Mineração em Grandes Massas de Dados Utilizando *Hadoop MapReduce* e Algoritmos Bio-inspirados: Uma Revisão Sistemática

Mining of Massive Data Bases Using Hadoop MapReduce and Bio-inspired algorithms: A Systematic Review

Sandro Roberto Loiola de Menezes [1]
Rebeca Schroeder Freitas [2]
Rafael Stubs Parpinelli [3]

Data de submissão: 24/08/2015, *Data de aceite: 06/02/2016*

Resumo: A Área de Mineração de Dados tem sido utilizada em diversas áreas de aplicação e visa extrair conhecimento através da análise de dados. Nas últimas décadas, inúmeras bases de dados estão tendenciando a possuir grande volume, alta velocidade de crescimento e grande variedade. Esse fenômeno é conhecido como *Big Data* e corresponde a novos desafios para tecnologias clássicas como Sistema de Gestão de Banco de Dados Relacional pois não tem oferecido desempenho satisfatório e escalabilidade para aplicações do tipo *Big Data*. Ao contrário dessas tecnologias, *Hadoop MapReduce* é um *framework* que, além de provêr processamento paralelo, também fornece tolerância a falhas e fácil escalabilidade sobre um sistema de armazenamento distribuído compatível com cenário *Big Data*. Uma das técnicas que vem sendo utilizada no contexto *Big Data* são algoritmos bio-inspirados. Esses algoritmos são boas opções de solução em problemas complexos multidimensionais, multiobjetivos e de grande escala. A combinação de sistemas baseados em *Hadoop MapReduce* e algoritmos bio-inspirados tem se mostrado vantajoso em aplicações *Big Data*. Esse artigo apresenta uma revisão sistemática de trabalhos nesse contexto, visando analisar critérios como: tarefas de mineração de dados abordadas, algoritmos bio-inspirados utilizados, disponibilidade das bases utilizadas e quais características *Big Data* são tratadas nos trabalhos. Como resultado, esse artigo discute os critérios analisados e identifica alguns modelos de paralelização, além de sugerir uma direção para trabalhos futuros.

[1]Programa de Pós-Graduação em Computação Aplicada, Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, Caixa Postal 631 - Joinville, Santa Catarina, Brasil. {sandrolmenezes@gmail.com}
[2]Departamento de Ciência da Computação, Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, Caixa Postal 631 - Joinville, Santa Catarina, Brasil. {rebeca.schroeder@udesc.br}
[3]Programa de Pós-Graduação em Computação Aplicada, Centro de Ciências Tecnológicas, Universidade do Estado de Santa Catarina, Caixa Postal 631 - Joinville, Santa Catarina, Brasil. {rafael.parpinelli@udesc.br}

***P*alavras-chave:** algoritmos bio-inspirados, Hadoop MapReduce, mineração de dados, big gata, revisão sistemática

Abstract: The Data Mining research field has been applied in several distinct areas and, in all of them, it aims to extract knowledge through data analysis. In recent decades, databases are tending to have large volume, high-speed growth rate and variety. This phenomenon is known as Big Data and represents new challenges for classical technologies such as Relational Database Systems. Unlike classical technologies, Hadoop MapReduce is a framework that, in addition to provide mechanisms for parallel processing, also offers fault tolerance and easy scalability over a distributed storage system compatible with Big Data. Some techniques that have been used in the context of Big Data are bio-inspired algorithms. These algorithms have being proved to be good solution options for complex multi-dimensional, multi-objectives and large scale problems. The mixing of Hadoop MapReduce and bio-inspired algorithms has proven to be advantageous in Big Data applications. This article presents a systematic review of Big Data applications in order to analyze: Which data mining tasks are being applied? What are the bio-inspired algorithms involved? Are the databases available to downlaod? and, What Big Data features are discussed in the works reviewed? As a result, this article discusses, analyzes and identifies some models of parallelization and suggests some directions for future research.

***K*eywords:** bio-inspired algorithms, Hadoop MapReduce, data mining, big data, systematic review

1 Introdução

Técnicas de mineração de dados têm recebido atenção por parte da comunidade científica devido aos benefícios que podem produzir em diversas áreas como Educação, Bioinformática, Economia, Negócios, entre outras. Tais benefícios podem ser obtidos através da descoberta de padrões, correlações e tendências de informações [47]. Esses padrões descobertos agregam conhecimento que podem ser úteis na tomada de decisão em um determinado processo. Na área de Negócios, por exemplo, dados de compras de clientes podem ser utilizados como fonte de mineração para obter um padrão de compras e assim embasar o planejamento estratégico de mercado [13]. Na área de segurança de redes, mineração de dados pode ser utilizada para auxiliar na detecção de invasão de intrusos [6]. No ramo das Ciências Biológicas, a mineração de dados contribui na descoberta de novos tipos de câncer [9]. Na Bioinformática, mineração de dados ajuda na identificação de espécies através da análise de DNA [14]. Na educação, mineração de dados tem descoberto novas técnicas de aprendizado e de avaliação de estudantes [47]. Além dessas áreas, mineração de dados vem

sendo introduzida em áreas como Análise de Dados Espaciais, Padrões Recognitivos, dentre outras [47].

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. Outro fator limitante é a dimensionalidade dos dados que influencia diretamente no desempenho do algoritmo: quanto maior a dimensionalidade de dados, maior será o tempo para processa-los [28]. 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 [35]. 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 na estação de processamento [35].

Percebe-se que existem alguns requisitos especificamente relacionados com grandes volumes de dados, como disponibilidade e desempenho eficiente, que tecnologias convencionais de mineração de dados não são adequadas para cumpri-las. Por isso é necessário desenvolver tecnologias inovadoras que possam cumprir de forma eficiente esses requisitos. Para alcançar esse avanço tecnológico é necessário intensificar pesquisas científicas no contexto de mineração de grandes volumes de dados. O grande aumento do volume de dados que vem ocorrendo nas duas últimas décadas, fomentado por várias fontes de dados como mídias sociais, sensores, arquivo de *logs* dentre outras, reforça a necessidade de mineração de dados em grandes massas de dados.

Volume de dados na ordem de *Petabytes*, que corresponde a 10^{15} *Bytes*, é uma das propriedades de *Big Data* [11]. Segundo a definição de [1], *Big Data* é definido como grande volume de informações ativas com alta velocidade e variedade que demanda custo e formas inovadoras para obter uma visão compreensível para a tomada de decisão . Complementarmente, *Big Data* é considerado a próxima fronteira da inovação, competição e produtividade[11]. Além de um grande Volume de dados, *Big Data* também possui as propriedades de Velocidade, Variedade, Veracidade e Valor. Essas propriedades de *Big Data* são conhecidas como 5 V's e serão detalhadas no corpo deste artigo. Para realizar mineração de dados em bases com propriedades *Big Data*, principalmente Volume, 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 escalabilidade pode ser vertical ou horizontal.

A escalabilidade vertical é obtida através do incremento de mais recursos computacionais (núcleos, memória, processamento) em um servidor original, incrementando assim o poder de processamento [56][38]. Já escalabilidade horizontal é obtida através do incremento de um grande número de servidores menores e mais baratos, fortemente interconectados em

formato de *cluster* [56][38]. A escolha desse tipo de escalabilidade vem crescendo pois é mais acessível e possui melhor desempenho de processamento em relação à escalabilidade vertical [56]. Assim, a linha de pesquisa deste trabalho considera o contexto de escalabilidade horizontal. Utilizando escalabilidade horizontal é possível realizar processamento distribuído e paralelo. Diante disso, nesse contexto é necessário utilizar técnicas com características mais adequadas não só ao processamento em paralelo como também ao cenário *Big Data*. Algoritmos bio-inspirados possuem conceitos relacionados a cooperação e coevolução que são vantajosas no contexto de paralelismo [44]. Além disso, esses algoritmos possuem algumas características que favorem a sua escolha para trabalhar com problemas de grande escala.

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 não-determinísticas para resolver problemas complexos de otimização [21]. Esses algoritmos englobam abordagens populacionais que processam um conjunto de indivíduos no qual cada indivíduo representa uma possível solução para o problema. Essa característica populacional tem contribuído na resolução de problemas de grande escala, alta dimensionalidade e multiobjetivos [12]. Outra característica vantajosa desses paradigmas bio-inspirados é o não determinismo aliado com a intensificação e diversificação. Isso implica em uma exploração mais eficiente do espaço de busca e em diferentes regiões [46]. Mineração em grandes massas de dados envolve espaço de busca de solução bastante extenso. Nesse caso, a diversificação pode beneficiar a busca da melhor solução. Soluções determinísticas tendem a falhar quando aplicadas em problemas que o espaço de busca é muito grande e tendem a aumentar consideravelmente [10] [54]. Mineração de dados em *Big Data* pode ser visto como um problema de otimização onde busca-se encontrar a melhor solução, como os valores dos centróides de uma tarefa de agrupamento, em um determinado tempo limite. Algoritmos Bio-inspirados são técnicas de pesquisa e otimização que possuem características adequadas para análise de dados com propriedades *Big Data* [12].

No contexto *Big Data*, é necessário embasar a técnica escolhida para realizar a mineração de dados numa infraestrutura que dê suporte de armazenamento e processamento para o desenvolvimento do mecanismo para mineração em grandes volumes de dados. Ferramentas convencionais como Sistemas de Gerenciamento de Banco de Dados Relacional não conseguem gerenciar *Big Data* com disponibilidade e desempenho eficientes [48] [19]. *Hadoop* é um projeto *open source* criado pela *Apache Software Fundation* que vem sendo utilizado como ferramenta no contexto *Big Data* [7]. A utilização dessa ferramenta no contexto *Big Data* é justificada pelo fornecimento de um *framework* de processamento em paralelo com tolerância a falhas, fácil escalabilidade e utilização (*Hadoop MapReduce*) [34] [52], além de um poderoso sistema de arquivos robusto e distribuído (*Hadoop Distributed File System (HDFS)*).

