UNIVERSIDADE FEDERAL DO CEARÁ
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ANTONIO CAVALCANTE ARAUJO NETO

G2P-DBSCAN: ESTRATÉGIA DE PARTICIONAMENTO DE DADOS E DE PROCESSAMENTO DISTRIBUÍDO DO DBSCAN COM MAPREDUCE

FORTALEZA, CEARÁ

2015

ANTONIO CAVALCANTE ARAUJO NETO

G2P-DBSCAN: ESTRATÉGIA DE PARTICIONAMENTO DE DADOS E DE PROCESSAMENTO DISTRIBUÍDO DO DBSCAN COM MAPREDUCE

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Área de concentração: Banco de Dados

Orientador: Prof. Dr. Javam de Castro Machado

FORTALEZA, CEARÁ

2015

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

A687g Araujo Neto, Antonio Cavalcante.
G2P-DBSCAN: Estratégia de particionamento de dados e de processamento distribuído do DBSCAN com MapReduce./ Antonio Cavalcante Araujo Neto. – 2015.
63 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2015.
Área de Concentração: Ciência da Computação.
Orientação: Prof. Dr. Javam de Castro Machado.
Coorientação: Prof. Dr. José Antônio Fernandes de Macêdo.

1. Particionamento de dados. 2. Clusterização. 3. MapReduce. I. Título.

CDD 005

ANTONIO CAVALCANTE ARAUJO NETO

G2P-DBSCAN: ESTRATÉGIA DE PARTICIONAMENTO DE DADOS E DE PROCESSAMENTO DISTRIBUÍDO DO DBSCAN COM MAPREDUCE

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação. Área de concentração: Banco de Dados

Aprovada em: __/__/____

BANCA EXAMINADORA

Prof. Dr. Javam de Castro Machado
Universidade Federal do Ceará - UFC
Orientador

Prof. Dr. José Antônio Fernandes de Macêdo
Universidade Federal do Ceará - UFC
Coorientador

Prof. Dr. José Neuman de Souza
Universidade Federal do Ceará - UFC

Prof. Dr. Altigran Soares da Silva
Universidade Federal do Amazonas - UFAM

À minha mãe.

AGRADECIMENTOS

Agradeço a minha mãe Ana Angélica, por toda dedicação dada a minha educação e por sempre ter acreditado incondicionalmente no meu potencial.

À toda minha família, pelo apoio e amor que sempre me foram dados e por acreditarem na minha capacidade de alcançar meus objetivos.

À minha namorada Camila, pela paciência, pelo carinho, por me acompanhar e me apoiar durante todo o percurso do mestrado e por contribuir diretamente na finalização desse trabalho.

Ao Professor Javam Machado, por todas as oportunidades que me foram concedidas ao longo desses dois anos, que foram de fundamental importância para os bons resultados obtidos.

Ao Professor José Antônio, pelas suas contribuições a esse trabalho e pela disponibilidade em participar da banca avaliadora.

Ao Professor Altigran Soares, que mesmo de maneira remota aceitou participar da banca avaliadora e compartilhar de sua experiência para o aperfeiçoamento deste trabalho.

Ao Professor Neuman de Souza, pela disponibilidade em participar da banca avaliadora e contribuir para a melhoria deste e de futuros trabalhos.

Por fim, aos amigos do ARIDa e do LSBD por todos os momentos compartilhados, por todas as contribuições e por todo o apoio durante esse período.

“DAS UTOPIAS
Se as coisas são inatingíveis... ora!
Não é motivo para não querê-las...
Que tristes os caminhos, se não fora
A presença distante das estrelas!”

(Mário Quintana)

RESUMO

Clusterizaçao é uma técnica de mineração de dados que agrupa elementos de um conjunto de dados de forma que os elementos que pertencem ao mesmo grupo são mais semelhantes entre si que entre elementos de outros grupos. Nesta dissertação nós estudamos o problema de processar o algoritmo de clusterização baseado em densidade DBSCAN de maneira distribuída através do paradigma MapReduce. Em processamentos distribuídos é importante que as partições de dados a serem processadas tenham tamanhos aproximadamente iguais, uma vez que o tempo total de processamento é delimitado pelo tempo que o nó com uma maior quantidade de dados leva para finalizar a computação dos dados a ele atribuídos. Por essa razão nós também propomos uma estratégia de particionamento de dados, chamada G2P, que busca distribuir o conjunto de dados de forma balanceada entre as partições e que leva em consideração as características do algoritmo DBSCAN. Mais especificamente, a estratégia G2P usa estruturas de grade e grafo para auxiliar na divisão do espaço em regiões de baixa densidade. Já o processamento distribuído do algoritmo DBSCAN se dá por meio de duas fases de processamento MapReduce e uma fase intermediária que identifica *clusters* que podem ter sido divididos em mais de uma partição, chamados de candidatos à junção. A primeira fase de MapReduce aplica o algoritmo DSBCAN nas partições de dados individualmente, e a segunda verifica e corrige, caso necessário, os *clusters* candidatos à junção. Experimentos utilizando dados reais mostram que a estratégia G2P-DBSCAN se comporta melhor que a solução utilizada para comparação em todos os cenários considerados, tanto em tempo de execução quanto em qualidade das partições obtidas.

Palavras-chave: DBSCAN. MapReduce. Particionamento de dados. Clusterização.

ABSTRACT

Clustering is a data mining technique that groups elements of a data set in such a way that elements from the same group are more similar to each other than to those from other groups. In this thesis we study the problem of processing the density-based clustering algorithm DBSCAN distributedly through the MapReduce paradigm. In distributed processing it is important that the partitions to be processed have roughly the same size, since the total processing time is bounded by the time that the node with a larger amount of data takes to complete the computation of the data assigned to it. For this reason we also propose a data partitioning strategy named G2P, which aims at distributing the data set in a balanced way between partitions and takes into account the characteristics of the DBSCAN algorithm. More specifically, the G2P strategy uses grid and graph structures to assist in the division of the space in low density regions. The distributed processing of the DBSCAN algorithm is done through two MapReduce phases and an intermediate phase that identifies clusters that may have been divided into more than one partition, called merge candidates. The first MapReduce phase applies the DSBCAN algorithm in the partitions individually. The second one checks and corrects, if necessary, merge candidates clusters. Experiments using real data sets show that the G2P-DBSCAN strategy outperforms the adopted baseline in all the considered scenarios, both in the runtime and in the quality of the partitions obtained.

Keywords: DBSCAN. MapReduce. Data partitioning. Clustering.

LISTA DE FIGURAS

LISTA DE TABELAS

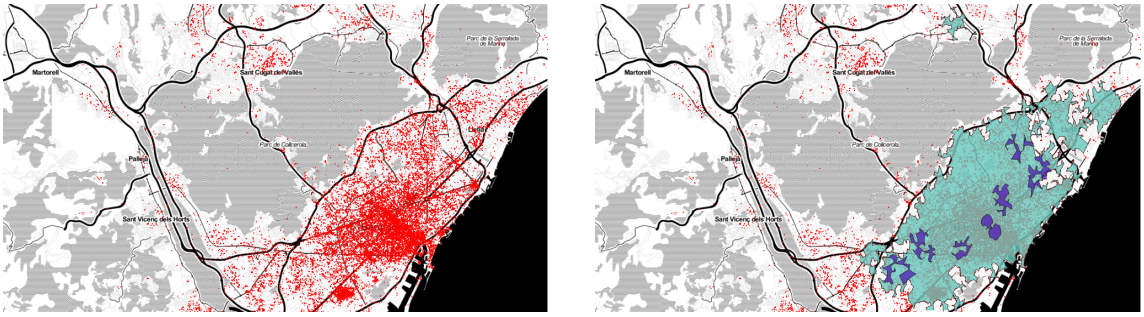
SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

A difusão do uso de dispositivos móveis tais como *smartphones* e GPS's tornou possível a coleta de grandes volumes de dados de localização de usuários. A extração de informação desses dados tem atraído cientistas de diversos domínios, não se tratando somente de um desafio intelectual, mas também de um tópico de grande importância para áreas tais como planejamento urbano, mobilidade sustentável, engenharia de transportes e saúde pública. Ainda, a existência desse grande volume de dados possibilita a criação de ferramentas que podem auxiliar no entendimento do comportamento humano e na descoberta de padrões e modelos ocultos que caracterizam o comportamento de indivíduos em seus cotidianos (GIANNOTTI et al., 2011).

