Análise da Qualidade de Diferentes Métricas para Agrupamento de Dados Utilizando Algoritmo Bio-Inspirado e Arquitetura MapReduce

Alternative Title: Quality Analysis of Different Metrics for Data Clustering Using Bio-Inspired Algorithm and MapReduce Architecture

Sandro Roberto Loiola de Menezes
Programa de Pós-Graduação em Computação Aplicada - UDESC
Joinville-SC, Brasil
sandrolmenezes@gmail.com

Rafael Stubs Parpinelli
Programa de Pós-Graduação em Computação Aplicada - UDESC
Joinville-SC, Brasil
rafael.parpinelli@udesc.br

RESUMO

Realizar tarefas de mineração de dados, como agrupamento, pode ser complexo devido alta dimensionalidade e volume dos dados minerados. Esse artigo propõe uma abordagem de agrupamento de dados utilizando Algoritmo Inspirado em Organismos Simbióticos (SOS) projetado na arquitetura MapReduce e analisa a evolução da qualidade dos agrupamentos, usando a medida de pureza, considerando 4 métricas de *fitness* diferentes. A qualidade dos agrupamentos obtidos por essa abordagem demonstram não apenas ser competitivos com a de outras abordagens como também verificou-se um aumento de desempenho utilizando a arquitetura *MapReduce*. Além disso, outra contribuição desse artigo é a análise da correlação da pureza do agrupamento com o valor de *fitness* obtido durante o processo de otimização. Percebeu-se que para algumas métricas de *fitness* existem alguns casos em que a pureza final encontrada no agrupamento é inferior a pureza encontrada em um momento anterior no processo de otimização.

Palavras-Chave

Agrupamento de Dados, *Hadoop MapReduce*, Algoritmos Bio-Inspirados.

ABSTRACT

Performing data mining tasks such as clustering can be very complex due to the high dimensionality and volume of data being mined. This paper proposes an approach for data clustering using the Symbiotic Organisms Search algorithm (SOS) developed in the MapReduce parallel architecture. Also, the cluster quality evolution is analysed using the purity measured considering four different fitness metrics. The cluster qualities obtained by the proposed approach not only shows to be competitive with other approaches but also increased its performance using the MapReduce architecture. Another contribution of this work is to bring to light the correlation between the cluster purity and the fitness value obtained during the optimization process. It was noticed that for some fitness metrics the final purity found by the optimization algorithm is less than the purity found in an earlier stage in the optimization process.

SBSI 2016, May 17th-20th, 2016, Florianópolis, Santa Catarina, Brazil

Categories and Subject Descriptors

I.2 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence
; J.1 [**Computer Applications**]: ADMINISTRATIVE DATA PROCESSING

General Terms

Algorithms, Experimentation

Keywords

Data Clustering, Hadoop MapReduce, Bio-Inspired Algorithms.

1. INTRODUÇÃO

Agrupamento de dados é uma tarefa de mineração que visa obter padrões para poder inferir conhecimento [7]. Esse processo consiste na divisão dos dados em grupos onde dados do mesmo grupo possuem características semelhantes. Os algoritmos *K-Means*, *Hierarchical Clustering* e *Density based clustering (DBSCAN)* são clássicos na aplicação de agrupamento de dados [7].

Entretanto, existem contextos de aplicação que mineração de dados fica impossibilitada ou se torna inviável pelas tecnologias convencionais como, por exemplo, alto consumo de tempo de processamento, baixo desempenho de algoritmos com o aumento da dimensionalidade e tamanho da base de dados [13]. Desta maneira, o processo de mineração de dados necessita de uma infraestrutura de processamento escalável para que seja possível alocar recursos conforme a necessidade. Além disso, é necessário utilizar técnicas que melhor se adequem tanto ao processamento em paralelo quanto à

complexidade dos dados que estão sendo minerados. De acordo com [6], algoritmos bio-inspirados possuem características que justificam sua escolha para serem aplicadas como técnicas de mineração de dados.

Algoritmos Bio-inspirados pertencem a uma subárea da Computação conhecida como Computação Natural, que utiliza a natureza como fonte de inspiração para desenvolver técnicas computacionais para resolver problemas complexos de otimização [18]. Esses algoritmos utilizam abordagens populacionais que processam um conjunto de indivíduos no qual cada indivíduo representa uma possível solução para o problema. Dada esta característica populacional, os indivíduos da população podem ser processados de forma paralela utilizando o paradigma *MapReduce* no contexto de mineração de dados [14]. Outra característica desses paradigmas bio-inspirados é o não determinismo aliado com rotinas de intensificação e diversificação da busca. Isso implica em uma exploração eficiente do espaço de soluções. Mineração em massas de dados normalmente compreendem espaços de busca bastante extensos e altamente não-lineares. Nesse caso, algoritmos determinísticos tendem a falhar quando aplicados em problemas com estas características [19].

Neste trabalho, o algoritmo inspirado em organismos simbióticos (*Symbiotic Organisms Search SOS)* é utilizado para realizar agrupamento dentro do *framework Hadoop MapReduce*. Esse *framework* provê processamento paralelo com tolerância a falhas e fácil escalabilidade. A abordagem desenvolvida é chamada de *MapReduce Clustering Symbiotic Organisms Search (MRCSOS)*. Dois objetivos são traçados neste trabalho. O primeiro objetivo verifica o desempenho do algoritmo SOS considerando dois critérios: tempo de processamento e pureza do agrupamento. O critério de tempo de processamento considera a execução do algoritmo SOS dentro da arquitetura MapReduce e sua execução de forma sequencial. O critério de pureza do agrupamento compara os resultados obtidos pelo algoritmo SOS dentro da arquitetura MapReduce com resultados obtidos por outros algoritmos bio-inspirados. O segundo objetivo faz uma reflexão sobre a análise do comportamento da qualidade de diferentes métricas de agrupamento segundo o grau de pureza durante o processo de otimização.

