ALGORITMO BIOBJETIVO BASEADO EM ENXAME DE PARTÍCULAS APLICADO À CLASSIFICAÇÃO DE DADOS HÍBRIDOS

ANA C.M.LAURENTINO[1], JOAO B.MENDES[2], MARCOS F.S.V.DANGELO[3], GUILHERME B.VILELA[4]

[2,3,4]*Departamento de Ciências da Computação - Unimontes*
[1]*Programa de Pós-Graduação em Modelagem Computacional e Sistemas - Unimontes*
Avenida Dr. Ruy Braga, S/N - Vila Mauriceia, Montes Claros - MG, 39401-089
Emails: anaclaudiaml2000@gmail.com, joao.mendes@unimontes.br, marcos.dangelo@unimontes.br, guilherme.vilela@unimontes.br

Abstract_ A version of multiobjective Particle Swarm Optimization (PSO) algorithm for classification of hybrid data is presented. The proposed algorithm, called hPSO in this paper, implements a routine for generating the initial swarm and a mechanism for perturbate the particles. Each funcionality developed by hPSO is described individually. The obtained results indicated that the proposed algorithm is promissor for hybrid data classification.

Keywords_ Multiobjective PSO, data mining, hybrid data classification

Resumo_ Uma versão multiobjetivo do algoritmo PSO é apresentada para classificação de dados híbridos. O algoritmo proposto, chamado, neste trabalho, de hPSO, implementa uma rotina para geração do enxame inicial e um mecanismo para perturbação das partículas. Cada funcionalidade desenvolvida pelo hPSO é descrita individualmente. Os resultados obtidos indicam que o algoritmo proposto é promissor para classificação de dados híbridos.

Palavras-chave_ PSO multiobjetivo, mineração dados, classificação de dados híbridos

1 Introdução

Os repositórios de dados são, nos dias atuais, importantes ativos para empresas do setor público ou privado. O número e o tamanho dos repositórios crescem com o passar do tempo, assim como, se diversificam os dados que neles são armazenados, tornando-se cada vez mais comum o armazenamento de imagens, dados espaciais, entre outros.

Produzir informações relevantes, além das que são explícitas, não é uma tarefa fácil, pois em meio a tantos dados, diversas relações e padrões de comportamento podem não ser percebidos sob uma análise comum. A mineração de dados possui papel fundamental na exploração de grandes conjuntos de dados.

A mineração de dados pode ser subdividida quanto às suas funcionalidades, que são: classificação, estimação, predição, entre outras. No contexto da mineração de dados, a classificação é um recurso importante no estabelecimento de percepção de padrões de relação e comportamento entre os dados. Dentro do escopo desta funcionalidade este trabalho foi proposto.

O objetivo deste trabalho é propor um método para classificar dados em grandes repositórios, compostos por dados convencionais (numéricos e textuais) e/ou dados espaciais (ponto, linha, polígono). A opção por trabalhar com bancos de dados híbridos se deu em função do crescente uso de técnicas de geoprocessamento, que resultam em armazenamento de dados que são representações do espaço.

O algoritmo proposto, Otimização por Enxame de Partículas Híbrido - *h*PSO (*Hybrid Particle Swarm Optimization*), consiste em uma abordagem multiobjetivo do algoritmo de Otimização por Enxame de Partícula - PSO. O algoritmo busca gerar partículas, que contêm regras para classificação dos dados, atendendo aos seguintes objetivos: maximizar (qualidade) x minimizar (complexidade).

Esta redação está organizada da seguinte forma: seção 2 faz uma revisão literária sobre conceitos e trabalhos relacionados; a seção 3 contempla a formulação do problema e uma descrição detalhada do algoritmo *h*PSO, proposto neste trabalho, para classificação de dados híbridos; na seção 4 analisa-se o desempenho do algoritmo proposto e, por último, na seção 5, apresentam-se conclusões acerca do trabalho realizado.

2 Revisão Bibliográfica

Nesta seção, estão relacionados estudos e conceitos que fundamentaram a construção deste trabalho.

A mineração de dados é recurso fundamental na descoberta de conhecimento não explícitos de grandes repositórios de dados. Segundo (Berry e Linoff, 1997), minerar dados é explorar e analisar

grandes conjuntos de dados, com objetivo de descobrir padrões de comportamento relevantes.

