fórmula, é chamada de peso inercial e serve como uma

espécie de "freio" que diminui gradativamente a velocidade da partícula a cada iteração. A nova posição da partícula é dada por:

$$\underbrace{X_i(t+1)}_{\text{Posição atual}} = \underbrace{X_i(t)}_{\text{Posição anterior}} + \underbrace{V_i(t+1)}_{\text{Velocidade atual}} \quad (3)$$

Abaixo, na Figura 2, segue a ilustração sobre a movimentação de uma partícula $X_i(t)$ num espaço bidimensional:

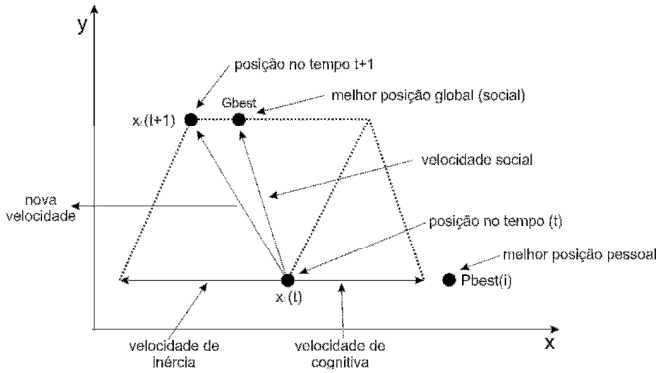

Figura 2 - Movimentação de uma partícula

O vetor 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 e *componente cognitivo* da partícula e proporciona a informação sobre a menor distância da solução ideal que aquela partícula já esteve. Esta posição é denominada como 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 vizinhança das partículas. 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. Neste trabalho vamos utilizar a versão do PSO chamada de PSO *Gbest,* ou somente PSO, como segue:

Algoritmo PSO
Início
1 Inicializar dimensão das partículas;
2 Inicializar posição das partículas X_i;{posições aleatórias no espaço de busca}
3 Inicializar velocidade das partículas em cada dimensão
4 Inicializar constantes c_1 e c_2
5 Inicializar número máximo de iterações iter_max
6 Inicializar variável de peso inercial *w*
7 Determinar função de aptidão ou função objetivo do problema
8 Determinar valor da aptidão inicial de cada partícula $Pbest_i$
9 Determinar *Gbest* inicial
10 Repetir enquanto não alcançar número máximo de iterações
11 Para cada partícula
12 Atualizar velocidade da partícula em cada dimensão
13 Atualizar posição da partícula em cada dimensão
14 Calcular nova aptidão da partícula
15 Fim_repetição
16 Atualizar *Pbest* de cada partícula
17 Atualizar *Gbest*
18 Fim_repetição
Fim_Algoritmo.

A. Variações do PSO básico

O PSO básico tem sido aplicado com sucesso a um grande número de problemas de otimização, em especial a problemas de Clusterização [18] e embora os resultados empíricos apresentados ilustrem a capacidade do PSO para resolver os problemas de otimização, estes resultados também mostram que o PSO básico tem problemas de convergência para boas soluções. Um grande número de modificações no PSO básico foram desenvolvidas para melhorar a velocidade de convergência e a qualidade das soluções encontradas pelo PSO. Essas modificações incluem a introdução de uma massa de inércia, fixação da velocidade, constrição de velocidade, diferentes maneiras de determinar as posições do *personal best* e *global best*, e diferentes modelos de velocidade.

B. Chaotic Particle Swarm Optimization - CPSO

Uma das muitas versões do PSO conhecida como CPSO [8], usa um mapa caótico ou sequência de números caóticos para substituir os componentes randômicos $r_1 e\ r_2$ da Equação 2. 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 [19], que gera números caóticos usando a seguinte fórmula:

$$\mathrm{Cr(t+1)} = \mathrm{k} * \mathrm{Cr(t)} * (1 - \mathrm{Cr(t)}) \quad (4)$$

No campo da engenharia, é bem reconhecido que a Teoria do Caos pode ser aplicada como uma técnica muito útil em aplicações práticas [6]. O Sistema Caótico pode ser descrito como um sistema não-linear limitado com comportamento dinâmico determinista que tem propriedades ergódicas e estocásticas [7]. Matematicamente, uma variável caótica é aleatória e imprevisível, mas também possui um elemento de regularidade [9]. Em [20] 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 3, 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" pequenos possibilitando um maior refinamento da pesquisa.