Algoritmos de análise são desenvolvidos para identificar e correlacionar fatos observáveis com dados obtidos de usuários. Por exemplo, é possível relacionar o fluxo de movimento de automóveis em uma cidade com o horário do dia, ou uma maior concentração de pessoas em um certo local com um grande evento ou até mesmo o impacto de uma notícia com a quantidade de comentários sobre a mesma nas redes sociais. Como podemos ver na Figura 3.3a, não é possível reconhecer padrões e seus limites apenas com a visualização dos dados sem tratamento algum, da forma como foram coletados. Além disso, tal visualização não permite identificar os parâmetros que definem esses padrões. Contudo, a partir de uma definição formal da informação desejada é possível buscar de forma algorítmica a presença de padrões ou de grupos de elementos similares em um conjunto de dados, como podemos ver na Figura 1.1b, onde, por exemplo, a cor azul identifica uma certa concentração de pontos que satisfazem um critério estabelecido.



(a) Dados espaciais brutos (b) Padrões encontrados

Figura 1.1: Visualização de padrões em dados espaciais

O processo de identificação de padrões ou de extração de qualquer tipo de informação de um conjunto de dados é chamado de mineração de dados (HAN; KAMBER, 2000). Dentre as técnicas de mineração, destacam-se algoritmos de agrupamento, classificação, mineração de texto e descoberta de regras de associação. Essas técnicas são capazes de produzir informações úteis a partir de dados brutos coletados de usuários, dependendo da qualidade dos mesmos e da informação que se deseja extrair.

As técnicas de classificação buscam categorizar conjuntos de dados em classes previamente estabelecidas, a partir de alguma medida de similaridade. Já as técnicas de mineração de texto buscam extrair informações de dados de texto não estruturados. Quanto às regras de associação, essas são criadas a partir da análise de um conjunto de dados e da descoberta de correlação entre os elementos desse conjunto. Dentre as técnicas mencionadas, esse trabalho foca naquelas em que os dados são agrupados de acordo com alguma medida de similaridade. Agrupamento e classificação são as principais técnicas que se encaixam nessa categoria. Contudo, na classificação, por ser uma técnica supervisionada, os elementos são divididos em classes conhecidas a priori. Já no agrupamento, a descoberta dos grupos, também chamados de *clusters*, é um processo não supervisionado, ou seja, os agrupamentos são feitos de maneira automática sem o auxílio de rótulos preestabelecidos. Como nem sempre se tem um conhecimento prévio sobre o conjunto de dados, o agrupamento se torna uma técnica mais atrativa para descoberta de estruturas nos dados.

No contexto de agrupamento, os algoritmos baseados em densidade são capazes de detectar *clusters* com formatos arbitrários em conjuntos de dados. Dentre esses algoritmos destaca-se o DBSCAN (*Density-Based Spatial Clustering of Application with Noise*) (ESTER et al., 1996) como um dos mais utilizados pela comunidade de banco de dados. O DBSCAN também apresenta como vantagem a capacidade de tratar a presença de ruídos nos dados.

A elevada taxa de produção de dados espaciais cria a necessidade de processamento de uma quantidade de dados cada vez maior. A alta complexidade do algoritmo DBSCAN não permite que sua execução centralizada seja eficiente nesses casos (GAN; TAO, 2015). Problemas que fazem uso intensivo de CPU em suas resoluções ou que processam grandes quantidades de dados são essencialmente resolvidos através da distribuição de tarefas entre diversos núcleos de processamento (SCHADT et al., 2010). Com o advento e a popularização de infraestruturas de computação em nuvem, a análise de grandes volumes de dados se tornou factível para uma gama maior de empresas e usuários. Isso se deve à facilidade de alocação e desalocação de recursos computacionais, fornecendo ao usuário um grande poder computacional a baixos custos, uma vez que o mesmo paga apenas pelos recursos utilizados.

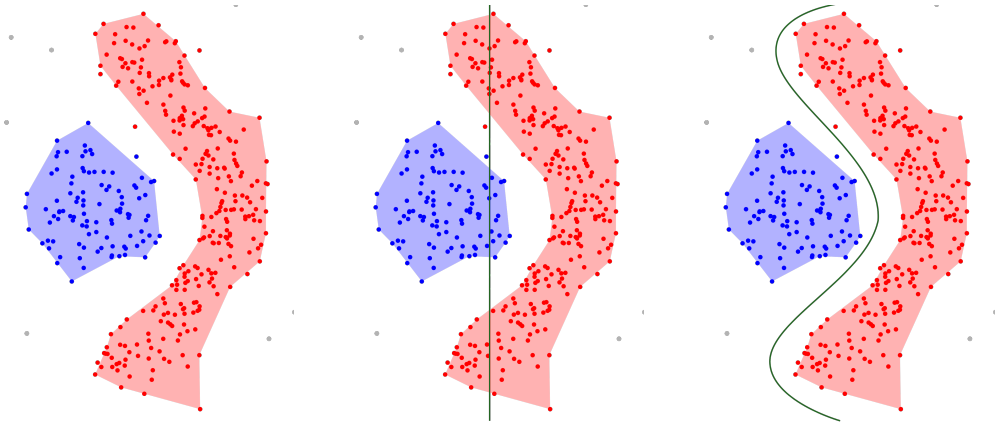
O surgimento do paradigma MapReduce (DEAN; GHEMAWAT, 2004) contribuiu de maneira significativa para a criação de programas distribuídos e paralelos relativamente sofisticados, uma vez que é necessário apenas a definição de duas funções, Map e Reduce, além da sua capacidade de ser executado em máquinas comuns, o que também colaborou com a sua popularização.

Considerando então a necessidade de processamento e a facilidade de alocar recursos computacionais, surgiram abordagens que se beneficiam de infraestruturas de larga escala para atender as demandas de processamento de dados. Diversos trabalhos propõem estratégias de computação distribuída e paralela em *clusters* computacionais para acelerar os processos de tratamento e análise de dados, tais como métodos de seleção de gene (ISLAM et al., 2015), classificação de dados (RÍO et al., 2015) e processamento de grafos (SVENDSEN; MUKHERJEE; TIRTHAPURA, 2015). Dentre esses trabalhos, (DAI; LIN, 2012), (HE et al., 2011) propõem estratégias para tornar a execução do DBSCAN mais eficiente utilizando o paradigma MapReduce (DEAN; GHEMAWAT, 2004). Em outros trabalhos a distribuição do algoritmo DBSCAN

é feita através de supercomputadores (WELTON; SAMANAS; MILLER, 2013) e de arquiteturas *shared-nothing* proposta pelos próprios autores (XU; JÄGER; KRIEGEL, 1999).

Sabemos que o tempo de execução do algoritmo DBSCAN é diretamente proporcional à quantidade de dados a processar. Em soluções que envolvem processamento distribuído, o tempo total de processamento é limitado pelo tempo que o nó com uma maior quantidade de dados leva para finalizar a computação dos dados a ele atribuídos. Dessa forma, é ideal que os dados estejam distribuídos de maneira aproximadamente uniforme entre os nós de computação. Assim, é preciso que estratégias de particionamento também sejam consideradas nessas soluções para garantir um bom balanceamento dos dados e, consequentemente, uma boa performance em suas execuções.

Nesta dissertação nós abordamos o problema de processar de maneira distribuída o algoritmo DBSCAN através do paradigma MapReduce, bem como o de particionar os dados de maneira conveniente para o processamento do DBSCAN distribuído. Embora outros trabalhos já tenham abordado este problema, como (DAI; LIN, 2012) e (HE et al., 2011), a estratégia de particionamento proposta por ambos divide o espaço recursivamente somente nas direções horizontal ou vertical, técnica conhecida como BSP (*Binary Space Partitioning*), que nem sempre é adequada para o particionamento de dados para agrupamento, como podemos ver na Figura 1.2b. Além disso, essas técnicas requerem que parte dos dados sejam replicados nas partições. Diferente desses trabalhos, nós propomos uma estratégia de particionamento baseada em estruturas de grade e grafo que beneficia a execução distribuída do algoritmo DBSCAN, uma vez que a divisão do espaço é feita em regiões menos densas, como mostrado na Figura 1.2c. Além disso, nossa proposta não requer replicação de dados e a junção de *clusters* acontece de maneira distribuída, diferentemente dos trabalhos citados acima.



(a) Conjunto de dados (b) BSP (Binary Space Partitioning) (c) Particionamento Ideal

Figura 1.2: Visualização de padrões em dados espaciais

1.2 Objetivos

Dado o cenário motivacional apresentado acima, o objetivo principal desse trabalho é estudar e propor estratégias que permitam a computação do algoritmo DBSCAN de forma eficiente, ou seja, capaz de processar um grande volume de dados em tempo hábil. Para tal, os seguintes objetivos específicos foram estabelecidos:

- Propor uma estratégia de processamento distribuído para o algoritmo DBSCAN;
- Propor uma estratégia de particionamento de dados para o processamento distribuído do algoritmo DBSCAN;

1.3 Contribuições

Nós propomos G2P (*Grid and Graph Partitioning*), uma estratégia de particionamento de dados espaciais baseada em dois tipos de estruturas de dados que representam o conjunto de dados de forma a beneficiar a execução distribuída do algoritmo DBSCAN. Tal estratégia tem como vantagens o balanceamento de carga entre as partições geradas e a maneira como o espaço é dividido, procurando minimizar o número de *clusters* quebrados em diferentes partições. Também propomos uma estratégia de processamento distribuído do algoritmo DBSCAN através do paradigma MapReduce e provamos a sua corretude em relação à execução centralizada do DBSCAN. A solução proposta nesse trabalho não requer que dados sejam replicados, diferentemente dos outros trabalhos que se propõem a resolver esse problema.

