UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Guilherme Priólli Daniel

Otimização de algoritmos de agrupamento espacial baseado em densidade aplicados em grandes conjuntos de dados

São José do Rio Preto
2016

Guilherme Priólli Daniel

Otimização de algoritmos de agrupamento espacial baseado em densidade aplicados em grandes conjuntos de dados

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Campus de São José do Rio Preto.

Financiadora: CAPES

Orientador: Prof. Dr. Carlos Roberto Valêncio

São José do Rio Preto
2016

Daniel, Guilherme Priólli.
Otimização de algoritmos de agrupamento espacial baseado em densidade aplicados em grandes conjuntos de dados / Guilherme Priólli Daniel. -- São José do Rio Preto, 2016
72 f. : il., tabs.

Orientador: Carlos Roberto Valêncio
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação - Matemática. 2. Sistemas de informação geográfica. 3. Sistemas de dados espaciais. 4. Computação em nuvem. 5. Big data. 6. Análise por agrupamento. 7. Algoritmos de computador. I. Valêncio, Carlos Roberto. II. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 518.72:91

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Guilherme Priólli Daniel

Otimização de algoritmos de agrupamento espacial baseado em densidade aplicados em grandes conjuntos de dados

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Financiadora: CAPES

Comissão Examinadora

Prof. Dr. Prof. Dr. Carlos Roberto Valêncio
UNESP – São José do Rio Preto, SP
Orientador

Profª. Drª. Rogéria Cristiane Gratão de Souza
UNESP – São José do Rio Preto, SP

Prof. Dr. Enzo Seraphim
UNIFEI – Itajubá, MG

São José do Rio Preto
12 de agosto de 2016

AGRADECIMENTOS

Primeiramente eu agradeço a Deus que me deu bênçãos para realizar mais uma etapa em minha vida.

Agradeço a minha mãe, Sra. Maria Gorete, que sempre me apoiou e deu todo o suporte para que eu pudesse concluir a pós-graduação. Agradeço ao meu pai, Sr. Vilson, que, mesmo não estando mais presente neste mundo, é uma grande inspiração para mim.

Também agradeço ao meu irmão, Otávio, e todos meus familiares – avós, tios e primos - que sempre estiveram presentes e torceram pelo meu sucesso.

Gostaria de agradecer ao Prof. Valêncio que mais uma vez confiou na minha capacidade, dedicação e trabalho e me possibilitou um grande aprendizado junto ao Grupo de Banco de Dados.

A todos os meus companheiros de GBD, principalmente Tico, André, Thati, Vitinho, que contribuíram para o meu trabalho e aprendizado.

Por fim, agradeço a todos os meus amigos que me apoiaram e torceram por minhas conquistas, em especial ao Rafael Medeiros que me ajudou no trabalho ao emprestar um computador e debater algumas ideias.

O sucesso é ir de fracasso em fracasso sem perder entusiasmo.
Winston Churchill

RESUMO

A quantidade de dados gerenciados por serviços Web de grande escala tem crescido significantemente e passaram a ser chamados de Big Data. Esses conjuntos de dados podem ser definidos como um grande volume de dados complexos provenientes de múltiplas fontes que ultrapassam a capacidade de armazenamento e processamento dos computadores atuais. Dentro desses conjuntos, estima-se que 80% dos dados possuem associação com alguma posição espacial. Os dados espaciais são mais complexos e demandam mais tempo de processamento que os dados alfanuméricos. Nesse sentido, as técnicas de MapReduce e sua implementação têm sido utilizadas a fim de retornar resultados em tempo hábil com a paralelização dos algoritmos de prospecção de dados. Portanto, o presente trabalho propõe dois algoritmos de agrupamento espacial baseado em densidade: o VDBSCAN-MR e o OVDBSCAN-MR. Ambos os algoritmos utilizam técnicas de processamento distribuído e escalável baseadas no modelo de programação MapReduce com intuito de otimizar o desempenho e permitir a análise em conjuntos Big Data. Por meio dos experimentos realizados foi possível verificar que os algoritmos desenvolvidos apresentaram melhor qualidade nos agrupamentos encontrados em comparação com os algoritmos tomados como base. Além disso, o VDBSCAN-MR obteve um melhor desempenho que o algoritmo sequencial e suportou a aplicação em grandes conjuntos de dados espaciais.

Palavras-chave: VDBSCAN-MR. OVDBSCAN-MR. Big Data. Prospecção de dados espaciais. Agrupamento espacial. MapReduce.

ABSTRACT

The amount of data managed by large-scale Web services has increased significantly and it arise to the status of Big Data. These data sets can be defined as a large volume of complex data from multiple data sources exceeding the storage and processing capacity of current computers. In such data sets, about 80% of the data is associated with some spatial position. Spatial data is even more complex and require more processing time than what would be required for alphanumeric data. In that sense, MapReduce techniques and their implementation have returned results timely with parallelization of data mining algorithms and could apply for Big Data sets. Therefore, this work develops two density-based spatial clustering algorithms: VDBSCAN-MR and OVDBSCAN-MR. Both algorithms use distributed and scalable processing techniques based on the MapReduce programming model in order to optimize performance and enable Big Data analysis. Throughout experimentation, we observed that the developed algorithms have better quality clusters compared to the base algorithms. Furthermore, VDBSCAN-MR achieved a better performance than the original sequential algorithm and it supported the application on large spatial data sets.

Keywords: VDBSCAN-MR. OVDBSCAN-MR. Big Data. Spatial data mining. Spatial clustering. MapReduce.

Lista de Figuras

Figura 1 - Relação dos 3 Vs do *Big Data* (Adaptado de (SINGH; SINGH, 2013)). . 16
Figura 2 - Exemplos de tipos de dados espaciais (YEUNG; HALL, 2007) (MEDEIROS, 2014). 18
Figura 3 - Exemplo de agrupamentos espaciais destacados pela cor verde (VALÊNCIO et al, 2013). 21
Figura 4 - Modelo de programação *MapReduce* (Adaptado de (HE et al, 2014)). 24
Figura 5 - Funcionamento do DBSCAN na busca por agrupamentos (VALÊNCIO et al, 2013). 26
Figura 6 - Exemplo do gráfico K-Dist (Adaptado de (LIU; ZHOU; WU, 2007)). 27
Figura 7 - Gráfico de Δd_k (Adaptado de (WANG et al, 2013)). 29
Figura 8 - Fluxograma do DBSCAN-MR (Adaptado de (DAI; LIN, 2012)). 29
Figura 9 - Exemplo das partições e das regiões de fronteira (DAI; LIN, 2012). 30
Figura 10 - Arquitetura do *Framework* (Adaptado de (YOO; BOULWARE, 2013)). 33
Figura 11 - Fluxograma do VDBSCAN-MR. 35
Figura 12 - Exemplo do fatiamento da partição (DAI; LIN, 2012). 40
Figura 13 - Fatia encontrada e novas partições (DAI; LIN, 2012). 40
Figura 14 - Exemplo das partições encontradas (DAI; LIN, 2012). 41
Figura 15 - Fluxograma do OVDBSCAN-MR. 47
Figura 16 - *Clusters* encontrados na aplicação do VDBSCAN-MR e VDBSCAN. 54
Figura 17 - *Clusters* encontrados na aplicação do OVDBSCAN-MR e OVDBSCAN. 55
Figura 18 - Agrupamentos encontrados pelo DBSCAN-MR. 56
Figura 19 - K-Dist gerado pelo algoritmo supervisionado. 60
Figura 20 - Agrupamentos obtidos pela execução supervisionada. 61
Figura 21 - Tempos de execução (em segundos) dos algoritmos VDBSCAN-MR e VDBSCAN. 63

Lista de Tabelas

Tabela 1 - Valores do Coeficiente de Silhueta dos agrupamentos do OVDBSCAN-MR. 57
Tabela 2 - Valores do Coeficiente de Silhueta dos agrupamentos do VDBSCAN-MR. 58
Tabela 3 - Valores do Coeficiente de Silhueta dos agrupamentos do DBSCAN-MR. 59
Tabela 4 - Valores do Coeficiente de Silhueta dos agrupamentos da execução supervisionada. 62
Tabela 5 – Tempo de execução dos algoritmos (em segundos) para cada quantidade de pontos. 63

Sumário

Capítulo 1 Introdução 11
1.1 Considerações Iniciais 11
1.2 Motivação 12
1.3 Objetivos e Metodologia 12
1.4 Organização do Trabalho 13
Capítulo 2 Fundamentação Teórica 14
2.1 Considerações Iniciais 14
2.2 *Big Data* 15
2.3 Análise de *Big Data* 17
2.4 Dados Espaciais 17
2.5 Prospecção de Dados Espaciais 18
2.6 Agrupamento de dados espaciais 20
2.7 Desafios 22
2.8 Computação em Nuvem 23
2.9 *MapReduce* 23
2.10 Algoritmos bases para o VDBSCAN-MR e OVDBSCAN-MR 25
2.10.1 DBSCAN 25
2.10.2 VDBSCAN 27
2.10.3 OVDBSCAN 28
2.10.4 DBSCAN-MR 29
2.11 Outros trabalhos no estado da arte 31
2.11.1 MR-DBSCAN 31
2.11.2 DBCURE-MR 32
2.11.3 *Framework* de prospecção de dados espaciais baseado em *MapReduce* 32
2.12 Considerações finais 33
Capítulo 3 Algoritmos de prospecção de dados espaciais para grandes conjuntos de dados 34
3.1 Considerações iniciais 34
3.2 VDBSCAN-MR 34
3.2.1 KDist-MR 35
3.2.2 Identificação dos valores de *Eps* 37
3.2.3 Estratégia de Particionamento 38
3.2.4 DBSCAN-MR 43
3.2.5 Processamento dos Resultados Obtidos 45
3.3 OVDBSCAN-MR 47
3.3.1 KDistList-MR 48
3.3.2 Encontra Valor de *k* 49
3.3.3 Encontra *Eps* 50
3.4 Qualidade dos resultados 50
3.5 Considerações Finais 51
Capítulo 4 Experimentos e Resultados 52
4.1 Considerações Iniciais 52
4.2 Testes de qualidade 52
4.2.1 Aplicação do VDBSCAN-MR 53

4.2.2 Aplicação do OVDBSCAN-MR 54
4.2.3 Comparação com o DBSCAN-MR 55
4.2.4 Estratégia de encontrar *Eps* automático 60
4.3 Teste de desempenho 62
4.4 Teste de escalabilidade 63
4.5 Considerações Finais 64
Capítulo 5 Conclusões 65
5.1 Considerações finais 65
5.2 Contribuições do trabalho 66
5.3 Trabalhos futuros 67
Referências 68

Capítulo 1
Introdução

1.1 Considerações Iniciais

Nos últimos anos a velocidade com que os dados são gerados e coletados tem motivado a criação aplicações que ultrapassam a escala de *exabytes* (10^{18}) de armazenamento. Este fator acrescido das características de heterogeneidade, complexidade e distribuição desses dados passaram a formar os conjuntos *Big Data*. O principal desafio em manipular esses conjuntos de dados decorre do rápido aumento de seu volume em comparação com a capacidade dos atuais sistemas de computação, no que se diz respeito aos recursos de armazenamento e processamento (KAISLER et al, 2013) (KATAL; WAZID; GOUDAR, 2013).

O termo *Big Data* tem se tornado comum na área de ciência da computação e pode ser definido como uma grande quantidade de dados que vai além da capacidade tecnológica de armazenar, gerenciar e processar eficientemente (KAISLER et al, 2013).

Além disso, aplicações que envolvem dados espaciais também têm sido impulsionadas em pesquisas nas áreas científica e industrial (ZHONG et al, 2012). Neste sentido, os dados espaciais estão inclusos nos conjuntos *Big Data*, e podem ser chamados de *Big Spatial Data*, os quais são gerados por meio de plataformas de mídias sociais, sensores remotos, redes de sensores, computadores de simulação e etc. Acredita-se que, atualmente, cerca de 80% dos dados são associados com uma posição espacial (TANG; FENG, 2014) (SHULIANG; GANGYI; MING, 2013).

Devido à alta complexidade de dados espaciais e seus relacionamentos a análise desses conjuntos de dados não é uma tarefa trivial. Dessa maneira, surgiu o conceito de prospecção

de dados espaciais, em inglês *Spatial Data Mining*, que consiste no processo de extrair conhecimentos implícitos, padrões interessantes, informações úteis e relações espaciais e não espaciais de grandes bases de dados espaciais (HEMALATHA; SARANYA, 2011).

Porém, a manipulação dos grandes conjuntos de dados espaciais por meio de processamento, análise e visualização tem sido pouco aplicada devido aos obstáculos de lidar com o tamanho e a complexidade desses dados (TANG; FENG, 2014).

1.2 Motivação

Cada vez mais as organizações necessitam de ferramentas que auxiliem na tomada de decisões. Assim, as técnicas de prospecção de dados espaciais têm sido aplicadas em diversas áreas. Contudo, o grande volume de dados espaciais das aplicações atuais dificulta o armazenamento e as técnicas de extração de conhecimento tornam-se pouco eficientes, o que exige uma arquitetura escalável para consultar dados espaciais em grandes bases de dados (KIM et al, 2014).

Recentemente, sistemas baseados em *MapReduce* foram propostos como uma solução escalável, de baixo custo e eficaz para o processamento de dados massivamente paralelo, o que torna possível suportar análises de grandes volumes de dados (BAKSHI, 2012) (AJI et al, 2013).

Portanto, a combinação das técnicas de prospecção de dados espaciais com a técnica de processamento em *MapReduce* tende a proporcionar aos algoritmos um melhor desempenho e melhor escalabilidade na prospecção de grande volume de dados espaciais.

1.3 Objetivos e Metodologia

Neste trabalho foram desenvolvidos dois algoritmos de agrupamento espacial baseado em densidade voltado para extração de conhecimento em grandes conjuntos de dados espaciais, são eles: OVDBSCAN-MR, derivado do OVDBSCAN (WANG et al, 2013), e VDBSCAN-MR, derivado do VDBSCAN (LIU; ZHOU; WU, 2007).

Ambos algoritmos foram adaptados dos algoritmos sequenciais de prospecção de dados espaciais para executar no modelo de programação *MapReduce*, cujo objetivo é obter desempenho e escalabilidade superiores aos algoritmos bases quando analisados em bancos de dados volumosos, sem que haja perda na qualidade dos resultados.

A metodologia de desenvolvimento adotada neste trabalho envolve as etapas de levantamento bibliográfico das áreas de *Big Data*, prospecção de dados espaciais e *MapReduce*, bem como fazer estudos comparativos com alguns trabalhos correlatos encontrados na literatura e implementação dos algoritmos propostos.

Os experimentos realizados para comprovar os resultados foram divididos em três grupos: testes de qualidade, que visa comparar os algoritmos com os seus algoritmos base; teste de desempenho, o qual busca comparar o tempo de execução do VDBSCAN-MR para diversos conjuntos de pontos; e teste de escalabilidade, o qual pretende-se mostrar que o VDBSCAN-MR pode ser aplicado em uma base com grande volume de dados espaciais.

1.4 Organização do Trabalho

A seguir é apresentada a estrutura deste trabalho:

- Capítulo 1 Introdução – são apresentadas as principais motivações do trabalho, assim como os objetivos e metodologia.

- Capítulo 2 Fundamentação teórica - são apresentados os principais conceitos que envolvem *Big Data* e sua análise, dados espaciais, prospecção de dados espaciais e *MapReduce*. Além disso, foram explicados os algoritmos base para os algoritmos desenvolvidos no trabalho e apresentados alguns algoritmos e ferramentas correlatas ao trabalho proposto.

- Capítulo 3 Trabalho desenvolvido - são descritos e detalhados os algoritmos desenvolvidos no trabalho, o VDBSCAN-MR e o OVDBSCAN-MR.

- Capítulo 4 Experimentos e resultados – contempla os experimentos realizados e a comparação e discussão dos resultados encontrados.

- Capítulo 5 Conclusões – são apresentadas as considerações finais referentes ao trabalho apresentado e os trabalhos futuros.

Capítulo 2
Fundamentação Teórica

2.1 Considerações Iniciais

Big Data é considerado o centro da ciência e do negócio moderno. É dito que irá revolucionar muitos campos, inclusive empresas, pesquisas científicas, administração pública, entre outros (SAGIROGLU; SINANC, 2013) (CHEN; ZHANG, 2014). Estes dados são gerados a partir de transações *on-line*, *e-mails*, vídeos, áudios, imagens, ferramentas de *streaming*, *logs*, mensagens, consultas de busca, registro de saúde, interações em redes sociais, dados científicos, sensores, celulares e aplicativos. As informações são armazenadas em bancos de dados que crescem massivamente e tornam-se difíceis de capturar, armazenar, gerenciar, compartilhar, analisar e visualizar por meio de ferramentas típicas de *software* de banco de dados (SAGIROGLU; SINANC, 2013).

Para a busca de conhecimento de aplicações que envolvem conjuntos *Big Data* foi necessário desenvolver uma técnica chamada de análise de *Big Data* que consiste em pesquisas para revelar padrões e correlações ocultas, a qual possibilitou melhor compreensão de um determinado domínio (SAGIROGLU; SINANC, 2013) (CHEN; ZHANG, 2014). Portanto, o desafio é armazenar e gerenciar o grande volume de dados, além de analisar e extrair conhecimento significativo a partir deles (BAKSHI, 2012).

2.2 *Big Data*

Big Data é uma coleção de conjuntos de dados em massa, com uma grande diversidade de tipos, que dificulta o processamento de dados que fazem uso das abordagens tradicionais de computação (CHEN; ZHANG, 2014). O termo abrange a complexidade, a variedade, os tipos e a coleta de dados, bem como o processamento em tempo real para obter valor por meio de análises inteligentes (TALIA, 2013).

Normalmente, este conceito se estende por três dimensões, chamados de 3 Vs, são eles (SINGH; SINGH, 2013):

- **Volume** – é um elemento de destaque do *Big Data* e se refere à quantidade de dados disponíveis para uma organização. O aumento desses dados em grande escala ultrapassa o escopo habilitado pelas técnicas de armazenamento e de análise atuais (KAISLER et al, 2013) (KATAL; WAZID; GOUDAR, 2013) (SAGIROGLU; SINANC, 2013) (SINGH; SINGH, 2013).

- **Velocidade** – atualmente a velocidade é requerida não só em *Big Data*, mas para todos os processos de computação. Esta característica não está limitada à velocidade de dados de entrada, mas também a velocidade do fluxo de dados. Assim, nossos sistemas não são capazes o suficiente para executar as análises sobre os dados que apresentam um comportamento dinâmico (KATAL; WAZID; GOUDAR, 2013) (SAGIROGLU; SINANC, 2013) (SINGH; SINGH, 2013).

- **Variedade** – os dados produzidos vêm de várias fontes e não são de uma única categoria, que podem ser: texto, imagem, vídeo, áudio, etc. Além disso, os dados podem ser estruturados, semiestruturados e não estruturados. Todos estes dados totalmente diferentes são difíceis de serem tratados pelos sistemas analíticos tradicionais existentes (KAISLER et al, 2013) (KATAL; WAZID; GOUDAR, 2013) (SAGIROGLU; SINANC, 2013) (SINGH; SINGH, 2013).

Na Figura 1 é possível ver a relação dos 3 Vs do *Big Data*.