Existem outras tecnologias disponíveis que fornecem paralelismo que poderiam ser

utilizadas neste trabalho como *Graphical Processing Unit (GPU)*, *Grid Computing* e *Message Passing Interface (MPI)*. *GPU* é uma tecnologia orientada a transferência de dados, *throughput-oriented*. Essa tecnologia adapta-se bem aos problemas que envolvem transferência e carga de dados, *workload* [42]. *Grid Computing* é uma tecnologia que disponibiliza um conjunto de recursos, como estações de trabalho e rede com alta velocidade, para serem utilizadas sob demanda à medida que as tarefas de processamento são executadas. Essa tecnologia é recomendada na resolução de problema computacional de grande escala, pois provê grande desempenho [2]. *MPI* é uma interface eficiente e flexível que provê um padrão de comunicação seguro, veloz e de alto desempenho para aplicações distribuídas em *cluster* [37]. Aplicações *MPI* fornecem tolerância a falhas utilizando pontos de checagem, *checkpoints*, ocasionando o problema de excesso de armazenamento em disco [33]. Dentre essas tecnologias citadas, o *Hadoop* provê um mecanismo automático e mais eficiente de tolerância a falhas [39] [26] conciliado com escalabilidade e paralelismo de alto desempenho [53].

O objetivo desse artigo será alcançado através da revisão sistemática da literatura [29]. Essa revisão busca identificar quais algoritmos bio-inspirados são utilizados, como o algoritmo utiliza a infraestrutura fornecida pelo *Hadoop*, quais bases são utilizadas e, por fim, caracterizar os trabalhos encontrados em modelos de execução.

Esse artigo está estruturado da seguinte maneira. A Seção 2 explica os conceitos relacionados à mineração de dados e *Big Data*. A Seção 3 detalha Algoritmos Bio-Inspirados. Na Seção 4 é detalhada a estrutura do *Hadoop*. Na Seção 5 será apresentado o método de pesquisa científica adotado nesse artigo. Na Seção 6 será exposta a síntese dos trabalhos relacionados com mineração de dados utilizando algoritmos bio-inspirados e *Hadoop* . Na Seção 7 é exposta a discussão sobre os trabalhos revisados e na Seção 8 é feita a conclusão dessa pesquisa.

2 Mineração de dados e *Big Data*

Mineração de dados, também conhecido como Descoberta de Conhecimento em Banco de Dados (*Knowledge Discovery in Databases KDD*), é o processo de extração automática de padrões que representa conhecimento implícito armazenado ou capturado em grandes bases de dados, na *Web*, em fluxo de dados ou em outros repositórios de informações [24]. O processo de análise envolve técnicas matemáticas, estatísticas e computacionais. Processos ou tarefas no domínio de mineração de dados visam construir modelos eficientes capazes de analisar e extrair conhecimento além de predizer tendências futuras no comportamento dos dados. Tarefas de mineração de dados podem ser classificadas em duas categorias: preditiva e descritiva. Modelos preditivos aprendem através da análise de um conjunto de dados, ou dados de treinamento, e fazem predições no comportamento de novos dados. Já técnicas descritivas proveem sumarização de dados. Dentre as tarefas mais comuns de mineração de

dados é possível citar: Classificação, Agrupamento, Associação e Regressão [22]. As tarefas de Classificação e Agrupamento são detalhadas na sequência por serem abordadas nos trabalhos revisados.

A classificação é uma tarefa preditiva que infere a determinação de qual classe um novo dado está classificado. Basicamente, essa tarefa divide-se em duas fases. Na primeira fase ocorre o processo de aprendizagem. Nesta fase o modelo preditivo que define qual classe um determinado dado pertence é construído a partir da análise de um conjunto de dados, ou dados de treinamento. Na maioria das vezes os dados de treinamento são rotulados, ou seja, cada dado já possui a informação correta de qual classe ele pertence (atributo meta ou descritor alvo). Nesses casos a classificação é dita como supervisionada. Na segunda fase ocorre a avaliação do modelo em dados de teste, disjunto dos dados de treinamento, para encontrar a precisão do padrão de classificação [22] [30]. Aplicar essa tarefa em grandes massas de dados com número muito grande de atributos e com características redundantes pode trazer prejuízos a precisão de classificação [36]. Desta maneira, é comum antes de realizar a análise para gerar o modelo de classificação executar uma etapa de seleção de atributos que serão considerados pelo algoritmo de aprendizagem [23].

A tarefa de agrupamento tem como principal objetivo dividir um conjunto de objetos de dados não rotulados em diferentes grupos. Cada grupo é formado por objetos que possuem uma maior similaridade entre os mesmos. Para determinar se a técnica utilizada no processo de agrupamento é eficiente deve ser mensurada a qualidade dos grupos formados. Esta medida de qualidade é feita através do cálculo da similaridade entre os elementos de um mesmo grupo e de grupos diferentes. Quanto maior é a similaridade dos elementos de um mesmo grupo maior é a qualidade do agrupamento. Dessa forma a similaridade interna do grupo deve ser maximizada. Quanto menor for a similaridade de objetos de grupos diferentes, maior será a qualidade do agrupamento, logo essa similaridade deve ser minimizada [5].

Executar tarefas de Mineração de Dados no contexto *Big Data* é bastante dificultoso. Portanto, é importante entender o que significa *Big Data*. O termo *Big Data* pode ser definido como um cenário que envolve imensa quantidade de dados não estruturados que requer tecnologias não convencionais para analisar os mesmos. Além do volume e estrutura, outras características tornam *Big Data* um problema muito complexo para ser resolvido utilizando sistema de banco de dados convencional como Sistemas Gerenciadores de Banco de Dados Relacionais [28] [30] [48].

A necessidade de desenvolver novas tecnologias de processamento para *Big Data* surgiu em vários domínios devido ao aumento do poder computacional, capacidade de armazenamento e aumento na produção de conteúdo digital [17]. Pode-se citar como exemplo de domínio *Big Data* as áreas da Ciência, Telecomunicação, Indústria, Negócios, Planejamento urbano, Mídia social e Saúde. *Big Data* provê grande potencial no processo decisório baseado em dados podendo trazer benefícios como nova visão de negócio, habilidade de medir e

monitorar fatores influentes no negócio, descoberta de novas oportunidades de vendas dentre outros benefícios. Dessa forma, investimentos direcionados para pesquisa e desenvolvimento relacionados com *Big Data* vem ganhando espaço nas Universidades, governo e indústria. Apesar de existirem várias questões em discussão sobre *Big Data*, as propriedades ou características mais significativas do mesmo são identificadas a seguir [18]. Essas características são referenciadas pelo termo 5 V's: Volume, Velocidade, Variedade, Veracidade e Valor.

Em 2003 todo o volume de dados gerado e armazenado no mundo inteiro foi inferior a 1.8 *Zettabytes*. Em apenas dois dias no ano de 2011 o volume de dados gerado foi superior a 1.8 *Zettabytes*, conforme relatório fornecido pela *International Data Corporation (IDC)* [11]. A tendência é de aumentar o volume de bancos de dados corporativos em 40 % a cada ano. Entre outros fatores, os avanços e barateamento das tecnologias de coleta e armazenamento possibilitaram às aplicações computacionais fornecerem funcionalidades como operações de gravação e recuperação de dados com uma capacidade superior ao fornecido anteriormente. Além disso, o rápido desenvolvimento de redes e o aumento da capacidade de coleta de dados são outros fatores que influenciaram na tendência de aumento de volume de dados [51].

O Volume é a característica representada pela palavra *Big* de *Big Data*. Essa característica impõe requisitos específicos para tecnologias e ferramentas utilizadas atualmente para manipular *Big Data* [18]. No contexto *Big Data* atualmente os dados existentes estão em *Petabytes* e é previsível o aumento para *Zettabytes* num futuro próximo [28]. As redes sociais existentes estão produzindo dados na ordem de *Terabytes* todos os dias e essa quantidade de dados é difícil de ser manipulada usando sistemas tradicionais.

A Velocidade no contexto *Big Data* lida com a velocidade da chegada de dados de várias fontes e com a velocidade em que os dados fluem [28]. Nesse caso, os mesmos são frequentemente gerados em alta velocidade, como dados gerados por matrizes de sensores ou múltiplos eventos [18]. Além desses dados serem gerados com uma alta velocidade eles também precisam ser processados em tempo real ou próximo disso. Como exemplo dessa característica pode-se citar os dados produzidos por dispositivos de sensores que seriam constantemente enviados para armazenamento em banco de dados.

No intuito de obter informações baseadas no histórico de dados armazenados, aplicações de *software* podem executar certas consultas numa determinada base de dados e assim poder deduzir importantes resultados. Essas informações podem auxiliar os usuários a encontrar tendências de negócio dando a possibilidade de alterar as suas estratégias [28]. Assim, pode-se perceber que existe um grande valor contido nos dados armazenados e que pode levar a muitas vantagens para a indústria e comércio, dentre outros ramos. Segundo [18], o valor é uma importante característica de um dado que é definida pelo valor agregado que o dado coletado pode trazer para um processo, atividade ou hipótese.

A Veracidade considera a inconsistência no fluxo de dados. O carregamento de dados

torna-se um desafio de ser manutenido. Especialmente em redes sociais com o incremento no uso que geram picos de carregamento de dados com a ocorrência de certos eventos [28]. Segundo [18], a Veracidade no contexto *Big Data* inclui principalmente dois aspectos: consistência dos dados que podem ser definidas por sua confiabilidade estatística e a confiabilidade dos dados definida pelo número de fatores incluindo a origem dos dados, métodos de coleta, processamento e infraestrutura confiável. A veracidade *Big Data* garante que o dado usado é confiável, autêntico e protegido de acessos e modificações não autorizadas [18].