Este artigo está estruturado da seguinte maneira. A Seção 2 apresenta os fundamentos que embasam esse trabalho. A Seção 3 descreve os trabalhos relacionados. O método utilizado é descrito na Seção 4. Na Seção 5 os experimentos e resultados são apresentados. A Seção 6 exibe a discussão sobre os resultados e a Seção 7 apresenta as considerações finais e trabalhos futuros.

2. FUNDAMENTOS

2.1 Algoritmos Bio-Inspirados

Algoritmos bio-inspirados tem sido utilizados como métodos de otimização para problemas complexos normalmente não estacionários e multi-dimensionais. Como esses algoritmos utilizam abordagem populacional, cada indivíduo da população representa uma potencial solução do problema que está sendo otimizado. A métrica utilizada para determinar o quanto uma solução é boa para um determinado problema é chamada de função de eficiência ou *fitness*. Através dessa função é possível determinar qual é o melhor indivíduo da população em um determinado momento. Dentre alguns algoritmos bio-inspirados aplicados em problemas de mineração de dados pode-se citar: Otimização por Enxame de Partículas e Otimização por Enxame de Vermes Luminescentes.

A Otimização por Enxame de Partículas (*Particle Swarm Optimization - PSO*) foi proposta por Kennedy e Eberhart (1995) [9]. Esse algoritmo é inspirado no comportamento coordenado dos pássaros ao voarem e do movimento de cardume de peixes. No contexto computacional, o peixe ou pássaro representa uma partícula e cada partícula representa uma solução do problema. Cada partícula armazena sua posição no espaço de busca, uma velocidade de deslocamento e sua melhor posição até a iteração atual. No processamento do enxame de partículas, a melhor posição de todas as partículas também é armazenada, chamada de solução global. No momento da atualização da velocidade de uma partícula, tanto a melhor posição pessoal, representando a intensificação do algoritmo, quanto a melhor posição global do enxame, representando a diversificação, são consideradas de forma estocástica na atualização da velocidade. Após a atualização da velocidade também é atualizada a nova posição da partícula no espaço de solução para avaliar o quanto essa nova posição é boa e assim avaliar se a solução gerada é melhor que a melhor solução pessoal e global. Número de gerações, número de avaliação de função *fitness* e estagnação da melhor solução são utilizados como critérios de parada.

A Otimização por Enxame de Vermes Luminescentes (*Glowworm Swarm Optimization - GSO*) foi proposta por Krishnanand e Ghose (2005) [12]. Esse algoritmo foi inspirado na capacidade de controle de emissão de luminosidade de vermes em determinadas situações. A substância que influencia no nível de luminosidade é a luciferina. Quanto mais intensa a luciferina mais atrativa é a região. Com o tempo de permanência do indivíduo em regiões sem luminosidade o nível de luciferina do mesmo tende a diminuir. Computacionalmente, cada verme luminoso representa uma solução no espaço de solução do problema. A região que possui indivíduos com maior quantidade de luciferina é considerada promissora. Dessa forma, baseando-se apenas em informações locais os indivíduos vão se movimentando até se compactarem numa determinada região. A função *fitness* é atualizada considerando o nível de luciferina encontrada na região definida por um raio além da taxa de decaimento da luciferina.

Vários outros algoritmos bio-inspirados são revisados em [18]. A próxima seção descreve com mais detalhes o algoritmo bio-inspirado empregado neste trabalho.

2.1.1 Algoritmo Inspirado em Organismos Simbióticos

O Algoritmo Inspirado em Organismos Simbióticos (*SOS*) foi criado por Min-Yuan Cheng e Doddy Prayogo em 2014. Esse algoritmo é inspirado na relação simbiótica entre organismos distintos dentro de um ecossistema [5].

Pode-se entender uma relação simbiótica como sendo uma relação estabelecida entre dois organismos buscando a sobrevivência ou a adaptação de um ou de ambos no ecossistema. As relações simbióticas mais comuns encontradas na natureza são mutualismo, comensalismo e parasitismo. Mutualismo denota uma relação simbiótica entre duas espécies diferentes em que as duas se beneficiam. Comensalismo é uma relação simbiótica em que um dos indivíduos é beneficiado e o outro não é afetado. O Parasitismo é uma relação

simbiótica que um dos indivíduos é beneficiado enquanto o outro é prejudicado. O algoritmo *SOS* modela computacionalmente estas três relações. Cada organismo representa uma solução e é um indivíduo da população.

No mutualismo, para cada organismo da população *Xi* é selecionado aleatoriamente outro organismo *Xj* para então gerar dois novos organismos *Xinew* e *Xjnew*, conforme Equações 1 e 2, respectivamente. O vetor de mutualismo representa as características do relacionamento de *Xi* e *Xj*, calculado na Equação 3.

$$X_{inew} = X_i + rand(0,1) * (X_{best} - MutualVetor * BF_1) \quad (1)$$

$$X_{jnew} = X_j + rand(0,1) * (X_{best} - MutualVetor * BF_2) \quad (2)$$

$$MutualVetor = \frac{(X_i + X_j)}{2} \quad (3)$$

Onde as variáveis BF_1 e BF_2 podem assumir valores 1 ou 2 de forma aleatória caracterizando o não-determinismo. A variável X_{best} representa o melhor organismo da população atual. A operação $(X_{best} - MutualVector * BF)$ nas equações (1) e (2) representam o mutualismo pois seu objetivo é aumentar a vantagem mutua de sobrevivência dos dois novos organismos no ecossistema utilizando o melhor indivíduo sem prejuízo a nenhum dos envolvidos. Após a reprodução, na fase de seleção é verificado qual é o melhor indivíduo. Se o novo organismo for melhor que o seu ancestral o novo organismo substitui seu antecessor no ecossistema. Caso contrário, o novo organismo é descartado.

