Revista Brasileira de Computação Aplicada, Julho, 2018

DOI: 10.5335/rbca.v10i2.7651
Vol. 10, Nº 2, pp. 18–28
Homepage: seer.upf.br/index.php/rbca/index

ISSN 2176-6649

ARTIGO ORIGINAL

Análise de diferentes métricas para agrupamento de dados utilizando um algoritmo de busca paralela baseado em organismos simbióticos

Sandro Roberto Loiola de Menezes[1], Mateus Boiani[1] and Rafael Stubs Parpinelli[1]

[1]Universidade do Estado de Santa Catarina (UDESC), Programa de Pós-graduação em Computação Aplicada (PPGCA)

*sandrolmenezes@gmail.com; mateus.boiani@edu.udesc.br; rafael.parpinelli@udesc.br

Submetido: 22/11/2017. **Revisado:** 14/06/2018. **Aceito:** 21/06/2018.

Resumo

Este artigo propõe uma abordagem para realizar agrupamento de dados utilizando o Algoritmo de Busca por Organismos Simbióticos (SOS) em uma arquitetura Hadoop MapReduce, chamado de MRCSOS. O algoritmo SOS é responsável pela exploração do espaço de busca enquanto a arquitetura Hadoop provê escalabilidade através do paralelismo. A principal contribuição deste trabalho é a correlação das métricas de pureza, entropia e diversidade genotípica utilizando diferentes métricas de agrupamento de dados durante o processo de otimização. Os resultados obtidos em três bases de dados mostraram que algumas métricas de agrupamento não mantém a qualidade do agrupamento durante toda otimização. Nas métricas analisadas, a função Silhueta ($F5$) foi a melhor de todas. Esta função consegue manter o melhor agrupamento durante todo processo de otimização. Além disto, o algoritmo MRCSOS com a função $F5$ obteve os melhores resultados, ou pelo menos competitivos, quando comparados com outras abordagens existentes na literatura.

Palavras-Chave: Agrupamento de Dados; Algoritmo de Busca por Organismos Simbióticos; Algoritmos Bio-inspirados; Hadoop MapReduce

Abstract

This paper proposes an approach to perform data clustering using the Symbiotic Organisms Search (SOS) algorithm using the Hadoop MapReduce, named MRCSOS. This combination provides an efficient exploration of the search space and the scalability with parallelism provided by the Hadoop. The main contribution of this work is the correlation analysis of purity, entropy and genotypic diversity using different metrics of data clustering during the optimization process. The results obtained with three different datasets showed that some clustering metrics did not maintain the clustering quality during the optimization process. In these cases the final data clustering got worst quality than the quality obtained in previous iterations. Among the analyzed metrics, the Silhouette function (F5) was better than others. This function can maintain the best cluster through the whole optimization process. Furthermore, MRCSOS using the function F5 got better results, or at least competitive, when compared with other existing approaches in literature.

Key words: Data Clustering; Symbiotic Organisms Search Algorithm; Bio-inspired Algorithms; Hadoop MapReduce

1 Introdução

Agrupamento de dados é uma tarefa de mineração que busca obter padrões para inferir conhecimento (Fahad et al.; 2014). O agrupamento consiste em dividir um conjunto de objetos de dados em diferentes grupos de acordo com a similaridade, onde cada grupo possui dados com características bem próximas.

Entretanto, existem alguns contextos de aplicação em que o processo de mineração de dados fica impossibilitado ou se torna inviável devido a existência de algum requisito que não é atendido de forma eficiente pelas tecnologias convencionais. Por exemplo, no contexto de grandes volumes de dados, um processo de mineração pode ocasionar em alto consumo de tempo de processamento (Menezes et al.; 2016). Outro fator limitante é a dimensionalidade dos dados que influencia diretamente no desempenho do algoritmo (Katal et al.; 2013). Além disso, as técnicas convencionais de mineração de dados precisam realizar a carga dos dados que serão analisados na memória antes de realizar a mineração (Leskovec et al.; 2014). No contexto de grandes volumes de dados, realizar esse carregamento na memória é mais um ponto crítico, pois delimita a capacidade máxima de dados que serão minerados em um determinado momento.

Para realizar mineração em grandes massas de dados, é necessário que o processamento seja realizado em uma infraestrutura de processamento escalável, para que seja possível realizar a alocação de recursos de acordo com a demanda requerida pelo volume de dados a ser processado. A linha de pesquisa deste trabalho considera o contexto de escalabilidade horizontal. Utilizando escalabilidade horizontal é possível realizar processamento distribuído e paralelo. Diante disso, é necessário utilizar técnicas com características mais adequadas ao processamento paralelo, como é o caso de algoritmos bio-inspirados.

Algoritmos bio-inspirados pertencem a uma vertente da computação conhecida como Computação Natural, que utiliza a natureza como fonte de inspiração para desenvolver técnicas computacionais não-determinísticas para resolver problemas complexos de otimização (Parpinelli and Lopes; 2011). Outra característica vantajosa desses paradigmas bio-inspirados é o não-determinismo aliado com a intensificação e diversificação da busca. Isso implica em uma exploração mais eficiente do espaço de soluções (Rajaraman et al.; 2015). Algoritmos bio-inspirados englobam abordagens populacionais que processam um conjunto de possíveis soluções para o problema e são chamados de indivíduos, partículas ou organismos, dependendo do algoritmo em consideração. Essa característica populacional dos algoritmos bio-inspirados os tornam naturalmente indicados para aplicação em arquiteturas paralelas.

Hadoop é um projeto com código aberto criado pela *Apache Software Foundation* que vem sendo utilizado como ferramenta no contexto de mineração de dados. A utilização dessa ferramenta é justificada pelo fornecimento de um *framework* de processamento paralelo, de fácil escalabilidade e utilização, além de usar um sistema de arquivos distribuído e com alta taxa de transferência (Mohammed et al.; 2014).

Existem duas abordagens na literatura (MRCPSO e MRCGSO) que possuem arquiteturas semelhantes a arquitetura utilizada nessa pesquisa. MRCPSO (*MapReduce Clustering Particle Swarm Optimization*) é uma abordagem de agrupamento de dados que utiliza o algoritmo *Particle Swarm Optimization*, que é um algoritmo inspirado no comportamento das formigas, projetado na arquitetura *Hadoop MapReduce* (Aljarah; 2013). MRCSGO (*MapReduce Clustering Glowworms Swarm Optimization*) realiza agrupamento baseado no algoritmo GSO, algoritmo inspirado no comportamento dos vermes luminescentes, utilizando a infraestrutura do *Hadoop MapReduce* (Aljarah; 2013).