Os seguintes itens resumem as contribuições deste trabalho:

- Estratégia de particionamento G2P (*Grid and Graph Partitioning*) adequada para o processamento distribuído do algoritmo DBSCAN.
- Estratégia de processamento distribuído do algoritmo DBSCAN através do paradigma MapReduce nomeada G2P-DBSCAN.

1.3.1 Publicações

Os esforços durante o processo de pesquisa para esta dissertação possibilitaram as seguintes publicações:

- Ticiana L. Coelho da Silva, Antônio C. Araújo Neto, Regis Pires Magalhães, Victor A. E. de Farias, José A. F. de Macêdo, Javam C. Machado. Efficient and Distributed DBSCAN Algorithm Using MapReduce to Detect Density Areas on Traffic Data. In: 16th International Conference on Enterprise Information Systems (ICEIS 2014).
- Antônio Cavalcante Araújo Neto, Ticiana Linhares Coelho da Silva, Victor Aguiar Evangelista de Farias, José Antônio Fernandes de Macêdo, Javam de Castro Machado. G2P:

A Partitioning Approach for Processing DBSCAN with MapReduce. In: 14th Web and Wireless Geographical Information Systems, 2015.

1.4 Estrutura dessa dissertação

Os próximos capítulos desta dissertação estão estruturados da seguinte maneira:

- O Capítulo 2 apresenta a fundamentação teórica necessária para o entendimento desse trabalho, detalhando os algoritmos, conceitos e tecnologias abordadas na solução proposta.
- O Capítulo 3 explica a estratégia de particionamento de dados G2P e a estratégia de DBSCAN distribuído em detalhes.
- O Capítulo 4 apresenta os trabalhos relacionados encontrados na literatura.
- O Capítulo 5 apresenta os resultados experimentais e compara o G2P-DBSCAN com trabalhos relacionados.
- O Capítulo 6 conclui esta dissertação com o resumo dos resultados alcançados e sugestões de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo consiste da fundamentação teórica necessária para o entendimento desse trabalho, incluindo os algoritmos e os problemas conhecidos na literatura discutidos nessa dissertação, bem como as tecnologias que foram utilizadas na implementação das soluções propostas. Uma visão geral sobre as categorias e algoritmos de agrupamento é apresentada na seção 2.1. Na seção 2.2 apresentamos o algoritmo DBSCAN em detalhes e suas características. A seção 2.3 discute a estrutura de índices *k*-d-Tree, que é utilizada nesse trabalho para auxiliar a manipulação do conjunto de dados pelo algoritmo DBSCAN. Na seção 2.4 discutimos o paradigma MapReduce e sua implementação aberta Hadoop. A seção 2.5 define e apresenta conceitos relacionados ao problema de particionamento de grafos e, finalmente, a seção 2.6 conclui esse capítulo.

2.1 Agrupamento

Agrupamento é o processo de encontrar grupos, também chamados de *clusters*, em um conjunto de dados a partir de suas características, de forma que elementos que pertencem ao mesmo grupo são mais similares entre si do que entre elementos de outros grupos. As medidas de similaridades podem variar de acordo com o algoritmo ou de acordo com os dados. Ao contrário do processo de classificação, onde existem rótulos predefinidos para categorizar os dados, no agrupamento não se sabe a priori quantos e quais *clusters* serão encontrados. Os principais métodos de agrupamento existentes na literatura podem ser categorizados da seguinte forma:

- Métodos baseados em particionamento

  Esses métodos consistem na maneira mais fundamental de agrupamento. Eles recebem como entrada, além de um conjunto de dados D, um inteiro k que representa o número de *clusters* que se deseja encontrar. Inicialmente, o centro de cada grupo, que não necessariamente é um ponto de D, é determinado. A cada passo, os elementos de D são associados ao *cluster* definido pelo centro mais próximo de cada ponto. A distância entre um elemento de D e um centro é calculada de acordo com algum critério de similaridade estabelecido pelo algoritmo. Após a etapa de associação, os centros dos grupos são atualizados. Esse processo é repetido até que um ótimo local seja atingido. Dentre os algoritmos mais conhecidos nessa categoria destacam-se o *k*-means e o *k*-medoids.

  No algoritmo *k*-means os centros dos grupos são conhecidos como centróides. A escolha dos centróides é comumente feita através de amostras aleatórias de D. Uma vez finalizada a etapa de associação de cada elemento do conjunto D ao seu centróide mais próximo, os valores dos centróides são atualizados de acordo com a média dos elementos que estão associados àquele centróide. Esses passos são repetidos até que os centróides não mudem.

  A dificuldade da escolha dos valores iniciais para os centróides, além do número de centróides são as principais desvantagens do algoritmo *k*-means. Ainda, não há garantia que

o algoritmo convirja para um ótimo global, tornando o resultado do agrupamento bastante sensível à inicialização dos centróides. Por outro lado, a baixa complexidade computacional do algoritmo descrito o torna atrativo para aplicações onde existe um conhecimento prévio sobre o domínio do conjunto de dados.

- Métodos hierárquicos

  Os métodos hierárquicos têm como resultado um aninhamento de grupos e o grau de similaridade entre esses grupos. Os algoritmos hierárquicos podem ser subcategorizados em aglomerativos e divisivos. Os aglomerativos funcionam de uma maneira *botton-up*, assumindo inicialmente que cada elemento do conjunto de dados representa um cluster. A partir daí, de acordo com algum critério similaridade, os clusters se unem. Já os divisivos trabalham de maneira *top-down*, considerando todo o conjunto de dados como um só cluster, que é dividido de maneira recursiva de acordo com a medida de similaridade estabelecida.

  O algoritmo BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*), considerado um dos métodos hierárquicos mais utilizados na literatura, faz a integração de outros métodos, tais como particionamento ou densidade, com organização hierárquica de *clusters*. A agrupamento acontece essencialmente em duas fases. Na primeira delas, uma árvore balanceada é construída através de estruturas chamadas *clustering feature* (CF), que sumarizam informações estatísticas sobre os *clusters*. Essa árvore, chamada de CF-*tree*, é utilizada para representar a hierarquia dos clusters. Na segunda fase, um algoritmo de agrupamento selecionado é aplicado às folhas da CF-*tree*, removendo clusters esparsos e agrupando os mais densos em clusters maiores. Uma das maiores vantagens do algoritmo BIRCH é sua complexidade linear em relação ao número de objetos a ser agrupado.

- Métodos baseados em densidade

  Os métodos baseados em densidade assumem que os elementos que pertencem a um determinado *cluster* seguem uma mesma distribuição. O objetivo dessas técnicas é identificar os *clusters* de acordo com os parâmetros que descrevem essa distribuição. Em outras palavras, os *clusters* são modelados como regiões densas do conjunto de dados, divididos por áreas de regiões esparsas. Dentre as vantagens desses métodos destaca-se a capacidade de identificar *clusters* de formatos arbitrários, enquanto que os métodos baseados em particionamento, por exemplo, apresentam melhores resultados com *clusters* de formato circular.

  Dentre os algoritmos baseados em densidade existentes na literatura, o DBSCAN se destaca como um dos mais utilizados. As características e detalhes do algoritmo DBSCAN são descritos na Seção 1.

- Métodos baseados em grade

  Enquanto que os outros métodos de agrupamento discutidos são orientados aos dados, os métodos baseados em grade são orientados ao espaço. Essencialmente, o espaço é dividido em células de uma estrutura de grade independente da distribuição dos dados.

Essa estrutura é capaz de representar os objetos do conjunto de entrada através das células da grade. Todas as operações de agrupamento são realizadas na estrutura de grade criada. Essa é uma das principais vantagens dessa classe de algoritmos, uma vez que seu desempenho não depende diretamente do número de entradas do conjunto de dados, mas sim do número de células em cada dimensão da grade.

O algoritmo STING (*STatistical INformation Grid*) se baseia na construção de diversas camadas de grade, onde células de uma camada mais alta são subdivididas para a criação de células nas camadas mais baixas. A partir dessa organização, informações estatísticas sobre as células são pré-computadas e armazenadas para a identificação dos *clusters*. Os resultados produzidos pelo STING se aproximam do agrupamento produzido pelo DBSCAN a medida que a granularidade da estrutura de grade se aproxima de 0, podendo também ser considerado como um método baseado em densidade. O STING também apresenta como vantagem complexidade linear de tempo em relação ao número de objetos a serem agrupados.