No comensalismo, cada organismo da população, X_i, é gerado um novo organismo, X_{inew}, utilizando o melhor organismo da população, X_{best}, e outro organismo aleatório, X_j, conforme a Equação (4).

$$X_{inew} = X_i + rand(-1,1) * (X_{best} - X_j) \quad (4)$$

A operação $(X_{best} - X_j)$ caracteriza o comensalismo, pois o novo organismo é beneficiado com a influência do melhor organismo do ecossistema sem prejudicar o melhor organismo. Se o novo organismo for melhor que o seu ancestral o novo organismo substitui seu antecessor no ecossistema. Caso contrário, o novo organismo é descartado.

No parasitismo, para cada organismo do ecossistema, é feita uma cópia do mesmo e após isso é alterada uma única dimensão aleatória de seu vetor. Dessa forma, um novo organismo, X_{inew}, chamado de vetor parasita é gerado. Em seguida é selecionado um outro organismo de forma aleatória, X_j, que assume o papel de hospedeiro ao participar da seleção de organismos. Se o *fitness* do vetor parasita for melhor que o *fitness* do hospedeiro o parasita substitui o hospedeiro no ecossistema.

Os organismos mais adaptados ao ecossistema são mantidos na evolução. Esse algoritmo possui os parâmetros tamanho da população, número máximo de iterações e máximo de avaliações. Esses dois últimos definem o critério de parada da execução.

2.2 Hadoop

Hadoop é uma ferramenta de código aberto utilizada para armazenar e manipular grandes massas de dados [17]. Com essa ferramenta, o grande problema existente relacionado com tempo no processo de leitura de uma grande massa de dados é amenizado com a leitura de múltiplos discos de forma simultânea. Nesse tipo de leitura podem ocorrer algumas falhas que também são contornadas através da replicação de dados. Esse armazenamento seguro e compartilhado é provido pelo componente *Hadoop Distributed File System (HDFS)* [11]. Outro componente provido pelo *Hadoop* é o sistema de análise chamado de *framework MapReduce*. *MapReduce* explora a arquitetura de armazenamento distribuída do *HDFS* provendo assim escalabilidade, confiabilidade e processamento paralelo [16]. Nas subseções a seguir os componentes *HDFS* e *Hadoop MapReduce* serão detalhados.

2.2.1 *HDFS*

HDFS é um sistema de arquivos projetado para armazenar dados com um padrão contínuo de acesso e em *cluster*. *HDFS* possui arquitetura cliente-servidor e utiliza o protocolo de comunicação *Transmission Control Protocol (TCP)*. Assim, existem dois tipos de nodo. Um deles é o servidor, chamado *namenode* ou referenciado também por *Master*. O *Master* é responsável por gerenciar a localização de blocos de arquivos fragmentados e replicados, além de manutenir os metadados e a árvore do sistema de arquivos. O outro tipo de nodo é o cliente, chamado de *datanode* ou também referenciado por *Worker*. O *Worker* possui como função armazenar e recuperar blocos de dados solicitados por aplicações de *software* ou pelo *Master*. Em alguns casos a utilização de *HDFS* pode não ser adequada, como nos casos que envolvem o contexto de baixa latência de acesso porque o tempo requerido para inicialização do serviço, divisão e junção das tarefas, além da comunicação entre os nós certamente inviabiliza a utilização do mesmo [11].

2.2.2 *MapReduce*

MapReduce é um modelo de programação que possui principalmente duas funções, uma chamada de *map* e outra chamada de *reduce*. Cada uma dessas funções recebe como entrada de dados um conjunto de pares chave-valor e após o processamento dessas informações, de acordo com a função implementada pelo programador, tem como saída um conjunto de pares chave-valor. Basicamente, existem duas fases de execução no processamento *MapReduce*, cada fase executando uma função. Na primeira fase é executado o processo de mapeamento através da função *map* e na segunda fase é executado o processo de redução através da função *reduce*. Essas fases podem ser executadas em paralelo através dos *datanodes* disponíveis em um determinado *cluster*. O tempo de processamento depende do volume de dados processados e o método de distribuição e replicação dos dados no *cluster*. Quanto maior o volume de dados maior será o tempo de processamento e quanto maior o *cluster* menor será o tempo de execução [16]. Um *job MapReduce* representa um processo completo de execução do *framework MapReduce* num determinado arquivo *HDFS*.

A Figura 1 mostra uma visão geral do fluxo de execução de um programa utilizando o *framework MapReduce*.

No fluxo com rótulo (1) a biblioteca *MapReduce* faz algumas cópias do programa do usuário para as estações *Master* e *Workers* do *cluster*. No fluxo com rótulo (2) o nodo *Master* faz a atribuição das tarefas *map* e *reduce* para os *workers*. No fluxo cujo rótulo é (3) os *workers* que irão realizar a execução da função *map* leem o conteúdo correspondente ao dado de entrada e transformam esses dados em pares no formato chave-valor. Os dados nesse formato são passados como entrada para a função *map* que é implementada pelo programador. A saída de dados da função *map* também