Neste trabalho, o algoritmo inspirado em organismos simbióticos (*Symbiotic Organisms Search* – SOS) é utilizado para realizar agrupamento dentro do *framework Hadoop MapReduce*. Essa abordagem provê processamento paralelo com tolerância a falhas e fácil escalabilidade. A abordagem desenvolvida é chamada de *MapReduce Clustering Symbiotic Organisms Search* (MRCSOS). Três objetivos são traçados neste trabalho. O primeiro objetivo verifica a qualidade dos resultados de agrupamento obtidos pelo MRCSOS em relação a outros algoritmos bio-inspirados. O segundo objetivo avalia a qualidade de diferentes métricas de agrupamento aplicadas no MRCSOS durante o processo de otimização. Com isso pretende-se verificar se o curso da qualidade do agrupamento é estável, ou seja, sempre melhora com a evolução do sistema ou se ocorrem instabilidades. O último objetivo é analisar o desempenho do algoritmo SOS dentro da arquitetura *MapReduce*. Esse objetivo visa verificar o nível de escalabilidade do sistema.

Este artigo está estruturado da seguinte maneira. A Seção 2 apresenta os fundamentos que embasam esse trabalho. O método utilizado é descrito na Seção 3. Na Seção 4 os experimentos, resultados e análises são detalhados. Por fim, a Seção 5 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 multidimensionais (Kar; 2016). Computação Evolucionária e Inteligência de Enxame são duas ramificações de algoritmos bio-inspirados que possuem a semelhança de representarem meta-heurísticas estocásticas e serem baseadas em população de possíveis soluções, chamadas de indivíduos. Cada possível solução deve ser avaliada para determinar o quão esta é boa ou não para um determinado problema. Esta função é chamada de função de eficiência ou função *fitness*.

Em uma população de indivíduos, cada indivíduo representa uma potencial solução do problema que está sendo otimizado. A população de indivíduos tende a mover-se para áreas onde estão localizadas as melhores soluções considerando o espaço de solução de um determinado problema. Além disso, Inteligência de Enxame e Computação Evolucionária mantém e sucessivamente melhoram uma população de potenciais soluções até que alguma condição de parada seja satisfeita.

2.1.1 Algoritmo SOS

O algoritmo SOS, que é utilizado neste trabalho, é um algoritmo do grupo da Computação Evolucionária. Esse algoritmo é inspirado na relação simbiótica entre organismos distintos dentro de um ecossistema (Cheng and Prayogo; 2014). 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. O algoritmo SOS modela computacionalmente três relações: 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.

Cada organismo representa uma solução e é um indivíduo da população e cada relação simbiótica possui uma característica específica para reprodução de novos organismos.

No mutualismo, para cada organismo da população O_i é selecionado aleatoriamente outro organismo O_j para então gerar dois novos organismos O_{inew} e O_{jnew}, conforme equações 1 e 2, respectivamente. O vetor de mutualismo, *MultialVetor*, representa as características do relacionamento de O_i e O_j e é obtido com o cálculo da média aritmética das dimensões dos organismos, calculado na Equação 3.

$$O_{inew} = O_i + rand(0,1) \times (O_{best} - MultialVetor \times BF_1) \quad (1)$$

$$O_{jnew} = O_j + rand(0,1) \times (O_{best} - MultialVetor \times BF_2) \quad (2)$$

$$MultialVetor = \frac{O_i + O_j}{2} \quad (3)$$

Onde as variáveis BF_1 e BF_2 podem assumir o valor 1 ou 2. O valor dessas variáveis é escolhido de forma aleatória caracterizando o não-determinismo, juntamente com os ponderadores aleatórios $rand(0,1)$. A variável O_{best} representa o melhor organismo da população atual. A operação $(O_{best} - MultialVetor * BF_2)$ nas Equações 1 e 2 representa 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, para cada organismo da população O_i, é gerado um novo organismo, O_{inew}, utilizando o melhor organismo da população, O_{best}, e outro organismo aleatório, O_j, conforme a Equação 4.

$$O_{inew} = O_i + rand(-1,1) \times (O_{best} - O_j) \quad (4)$$

A operação $(O_{best} - O_j)$ 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, O_{inew}, chamado de vetor parasita é gerado.Em seguida é selecionado um outro organismo de forma aleatória, O_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.

2.1.2 Diversidade Genotípica Populacional

Existe uma métrica, chamada diversidade genotípica, que identifica o grau de heterogeneidade ou homogeneidade entre os indivíduos de uma população. Essa métrica serve para monitorar o balanceamento entre a intensificação e a diversificação. Esse balanceamento é fundamental para o bom desempenho de um algoritmo populacional. A intensificação está relacionada com a concentração das buscas numa determinada região do espaço de solução que é considerada promissora. A diversificação está relacionada com a busca global. A mesma amplia as buscas por soluções em diferentes regiões visando identificar uma possível região promissora ainda não explorada. Quando o algoritmo apresenta elevada intensificação e baixa diversificação, caracterizado pela pouca diversidade genotípica na população de indivíduos, o processo de otimização se torna ineficiente. Isso porque a busca do algoritmo não irá contemplar diferentes regiões no espaço de solução. A análise dos resultados no decorrer do processo de otimização irá considerar a diversidade genotípica da população. A medida de diversidade genotípica (MDG) pode ser calculada conforme Equação 5 (Corriveau et al.; 2013).

$$MDG = \frac{\sum_{i=1}^{TP-1} ln(1 + min_{j \in [i+1,TP]} \frac{1}{D} \sqrt{\sum_{k=1}^{D} (O_{i,k} - O_{j,k})^2})}{NMD} \quad (5)$$

Onde TP é o tamanho da população, NMD é o normalizador da métrica da diversidade. Essa variável mantém a maior diversidade genotípica obtida durante toda a otimização. O número total de dimensões de um indivíduo é representado por D, O_j e O_i são indivíduos da população e k representa uma dimensão do indivíduo.

2.2 Agrupamento de Dados

A tarefa de agrupamento tem como principal objetivo dividir um conjunto de objetos de dados em diferentes grupos Aljarah (2013). Seja G uma base de

dados, o agrupamento consiste em encontrar partições de G, denotadas por $\tau(G)$, de tal forma que $\tau(G) = \{c_1, c_2, ..., c_k\}$, onde cada c_i representa um centroide e k representa o número de centroides. Não deve existir grupos vazios e os mesmos devem ser disjuntos. Além disso, a união dos grupos devem formar G. A função utilizada para realizar o agrupamento é chamada de função *fitness* no contexto de algoritmos populacionais. Quatro dessas funções referenciadas por $f1, f2, f3, f4$ estão disponíveis em Aljarah (2013). A função $f5$ é a versão simplificada da função de Silhueta disponível em (Ferrari et al.; 2014).

Os cálculos comuns para funções $f1$, $f2$, $f3$ são expostos a seguir. O cálculo da distância intragrupo está definida na Equação 6.

$$IntraD_i = \sum_{i=1}^{|cr_j|} Distance(cr_{ji}, c_j) \tag{6}$$

Onde cr_{ji} representa os itens de dados pertencente ao centroide c_j e $|cr_j|$ representa a quantidade de itens que pertencem ao centroide c_j. A distância utilizada na Equação 6, 7, 8 é a Euclidiana. A distância intergrupo é calculada conforme Equação 7.

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

Onde k é o número de centroides e c_i, c_j representam os centroides do organismo. O cálculo do erro, *Sum Squared Error* (SSE), é calculado na Equação 8.