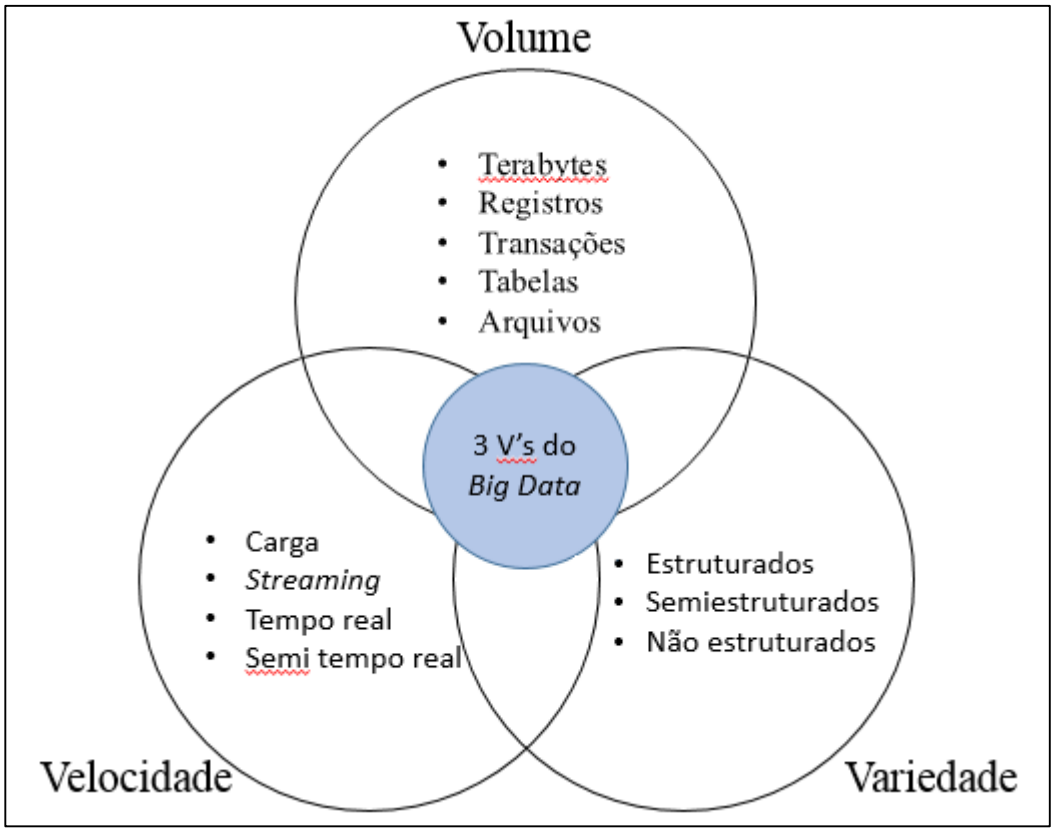

Figura 1 - Relação dos 3 Vs do *Big Data* (Adaptado de (SINGH; SINGH, 2013)).

No estado da arte é possível encontrar outras propriedades estendidas ao conceito de *Big Data*, como por exemplo, variabilidade, veracidade, valor e complexidade, que estão definidas a seguir:

- **Variabilidade** - considera as inconsistências do fluxo de dados. Cargas de dados tornam-se difíceis de serem mantidas, especialmente com o aumento do uso dos meios de comunicação social, que geralmente provoca pico de cargas de dados de acordo com certos eventos que ocorrem (KATAL; WAZID; GOUDAR, 2013).
- **Veracidade** – é a habilidade de confiar nas informações, uma vez que as fontes de dados são de qualidades distintas, devido às diferenças de compreensão, precisão e atualização dos dados fornecidos (DONG; SRIVASTAVA, 2015).
- **Valor** - mede a utilidade dos dados na tomada de decisões, ou seja, pode-se inferir resultados interessantes a partir dos dados selecionados e classificá-los de acordo com as necessidades. Esses relatórios ajudam a encontrar tendências de negócios, que visam a mudança das estratégias (KAISLER et al, 2013) (KATAL; WAZID; GOUDAR, 2013).
- **Complexidade** - mede o grau de interligação e interdependência de grandes estruturas de dados de tal forma que uma pequena mudança em um ou alguns

elementos podem produzir grandes, pequenas ou nenhuma alteração no sistema e afetar ou não seu comportamento (KAISLER et al, 2013).

2.3 Análise de *Big Data*

Análise de *Big Data* é um tema abordado atualmente e pode ser entendido como o processo de examinar dados em massa, heterogêneos e desestruturados a fim de descobrir padrões ocultos, correlações desconhecidas e outras informações úteis, com o intuito de auxiliar na tomada de decisões (TALIA, 2013) (ZHENG; ZHU; LYU, 2013).

As análises tradicionais não são suficientes para extrair valores a partir de conjuntos *Big Data* e automatizar a tomada de decisões. Muitas vezes, os analistas devem explorar os dados visualmente ou numericamente a fim de obter conhecimento. Por outro lado, as análises avançadas buscam combinar técnicas de análise de *Big Data* com técnicas de descoberta de conhecimento para inferir relações complexas entre os dados e medir quantitativamente os valores e qualidade dos dados, a fim de produzir conhecimento em um tempo reduzido (TALIA, 2013) (OSMAN; EL-REFAEY; ELNAGGAR, 2013).

Para a análise de *Big Data* podem ser utilizadas várias técnicas, como: aprendizagem de máquina, computação em nuvem, *crowdsourcing*, *Text Mining*, *Data Mining*, *Spatial Data Mining*, análise de séries temporais, processamento de fluxo e visualização (ZHANG, 2013).

2.4 Dados Espaciais

Com o rápido crescimento de dados espaciais impulsionados por aplicações industriais e científicas, estes conjuntos de dados têm evoluído para as características de *Big Data*, os quais passaram a ser considerados *Big Spatial Data* (AJI et al, 2013) (TANG; FENG, 2014).

Os dados espaciais fazem referência à localização espacial e emergiram a fim de simbolizar fenômenos geográficos do mundo real. Normalmente, são representados por objetos geométricos multidimensionais, como por exemplo: pontos, linhas, polígonos, entre outros. Dessa maneira, esses objetos espaciais são mais complexos que os objetos convencionais. Em comparação com outros tipos de dados, os dados espaciais contêm características espaciais, temporais, multidimensional e volume. Os predicados de consultas espaciais são complexos, uma vez que as consultas são baseadas não apenas sobre o valor dos

atributos alfanuméricos, mas também sobre a localização espacial e suas relações (ZHONG et al, 2012) (SHULIANG; GANGYI; MING, 2013) (WANG et al, 2009). Na Figura 2 são exemplificados alguns tipos de dados espaciais existentes.

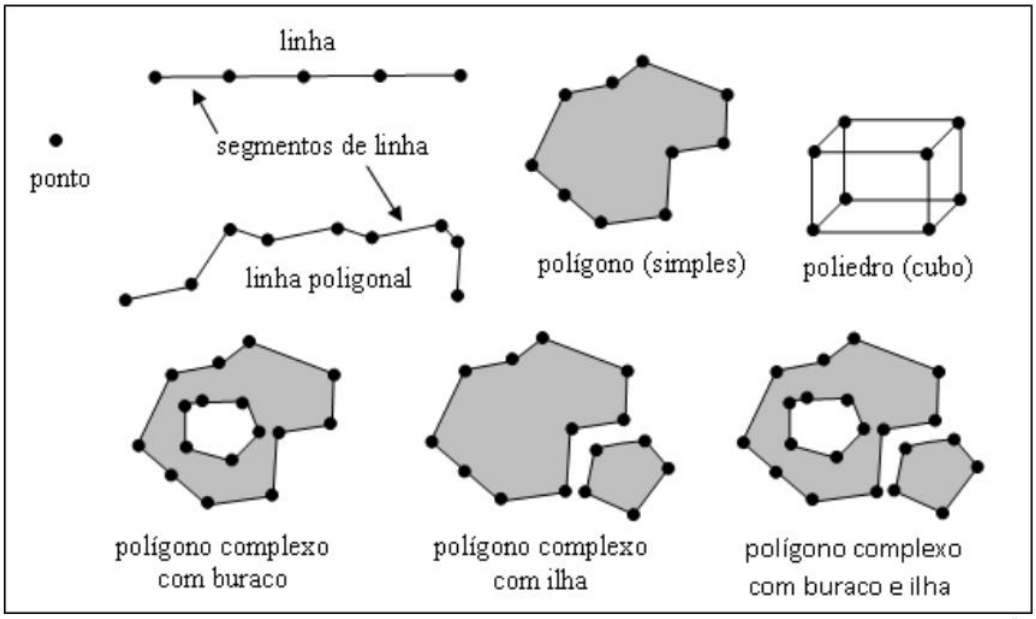

Figura 2 - Exemplos de tipos de dados espaciais (YEUNG; HALL, 2007) (MEDEIROS, 2014).

Por conta dessas características, a manipulação eficiente de grandes conjuntos de dados espaciais por meio de processamento, análise e visualização não têm sido exploradas profundamente. Porém, esta manipulação é importante para a descoberta de conhecimento espacial interessante que não é viável de ser feita por meio de técnicas de computação convencional, pois o processamento requer intensivos acessos de E/S no disco e computação espacial. Sendo assim, os conceitos de computação em nuvem e computação de alto desempenho são amplamente utilizados para realizarem essas tarefas (ZHONG et al, 2012) (TANG; FENG, 2014).

Ao se trabalhar com dados espaciais, a análise de *Big Data* pode ser feita por meio das técnicas de prospecção de dados espaciais para poder extrair informações relevantes dos conjuntos de *Big Spatial Data*.

2.5 Prospecção de Dados Espaciais

Prospecção de dados espaciais é o processo de descobrir padrões interessantes e até então desconhecidos, mas potencialmente úteis em banco de dados espaciais. Normalmente, as

técnicas de prospecção de dados espaciais são derivadas das técnicas convencionais de prospecção de dados. Porém, extrair padrões interessantes e úteis a partir de conjuntos de dados espaciais é mais complexo que extrair os padrões de dados alfanuméricos, devido à complexidade de tipos de dados espaciais, relações espaciais, e auto correlação espacial (SHEKHAR et al, 2003) (HEMALATHA; SARANYA, 2011).

Algumas das diversas técnicas utilizadas para extrair conhecimento de dados espaciais estão descritas a seguir (JIN; MIAO, 2010) (MENNIS; GUO, 2009):

- Classificação espacial - possui a finalidade de categorizar os dados segundo suas características. Sendo assim, os métodos de classificação espacial consideram os atributos espaciais e não espaciais do objeto, além dos atributos dos objetos vizinhos e as suas relações espaciais;

- Caracterização espacial – consiste na descrição de predicados espaciais ou não espaciais para a identificação de características de um grupo de dados específico;

- Associação espacial – consiste na geração de regras que definem a correlação ou associação de um objeto a outros e na identificação de padrões e frequências em um conjunto de dados por meio de seus atributos espaciais ou não espaciais;

- Agrupamento espacial – são formados agrupamentos dos itens de um conjunto de dados, em que são considerados a posição geográfica dos objetos e as suas características, de modo que os objetos de um mesmo grupo são semelhantes entre si e diferentes dos de outros grupos;

- Tendência espacial – tem por objetivo encontrar padrões relacionados às mudanças e tendências dos atributos não espaciais a partir do movimento de um objeto espacial;

- Geovisualização – explora o uso de técnicas e ferramentas visuais para a exibição dos dados a fim de facilitar o entendimento e análise dos dados geoespaciais.

Dentre as técnicas de algoritmos de prospecção de dados espaciais existentes na literatura, destaca-se a categoria de agrupamento espacial, pois possibilita a identificação dos agrupamentos de maneira simples a partir dos pontos geográficos, não é necessário conhecimento prévio e pode ser aplicada em diversos tipos de aplicações (PATTABIRAMAN et al., 2009).

2.6 Agrupamento de dados espaciais

A técnica de agrupamento espacial, também chamada como *clustering*, é muito útil na área de mineração de dados e é usada para descobrir padrões de distribuição nos dados. O agrupamento é feito com base na similaridade das características e na posição dos objetos. Dessa maneira, o objetivo é que objetos de mesmo agrupamento sejam muito similares entre si e muito diferentes dos objetos de outros agrupamentos (JIN; MIAO, 2010) (KIM et al, 2014).

Os métodos de agrupamento espacial podem ser classificados em quatro grupos (JIN; MIAO, 2010) (KIM et al, 2014) (PATEL; MENARIA, 2014):

- Métodos hierárquicos – buscam construir uma hierarquia entre os agrupamentos encontrados. Assim, os agrupamentos são gerados em várias iterações, sendo que a cada iteração do algoritmo diminui o espaço da busca. Além disso, esses métodos podem ser subdivididos em dois tipos: aglomeração e divisão.
- Métodos de particionamento – consistem em particionar um conjunto de objetos em grupos não sobrepostos, ou seja, cada objeto pertencente a um agrupamento diferente. Além disso, cada agrupamento pode ser representado por um centroide. Os itens são inseridos em um agrupamento com base numa medida de proximidade ou de diferença.
- Métodos baseados em densidade – analisam a densidade dos objetos, ou seja, calculam o número de objetos em uma determinada região de objetos de dados a fim de formar agrupamentos dos pontos e seus vizinhos. Assim, tais algoritmos são capazes de encontrar agrupamentos de formas arbitrárias.
- Métodos baseados em grade – consistem na utilização de uma estrutura que divide toda a região em número limitado de células e trabalham com os objetos pertencentes a estas células.

Na Figura 3, é possível ver agrupamentos espaciais obtidos a partir da aplicação de um algoritmo de agrupamento espacial baseado em densidade, destacados pelos objetos geométricos de cor verde.

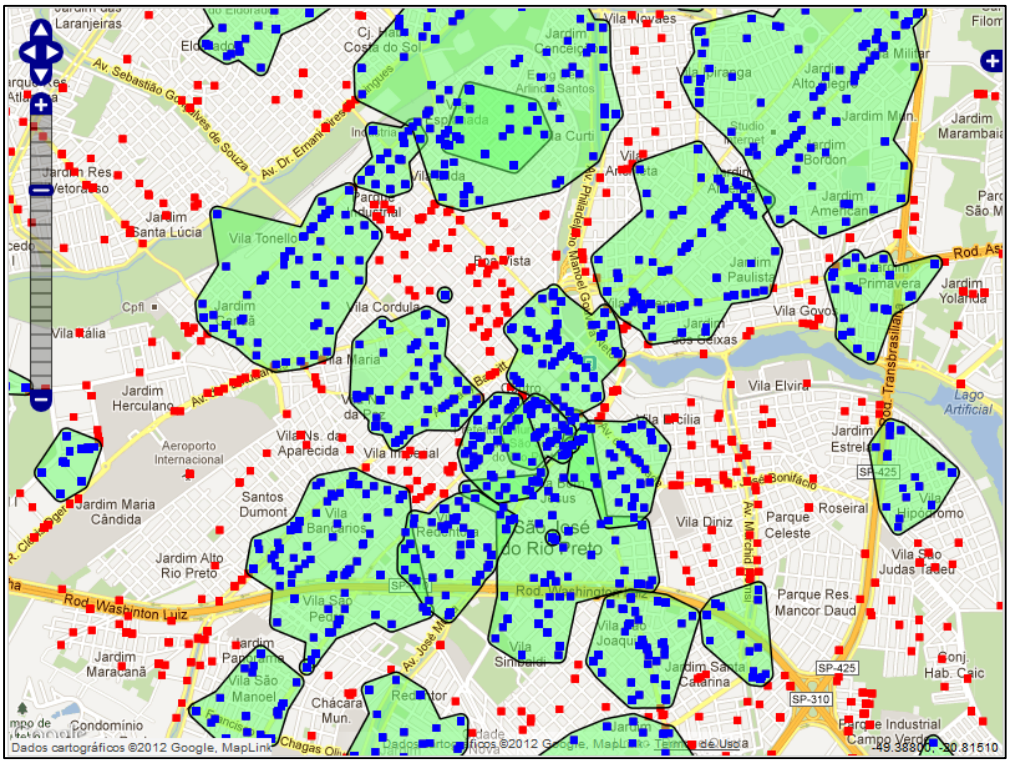

Figura 3 - Exemplo de agrupamentos espaciais destacados pela cor verde (VALÊNCIO et al, 2013).

Os algoritmos de agrupamento têm se mostrado relevantes para análise de sistemas com grandes volumes de dados (FAHAD et al, 2014). A vantagem de se utilizar este processo é que os agrupamentos são encontrados sem a necessidade de se ter um conhecimento preliminar dos dados analisados, além de possuir inúmeras aplicações que demandam desse tipo de técnica, tais como: sistemas de informações geográficas, análise de imagens de satélite, análise de imagens médicas, marketing, biologia molecular, dados multimídia e etc. (JIN; MIAO, 2010) (KIM et al, 2014).

Apesar das vantagens, alguns problemas na utilização de algoritmos de agrupamento é a falta de consenso na definição de suas propriedades e a falta de uma categorização formal. Ao avaliar métodos de agrupamento em conjuntos *Big Data*, alguns critérios específicos devem ser levados em consideração na avaliação de pontos fortes e fracos dos algoritmos, principalmente as propriedades de volume, velocidade e variedade. Assim, as características que devem ser analisadas são: tamanho da base de dados, alta dimensionalidade, manipulação de ruídos, tipo de dados, formato dos agrupamentos, complexidade do algoritmo e parâmetros de entrada (FAHAD et al, 2014).

2.7 Desafios

Embora a análise de *Big Data* tenha trazido benefícios reais e significativos, esta área enfrenta ainda muitos desafios técnicos que devem ser abordados. O volume causa problemas no armazenamento e manipulação dos dados, mas apesar de ser o problema mais tratado, não é o único. A variedade gera dificuldade na representação e na interpretação semântica, visto que os dados vêm de diversas fontes. O problema da velocidade é por conta da demanda de processamento na pesquisa, recuperação e visualização dos dados. A veracidade gera incerteza nos resultados obtidos (LABRINIDIS; JAGADISH, 2012) (ZHANG, 2013).

Outro desafio é como lidar corretamente com os vários tipos de inconsistências e conflitos durante o processo de extração de conhecimento, principalmente nas fases de pré-processamento e análise. Além disso, ainda existem problemas adicionais como privacidade, segurança, proveniência e modelagem (ZHANG, 2013).

Portanto, os principais objetivos que devem ser tratados para contornar os problemas e desafios das aplicações de *Big Data* são:

- Projetar sistemas capazes de lidar com a grande quantidade de dados de forma eficiente e eficaz (KAISLER et al, 2013) (KATAL; WAZID; GOUDAR, 2013).

- Filtrar os dados mais importantes e analisá-los para extrair significado relevante e confiável para a tomada de decisão (KAISLER et al, 2013) (KATAL; WAZID; GOUDAR, 2013).

Por conta desses desafios enfrentados ao se trabalhar com *Big Data*, as técnicas e algoritmos de análise de dados estão além do alcance da capacidade de TI de uma organização (ZHENG; ZHU; LYU, 2013). Sendo assim, as análises tradicionais não são suficientes para extrair valores em conjuntos *Big Data*, pois atualmente as técnicas exigem abordagens de alto desempenho para produzir resultados em tempo hábil. Dessa maneira, infraestruturas de computação em nuvem têm sido utilizadas para suprir as necessidades computacionais de armazenamento e processamento desse volume de dados (TALIA, 2013) (OSMAN; EL-REFAEY; ELNAGGAR, 2013). Suportada pela arquitetura de computação em nuvem, a técnica de *MapReduce* surgiu como uma solução escalável e eficiente para programação paralela para suportar a análise de *Big Data* (AJI et al, 2013).

2.8 Computação em Nuvem

A computação em nuvem é um novo paradigma de computação orientada a serviços que surgiu nos últimos anos e tem sido bem-sucedida, pois revolucionou a maneira como a infraestrutura de computação é utilizada. Suas características são: a elasticidade, o baixo investimento inicial, a possibilidade de pagar pelo que usar (*pay-per-use*), baixo tempo de mercado e transferência de risco (AGRAWAL; DAS; EL ABBADI, 2011) (ZHANG et al, 2011).