2.2 O algoritmo DBSCAN

DBSCAN (Density-based Spatial Clustering of Applications with Noise) é um algoritmo de agrupamento baseado em densidade vastamente utilizado pela comunidade científica. Seu objetivo principal consiste em encontrar concentrações de elementos que estão espacialmente próximos. Em outras palavras, o algoritmo busca por pontos que possuem mais que um limiar de vizinhos dentro de um certo raio. Caso um elemento p satisfaça essa propriedade, os vizinhos de p pertencerão ao mesmo *cluster* que p e o mesmo processo é aplicado a todos os seus vizinhos. Além do conjunto de dados a ser agrupado, o DBSCAN recebe também dois parâmetros de entrada: $minPoints$ e eps. O primeiro deles se refere à quantidade mínima de pontos em um certo raio de vizinhança para a formação de um *cluster*. Já o segundo parâmetro se refere ao raio no qual a verificação de vizinhança é realizada. A função de distância utilizada para determinar a vizinhança de um certo ponto é definida de acordo com o tipo de dado a ser agrupado e deve obedecer às restrições de uma função de distância, tais como simetria e a desigualdade triangular, além de assumir que a distância entre dois elementos x e y só é igual a 0 se $x = y$. Dentre suas vantagens, o algoritmo DBSCAN se destaca por ser capaz de encontrar *clusters* com formatos arbitrários, além de ser capaz de lidar com ruídos nos dados, característica que não está presente na maioria dos algoritmos de agrupamento, como mostrado na Figura 2.1.

As definições básicas utilizadas no DBSCAN são apresentadas a seguir:

- $|A|$: cardinalidade do conjunto A.
- $N_{eps}(p)$: é o conjunto de pontos q que estão a uma distância menor que eps do ponto p. Também é chamado de conjunto dos vizinhos de p.
- Diretamente alcançável por densidade (DDR): um ponto p é DDR a partir de um ponto q se $p \in N_{eps}(q)$ e $|N_{eps}(q)| \geq minPoints$.

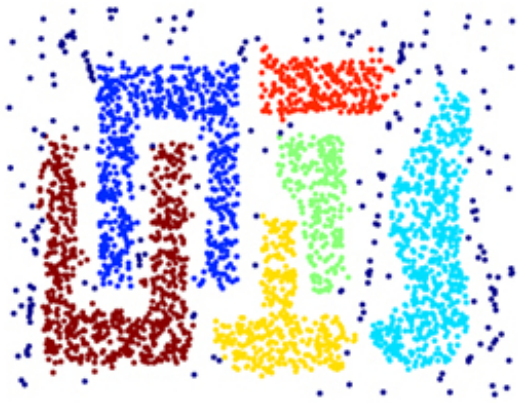


Figura 2.1: *Clusters* de formatos arbitrários encontrados pelo algoritmo DBSCAN
Fonte: *Data Mining - The Hypertextbook*

- Alcançável por densidade (DR): um ponto p é DR a partir de um ponto q, se existe uma sequência de pontos $\{p_1, ..., p_n\}$ onde $p_1 = p$ e $p_n = q$, tal que p_{i+1} é DDR a partir de p_i.
- Conectado por densidade (DC): Um ponto p está conectado por densidade a um ponto q se existe um ponto o tal que p e q são DR a partir de o.
- *Core point*: um ponto p é classificado como *core point* se $|N_{eps}(o)| \geq minPoints$.
- Border point: um ponto p é classificado como *border point* se $|N_{eps}(p)| < minPoints$ e p é DDR a partir de um core point.
- Noise: Um ponto p é classificado como *noise* se $|N_{eps}(q)| < minPoints$ e p não é DDR a partir de nenhum *core point*.

No contexto do algoritmo DBSCAN, um *cluster C* é definido como um subconjunto não vazio dos dados que satisfaz as seguintes propriedades:

- Maximalidade: Para quaisquer dois pontos p e q, se $p \in C$ e q é alcançável por densidade (DR) a partir de p, então $q \in C$.
- Conectividade: Para quaisquer dois pontos $p, q \in C$, p é q são conectados por densidade (DC).

O Algoritmo 1 mostra o pseudocódigo do DBSCAN. Para cada elemento p ainda não visitado o conjunto dos seus vizinhos $N_{eps}(p)$ é encontrado, como podemos ver entre as linhas 2 e 5. Caso a cardinalidade desse conjunto de vizinhos seja maior que o valor de $minPoints$ (Linha 6 do Algoritmo), um novo *cluster C* é criado e p e seus vizinhos serão atribuídos a C. Ainda, os pontos não visitados de C serão expandidos em um processo similar. A agrupamento acaba quando todos os elementos do conjunto foram visitados.

O Algoritmo 2 implementa a função de expansão de um *cluster C*. A expansão de um *cluster* a partir de um ponto p encontra todos os elementos que são conectados por

densidade (DC) a p. Essa função recebe como entrada p, seu conjunto de vizinhos $NeighborPts$, o identificador C do *cluster* a ser expandido, e os parâmetros eps e $minPoints$. Para cada ponto p' do conjunto de vizinhos de p, caso esse ponto ainda não tenha sido visitado, sua vizinhança é recuperada e adicionada ao conjunto $NeighborPts$ de vizinhos que serão verificados (linha 8). Após sua verificação, caso p' não pertença a nenhum *cluster*, ele é adicionado ao *cluster* C (linhas 11 a 13). O processo finaliza quando o conjunto $NeighborPts$ está vazio.

Algoritmo 1: DBSCAN

```
   Entrada: D, eps, minPoints
1  Início
2      para p ∈ D faça
3          se p ainda não foi visitado então
4              marcar p como visitado;
5              NeighborPts = N_eps(p);
6              se |NeighborPts| < minPoints então
7                  mark p as noise;
8              senão
9                  C = próximo cluster;
10                 expandirCluster(p, NeighborPts, C, eps, minPoints)
11             fim
12         fim
13     fim
14 fim
```

Algoritmo 2: expandirCluster

```
   Entrada: p, NeighborPts, C, eps, minPoints
1  Início
2      adicionar p ao cluster C;
3      para p' ∈ NeighborPts faça
4          se p' ainda não foi visitado então
5              marcar p' como visitado;
6              NeighborPts' = N_eps(p')
7              se |NeighborPts'| >= minPoints então
8                  NeighborPts = NeighborPts ∪ NeighborPts'
9              fim
10         fim
11         se p' ainda não é membro de nenhum cluster então
12             adicionar p' ao cluster C
13         fim
14     fim
15 fim
```

Como podemos ver nos Algoritmos 1 e 2, a complexidade do DBSCAN depende diretamente do custo computacional para recuperação da vizinhança de um ponto (linhas 7 e 6 dos Algoritmos 1 e 2, respectivamente). Em uma solução ingênua essa operação poderia ser

executada em tempo linear, onde uma simples busca exaustiva em todo o conjunto de dados retornaria apenas os elementos a uma certa distância do ponto de consulta. Tal solução faria com que a complexidade do algoritmo DBSCAN fosse $\mathcal{O}(n^2)$. Por outro lado, com o auxílio de estruturas de índices, como *k*-d-Trees ou R-Trees, a complexidade do DBSCAN pode ser significantemente reduzida. Recentemente foi provado em (GAN; TAO, 2015) que para dimensões maiores que 2 o algoritmo DBSCAN executa em uma complexidade $\Omega(n^{4/3})$. No entanto, consideraremos nesse trabalho conjuntos de dados de apenas duas dimensões.

2.3 *k*-d-Tree

Ao lidar com conjuntos de dados volumosos, certas operações sobre esses dados podem ser computacionalmente custosas caso não haja uma estrutura de índices para auxiliá-las. No contexto desse trabalho, operações como recuperação de vizinho mais próximo ou de vizinhança de um ponto se tornam mais eficientes através do uso de estruturas de índices.

A *k*-d-Tree é uma estrutura de indexação de dados espaciais *k*-dimensionais. Essa estrutura é representada por uma árvore binária onde cada nó é um objeto com k dimensões. Os nós internos dessa árvore podem ser vistos como hiperplanos que dividem o espaço em dois subespaços. Por exemplo, considere um hiperplano H no espaço $\mathbb{R}^2$ que divide a dimensão x em um certo ponto $p = (a,b)$. Como a divisão é feita no eixo x, o hiperplano h é perpendicular à dimensão x e paralelo às demais dimensões. Assim, todos os pontos que possuem um valor menor que a na dimensão x estarão à esquerda de h e os pontos que possuem um valor maior que a estarão à direita de h.