$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{|c_i|}(Distance(c_i, c_j))^2 \tag{8}$$

Onde $|c_i|$ representa a quantidade de itens cobertos pelo centroide c_j e c_i representam itens cobertos por c_j. Sendo assim, a função $f1$ é definida pela Equação 9, $f2$ conforme Equação 10 e $f3$ conforme Equação 11.

$$f1 = \frac{InterDist}{SSE} \times \frac{IntraD_i}{max(IntraD_i)} \tag{9}$$

$$f2 = \frac{\frac{InterDist}{IntraD_i}}{max(IntraD_i)} \tag{10}$$

$$f3 = \frac{1}{SSE \times \frac{IntraD_i}{max(IntraD_i)}} \tag{11}$$

As funções $f1$ e $f2$ consideram a distância intergrupo em suas equações, sendo que a função $f2$ dá um peso maior para a distância intergrupo. As três funções consideram a maior distância intragrupo em suas equações. A distância intragrupo apresentada nas funções acima representa a média dessa distância de todos os grupos da solução. Além disso, as três funções são de maximização, ou seja, quanto maior seu valor melhor é a solução representada pela solução candidata. A função $f4$ é uma função de minimização e não utiliza nenhuma das equações apresentadas anteriormente e é definida conforme Equação 12.

$$f4 = \frac{\sum_{j=1}^{k}\frac{\sum_{i=1}^{n_i} Distance(R_i, C_j)}{n_j}}{k} \tag{12}$$

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 na Equação 12 é a distância de *Manhattan*.

A função $f5$ é uma função de maximização e pode ser definida conforme a Equação 13 (Ferrari et al.; 2014).

$$f5 = \frac{1}{n}\sum_{i=1}^{n}\frac{b(i) - a(i)}{max(a(i), b(i))} \tag{13}$$

Onde n é o número de itens de dados, $a(i)$ é a distância do item i até o centroide do grupo que esse item pertence e $b(i)$ é a distância do item i até os centroides dos outros grupos. A distância utilizada é a distância Euclidiana. Os resultados dessa função variam no intervalo $[-1, 1]$. Para que seja possível avaliar qual das funções possui melhor qualidade no agrupamento é necessário definir as métricas de qualidade. Para que seja possível avaliar qual das funções possui melhor qualidade no agrupamento é necessário definir as métricas de qualidade.

A métrica de pureza *FScore* é utilizada para avaliar a qualidade dos grupos formados (Zhao and Karypis; 2002). A mesma utiliza a informação do rótulo do dado, ou classe, na realização do cálculo da pureza. Dessa forma, a base de dados precisa ser rotulada para utilizar essa métrica.

Dada uma classe da dados representada por L_r, o seu tamanho é referenciado por n_r. Para um grupo identificado por S_i e tamanho n_i, suponha n_i^r registros no grupo S_i pertencente a L_r, então o valor F dessa classe nesse grupo é definida pela Equação 14[12].

$$F(L_r, S_i) = \frac{2 \times R(L_r, S_i) \times P(L_r, S_i)}{R(L_r, S_i) + P(L_r, S_i)} \tag{14}$$

Onde $R(L_r, S_i) = n_i^r/n_r$ e $P(L_r, S_i) = n_i^r/n_i$. O *FScore* da classe L_r é o máximo valor F encontrado em qualquer grupo, conforme Equação 15.

$$FScore(L_r) = max_{S_i \in T} F(L_r, S_i) \tag{15}$$

Onde T é o conjunto de agrupamentos. Enfim, a pureza de todo agrupamento é calculada considerando a média de todas as classes, conforme a Equação 16.

$$FScore(L_r) = \sum_{r=1}^{c}(\frac{n_r}{n})FScore(L_r) \quad (16)$$

Onde c é o número total de classes. Uma solução de agrupamento perfeita é aquela cujo resultado da Equação 16 é igual a 1 (Zhao and Karypis; 2002). Isso indica que existe pelo menos um grupo de cada classe totalmente homogêneo.

Definida a métrica de qualidade dos grupos, é necessário avaliar o desempenho do MRCSOS com relação a escalabilidade no uso dos recursos computacionais.

2.3 *Hadoop*

Hadoop é uma ferramenta de código aberto utilizada para armazenar e manipular grandes massas de dados (Nurain et al.; 2012). 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 compartilhado é provido pelo componente HDFS (Katal et al.; 2013). 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 (Narayan et al.; 2012).

HDFS é um sistema de arquivos projetado para armazenar grandes quantidades de dados com um padrão contínuo de acesso e em *clusters*. HDFS possui arquitetura cliente-servidor e utiliza o protocolo de comunicação *Transmission Control Protocol* (TCP). 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 no sistema de arquivos, 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* tem função de 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. Por exemplo, casos que envolvem o contexto de baixa latência de acesso a dados visto que o tempo requerido para inicialização do serviço, divisão e junção das tarefas, além da comunicação entre os nós, certamente inviabiliza a utilização do mesmo (Katal et al.; 2013).

MapReduce é um modelo de programação que possui principalmente duas funções chamadas de *map* e *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. Existem duas fases de execução no processamento *MapReduce*. A primeira fase executa 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*. Um *job MapReduce* representa um processo completo de execução do *framework MapReduce* num determinado arquivo HDFS.

Visto que os conceitos que embasam a linha de pesquisa deste trabalho foram descritos, a próxima seção apresenta o algoritmo desenvolvido.

3 MRCSOS

Esta seção detalha como o algoritmo SOS é projetado na arquitetura *Hadoop MapReduce* para agrupamento de dados. Para uma macrovisão do fluxo de processos existentes na execução do MRCSOS é apresentado o fluxograma da Figura 1.

Nele também é possível identificar quais processos são executados em paralelo, ou seja, via *job Hadoop MapReduce* e quais são executados de maneira sequencial.

No fluxograma é possível identificar a fase de inicialização, grupo de processos tracejado em azul, contendo os processos de recebimento dos parâmetros: número de centroides, tamanho da população, arquivo de *cache* distribuído, arquivo de dados e quantidade 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 D-dimensional, onde D representa a quantidade de atributos da base de dados.

Os valores dos centroides são inicializados com itens selecionados aleatoriamente 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 sejam válidas. Ainda nessa fase é calculada a função *fitness* de cada organismo da população. Percebe-se que o processo que calcula o *fitness* dos organismos está destacado em verde informando que esse processo é executado em paralelo.

Na sequência, o fluxo de execução entra na fase de processamento, grupo de processos tracejado em vermelho, possuindo os processos que abrangem as relações simbióticas de mutualismo, comensalismo e parasitismo, além do processo de seleção do melhor organismo.

Os processos de cálculo do *fitness* presentes no mutualismo, parasitismo e comensalismo estão destacados em verde indicando sua execução em paralelo. Percebe-se também nessa fase a existência de um processo decisório indicando um ciclo de iterações. O ciclo é executado até que a quantidade máxima de iterações seja alcançada.