A Variedade de dados está relacionada com os tipos de dados que são gerados na formação do *Big Data*. Segundo [28], os dados produzidos não são apenas dados tradicionais ou estruturados, mas também semiestruturados e não-estruturados como: páginas *web*, arquivos de *log* , e-mails, documentos, dados de sensores de dispositivo, dados de redes sociais, dentre outros. A variedade impõe novos requisitos para armazenamento de dados e projetos de banco de dados que devem adaptar-se dinamicamente ao formato dos dados que estão sendo manipulados [18].

Diante da complexidade imposta ao processo de mineração de dados com as características 5 V's como, por exemplo, o processamento de grandes massas de dados com alta dimensionalidade considerando restrições relacionadas com recursos e tempo, torna-se necessário utilizar técnicas de otimização já consolidadas na resolução de problemas complexos do mundo real. A próxima seção descreve os principais algoritmos bio-inspirados encontrados na literatura e discute porque tais algoritmos podem ser utilizados como técnicas para manipular *Big Data*.

3 Algoritmos Bio-Inspirados

Algoritmos bio-inspirados tem sido utilizados como métodos de otimização para problemas complexos normalmente não estacionários e multi-dimensionais [49]. A natureza tem sido utilizada como fonte de inspiração para o desenvolvimento de algoritmos de otimização, sendo denominados de Algoritmos Bio-Inspirados. Segundo [21], essa categoria de algoritmos pode ser subdividida em quatro grupos. São eles: Computação Evolucionária, Inteligência de Enxame, Redes Neurais Artificiais e Sistemas Imunológicos Artificiais. Esse artigo aborda algoritmos pertencentes à Inteligência de Enxame e Computação Evolucionária visto que apenas algoritmos pertencentes a esses dois grupos foram utilizados nos artigos revisados. É importante mencionar que Inteligência de Enxame e Computação Evolucionária possuem a semelhança de representarem meta-heurística estocásticas e serem baseadas em população de possíveis soluções, aqui abstraídas pelo termo indivíduo.

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 todo o espaço de solução

de um determinado problema. Além disso, Inteligência de Enxame e Computação Evolucionária mantem e sucessivamente melhoram uma população de potenciais soluções até que alguma condição de parada seja satisfeita . A métrica utilizada para determinar o quanto uma solução é boa para um determinado problema é uma função de eficiência ou função *fitness*. Através dessa função é possível determinar qual é o melhor indivíduo da população em um determinado momento [12].

Inteligência de Enxame é um conjunto de técnicas de pesquisa e otimização que são inspiradas no comportamento social de alguns organismos vivos como formigas, abelhas, pássaros, baratas, dentre outros. Auto-organização e controle descentralizado são características essenciais da Inteligência de Enxame. Essas características conduzem ao comportamento emergente que surge através de interações locais entre componentes do sistema e não pode ser obtido por um componente atuando sozinho [43]. Inteligência de Enxame possui algumas meta-heurísticas inspirados na natureza. Otimização por Enxame de Partícula, Otimização por Colônia de Formigas e Otimização por Enxame de Vermes Luminescentes são exemplos de algoritmos pertencentes a Inteligência de Enxame que foram abordados nos artigos revisados.

A Computação Evolucionária é outro paradigma inserido na Computação Natural e baseia-se na teoria de Charles Darwin para simular o processo evolutivo. A teoria de Darwin apresenta o processo de seleção natural que ocorre na natureza evidenciando a sobrevivência dos mais bem adaptados [57]. No contexto computacional, cada indivíduo representa uma solução candidata do espaço de solução do problema tratado. Computação Evolucionária é composta por vários subprocessos inspirados na evolução das espécies. São eles: seleção, reprodução, combinação ou *crossover* e mutação. Algoritmos Genéticos, Algoritmo *CHC*, Evolução Diferencial e Programação Genética são exemplos de algoritmos evolucionários utilizados nos artigos revisados.

A seguir são detalhados alguns algoritmos pertencentes à Inteligência de Enxame e Computação Evolucionária. A relação entre esses algoritmos e as tarefas de mineração de dados será apresentada na Seção 6.

3.1 Otimização por Enxame de Partícula

Em 1995, Otimização por Enxame de Partícula (*Particle Swarm Optimization - PSO*) foi proposta por Kennedy [32]. Esse algoritmo é inspirado no comportamento coordenado dos pássaros ao voarem e do movimento de cardume de peixes. O peixe ou o pássaro é considerado como uma partícula no algoritmo. No contexto computacional, cada partícula representa uma solução do problema. A população de partículas é inicializada de forma aleatória dentro do domínio do espaço de solução do problema. Além da solução, cada partícula armazena uma posição no espaço de busca, uma velocidade de deslocamento no espaço de

busca 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 solução é boa. Com isso será possível avaliar se a solução é melhor que a melhor solução pessoal e global. Esse processo é executado até o critério de parada ser satisfeito. Número de gerações, número de avaliação de função *fitness* e estagnação da melhor solução podem ser utilizados como critérios de parada. Os parâmetros do *PSO* são: o tamanho da população, o peso da inércia e os coeficientes de aceleração.

3.2 Otimização por Colônia de Formigas

Proposto por Dorigo em 1999, a Otimização por Colônia de Formigas (*Ant Colony Optimization - ACO*) é inspirada no comportamento das formigas pela busca de alimentos [31]. Foi observado que, no decorrer do tempo, as formigas são capazes de descobrir o caminho mais curto entre a fonte de alimento e sua colônia, ou seja, o melhor caminho. Isso ocorre graças ao rastro de feromônio que a formiga deposita no caminho percorrido entre a fonte de alimento e a colônia. Como o feromônio é uma substância atrativa para as formigas, quando elas depositam esta substância no caminho, o mesmo fica mais atrativo e recruta outras formigas para este caminho. Entretanto, quando diminuir a quantidade de formigas em determinado caminho, o feromônio tende a diminuir pois essa substância é volátil e evapora rapidamente. No contexto computacional, o caminho da formiga representa uma solução para um problema. Os parâmetros definidos no *ACO* são: peso relativo da trilha, a atratividade da trilha, o tamanho da população e a taxa de evaporação do feromônio.

3.3 Otimização por Enxame de Vermes Luminescentes

Otimização por Enxame de Vermes Luminescentes (*Glowworm Swarm Optimization - GSO*) foi proposto por Krishnanand e Ghose em 2005. Foi inspirado na capacidade do controle de emissão de luminosidade própria desses vermes em determinadas situações. Por exemplo, para atrair a fêmea ou para atrair uma presa numa determinada área de abrangência de formato circular [32]. A substância que influencia no nível de luminosidade é a luciferina. O verme escolhe a direção para onde se movimentar baseado na intensidade de luminescência das regiões vizinhas. 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 mais luciferina está mais

próxima da solução ótima. 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. O *GSO* possui os seguintes parâmetros: número de indivíduos, taxa de decaimento da luciferina, raio de abrangência da vizinhança e ponderador da função objetiva considerada no cálculo do nível de luciferina.

3.4 Algoritmos Genéticos

Propostos por John Holland em 1975, os Algoritmos Genéticos (*Genetic Algorithms - GA*) são os mais conhecidos e utilizados dentre os Algoritmos Evolucionários [15]. São inspirados no processo de Seleção Natural, segundo Darwin, onde os indivíduos mais bem adaptados ao meio possuem maior chance de sobreviver e assim reproduzir mais dependentes. Além da seleção natural, a hereditariedade genética também é inspiração desses algoritmos. Cada indivíduo, também chamado de cromossomo, possui um conjunto de tamanho fixo de *bits*, ou genes, que são representados em uma base de codificação. A reprodução desses indivíduos é realizada através da recombinação, ou *crossover*, de dois indivíduos pré-selecionados. Após isso, ainda na reprodução ocorre o processo de mutação nos descendentes recém-gerados. As características de intensificação e diversificação são expressos, respectivamente, através da combinação e mutação. Parâmetros do *GA* são: o tamanho da população, a probabilidade de *crossover* e probabilidade de mutação.

3.5 Evolução Diferencial

Evolução Diferencial (*Differential Evolution - DE*) foi proposta por Storn e Price [50]. Também possui uma população de indivíduos onde cada indivíduo possui um tamanho fixo e representa uma solução. O indivíduo da população é representado por um vetor e cada indivíduo possui um custo calculado com base no próprio vetor. Os indivíduos inicialmente terão os vetores inicializados com valores aleatórios dentro do domínio de solução do problema. A seleção de indivíduos para reprodução é feita de forma aleatória. Seleciona-se um indivíduo alvo da reprodução e mais três indivíduos aleatoriamente. O processo de reprodução ocorre através da adição da diferença ponderada entre dois dos três indivíduos ao terceiro indivíduo resultando num novo indivíduo. Nesta etapa ocorre a operação de mutação que caracteriza a diversificação. Após isso, ocorre o processo de combinação ou *crossover* do indivíduo mutado e o indivíduo alvo. Como resultado da combinação, resulta-se o indivíduo teste. A combinação caracteriza a intensificação do algoritmo *DE*. Após isso é avaliado o custo do indivíduo teste e do indivíduo alvo. Se o indivíduo teste possuir custo inferior ao do indivíduo alvo ele substituirá o indivíduo alvo na próxima geração. Caso contrário, o indivíduo teste será descartado. Serão produzidas novas gerações até que o critério de parada seja atendido. Normalmente o critério de parada é o número de gerações. Os parâmetros utilizados no *DE*

são: o tamanho da população, a probabilidade de *crossover* e o fator de mutação, número de dimensões do vetor, número de gerações.