Em sua construção, as dimensões dos dados são consideradas uma por vez. Em uma iteração para a dimensão i, o conjunto de dados é ordenado de acordo com suas coordenadas i, e um nó interno da árvore é criado para o ponto médio de acordo com a ordenação, dividindo o espaço em dois subespaços. O processo é repetido de maneira recursiva para os subespaços obtidos, até que um dado limite do número de pontos por folha da árvore seja alcançado.

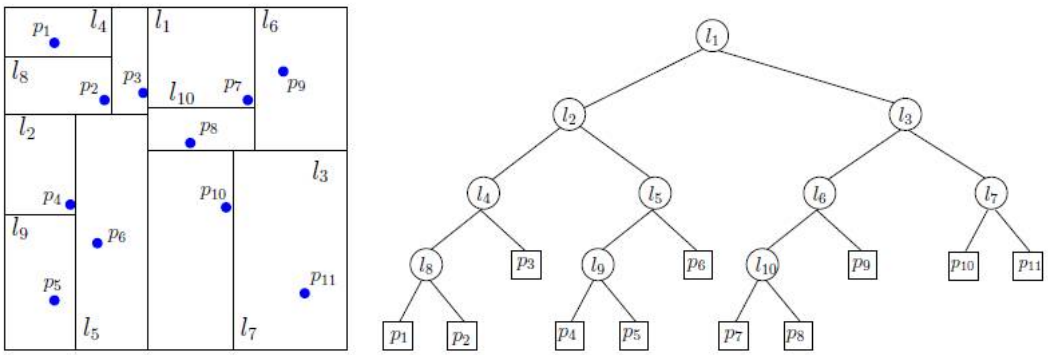


Figura 2.2: Exemplo de *k*-d-Tree.
Fonte: École Polytechnique - INF562 - TD 3

A Figura 2.2 exemplifica a indexação de um conjunto de dados em uma *k*-d-Tree. Os pontos azuis representam as entradas do conjunto de dados que está sendo indexado. Os nós internos da árvore, representados pela letra l, podem ser visualizados na disposição do espaço. Podemos ver que a raiz l_1 representa um hiperplano que divide todo o espaço em dois, considerando o ponto médio p_3 na dimensão x.

2.4 MapReduce

Para gerenciar, processar e analisar de maneira eficiente grandes volumes de dados, diversas soluções tem sido propostas, incluindo a migração ou mesmo desenvolvimento de aplicações no ambiente de Computação nas Nuvens, bem como sistemas baseados em *Distributed Hash Table* (DHT) ou *arrays* multidimensionais (SOUSA et al., 2010). Dentre essas, o paradigma MapReduce tem se destacado como uma das abordagens mais utilizadas para abordar problemas do gênero, uma vez que foi concebido com o objetivo de tornar simples e eficiente o processamento distribuído através de um conjunto de computadores comuns.

O modelo de programação do paradigma MapReduce é baseado em duas primitivas de programação funcional: Map e Reduce. Ambas as funções são definidas pelo desenvolvedor da aplicação, e são executadas da seguinte maneira: Primeiramente, a função Map recebe uma lista de pares chave-valor da forma (K_1, V_1) como entrada e, após sua execução, retorna um conjunto de pares chave-valor intermediário da forma (K_2, V_2) como saída. O conjunto intermediário é gerado de acordo com a função Map definida. Finalizada a fase de aplicação da função Map, a função Reduce receberá como entrada os pares intermediários gerados, de forma que esses estarão agrupados de acordo com suas chaves K_2. Esse agrupamento é feito pela fase de *Shuffle*, que envia todos os elementos com uma certa chave para ser processado juntamente com os outros que possuem a mesma chave. Finalmente, ao ser aplicada, a função Reduce retorna o resultado total do processamento, também na forma de pares chave-valor (K_3, V_3). É importante notar que toda a execução é feita de maneira distribuída.

Para ilustrar como um processo MapReduce funciona, considere um conjunto de cartas de três cores diferentes e que todas elas estão misturadas, como mostrado na Figura 2.3. Separá-las por cor demandaria uma quantidade de tempo para uma só pessoa fazê-lo. Porém, o mesmo trabalho sendo feito por três pessoas ao invés de apenas uma se tornaria consideravelmente mais rápido e fácil para todos. O processo de divisão das cartas com cores variadas entre as três pessoas corresponde à etapa Map de um processo MapReduce. Após a divisão, cada pessoa fica responsável por um subconjunto de cartas para separá-las por cores, colocando as vermelhas em uma pilha, as amarelas em outra, e as azuis em uma terceira pilha, que corresponde à etapa shuffle da Figura 2.3. No momento em que cada indivíduo termina o processo de separar seus subconjuntos de cartas, temos parte do processo completo, restando apenas a junção das cartas de mesma cor que estão com pessoas diferentes. A partir desse momento, cada uma das três pessoas ficará responsável por uma das cores. Alguém receberia todas as cartas azuis dos demais, outro receberia todas as amarelas e assim por diante. Finalmente, todo o conjunto de cartas estaria separado por cores e todas as cartas de uma mesma cor estariam juntas, etapa Reduce da Figura 2.3.

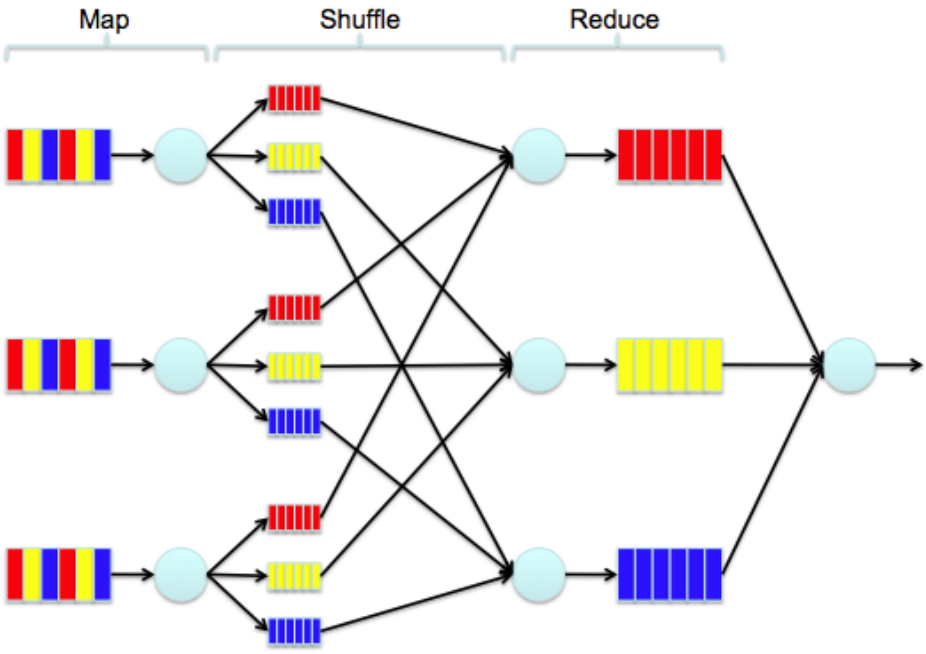


Figura 2.3: Exemplo de um processo MapReduce
Fonte: *Umass Lowel*

Um dos fatores que fazem do paradigma MapReduce atrativo é a sua capacidade de gerenciar grandes infraestruturas de recursos computacionais de maneira transparente para usuário. Assim, é possível que processamentos distribuídos relativamente sofisticados sejam criados e executados com a definição de apenas duas funções: Map e Reduce.

Dentre as implementações mais conhecidas do paradigma MapReduce destaca-se o framework de código aberto Hadoop (WHITE, 2009), desenvolvido pela Apache Software Foundation, que provê não só a execução de programas MapReduce mas também um sistema de arquivos distribuídos chamado HDFS (Hadoop File System). Ainda, a execução de tarefas nos nós de processamento e a tolerância a falhas são de total responsabilidade do framework, restando ao desenvolvedor apenas a programação das funções Map e Reduce, além da configuração da infraestrutura.

O framework Hadoop funciona em uma arquitetura mestre-escravo, onde o nó mestre tem como responsabilidade abrigar os serviços que gerenciam as tarefas executadas nos nós escravos, além de ser responsável pelo sistema de tolerância a falhas, reiniciando as tarefas que não puderam ser completadas. Tal arquitetura é facilmente implantada em um conjunto de máquinas, sejam físicas ou virtuais.

Os módulos que gerenciam o funcionamento do sistema total podem ser vistos na Figura 2.4, onde cada cliente submete uma aplicação para o módulo Gerenciador de Recursos, no nó mestre, e esse, por sua vez, solicita recursos para a execução dessa aplicação. Uma vez implantada, a aplicação se hospedará em um contêiner e terá um processo de gerenciamento próprio, que se comunicará com o gerenciador de recursos para informar o estado de seu processamento.