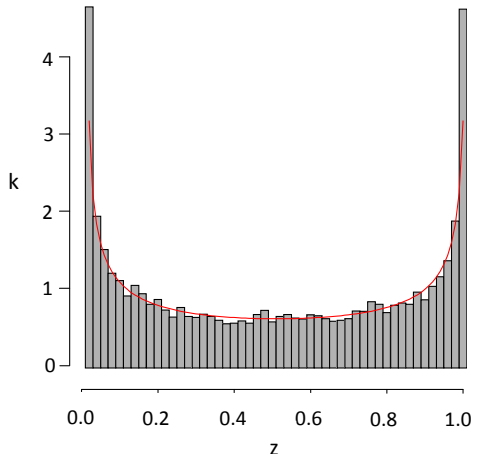

Figura 3 - Histograma de 10.000 iterações do mapa

Na Equação 4, 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.

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 aos números randômicos fica então da seguinte forma:

$$V_i(t+1) = w_i(t) * V_i(t) + c_1 * Cr(t) * (Pbest_i - X_i(t)) + c_2 * (1 - Cr(t)) * (Gbest - X_i(t)) \quad (5)$$

IV. Enhanced Chaotic Particle Swarm Optimization-Iterated Local Search - ECPSO-ILS

No ECPSO-ILS, utilizado para Clusterização de Dados, cada partícula é uma solução em potencial que representa o conjunto dos k centroides necessários para resolver o problema. Cada partícula X_i é construída como: $X_i = (m_{i1}, \ldots, m_{ij}, \ldots, m_{iNc})$, onde m_{ij} refere-se ao j-ésimo elemento do vetor de centroides da i-ésima partícula. A dimensão de cada partícula, D, é dada pela multiplicação do número de dimensões de cada objeto, *d*, pelo número de centroides ou *clusters*, *k*, ou seja, $D = k * d$.

Na variação do PSO proposta neste trabalho, procuramos melhorar ainda mais o poder do algoritmo CPSO. Propomos usar um peso inercial *w* com um componente caótico para termos os mesmos benefícios obtidos com o comportamento caótico dos componentes *Gbest* e *Pbest* da equação de velocidade original do algoritmo CPSO. Nesta abordagem o índice do vetor *w* é determinado por um número caótico o que faz com que a componente velocidade atual da equação de velocidade tenha um comportamento caótico. A equação de velocidade da versão proposta fica da seguinte forma:

$$V_i(t+1) = w\_c_i(t) * V_i(t) + c_1 * Cr(t) * (Pbest_i - X_i(t)) + c_2 * (1 - Cr(t)) * (Gbest - X_i(t)) \quad (6)$$

Onde w_c_i representa o vetor de peso inercial aplicado à velocidade da partícula, que nesta versão não decresce de forma constante dentro de uma faixa de valores pré-definidos e sim de acordo com o mapa logístico da Equação 4, ou seja, a cada iteração um valor de peso inercial é escolhido do vetor de pesos inerciais de forma caótica.

Na estratégia de aceleração utilizada neste algoritmo utilizamos uma chamada ao algoritmo ILS, na primeira iteração do PSO. O ILS é chamado passando-se os centroides da partícula com melhor fitness da primeira iteração do PSO. No algoritmo ILS utilizado nesta estratégia, que desenvolvemos com o nome de ILS-PSO (ver Tabela 2, fazemos com que na busca local seja executado o algoritmo *K-Means* padrão para refinar os centroides recebidos do PSO. A perturbação utilizada consiste na aplicação de uma pequena variação no valor de cada dimensão dos centróides.