O fluxo de execução chega na fase de finalização, grupo de processos tracejado de amarelo, onde está presente o processo de cálculo da pureza que é executado em paralelo. Ao final desse processo é obtida a pureza indicando o nível de qualidade do agrupamento formado.

Na próxima seção serão descritos os experimentos e apresentados os resultados obtidos, além da análise comparativa com resultados de outras abordagens.

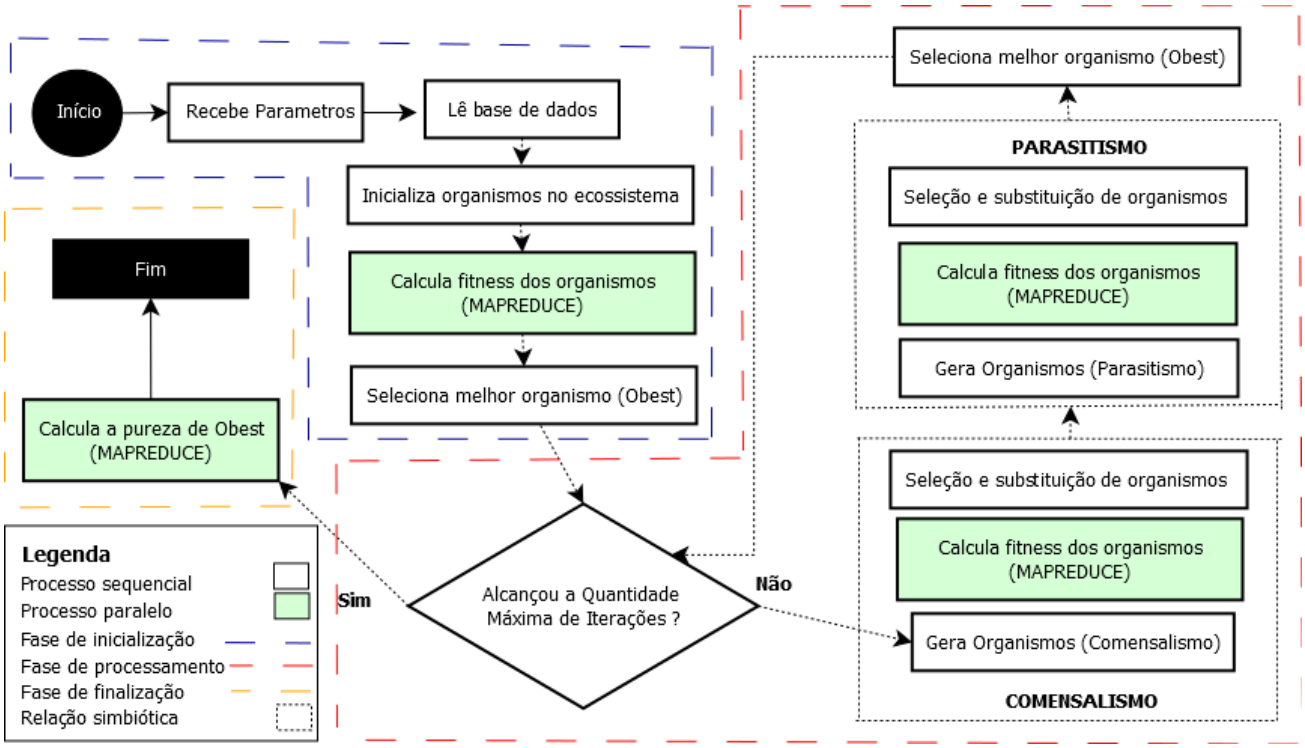

Figura 1: Fluxograma MRCSOS

4 Experimentos, Resultados e Análises

Os experimentos foram realizados com três bases de dados amplamente utilizadas na literatura: *MAGIC Gamma Telescope*, *Poker Hand* e *Electricity*. A base *MAGIC Gamma Telescope* possui 19.020 instâncias de dados, 10 atributos previsores, atributo-meta com 2 classes, 3 *Megabytes* e não existe ausência de valores nos atributos. A mesma está disponível no *UCI Machine Learning Repository* (https://archive.ics.uci.edu/ml/index.html). A base *Poker Hand* também está no repositório *UCI* e possui 10 atributos que representam 5 cartas. Essa base possui 1.025.010 instâncias de dados, 10 atributos previsores, atributo-meta com 10 classes, 25,2 *Megabytes* e não possui valores faltantes. A base de dados *Electricity* possui 45.312 instâncias de dados, 8 atributos previsores, atributo-meta com 2 classes, 6 *Megabytes* e está disponível no *Massive Online Analysis (MOA)* (http://moa.cms.waikato.ac.nz/datasets/).

Todos os resultados foram coletados utilizando um *cluster Hadoop* com 9 computadores. Cada computador possui a seguinte configuração: 8 *Gigabytes* de *RAM*, processador *Intel(R) Core(TM)* i7-4770 *CPU* 3.40 *GHz*, Sistema Operacional Ubuntu 14.04 LTS 64-bit, *Hadoop* versão 2.7.1 e Java 1.7.

O algoritmo MRCSOS foi desenvolvido na linguagem de programação Java. Os parâmetros utilizados foram: tamanho da população igual a 100, número de execuções igual a 10, número máximo de iterações igual a 100. Com esta configuração tem-se um total de 40.000 avaliações de função *fitness* por execução. A parametrização foi definida dessa forma para que a quantidade total de avaliações por execução não fosse superior a quantidade de avaliações utilizadas em Aljarah (2013) que é de 100.000 avaliações por execução e por questões de disponibilidade de uso dos recursos computacionais.

Dois tipos de análises são realizadas considerando os resultados obtidos. A primeira análise refere-se a qualidade dos agrupamentos encontrados e a segunda análise está relacionada com o desempenho do MRCSOS.

4.1 Análise de Qualidade

A análise de qualidade dos resultados obtidos contempla a métrica de pureza, Equação 16, resultante de 10 execuções de cada algoritmo para cada base de dados. Além disso, é analisada a correlação entre diferentes funções *fitness* do agrupamento, diversidade genotípica e pureza no decorrer do processo de agrupamento de dados.

Os resultados das médias de pureza dos agrupamentos estão exibidos na Tabela 1 onde a primeira coluna informa o algoritmo empregado. Os resultados obtidos pelos algoritmos K-Means, MRCPSO e MRCGSO não informam o desvio padrão e foram coletados de Aljarah (2013). Na segunda coluna são exibidos os resultados de pureza na base *Magic*, onde os mesmos correspondem a média e desvio padrão e melhor pureza encontrada dentre todas as execuções destacada entre parênteses. Na terceira coluna são exibidos os resultados na mesma estrutura da segunda coluna, entretanto, para a base *Electricity*. Da mesma forma estão descritos na quarta coluna os resultados utilizando a base *Poker Hand*.

As Tabelas 2, 3 e 4 apresentam os p-valores do teste estatístico de Kruskal-Wallis Hollander and Wolfe (1999). Este teste não paramétrico foi selecionado devido a refutação da hipótese nula de que as amostras dos dados pertencem a uma distribuição normal. Toda análise estatística considera uma confiança de

90%. Em negrito estão anotadas a existência de diferença estatística entre os resultados obtidos.