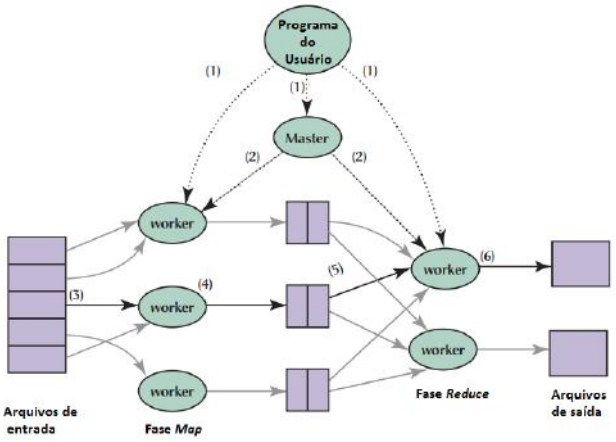

Figura 1: Execução *job MapReduce*. Adaptado de [8].

está em formato de pares chave-valor e fica armazenada na memória. No fluxo correspondente ao rótulo (4) os dados de saída da função *map* que estão em memória são gravados em disco local, chamados de arquivos intermediários, e posteriormente é enviada a localização desses arquivos para o *Master*. No fluxo (5) os *workers* responsáveis pra fazer a redução são notificados pelo *Master* com a informação de localização dos arquivos intermediários. Assim, os *workers* fazem a leitura remota dos arquivos intermediários. Os *workers* de redução ordenam os dados no formato de pares chave-valor de tal forma que todos os valores de uma mesma chave são agrupados e relacionados àquela chave. Assim todos os pares terão chave diferente e cada chave terá o conjunto de valores relacionado a ela. Em seguida, ainda no fluxo de número (5), cada chave com seus valores são enviados para a função *reduce* processar de acordo com a implementação realizada pelo programador. Após o processamento, representado pelo fluxo de número (6), a saída de dados é persistida em um arquivo final. Ao finalizar todas as tarefas de mapeamento e redução, o *Master* retorna para o programa do usuário [8].

3. TRABALHOS RELACIONADOS

O objetivo dessa seção é identificar as principais abordagens que realizam agrupamento de massas de dados e utilizam algoritmos bio-inspirados e o paradigma *MapReduce*.

Como todos os trabalhos utilizam o *framework Hadoop*, a paralelização com tolerância a falhas e balanceamento de carga é uma característica implícita e comum em todos os trabalhos comentados a a seguir.

Aljarah (2012) [2] propôs um algoritmo que realiza a tarefa de agrupamento e tem como base o algoritmo *PSO* adaptado ao paradigma *MapReduce*. Os resultados mostraram uma melhor qualidade comparado com o algoritmo *K-Means* além de possuir uma boa relação de desempenho (*speedup*) mantendo a qualidade do agrupamento. Esse mesmo algoritmo foi utilizado em Aljarah (2013) [3] como um componente que realiza agrupamento em volume de dados de tráfico de redes de grande escala. Com essa arquitetura os resultados obtiveram uma boa taxa de detecção de invasão verdadeira e baixa taxa de detecção falsa. Além disso, foi mantida uma boa escalabilidade e *speedup*.

Judith (2015) [10] apresenta um sistema que realiza tarefa de agrupamento em documentos utilizando o algoritmo *PSO*. Além de *PSO*, o sistema também utiliza o algoritmo de agrupamento de dados *K-Means* e a técnica de redução de dimensionalidade *Latent Semantic Indexing (LSI)*. O sistema também demonstrou bom *speedup* com a utilização do *framework Hadoop MapReduce*.

Al-Mad (2014) [1] apresenta um algoritmo que realiza a tarefa de agrupamento e tem como base o algoritmo *GSO*. A cada iteração do algoritmo *GSO* ocorre a execução de um *job MapReduce* no arquivo de dados. Os resultados mostram que esse algoritmo possui um bom tempo de performance e mantem melhor qualidade nos grupos formados do que o *K-Means* e pelo algoritmo proposto por Aljarah (2012) [2].

O algoritmo *ACO* foi proposto por Bhavani (2014) [4] para realizar a tarefa de agrupamento. Nessa abordagem o *ACO* é utilizado na poda da uma árvore de cobertura mínima até a formação de grupos. Esse algoritmo conseguiu alcançar um bom *speedup*. O mecanismo distribui o algoritmo *ACO* de tal forma que no mapeamento a quantidade de feromônios influencia no cálculo da arvore de cobertura mínima. Já na redução o feromônio é atualizado.

Uma revisão mais detalhada sobre algoritmos bio-inspirados que utilizam o paradigma *MapReduce* aplicados em problemas de mineração de dados pode ser encontrada em [14].

4. MODELO DESENVOLVIDO: MRCSOS

Essa seção descreve como foi desenvolvida a abordagem empregada neste trabalho, o *MapReduce Clustering Symbiotic Organisms Search (MRCSOS)*. A versão canônica do algoritmo *SOS* possui três relações simbióticas: mutualismo, comensalismo e parasitismo [5]. Destas relações, mutualismo e comensalismo atuam como rotinas de intensificação e o parasitismo como rotina de diversificação. Com esse comportamento, o algoritmo *SOS* deixa de explorar mais regiões no espaço de busca pois intensifica demasiadamente a busca, como mostrado em [15]. Dessa forma, *MRCSOS* não utiliza mutualismo no processo evolutivo. Assim, favorece um maior equilíbrio entre a intensificação e a diversificação. O funcionamento do *MRCSOS* pode ser dividido em três principais fases: inicialização, processamento e verificação da qualidade.