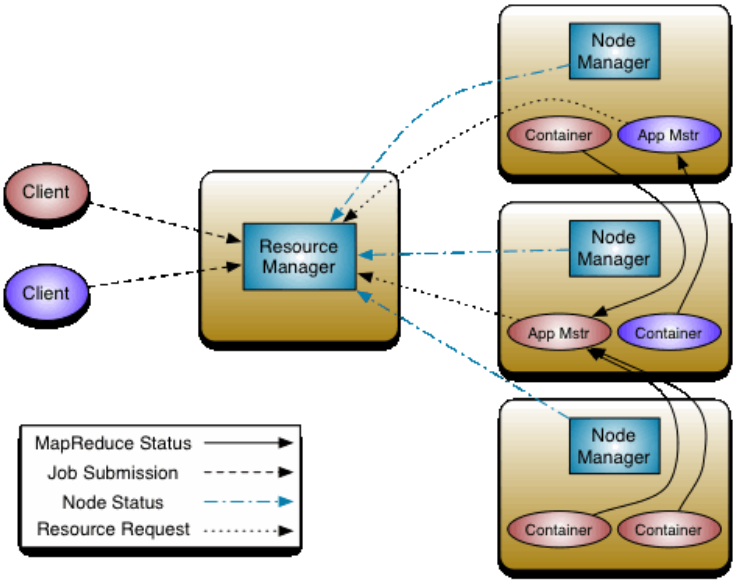


Figura 2.4: Arquitetura Hadoop YARN
Fonte: http://hadoop.apache.org/

2.5 Particionamento de Grafos

Nessa seção nós apresentamos algumas definições relacionadas ao problema de particionamento de grafos, que será utilizado como modelo para o problema de particionamento de dados abordado nesse trabalho. A seguir apresentamos conceitos e definições básicas do modelo de grafos utilizado e como ele é particionado. Tais noções são essenciais para o problema de particionamento estudado nessa dissertação.

Uma estrutura de grafo pode ser vista como a representação de um conjunto de objetos que se relacionam entre si. Em teoria dos grafos, esses objetos são chamados de vértices e os relacionamentos entre eles são chamados de arestas. Tanto os vértices quanto as arestas podem ter características que os definem, como pesos ou quaisquer anotações. Ainda, as arestas de um grafo podem ser direcionadas ou não direcionadas. Por exemplo, em um grafo direcionado, a existência de uma aresta que parte de um vértice v e chega em um vértice u não implica na existência de uma aresta que parte de u e chega em v.

A estrutura de grafo utilizada nessa dissertação é ponderada nos vértices e nas arestas e é denotada por $G(V,E)$, onde V corresponde ao conjunto de vértices e E ao conjunto de arestas. O peso de um vértice v_i é denotado por $\psi(v_i)$, e o peso de uma aresta (v_i,u_j) por $\omega(v_i,u_j)$.

Considerando $S = \{v_1,...,v_m\}$ como um subconjunto de vértices do grafo G, o peso de S é dado pela soma dos pesos dos vértices v_i pertencentes à S e é denotado por $\Psi(S). = \sum_{i=1}^{m} \psi(v_i)$. De maneira similar, sendo $R = \{(v_1,u_1),...,(v_n,u_n)\}$ um subconjunto de E, o peso de R é dado por $\Omega(R) = \sum_{i=1}^{n} \omega(v_i,u_i)$.

A partir desses conceitos, o problema de particionamento de grafos chamado k-partições, apresentado originalmente em (W; SHEN, 1971), é definido da seguinte maneira:

Definição 1. (Problema *k*-partições) *Um k-particionamento de um grafo G(V,E) é uma divisão do conjunto de vértices V em k subconjuntos disjuntos S_1, S_2, ..., S_k, de maneira que o peso desses conjuntos S_i são aproximadamente iguais e o peso do conjunto de corte é mínimo.*

Definição 2. (Conjunto de corte) *Conjunto de arestas (u,v) tal que $v \in S_i$ e $u \in S_j$, com $i \neq j$. Esse conjunto contém as arestas que possuem cada uma de suas extremidades em partições diferentes.*

Dentre as soluções existentes na literatura para o problema *k*-partições, em (KARYPIS; KUMAR, 1998) os autores apresentam uma estratégia que inicialmente compacta o grafo original e, a partir desse grafo reduzido, divide o conjunto de vértices em *k* subconjuntos. Esse particionamento inicial é posteriormente generalizado para o grafo original em diversas etapas, sendo refinado em cada uma delas. Esse processo apresenta complexidade linear no número de arestas do grafo original, ou seja, $\mathcal{O}(|E|)$.

A Tabela 2.1 resume os conceitos e as notações explicadas nessa seção e que serão utilizadas na descrição da abordagem proposta.

Notação	**Significado**
G	Grafo $G(V,E)$
$\psi(v_i)$	Peso do vértice $v_i \in V$
$\Psi(S)$	Peso do conjunto de vértices $S \subseteq V$
$\omega(v_i,u_j)$	Peso da aresta $(v_i,u_j) \in E$
$\Omega(R)$	Peso do conjunto de arestas $R \subseteq E$

Tabela 2.1: Notações

2.6 Conclusão

Nesse capítulo nós definimos formalmente os conceitos e definições básicas necessários para a compreensão do trabalho proposto e que serão úteis na formalização do problema nos próximos capítulos. Além do algoritmo DBSCAN, que é um dos objetos de estudo dessa dissertação, apresentamos também uma visão geral sobre agrupamento e sobre o paradigma MapReduce, que será utilizado como ferramenta para a distribuição do processamento do DBSCAN. E por último, definimos o modelo de grafo utilizado para modelar o problema de particionamento de dados também estudado nesse trabalho.

3 G2P-DBSCAN: PARTICIONAMENTO E DISTRIBUIÇÃO

3.1 Introdução

Recentemente, o aumento na quantidade de dados gerados por dispositivos móveis gerou um crescimento da necessidade de gerar informações a partir desses dados. Para isso, se tem investido em técnicas que permitem que análises de dados sejam feitas de maneira eficiente com a infraestrutura e tecnologia disponíveis. Aplicações que envolvem um grande número de usuários em diversas localizações, como redes sociais, serviços de navegação ou até mesmo processamentos em tempo real que auxiliam na tomada de decisão no gerenciamento do tráfego urbano, requerem alternativas eficientes que sejam capazes de computar resultados suficientemente precisos em tempo hábil.

Nesse capítulo nós propomos G2P-DBSCAN, uma estratégia para computar o algoritmo de agrupamento baseado em densidade DBSCAN através do paradigma MapReduce. É sabido que em processamentos distribuídos o tempo total de processamento é definido pelo nó de computação que leva mais tempo para terminar sua execução. Assim é importante que os nós de computação processem volumes de dados aproximadamente do mesmo tamanho. Visto isso, propomos também uma estratégia de particionamento de dados que atua como uma fase de pré-processamento. Essa fase leva em consideração as características do algoritmo DBSCAN e dos dados a serem processados, de forma a auxiliar no processo de agrupamento distribuída.

3.2 G2P-DBSCAN: Visão Geral

Nesta seção nós apresentamos uma visão geral da nossa estratégia de processamento do algoritmo DBSCAN distribuído. G2P-DBSCAN consiste em duas fases principais: 1) G2P (*Grid and Graph Partitioning*), uma estratégia de particionamento de dados; 2) Estratégia de processamento distribuído do algoritmo DBSCAN através do paradigma MapReduce. Cada uma dessas fases é composta de subestágios.

A Figura 3.1 mostra os estágios da nossa estratégia de particionamento G2P. Primeiramente os dados são dispostos em uma estrutura de grade, que é criada para representar de maneira discretizada os elementos do conjunto de dados dispostos no espaço. Em seguida, essa estrutura é transformada em um grafo que será particionado pela uma ferramenta de particionamento de grafos METIS (KARYPIS; KUMAR, 1998). Dessa forma, os estágios do G2P consistem no processo de construção dessas duas estruturas de dados, bem como no particionamento do grafo em si.