Em relação aos resultados na base *Magic*, as médias de pureza obtidas pelo K-Means, MRCPSO e MRCGSO são superiores à média de pureza das versões de MRCSOS(f1) e MRCSOS(f4). Contudo, as médias dos resultados obtidos nesta base utilizando as versões MRCSOS(f2), MRCSOS(f3) e MRCSOS(f5) são as maiores considerando os algoritmos demais algoritmos. De acordo com a Tabela 2, verifica-se também que os resultados obtidos pelas versões MRCSOS(f2), MRCSOS(f3) e MRCSOS(f5) são estatisticamente equivalentes. Dentre essas abordagens, a MRCSOS(f5) se mostrou ser a mais estável por possuir o desvio padrão igual a 0.

Na base *Electricity*, as médias da pureza obtida pelas versões MRCSOS(f1), MRCSOS(f2) e MRCSOS(f3) são inferiores às médias obtidas pelas versões MRCPSO e MRCGSO. As versões MRCSOS(f4) e MRCSOS(f5) apresentam média de pureza superior a dessas abordagens. Porém, dados os resultados do teste estatístico exposto na Tabela 3, evidencia-se diferença com significância somente entre MRCSOS(f1) e MRCSOS(f5). Nesta base, todas as versões de MRCSOS obtiveram melhores resultados que o K-Means.

Os resultados obtidos utilizando a base *Poker Hand* evidenciam que a abordagem MRCGSO encontrou o melhor resultado de pureza. O segundo melhor resultado nessa base foi obtido pela abordagem MRCPSO. Todas as versões de MRCSOS obtiveram resultados melhores que o K-Means. Dentre as versões de MRCSOS, a versão MRCSOS(f5) obteve o melhor resultado da média de pureza. Entretanto, evidencia-se pelos resultados do teste estatístico (Tabela 4) que existe equivalência no desempenho das abordagens MRCSOS(f2), MRCSOS(f3), MRCSOS(f4) e MRCSOS(f5). Nessa base, os experimentos com MRCSOS(f5) foram ampliados para fins de análise de convergência e os resultados obtidos foram 0.50 ± 0.70 para um total de 80.000 avaliações de *fitness*.

Visto os resultados de pureza conforme Tabela 1, percebe-se que a abordagem MRCSOS(f5) se sobressai diante das demais na busca por soluções de agrupamento de dados com alto grau de pureza. Apesar de obter o terceiro melhor resultado de pureza na base *Poker Hand* a mesma obteve os melhores resultados de pureza juntamente com MRCSOS(f2) e MRCSOS(f3) na base *Magic*. MRCSOS(f5) também obteve o melhor resultado da média de pureza na base *Electricity*. Nota-se, dentre os algoritmos analisados, que MRCSOS(f5) tende a encontrar os melhores resultados de pureza nas bases experimentadas.

Aplicando o ranqueamento estatístico de Friedman tem-se a seguinte ordem considerando os dados da Tabela 1, do melhor geral para o pior geral: MRCSOS(f5), MRCGSO, MRCSOS(f3), MRCSOS(f2), MRCPSO, MRCSOS(f4), MRCSOS(f1) e K-Means.

Tabela 1: Resultados obtidos para a pureza dos agrupamentos.

	Magic	*Electricity*	*Poker Hand*
K-Means	0.60	0.51	0.11
MRCPSO	0.65	0.58	0.51
MRCGSO	0.66	0.58	0.53
MRCSOS(f1)	0.54±0.305 (0.55)	0.54±0.249 (0.56)	0.19±0.123 (0.22)
MRCSOS(f2)	0.69±0.001 (0.69)	0.54±0.278 (0.56)	0.34±0.218 (0.39)
MRCSOS(f3)	0.69±0.008 (0.69)	0.55±0.303 (0.56)	0.28±0.236 (0.32)
MRCSOS(f4)	0.56±0.189 (0.56)	0.61±0.335 (0.64)	0.18±0.029 (0.19)
MRCSOS(f5)	0.69±0.000 (0.69)	0.67±0.000 (0.67)	0.35±0.256 (0.39)

Tabela 2: *p*-valores do teste Kruskal-Wallis para a base Magic.

MRCSOS	F1	F2	F3	F4
F2	0.1178	-	-	-
F3	**0.0513**	0.3276	-	-
F4	0.1985	**0.0210**	**0.0066**	-
F5	0.3111	0.2441	0.1272	**0.0901**

Tabela 3: *p*-valores do teste Kruskal-Wallis para a base Electricity

MRCSOS	F1	F2	F3	F4
F2	0.4388	-	-	-
F3	0.3790	0.4388	-	-
F4	0.1777	0.2206	0.2689	-
F5	**0.0953**	0.1240	0.1584	0.3501

Tabela 4: *p*-valores do teste Kruskal-Wallis para a base Poker Hand.

MRCSOS	F1	F2	F3	F4
F2	**0.0443**	-	-	-
F3	0.1747	0.2216	-	-
F4	0.3737	**0.0837**	0.2697	-
F5	**0.0791**	0.3854	0.3172	0.1381

4.2 Correlações

Após realizar a análise da qualidade do agrupamento de dados considerando a métrica de pureza, a seguir é descrita a análise de correlação do comportamento entre as funções *fitness*, diversidade genotípica e pureza.

Esta análise correlaciona o gráfico de convergência (iterações *versus fitness*) das diferentes funções de agrupamento com seus respectivos gráficos de diversidade genotípica e pureza. O objetivo é observar o comportamento destas funções com relação a estes indicadores. Ou seja, avaliar a qualidade das funções de agrupamento segundo o grau de pureza durante a evolução do processo de otimização considerando também a diversidade das soluções envolvidas nesse processo.

O eixo-x nos gráficos representa a quantidade de iterações. Nos eixos-y estão os valores da função *fitness*, os valores da diversidade genotípica e os percentuais das purezas do agrupamento, respectivamente. Vale ressaltar que os gráficos foram gerados utilizando a média de 10 execuções.

O gráfico de *fitness* serve para evidenciar a evolução da função *fitness* à medida que as iterações vão acontecendo. Uma atenção deve ser dada ao gráfico de pureza na análise de instabilidades. A instabilidade mencionada refere-se ao comportamento oscilante do valor da métrica de qualidade. Ou seja, quando o comportamento do resultado da métrica analisada não é constante crescente ou decrescente no decorrer das iterações (comportamento não-monotônico).

Quando isso ocorre, indica que a melhoria da função *fitness* nem sempre reflete na melhoria da qualidade do agrupamento. Por vezes, em alguns gráficos de pureza haverá um marcador indicando tais instabilidades. Além disso, deve-se atentar para o resultado final da pureza na última iteração do processo, verificando se existe algum resultado nas iterações anteriores melhor que o resultado final.