A manipulação dos conjuntos de dados é viabilizada pelos Sistemas Gerenciadores de Bancos de Dados - SGBD. Segundo (Elmasri e Navate, 2005), um SGBD é composto por um conjunto de programas que permite criar e manter um banco de dados, facilitando a definição, construção, manipulação e compartilhamento dos dados entre diferentes aplicações e usuários.

Conforme (INPE, 2000), os fenômenos do mundo podem ser representados por diferentes tipos de dados, como: dados temáticos, dados cadastrais, dados de redes, dados de modelos numéricos e dados de imagens, cujos exemplos são respectivamente: dados de solo, dados de cadastro urbano e rural, dados de rede de esgoto e logradouros, dados geofísicos e topográficos e dados fotos áreas.

O crescente tamanho dos repositórios e o aumento das variedades de dados armazenados, requer, cada vez mais, recursos e novas técnicas para exploração e extração de conhecimento.

Neste contexto, surgiram alguns trabalhos como:

- O Algoritmo Multiobjetivo Baseado na Programação Genética e Nichos para Classificação de Dados Híbridos - MDMGeo foi proposto por (Pereira, 2012), para classificação de dados híbridos;
- O *m*DPSO proposto por (Prates, 2017) que consiste em uma adaptação do PSO, que trabalha com a otimização por enxame de partículas, para classificação de bases de dados convencionais.

3 Desenvolvimento

A função objetivo do problema abordado neste trabalho compreende otimizar os objetivos: maximizar a efetividade e minimizar a complexidade das regras. Para fins de implementação, a função pode ser escrita como a maximização de dois objetivos: a efetividade e o inverso da complexidade; disposta na Equação 1, conforme (Pereira, 2012).

$$\textbf{Maximizar} = F(x) = |f_1(I,X), f_2(I,X)| \quad (1)$$

A complexidade de uma partícula diz respeito ao número de termos que compõem a regra associada à partícula. Em relação a este valor, o objetivo é minimizá-lo, mas pode ser traduzido na maximização de seu inverso, que é obtido pela Equação 2.

$$f_1(I,X) = 1 / (\textit{número de termos}) \quad (2)$$

A efetividade de uma regra, corresponde ao seu valor de *fitness,* que é obtido através da Equação 3. O seu valor indica o quão boa é a regra para classificação dos dados de uma classe específica.

$$f_2(I, X) = \textit{sensibilidade x especificidade} \quad (3)$$

3.1 *h*PSO - Uma versão multiobjetivo do PSO para classificação de dados híbridos

O *h*PSO é uma adaptação do algoritmo *m*DPSO (Prates, 2017) que é uma variante multiobjetivo do PSO para tratamento de dados convencionais (contínuos).

O *h*PSO foi projetado para otimização de problemas biobjetivos envolvendo bases de dados híbridas - compostas por dados convencionais e espaciais. São características principais do *h*PSO:

- Inserção dos operadores geográficos na construção das sentenças SQL. Esta funcionalidade permite o tratamento de bases de dados que contemplam informações georeferenciadas.
- Novo mecanismo para geração do enxame inicial;
- Adição de novo operador de mutação.

Também foi introduzido mecanismo de validação para garantir a corretude (sintaxe SQL) das regras geradas pelos operadores implementados pelo *h*PSO. O pseudocódigo do *h*PSO é apresentado no Algoritmo 1 e as rotinas de criação do enxame inicial e mutação de classes, estão detalhadas nas subseções 3.3 e 3.4, respectivamente.

Algoritmo 1 *h*PSO

Entrada: ω - taxa de mutação; c1 - influência do *PBest*; c2 - influência do *GBest*; *T* - Tamanho do enxame; *Q (K)* - quantidade de classes;
E ← Criar enxame inicial (K, T) (Ver subseção 3.3)
enquanto critério de parada não atingido **faça**
  para cada partícula $p \in$ E **faça**
    Avaliar partícula *p*;
    Atualizar repositórios *PBest* e *GBest*;
    Aplicar operador de turbulência; (Ver subseção 3.4)
    Atualizar posição *p*;
  fim para
  Aplicar busca local nas partículas de *GBest;*
fim enquanto

3.2 Representação do Indivíduo

O indivíduo ou partícula do *h*PSO é composto por um vetor, de tamanho variável, que armazena os

termos que compõe a sua regra (ilustrado na Figura 1) e um rótulo que identifica sua classe.