A Figura 3.2 mostra o fluxo do processamento da estratégia de DBSCAN distribuído. Um processo MapReduce é executado para agrupar, individualmente, cada uma das partições encontradas previamente pelo G2P. A partir daí, um estágio intermediário verifica quais, dentre os *clusters* encontrados, são candidatos a se tornarem um só. Tal passo é necessário para corrigir possíveis *clusters* que foram separados em partições diferentes. Por último, um outro trabalho MapReduce é lançado para checar se os candidatos achados no passo anterior

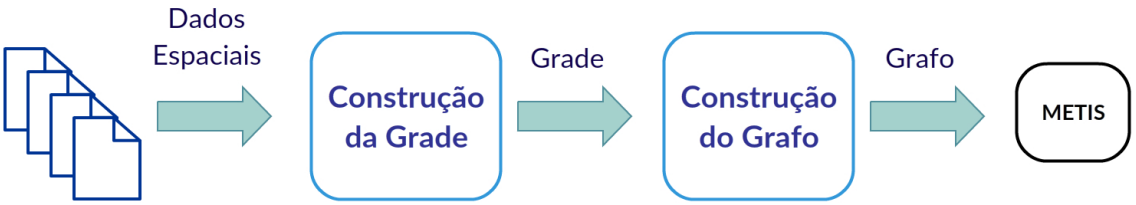


Figura 3.1: G2P (*Grid and Graph Partitioning*)

são válidos e, em caso positivo, renomeá-los para que, globalmente, sejam tratados como um só *cluster*.

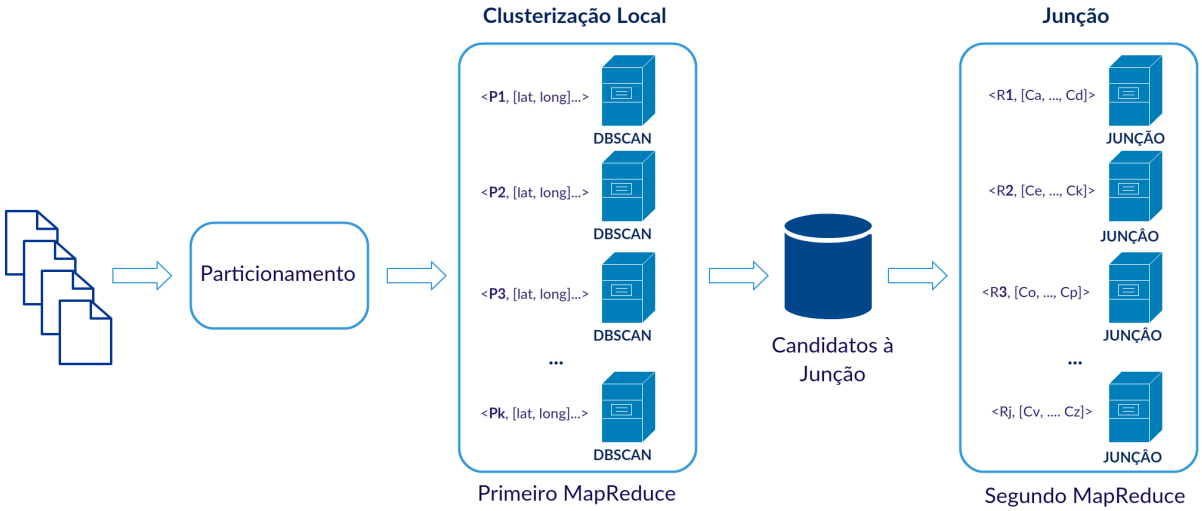


Figura 3.2: G2P-DBSCAN: Visão geral

Cada uma das etapas principais do G2P-DBSCAN e os seus subestágios mencionados acima são explicados em detalhes nas próximas seções.

3.3 G2P: *Grid and Graph Partitioning*

Grafos são amplamente utilizados para modelar problemas de diversos domínios, principalmente pela existência de técnicas e algoritmos bem consolidados que podem ser aplicados a eles, tais como coloração, descoberta de fluxos e particionamento. Dentre suas possíveis aplicações podemos citar processamentos paralelos, consultas em redes de ruas, processamento de imagens, redes sociais e bioinformática (BULUÇ et al., 2013). É comum que esses problemas sejam modelados como grafos, o que torna possível que eles sejam resolvidos com algoritmos para grafos, e que depois sejam traduzidos de volta para sua versão original.

A ideia da estratégia G2P consiste em transformar o problema de particionamento de dados em um problema de particionamento de grafos. Assim, consideramos a definição do nosso problema como descrita a seguir.

Definição do problema: Dado um conjunto de dados $\mathscr{D}$ de d dimensões e um

conjunto de k máquinas virtuais, o problema consiste em dividir $\mathscr{D}$ em k partições disjuntas que satisfazem as seguintes propriedades:

1. As partições devem ter aproximadamente a mesma quantidade de elementos. Considerando que em um ambiente distribuído o tempo total de processamento é diretamente proporcional ao processamento no nó de computação mais demorado, ou seja, o que tem uma maior quantidade de dados para processar, é importante que os tamanhos das partições sejam aproximadamente iguais.

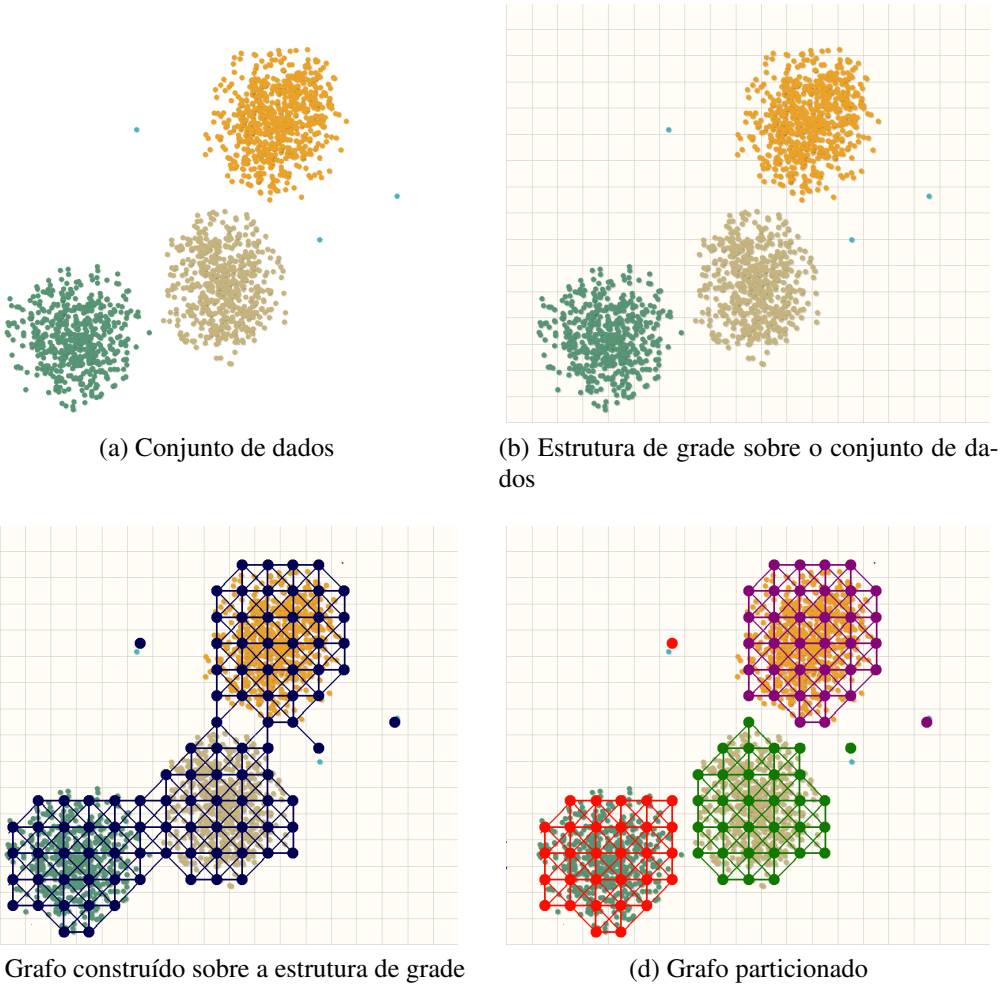
2. A divisão do espaço deverá ser feita em regiões onde a concentração de pontos é mais baixa. Visto que o algoritmo DBSCAN procura por concentrações que satisfazem um certo critério de densidade, é interessante que essas concentrações não sejam divididas em partições distintas.

O problema de particionamento de dados abordado neste trabalho requer que a distribuição dos dados e a densidade de diferentes regiões do espaço sejam consideradas como critério de decisão de como o particionamento será realizado. A partir da distribuição dos dados a estratégia de particionamento é capaz de balancear a quantidade de elementos presentes em cada uma das partições geradas. Já a informação sobre as densidades das diferentes regiões do espaço permite que regiões com uma grande concentração de elementos não sejam divididas em partições distintas.