O fato de chamarmos o ILS após a primeira iteração do PSO já fornece bons centroides para a busca local realizada pelo *K-Means* no ILS. Este procedimento faz com que o ILS já comece sua pesquisa numa área promissora do espaço de busca e retorne centroides melhorados para compor a partícula a ser devolvida para o PSO. Esta partícula é de excelente qualidade pois o algoritmo ILS-KM por sí só já tem excelentes resultados, conforme mostrado na Tabela 2, restando ao PSO apenas o refinamento ainda maior das partículas, possibilitando assim que o algoritmo chegue a uma convergência rápida para uma solução de alta qualidade. O procedimento de chamada ao ILS não precisa ser feito em todas as iterações do PSO, pois se assim o fosse o ILS seria executado desnecessariamente, aumentando muito o custo computacional, pois não conseguiria trazer resultados melhores que os do PSO, ou seja, a chamada do ILS só melhora a partícula quando ela ainda encontra-se no início das iterações.

Abaixo o algoritmo para o método ECPSO-ILS proposto para a Clusterização de Dados.

Algoritmo ECPSO-ILS;
Início
1. Inicializar dimensão da partícula *D* e número de *clusters k*;
2. Inicializar posição das partículas;
3. Inicializar velocidade das partículas
4. Inicializar parâmetros: $w\_c, c_1 \; e \; c_2$
5. Determinar aptidão das partículas $Pbest_i$ e $Gbest$ inicial
6. Inicializar aleatoriamente o número caótico *Cr*
7. Enquanto Iter_PSO <= $(10 * k * d)$ faça
8. Calcular o índice caótico do vetor w_c
9. Inicializar aleatoriamente o número caótico *Cr*
10. Para cada partícula
11. Calcular para cada dimensão o número caótico *Cr* de acordo com a Eq. 4
12. Atualizar velocidade da partícula em cada dimensão usando Eq. 6
13. Atualizar posição da partícula em cada dimensão usando Eq. 3
14. Calcular nova aptidão da partícula
15. Fim_para
16. Se Iter_PSO = 1 então
17. Chamar ILS passando a partícula com melhor fitness

18. Atualizar melhor partícula do ECPSO-ILS caso tenha tido melhora no ILS
19. Fim_se.
20. Atualizar $Pbest_i$
21. Atualizar $Gbest$
22. Fim Enquanto
Fim_Algoritmo.

V. RESULTADOS COMPUTACIONAIS

O algoritmo proposto foi implementado utilizando a linguagem de programação MatLab versão 7.11.0 (R2010b). De modo a demonstrar a eficiência do ECPSO-ILS, foram comparados os resultados obtidos com os seguintes algoritmos: *K-Means*, PSO, K-PSO, ACPSO e CPSO. O ECPSO-ILS foi executado um total de 20 vezes para cada uma das cinco bases de dados de referência constantes na Tabela 1. As medidas reportadas são média aritmética, desvio padrão e melhor resultado obtido em 20 execuções seguidas.

A. Bases de dados

O experimento consiste em aplicar a versão do ECPSO-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 são descritas as características das bases de dados utilizadas.

Tabela 1: Características das bases de dados

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
Vowel	6	3	871

1. A base de dados *Vowel* (vogal) é constituído 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. 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. 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 características demográficas e sócio-econômicas.
4. A base de dados *Breast Cancer* (Câncer 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. A base de dados *Wine* (vinho) consiste em 178 objetos identificados por 13 características diferentes obtidas por análise química dos vinhos que são produzidos na mesma região, na Itália, mas são derivados de três cultivos diferentes.