Na avaliação da regra associada à uma partícula, o vetor de termos é utilizado na construção de um predicado lógico da cláusula WHERE de uma sentença SQL. A sentença resultante é executada por um servidor de banco de dados.

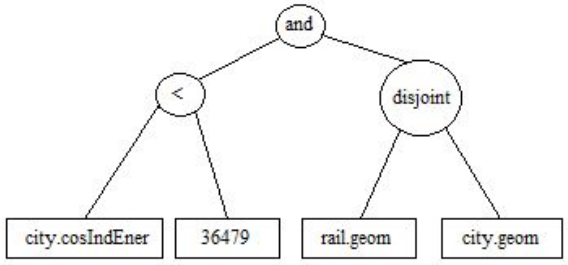

Figura 1. Representação de um indivíduo

A construção dos termos no *h*PSO, semelhante à (Prates, 2017), utiliza os operadores relacionais ("=", "not", "=", "<", ">", ">=", e "<=") acrescidos, neste trabalho, dos operadores espaciais: "*disjoint*", "*touches*", "*overlaps*", "*equals*", "*intersects*", "*crosses*", "*within*" e "*contains*", cujas funcionalidades são descritas em (Pereira, 2012). Os termos compostos por operador convencional podem apresentar duas estruturas: <*atributo, operador, valor numérico*> ou <*atributo, operador, atributo*>. Termos associado a um operador espacial utilizam estrutura do tipo <*operadorGeografico (atributo, atributo)*>, na qual é obrigatório o uso de apenas atributos geográficos.

Dada a formulação biobjetivo do problema trabalhado, os repositórios *gbest* e *pbest* armazenam soluções não dominadas, encontradas para a classe e pela partícula, respectivamente. O gerenciamento do repositório *gbest* e *pbest* é similar ao implementado por (Prates, 2017).

3.3 Criação do Enxame Inicial

A rotina proposta para geração do enxame inicial aplica o conceito de nicho, que otimiza o funcionamento do algoritmo possibilitando a busca por regras para diferentes classes do problema. Os nichos são subpopulações cujo objetivo é resolver subdivisões do problema proposto, encontrando ótimos locais para os subproblemas.

Na abordagem proposta, foi criado um nicho N_K para cada classe C_K. Todos os nichos devem possuir número igual de partículas, cujo valor é definido pela divisão do tamanho total da população T, pela quantidade de classes κ do problema.

O mecanismo de geração do do enxame inicial implementado pelo hPSO, detalhado no Algoritmo 2, compreende duas etapas:

- *1ª etapa:* Gera-se um total de 2*T partículas. Nesta etapa, para a atribuição do rótulo ou classe, a aptidão do indivíduo é considerada. Assim, a partícula é rotulada com a classe com o maior número de registros identificado pelo conjunto de regras da partícula e inserida na lista de sua classe L_K. Caso uma regra retorne um número de registros iguais, para diferentes classes, a escolha da classe rótulo será feita de forma aleatória. Ao final, as listas L_K são ordenadas por ordem decrescente de *fitness* das mesmas. Os nichos N_K de cada uma das classes recebem partículas da lista de L_K de mesma classe. A transferência ocorre até que as partículas de L_K se encerrem, ou, que o nicho N_K esteja completo, tenha atingido seu tamanho máximo.
- 2ª *etapa*: todos os nichos N_K que não tiverem alcançado sua população total, nesta etapa, obrigatoriamente recebem partículas até que a população total seja completada. Mas, diferentemente da 1ª etapa, a aptidão do indivíduo não é considerada para atribuição do rótulo. As partículas são criadas e recebem como rótulo a classe do nicho que está sendo preenchido.

O objetivo desta rotina é possibilitar que os nichos recebam partículas adequadas aos seus subobjetivos, facilitando o processo de convergência do algoritmo.

A opção por não criar toda a população com apenas partículas aptas às classes de seus rótulos, se deu em função de quão lento seria esse procedimento, visto que, as consultas espaciais já são custosas e muitas partículas seriam descartadas por serem aptas a classes cujos nichos estão preenchidos.