Os modelos de serviços de nuvens mais populares são: Infraestrutura como um Serviço (IaaS), Plataforma como um Serviço (PaaS) e Software como um Serviço (SaaS). Este conceito também pode ser estendido para Banco de Dados como um Serviço ou Armazenamento como um Serviço (AGRAWAL; DAS; EL ABBADI, 2011).

Com a proliferação de aplicativos que utilizam plataformas de computação em nuvem, resultou um grande aumento na escala de dados gerados (AGRAWAL; DAS; EL ABBADI, 2011). Sendo assim, os serviços de computação em nuvem oferecem soluções escaláveis de acesso a recursos de maneira ágil, escalável e eficiente para o tratamento de conjuntos *Big Data* (ZHANG et al, 2011) (O'DRISCOLL; DAUGELAITE; SLEATOR, 2013).

Pode-se dizer também que a computação em nuvem está fortemente ligada ao processamento paralelo e distribuído, pois suas aplicações são baseadas no modelo cliente-servidor, onde um *software* executa na máquina do cliente e todo o processamento é realizado na nuvem (MARINESCU, 2013).

Ligado ao modelo de computação em nuvem e programação distribuída, surgiu o conceito de *MapReduce*, que é uma tecnologia de processamento e análise de dados que tem sido revolucionária no campo da ciência da computação e é uma das tecnologias mais utilizadas para processamento de *Big Data* (O'DRISCOLL; DAUGELAITE; SLEATOR, 2013).

2.9 *MapReduce*

Atualmente, existem diversas estratégias que buscam aperfeiçoar o desempenho de algoritmos. No entanto, *MapReduce* é uma tecnologia revolucionária na área de processamento e análise de dados, além de ser comumente utilizada para o processamento de aplicações em conjuntos *Big Data* (O'DRISCOLL; DAUGELAITE; SLEATOR, 2013).

MapReduce é um modelo de programação proposto pelo Google em 2003, com a finalidade de desenvolver aplicações paralelas de maneira simples e poderosa para o processamento eficiente de grandes quantidades de dados em uma grande rede de computadores (LOEWEN; GALLOWAY; VRBSKY, 2013) (WHITE, 2015). Este modelo permite ao desenvolvedor executar o programa de forma distribuída sem se preocupar com os detalhes da programação distribuída (CHEN; ZHENG; CHEN, 2013).

A arquitetura do *MapReduce* é formada por um conjunto de máquinas conectadas em rede, onde um nó é definido como mestre e os outros como escravos. O mestre é responsável por controlar a aplicação e dividir as tarefas, enquanto os escravos são os nós responsáveis pelo processamento dos dados (O'DRISCOLL; DAUGELAITE; SLEATOR, 2013) (WHITE, 2015).

Em resumo, o fluxo de execução do programa é baseado em duas funções principais: *Map* e *Reduce*. Ao iniciar a execução do programa, o *Map* irá processar os dados de entrada e gerar um conjunto de pares chave/valor intermediário. Por fim, a função *Reduce* é aplicada aos valores que possuem a mesma chave, com o propósito de combinar os dados derivados adequadamente (CHEN; ZHENG; CHEN, 2013). O modelo de programação em *MapReduce* pode ser visto na Figura 4.

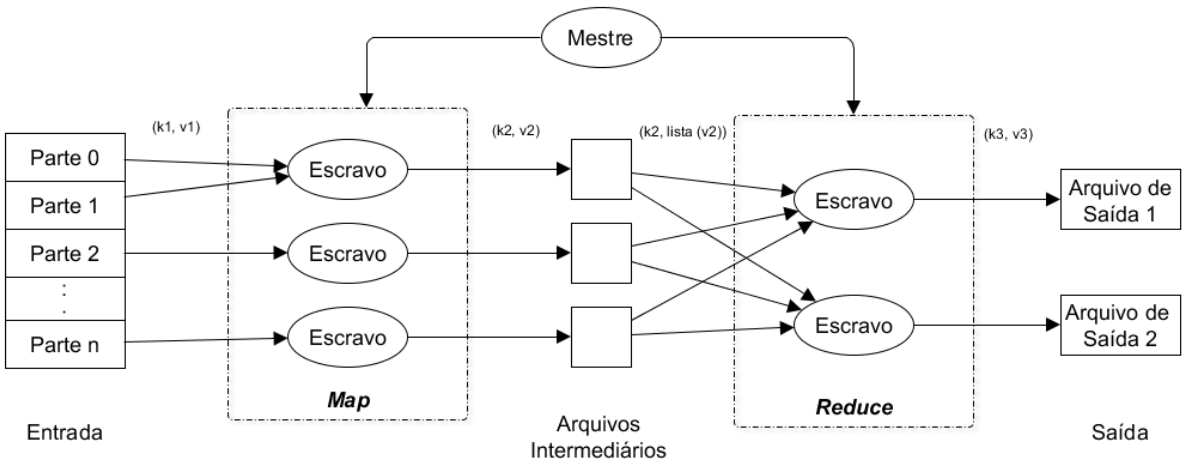

Figura 4 - Modelo de programação *MapReduce* (Adaptado de (HE et al, 2014)).

Numa visão mais detalhada, para cada entrada *(k1, v1)*, a função *Map* gera uma lista de saída intermediária *(k2, v2)*. Em seguida, na fase intermediária, também conhecida como embaralhamento, em inglês *shuffle*, os pares intermediários são agrupados como *(k2, lista (v2))* e transferidos para a função *Reduce*. Por fim, a função *Reduce* gera a lista de pares de saída final *(k3, v3)* (HE et al, 2014).

Dentre as plataformas que se baseiam no modelo *MapReduce*, o Hadoop é um dos *frameworks* mais utilizados, mantido pela *Apache Software Foundation*, implementado em Java, proporciona um alto nível de abstrações, fácil de usar e tem se mostrado eficiente para análise de *Big Data* (AJI et al, 2013) (ELDAWY; MOKBEL, 2013).

O Hadoop é uma plataforma de software de código aberto baseada na solução proposta pelo Google, em que fornece uma computação confiável, escalável e distribuída (O'DRISCOLL; DAUGELAITE; SLEATOR, 2013) (WHITE, 2015). Sua estrutura é composta por quatro módulos:

- *Hadoop Common* - utilitários comuns que suportam os outros módulos.
- *Hadoop Distributed File System* (HDFS) - sistema de arquivos distribuídos.
- *Hadoop YARN* - plataforma de escalonamento e gestão de recurso de *cluster*.
- *Hadoop MapReduce* - sistema baseado no YARN para o processamento distribuído dos dados.

Nos últimos anos, sistemas de *MapReduce* tem se mostrado eficiente para análise de conjuntos *Big Data* em diversos tipos de aplicação. Inclusive vários algoritmos de prospecção de dados têm sido adaptados e implementados com uso de tal tecnologia, de modo que o programa execute de maneira distribuída, e obtiveram bons resultados (ELDAWY; MOKBEL, 2013) (KIM et al, 2014).

2.10 Algoritmos bases para o VDBSCAN-MR e OVDBSCAN-MR

Dos métodos de agrupamentos existentes na literatura, os algoritmos baseados em densidade são bastante utilizados, cujo DBSCAN é um dos mais consagrados. Desta maneira, a grande maioria dos trabalhos que serviram como base para os algoritmos VDBSCAN-MR e OVDBSCAN-MR derivam do DBSCAN. A seguir são apresentadas as particularidades de cada algoritmo.

2.10.1 DBSCAN

O DBSCAN, sigla que vem do inglês *Density Based Spatial Clustering of Applications with Noise*, foi desenvolvido em 1996 e tinha como objetivo encontrar agrupamentos e ruídos

em bases de dados espaciais. Assim, utiliza-se de dois parâmetros de entrada para sua execução: o primeiro refere-se ao número de pontos mínimos que um agrupamento deve possuir, denominado *MinPts*, o segundo é o raio em que será efetuada a busca por vizinhos a partir do ponto selecionado, denominado *Eps* (ESTER et al, 1996).

Para encontrar os agrupamentos, o DBSCAN inicia a busca por um ponto arbitrário e compara a distância deste ponto com todos os outros pontos da base de dados. O ponto é marcado como centro se em uma circunferência de raio *Eps* possuir o número mínimo de pontos para formar o agrupamento. O ponto é considerado borda se na circunferência *Eps* existem pontos alcançáveis, mas não o suficiente para formar um agrupamento. Por fim, o ponto é denominado ruído, caso o ponto não seja alcançável por nenhum outro numa circunferência de raio *Eps*. Este processo é executado até que todos os pontos sejam marcados. Na Figura 5 é possível ver o comportamento do DBSCAN na classificação dos pontos, onde o ponto roxo é centro, o verde é borda, o vermelho é ruído e os azuis ainda não foram processados.

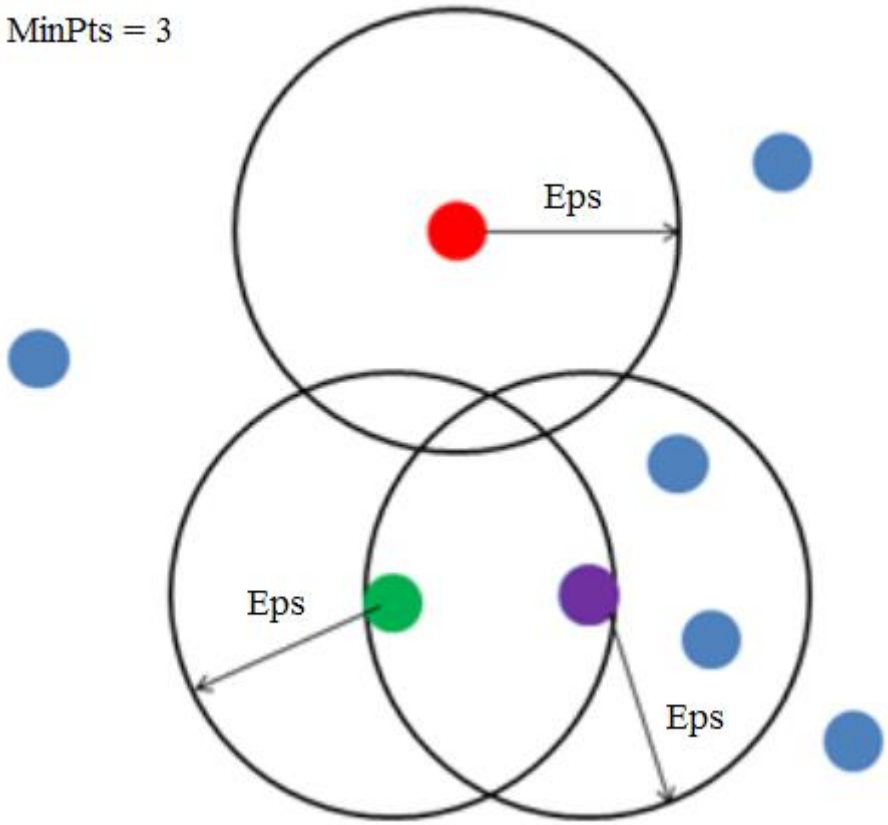

Figura 5 - Funcionamento do DBSCAN na busca por agrupamentos (VALÊNCIO et al, 2013).

2.10.2 VDBSCAN

Em bases de dados reais, os pontos podem estar espalhados em diferentes densidades, o que dificulta a formação de agrupamentos significativos pelo DBSCAN. Dessa forma, em 2007, foi proposto o algoritmo VDBSCAN, do inglês *Varied Density Based Spatial Clustering of Applications with Noise*, em que apesenta uma contribuição no algoritmo DBSCAN tradicional, onde oferece a formação de agrupamentos com densidades variadas (LIU; ZHOU; WU, 2007).

A implementação do VDBSCAN elimina o parâmetro de entrada *Eps*, uma vez que as buscas serão realizadas com valores de raios diferentes, e divide a execução em duas partes importantes: encontrar valores de *Eps* e formação de agrupamentos.

Para encontrar os valores de *Eps*, são aferidas as distâncias de cada ponto até seu k-ésimo vizinho mais próximo, onde *k* é igual ao parâmetro *MinPts* do DBSCAN, e a partir desses cálculos é gerado um gráfico, chamado de K-Dist, que contém essas distâncias de forma ordenada. Para cada mudança brusca no valor do gráfico é escolhido visualmente um valor para o raio *Eps* a ser buscado na base de dados. Na Figura 6 é mostrado um exemplo do gráfico K-Dist, onde os valores de *Eps* identificados foram 2,0 e 4,0, pois foram os pontos em que ocorrem mudanças bruscas na distribuição dos dados.

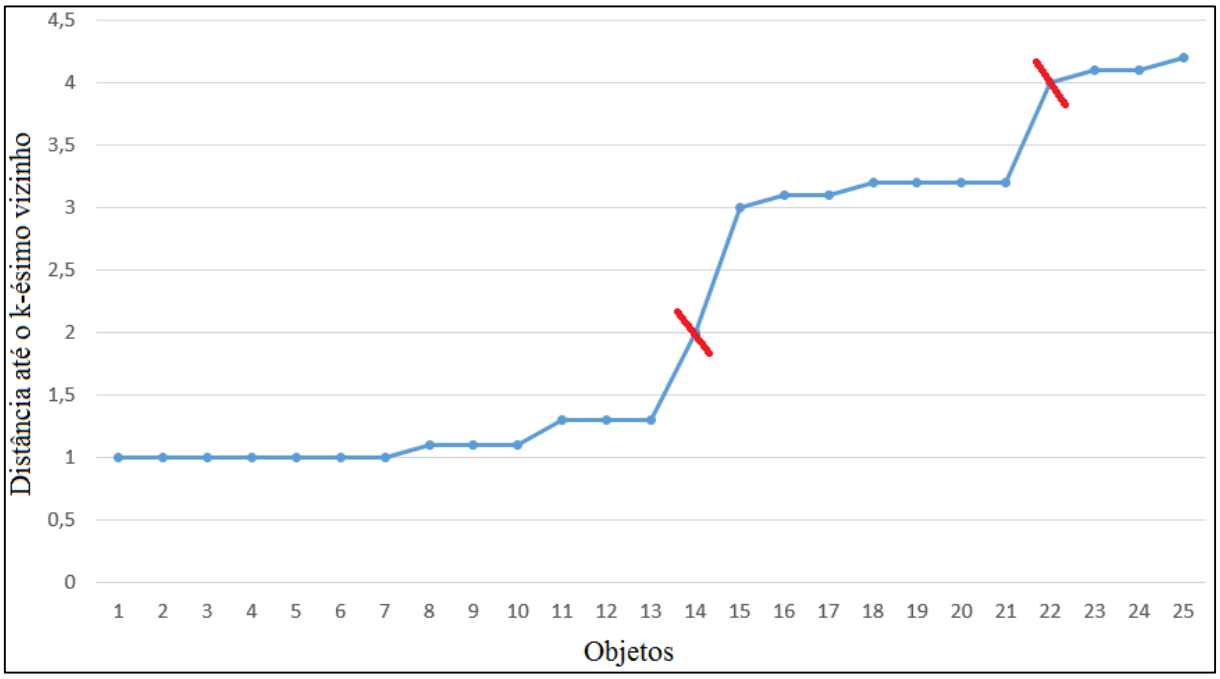

Figura 6 - Exemplo do gráfico K-Dist (Adaptado de (LIU; ZHOU; WU, 2007)).

Na etapa de formação de agrupamentos, o DBSCAN tradicional é executado para cada valor de *Eps* encontrado na etapa anterior.

2.10.3 OVDBSCAN

Embora o VDBSCAN seja muito efetivo em encontrar agrupamentos de densidade variada, ainda é difícil saber qual o melhor valor de *k*, ou seja, o número mínimo de pontos para formar o agrupamento. Sendo assim, em 2013, uma nova proposta surgiu para selecionar automaticamente o valor ótimo de *k* para o VDBSCAN de acordo com a densidade da base de dados em questão, isto porque diferentes valores de *k* podem resultar em *clusters* diferentes. Este algoritmo foi denominado OVDBSCAN, que consiste em adicionar apenas a etapa de encontrar o valor ótimo para *k* antes de executar o VDBSCAN (WANG et al, 2013).

Para encontrar o valor ótimo de *k*, inicialmente, é definido um domínio de possíveis valores de *k*. Depois, são calculados os gráficos K-Dists de todos os objetos para todos os valores de *k* do domínio estabelecido. Em seguida, são calculadas as taxas de variação de cada gráfico K-Dist e as coloca em um outro gráfico, chamado de $\Delta d_{k,}$ e, a partir deste novo gráfico, é escolhido um limiar de busca, que é determinado pelo maior valor do gráfico descartando os valores 20% mais altos. Após isso, o valor de *k* é escolhido para o primeiro Δd_k encontrado que excede o limiar definido. Por fim, o VDBSCAN é executado para o *k* selecionado.

Na Figura 7 é mostrado um exemplo do gráfico do Δd_k para um domínio inicial de *k* sendo [1, 10]. Neste caso, o limiar fixado foi $\Delta d_k = 1.0178$, pois acima disso estão os 20% maiores valores, o que é considerado muito alto. Portanto, como o valor de *k* é o primeiro Δd_k que excede o limiar, escolhe-se $k = 5$.

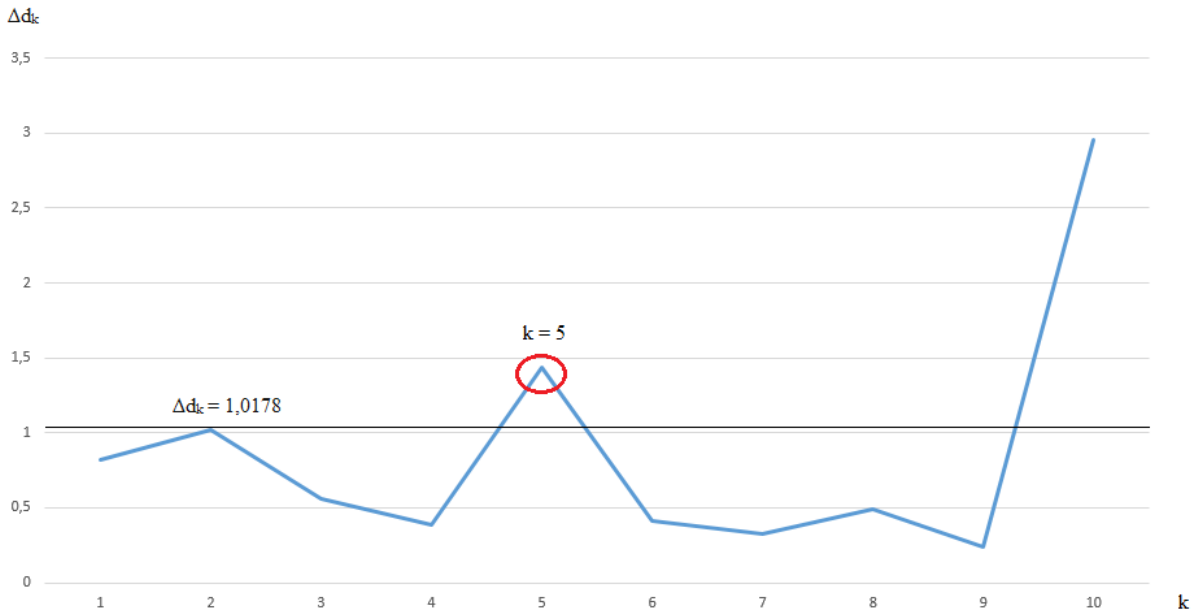

Figura 7 - Gráfico de Δd_k (Adaptado de (WANG et al, 2013)).

2.10.4 DBSCAN-MR

Com o crescente aumento nos conjuntos de dados, os algoritmos de agrupamento passaram a não suportar tais aplicações. Dessa maneira, em 2012, foi proposto um novo algoritmo para solucionar o problema de escalabilidade do DBSCAN tradicional, o qual foi nomeado de DBSCAN-MR, ou seja, *DBSCAN with Map/Reduce*. Tal algoritmo melhora a escalabilidade do DBSCAN tradicional ao particionar o espaço de busca em regiões menores e paralelizar o processamento na plataforma Hadoop (DAI; LIN, 2012).