B. Métodos e parâmetros de configuração

A quantidade de partículas que são usadas no experimento depende de cada base de dados a ser clusterizada. Depende das características dos objetos e do número de *clusters*. São inicializadas aleatoriamente N partículas, N é dado por *3*D*, onde *D* é a dimensão de cada partícula e é dado por: $\mathrm{D} = \mathrm{k} * \mathrm{d}$. Onde *k* representa o número de cluster a ser informado previamente e *d* representa o número de dimensões ou características de cada objeto. Cada partícula representa os *k* centroides da solução e recebem inicialmente valores aleatórios entre os máximos e mínimos valores de cada dimensão dos objetos a serem clusterizados. O número máximo de iterações realizadas pelo ECPSO-ILS também é dependente dos objetos e do número de cluster e é dado por $\mathrm{Max_{iter}} = 10 * \mathrm{k} * \mathrm{d}$. Os coeficientes de aceleração, constantes c_1 *e* c_2, combinados com a variável caótica Cr, controlam a influência estocástica dos componentes social e cognitivo na velocidade da partícula. Os valores de c_1 *e* c_2 são definidos como 1.5 e 2.0, respectivamente. O valor de c_2 maior que c_1 faz com que as partículas tenham a tendência a se aproximarem um pouco mais rapidamente em direção a um ótimo global que geralmente retorna da chamada do ILS.

O peso inercial, vetor *w_c,* é inicializado com valores entre *0.1 e 0.75*. A cada iteração um valor de *w_c* é escolhido de acordo com o valor do número caótico *Cr* (Eq. 4) calculado para cada iteração, desta forma *w_c* pode assumir valores grandes ou pequenos, possibilitando assim que a partícula possa fazer "vôos" mais longos e possa fugir de um ótimo local, mesmo no final das iterações.

O valor de fitness utilizado é a soma das distâncias intra-cluster. Quanto menor a soma das distâncias, melhor o resultado da clusterização. O valor de aptidão de cada partícula pode ser calculado com a seguinte função:

$$\text{fitness} = \sum_{i=1}^{k} \sum_{j=1}^{n} (X_i - Z_j) \quad (7)$$

Na Eq. 7, *k* e *n* são os números de *clusters* e de objetos do conjunto de dados respectivamente. Z_j é o valor do centroide j e X_i é o valor do objeto de dados i.

A medida utilizada para determinar a pertinência de um objeto a um cluster foi a menor distância euclidiana entre o objeto e um dos centroides.

O ILS é chamado somente na primeira iteração do PSO para acelerar a convergência. Ele recebe os centroides contidos na melhor partícula da primeira iteração do PSO e os utiliza como solução inicial. O número de iterações utilizado no ILS é 3 vezes o número de iterações do PSO. A perturbação é feita mudando-se a posição de um ou mais centroides da partícula ao mesmo tempo, num percentual aleatório entre 0 e 5 por cento do valor original e então é

executado o *K-Means* como busca local para calcular a nova distância intra-cluster dos centroides perturbados.

C. Resultados experimentais e discussões

O algoritmo proposto foi implementado utilizando a linguagem de programação MatLab versão 7.11.0 (R2010B). De modo a demonstrar a eficiência do ECPSO-ILS, foram comparados os resultados obtidos com os seguintes algoritmos: *K-Means*, PSO, K-PSO, ACPSO e CPSO conforme Tabela 2. O ECPSO-ILS foi executado um total de 20 vezes para cada base de dados utilizada. As medidas reportadas são média aritmética, desvio padrão e melhor resultado obtido em 20 execuções seguidas.

Tabela 2: Resultados computacionais

Base	Medida	*K-Means*	PSO	K-PSO	CPSO	ACPSO	ILS-KM	ECPSO-ILS
Vowel	Média	159242,87	168477,00	149375,70	151337,00	149051,84	149373,37	**148981,32**
	Desvio	916	3715,73	155,56	3491,43	67,27	5,64	**34,31**
	Melhor	149422,26	163882,00	149206,10	148996,50	148970,84	149369,63	**148967,24**
Iris	Média	106,05	103,51	96,76	96,90	96,66	96,65	**96,65**
	Desvio	14,11	9,69	0,07	0,303	0,001	0,00	**0,00**
	Melhor	97,33	96,66	96,66	96,66	96,66	96,65	**96,65**
CMC	Média	5693,60	5734,20	5532,90	5532,23	5532,20	5542,18	**5532,18**
	Desvio	473,14	289,00	0,09	0,04	0,01	0,00	**0,00**
	Melhor	5542,20	5538,50	5532,88	5532,19	5532,19	5542,18	**5532,18**
Câncer	Média	2988,30	3334,60	2965,80	2964,49	2964,42	2986,96	**2964,38**
	Desvio	0,46	357,66	1,63	0,12	0,03	0,00	**0,00**
	Melhor	2987	2976,30	2964,50	2964,40	2964,39	2986,96	**2964,38**
Wine	Média	18061,00	16311,00	16294,00	16292,90	16292,31	16555,67	**16292,18**
	Desvio	793,21	22,98	1,70	0,78	0,03	0,00	**0,00**
	Melhor	16555,68	16294,00	16292,00	16292,19	16292,18	16555,67	**16292,18**