3.6 Programação Genética

Em 1992 John Koza utilizou *GA's* para desenvolver programas computacionais aplicados em certas tarefas computacionais, surgindo assim a Programação Genética (*Genetic Programming - GP*) [31]. O indivíduo da população é um programa composto por funções e terminais. Os indivíduos são de tamanhos variados além de serem representados por árvores. As folhas das árvores são valoradas com variáveis e os nós internos com funções. A população inicial de indivíduos é gerada de forma aleatória considerando o domínio de solução do problema. Após avaliar o *fitness* de toda população de programas, são selecionados alguns programas para reprodução. A seleção pode ser feita com base em torneio ou roleta dentre outros critérios de seleção. Na reprodução de indivíduos para próxima geração além do *crossover* e mutação existe também o operador de reprodução. Na operação de *crossover* um ponto aleatório é escolhido para corte em cada árvore e as sub árvores geradas após o corte são trocadas. Na mutação uma folha ou nó de uma árvore é alterada de forma aleatória. Já na operação de reprodução o programa é copiado sem qualquer alteração para a próxima geração. Esse processo é realizado até que seja satisfeito algum critério de parada que normalmente considera o número de gerações. Nesse processo evolutivo os programas vão evoluindo de tal forma que o melhor indivíduo da última geração consiga representar uma função de aproximação do problema analisado. São parâmetros da *GP*: tamanho da população, probabilidade de *crossover*, probabilidade de mutação e número de gerações.

No cenário *Big Data*, para obter um melhor desempenho no processamento de algoritmos bio-inspirados é necessário embasar esses algoritmos numa infraestrutura com características adequadas a esse contexto e assim maximizar os benefícios dessa técnica. A ferramenta *Hadoop*, detalhado na Seção seguinte, fornece a infraestrutura com recursos mais adequados à técnica de mineração de dados.

4 *Hadoop*

Hadoop é uma ferramenta de código aberto utilizada para armazenar e manipular grandes massas de dados [41]. 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)* [28]. 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 [40]. Nas subseções a seguir, os componentes *HDFS* e *Hadoop MapReduce* serão detalhados.

4.1 *HDFS*

HDFS é um sistema de arquivos projetado para armazenar arquivos de dados muito grande com um padrão contínuo de acesso de dados 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 do 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* 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 de dados 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 [28].

4.2 *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 [40]. 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*

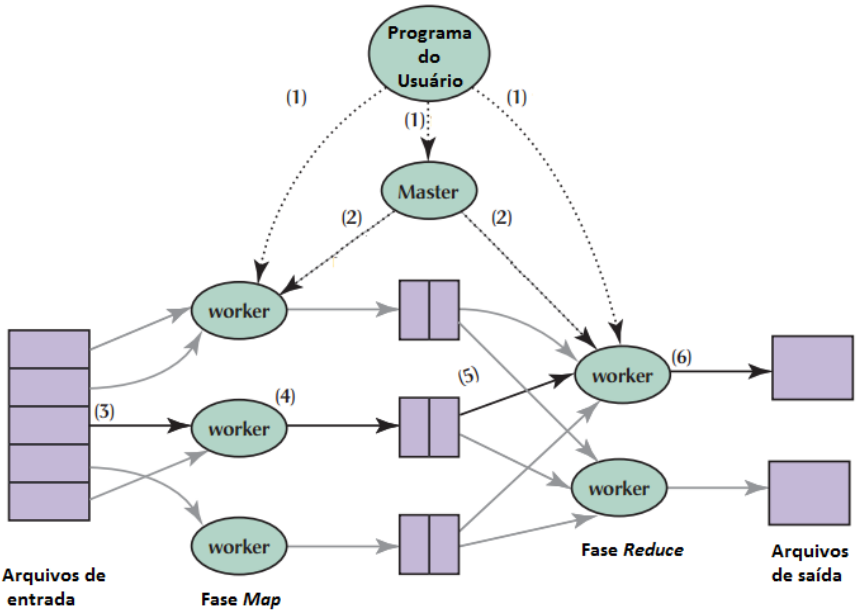

Figura 1. Execução *job MapReduce*. Adaptado de [16].

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 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 [16].

Visto que Algoritmos Bio-inspirados embasados no *framework Hadoop MapReduce* podem compor boas soluções no processo de mineração de *Big Data*, esse artigo faz a revisão da literatura de trabalhos que atuam nesse contexto de solução utilizando o método descrito na próxima Seção.

5 Metodologia

A revisão sistemática da literatura proposta por [29] foi o método de pesquisa escolhido para a realização da pesquisa contida nesse artigo. Isso reduz a possibilidade de obter resultados tendenciosos além de produzir artefatos capazes de viabilizar a reprodução da revisão apresentada nesse artigo. As próximas seções detalham os principais componentes do protocolo utilizado.

A revisão sistemática exibida nesse artigo visa responder as seguintes questões de pesquisa.

- Pergunta Principal

Questão 1 - Quais são os *softwares* aplicados em mineração de grandes massas de dados que utilizam algoritmos bio-inspirados projetados em *Haddop MapReduce*?

- Perguntas Secundárias

Questão 2 - Em quais tarefas ou etapas de mineração de dados o mecanismo é aplicado?

Questão 3 - Quais são os algoritmos bio-inspirados utilizados?

Questão 4 - Quantos algoritmos bio-inspirados são paralelizados?

Questão 5 - Como o algoritmo está projetado no *Hadoop MapReduce*?

Questão 6 - Quais bases utilizadas estão disponíveis?

Questão 7 - Quais V's são considerados?

A estratégia de busca está dividida em duas partes: chave de busca e fonte de pesquisa.

A chave de busca representa a expressão que será utilizada pelos mecanismos de busca para realizar a pesquisa. Assim, a chave foi definida com a composição de termos que apresentam mineração em grandes massas de dados. A chave é composta da seguinte forma:

(Swarm Intelligence OR Evolutionary Computation OR Bio Inspired Algorithm) AND (Hadoop MapReduce OR Hadoop Map Reduce) AND (Data OR Mining)

Dada a relevância na área da Ciência da Computação, os repositórios de busca utilizados foram: ACM Digital Library, IEEE Xplore, ScienceDirect e Scopus. Após a busca com a chave nos repositórios citados foi aplicado os critérios de seleção de artigos citados a seguir.

A Tabela 1 apresenta a quantidade de artigos retornados na pesquisa com a chave de busca em cada repositório, totalizando 237 artigos.

Tabela 1. Resultado da busca com a chave de pesquisa em cada repositório

IEEE	*ACM*	*ScienceDirect*	*Scopus*
168	52	10	7

A Tabela 2 exibe os critérios de inclusão e exclusão dos artigos encontrados na busca. Se o artigo caracterizar a presença de pelo menos um critério de exclusão, esse artigo não será utilizado na revisão.

Tabela 2. Critérios de Inclusão e Exclusão

Critérios de Inclusão	**Critérios de Exclusão**
CI 1 - Artigo propõe alguma aplicação de software que realiza mineração em grandes volumes de dados. Além disso, o mesmo deve envolver algum algoritmo bio-inspirado projetado em Hadoop MapReduce.	CE1 - Artigos duplicados. CE2 - Artigos que não estejam escritos em inglês. CE3 - Artigos publicados há mais de 10 anos.

No intuito de garantir a qualidade dessa revisão é necessário analisar cada artigo selecionado considerando os critérios citados. Assim, em cada artigo selecionado será necessário analisar as seguintes partes: Título, Resumo, Palavras chave, Mecanismo proposto, Resultados e Conclusão.

Após aplicar os critérios de inclusão e exclusão nos artigos encontrados com a chave de busca foram obtidos 8 artigos, conforme Tabela 3.

Com o intuito de complementar a busca, a pesquisa por trabalhos relacionados foi realizada diretamente em alguns *Journals* importantes para a área. Dessa forma, foi encontrado e incluído nesta revisão um artigo científico de cada um dos seguintes *Journals*: *Scientific Research and Essays*, *International Journal on Recent and Innovation Trends in Computing and*

Tabela 3. Trabalhos Selecionados por repositório

IEEE	*ACM*	*ScienceDirect*	*Scopus*
4	2	0	2

Communication, *International Journal of Scientific & Technology Research Volume* e *Mathematical Problems in Engineering*. Todos os 4 trabalhos encontrados possuem os critérios de inclusão e não possuem critérios de exclusão. Desta maneira, no total, foram selecionados 12 artigos científicos para compor esta revisão.

Para cada artigo selecionado, considerando todo o processo de seleção, este deverá ser analisado conforme procedimento de avaliação, de tal forma que seja possível extrair do artigo as seguintes informações: Tarefa ou etapa de mineração de dados; Algoritmo bio-inspirado utilizado; Se a base utilizada está disponível; E quais V's estão sendo considerados.