Os experimentos foram realizados com quatro bases de dados sintéticas e uma base de dados real, em que a mais volumosa continha um milhão de pontos. Com os resultados foi possível verificar que o DBSCAN-MR teve maior eficiência e escalabilidade que os algoritmos sequenciais (DAI; LIN, 2012).

Na Figura 8 são apresentadas as fases do DBSCAN-MR.

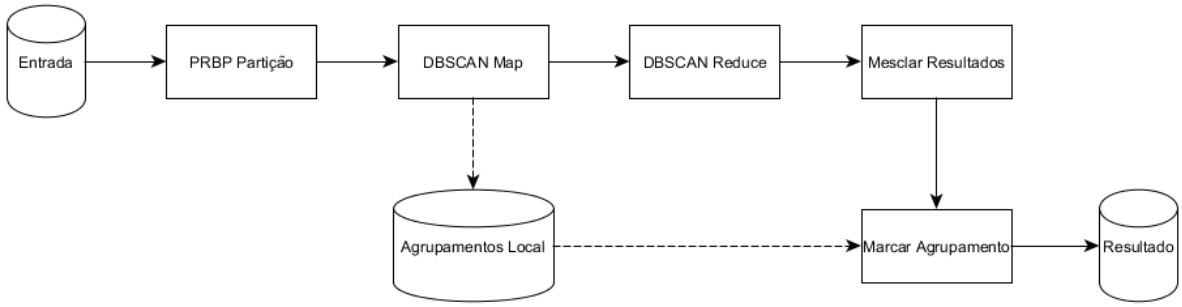

Figura 8 - Fluxograma do DBSCAN-MR (Adaptado de (DAI; LIN, 2012)).

Primeiramente, o conjunto de dados deve ser particionado, onde cada nó agrupa um subconjunto dos dados originais separadamente. Para particionar a região de busca, foi desenvolvido o algoritmo PRPB, da sigla em inglês *Partition with Reduced Boundary Points*, ou seja, partição com redução dos pontos de fronteira. Este método consiste em balancear as cargas de cada nó e escolher as melhores regiões de divisões. Como os pontos de um mesmo agrupamento podem estar em diferentes partições, os pontos de fronteira são duplicados em ambas as partições para possibilitar encontrar estes casos nas próximas etapas.

Antes de iniciar o particionamento, é verificado se a região possui o número mínimo de pontos necessários para ser repartida. Se puder ser dividida, a região é delimitada em fatias verticais e horizontais de valor duas vezes maior que o raio de busca dos agrupamentos. Depois, para cada fatia é contado o total de pontos presentes e verificado o balanceamento de carga. A fatia com menor número de pontos e que satisfaça o balanceamento de carga é escolhida para dividir a partição. Este processo é feito até que não tenha mais partições para se dividir.

Na Figura 9 é possível ver um exemplo da divisão do espaço, das partições encontradas, as regiões de fronteira e seus pontos. As partições estão representadas pelos retângulos contornados em vermelho, as regiões de fronteira pelas fatias coloridas em verde, amarelo e rosa, onde os retângulos azuis são as intersecções entre as regiões de fronteira, e os pontos estão representados pelas circunferências, sendo que os pontos preenchidos com a cor vermelha são considerados pontos de fronteira.

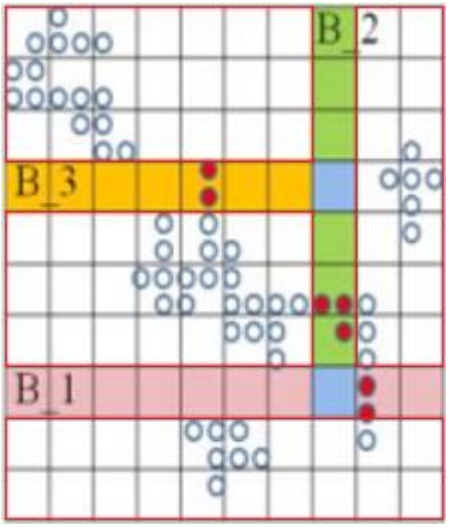

Figura 9 - Exemplo das partições e das regiões de fronteira (DAI; LIN, 2012).

Depois da divisão das regiões, as partições são enviadas para o sistema de arquivos distribuídos do Hadoop (HDFS) para se executar os métodos DBSCAN *Map* e DBSCAN

Reduce. O DBSCAN *Map* gera dois arquivos de saída, um com os agrupamentos locais e outro com os agrupamentos que envolvem os pontos de fronteira. O DBSCAN *Reduce* executa os pontos de fronteira e verifica se pontos com mesmo índice estão em agrupamentos diferentes.

Por conseguinte, ao final do DBSCAN *Reduce*, é necessário identificar se um *cluster* abrange mais de uma partição, por isso, os pontos de fronteira devem ser examinados e, caso necessário, os *clusters* são unificados. Por fim, os pontos são remarcados e salvos na base de dados.

2.11 Outros trabalhos no estado da arte

Nesta seção são apresentados outros trabalhos encontrados na literatura que utilizam o método de *MapReduce* para otimização de desempenho em algoritmos de prospecção de dados espaciais em bases volumosas.

2.11.1 MR-DBSCAN

Em 2011, foi proposto um algoritmo que se baseia no DBSCAN e utiliza a técnica de *MapReduce* para processamento paralelo, chamado MR-DBSCAN. Apesar de também ser baseado no DBSCAN, este algoritmo possui uma abordagem um pouco diferente do anterior. Este algoritmo é implementado com quatro etapas de *MapReduce* e também adota uma estratégia de particionamento dos dados. Os resultados revelaram que teve um aumento da velocidade de execução do DBSCAN (HE et al, 2011).

Já em 2013, os autores reformularam MR-DBSCAN e propuseram uma nova abordagem para o algoritmo. Na nova versão, todos os subprocessos críticos são paralelizados. Assim, não possui gargalo de desempenho causado pelo processamento sequencial. Também foi proposto um método de particionamento dos dados baseado no custo computacional (HE et al, 2014).

Para validar o algoritmo foram realizados testes em bases de dados reais de até 1,2 bilhão de pontos. Nos resultados dos experimentos o MR-DBSCAN se mostrou eficiente e escalável no processamento de conjuntos *Big Data* (HE et al, 2014).

2.11.2 DBCURE-MR

Em 2014, foi proposto um novo algoritmo de agrupamento espacial baseado em densidade robusto para encontrar os clusters com densidades variadas, chamado DBCURE. Sua principal vantagem é que desde sua concepção foi estrategicamente planejado e projetado para ser de fácil adequação ao *MapReduce*. Dessa maneira, o DBCURE-MR foi desenvolvido logo em seguida, que é a implementação do DBCURE com a técnica de *MapReduce*. Dessa maneira, enquanto os algoritmos tradicionais encontram um agrupamento por vez, o DBCURE-MR tem a capacidade de encontrar vários agrupamentos em paralelo. Por meio dos resultados encontrados, foi possível identificar que o DBCURE-MR localizou os agrupamentos de forma mais eficiente que DBCURE (KIM et al, 2014).

2.11.3 *Framework* de prospecção de dados espaciais baseado em *MapReduce*

Em 2013, foi proposto um *framework* para prospecção de dados espaciais, cujo objetivo é descobrir conjuntos de eventos prevalentes co-localizados. Para o desenvolvimento deste trabalho, foi utilizada a plataforma *Hadoop* como implementação do *MapReduce* com a finalidade de lidar com dados espaciais em grande escala em ambientes de sistemas de arquivos distribuídos (YOO; BOULWARE, 2013).

Sua execução foi dividida em três fases, sendo que cada fase é um método distribuído em *MapReduce* e a saída de cada etapa é usada como entrada para a outra. Com isso, o método proposto apresenta uma melhor escalabilidade e desempenho em relação aos algoritmos tradicionais (YOO; BOULWARE, 2013).

Assim, a primeira etapa em *MapReduce* consiste na busca dos objetos vizinhos, onde recebem os dados espaciais como entrada e gera os pares dos vizinhos como saída. Esta saída, por sua vez, será a entrada para a etapa seguinte, onde serão geradas as transações de vizinhança e retornado o total de vizinhos por objeto. Por fim, a saída da etapa anterior é a entrada da etapa final, a qual localiza e retorna os conjuntos de eventos co-localizados predominantes como saída final, conforme é exibido na arquitetura geral do *framework* apresentada na Figura 10 (YOO; BOULWARE, 2013).

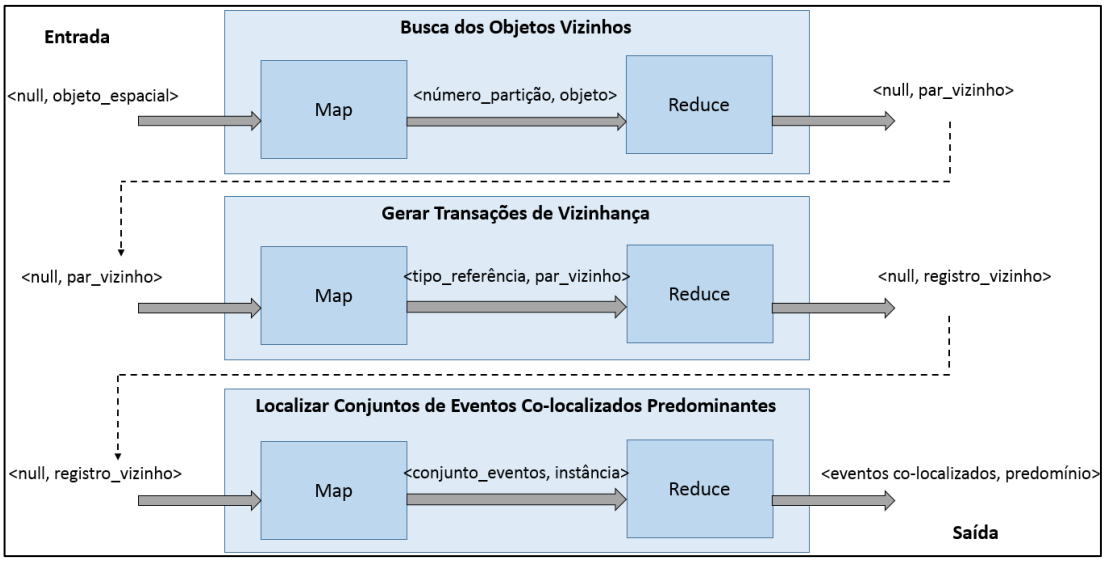

Figura 10 - Arquitetura do *Framework* (Adaptado de (YOO; BOULWARE, 2013)).

2.12 Considerações finais

Neste capítulo foram apresentados os principais conceitos que envolvem *Big Data* e sua análise, a complexidade de se trabalhar com dados espaciais e técnicas de análise de dados espaciais, principalmente a técnica de agrupamento espacial. Foram explicitados os desafios e problemas encontrados em aplicações de *Big Data*. Na área de processamento de grandes volumes de dados foi definido o método *MapReduce* e seu *framework* Hadoop.

Também foram apresentados os algoritmos DBSCAN, VDBSCAN, OVDBSCAN e DBSCAN-MR que são base para os algoritmos VDBSCAN-MR e OVDBSCAN-MR. Foi visto que todos os algoritmos são melhorias ou novas abordagens do DBSCAN, algoritmo consagrado na área de prospecção de dados espaciais.

Além disso, também foram descritos outros algoritmos e ferramentas da área de prospecção de dados espaciais com implementação em *MapReduce* encontrados na literatura.

Capítulo 3
Algoritmos de prospecção de dados espaciais para grandes conjuntos de dados

3.1 Considerações iniciais

Neste capítulo os dois algoritmos propostos, VDBSCAN-MR e OVDBSCAN-MR, são apresentados e contextualizados com cada algoritmo base descrito no capítulo anterior. Ambos são algoritmos de agrupamento espacial baseado em densidade, que se diferenciam dos algoritmos tradicionais, pois implementam a técnica de *MapReduce* a fim de possibilitar sua aplicação em grandes conjuntos de dados, sendo que o VDBSCAN-MR obteve um bom desempenho até para conjuntos com características de *Big Data*.

3.2 VDBSCAN-MR

O VDBSCAN-MR é um algoritmo da categoria de agrupamento de dados espaciais baseado em densidade que implementa o modelo de programação *MapReduce* sobre o *framework* Hadoop para permitir a análise de grandes bases de dados. O algoritmo incorpora técnicas dos algoritmos DBSCAN, VDBSCAN e DBSCAN-MR, citados anteriormente, e as aplica de maneira diferenciada a fim de explorar a infraestrutura disponibilizada em um *cluster* Hadoop.

Este algoritmo possui um único parâmetro de entrada que é o valor de *k*, ou seja, o número mínimo de pontos para se formar um agrupamento. Tal parâmetro tem origem no algoritmo VDBSCAN.

Em sua execução, o VDBSCAN-MR percorre duas etapas de processamento distribuído no *MapReduce* e as restantes são efetuadas sequencialmente na máquina mestre do *cluster* Hadoop. Além disso, antes de se iniciar o processamento distribuído, os dados precisam ser copiados para o sistema de arquivos distribuídos do Hadoop, o HDFS (*Hadoop Distributed File System*).

O fluxograma do algoritmo pode ser visualizado na Figura 11 e as principais etapas do algoritmo, destacadas na cor azul, são explicadas detalhadamente nas subseções seguintes.

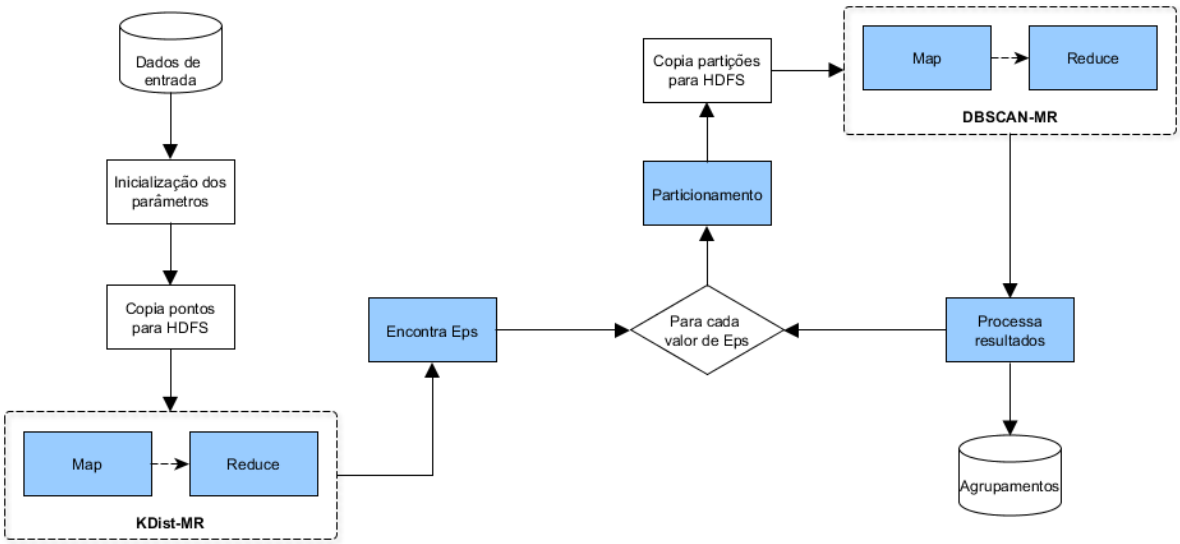

Figura 11 - Fluxograma do VDBSCAN-MR.

3.2.1 KDist-MR

Na etapa KDist-MR são calculadas as distâncias de cada ponto até todos os outros pontos do conjunto de dados e selecionado o valor da distância do ponto até seu k-ésimo vizinho para ser gerado o gráfico K-Dist, o qual será usado posteriormente para encontrar os valores de *Eps*.

Como exibido no fluxograma da Figura 11, esta etapa é realizada no *cluster* Hadoop e, por isso, contém as funções *Map* e *Reduce*. Antes de iniciar o processamento, um arquivo de texto com a lista dos pontos é enviado ao sistema de arquivos distribuído HDFS, para ser acessado pela aplicação. A partir daí a função *Map* faz a leitura dos pontos, percorre a lista dos pontos e cria os pares do ponto com seus vizinhos *(ponto, ponto_vizinho)*, que serão os pares chave/valor da saída intermediária. Posteriormente, a função *Reduce* é encarregada de

ler os pares intermediários, fazer o cálculo das distâncias euclidianas entre o ponto e os pontos vizinhos e adicioná-las em uma lista de distâncias, a qual é ordenada para selecionar a distância até o k-ésimo vizinho mais próximo, e gerar a saída chave/valor *(ponto, distância)*, cujos valores são os do gráfico K-Dist. O arquivo de saída também é salvo no sistema de arquivos distribuído. A seguir são apresentados os pseudocódigos das funções *Map* e *Reduce* do KDist-MR.

KDist-MR – Função Map

Entrada:

Pontos[] // Lista de pontos

Procedimento:

1. *Para cada ponto de Pontos[]*
2. *Para cada ponto_vizinho de Pontos[]*
3. *Cria os pares <ponto, ponto_vizinho>*

Saída:

Pares <ponto, ponto_vizinho> // Pares intermediários do MapReduce

KDist-MR – Função Reduce

Entrada:

Pares <ponto, ponto_vizinho> // Saídas da função Map

Procedimento:

1. *Para cada ponto_vizinho*
2. *Calcula a distância do ponto com o ponto_vizinho*
3. *Adiciona o valor na lista de distâncias*
4. *Ordena a lista de distâncias*
5. *Seleciona a distância do k-ésimo vizinho mais próximo*
6. *Cria os pares <ponto, distância>*

Saída:

Pares <ponto, distância> // Saída final do MapReduce

Após a execução do KDist-MR, o resultado de saída será salvo no HDFS e usado nas funções seguintes do VDBSCAN-MR.

3.2.2 Identificação dos valores de *Eps*

O objetivo deste método é encontrar os valores de *Eps* por meio do gráfico K-Dist, ou seja, os raios de buscas que serão usados para encontrar os agrupamentos. Este método é derivado do VDBSCAN e mantém a execução de forma sequencial. No entanto, possui a vantagem de utilizar o valor de saída do KDist-MR, uma vez que já foram calculadas as distâncias entre os pontos e selecionada a distância até o k-ésimo vizinho.

Para encontrar os valores de *Eps* é gerado um gráfico com as medidas das distâncias, retornadas pelo KDist-MR, até o k-ésimo vizinho mais próximo de cada ponto em ordem crescente. Então são identificados saltos no gráfico, os quais são as possíveis diferenças de densidades de distribuição dos pontos no espaço e, consequentemente, os valores escolhidos para *Eps*.

No algoritmo VDBSCAN tradicional, a identificação dos saltos é feita de maneira supervisionada, ou seja, o autor monta o gráfico e define visualmente os valores de *Eps*. No entanto, em determinadas bases de dados, principalmente as com grande volume de dados, a identificação dos saltos no gráfico de maneira visual não é muito precisa. Assim, no algoritmo proposto, é proposta uma abordagem automática para identificação dos saltos no gráfico K-Dist do VDBSCAN, ou seja, é realizado um cálculo que exclui a análise visual do gráfico.

Esta nova abordagem encontra pequenos saltos e mais valores de *Eps* para a busca dos agrupamentos, os quais permitem um melhor resultado. A técnica elaborada consiste em percorrer todos os valores do gráfico das distâncias e fazer um somatório das diferenças entre cada valor e seu anterior. Dessa forma, é determinada a média dos saltos dos valores do gráfico, a qual foi denominada de salto médio. Por fim, os valores de *Eps* são encontrados pelas mudanças bruscas no gráfico, as quais foram definidas empiricamente como a diferença das distâncias que são duas vezes maiores que o salto médio. A nova abordagem de obtenção dos *Eps* é apresentada no pseudocódigo colocado no quadro seguinte.