O gráfico da diversidade genotípica auxilia a análise e informa quanto o processo de otimização ainda possui de capacidade para evoluir. Desta maneira, se a diversidade genotípica não estiver próxima de 0 e a função *fitness* não estiver estagnada, o processo de otimização pode evoluir em mais iterações. Cada figura pode considerar o cenário de mais de uma função. Isso indica que as mesmas possuem comportamento com características semelhantes umas das outras na base de dados em consideração.

Nos resultados das funções f1 e f4 na base *Magic*, conforme Figura 2, percebe-se que mesmo com melhora constante da função *fitness* (comportamento monotônico) existe instabilidade acentuada na pureza no início do processo. Sendo assim, neste período é onde se apresenta o melhor resultado de pureza. Porém, nas iterações seguintes ocorre piora sistemática da pureza.

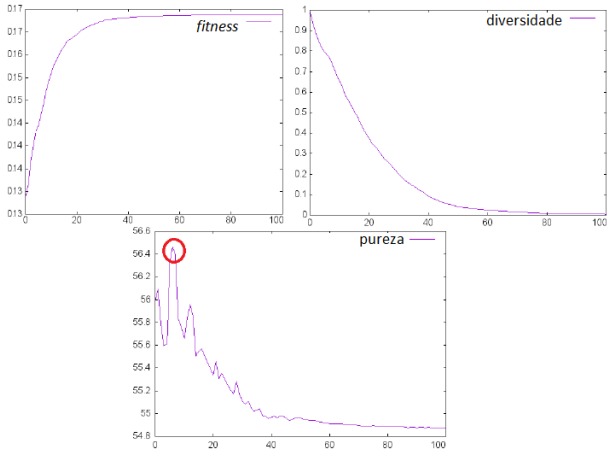

Figura 2: Gráficos da evolução do *fitness*, diversidade genotípica e pureza, características das funções f1 e f4 na base *Magic*.

No gráfico da diversidade genotípica pode-se perceber que, no decorrer do processo, a diversidade é decrementada constantemente e no final do processo não existe mais diversidade genotípica. Visto que, ao final das 100 iterações não existe mais diversidade genotípica e a função *fitness* aparenta estar estagnada, pode-se entender que o sistema convergiu para um atrator local.

Na Figura 3, os gráficos característicos das funções f2, f3 e f5 demonstram que não ocorre instabilidade na pureza. A pureza sempre melhora à medida que o *fitness* melhora. A diversidade genotípica é decrementada constantemente até o final do processo de otimização.

O próximo cenário, conforme Figura 4, caracteriza o comportamento dos resultados obtidos pelas funções f1, f2, f3, f4 na base *Electricity*.

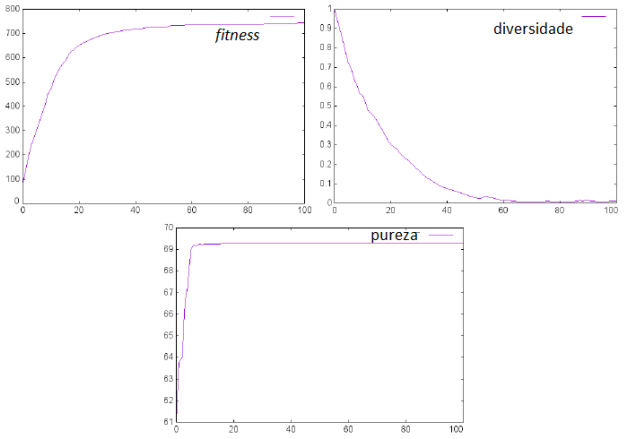

Figura 3: Gráficos da evolução do *fitness*, diversidade genotípica e pureza, características para as funções f2, f3 e f5 na base *Magic*.

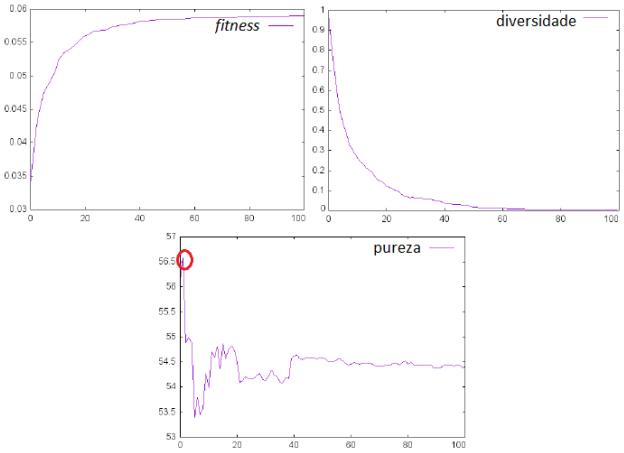

Figura 4: Gráficos da evolução do *fitness*, diversidade genotípica e pureza, características para as funções f1, f2, f3 e f4 na base *Electricity*.

A função f5 é a única função na base *Electricity* cujo melhor resultado de pureza é mantido até o final do processo, ou seja, não ocorre perda de qualidade, conforme Figura 5. A pureza dessa função não apresenta instabilidade. A diversidade genotípica na iteração 100 ainda existe e a função *fitness* ainda não parou de convergir. Isso indica que o sistema ainda possui a capacidade de evoluir se for aumentado o número de iterações.

A Figura 6 exibe o comportamento das funções f1, f2, f3, f4 na base *Poker Hand*. O gráfico de pureza exibido nessa figura demonstra que ocorrem algumas instabilidades, mesmo com a tendência do comportamento ascendente. Apesar disso, a pureza no final do processo é inferior que a pureza obtida em iteração anterior. No final do processo ainda existe diversidade genotípica.

Dentre as funções aplicadas, a f5 é a função que obteve maior média de pureza utilizando a base *Poker Hand* (Figura 7). Além disso, essa função foi a única cuja pureza encontrada no final do processo de otimização é a melhor pureza considerando todo o processo de otimização. A diversidade genotípica

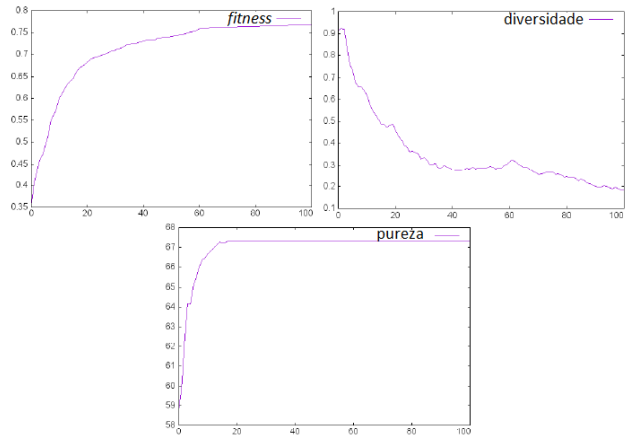

Figura 5: Gráficos da evolução do *fitness*, diversidade genotípica e pureza, características para a função f5 na base *Electricity*.