Na fase de inicialização são recebidos os parâmetros: quantidade de organismos no ecossistema, base de dados, número de centroides, quantidade máxima de avaliações e quantidade máxima de iterações. Com essas informações o *MRCSOS* inicializa os organismos de acordo com a quantidade parametrizada. Cada organismo possui um identificador único e representa um conjunto de centroides. Cada centroide possui um identificador e é representado por um vetor de n dimensões onde n é igual a quantidade de atributos da base de dados. Os valores dos centroides são inicializados aleatoriamente com itens da base de dados. Dessa forma é possível garantir que cada centroide possua pelo menos um item agrupado. Isso garante que todas as soluções iniciais são válidas, pois cada centroide deve agrupar pelo menos um item da base. Finalizando a fase de inicialização é calculada a função *fitness* de cada organismo da população. Na fase de processamento ocorre a evolução dos organismos através das relações simbióticas de comensalismo e parasitismo, como descrito na Seção 2.1.1. A fase de processamento é executada até que a condição de parada, número máximo de iterações,

seja alcançada. Após a fase de processamento, a execução do *MRCSOS* é finalizada com a verificação da qualidade do agrupamento do melhor organismo do ecossistema, chamado de *Xbest*.

A função *fitness* é a única função do *MRCSOS* que está projetada na arquitetura *MapReduce*. Isso porque é necessário ler toda a base de dados para calcular a distância de cada item ao respectivo centroide que o agrupa além da distância entre os centroides. A função *map* processa toda a população de organismos e para cada organismo é verificado qual é o centroide que possui a menor distância até o item corrente. A função *reduce* recebe como entrada o identificador do organismo e o conjunto de informações do centroide. Com essas informações é calculada a intradistância por centroide. Quanto menor a intradistância melhor será a qualidade do agrupamento. A interdistância é a distância entre os centroides do agrupamento e a mesma deve ser maximizada para aumentar a qualidade dos grupos. Esse trabalho utiliza quatro funções para avaliar os organismos do ecossistema e são referenciadas por *f1*, *f2*, *f3* e *f4* disponíveis em [3]. Os cálculos comuns, como a intradistância Equação (5), para *f1*, *f2*, *f3* são expostos a seguir.

$$IntraD_j = \sum_{i=1}^{|cr_j|} Distance(cr_{ji}, c_j) \tag{5}$$

Onde $cr_{j,i}$ representa itens de dados cobertos pelo centroide c_j e $|cr_j|$ representa a quantidade de itens cobertos pelo cenrtroide c_j. A distância utilizada na Equação 2 é a euclidiana. A interdistância é calculada conforme Equação (6).

$$InterDist = \sum_{i=1}^{k} \sum_{j=i}^{k} (Distance(c_i, c_j))^2 \tag{6}$$

Onde k é o número de centroides e c_i, c_j representam os centroides do organismo. E a Equação (7), *Sum Squared Errors (SSE)* .

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{|C_j|} (Distance(c_i, c_j))^2 \tag{7}$$

Onde $|C_j|$ representa a quantidade de itens cobertos pelo centroide x_i e c_j representa itens cobertos por c_j. Enfim, a função *f1* é definida pela Equação (8), *f2* conforme Equação (9) e *f3* conforme Equação (10).

$$f1 = \frac{InterDist}{SSE} \times \frac{intraD}{max(intraD)} \tag{8}$$

$$f2 = \frac{InterDist}{\frac{intraD}{max(intraD)}} \tag{9}$$

$$f3 = \frac{1}{SSE \times \frac{intraD}{max(intraD)}} \tag{10}$$

As funções *f1* e *f2* consideram a interdistância em suas equações, sendo que a função *f2* dá um peso maior a interdistância. As três funções consideram a maior intradistância em suas equações. Além disso, as três funções são de maximização, ou seja, quanto maior seu valor melhor é a solução representada pelo organismo.

A função *f4* é uma função de minimização e não utiliza nenhuma das equações apresentadas anteriormente e é definida conforme Equação (11).

$$f4 = \frac{\sum_{j=1}^{k} \frac{\sum_{i=1}^{n_j} Distance(R_i, C_j)}{n_j}}{k} \tag{11}$$

Onde k é o número de centroides, n_j é o número de itens cobertos pelo centroide C_j e R_i representa os itens cobertos por C_j. A distância utilizada é a distância de *Manhattan*.

5. EXPERIMENTOS

Os experimentos foram realizados utilizando duas bases de dados amplamente empregadas na literatura: *Magic* e *Electricity*. A base *Magic* possui 19.020 instâncias de dados, 10 atributos, atributo-meta com 2 classes e 3 *Megabytes* e está disponível no *UCI*[1]. A base *Electricity* possui 45.312 instâncias de dados, 8 atributos, atributo-meta com 2 classes e 6 *Megabytes* e está disponível no *MOA*[2].

Os experimentos foram realizados em um *cluster* formado por três nodos, um servidor e dois clientes, com a seguinte configuração: 3.9 *Gigabytes* de RAM, processador *AMD Phenom* (tm) II X4 B93 Processor x4, 98.3 *Gigabytes* de HD, Sistema Operacional *Ubuntu* 14.04 LTS 64-bit, Hadoop versão 2.7.1, *Java* 1.7. O algoritmo *MRCSOS* foi desenvolvido na linguagem de programação Java. Os parâmetros utilizados foram: número de centroides igual a 2, tamanho da população igual a 100, número de execuções igual a 10, número máximo de iterações igual a 50 e 10.100 avaliações de *fitness* por execução.

A qualidade dos grupos é medida utilizando a métrica de pureza *FScore* [20], assim como *MRCPSO* [1] e *MRCGSO* [2]. Os resultados do *MRCSOS* consideram a média e desvio-padrão de 10 execuções independentes.