6 Trabalhos Revisados

Esta seção apresenta os trabalhos encontrados na literatura que empregam algoritmos bio-inspirados projetados no *framework Hadoop MapReduce* aplicados em mineração de grandes massas de dados. Após todo processo de seleção já descrito, foram selecionados 12 artigos diretamente relacionados com o escopo deste trabalho. O objetivo dessa revisão é identificar as principais contribuições que esses trabalhos realizaram em mineração de grandes massas de dados, tornando possível obter uma visão mais apurada dos recursos e dificuldades existentes atualmente nessa área. 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) [5] propôs um algoritmo que realiza a tarefa de agrupamento e tem como base o algoritmo *PSO* adaptado ao paradigma *MapReduce*. O algoritmo é composto por três módulos principais que são executados sequencialmente a cada iteração. No primeiro módulo ocorre a atualização do centróide de cada partícula do enxame através de um *job MapReduce* no arquivo que mantem o enxame. No segundo módulo ocorre outro *job MapReduce* nos grandes arquivos de dados e faz o cálculo do *fitness* considerando os novos centróides. No terceiro módulo ocorre a combinação das informações atualizadas no primeiro e segundo módulos, além de calcular o melhor resultado individual de cada partícula e o melhor resultado global do enxame finalizando a iteração do algoritmo. As bases *MAGIC Gamma Telescope Data Set* (19.020 instâncias de dados, 10 dimensões, atributo-meta com 2 classes e 3 *Megabytes* de tamanho), *Poker Hand Data Set* (1.025.010 instâncias de dados, 10 dimensões, atributo-meta com 10 classes e 49 *Megabytes* de tamanho) e *Covertype Data Set* (581.012 instâncias de dados, 54 dimensões, atributo-meta com 7 classes e 199.2 *Megabytes*

de tamanho) disponíveis no repositório *UCI*[4] foram utilizadas nesse trabalho. Também foi utilizada a base *Electricity* (45.312 instâncias de dados, 8 dimensões, atributo-meta com 2 classes e 6 *Megabytes* de tamanho) que está disponível no *MOA*[5]. Os resultados mostraram uma melhor qualidade comparando 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) [6] como um componente que realiza agrupamento em volume de dados de tráfico de redes de grande escala. O algoritmo compõe uma arquitetura de sistema de detecção de intrusos. Os dados analisados são coletados de tráfego de redes. 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*. O mecanismo proposto nos dois trabalhos distribui o cálculo da velocidade e dos centróides de cada partícula no primeiro *job MapReduce*. O segundo *job* calcula a média das distâncias e, por último, a função *fitness* é executada em uma rotina sequencial complementar.

Judith (2015) [27] apresenta um sistema que realiza tarefa de agrupamento em documentos utilizando o algoritmo *PSO*. Além de *PSO*, o sistema proposto também utiliza o algoritmo de agrupamento de dados *K-Means* e a técnica de redução de dimensionalidade *Latent Semantic Indexing (LSI)*. A utilização do algoritmo *PSO* projetado no paradigma *MapReduce* visa encontrar melhores valores para os centróides explorando assim a característica vantajosa da busca global. Na fase de mapeamento o *fitness* é calculado considerando a distância dos documentos até o centróide do *cluster*. Na fase de redução os valores dos centróides são atualizados. Após isso, o *K-Means* projetado em *MapReduce* é executado visando melhorar a qualidade dos *clusters*. Na sequência, a técnica *LSI* é aplicada como método de redução de dimensionalidade dos documentos melhorando a eficiência do processo de agrupamento. A base de documentos *Reuters-21578* (21578 documentos, 131 classes, 42686 termos e tamanho de 28 *Megabytes*) foi utilizada nos experimentos e está disponível no *UCI* (http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html). Também foi utilizada a base de dados RV1 (1000000 documentos, 860 classes, 3268650 termos). A qualidade do agrupamento de dados do sistema proposto foi melhor que a qualidade obtida pelo *K-Means* executado isoladamente. *K-Means* é fortemente dependente dos centróides iniciais e converge facilmente ao ótimo local. Assim, em vez de inicializar os centróides com valores aleatórios é utilizado o PSO. Dessa forma a qualidade dos *clusters* melhora. A utilização da técnica *LSI* também impactou na melhora da qualidade dos *clusters*. O sistema também demonstrou bom *speedup* com a utilização do *framework Hadoop MapReduce*.

Al-Mad (2014) [3] 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

[4] *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.
[5] *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/.

job MapReduce no arquivo de dados. Nesse processo, cada indivíduo procura apenas um centróide como subsolução no mapeamento. Na redução são combinadas as subsoluções compondo a solução global. As bases utilizadas foram: *MAGIC Gamma Telescope Data Set*, *Poker Hand Data Set* e *Covertype Data Set* disponíveis no *UCI*. Também foi utilizada a base *Electricity* que está disponível no *MOA*. 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) [5]. Nesse mecanismo, o mapeamento calcula a distância de cada indivíduo do enxame GSO para cada objeto de dados distribuído no paradigma *MapReduce*. Após finalizar a execução do *job MapReduce* é atualizado o *fitness* e a luciferina.

O algoritmo *ACO* foi proposto por Bhavani (2014) [8] para realizar a tarefa de agrupamento. Para a aplicação do ACO, em uma etapa de pré-processamento, antes da execução do *job MapReduce*, um grafo é construído a partir dos dados que serão agrupados. No processo de mapeamento a colônia de formigas encontra a árvore de cobertura mínima do grafo e na redução ocorre a atualização do feromônio tendo como base a média da menor árvore de cobertura mínima. Ao final de todas as iterações ocorre a poda da árvore que utiliza a quantidade de feromônio encontrando no final do processo. Quanto menos feromônio maior é a probabilidade da poda. Em relação aos dados analisados, foi citado apenas o tamanho de três bases: 15,7 *Kilobytes*, 4,2 *Megabytes* e 17,8 *Megabytes*. Esse algoritmo conseguiu alcançar um bom *speedup*. O mecanismo distribui o algoritmo ACO de tal forma que no mapeamento a quantidade de feromônio influencia no cálculo da arvore de cobertura mínima. Já na redução o feromônio é atualizado.

Outro trabalho que utiliza *ACO* para realizar a tarefa de agrupamento foi apresentado por Yang (2012) [55]. No sistema apresentado, antes de executar o *job MapReduce* o componente global inicializa os parâmetros, gerencia os movimentos e posições dos agentes utilizando coordenadas. Dessa forma, a cada iteração após o movimento das formigas realizando o agrupamento de dados é executado o *job MapReduce*. No mapeamento ocorre o calculo de similaridade dos objetos de dados. Na redução ocorre o média global de similaridade. Nos experimentos foram utilizadas as bases Iris (150 instâncias de dados, 4 dimensões, atributo-meta com 3 classes, 4,4 *Kilobytes* de tamanho), Ecoli (336 instâncias de dados, 8 dimensões, atributo-meta com 8 classes, 19 *Kilobytes* de tamanho) e *Letter Recogination* (20,000 instances, 26 classes, 17 attributes) disponíveis no *UCI*. Foi comprovado que o tempo de execução do sistema em bases grandes diminui à medida que é incrementado o número de nodos que compõe o *cluster*. Entretanto, percebe-se que o tempo de execução do sistema em pequenas bases de dados aumenta a medida que é incrementado o número de nodos que compõe o *cluster* devido ao tempo consumido na comunicação entre os nodos na rede. Em relação a qualidade do agrupamento, foi comparado os resultados da versão do sistema com e sem a arquitetura *MapReduce* e não foi possível concluir melhoria na qualidade dos grupos.

Hans (2015) [25] apresenta um aplicativo de *software* que realiza a tarefa de agrupamento de dados utilizando *GA*. Na fase de mapeamento é calculado o *fitness* dos indivíduos considerando utilizando a inter e intra distancia dos *clusters*. Na fase de redução, novos indivíduos com novos centróides gerados e analisados. Dessa forma, a cada iteração o *job MapReduce* é executado. A base *Europe* (169308 instâncias de dados, 2 dimensões, atributo-meta com 2 classes, 3,6 *Megabytes* de tamanho) disponíveis no *SIPU*[6] foi utilizada nesse trabalho. Foi comparado a versão do aplicativo executado de forma sequencial com a versão projetada em *Hadoop MapReduce*. Foi demonstrado que os resultados de qualidade dos *clusters* foram semelhantes. Em relação ao tempo de execução, a versão do aplicativo projetado em *Hadoop MapReduce* consumiu menos tempo.

O algoritmo *GA* também foi utilizado para realizar a etapa de seleção de atributos antes de realizar a geração de regras de classificação em Ferrucci (2015) [20]. Como o problema de seleção de atributos otimiza os dados de treinamento para realizar a tarefa de classificação esse trabalho foi considerado como tarefa de classificação. Para tal, foi construído um *framework* de desenvolvimento que paraleliza o algoritmo *GA* utilizando *Hadoop MapReduce*. O *framework* adapta o paradigma *MapReduce* de tal forma que é possível executar em uma única iteração vários processos de mapeamento antes e depois da redução. Dessa forma foi possível mapear todos os processos que ocorrem em uma geração de indivíduos em *GA* em um único *job MapReduce*. O processamento da população de indivíduos é realizada de forma segmentada no modelo de ilhas e o GA é executado de forma independente em cada ilha. Dessa forma o algoritmo mantém várias populações diferentes sendo assim multipopulacional. Além disso, ocorre migração de indivíduos entre as ilhas de acordo com a parametrização, favorecendo a co-evolução. As bases utilizadas disponíveis no *UCI* são: *Statlog (German Credit Data) Data Set* (1000 instâncias de dados, 20 dimensões, atributo-meta com 2 classes, 78 *Kilobytes* de tamanho) e *Audiology Data Set* (226 instâncias de dados, 69 dimensões, atributo-meta com 24 classes, 32 *Kilobytes* de tamanho). Os resultados demonstram que a precisão e tempo de execução foram melhores para a aplicação desenvolvida como base no *framework Hadoop MapReduce* em comparação com o *Weka*[7] e com uma versão pseudo-distribuída com apenas uma ilha.