Os resultados de *K-Means*, PSO, K-PSO, CPSO e ACPSO podem ser encontrados em [8] bem como os parâmetros utilizados.

Para o ILS-KM, mostrado na Tabela 2 acima, foi desenvolvido um algoritmo para compará-lo com o ECPSO-ILS proposto aqui, onde a busca local do ILS é o algoritmo *K-Means* padrão. Este algoritmo obteve excelentes resultados quando comparado com os outros da Tabela 2. O número de iterações utilizado neste algoritmo foi: $Iter\_\max_{ils} = 3 * (10\ x\ k\ x\ d)$, onde k indica o número de *clusters* e d a dimensão de cada cluster. Isto dá um número três vezes maior que o número de iterações do ECPSO-ILS. Neste ILS a perturbação utilizada é a movimentação de um ou mais centroides da solução num valor percentual que varia aleatoriamente entre 1 e 5 por cento do valor do centroide original.

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

VI. CONCLUSÕES

Neste trabalho foi apresentada uma nova versão hibridizada do algoritmo PSO com ILS e *K-Means* para resolver o problema da Clusterização de Dados chamado de ECPSO-ILS. Nesta versão foram introduzidas duas modificações importantes em relação às hibridizações anteriores: na primeira utilizou-se um número caótico para gerar o valor do índice do peso inercial e na segunda foi inserida uma estratégia de aceleração utilizando a metaheurística ILS com uma busca local como sendo o algoritmo *K-Means* padrão, que possibilitou uma convergência mais rápida do algoritmo. Os resultados foram testados utilizando-se cinco bases de dados de referência e comparados com o desempenho de seis algoritmos utilizando a mesma métrica, ou seja, minimização da distância intra-cluster, para pesquisar centroides ideais num espaço euclidiano n-dimensional. Verificou-se que o ECPSO-ILS chega a resultados melhores que outros algoritmos utilizados como comparação e com um menor custo computacional. Pretende-se ainda, como trabalhos futuros, utilizar este algoritmo em aplicações práticas como a segmentação de imagens médicas.

REFERÊNCIAS

[1] Han, J., Kamber, M., & Tung, A. K. H. (2001). *Spatial clustering methods in data mining: A survey*. London: Taylor & Francis.

[2] 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.

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

[4] 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.

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

[6] Coelho, L. d. S., & Mariani, V. C. (2009). *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.

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

[8] 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.

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

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

[11] Bandyopadhy, S., & Maulik, U. (2002). *An evolutionary technique based on K- Means algorithm for optimal clustering in RN*. Information Science, 146, 221–237.

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

[13] Mitra S. and Acharya T., 2004. Data Mining. Wiley Publications.

[14] Souza, M. J. F. *Inteligência Computacional para Otimização.* Notas de aula. Departamento de Computação, Universidade Federal de Ouro Preto, 2011.

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

[16] Kennedy, J., R.C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 2001.

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

[18] Omran, M., Engelbrecht, A. P., & Salman, A. (2005). *Particle Swarm Optimization method for image clustering.* International Journal on Pattern Recognition and Artificial Intelligence, 19, 297–322.

[19] May, R. M. (1976). Simple mathematical models with very complicated dynamics. Nature, 261, 459–467.

[20] 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.