6. RESULTADOS E ANÁLISES

Os resultados das médias de pureza dos agrupamentos estão exibidos na Tabela 1. A primeira coluna informa o algoritmo empregado. A segunda coluna exibe os resultados de pureza na base *Magic*, onde os mesmos correspondem média e desvio-padrão e melhor pureza em todas as execuções entre parênteses. Na terceira coluna são exibidos os resultados na mesma estrutura da segunda coluna, entretanto, para a base *Electricity*. Em relação aos resultados na base *Magic*,

Tabela 1: Resultados obtidos para as purezas dos agrupamentos

Algoritmo	*Magic*	*Electricity*
K-Means	0.60	0.51
MRCPSO	0.65	0.58
MRCGSO	0.66	0.58
MRCSOS(f1)	0.62 ±3.25(0.65)	0.56 ±2.29(0.60)
MRCSOS(f2)	0.63 ±2.21(0.65)	0.59 ±4.93(0.66)
MRCSOS(f3)	0.62 ±3.14(0.65)	0.61 ±4.41(0.66)
MRCSOS(f4)	0.63 ±1.49(0.65)	0.62 ±3.04(0.66)

[1] *UCI Machine Learning Repository* é um repositório de dados utilizados pela comunidade de aprendizado de máquinas para análise empírica de algoritmos, https://archive.ics.uci.edu/ml/index.html.

[2] *Massive Online Analysis (MOA)* é um *framework* de mineração de dados que disponibiliza algumas bases de dados para realizar analises de mineração de dados. http://moa.cms.waikato.ac.nz/datasets/.

as médias de purezas obitidas por *MRCPSO* e *MRCGSO* são superiores a média da pureza de todas as versões de *MRCSOS*. Apesar disso, nas melhores execuções do *MRCSOS* os resultados se mostram competitivos. Verifica-se também que os resultados obitidos pelo *MRCSOS* são equivalentes para todas as funções de *fitness*, possuindo sobreposição dos intervalos de desvio-padrão.

Na base *Electricity* a média da pureza obitida por *MRCSOS* em relação ao *MRCPSO* e ao *MRCGSO*, apenas a versão *MRCSOS(f1)* apresentou a média da pureza inferior. *MRCSOS(f2)*, *MRCSOS(f3)* e *MRCSOS(f4)* apresentam média de pureza superior a dessas abordagens. Nas duas bases, *MRCSOS* obteve melhores resultados que o *K-Means*.

Em relação ao tempo de processamento, todas as quatro funções foram executadas nas duas bases de forma sequencial e paralela com *MapReduce* em 2 nodos. De maneira geral, todas as funções apresentaram praticamente o mesmo percentual de ganho de desempenho ao executar de forma paralela. Na base *Magic* o desempenho melhorou cerca de 18% (média de 66,8 minutos para 54,9 minutos) e na base *Electricity* o desempenho melhorou cerca de 21% (média de 101,2 minutos para 80,1 minutos).

A correlação entre função *fitness* e pureza do agrupamento utilizando as funções *f1, f2, f3* e *f4* com a média de 10 execuções do algoritmo *MRCSOS* são apresentados nas Figuras 2 e 3 para as bases *Magic* e *Electricity*, respectivamente. O objetivo dessa análise é refletir sobre o comportamento da qualidade das funções de agrupamento segundo o grau de pureza durante o processo de otimização. O eixo-x dos gráficos representa a quantidade de iterações e o eixo-y representa os valores da função *fitness* (gráficos da esquerda) e o percentual de pureza do agrupamento (gráficos da direita). O gráfico de *fitness* sempre possui comportamento assintótico crescente nas funções de maximização (*f1*, *f2* e *f3*) e decrescente na função de minimização (*f4*). Esse gráfico serve para evidenciar a evolução da função *fitness* a medida que vão sendo executadas as iterações do algoritmo. A atenção maior deve ser dada ao gráfico de pureza para analisar as instabilidades do grau de pureza.

O comportamento da pureza da função *f1* nas duas bases nas iterações iniciais, mesmo com o *fitness* melhorando, demonstra declínio da pureza. Além disso, nas duas bases, a pureza final do agrupamento foi a melhor encontrada no curso da evolução do sistema. O comportamento da função *f2* é parecido nas duas bases em relação a instabilidade, ou seja, várias vezes a pureza tende a aumentar e diminuir. Entretanto, na base *Magic* da função *f2* não obteve como pureza final do agrupamento a melhor pureza encontrada ao longo do processamento, ou seja, o processo de otimização não está conservando a melhor solução encontrada. Já a função *f3* apresenta uma evolução estável tanto da função *fitness* quanto da pureza do agrupamento na base *Magic*. Entretanto, na base *Electricity*, ao final do processamento houve perda na pureza, não recuperada posteriormente. A função *f4* é de minimização, ou seja, quanto menor o seu valor melhor é a solução. Nas duas bases a função *f4* apresenta comportamento similar. Nas iterações iniciais, até próximo da iteração 5, ocorre sempre a melhora da pureza mas no final do agrupamento ocorre declínio do grau de pureza sem ocorrer ascensão posterior.

Considerando os experimentos que geraram os gráficos nas duas bases, percebeu-se que mesmo quando a interdistância aumenta consideravelmente e a intradistância também aumenta, em alguns casos, o valor do *fitness* melhora. Entretanto, nesse mesmo caso nem sempre o grau de pureza do agrupamento melhora, ocasionando o declínio da pureza dos dados. As funções *f1* e *f2* utilizam essa distância em suas equações gerando instabilidade na pureza no decorrer da evolução do sistema. Dentre as cinco funções, essas duas apresentam o comportamento mais instável. A função *f3* é a que apresenta maior estabilidade na evolução da pureza, principalmente na base *Magic*. Esse comportamento é atribuído ao fato da interdistância não ser considerada no cálculo dessa função. Além disso, a utilização da maior intradistância e a média da intradistância nessa função ajudam na correlação direta entre a evolução do *fitness* e da pureza do agrupamento. A função *f4* apresenta boa estabilidade na pureza do agrupamento.