Algoritmo 2 criarEnxameInicial()

Entrada: *T* - Tamanho da população; *E* - enxame; *K* - número de termos da classe; *Q* - quantidade de classes;

E ← ∅;
para cada classe C_K do problema **faça**
  C_K.gbest ← ∅;
  L_K ← ∅;
fim para
enquanto n ← até (2 x T) **faça**
  t ← rand(1, K);
  p.*regra* ← Criar regra de *t* termos;
  p.*classe* ← Calcular e atribuir classe de aptidão;
  p.*pbest* ← ∅;
  se p.*classe* != ∅ **então**

```
    Lκ ← Inserir p na lista Lκ, onde κ igual a p.classe;
  fim se
fim enquanto
para cada classe Cκ do problema faça
  Lκ ← Ordenar partículas em ordem decrescente de fitness;
fim para
para cada lista Lκ faça
  para cada p ← em Lκ faça
    se tamanho do nicho Nκ < (T / Q) então
      Nκ ← p
    fim se
  fim para
fim para
para cada Nκ faça
  enquanto tamanho do nicho Nκ < (T / Q) faça
    t ← rand(1, K);
    p.regra ← Criar regra de t termos;
    p.classe ← κ
    p.pbest ← ∅;
    Nκ ← p;
  fim enquanto
  E ← Adicionar nicho Nκ ao enxame;
fim para
retorna E
```

Em todas as fases, o processo de criação dos termos das regras é feito através da seleção de um operador e atributos adequados, aleatoriamente. Para cada um dos termos da regra, a seleção do operador é feita através do método da roleta proposto por (Goldberg, 1989), respeitando os pesos dispostos nas Tabela 1 e Tabela 2, para operadores convencionais e espaciais, respectivamente.

Os pesos foram definidos observando as probabilidades propostas para os operadores convencionais, por (Prates, 2017), adaptando-os ao problema deste trabalho. Para os operadores espaciais, os pesos foram definidos através da análise dos operadores pertinentes as geometrias presentes na base utilizada, e ainda, testes experimentais. Os operadores com maior relevância ao problema, dos convencionais e espaciais, foram definidos com a mesma probabilidade, para que as chances das regras possuírem ambos os tipos de operadores fossem iguais.

Tabela 1. Pesos para seleção de operadores convencionais.

Operadores	>	>=	<	<=	!=	=
Pesos	0.98	0.98	0.98	0.98	0.20	0.20

Tabela 2. Pesos para seleção de operadores espaciais.

Operadores	Pesos	Operadores	Pesos
disjoint	0.98	intersects	0.98
touches	0.66	crosses	0.66
within	0.44	overlaps	0.44
equals	0.20	contains	0.20

3.4 Mutação de Classes

Os operadores de mutação, propostos por (Prates, 2017), no *m*DPSO, adição de termo, mudança de operador e alteração de valor de atributo, foram concentrados no operador de turbulência, que basicamente, define a probabilidade de cada um deles ocorrer. Neste trabalho, os operadores citados foram mantidos com as devidas adaptações para trabalharem com operadores espaciais.

A mutação por classes, proposta por este trabalho, foi acrescentada ao operador de turbulência, com o objetivo de melhorar a diversidade dos nichos, através da troca de material genético. O operador promove a troca das partículas que não são aptas ao nicho que se encontra, seja por ter sido uma partícula de origem da segunda etapa de geração do enxame inicial, ou, porque ao longo do processo de evolução deixou de ser apta a subpopulação que pertence.

O operador de turbulência controla a ocorrência dos operadores de mutação, para isso, divide a população em três grupos de tamanho igual, *C*1 , C2 e C3. A definição das partículas de cada grupo é dada através de uma escolha aleatória, evitando que partículas sejam encaminhadas sempre ao mesmo operador de mutação.

Um terço das partículas, C1, serão encaminhadas para o operador de alteração de valor do atributo, que realiza mutação gaussiana, ou, uniforme, que possuem probabilidade igual de serem aplicadas. As partículas enviadas a este operador, sofrerão sua mutação se possuírem atributo numérico. Caso possuam não possuam, será utilizado o operador adicionar de mutação que adiciona um atributo.

As partículas do segundo terço, C2, recebem a adição de um novo termo, ou, mutação de operador, com 50% de chance cada. A primeira possibilidade consiste na adição de um termo gerado e acrescentado a regra. A segunda, implica em uma troca de operador de um termos, selecionado aleatoriamente.