Encontrar Valores de Eps

Entrada:

Pares <ponto, distância> // Saída do KDist-MR

Procedimento:

1. *Ordenar distâncias e gerar o vetor de distâncias kdist[]*
2. *Para i = 0 até i = kdist.size()*
3. *somatório += kdist[i] – kdist[i-1]*
4. *contador++*
5. *salto_médio = somatório/contador*
6. *Para i = 0 até i = kdist.size()*
7. *Se kdist[i] – kdist[i-1] > 2 * salto_médio*
8. *Eps[] = kdist[i]*

Saída:

Eps[] // Lista dos valores de Eps encontrados

Após determinar os valores de *Eps*, o algoritmo executará a etapa do DBSCAN para cada valor encontrado a fim de gerar os agrupamentos com densidades variadas. No entanto, devido a inclusão do *MapReduce*, a etapa do DBSCAN foi modificada no VDBSCAN-MR e então o algoritmo seguirá para a fase de particionamento.

3.2.3 Estratégia de Particionamento

No algoritmo VDBSCAN tradicional, o DBSCAN é executado sequencialmente para cada valor de *Eps*. Porém, ao se tratar de *Big Data*, o DBSCAN terá naturalmente uma execução mais lenta à medida em que os dados crescem em volume, dado que no pior caso atinge uma complexidade de $O(n^2)$. Assim, certamente implicaria em uma dificuldade importante na capacidade de finalizar a execução diante de tais crescimentos na quantidade de dados (TARIQ; ASGHAR; SAJID, 2010).

À vista disso, nesta abordagem deverá ser executado de forma distribuída pelo modelo de programação *MapReduce*. No entanto, o DBSCAN não é facilmente adaptável às técnicas de *MapReduce*, ou seja, não é possível obter uma função *Map* e uma função *Reduce* que consigam mapear, reduzir e gerar o resultado final do algoritmo. Consequentemente, para atingir o desempenho esperado sem alterar os resultados, foi proposta uma estratégia de

particionamento para processar os dados, isto é, o espaço de busca será fracionado em regiões menores e a processamento dos objetos ocorrerá separadamente em cada região.

O trabalho proposto se baseia no algoritmo DBSCAN-MR e sua estratégia de particionamento, chamada de PRBP, o qual possui dois parâmetros de entrada β e θ, onde β é o número inteiro que indica a quantidade mínima de pontos para que uma partição seja dividida em duas novas partições e θ é um número real entre 0 e 0,5 usado para realizar o balanceamento de carga entre as partições. Na abordagem implementada, é utilizado apenas o parâmetro θ, visto que o algoritmo também se baseia no VDBSCAN e a cada execução do DBSCAN o número de pontos pode ser diminuído, o que atrapalha o uso do β. Assim, o valor de β foi definido como 30% do total de pontos da região processada.

Tal valor não poderia ser alto, pois encontraria poucas partições com muitos pontos e não teria um ganho computacional na aplicação do DBSCAN-MR, e também não poderia ser baixo, pois encontraria muitas partições com pouquíssimos pontos e aumentaria o processamento final dos resultados. Dessa maneira, o valor de 30% foi considerado bom para o parâmetro.

Ao iniciar o particionamento, primeiramente é criada uma partição que ocupa toda a região buscada. Depois verifica se esta deve ser dividida em novas partições de acordo com o valor de β, caso satisfaça a condição necessária, a partição é fatiada em vários retângulos horizontais e verticais, onde o lado desse retângulo corresponde a duas vezes o valor de *Eps* buscado. Para cada fatia criada é contabilizada a quantidade de objetos contidos na região. Na Figura 12 é apresentado um exemplo de um espaço fatiado, o total de pontos presentes em cada fatia horizontal e vertical, os quais são representados pelas circunferências, e a fatia com menor número de pontos que será candidata a ser região de fronteira, a qual está em destaque na cor amarela.

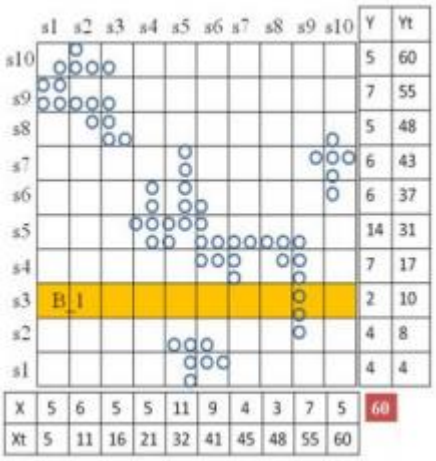

Figura 12 - Exemplo do fatiamento da partição (DAI; LIN, 2012).

Depois de o espaço ser fatiado e contado o total de pontos, é verificado para cada fatia se satisfaz o balanceamento de carga. Por fim, a fatia que possuir menos pontos e respeitar o balanceamento de carga é a escolhida para dividir o espaço. Assim, a partição inicial é excluída e dá origem a duas novas partições de modo que a fatia fique presente em ambas, onde passa a ser chamada de região de fronteira. Os pontos contidos na região de fronteira são marcados para poderem ser identificados na execução do DBSCAN-MR. Na Figura 13 são exibidas as duas novas partições divididas pela fatia de cor amarela, que foi detectada como região de fronteira e, por isso, está presente em ambas partições.

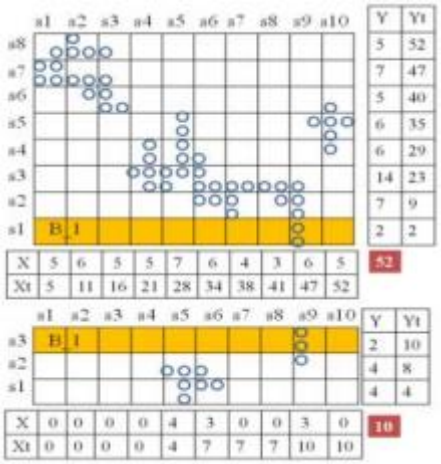

Figura 13 - Fatia encontrada e novas partições (DAI; LIN, 2012).

Posteriormente, as novas partições criadas também devem ser verificadas, pois podem ser divididas em partições ainda menores, isto é, o processo é realizado até que todas as partições sejam indivisíveis em conformidade com os parâmetros β e θ. Na Figura 14 pode ser observado um exemplo das várias partições, que estão representadas por diferentes cores, as

regiões de fronteira, que estão em branco, e os objetos presentes em cada partição e região de fronteira, que estão representados por diferentes símbolos, sendo que os pontos de fronteira são as circunferências pintadas de preto.

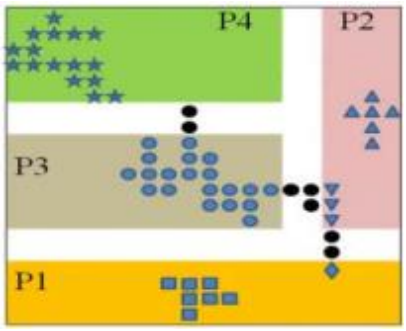

Figura 14 - Exemplo das partições encontradas (DAI; LIN, 2012).

A seguir são dados os pseudocódigos de cada etapa de partição. O método recebe a lista de pontos, define o parâmetro β, identifica a região que possui todos os pontos, a qual é a partição inicial, e a inclui na lista de partições. Depois, para cada partição é chamado o método responsável por verificar se a partição necessita ser dividida em partições menores ou não.

PRBP Modificado

Entrada:

Pontos[] // Lista de pontos

Procedimento:

1. *β = 0.3 * total de Pontos[]*
2. *partição = toda região dos pontos*
3. *Adiciona partição na lista de partições*
4. *Para cada partição*
5. *VerificaParticionamento()*

Saída:

Partições

O método *VerificaParticionamento()* recebe como entrada os parâmetros β, θ e a partição a ser verificada. Seu processo inicia com a definição do número mínimo de pontos, que no caso é o total de pontos presentes na partição verificada. Depois, é verificado se a partição

possui um número de pontos maior que o mínimo necessário para a partição ser dividida, dado por β. Caso satisfaça esta condição, a partição é dividida conforme mostrado na Figura 12. Após isso, é verificado se o número de fatias em cada eixo é maior ou igual a 3, pois caso for menor não é possível realizar a partição. Se sim, percorre cada eixo da partição e cada fatia da dimensão e verifica se o total de pontos da partição multiplicado pelo valor do balanceamento de carga, θ, é menor que o total de pontos da fatia. Também é verificado se o total de pontos da partição multiplicado pelo complemento do valor de balanceamento de carga, 1 menos θ, é maior que o total de pontos da fatia. Se todas essas condições forem atendidas, ainda é verificado se o total de pontos da fatia é menor que o número mínimo de pontos e, caso seja, esta fatia é selecionada para dividir a partição e ser região de fronteira. Por fim, após percorrer todas as fatias e identificar a fatia que melhor atende aos requisitos para a divisão, é chamado o método *DividePartição()*. A seguir é detalhado o pseudocódigo do método *VerificaParticionamento()*.

Verifica Particionamento

Entradas:

β // número mínimo de pontos para que a partição seja particionada

θ // valor usado no balanceamento de carga

Partição

Procedimento:

```
1. min_pontos = Partição.TotalPontos
2. Se Partição.TotalPontos > β
3.     Fatia a Partição
4.     Se número de fatias em cada dimensão >= 3
5.         Para cada dimensão da Partição
6.             Para cada fatia da dimensão
7.                 Se Partição.TotalPontos * θ <
fatia.total_pontos e Partição.TotalPontos * (1 – θ) > fatia.total_pontos
8.                     Se fatia.total_pontos < min_pontos
9.                         Seleciona fatia para divisão
10.                        min_pontos = fatia.total_pontos
11.        Se encontrou a melhor fatia
12.            DividePartição()
```

O método *DividePartição()* reproduz computacionalmente o processo realizado na Figura 13. Para isto, ele recebe como entrada a partição inicial e a fatia selecionada para dividir a partição. Assim, cria-se duas novas partições, onde a fatia selecionada fica presente nas duas regiões. Depois marca os pontos da fatia como pontos de fronteira, remove a partição inicial e adiciona as novas partições da lista de partições. Tal método é detalhado a seguir.

Divide Partição

Entradas:

Partição

Fatia selecionada para dividir a partição

Procedimento:

1. *Cria as duas partições a partir da fatia selecionada*
2. *Marca os pontos da fatia como pontos de fronteira*
3. *Remove a Partição da lista de partições*
4. *Adiciona as novas partições na lista*

Ao finalizar o processo de particionamento, os pontos pertencentes a cada partição são colocados em arquivos separados e copiados para o HDFS, onde serão acessados pelo DBSCAN-MR.

3.2.4 DBSCAN-MR

Nesta etapa é onde se executa o algoritmo DBSCAN para cada partição encontrada. Como no particionamento, este método também é derivado do DBSCAN-MR, porém com algumas modificações, já que será executado várias vezes durante o processamento do algoritmo.

Conforme já descrito, esta etapa é implementada no modelo *MapReduce* e tem como principal objetivo encontrar os agrupamentos dentro de cada partição. Sendo assim, a função *Map* recebe cada partição com seus pontos e executa o DBSCAN tradicional para encontrar os agrupamentos internos. Após todos os pontos da partição terem sido classificados como centro, borda ou ruído, cada ponto é inserido num novo arquivo de texto salvo no HDFS que contém o ponto, sua classificação e o agrupamento associado. Por fim, os pontos da região de

fronteira são identificados e colocados no arquivo intermediário que será lido pela função *Reduce*. O pseudocódigo da função *Map* é apresentado a seguir.

DBSCAN-MR – Função *Map*

Entradas:

Pontos[] // Lista de pontos da partição

Eps // valor da busca

Procedimento:

1. *Executa o DBSCAN*
2. *Para cada ponto de Pontos[]*
3. *Se o ponto for de fronteira*
4. *Cria os pares <id_ponto, Eps+id_particao+id_cluster+ classificação>*
5. *Adiciona o ponto ao arquivo de agrupamentos*
6. *Salva o arquivo no HDFS*

Saídas:

Pares dos pontos de fronteiras com seus agrupamentos e marcações - <id_ponto, Eps+id_particao+id_cluster + classificação>

Arquivo de agrupamentos da partição

Ao finalizar a execução da função *Map*, os pontos da região de fronteira são enviados para a função *Reduce*, onde serão identificados os pontos que fazem parte de agrupamentos em diferentes partições, os quais serão usados para identificar os *clusters* que serão fundidos na próxima etapa.

Para isso, é verificada a classificação de cada ponto de fronteira e, se o ponto estiver marcado como centro, o seu agrupamento é um possível candidato a se unir com outros agrupamentos. Além disso, se o agrupamento não estiver na lista de agrupamentos do ponto, deve ser adicionado na lista. No final, a saída gerada é composta pelo ponto, a lista de agrupamentos e a informação se cada agrupamento deverá ser mesclado com outros. A seguir é apresentado o pseudocódigo da função *Reduce*.

DBSCAN-MR – Função *Reduce*

Entrada:

Saídas da função Map // lista de pontos de fronteira com seus respectivos agrupamentos e marcações

Procedimento:

1. *mesclar_cluster = false*
2. *Para cada ponto*
3. *Se estiver marcado como centro*
4. *mesclar_ cluster = true*
5. *Se o id_cluster não estiver na lista de clusters*
6. *Adiciona o cluster na lista*
7. *Cria pares de saída <id_ponto, lista de cluster + mesclar_cluster>*

Saída:

Pares <id_ponto, lista + mesclar_cluster>

Ao final da função *Reduce*, os resultados são salvos no HDFS e serão processados pelo método seguinte, juntamente com os arquivos dos agrupamentos gerados pela função *Map*.

3.2.5 Processamento dos Resultados Obtidos

Por fim, é realizado o processamento dos agrupamentos encontrados, cujo objetivo é identificar os agrupamentos que abrangem múltiplas partições e remarcar seus pontos. Para tanto, é usada a lista de combinação de agrupamentos dos pontos de fronteira, onde cada combinação indica os agrupamentos que serão mesclados.

Inicialmente, o método faz a leitura dos dados de saída do DBSCAN-MR. Então, para cada ponto de fronteira é verificado se o agrupamento que ele está contido precisa ser mesclado com outros, isto é, *mescla_cluster* igual a *true*. É importante recordar que o ponto só será marcado como *true* se ele for centro de algum agrupamento. Caso seja necessário mesclar, adiciona os agrupamentos do ponto de fronteira na lista final de agrupamentos. Depois, percorre cada agrupamento final de cada agrupamento do ponto de fronteira, verifica se o agrupamento do ponto de fronteira pertence a um agrupamento final e, então, mescla as listas. Após isso, é comparado cada agrupamento final com todos os outros e verificado se é preciso mesclar mais agrupamentos finais. Por fim, são carregados os pontos agrupados do

arquivo de saída do DBSCAN-MR Map e verificado se os pontos agrupados pertencem a algum ponto final, se sim, eles são remarcados.

O pseudocódigo deste método pode ser visualizado a seguir.

Unir Clusters

Entrada:

Saídas do DBSCAN-MR

Procedimento:

```
1. Para cada ponto de fronteira
2.     Se mesclar_cluster = true
3.         Adiciona os clusters do ponto na lista cluster_final
4.     Para cada cluster do ponto de fronteira
5.         Para cada cluster_final
6.             Se cluster ⊂ cluster_final
7.                 Mescla as listas
8.     Para i = 0; i <= cluster_final.size()
9.         Para j = i+1; j <= cluster_final.size()
10.            Se cluster de cluster_final[j] ⊂ cluster_final[i]
11.                Mescla cluster_final
12. Carrega pontos agrupados do arquivo de saída do DBSCAN-MR Map
13. Para cada cluster de cluster_final[]
14.    Para cada ponto de pontos_agrupados[]
15.        Se o ponto ⊂ cluster_final
16.            Altera o id_cluster do ponto
```

Saída:

Pontos agrupados

Finalmente, após os agrupamentos terem sido mesclados, o algoritmo salva-os na base de dados e as etapas de particionamento, DBSCAN-MR e processamento dos resultados são executadas novamente para o próximo valor de *Eps* com os pontos que ainda não possuem nenhum agrupamento.

3.3 OVDBSCAN-MR

Analogamente ao VDBSCAN-MR, o OVDBSCAN-MR também é um algoritmo da categoria de agrupamento de dados espaciais baseado em densidade que utiliza em sua implementação o modelo *MapReduce* com o *framework* Hadoop para permitir a análise em grandes volumes de dados.

O algoritmo é derivado dos algoritmos DBSCAN, VDBSCAN, OVDBSCAN e DBSCAN-MR, descritos no capítulo anterior, e os implementa de uma forma que permite distribuir o processamento na infraestrutura do Hadoop.

No algoritmo proposto, o parâmetro de entrada é o domínio inicial para os possíveis valores de *k*, o qual é herdado do algoritmo OVDBSCAN. Este será o domínio de busca para encontrar o melhor valor de *k* para a busca dos agrupamentos.

O OVDBSCAN-MR, como no VDBSCAN-MR, também possui duas etapas de processamento distribuído no *MapReduce* e as outras executadas sequencialmente. No entanto, ele possui algumas diferenças, uma vez que ele possui uma etapa a mais na hora de processar, que é a etapa de encontrar o melhor valor para *k* e alterações em outras etapas.

Na Figura 15 é mostrado o fluxograma do algoritmo, onde as etapas destacadas em azul são detalhadas a seguir, uma vez que as etapas pintadas de cinza são iguais às do VDBSCAN-MR.

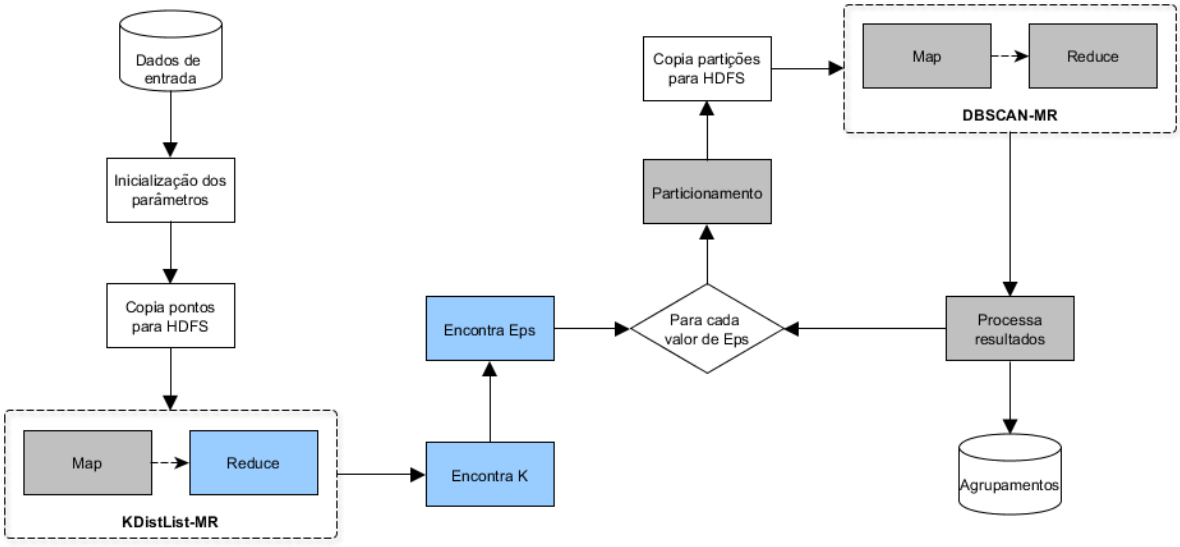

Figura 15 - Fluxograma do OVDBSCAN-MR.