Diante dos gráficos apresentados, o principal fato observado é que as funções *f2* e *f4* na base *Magic* e as funções *f2*, *f3* e *f4* na base *Electricity* obtiveram pureza final inferior a pureza obtida ao longo do processo de agrupamento. Percebe-se que esse comportamento varia de acordo com a base de dados pois a função *f3* não apresentou esse comportamento na base *Magic*. Dessa forma, baseado nos experimentos, é possível perceber que em determinadas bases algumas funções não conseguem garantir em todos os casos que o grau de pureza do agrupamento seja o melhor já encontrado durante o processo de otimização.

De maneira geral, a função *f4* apresenta melhor resultado considerando a pureza e o menor desvio-padrão, conforme Tabela 1. Entretanto seu resultado final poderia ser melhor visto que tanto na base *Magic* quanto na base *Electricity* ocorreu perda de qualidade do agrupamento no processo de otimização, conforme Figuras 2 e 3.

7. CONSIDERAÇÕES FINAIS

Esse artigo apresenta uma abordagem para agrupamento utilizando o algoritmo bio-inspirado *SOS* desenvolvido na arquitetura *Hadoop MapReduce*, o *MRCSOS* . Foram realizados experimentos do *MRCSOS* utilizando quatro funções de *fitness* diferentes no intuito de correlacionar a evolução do *fitness* com a pureza do agrupamento à medida que o sistema evolui e também comparar os resultados dessa abordagem com outras abordagens existentes. Este trabalho traz uma reflexão sobre a análise do comportamento da pureza de diferentes funções de agrupamento durante o processo de otimização. Ao contrário do que se esperava, na análise de resultados foi verificado que em alguns casos algumas funções obtiveram o grau de pureza final inferior ao grau de pureza obtido em iterações prévias. Isso indica que ocorre a perda da qualidade do agrupamento devido a não identificação da melhor solução encontrada por parte da métrica ao longo de todo o processo. Considerando o desvio padrão, os resultados da média final da qualidade do agrupamento de dados do *MRCSOS* são competitivos com os resultados obtidos outras abordagens semelhantes como *MRCPSO* e *MRCGSO*. Já em relação aos resultados obtidos pelo algoritmo clássico de agrupamento *K-Means* os resultados obtidos por *MRCSOS* demonstram ser melhores.

Como trabalhos futuros serão realizados experimentos em uma infraestrutura de nuvem para que seja possível extrapolar o número de nodos do *cluster Hadoop* e assim analisar mais a fundo a escalabilidade do *MRCSOS*. Outro direcionamento é o uso de diferentes funções de *fitness* e outros indi-

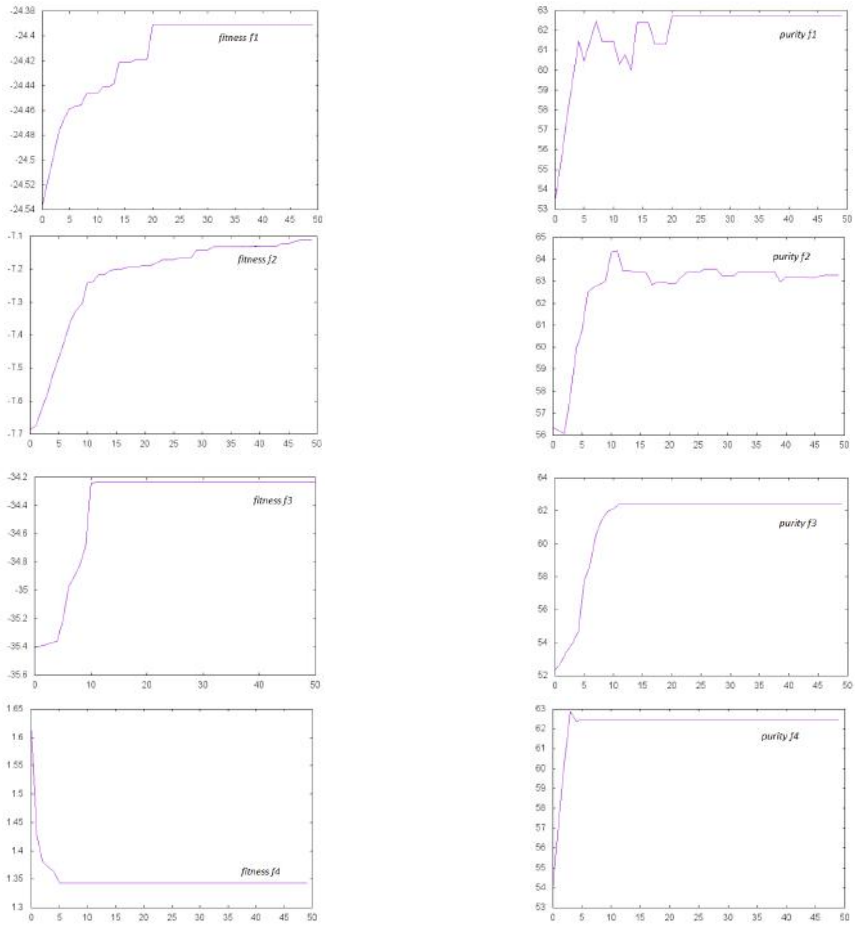

Figura 2: Gráficos de convergência do *fitness*, coluna da esquerda, e da pureza, coluna da direita, do *MRCSOS* para a base *Magic*.

cadores de qualidade, além de utilizar outras bases de dados.