Das partículas do terceiro terço, C3, com probabilidade de 50%, podem sofrer nenhuma mutação, ou, serem responsável por dispararem a mutação de classes. O fato de uma partícula disparar a mutação por classes, diferentemente dos demais operadores de mutação, não significa que a mutação será nela aplicada.

A mutação por troca de classes é único operador que promove a troca de material genético entre as subpopulações. O seu objetivo é através dessa troca, melhorar a diversidade de toda a população. No Algoritmo 3, apresenta-se o seu pseudocódigo, projetado, para problemas com três classes, conforme base de dados que será aplicada nos experimentos.

Algoritmo 3 mutarClasses()

```
para cada nicho Nκ selecionado aleatoriamente faça
    Ordenar partículas de Nκ por fitness decrescente;
    para p ← Nκ até 3 faça
        p1 ← Clonar p;
        p2 ← Clonar p;
        p1.classe ← Recebe classe de outro Nκ;
        p2.classe ← Recebe classe de outro Nκ;
        Avaliar p1;
        Avaliar p2;
        se p1.fitness >= p2.fitness && partículas
recebidas do nicho Np1.classe r =< 3 então
            Np1.classe ← p1;
        se não
            Np2.classe ← p2;
        fim se
    fim para
fim para
```

A mutação de classes consiste na seleção de três indivíduos com menor *fitness*, de cada uma das classes, para serem migrados para as demais populações. A escolha pelos indivíduos de menor *fitness*, se deu pela percepção, que especialmente ao longo da evolução das partículas, algumas podem tornar-se mais aptas as outras subpopulações, apresentando assim, uma baixo *fitness* no nicho em que se encontram.

Os três piores indivíduos de cada nicho, serão obrigatoriamente alocados em outra subpopulação. Exemplificando, em um problema com nichos A, B e C, os indivíduos de A deverão migrar para B e/ou C, os de B para A e/ou C e os de C para A e/ou B. Ao final das migrações todos as subpopulações deverão apresentar o mesmo tamanho.

A rotina da mutação por classes foi definida através da seleção aleatória de cada um dos nichos, evitando que a ordem das classes seja sempre a mesma. Com o nicho selecionado, cada uma das suas três piores partículas, são clonadas, e recebem o rótulo dos demais nichos e, é então, avaliada. A partícula será migrada prioritariamente para o nicho cuja avaliação com seu rótulo tiver apresentado melhor *fitness*. Se o nicho receptor já tiver recebido três partículas, obrigatoriamente a partícula será migrada para o terceiro nicho. Caso a avaliação da partícula apresente o mesmo *fitness*. Utilizando a analogia dos nichos A, B e C, supondo que o nicho C foi o primeiro sorteado. Então, cada uma das suas três piores partículas são avaliadas com a classe dos nichos A e B. Caso a partícula apresente melhor *fitness* para a classe A, será migrada para o nicho A, desde que o nicho não tenha recebido três partículas. Se houver, a partícula será encaminhada para o nicho B, independente do *fitness*.

4 Resultados

Para experimentação do algoritmo proposto, foi utilizada a base de dados disponibilizada no repositório (Geominas, 2010), chamada de "Desenvolvimento Urbano"'. A base em questão traz informações sobre o desenvolvimento urbano de 852 cidades cujas classificações são: 264 "alto", 296 "médio" e 292 "baixo".

A base é composta por atributos numéricos e geográficos, que trazem informações como quantidade de escolas, índice GINI, entre outros, armazenados como números e informações espaciais dos municípios (polígonos), ferrovias e rodovias (linhas).

O *h*PSO foi desenvolvido utilizando a linguagem Java, os dados utilizados ficaram armazenados no banco de dados *PostgreSQL* 9.1, instalado com o plugin *Postgis* 1.5.3, que é responsável pela importação das funções e configurações necessárias para armazenar e manipular bases de dados geográficas. Nos experimentos foi utilizado um computador Intel (R) Core (TM) i3-3217U CPU 2.2GHz, com 4.00GB de memória RAM e sistema operacional Windows 7 64-bits.