Em Peralta (2015) [45], também é abordada a etapa de seleção de atributos da tarefa de classificação de dados. Nesse caso, o algoritmo bio-inspirado utilizado é uma variação do *GA*. Diferente de *GA*, esse algoritmo verifica a similaridade entre dois possíveis indivíduos antes de aplicar o operador *crossover*. O sistema proposto é composto por duas fases principais. Cada fase representa um *job MapReduce*. Na primeira fase, o algoritmo é executado na fun-

[6] *Speech and Image Processing Unit* é um grupo de pesquisa da *School of Computing University of Eastern Finland* que trabalha com padrões recognitivos, compressão e mineração de dados. Dentre outros benefícios à comunidade científica, algumas bases de dados para realizar a tarefa de agrupamento são disponibilizadas, http://cs.joensuu.fi/sipu/datasets/.

[7] *Weka* é uma coleção de algoritmos de aprendizado de máquina utilizados em tarefas de mineração de dados. Os algoritmos aplicados diretamente em um conjunto de dados (*dataset*) ou ser chamado de um código *Java* [20].

ção de mapeamento produzindo como saída um vetor binário. Esse vetor possui informações de relevância de cada atributo dos dados mapeados. Na função de redução, calcula-se a média dos vetores emitidos na saída do mapeamento e com isso é obtido o vetor final, chamado de vetor *x*. O vetor *x* define quais atributos serão mantidos ou não na base de treinamento para realização da tarefa de classificação. No segundo *job MapReduce*, o vetor *x* é utilizado para remover os atributos menos relevantes na base de dados. Na fase de mapeamento é verificado para cada atributo de dados se o mesmo é relevante ou não de acordo com o vetor *x*. Se o atributo não for relevante o mesmo não é emitido para a fase de redução, assim esse atributo é removido da base de treinamento. Caso contrário, o atributo é mantido na base de dados de treinamento. Na função de redução os dados são apenas concatenados. Após a execução do segundo *job MapReduce*, a base de treinamento possuirá apenas dados com os atributos relevantes. Com isso, a base de treinamento reduzida é utilizada por algoritmos de classificação. Foram utilizadas duas bases nos experimentos. Uma base (500000 instâncias de dados, 2000 dimensões, o autor não explicitou mais informações) foi gerada artificialmente pelo *Pascal Large Scale Learning Challenge* (http://largescale.ml.tu-berlin.de/instructions/), essa base não está disponível. A outra base de dados (32000000 instâncias de dados, 631 dimensões, atributo-meta com 2 classes, 56 *Gigabytes* de tamanho) está disponível na página *web* do evento *Evolutionary Computation for Big Data and Big Learning Workshop* realizado em 2014 (http://http://cruncher.ncl.ac.uk/bdcomp/). Foi comparado o tempo de execução do sistema projetado em *Hadoop MapReduce* com sua versão sequencial. O resultados demonstram que o tempo de execução da versão sequencial cresce numa tendência quase exponencial a medida que o tamanho da base de dados aumenta. Foi percebido também que a precisão dos algoritmos tradicionais utilizados na classificação obtiveram melhor precisão na tarefa de classificação utilizando a base de dados após a etapa de seleção de atributos.

Al-Madi (2013) [4] apresenta um algoritmo que realiza classificação e tem como base a implementação o algoritmo *GP*. No mapeamento é calculado o *fitness* de cada programa. Na redução ocorre a seleção, *crossover* e mutação, além de gerar nova população. Foram utilizados seis *Datasets* disponíveis na UCI, porém, o artigo não deixa claro quais foram as bases. Resultados mostram que foi obtida uma alta precisão de classificação além de manter escalabilidade e diminuir o tempo de execução por mapeamento.

Daoudi (2014) [14] apresenta uma estratégia para realizar a tarefa de agrupamento. Esse mecanismo utiliza o algoritmo *DE* e é composto por três níveis. No primeiro nível ocorre processamento *MapReduce* onde atuam a mutação e seleção que calculam novos centróides. No segundo nível ocorre outro processo *MapReduce* que calcula a menor distância entre o registro e os centróides avaliando assim os centróides gerados na fase anterior. Por último, ocorre um processo de atualização do *fitness* e seleção dos indivíduos. Foram utilizadas as bases: armstrong-v1 (12.582 instâncias de dados, atributo-meta com 2 classes, 274 *Kilobytes* de tamanho), armstrong-v2 (12.582 instâncias de dados, atributo-meta com 3 classes, 274 *Kilobytes* de tamanho), bhattacharjee (12.600 instâncias de dados, atributo-meta com 5 clas-

ses, 1,7 *Megabytes* de tamanho), chowdary (22.283 instâncias de dados, atributo-meta com 2 classes, 88 *Kilobytes* de tamanho), dyrskjot (7.129 instâncias de dados, atributo-meta com 3 classes, 242 *Kilobytes* de tamanho), golubv1 (7.129 instâncias de dados, atributo-meta com 2 classes, 270 *Kilobytes* de tamanho), golubv2 (7.129 instâncias de dados, atributo-meta com 3 classes, 270 *Kilobytes* de tamanho), laiho (22.883 instâncias de dados, atributo-meta com 2 classes, 406 *Kilobytes* de tamanho), nutt-v1 (12.625 instâncias de dados, atributo-meta com 4 classes, 403 *Kilobytes* de tamanho), nutt-v2 (12.625 instâncias de dados, atributo-meta com 2 classes, 403 *Kilobytes* de tamanho), nutt-v3 (12.625 instâncias de dados, atributo-meta com 2 classes, 403 *Kilobytes* de tamanho) , pomeroy-v1 (7.129 instâncias de dados, atributo-meta com 2 classes, 108 *Kilobytes* de tamanho), pomeroy-v2 (7.129 instâncias de dados, atributo-meta com 5 classes, 108 *Kilobytes* de tamanho), ramaswamy (16.063 instâncias de dados, atributo-meta com 14 classes, 835 *Kilobytes* de tamanho), shipp-2002-v1 (7.129 instâncias de dados, atributo-meta com 3 classes, 217 *Kilobytes* de tamanho), su (12.533 instâncias de dados, atributo-meta com 10 classes, 939 *Kilobytes* de tamanho), west (7.129 instâncias de dados, atributo-meta com 2 classes, 212 *Kilobytes* de tamanho) e Yeoh-v2 (12.625 instâncias de dados, atributo-meta com 6 classes, 2,5 *Megabytes* de tamanho). As mesmas estão disponíveis em *Schliep lab*[8] . Os resultados comprovam que esse modelo é mais preciso que *K-Means* e é mais estável e robusto que Aljarah (2012) [5] pois possui melhor desempenho a medida que o volume da base de dados aumenta.

Bhavani (2011) [9] apresenta duas estratégias para realizar agrupamento. Uma envolve *DE* e *K-Means* e a outra envolve o uso de *DE*, *ACO* e *K-Means*. Em ambas estratégias o algoritmo *DE* é utilizado para realizar a atualização global dos centróides enquanto que *K-Means* é utilizado localmente. Entretanto, no primeiro mecanismo o limiar de ativação dos centróides é atualizado de forma randômica e é verificada uma deterioração da velocidade de convergência dos grupos. Esse problema é contornado no segundo mecanismo com a utilização do *ACO* para realizar a atualização do limiar. Em relação as bases utilizadas, esse artigo menciona apenas que as bases são da área da bioinformática. Assim, o segundo mecanismo obteve resultados com melhor qualidade e desempenho. Ambos mecanismos aplicam o algoritmo DE apenas na redução, mas no segundo mecanismo o algoritmo o *ACO* é distribuído de tal forma que o feromônio seja analisado no mapeamento e é atualizado na redução.

Na próxima Seção é apresentada a discussão comparativa entre os trabalhos descritos considerando alguns aspectos importantes.

[8] *Schliep lab bioinformatics*, é o laboratório do Departamento de Bioinformática da Freie Universität Berlin que disponibiliza alguns *Datasets* dessa área para serem utilizados por algoritmos de aprendizado de máquina. http://bioinformatics.rutgers.edu/.

7 Discussão

Nessa seção, serão discutidos alguns critérios que visam identificar as semelhanças e diferenças entre os 12 trabalhos apresentados na seção anterior. Quatro critérios foram analisados. São eles: tarefa de mineração de dados abordada, tipo de algoritmo bio-inspirado utilizado, como o algoritmo bio-inspirado foi projetado no *Hadoop MapReduce*, a disponibilidade das bases utilizadas e quais V's foram abordados. A distribuição dos dados minerados em *HDFS* é uma característica comum a todos os trabalhos.

7.1 Tarefa de mineração de dados

Esse critério busca evidenciar qual é a proporção de cada tarefa de mineração de dados abordada considerando todos os trabalhos levantados na revisão. Como visto na Figura 2, pesquisas que realizam agrupamento de dados são mais expressivas que os trabalhos que realizam a tarefa de classificação.

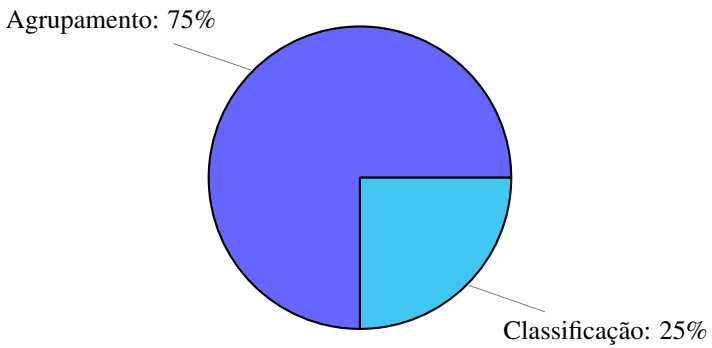

Figura 2. Distribuição de tarefas de mineração

A proporção de trabalhos que realizam agrupamento é bem mais significante do que a proporção que aborda a tarefa de classificação. Isso pode ser atribuído ao fato de que abordar a tarefa de agrupamento utilizando o paradigma *MapReduce* se torna um processo onde as características do problema se beneficiam das características do paradigma, e vice-versa. A computação necessária para realizar agrupamento é, basicamente, o cálculo da distância entre o centróide representado por uma possível solução (indivíduo ou partícula, por exemplo) e o dado que está sendo agrupado. Mesmo que todo o arquivo de dados seja mapeado de forma independente e distribuída, o cálculo da distância possui todas as informações necessárias para ser efetuado. Essa tarefa se adequa bem ao processo de mapeamento pois todo volume de dados será processado de forma independente a cada iteração. Já na tarefa de classifica-

ção, é necessário armazenar e controlar a frequência de ocorrência dos dados. Isso deixa o mecanismo mais complexo devido ao controle que será necessário para analisar a frequência de dados, visto que o processo de mapeamento analisa cada objeto do volume de dados de forma independente.