8. REFERÊNCIAS

[1] N. Al-Madi, I. Aljarah, and S. A. Ludwig. Parallel glowworm swarm optimization clustering algorithm based on mapreduce. In *Swarm Intelligence (SIS), 2014 IEEE Symposium on*, pages 1–8. IEEE, 2014.

[2] I. Aljarah and S. A. Ludwig. Parallel particle swarm optimization clustering algorithm based on mapreduce methodology. In *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*, pages 104–111. IEEE, 2012.

[3] I. Aljarah and S. A. Ludwig. Mapreduce intrusion detection system based on a particle swarm optimization clustering algorithm. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 955–962. IEEE, 2013.

[4] R. Bhavani and G. S. Sadasivam. A novel ant based clustering of gene expression data using mapreduce framework. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2:398–402, 2014.

[5] M.-Y. Cheng and D. Prayogo. Symbiotic organisms search: A new metaheuristic optimization algorithm. *Computers & Structures*, 139:98–112, 2014.

[6] S. Cheng, Y. Shi, Q. Qin, and R. Bai. Swarm intelligence in big data analytics. In *Intelligent Data Engineering and Automated Learning–IDEAL 2013*, pages 417–426. Springer, 2013.

[7] M. Dalal and N. Harale. A survey on clustering in data mining. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, pages 559–562. ACM, 2011.

[8] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[9] R. C. Eberhart, J. Kennedy, et al. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[10] J. Judith and J. Jayakumari. An efficient hybrid distributed document clustering algorithm. *Scientific Research and Essays*, 10(1):14–22, 2015.

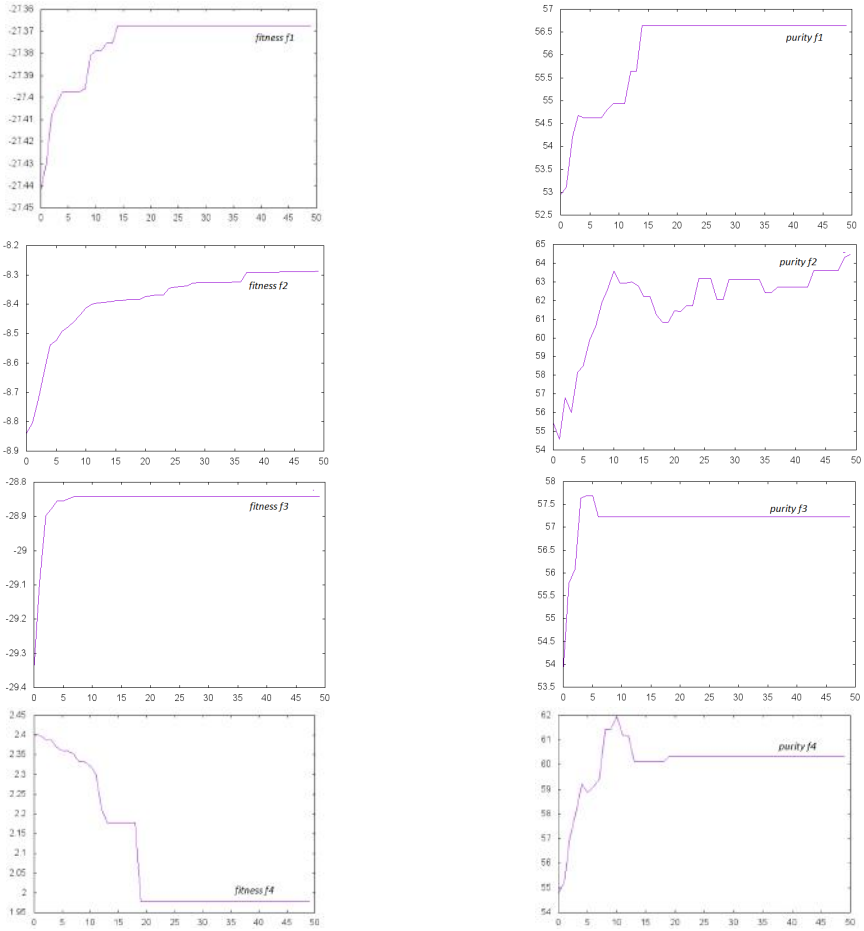

Figura 3: Gráficos de convergência do *fitness*, coluna da esquerda, e da pureza, coluna da direita, do *MRCSOS* para a base *Electricity*.

[11] A. Katal, M. Wazid, and R. Goudar. Big data: Issues, challenges, tools and good practices. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 404–409. IEEE, 2013.

[12] K. Krishnanand and D. Ghose. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 84–91. IEEE, 2005.

[13] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[14] S. R. L. Menezes, R. S. Freitas, and R. S. Parpinelli. Mineração em grandes massas de dados utilizando hadoop mapreduce e algoritmos bio-inspirados: Uma revisão sistemática. *Revista de Informática Teórica e Aplicada*, 2016. Trabalho aceito para publicação.

[15] W. Migliorini, L. Andre, and R. Parpinelli. Análise das rotinas de intensificação e diversificação no algoritmo inspirado em organismos simbioticos. In *Congresso Brasileiro de Inteligência Computacional*, 2015.

[16] S. Narayan, S. Bailey, and A. Daga. Hadoop acceleration in an openflow-based cluster. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 535–538. IEEE, 2012.

[17] N. Nurain, H. Sarwar, M. P. Sajjad, and M. Mostakim. An in-depth study of map reduce in cloud environment. In *Advanced Computer Science Applications and Technologies (ACSAT), 2012 International Conference on*, pages 263–268. IEEE, 2012.

[18] R. S. Parpinelli and H. S. Lopes. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, 3(1):1–16, 2011.

[19] X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu. *Swarm intelligence and bio-inspired computation: theory and applications*. Newnes, 2013.

[20] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM, 2002.