No algoritmo, foram mantidos os parâmetros de configuração propostos por (Prates, 2017), com os valores valores 0.9, 0.8 e 0.8 para ω, c1 e c2, respectivamente, população de tamanho 90 e 30000 avaliações como critério de parada.

O experimento contou a validação cruzada 10-*fold* (Cohen, 1995), ou seja, 10 partições sem elementos repetidos, definidos randomicamente, para treinamento e teste dos indivíduos.

Foram realizadas 5 execuções do algoritmo, gerando um total de 50 resultados, 1 para cada partição, assim como nos experimentos de (Pereira, 2012), com os quais os resultados obtidos, pelo *h*PSO, serão comparados. As métricas utilizadas para avaliação do comportamento do algoritmo serão: acurácia e efetividade (sensibilidade x especificidade), cuja obtenção e relevância estão descritas em (Pereira, 2012).

Nesta seção, são apresentados os resultados obtidos pelo *h*PSO .

Os 50 resultados gerados foram resumidos em valores globais da acurácia e do produto da sensibilidade x especificidade (efetividade) e e comparados com o MDMGeo, de (Pereira, 2012).

O *h*PSO apresentou resultados globais, de acurácia e efetividade, levemente superiores aos obtidos por (Pereira, 2012), nos testes executados para a base de dados de "Desenvolvimento Urbano", conforme Tabela 3.

Dado o comportamento estocástico do algoritmo *h*PSO e a pequena diferença apresentada entre os resultado de ambos os algoritmos, é possível inferir, os mesmos, possuem desempenhos similares diante do problema estudado.

Tabela 3. Acurácia Global e Especificidade x Sensibilidade

Algoritmo	Acurácia	Sensibilidade x Especificidade
*h*PSO	0.7840	0,6547
MDMGeo	0.7706	0,6484

5 Conclusões

Este trabalho propôs a adaptação de um algoritmo de otimização por enxame de partículas, para classificação de dados híbridos. Os resultados obtidos demonstram que a classificação de dados, através da criação de regras se-então, pode ser realizada com métodos que não sejam da forma de árvores de decisão ou redes neurais, que segundo (Castanheira, 2008), e outros, são as técnicas mais utilizadas para esse tipo de problema.

Com os resultados obtidos, nos trabalhos iniciais do *h*PSO, semelhantes a outros estudos, pode-se almejar através de uma maior exploração dos procedimentos de busca local e recombinação, o aprimoramento do processo de geração das regras e melhoria do desempenho do algoritmo, para o próprio problema de "Desenvolvimento Urbano".

A abordagem do algoritmo pode ser considerada relevante e adaptável a diversos problemas complexos e reais, deixando em aberto novos estudos e aplicações acerca de sua proposta.

Agradecimentos

Os autores agradecem às agências de fomento FAPEMIG e CNPq pelo apoio financeiro

Referências Bibliográficas

Berry, M. J. A. and Linoff, G. (1997). Data Mining Techniques for Marketing, Sales, and Customer Support. New Work: John Wiley & SonS, Inc, pp 1-19.

Castanheira, L. G. (2008). *Aplicação de Técnicas de Mineração de Dados em Problemas de Classificação de Padrões*. Ms. Universidade Federal de Minas Gerais.

Cohen, P. R. (1995). Empirical Methods for Artificial Intelligence. Cambridge: The Mit Press.

Elsimare, R. and Navathe, S. B. (2005). Sistemas de Banco de dados. São Paulo : Pearson Addison Wesley, pp 3 -17.

GeoMINAS. (2010). Programa Integrado de Uso da Tecnologia de Geoprocessamento pelos Órgãos do Estado de Minas Gerais. Disponível em: http://www.geominas.mg.gov.br. Acesso em: 10 jan. 2018.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Boston: Addison Wesley.

INPE. (2000). Instituto Nacional de Pesquisas Espaciais. Disponível em: http://www.dpi.inpe.br/spring/portugues/index.html Acesso em: 05 abr. 2018.

Pereira, M. de A. (2012). *Classificação de Dados Híbridos Através de Algoritmos Evolucionários*. Dr. Universidade Federal de Minas Gerais.

Prates, T. S. (2017). *Proposta de um Algoritmo Baseado em Particle Swarm Optimization (PSO) para Classificação de Dados*. Mestrado. Ms. Universidade Estadual de Montes Claros.