Percebe-se que apenas duas tarefas de mineração de dados foram abordadas nos trabalhos revisados. Considerando o protocolo de pesquisa e repositório de artigos científicos utilizados nessa pesquisa, não foi encontrado nenhum trabalho que aborde tarefas como Associação e Regressão. Portanto, essas tarefas possuem vasto campo de exploração considerando a linha de solução abordada nessa revisão.

7.2 Tipos de algoritmos bio-inspirados

Este critério visa sumarizar a variedade de algoritmos bio-inspirados encontrados dentro do contexto da revisão. A grande diversidade de algoritmos bio-inspirados aplicados em mineração de dados utilizando *Hadoop MapReduce* pode ser evidenciada na Figura 3.

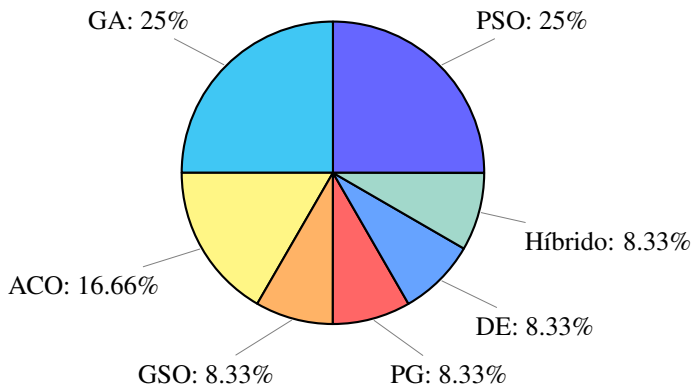

Figura 3. Distribuição de tipos de algoritmos bio-inspirados

Observa-se que *PSO* e *GA* são mais utilizados por serem os algoritmos mais conhecidos pela busca eficiente de solução global. Além de terem bom desempenho em problemas de domínio contínuo, sua fácil compreensão e desenvolvimento são características interessantes que favorecem suas escolhas.

De maneira geral, cada pesquisa utiliza as vantagens particulares de cada algoritmo bio-inspirado, explorando a diferenciada capacidade de diversificação e intensificação na exploração do espaço de solução. Apenas um trabalho utiliza uma abordagem híbrida com mais de um algoritmo bio-inspirado, *DE* e *ACO*. Existem outros algoritmos bio-inspirados como

Colônia de Abelhas Artificiais (*Artificial Bee Colony - ABC*), Algoritmo do Morcego (*Bat Algorithm - BA*), Algoritmo de Vagalumes (*Firefly Algorithm - FA*) que possuem potencial de serem utilizados com *Hadoop MapReduce* em mineração de *Big Data*.

7.3 Projeto do *Hadoop MapReduce*

Três modelos de paralelismo foram identificados e são descritos a seguir.

A principal característica do Modelo 1 é a execução de apenas um *job MapReduce* por ciclo de iteração do algoritmo (conforme Figura 4, onde o ícone com o desenho do planeta Terra representa o algoritmo bio-inspirado). Além disso, cada execução do *job MapReduce* processa toda a população de indivíduos no mapeamento do volume de dados. Como apresentado na Figura 4, o algoritmo bio-inspirado está projetado no *job Hadoop MapReduce*, portanto será paralelizado. Sete trabalhos usam esse modelo.

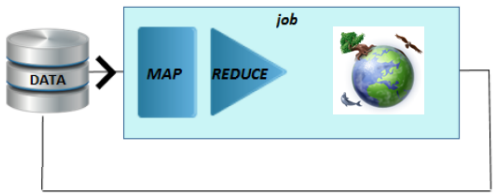

Figura 4. Modelo 1:Um *job MapReduce*

O Modelo 2 tem como principal característica a execução de dois *jobs MapReduce* executados um após o outro (Rótulos 1 e 2 da Figura 5). Logo em seguida, é executada uma rotina que finaliza o ciclo de iteração do algoritmo (Rótulo 3 da Figura 5).

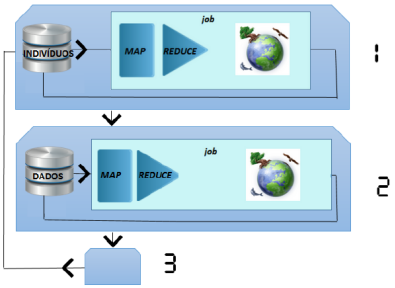

Figura 5. Modelo 2: Dois *jobs MapReduce*

Conforme Figura 6, o primeiro *job MapReduce* processa dados que representam a

movimentação dos indivíduos da população no espaço de solução. Já o segundo *job MapReduce* processa o volume de dados que está sendo minerado considerando a população de indivíduos. Na sequência uma rotina é executada para realizar as últimas operações de um ciclo do algoritmo bio-inspirado. Esse modelo processa a população de indivíduos de forma paralela através do *job MapReduce* representado pelo Rótulo 1. Quatro trabalhos utilizam esse modelo. Apesar de [27] não realizar a execução de um *job MapReduce* no arquivo com a população de indivíduos, o mesmo foi classificado no Modelo 2 por executar dois *jobs MapReduce*, um após o outro, que corresponde a principal característica desse modelo.

O Modelo 3 tem como principal característica a execução de um *job MapReduce* onde múltiplas funções de mapeamento são processadas antes e depois da função de redução (Figura 6).

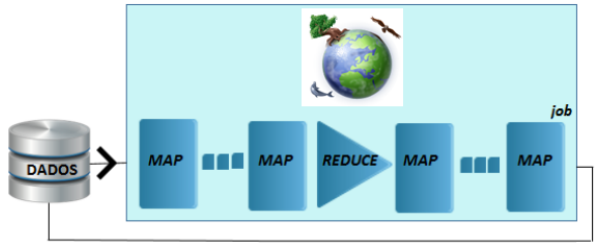

Figura 6. Modelo 3: *Job com múltiplos Maps e Reduce*

Esse modelo consegue executar em um único *job MapReduce* em várias fases do algoritmo bio-inspirado. Cada fase do algoritmo pode ser executada na função de mapeamento ou de redução. A desvantagem é a complexidade necessária para alterar seu *framework Hadoop MapReduce* e assim alterar o comportamento padrão do *job MapReduce* que é executar apenas um mapeamento e uma redução. Apenas um trabalho utilizou esse modelo.

A Figura 7 exibe a sumarização dos trabalhos revisados agrupados pelo modelo que cada trabalho melhor se assemelha.

O Modelo 1 tende a ser mais vantajoso por ser mais simples de ser desenvolvido e por conseguir utilizar os benefícios do *framework MapReduce* no processamento dos dados que estão sendo minerados. Esse modelo é o mais utilizado nos trabalhos revisados. Como não é vantajoso executar o *job MapReduce* em arquivos pequenos [55] [28], como ocorre no arquivo que armazena a população de indivíduos no Modelo 2, este modelo torna o desenvolvimento mais trabalhoso sem ganho de desempenho com a paralelização devido ao tempo consumido na comunicação entre os nodos do *cluster*. Dessa forma, a ocorrência do Modelo 2 nos trabalhos revisados foi menor que a ocorrência do Modelo 1. Apenas um trabalho foi classificado no Modelo 3 devido a particularidade não muito comum de envolver alteração

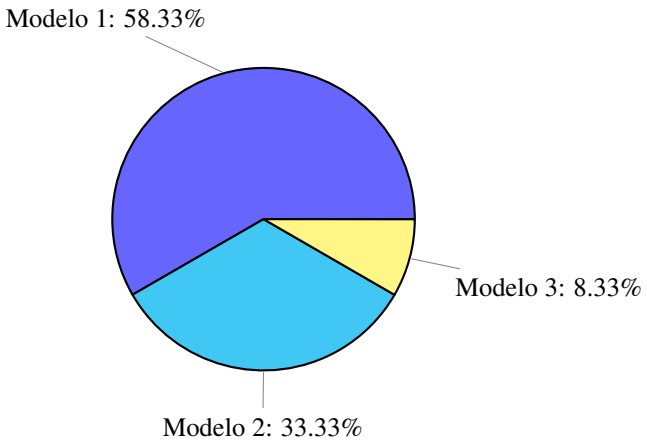

Figura 7. Modelos de projeto *Hadoop MapReduce*

no comportamento padrão do *framework Hadoop MapReduce*.

7.4 Disponibilidade de Base de Dados e V's abordados

A grande maioria dos trabalhos revisados, 75%, utilizam bases disponíveis em repositórios na *web*. Os repositórios mais utilizados são: *UCI Machine Learning Repository e Massive Online Analysis - MOA*.

A disponibilidade dessas bases é muito importante para que seja possível comparar diferentes mecanismos e assim conseguir avaliar as vantagens e desvantagens entre eles.