3.3.1 KDistList-MR

A etapa KDistList-MR é parecida com a etapa KDist-MR do VDBSCAN-MR, mas a saída não contém apenas a distância do k-ésimo vizinho e sim, as distâncias de todos os vizinhos dentro do domínio de possíveis valores de *k*.

Então, esta etapa é responsável por calcular as distâncias de cada ponto até todos os outros pontos do conjunto de dados e gerar uma lista dos valores das distâncias até cada ponto, que será retornada para a aplicação sequencial encontrar o melhor valor de *k* e posteriormente os valores de *Eps*.

Como descrito anteriormente, este processo é distribuído nas funções de *Map* e *Reduce* do *cluster* Hadoop e, antes de iniciá-lo, um arquivo com a lista dos pontos deve ser enviado ao HDFS. A função *Map* lê os pontos e cria os pares intermediários de chave/valor, que são *(ponto, ponto_vizinho)*, e é idêntica à função *Map* do VDBSCAN-MR. A função *Reduce* lê os dados intermediários, calcula a distância entre o *ponto* e o *ponto_vizinho* e adiciona este valor na lista de distâncias. Depois, a lista de distâncias é ordenada e é gerada a saída chave/valor, que são *(ponto, distância [])*. A seguir é apresentado o pseudocódigo da função *Reduce* do KDistList-MR, uma vez que a função *Map* já foi apresentada anteriormente.

KDistList-MR – Função Reduce

Entrada:

Pares <ponto, ponto_vizinho> // Saídas da função Map

Procedimento:

1. *Para cada ponto_vizinho*
2. *Calcula a distância do Ponto com o ponto_vizinho*
3. *Adiciona o valor na lista de distâncias*
4. *Ordena a lista de distâncias*
5. *Cria os pares <ponto, distâncias[]>*

Saídas:

Pares <ponto, distâncias[]>

Após a execução do KDistList-MR, o resultado de saída também é salvo no HDFS e usado nas funções seguintes do OVDBSCAN-MR.

3.3.2 Encontra Valor de *k*

Este método tem como objetivo encontrar o valor ideal de *k* que será usado para escolha dos *Eps*. Esta etapa é derivada do algoritmo OVDBSCAN, que em sua abordagem sequencial calcula as distâncias dos pontos aos seus vizinhos, porém para a otimização do algoritmo este cálculo foi realizado anteriormente pelo KDistList-MR. Assim, esta função é responsável por ler os arquivos de saída no HDFS para depois iniciar a busca pelo melhor valor de *k*.

Além da leitura do resultado da fase anterior, esta função recebe o parâmetro do algoritmo, que é o domínio dos possíveis valores de *k*. A partir daí o algoritmo executa sequencialmente os mesmos passos do OVDBSCAN, descrito na seção 2.10.3, ou seja, para cada valor de *k* dentro do domínio definido é percorrido todos os pontos e calcula-se as taxas de variação, Δd_k, do K-Dist. Após isso, é montado o gráfico com os valores de Δd_k e definido o limiar de busca. Por fim, encontra-se o primeiro valor de Δd_k acima do limiar e seleciona o valor de *k*. Este processo está detalhado melhor no pseudocódigo a seguir.

Encontrar K

Entrada:

min_k // Menor valor do domínio de k

max_k // Maior valor do domínio de k

Procedimento:

1. *Leia saída o KDistList-MR*
2. *Para i = min_k até i = max_k*
3. *Para cada ponto*
4. Δd_k *+= ponto.distancia[i+1] – ponto.distancia[i]*
5. *Monta o gráfico com os valores de* Δd_k
6. *Define o limiar*
7. *Encontra o valor de k*

Saídas:

Valor de k

Encontrado o valor de *k*, este será usado para a busca dos valores de *Eps* no método seguinte.

3.3.3 Encontra *Eps*

Assim como no VDBSCAN-MR, este método do OVDBSCAN-MR também tem por objetivo encontrar os valores de *Eps* que serão usados nas buscas de agrupamentos. No entanto, a diferença deste para o do outro algoritmo é que neste método é preciso selecionar o gráfico K-Dist do valor *k* encontrado anteriormente para depois encontrar os valores de *Eps*. A partir daí o método e o pseudocódigo são os mesmos da subseção 3.2.2.

3.4 Qualidade dos resultados

Um ponto importante na prospecção de dados espaciais é a qualidade dos resultados, que neste caso, são os agrupamentos formados pelo algoritmo. Tendo isso em vista, também foi implementado um método para verificar a qualidade dos agrupamentos encontrados.

Dos vários métodos existentes na literatura, um dos mais utilizados é o coeficiente de silhueta que é um método que mede a qualidade dos agrupamentos em relação a todos os agrupamentos encontrados, ou seja, ele não utiliza medida externa aos dados. Para isso, é calculada a média das distâncias de um objeto para os demais objetos pertencentes ao seu *cluster* e a distância média mínima de um objeto para todos os outros *clusters* encontrados. A partir desses dois cálculos é indicado o coeficiente de silhueta (HAN; KAMBER, 2011).

Antes de apresentar as fórmulas, é necessário considerar que um objeto (o) pertencente a um agrupamento *Ci*, onde $Ci \in C = \{C1, C2, C3, Ck\}$. Assim, o cálculo da média das distâncias de *(o)* para os outros objetos de *Ci* é dado pela equação (1) e é nomeado de *a(o)*.

$$\boldsymbol{a(o)} = \frac{\sum_{o' \in Ci\ o \neq o'} \boldsymbol{dist(o,o')}}{|\boldsymbol{C_i}| - \mathbf{1}} \tag{1}$$

O cálculo da distância média mínima de *(o)* para todos os outros agrupamentos, nomeado de *b(o)*, é obtido pela equação (2).

$$\mathrm{b}(\boldsymbol{o}) = \boldsymbol{min}_{\boldsymbol{C_j}:1 \leq j \leq k, i \neq j} \left\{ \frac{\sum_{o' \in Cj} \boldsymbol{dist(o,o')}}{|\boldsymbol{C_j}|} \right\} \tag{2}$$

Por fim, o coeficiente de silhueta, chamado de *s(o)*, é obtido pela equação (3).

$$\boldsymbol{s(o)} = \frac{\boldsymbol{b(o)} - \boldsymbol{a(o)}}{\boldsymbol{max}\{\boldsymbol{a(o)}, \boldsymbol{b(o)}\}} \tag{3}$$

Os valores finais do coeficiente de silhueta podem variar de 1 a -1. Se o agrupamento *Ci* tiver o valor *s(o)* negativo significa que o agrupamento é similar aos demais agrupamentos, enquanto que se o valor de *s(o)* for próximo de 1 indica que o agrupamento está mais compacto e dissimilar entre os outros agrupamentos e, portanto, a qualidade é alta (HAN; KAMBER, 2011).

3.5 Considerações Finais

Neste capítulo foram apresentados e descritos os detalhes de implementação dos dois algoritmos propostos: o VDBSCAN-MR e o OVDBSCAN-MR. Além disso, para se obter um melhor entendimento, certas etapas foram ilustradas com imagens e fluxogramas e também foram feitos pseudocódigos para exemplificar cada método.

Foi possível perceber também que os dois algoritmos são bem parecidos e possuem muitas etapas em comum, pois tem como base algoritmos de origens semelhantes, como: DBSCAN, VDBSCAN, OVDBSCAN e DBSCAN-MR.

Por fim, foi descrito o método que será usado para medir a qualidade dos agrupamentos encontrados pelos algoritmos, o coeficiente de silhueta.

No Capítulo 4 são apresentadas as análises dos resultados obtidos com a execução do algoritmo implementado.

Capítulo 4
Experimentos e Resultados

4.1 Considerações Iniciais

Neste capítulo são apresentados os experimentos realizados com os algoritmos propostos, VDBSCAN-MR e OVDBSCAN-MR, com o objetivo de validar as estratégias implementadas. Foram executados testes comparativos entre os algoritmos desenvolvidos e os algoritmos tradicionais a fim de verificar a contribuição do presente trabalho.

Os testes têm o intuito de verificar três pontos importantes: a qualidade dos agrupamentos encontrados e o desempenho em relação aos algoritmos originais, além da escalabilidade da arquitetura *MapReduce*.

Para validar a qualidade dos resultados foi calculado o coeficiente de silhueta para os agrupamentos formados em ambos os algoritmos. Para o teste de desempenho e de escalabilidade foram medidos os tempos de execução para diferentes tamanhos de conjuntos de dados.

4.2 Testes de qualidade

Os experimentos a seguir são para comprovar a qualidade dos resultados obtidos pelos algoritmos propostos comparados com os algoritmos originais. Assim, o VDBSCAN-MR é comparado ao VDBSCAN, o OVDBSCAN-MR é comparado ao OVDBSCAN; e depois o VDBSCAN-MR e OVDBSCAN-MR são comparados entre si e com o DBSCAN-MR. Por

fim, foi realizado um experimento para comprovar a utilização da estratégia desenvolvida de encontrar os valores de *Eps* automaticamente ao invés de usar a forma supervisionada.

Para todos os experimentos foram utilizados um conjunto de dados de uma base sintética com 3.100 dados georreferenciados e pertencentes ao continente africano. As aplicações em *MapReduce* foram executadas num *cluster* Hadoop com 3 máquinas virtuais em um computador com processador Intel(R) Core (TM) i5-4200U CPU 1.6 GHz, sendo cada máquina composta por 1 vCPU e 2 GB memória RAM. As aplicações sequenciais foram executadas em uma máquina deste *cluster*.

4.2.1 Aplicação do VDBSCAN-MR

O experimento inicial tem por objetivo comprovar a validade do algoritmo VDBSCAN-MR. Então, primeiramente foi executado o VDBSCAN e analisados os resultados. Em seguida foi executado o VDBSCAN-MR, o qual deveria retornar os mesmos resultados que o primeiro.

O parâmetro *MinPts*, que é comum aos dois algoritmos, foi escolhido aleatoriamente como *MinPts* = 30, pois foi considerado uma quantidade mínima de pontos interessante para formar os agrupamentos. Já o outro parâmetro do VDBSCAN-MR, usado para a partição do espaço de busca, foi $\theta = 0{,}12$, pois depois de alguns testes era o valor em que se dividia melhor o espaço de busca.

O VDBSCAN-MR encontrou 116 valores de *Eps*, ou seja, as etapas de particionamento e DBSCAN-MR foram executadas 116 vezes, uma para cada valor de *Eps*, para encontrar os agrupamentos com densidades variadas. O resultado obtido em ambos os algoritmos foi de 33 agrupamentos que podem ser visualizados na Figura 16, o que comprova que o particionamento do espaço de busca e a distribuição das tarefas em *MapReduce* não influenciaram os resultados.

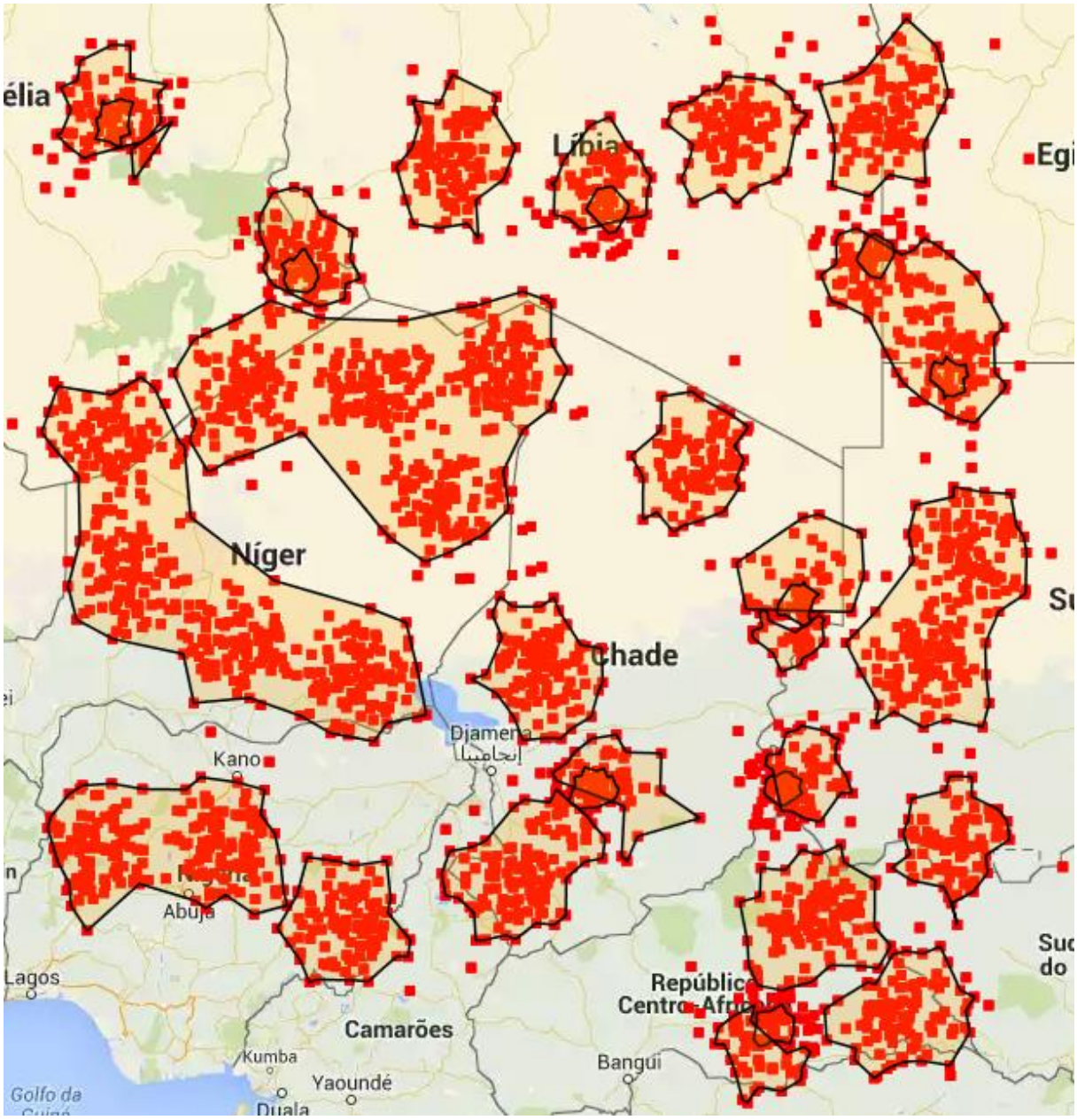

Figura 16 - *Clusters* encontrados na aplicação do VDBSCAN-MR e VDBSCAN.

4.2.2 Aplicação do OVDBSCAN-MR

Neste experimento, como no anterior, o objetivo é comprovar a validade do outro algoritmo proposto, o OVDBSCAN-MR. Assim, inicialmente foi executado o OVDBSCAN e em seguida o OVDBSCAN-MR, onde os agrupamentos encontrados em cada algoritmo deveriam ser os mesmos.

O domínio escolhido para os possíveis valores de k foi [50, 300] e o parâmetro θ para o OVDBSCAN-MR foi de 0,12.

Na execução, os algoritmos calcularam k = 72 e encontraram 107 valores de *Eps*. Os resultados obtidos em ambos os algoritmos foram os mesmos 31 agrupamentos que podem ser

vistos na Figura 17, o que mais uma vez comprova que as técnicas implementadas não alteraram os resultados.

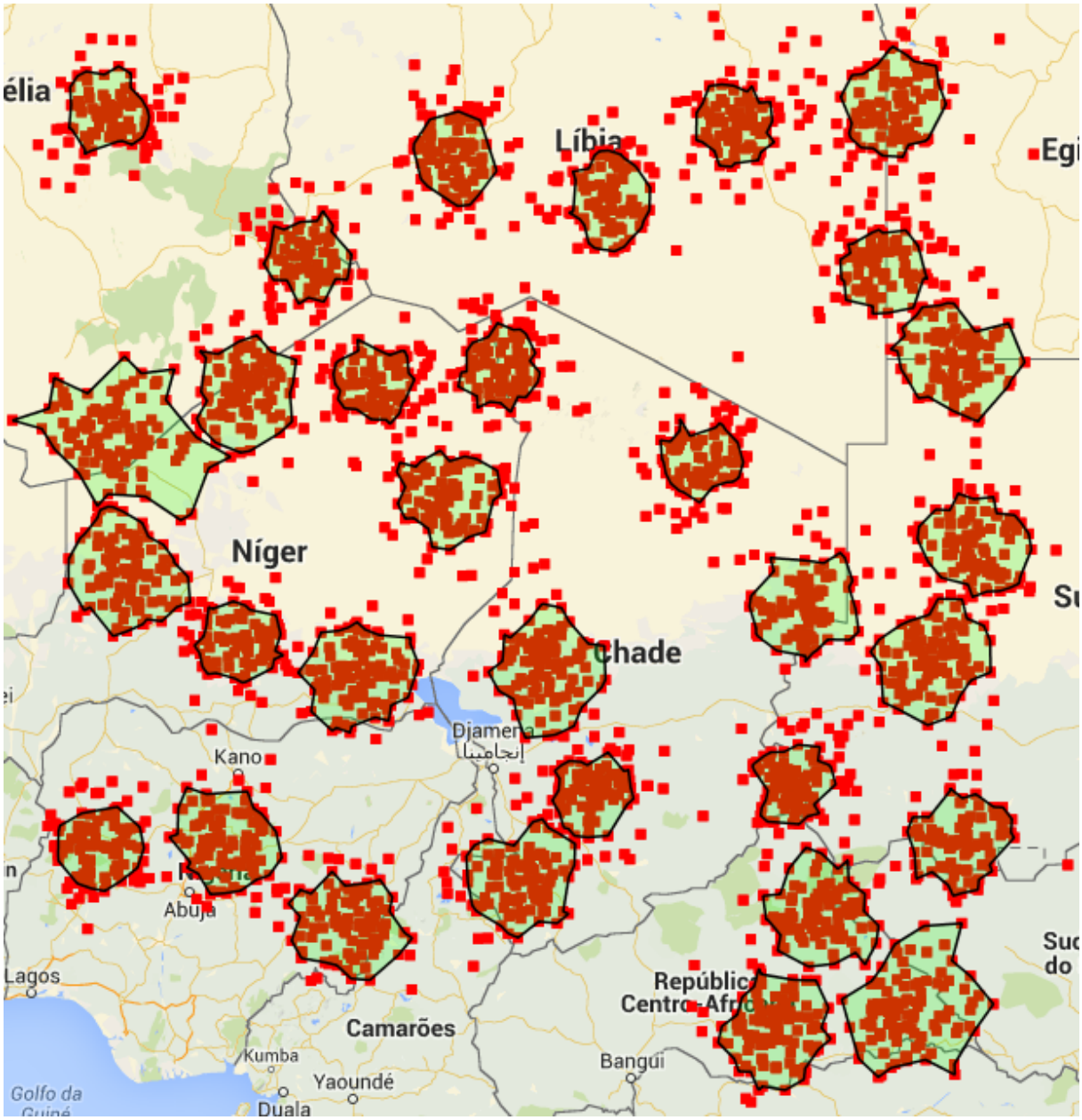

Figura 17 - *Clusters* encontrados na aplicação do OVDBSCAN-MR e OVDBSCAN.

4.2.3 Comparação com o DBSCAN-MR

Este experimento tem como objetivo comparar a qualidade dos agrupamentos encontrados pelos algoritmos VDBSCAN-MR e OVDBSCAN-MR com o DBSCAN-MR. Assim, os três algoritmos foram executados em *MapReduce* nas mesmas configurações do experimento anterior e na mesma base de dados sintética.