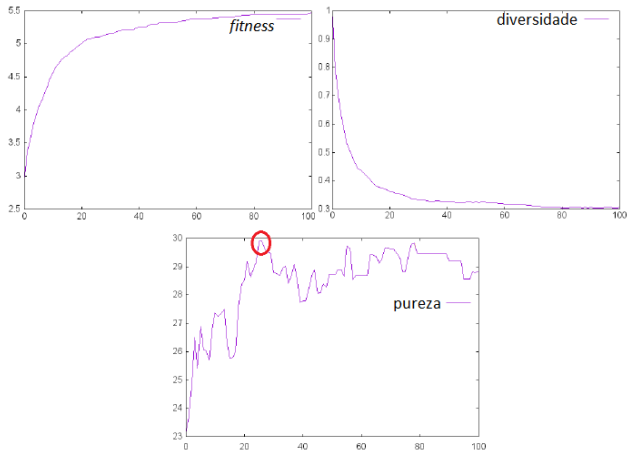

Figura 6: Gráficos da evolução do *fitness*, diversidade genotípica e pureza, características para as funções f1, f2, f3 e f4 na base *Poker Hand*.

final ficou bastante elevada, em torno de 0,77 e a função *fitness* ainda tem potencial de melhora.

Nas bases *Magic* e *Electricity*, o comportamento da função *fitness* em 100 iterações indica que as 5 funções convergiram. A diversidade genotípica é praticamente inexistente ao final das 100 iterações em todas as versões do MRCSOS. Contudo, na base *Poker Hand* a diversidade genotípica se mantém alta nas últimas iterações, especialmente a versão MRCSOS(f5).

Dessa forma, entende-se que a quantidade de iterações igual a 100 é suficiente nas bases *Magic* e *Electricity* pois o *fitness* consegue convergir e não existe mais diversidade genotípica. Ao contrário do que ocorre nessas duas bases, a não convergência da função *fitness* e a existência de diversidade genotípica na base *Poker Hand* indicam que 100 iterações não são suficientes para o algoritmo evoluir e explorar a diversidade de soluções ainda existentes. Dessa forma, foi realizado um experimento com 200 iterações, totalizando 80.000 avaliações de *fitness*, em cada uma das 10 execuções exclusivamente para a versão MRCSOS(f5) na base *Poker Hand*. A média da pureza obtida foi de 0.5 com desvio padrão de 0.70. Desta forma,

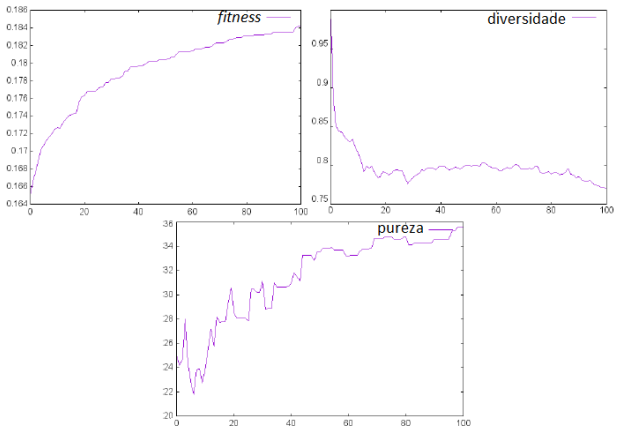

Figura 7: Gráficos da evolução do *fitness*, diversidade genotípica e pureza, características para a função f5 na base *Electricity*.

os resultados obtidos com MRCSOS(f5) com 200 iterações se mostra competitivo com os obtidos pelos algoritmos da literatura (Tabela 1).

Em relação ao comportamento da pureza, percebe-se que mesmo com a constante evolução da função *fitness*, as versões MRCSOS(f1), MRCSOS(f2), MRCSOS(f3) e MRCSOS(f4) apresentaram resultados com grau de pureza pior ao final do processo quando comparados aos resultados obtidos em iterações anteriores nas bases *Electricity* e *Poker Hand*. O mesmo comportamento ocorre com os resultados de pureza obtidos pelas versões MRCSOS(f1) e MRCSOS(f4) na base *Magic*. Apenas nessa base as versões MRCSOS(f2), MRCSOS(f3) e MRCSOS(f5) conservaram o melhor resultado de pureza até o final do processo de agrupamento. Foi percebido que, pelo fato das funções de agrupamento f2 e f3 utilizaram a razão entre a maior distância intra-grupo e a distância $intraD_i/max(intraD_i$ no denominador de suas equações, conforme Equação 10 e 11, respectivamente, as mesmas alcançaram o comportamento de pureza crescente assintótico na base *Magic*. Exclusivamente nessa base, quando o resultado de $intraD_i/max(intraD_i$ diminuí, o resultado das funções f2 e f3 aumenta, indicando a melhoria da função. Quando isso ocorre, a pureza do agrupamento também melhora. Em todos os casos citados onde ocorre a perda de qualidade, a instabilidade está presente no comportamento dos gráficos de pureza. Assim, entende-se que a presença da instabilidade no comportamento dos gráficos de pureza é um indício que pode ocorrer perda de pureza no decorrer do processo de agrupamento de dados.

A versão MRCSOS(f5) é a única que obteve média de pureza final superior do que em qualquer iteração anterior nas bases *Magic*, *Electricity* e *Poker Hand*. A função f5 valoriza os agrupamentos que possuem itens de dados cobertos por um centroide que são mais afastados do segundo centroide mais próximo, conforme Equação 13. Assim, essa função contempla no seu resultado a proximidade com itens similares, cálculo do item até o centroide mais próximo e distância com itens não-similares ao considerar a distância com o segundo centroide mais próximo. Atribui-se a essa estratégia como critério de qualidade de agrupamento para a não ocorrência da perda de pureza na obtenção de seus resultados finais. A in-

fluência da distância de um item de dado ao segundo centroide mais próximo está presente apenas nessa função. Entende-se que a influência dessa distância contribuiu para que os resultados da versão MRCSOS(f5) sejam os melhores resultados de qualidade do que os resultados obtidos pelas outras versões do MRCSOS. Finalizada a análise de qualidade, a seguir é apresentada a análise de desempenho do algoritmo SOS na arquitetura *Hadoop/MapReduce*.

4.3 Análise de Desempenho

Essa seção visa analisar o desempenho da abordagem proposta, MRCSOS, quando a mesma é executada em paralelo. *Speedup* é a métrica de avaliação de desempenho utilizada nessa análise e expressa quanto mais rápida é a execução de um programa em uma arquitetura paralela do que quando executado de maneira sequencial, definida na Equação 17.

$$Speedup = \frac{T}{T_n} \tag{17}$$

Onde T representa o tempo de processamento do algoritmo sequencial (sem paralelismo) e T_n é o tempo de execução do algoritmo na arquitetura *MapReduce* utilizando n computadores. A versão MRCSOS(f5) é a versão analisada nesta Seção. A escolha é justificada pelo fato desta apresentar os melhores resultados de qualidade de pureza nas bases analisadas.

O tempo de processamento, medido em minutos, gasto por MRCSOS(f5) considerando as três bases é apresentado a seguir. Os experimentos que geraram os resultados de desempenho foram executados em um *cluster Hadoop* com 1, 2, 4 e 8 *workers* ou *datanodes*. Os resultados de tempo com a média de 10 execuções estão dispostos na Tabela 5.