Em relação aos 5 V's que caracterizam *Big Data*, o único V abordado nos trabalhos é o Volume. Ainda assim, o Volume de dados minerado nos trabalhos não chegam a medir em *Petabytes*, que corresponde a 10^{15} *Bytes*, conforme a definição dessa característica. As bases de dados utilizadas estão em *Megabytes*, que corresponde a 10^{6} *Bytes*. Apenas uma base utilizada alcançou a grandeza de *Gigabytes*. O conceito de *Big Data* menciona dados com volume em *Terabytes*. Apesar disso, percebe-se que os trabalhos encontrados na literatura ainda não conseguiram colocar em prática a manipulação de dados em *Terabytes*.

8 Conclusão

Esse artigo apresentou a revisão de trabalhos que realizam mineração em grandes massas de dados utilizando algoritmos bio-inspirados embasados no *framework Hadoop MapReduce*. Essa combinação pode proporcionar soluções de mineração de dados mais adequadas ao contexto de grandes massas de dados. Com isso, consegue-se entender e perceber o quão flexível é projetar algoritmos dessa natureza no *framework Hadoop MapReduce*. Isso porque é simples e flexível adaptar a execução do algoritmo bio-inspirado na função *map*, *reduce* ou em ambas. Dessa forma obtém-se paralelização, tolerância a falhas e escalabilidade embasada no *framework MapReduce* com o grande poder de otimização fornecido pelos algoritmos bio-inspirados.

Entretanto, pode-se perceber que a execução do *job MapReduce* em arquivos de dados relativamente pequenos não é recomendado devido ao tempo consumido na comunicação entre os nodos do *cluster*.

Na busca eficiente por resultados de mineração de dados, características como a intensificação e a diversificação no espaço de busca pertencentes aos algoritmo bio-inspirados, são responsáveis por proporcionar às pesquisas desenvolvidas a encontrar as melhores soluções no espaço de busca.

Também percebeu-se a disponibilidade de bases na *web* que foram utilizadas nos experimentos dos trabalhos revisados. Isso favorece o processo de avaliação comparativa entre os trabalhos existentes e identifica *benchmarks* para desenvolvimentos futuros.

Com relação às características definidas pela contextualização de *Big Data*, nem todos os V's são abordados pelos trabalhos encontrados. Todos trabalhos abordam apenas o volume dentre as 5 propriedades *Big Data*. Espera-se que, com a evolução contínua do estado da arte, progressivamente outras características *Big Data* sejam abordadas na proposta de solução dos trabalhos futuros de tal forma que todos os 5 V's sejam abordados.

Contribuição dos autores:

- Sandro Roberto Loiola de Menezes: Pesquisa e escrita do manuscrito.

- Rebeca Schroeder Freitas: Correções e direcionamentos da pesquisa.

- Rafael Stubs Parpinelli: Coordenação do trabalho, correções e direcionamentos da pesquisa.

Referências

[1] Big data defintion, gartner, inc. `http://www.gartner.com/it-glossary/big-data/`. Accessed: 2015-08-07.

[2] R. Al-Khannak and B. Bitzer. Grid computing as an innovative solution for power system's reliability and redundancy. In *Clean Electrical Power, 2009 International Conference on*, pages 790–797. IEEE, 2009.

[3] 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.

[4] N. Al-Madi and S. A. Ludwig. Scaling genetic programming for data classification using mapreduce methodology. In *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*, pages 132–139. IEEE, 2013.

[5] 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.

[6] 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.

[7] A. Asbern and P. Asha. Performance evaluation of association mining in hadoop single node cluster with big data. In *Circuit, Power and Computing Technologies (ICCPCT), 2015 International Conference on*, pages 1–5. IEEE, 2015.

[8] 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.

[9] R. Bhavani, G. S. Sadasivam, and R. Kumaran. A novel parallel hybrid k-means-deaco clustering approach for genomic clustering using mapreduce. In *Information and Communication Technologies (WICT), 2011 World Congress on*, pages 132–137. IEEE, 2011.

[10] S. Binitha and S. S. Sathya. A survey of bio inspired optimization algorithms. *International Journal of Soft Computing and Engineering*, 2(2):137–151, 2012.

[11] M. Chen, S. Mao, and Y. Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.

[12] 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.

[13] P.-T. Chung and S. H. Chung. On data integration and data mining for developing business intelligence. In *Systems, Applications and Technology Conference (LISAT), 2013 IEEE Long Island*, pages 1–6. IEEE, 2013.

[14] M. Daoudi, S. Hamena, Z. Benmounah, and M. Batouche. Parallel diffrential evolution clustering algorithm based on mapreduce. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 337–341. IEEE, 2014.

[15] L. Davis et al. *Handbook of genetic algorithms*, volume 115. Van Nostrand Reinhold New York, 1991.

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

[17] Y. Demchenko, C. de Laat, and P. Membrey. Defining architecture components of the big data ecosystem. In *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, pages 104–112. IEEE, 2014.

[18] Y. Demchenko, P. Grosso, C. de Laat, and P. Membrey. Addressing big data issues in scientific data infrastructure. In *Collaboration Technologies and Systems (CTS), 2013 International Conference on*, pages 48–55. IEEE, 2013.

[19] K. A. Doshi, T. Zhong, Z. Lu, X. Tang, T. Lou, and G. Deng. Blending sql and newsql approaches: reference architectures for enterprise big data challenges. In *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2013 International Conference on*, pages 163–170. IEEE, 2013.

[20] F. Ferrucci, P. Salza, M. T. Kechadi, and F. Sarro. A parallel genetic algorithms framework based on hadoop mapreduce. pages 1664–1667, 2015.

[21] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister. A brief review of nature-inspired algorithms for optimization. *Electrotechnical Review*, 80(3), 2013.

[22] A. Ghosh and B. Nath. Multi-objective rule mining using genetic algorithms. *Information Sciences*, 163(1):123–133, 2004.

[23] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[24] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques: concepts and techniques*. Elsevier, 2011.

[25] N. Hans, S. Mahajan, and S. Omkar. Big data clustering using genetic algorithm on hadoop mapreduce. *International Journal of Scientific Technology Research*, 4, 2015.

[26] C. Jin, C. Vecchiola, and R. Buyya. Mrpga: an extension of mapreduce for parallelizing genetic algorithms. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 214–221. IEEE, 2008.

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

[28] 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.

[29] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15, 2009.

[30] P. Koturwar, S. Girase, and D. Mukhopadhyay. A survey of classification techniques in the area of big data. *International Journal of Advance Foundation and Research in Computer*, 1, 2015.

[31] J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*, volume 1. MIT press, 1992.

[32] 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.

[33] M. Kutlu, G. Agrawal, and O. Kurt. Fault tolerant parallel data-intensive algorithms. In *High Performance Computing (HiPC), 2012 19th International Conference on*, pages 1–10. IEEE, 2012.

[34] Y. Lai and S. ZhongZhi. An efficient data mining framework on hadoop using java persistence api. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 203–209. IEEE, 2010.

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

[36] H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*. Springer Science & Business Media, 1998.

[37] M. Maffina and R. RamPriya. An improved and efficient message passing interface for secure communication on distributed clusters. In *Recent Trends in Information Technology (ICRTIT), 2013 International Conference on*, pages 329–334. IEEE, 2013.

[38] M. Michael, J. E. Moreira, D. Shiloach, and R. W. Wisniewski. Scale-up x scale-out: A case study using nutch/lucene. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

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

[40] 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.

[41] 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.

[42] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[43] 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.

[44] R. S. Parpinelli and H. S. Lopes. Biological plausibility in optimisation: an ecosystemic view. *International Journal of Bio-Inspired Computation*, 4(6):345–358, 2012.

[45] D. Peralta, S. del Río, S. Ramírez-Gallego, I. Triguero, J. M. Benitez, and F. Herrera. Evolutionary feature selection for big data classification: A mapreduce approach. *Mathematical Problems in Engineering*, 501:246139, 2015.

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

[47] R. B. Sachin and M. S. Vijay. A survey and future vision of data mining in educational field. In *Advanced Computing & Communication Technologies (ACCT), 2012 Second International Conference on*, pages 96–100. IEEE, 2012.

[48] E. Sivaraman and R. Manickachezian. High performance and fault tolerant distributed file system for big data storage and processing using hadoop. In *Intelligent Computing Applications (ICICA), 2014 International Conference on*, pages 32–36. IEEE, 2014.

[49] C. Smutnicki. New trends in optimization. In *Intelligent Engineering Systems (INES), 2010 14th International Conference on*, pages 285–285. IEEE, 2010.

[50] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[51] P. P. Tallon. Corporate governance of big data: Perspectives on value, risk, and cost. *Computer*, 46(6):32–38, 2013.

[52] D. Wu, Z. Li, R. Bie, and M. Zhou. Research on database massive data processing and mining method based on hadoop cloud platform. In *Identification, Information and Knowledge in the Internet of Things (IIKI), 2014 International Conference on*, pages 107–110. IEEE, 2014.

[53] J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares, and X. Qin. Improving mapreduce performance through data placement in heterogeneous hadoop clusters. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–9. IEEE, 2010.

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

[55] Y. Yang, X. Ni, H. Wang, and Y. Zhao. Parallel implementation of ant-based clustering algorithm based on hadoop. In *Advances in Swarm Intelligence*, pages 190–197. Springer, 2012.

[56] H. Yu, J. E. Moreira, P. Dube, I.-h. Chung, and L. Zhang. Performance studies of a websphere application, trade, in scale-out and scale-up environments. In *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, pages 1–8. IEEE, 2007.

[57] I. Zelinka, D. Davendra, J. Lampinen, R. Senkerik, and M. Pluhacek. Evolutionary algorithms dynamics and its hidden complex network structures. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 3246–3251. IEEE, 2014.