Os parâmetros utilizados e resultados encontrados para os algoritmos VDBSCAN-MR e

OVDBSCAN-MR foram os mesmos que nos experimentos anteriores exibidos na Figura 16 e na Figura 17.

Já para o DBSCAN-MR, os parâmetros escolhidos foram β = 300, θ = 0,12, *MinPts* = 30 e *Eps* = 1,5, pois nos testes retornaram melhores resultados. O resultado obtido foi 17 agrupamentos, os quais são vistos na Figura 18.

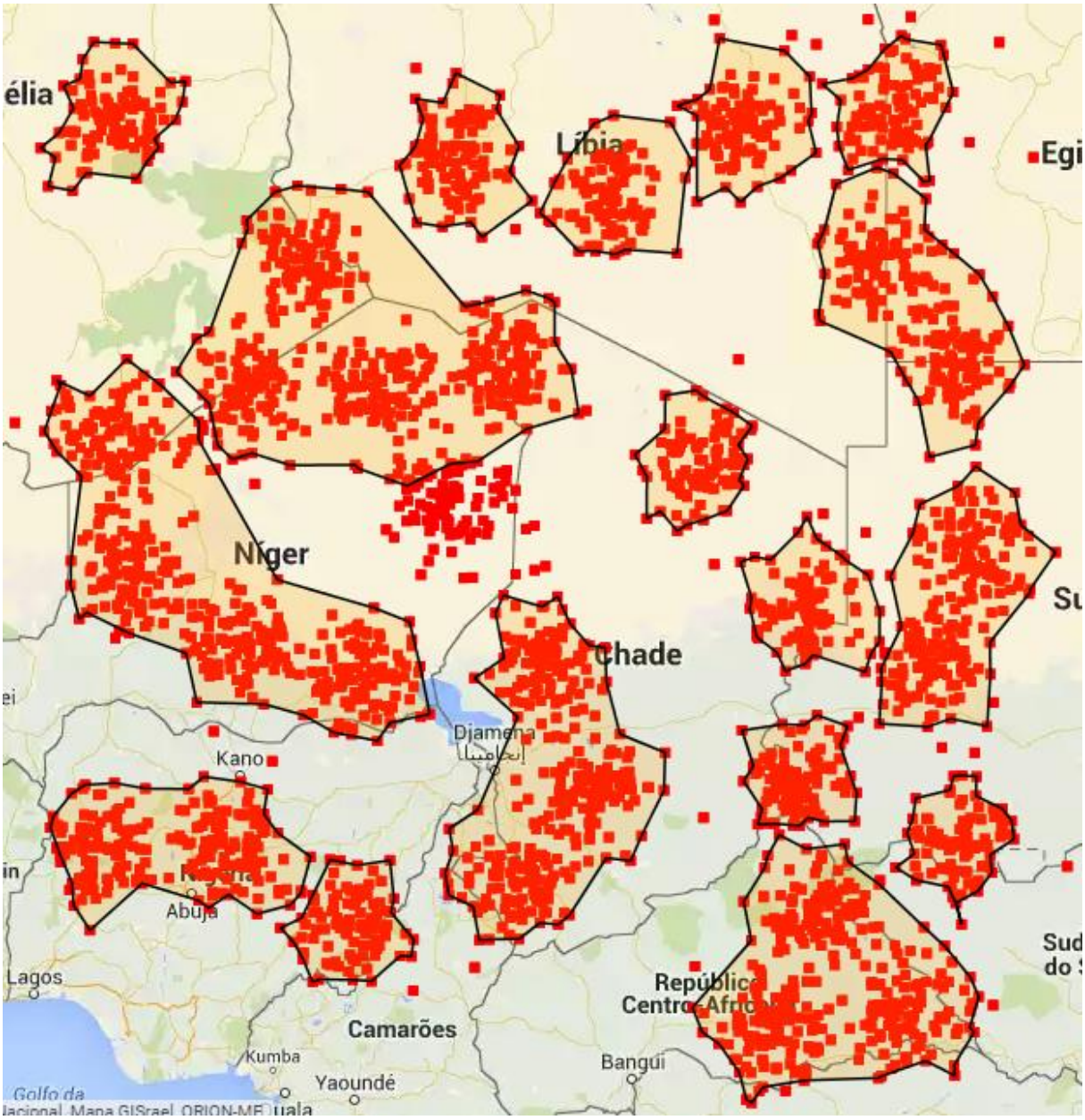

Figura 18 - Agrupamentos encontrados pelo DBSCAN-MR.

Ao comparar visualmente as imagens dos agrupamentos obtidos pelo DBSCAN-MR, VDBSCAN-MR e OVDBSCAN-MR, percebe-se que os resultados dos dois algoritmos propostos se mostraram de melhor qualidade que o do DBCAN-MR, visto que o último

encontrou agrupamentos grandes em regiões que poderiam formar vários agrupamentos menores e não agrupou alguns pontos próximos que ficaram marcados como ruídos, mesmo sendo selecionados os melhores parâmetros.

Já quando a comparação é feita entre os dois algoritmos propostos, o OVDBSCAN-MR apresenta um melhor resultado, pois em alguns casos o VDBSCAN-MR também encontrou agrupamentos grandes que poderiam ser mais compactos e outros muito menores que poderiam ter sido expandidos.

Então, para comprovar realmente a eficácia dos novos algoritmos, foi calculado para cada agrupamento das três abordagens o Coeficiente de Silhueta das distâncias entre os pontos. Os valores obtidos podem ser vistos na Tabela 1, na Tabela 2 e na Tabela 3.

Tabela 1 - Valores do Coeficiente de Silhueta dos agrupamentos do OVDBSCAN-MR.

ID do *Cluster*	Coeficiente de Silhueta da Distância
1	0,9124181300869537
2	0,9438373198848716
3	0,9025775247164832
4	0,883091411696028
5	0,8751280291876732
6	0,9114187601139957
7	0,8958671021496419
8	0,902670927751816
9	0,8865606758656569
10	0,9026337736952497
11	0,8990078859882311
12	0,8945878533167938
13	0,91343662532803
14	0,903425192766289
15	0,9243889618275344
16	0,8977375487905926
17	0,9081578287507441
18	0,9161466171203381
19	0,8590261236481292
20	0,8704802690204984
21	0,9111404795884588
22	0,8991728077176552
23	0,9004494593297302
24	0,9030307112558118
25	0,892405579151596
26	0,907122042264554

27	0,9287783520284801
28	0,9314311113423196
29	0,9217967270454108
30	0,8704151793700434
31	0,9203576908046208
Média	**0,9028612484388461**
Desvio-padrão	**0,0187962370450290**

Tabela 2 - Valores do Coeficiente de Silhueta dos agrupamentos do VDBSCAN-MR.

ID do *Cluster*	Coeficiente de Silhueta da Distância
1	0,8346677009912153
2	0,8213221674980195
3	0,6488466324145044
4	0,9604065743177604
5	0,949351251559421
6	0,8423307566214114
7	0,9015562906973821
8	0,6656102283537383
9	0,8931486622553404
10	0,8486059247571912
11	0,9595606447146345
12	0,8970638673728659
13	0,8986015907368847
14	0,9413127686019636
15	0,8971496308851948
16	0,901144581399542
17	0,7879010359119086
18	0,8509271003191022
19	0,896730953712057
20	0,8923884325322469
21	0,9184099049539911
22	0,9677013169580293
23	0,8435304233046299
24	0,9580359712983222
25	0,8558696075670018
26	0,9153313318662477
27	0,9416202219882379
28	0,8708737030849962
29	0,8833766000330189
30	0,9602508737805757

31	0,8986344711641377
32	0,952500961790277
33	0,9536542963937044
Média	**0,8851035296919866**
Desvio-padrão	**0,0748657907582087**

Tabela 3 - Valores do Coeficiente de Silhueta dos agrupamentos do DBSCAN-MR.

ID do *Cluster*	Coeficiente de Silhueta da Distância
1	0,7820909359335658
2	0,8685940865165808
3	0,8718214856723714
4	0,8948472737360337
5	0,8339160935600336
6	0,8915325796941135
7	0,882319127665212
8	0,7827790842767194
9	0,6378422270624385
10	0,7759843064486316
11	0,8805122505692001
12	0,6490401925791276
13	0,9202184534593318
14	0,9006183277361006
15	0,8393520022603687
16	0,8386507243433362
17	0,6672716128600215
Média	**0,8186700449631287**
Desvio-padrão	**0,0905658663655677**

Ao observar os valores dos coeficientes do OVDBSCAN-MR constata-se que não tiveram grande variação, o que comprova que os agrupamentos deste algoritmo foram os mais bem distribuídos no espaço. Já o VDBSCAN-MR teve alguns agrupamentos com pontos muito próximos e outros com pontos bem dispersos e, por isso, teve uma taxa de variação muito alta, o que não é muito interessante ao ser analisado o espaço como um todo. Por fim, o DBSCAN-MR teve alguns coeficientes muito baixos e apresentou a maior taxa de variação entre eles, ou seja, teve menos eficácia no processo de encontrar os *clusters*.

Pela análise dos coeficientes, das médias e dos desvios-padrão dos agrupamentos de ambos os algoritmos, conclui-se que o OVDBSCAN-MR foi o mais eficaz, pois teve a maior média e o menor desvio-padrão, o VDBSCAN-MR foi o segundo mais eficaz, pois teve uma

média boa, mas apresentou um desvio-padrão maior, e o DBSCAN-MR foi o menos eficaz, uma vez que apresentou a menor média e o maior desvio-padrão.

4.2.4 Estratégia de encontrar *Eps* automático

Como foi descrito no Capítulo 2, a estratégia de encontrar o *Eps* no VDBSCAN é feita pelo gráfico K-Dist de forma supervisionada. Desta maneira, este experimento visa comparar os resultados gerados pelo algoritmo que utiliza a estratégia tradicional, ou seja, analisar visualmente o gráfico, com a estratégia automática proposta no trabalho.

Esta estratégia foi adotada para os dois algoritmos propostos, mas neste experimento apenas o OVDBSCAN-MR foi usado, uma vez que a técnica é igual para os dois algoritmos propostos. Dessa maneira, foi executado duas vezes com os mesmos parâmetros, porém em uma usou a estratégia de encontrar os *Eps* automaticamente e em outra usou o método do gráfico.

Para a estratégia automática foram encontrados 107 valores de *Eps* que vão de 1,0837778 a 3,6484156. Os agrupamentos encontrados para esses valores de *Eps* foram os mesmos do primeiro experimento mostrado na Figura 17.

Para a aplicação supervisionada do algoritmo foi gerado o gráfico K-Dist exibido na Figura 19 e escolhido o valor de *Eps* visualmente pelo gráfico, ou seja, pelo método tradicional.

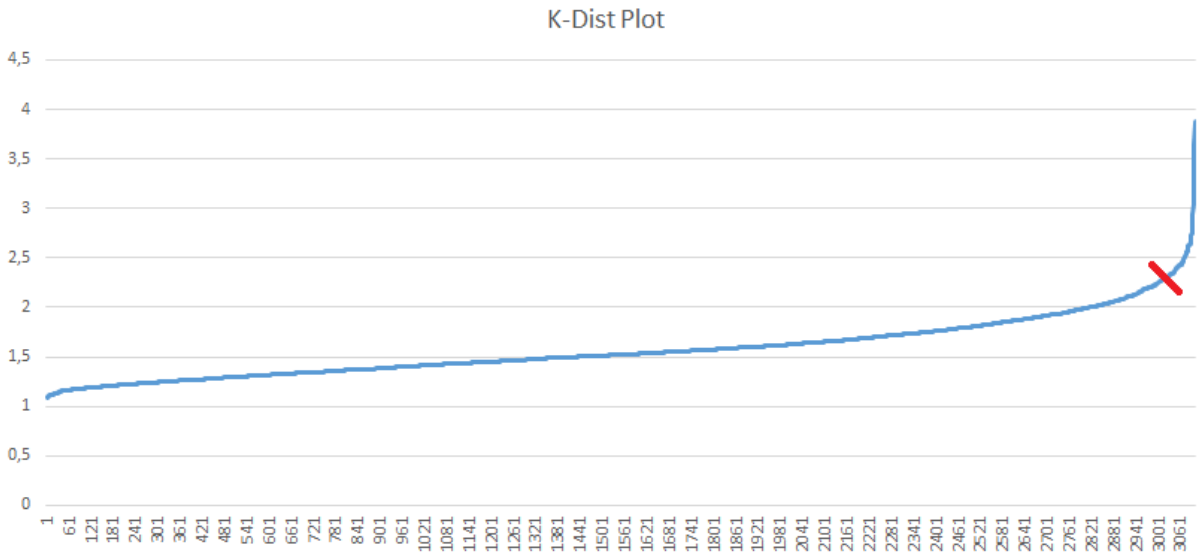

Figura 19 - K-Dist gerado pelo algoritmo supervisionado.

Ao analisar o gráfico é possível observar que, no conjunto de dados analisado, o gráfico não possui saltos e a tarefa de determinar os valores de *Eps* se torna difícil. Assim, o valor de

Eps determinado foi 2,3, pois é um ponto onde a região do gráfico apresenta o maior salto. A partir daí, os agrupamentos gerados podem ser vistos na Figura 20.

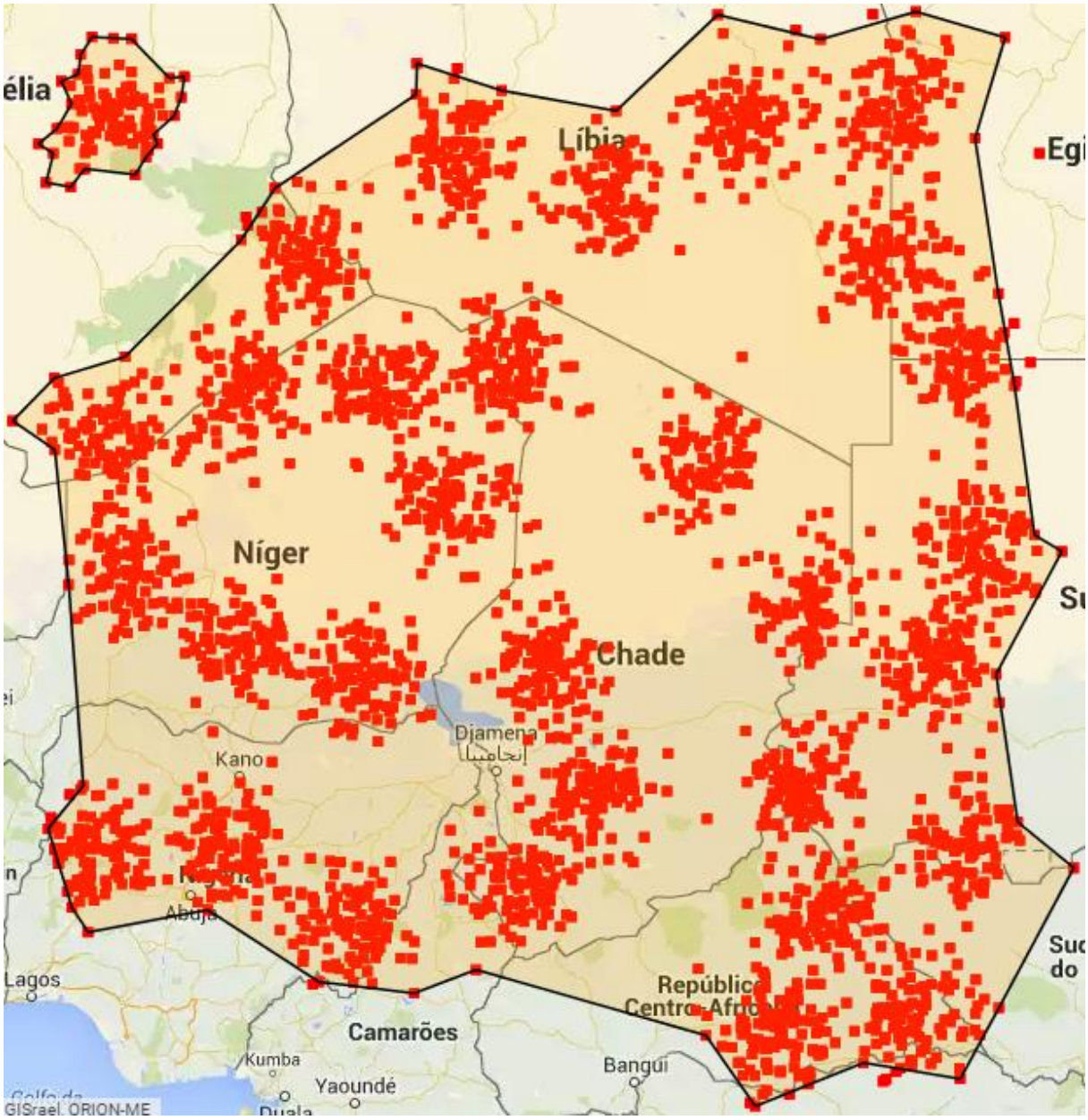

Figura 20 - Agrupamentos obtidos pela execução supervisionada.

Visualmente já é possível perceber que o resultado encontrado para este valor de *Eps* não foi dos melhores, uma vez que um agrupamento engloba a grande maioria dos dados e sua extensão ocupa quase toda região do conjunto de dados.

Ainda assim, para confirmar a má qualidade dos agrupamentos encontrados, foram calculados os valores dos coeficientes de silhueta dos agrupamentos e obtiveram-se os resultados apresentados na Tabela 4.

Tabela 4 - Valores do Coeficiente de Silhueta dos agrupamentos da execução supervisionada.

ID do *Cluster*	Coeficiente de Silhueta da Distância
1	0,13479874413476978
2	0,6822571609450336
Média	**0,4085279525399017**
Desvio-padrão	**0,3871115589441890**

Os coeficientes de silhueta apresentados na Tabela 4 são extremamente baixos, o coeficiente médio é bem menor em relação ao algoritmo que utiliza a abordagem automática e o desvio-padrão é bem alto. Portanto, comprova-se que a técnica implementada é útil para encontrar melhores valores de *Eps* e, por consequência, retornar agrupamentos de mais qualidade.

4.3 Teste de desempenho

Este experimento teve como propósito mostrar a eficiência dos algoritmos alvos deste trabalho. No entanto, o algoritmo OVDBSCAN-MR não teve um bom desempenho quando se aumentou o tamanho do conjunto de dados a serem analisados, pois na tarefa inicial de encontrar o valor ótimo para *k* ocorria estouro de memória devido ao fato de ser sequencial. Assim, as comparações foram feitas somente entre o VDBSCAN-MR e o VDBSCAN, ou seja, foram medidos os tempos de execução do algoritmo processado na arquitetura *MapReduce* e comparado com os tempos de execução do algoritmo sequencial.

A base de dados utilizada foi a *GEOnet Names Server* – GNS da Agência Nacional de Inteligência-Geoespacial – NGA, em inglês *National Geospatial-Intelligence Agency*, com os dados do dia 21 de setembro de 2015 (NGA, 2015).

A arquitetura usada para o processamento foi de 5 máquinas virtuais em um computador com processador AMD FX(tm)-6300 Six-Core, cada máquina com 1 vCPU e 3 GB memória RAM.

Para o experimento foram selecionadas quatro diferentes quantidades de pontos georreferenciados aleatórios do conjunto total de dados: 50 mil, 100 mil, 150 mil e 200 mil pontos. Foram escolhidas essas quantidades de dados para efetuar a comparação, pois a partir disso o VDBSCAN já não conseguia finalizar a execução em tempo hábil.

Então, foram executados os algoritmos VDBSCAN e VDBSCAN-MR para cada conjunto de dados e aferidos os tempos de execução. Na Tabela 5 estão indicados os tempos de execução em segundos.

Tabela 5 – Tempo de execução dos algoritmos (em segundos) para cada quantidade de pontos.