Percebe-se, de forma geral, que à medida que o número de computadores é incrementado o tempo de processamento é reduzido.

Tabela 5: Resultado da média de tempo de execução, medido em minutos da função f5.

Nº de Computadores	*Magic*	*Electricity*	*Poker Hand*
1	256.7	271.2	1548.6
2	198.3	198.4	1053.6
4	174.4	159.3	811.9
8	161.9	147.2	723.7

Com os tempos de processamento obtidos por MRCSOS(f5) utilizando 1, 2, 4 e 8 computadores, foram calculados os *speedups* apresentados na Tabela 6.

Tabela 6: Resultados de *speedups*.

Algoritmo	*Magic*	*Electricity*	*Poker Hand*
2	1.29	1.36	1.48
4	1.47	1.70	1.90
8	1.59	1.84	2.14

Nas três bases de dados, à medida que os recursos são adicionados ao processamento do MRCSOS(f5) o valor do *speedup* sempre aumenta. Esse aumento indica que, com a adição de recursos, há um ganho de tempo de processamento em relação ao processamento com menos recursos. Além disso, o fato de todos os valores de *speedup* exibidos na Tabela 6 possuírem valores maior que 1, evidencia que os experimentos em paralelo foram executados mais rápidos que de maneira sequencial.

Percebe-se que a redução do tempo de execução utilizando a maior base experimentada, base *Poker Hand*, foi a maior obtida nos experimentos. Esse fato vai de encontro com a recomendação da utilização do *Hadoop* em grandes bases de dados.

Na análise de desempenho percebeu-se que no momento da execução do *job MapReduce* não utilizou-se 100% dos recursos disponíveis de todo o *cluster*. Apesar da configuração do *Hadoop* permitir a utilização de todo o recurso para processamento, foi visto que quanto menor o *cluster* utilizado nos experimentos maior é o percentual de tempo da execução do *job MapReduce* utilizando 100% dos recursos do *cluster*. Recomenda-se que o *Hadoop* seja utilizado com volume de dados maior para que se possa obter melhores resultados de *speedup*.

De maneira geral, a aplicação do algoritmo SOS na arquitetura paralela foi fundamental para a obtenção dos resultados.

5 Conclusões e Trabalhos Futuros

Este trabalho apresenta a aplicação de uma abordagem de agrupamento de dados que utiliza o algoritmo bio-inspirado SOS projetado na infraestrutura *Hadoop MapReduce* (MRCSOS).

São relatados experimentos com MRCSOS utilizando cinco funções (f1, f2, f3, f4 e f5) de *fitness* em três bases de dados no intuito de analisar a média final da pureza do agrupamento. Além disso, também foi analisado o desempenho do algoritmo SOS em relação a utilização dos recursos fornecidos pelo *Hadoop MapReduce*. Contudo, a principal contribuição deste trabalho é a análise que correlaciona o gráfico de convergência (iterações *versus fitness*) das diferentes funções de agrupamento com seus respectivos gráficos de diversidade genotípica e pureza.

Em relação a análise comparativa dos resultado finais de pureza, os resultados da versão MRCSOS(f5) foi a que obteve os melhores resultados nas bases *Magic* e *Electricity*. Na *Poker Hand*, a versão MRCSOS(f5) obteve resultados competitivos aos resultados obtidos pelas abordagens MRCPSO e MRCPSO. De forma geral, os resultados finais de pureza da versão MRCSOS(f5) são os melhores obtidos dentre as outras quatro versões de MRCSOS.

Em relação ao comportamento do curso da pureza das diferentes funções analisadas espera-se que sempre ocorra melhoria contínua à medida que o processo de otimização vai evoluindo. Entretanto, o comportamento dos resultados de pureza das funções f1, f2, f3 e f4 demonstram a perda de pureza pois a pureza final é inferior a pureza obtida em iterações prévias. Esse comportamento evidência que ocorre perda de pureza de agrupamento devido a não identificação da melhor solução encontrada por parte dessas funções ao longo de todo o processo. Contudo, o comportamento dos resultados de pureza obtidos pela função f5 demonstram que esta consegue conservar o melhor

resultado de pureza até o final do processo nas três bases analisadas.

O desempenho do algoritmo SOS na arquitetura *Hadoop MapReduce* não ficou próximo da linear. Na análise de desempenho foi identificado que quanto menor o *cluster* maior é o tempo de utilização de 100% dos recurso disponíveis. Esse comportamento pode ser justificado pela utilização de bases com pouco volume e pela maneira com a qual o *Hadoop* utiliza o sistema de arquivos HDFS com inúmeras chamadas de arquivos.

Em relação aos trabalhos futuros, pretende-se desenvolver uma abordagem do algoritmo SOS para realizar agrupamento de dados utilizando o *framework Spark*. Estudos indicam que o *Spark*, que trabalha no topo do sistema de arquivos HDFS do *Hadoop*, pode alcançar *speedups* ainda maiores. Além disso, pretende-se ampliar a quantidade de funções de agrupamentos analisadas e também aumentar o número de bases de dados com volume maior.

Referências

Aljarah, I. M. (2013). *MapReduce-enabled scalable nature-inspired approaches for clustering*, PhD thesis, North Dakota State University.

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

Corriveau, G., Guilbault, R., Tahan, A. and Sabourin, R. (2013). Review of phenotypic diversity formulations for diagnostic tool, *Applied Soft Computing* **13**(1): 9–26.

Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A. Y., Foufou, S. and Bouras, A. (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis, *Emerging Topics in Computing, IEEE Transactions on* **2**(3): 267–279.

Ferrari, D. G. et al. (2014). *Seleção de algoritmos para a tarefa de agrupamento de dados: uma abordagem via meta-aprendizagem*, PhD thesis, Universidade Presbiteriana Mackenzie.

Hollander, M. and Wolfe, D. A. (1999). Nonparametric statistical methods.

Kar, A. K. (2016). Bio inspired computing - a review of algorithms and scope of applications, *Expert Systems with Applications* **59**: 20–32.

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

Leskovec, J., Rajaraman, A. and Ullman, J. D. (2014). *Mining of massive datasets*, Cambridge University Press.

Menezes, S., Freitas, R. and Parpinelli, R. (2016). Mineração em grandes massas de dados utilizando hadoop mapreduce e algoritmos bio-inspirados: Uma revisão sistemática, **23**: 69.

Mohammed, E. A., Far, B. H. and Naugler, C. (2014). Applications of the mapreduce programming framework to clinical big data analysis: current landscape and future trends, *BioData mining* **7**(1): 22.

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

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

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

Rajaraman, S., Vaidyanathan, G. and Chokkalingam, A. (2015). Performance evaluation of bio-inspired optimization algorithms in resolving chromosomal occlusions, *Journal of Medical Imaging and Health Informatics* **5**(2): 264–271.

Zhao, Y. and Karypis, G. (2002). Evaluation of hierarchical clustering algorithms for document datasets, *Proceedings of the eleventh international conference on Information and knowledge management*, ACM, pp. 515–524.