Total de Pontos	**50000**	**100000**	**150000**	**200000**
VDBSCAN	39,90	176,72	442,17	1003,37
VDBSCAN-MR	11,80	74,37	95,28	221,43

Ao comparar os valores da tabela é possível ver que o algoritmo proposto obteve um melhor desempenho em relação ao algoritmo sequencial. Para melhor visualização, na Figura 21 é mostrado o gráfico de comparação entre os tempos de execução para cada quantidade de pontos.

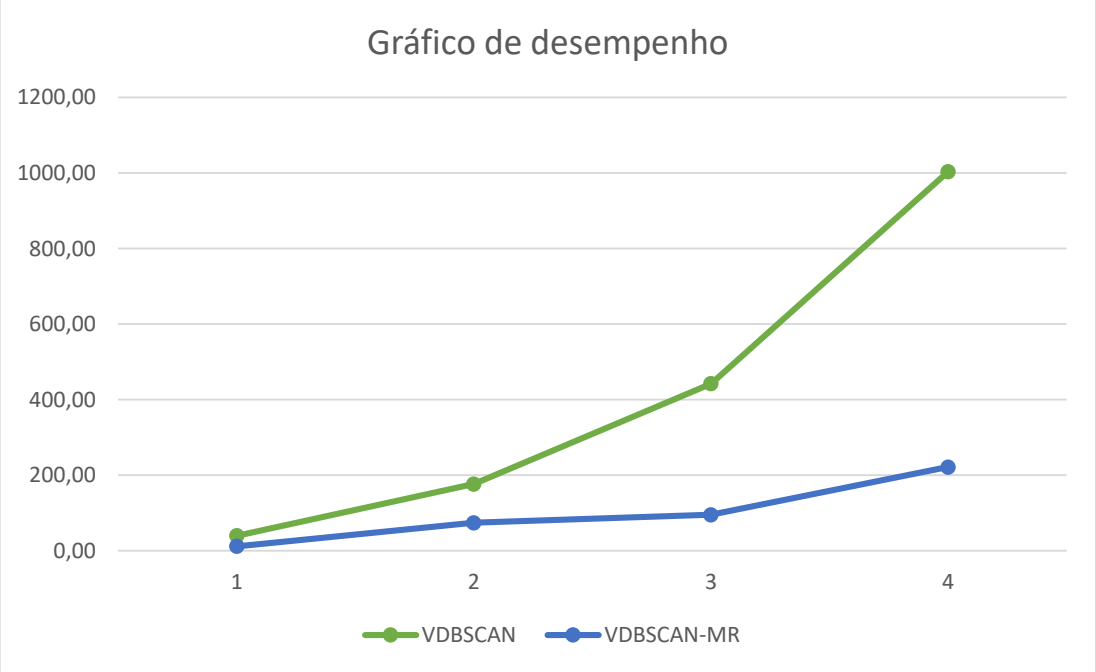

Figura 21 - Tempos de execução (em segundos) dos algoritmos VDBSCAN-MR e VDBSCAN.

4.4 Teste de escalabilidade

Além do teste de desempenho, foi realizado um teste de escalabilidade com o objetivo de comprovar que o VDBSCAN-MR suporta grandes quantidades de pontos. O OVDBSCAN-MR não foi testado para este caso pelo mesmo motivo do teste anterior.

Para este experimento, a base de dados utilizada foi a GNS da NGA (NGA, 2015) com um total de 1 milhão de pontos georreferenciados. A arquitetura usada foi a EMR – *Elastic MapReduce* da AWS - *Amazon Web Services* (AWS, 2016) com 3 instâncias (1 mestre e 2 escravos) da máquina m3.xlarge que possuem processadores Intel Xeon E5-2670 v2 (*Ivy Bridge*), com 4 núcleos, 15 GB de memória RAM e armazenamento em dois SSDs de 40 GB.

Como já foi dito no experimento anterior, a partir de 300 mil pontos o algoritmo sequencial passou a não finalizar em tempo hábil e, assim, não foi testado com essa quantidade de dados.

A execução do VDBSCAN-MR para 1 milhão de pontos encontrou 74 agrupamentos e durou cerca de 2751 minutos, que é mais de 45 horas, ou seja, quase dois dias, o que prova que o algoritmo suporta uma grande quantidade de dados quando se aplica as técnicas de *MapReduce* e particionamento. Além disso, é possível diminuir o tempo de execução com o aumento da quantidade de nós de processamento.

4.5 Considerações Finais

Neste capítulo foram apresentados alguns experimentos para validar os algoritmos VDBSCAN-MR e OVDBSCAN-MR, testes de desempenho e escalabilidade do VDBSCAN-MR, além de compará-los com os algoritmos sequenciais.

Inicialmente os novos algoritmos foram comparados aos algoritmos VDBSCAN e OVDBSCAN e foi constatado que não houve perda nos resultados com a inserção da etapa de particionamento e da técnica de *MapReduce*.

Depois, os resultados obtidos pelos algoritmos foram comparados aos resultados do DBSCAN-MR para mostrar que a qualidade dos resultados encontrados pelos novos algoritmos foi melhor. Para comprovar, foi calculado o Coeficiente de Silhueta para cada agrupamento, depois o coeficiente médio e o desvio padrão.

Os testes de desempenho e escalabilidade foi executado apenas para o VDBSCAN-MR, pois o OVDBSCAN-MR, apesar de apresentar ótima qualidade nos resultados, não conseguiu executar para grandes quantidades de dados por conta de suas etapas sequenciais.

O VDBSCAN-MR teve um desempenho bem melhor que o algoritmo sequencial para média quantidade de dados e conseguiu finalizar a execução para grande quantidade de dados, diferentemente do algoritmo sequencial que não finalizou a execução.

Capítulo 5
Conclusões

5.1 Considerações finais

Os algoritmos de agrupamentos espaciais são técnicas importantes na tarefa de prospecção de dados espaciais. Contudo, o aumento considerável do volume das bases de dados espaciais fez com que esses conjuntos passassem a ter características de *Big Data*, os quais dificultaram o processo de análise de dados.

Este cenário serviu de motivação para o trabalho, o qual desenvolveu dois algoritmos de agrupamento espacial a fim de serem eficientes e escaláveis em aplicações com grandes quantidades de dados espaciais.

Para isto, foi realizado um levantamento bibliográfico das áreas de *Big Data* e prospecção de dados espaciais, onde foram apresentados diferentes tipos de algoritmos de agrupamento espacial, ferramentas de manipulação e otimização no processo de análise de *Big Data*, além dos principais desafios em se trabalhar na área.

Em seguida, foram definidos os algoritmos base para o trabalho e, a partir deles, os novos algoritmos desenvolvidos foram descritos e cada etapa de seu funcionamento foi detalhada.

Por meio dos experimentos realizados, foi possível confirmar que a inclusão de novas abordagens, como a etapa de particionamento e as técnicas de *MapReduce* não alteraram a qualidade dos resultados dos algoritmos base. Além disso, foi possível comprovar que a nova

abordagem automática para identificação dos saltos no gráfico K-Dist possibilitou encontrar um maior número de Eps e, assim, encontrar melhores resultados.

Também foram realizados testes de desempenho e escalabilidade para o algoritmo VDBSCAN-MR, onde foi possível comprovar que a tecnologia de *MapReduce* permitiu ao algoritmo um desempenho melhor que o algoritmo sequencial e a processar quantidade de dados que antes não era possível, o que pode ser usado para aplicações de *Big Data*.

5.2 Contribuições do trabalho

A seguir é apresentada uma lista com as melhorias dos algoritmos desenvolvidos em relação aos algoritmos base com o objetivo de destacar as contribuições do trabalho realizado:

- A etapa de gerar o K-Dist foi implementada em *MapReduce*, o que permitiu uma melhor escalabilidade e desempenho aos algoritmos VDBSCAN-MR e OVDBSCAN-MR;
- O método de identificação automática dos saltos do gráfico K-Dist possibilitou a descoberta de um maior número de valores de Eps e retirou a análise visual, além de ter obtido uma melhoria de resultados em ambos algoritmos;
- A inclusão da etapa de particionamento do espaço de busca permitiu a paralelização do algoritmo DBSCAN no modelo *MapReduce*;
- A implementação da etapa do DBSCAN em *MapReduce* proporcionou melhor desempenho e escalabilidade com relação ao DBSCAN tradicional;
- A etapa de processamento dos resultados do DBSCAN precisou ser incluída devido a paralelização pela técnica *MapReduce*;
- O algoritmo VDBSCAN-MR apresentou rendimento computacional superior ao algoritmo VDBSCAN;
- O algoritmo VDBSCAN-MR viabilizou a aplicação do VDBSCAN a conjuntos de dados com características *Big Data*;
- O algoritmo OVDBSCAN-MR obteve melhor eficácia nos resultados que o VDBSCAN-MR.

5.3 Trabalhos futuros

Para a continuidade do trabalho desenvolvido é interessante analisar alguns pontos dos algoritmos que podem ser melhorados e, consequentemente, tornar ainda melhor o processo de descoberta de conhecimento em dados espaciais, principalmente ao se tratar de conjuntos *Big Data*.

Um dos principais pontos que se deve melhorar é o desempenho da etapa de gerar o gráfico K-Dist, pois mesmo com a implementação da etapa em *MapReduce* o desempenho não é tão bom devido a comparação de cada ponto com todos os outros pontos da base. Esta etapa é a que toma mais tempo do algoritmo.

Outro ponto importante a ser investigado é a etapa de encontrar o valor de *k* ideal no OVDBSCAN-MR. Isso porque é uma etapa que demanda alto poder de processamento e em grandes quantidades de dados deixa o algoritmo ineficiente. Assim, deve-se estudar a possibilidade de paralelizar esta etapa ou até implementá-la em *MapReduce*.

Também deve-se estudar a etapa de particionamento a fim de melhorar a divisão do espaço de busca de maneira que cada região fique com quantidades próximas de pontos, pois em alguns casos, uma partição fica muito maior e com muito mais pontos que as outras e, por isso, demora mais para executar a etapa do DBSCAN.

Outra etapa importante para ser aprimorada é a de identificação automática do salto do gráfico K-Dist a fim de tentar obter os melhores valores de *Eps*, pois em alguns casos são encontrados *Eps* que não encontram nenhum agrupamento.

Por fim, deve-se analisar de maneira geral os algoritmos e tentar aplicar *MapReduce* ou algum outro tipo de paralelismo, como *multithreading*, com o objetivo de se obter um melhor desempenho em grandes conjuntos de dados.

REFERÊNCIAS

AGRAWAL, D.; DAS, S.; EL ABBADI, A. Big data and cloud computing: current state and future opportunities. In: PROCEEDINGS OF THE 14TH INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY. ACM, 2011. p. 530-533.

AJI, A. et al. Demonstration of hadoop-gis: A spatial data warehousing system over MapReduce. In: PROCEEDINGS OF THE 21ST ACM SIGSPATIAL INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS. ACM, 2013. p. 528-531.

_____. Hadoop GIS: a high performance spatial data warehousing system over MapReduce. **Proceedings of the VLDB Endowment**, [s.1.], v. 6, n. 11, p. 1009-1020, 2013.

AMAZON WEB SERVICES. **AWS: – Amazon Web Services**. 2016. Disponível em: <https://aws.amazon.com>. Acesso em: 11 fev. 2016.

BAKSHI, K. Considerations for big data: architecture and approach. In: IEEE AEROSPACE CONFERENCE, 2012. IEEE, 2012. p. 1-7.

CHEN, C. L. P.; ZHANG, C. Y. Data-intensive applications, challenges, techniques and technologies: a survey on big data. **Information Sciences**, Philadelphia, v. 275, p. 314-347, 2014.

CHEN, J.; ZHENG, G.; CHEN, H. ELM-MapReduce: MapReduce accelerated extreme learning machine for big spatial data analysis. In: 10TH IEEE INTERNATIONAL CONFERENCE ON CONTROL AND AUTOMATION (ICCA), 2013. IEEE, 2013. p. 400-405.

DAI, B. R.; LIN, I. C. Efficient map/reduce-based dbscan algorithm with optimized data partition. In: 5TH IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING (CLOUD), 2012. p. 59-66.

DONG, X. L.; SRIVASTAVA, D. Big data integration. **Synthesis Lectures on Data Management**, [s.1], v. 7, n. 1, p. 1-198, 2015.

ELDAWY, A.; MOKBEL, M. F. A demonstration of spatialhadoop: an efficient MapReduce framework for spatial data. **Proceedings of the VLDB Endowment**, [s.1], v. 6, n. 12, p. 1230-1233, 2013.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: **Kdd**. 1996. p. 226-231.

FAHAD, A. et al. A survey of clustering algorithms for big data: taxonomy and empirical analysis. **IEEE Transactions on Emerging Topics in Computing**, New York, v. 2, n. 3, p. 267-279, 2014.

HE, Y. et al. MR-DBSCAN: a scalable MapReduce-based DBSCAN algorithm for heavily skewed data. **Frontiers of Computer Science**, Beijing, v. 8, n. 1, p. 83-99, 2014.

_____. MR-DBSCAN: an efficient parallel density-based clustering algorithm using MapReduce. In: 17TH IEEE INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS (ICPADS), 2011. p. 473-480.

HEMALATHA, M.; SARANYA, N. N. A recent survey on knowledge discovery in spatial data mining. **International Journal of Computer Science**, Canakkale, v. 8, n. 3, p. 473–479, 2011.

JIN, H.; MIAO, B. The research progress of spatial data mining technique. In: 3RD IEEE INTERNATIONAL CONFERENCE ON COMPUTER SCIENCE AND INFORMATION TECHNOLOGY (ICCSIT), 2010. p. 81-84.

KAISLER, S. et al. Big data: Issues and challenges moving forward. In: 46TH HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES (HICSS), 2013. p. 995-1004.

KATAL, A.; WAZID, M.; GOUDAR, R. H. Big data: Issues, challenges, tools and Good practices. In: 6TH IEEE INTERNATIONAL CONFERENCE ON CONTEMPORARY COMPUTING (IC3), 2013. p. 404-409.

KIM, Y. et al. DBCURE-MR: an efficient density-based clustering algorithm for large data using MapReduce. **Information Systems,** Elmsford, v. 42, p. 15-35, 2014.

LABRINIDIS, A.; JAGADISH, H. Challenges and opportunities with big data. **Proceedings of the VLDB Endowment**, [s.l.], v. 5, n. 12, p. 2032–2033, 2012.

LIU, P.; ZHOU, D.; WU, N. VDBSCAN: varied density based spatial clustering of applications with noise. In: INTERNATIONAL CONFERENCE ON SERVICE SYSTEMS AND SERVICE MANAGEMENT, 2007. p. 1-4.

LOEWEN, G.; GALLOWAY, M.; VRBSKY, S. On the performance of apache hadoop in a tiny private iaas cloud. In: TENTH INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY: NEW GENERATIONS (ITNG), 2013. p. 189-195.

MARINESCU, D. C. **Cloud computing:** theory and practice. Newnes, 2013.

MEDEIROS, C. A. **Extração de conhecimento em bases de dados espaciais**: algoritmo CHSMST+. 106f. 2014. Dissertação (Mestrado em Ciência da Computação) - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista "Júlio de Mesquita Filho", São José do Rio Preto, 2014.

MENNIS, J.; GUO, D. Spatial data mining and geographic knowledge discovery—an introduction. **Computers, Environment and Urban Systems**, New York, v. 33, n. 6, p. 403–408, 2009.

NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY. **NGA GEOnet Names Server (GNS)**. 2015. Disponível em: <http://geonames.nga.mil/gns/html/index.html>. Acesso em: 25 set. 2015.

OSMAN, A.; EL-REFAEY, M.; ELNAGGAR, A. Towards real-time analytics in the cloud. In: NINTH IEEE WORLD CONGRESS ON SERVICES (SERVICES), 2013. p. 428-435.

O'DRISCOLL, A.; DAUGELAITE, J.; SLEATOR, R. D. 'Big data', hadoop and cloud computing in genomics. **Journal of Biomedical Informatics**, San Diego, v. 46, n. 5, p. 774-781, 2013.

PATEL, V. D.; MENARIA, S. Performance optimization of clustering on GPU. **International Journal for Scientific Research & Development**, Gujarat, v. 1, n. 4, p. 2–5, 2014.

PATTABIRAMAN, V. et al. A novel spatial clustering with obstacles and facilitators constraint based on edge deduction and k-medoids. In: INTERNATIONAL CONFERENCE ON COMPUTER TECHNOLOGY AND DEVELOPMENT, 2009, Kota Kinabalu. **Proceedings**… Kota Kinabalu, 2009. v.1, p.402-406.

SAGIROGLU, S.; SINANC, D. Big data: a review. In: INTERNATIONAL CONFERENCE ON COLLABORATION TECHNOLOGIES AND SYSTEMS (CTS), 2013. p. 42-47.

SHEKHAR, S. et al. Trends in spatial data mining. **Data mining:** next generation challenges and future directions. p. 357-380, 2003.

SHULIANG, W.; GANGYI, D.; MING, Z. Big spatial data mining. In: IEEE INTERNATIONAL CONFERENCE ON BIG DATA, 2013. p. 13-21.

SINGH, S.; SINGH, N. Big data analytics. In: INTERNATIONAL CONFERENCE ON COMMUNICATION, INFORMATION & COMPUTING TECHNOLOGY (ICCICT), 2012. p. 1-4.

TALIA, D. Clouds for scalable big data analytics. **Computer**, 2013.

TANG, W.; FENG, W. Parallel map projection of vector-based big spatial data: coupling cloud computing with graphics processing units. **Computers, Environment and Urban Systems**, 2014.

TARIQ, A.; ASGHAR, S.; SAJID, N. A. Critical analysis of DBSCAN variations. In: INFORMATION AND EMERGING TECHNOLOGIES (ICIET), 2010. p. 1-6.

VALÊNCIO, C. R. et al. VDBSCAN+: performance optimization based on GPU parallelism. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING, APPLICATIONS AND TECHNOLOGIES (PDCAT), 2013. p. 23-28.

WANG, J. et al. Research of GIS-based spatial data mining model. In: SECOND INTERNATIONAL WORKSHOP ON KNOWLEDGE DISCOVERY AND DATA MINING, 2009.

WANG, W. et al. Improved VDBSCAN with global optimum K. In: PROCEEDINGS OF THE THIRD INTERNATIONAL CONFERENCE ON DIGITAL INFORMATION AND COMMUNICATION TECHNOLOGY AND ITS APPLICATIONS (DICTAP). 2013. p. 225-228.

WHITE, T. **Hadoop:** the definitive guide. 2015.

YEUNG, A. K. W.; HALL, G. B. **Spatial database systems:** design, implementation and project management. Amsterdam: Springer, 2007.

YOO, J. S.; BOULWARE, D. A framework of spatial co-location mining on MapReduce. In: IEEE INTERNATIONAL CONFERENCE ON BIG DATA, 2013. p. 44-44.

ZHANG, D. Inconsistencies in big data. In: 12TH IEEE INTERNATIONAL CONFERENCE ON COGNITIVE INFORMATICS & COGNITIVE COMPUTING (ICCI* CC), 2013. p. 61-67.

ZHANG, L. et al. Moving big data to the cloud: an online cost-minimizing approach. **IEEE Journal on Selected Areas in Communications,** New York, v. 31, n. 12, p. 2710-2721, 2013.

ZHENG, Z.; ZHU, J.; LYU, M. R. Service-generated big data and big data-as-a-service: an overview. In: IEEE INTERNATIONAL CONGRESS ON BIG DATA (BIGDATA CONGRESS), 2013. p. 403-410.

ZHONG, Y. et al. Towards parallel spatial query processing for big spatial data. In: 26TH IEEE INTERNATIONAL ON PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM WORKSHOPS & PHD FORUM (IPDPSW), 2012. p. 2085-2094.