A forma de modelar o problema apresentado utilizando uma estrutura de grafos é fundamental no processo de particionamento. A construção do grafo precisa considerar a maneira como o mesmo será particionado, de forma que as partições resultantes atendam os requisitos do problema de particionamento de dados para o DBSCAN distribuído. Para satisfazer as duas propriedades enumeradas acima é preciso interpretá-las em termos do problema de particionamento de grafos. A primeira delas diz respeito ao número de elementos em cada partição. A solução para o particionamento de grafos retorna partições cuja soma dos pesos dos vértices é aproximadamente igual. Assim, é preciso que a quantidade de elementos em uma partição seja de alguma forma descrita pelos pesos dos vértices. A segunda propriedade afirma que áreas com grandes concentrações de pontos não devem ser separadas em partições distintas. Na geração das partições no grafo, a solução tenta minimizar a soma do peso das arestas que serão cortadas, isto é, aquelas que terão suas extremidades em partições diferentes. Dessa forma, a quantidade de pontos em uma certa região precisa ser representada como peso de uma aresta do grafo.

As principais vantagens da estratégia G2P são a capacidade de cortar o espaço em múltiplas direções, além de horizontal e vertical, e o balanceamento de carga entre as partições obtidas. Além disso, é possível, a partir das propriedades do problema de particionamento de grafos, gerar partições apropriadas para o processamento do DBSCAN de maneira distribuída, de forma que uma região de alta densidade esteja inteiramente contida em apenas uma partição. A importância dessa propriedade decorre do objetivo do algoritmo DBSCAN, que consiste de encontrar concentrações de pontos que satisfaçam uma certa densidade.

Um exemplo do processo de particionamento pode ser visto na Figura 3.3. Partindo de um conjunto de dados espaciais (Figura 3.3a) uma estrutura de grade é construída sobre esses dados. Em seguida, cada célula da grade se torna um nó do grafo. No último passo (Figura 3.3d) o grafo é particionado, refletindo em partições do conjunto de dados.



(a) Conjunto de dados

(b) Estrutura de grade sobre o conjunto de dados

(c) Grafo construído sobre a estrutura de grade

(d) Grafo particionado

Figura 3.3: Processo de particionamento

A estratégia de particionamento G2P pode ser dividida nos seguintes estágios:

- **Estágio I:** Transformar a entrada do problema de particionamento de dados para o processamento distribuído do algoritmo DBSCAN em uma entrada para o problema de particionamento de grafos. Esse estágio é detalhado na Seção 3.3.1.
- **Estágio II:** Transformar as partições do grafo em partições do conjunto de dados de entrada, de forma que os dados de uma mesma partição possam ser processados de maneira isolada das demais partições. Esse estágio é apresentado na Seção 3.3.2.

3.3.1 Estágio I

Para a criação do grafo G, o conjunto de dados precisa ser representado de uma maneira na qual os pontos, ou mesmo grupos de pontos, possam ser vistos como nós de um grafo e que tenham algum relacionamento uns com os outros. A escolha da estrutura de grade para esse mapeamento entre os dados e o grafo se deu principalmente pelo fato de uma grade conseguir representar, de acordo com a granularidade desejada, características de regiões do espaço, tal como o número de pontos em cada célula, o que é uma ferramenta útil no momento de escolher em que região do espaço os dados serão divididos em partições diferentes. Através dessa estrutura é possível descrever as propriedades definidas no problema de particionamento de dados em termos de uma estrutura de grafo.

3.3.1.1 Construção da Grade

A construção da grade consiste em, para cada ponto, alocá-lo em uma célula de acordo com a sua posição no espaço. Para isso, assumimos um sistema de coordenadas para ser utilizado como referência para que os pontos sejam posicionados nas diferentes células. Cada célula da grade abrange uma região do espaço e os pontos nessa região. Assim, afirmar que um ponto p pertence à uma determinada célula C significa que a localização do ponto p é coberta pela célula C.

Além disso, para qualquer par de células adjacentes, nós desejamos que a máxima distância possível entre quaisquer dois pontos que estejam contidos nessas células seja igual a eps. Essa definição auxilia na representação de regiões densas do espaço. Sabemos, pela definição do algoritmo DBSCAN, que os *clusters* são formados a partir de uma verificação da quantidade de elementos em um certo raio. Ao saber que existe uma grande quantidade de pontos em células adjacentes, podemos considerar essas células como uma região densa, onde quaisquer dois pontos estarão a no máximo eps unidades de distância e, assim, desejamos que essas células pertençam a uma mesma partição. Temos que a diagonal entre duas células adjacentes corresponde à máxima distância possível entre dois pontos que estejam contidos nessas células. Uma vez que essa distância é dada por eps, as laterais das células terão tamanho igual a $\frac{eps}{\sqrt{5}}$. Note que a qualidade do balanceamento entre as partições depende do tamanho da célula da grade. Quanto menor a célula, melhor a discretização do espaço e também a representação do conjunto de dados.

Também, podemos ver que se a quantidade de pontos de duas células adjacentes juntas é maior que $minPoints$, os pontos de ambas as células pertencem ao mesmo *cluster* e, além disso, todos eles se classificam como *core points*, uma vez que a partir de um raio eps, cada um deles consegue alcançar pelo menos $minPoints$ elementos. Essas características na construção da grade impactam diretamente no particionamento dos dados, uma vez que com tal estrutura é possível representar as regiões densas que desejamos que não sejam divididas em partições distintas.

Consideremos o ponto $o = (0,0)$ como a origem do sistema de coordenadas e $p =$

(x,y) um ponto do conjunto de dados. A célula $C_{i,j}$ à qual o ponto p pertence é tal que:

$$i = \lceil \frac{(x)\sqrt{5}}{eps} \rceil \tag{3.1}$$

e

$$j = \lceil \frac{(y)\sqrt{5}}{eps} \rceil \tag{3.2}$$

Essas equações foram obtidas através do valor da diagonal traçada entre duas células adjacentes que definimos como sendo eps. Através de uma única varredura no conjunto de dados, para cada elemento calculamos à qual célula da grade tal elemento pertence. Esse procedimento é realizado em tempo linear com relação ao tamanho do conjunto de dados.

3.3.1.2 Construção do Grafo

Depois que a estrutura de grade é construída, o próximo passo corresponde à criação do grafo G que será passado como entrada para o algoritmo de particionamento de grafos. Como mencionado anteriormente, esse grafo precisa ser capaz de representar as regiões densas do espaço. Segundo as definições apresentadas no Capítulo 2, a solução para o problema de particionamento de grafos considera os pesos dos vértices para fins de balanceamento das partições e considera também o peso das arestas afim de dividir o grafo nas arestas com pesos menores. Considerando essas características, é preciso que regiões menos densas sejam representadas por arestas com peso mais baixo e que, consequentemente, regiões mais densas sejam representadas por arestas com peso maior.

Na estrutura de grade, o número de pontos pertencentes a uma célula é um indicador da densidade naquela região do espaço. Além disso, o somatório do número de pontos de células adjacentes também pode ser utilizado como indicativo para a densidade da região do espaço coberta por elas, uma vez que a maior distância possível entre quaisquer dois pontos dessas células é igual a eps. Dessa forma, no nosso modelo de grafo cada célula da grade corresponde a um vértice e uma aresta é criada para cada duas células adjacentes, tendo como peso a soma do número de pontos pertencentes a elas. Esse modelo permite descrever através das arestas do grafo a densidade de regiões do espaço. Ainda, o peso de cada vértice do grafo é o número de pontos contidos na célula que esse vértice representa. Desse modo, o balanceamento do peso das partições terá como consequência o balanceamento do número de pontos em cada partição.

Considerando $|C_{i,j}|$ como o número de pontos na célula $C_{i,j}$, o processo de construção do grafo $G(V,E)$ se dá através dos seguintes passos:

1. Para cada célula $C_{i,j}$ na grade contendo pelo menos um elemento do conjunto de dados, adicionar um vértice $v_{i,j}$ ao conjunto de vértices V e definir seu peso $\psi(v_{i,j}) = |C_{i,j}|$.

2. Para cada par de células adjacentes $C_{i,j}$ e $C_{k,l}$, onde $|i-k| = 1$ ou $|j-l| = 1$, adicionar

uma aresta $(v_{i,j}, v_{k,l})$ ao conjunto de arestas E e definir seu peso como $\omega(v_{i,j}, v_{k,l}) = \psi(v_{i,j}) + \psi(v_{k,l})$.

3.3.2 Estágio II

O problema de particionar grafos em k partições balanceadas é NP-Completo (GAREY; JOHNSON, 1990). Contudo, existem diversas ferramentas bem conhecidas que implementam soluções aproximativas para esse problema. Nesse trabalho utilizamos a ferramenta METIS (KARYPIS; KUMAR, 1998) para o particionamento do grafo gerado anteriormente. Como resultado teremos k conjuntos disjuntos de vértices de G, que chamamos de partições. A complexidade da estratégia G2P corresponde às complexidades de criação da estrutura de grade e do particionamento do grafo em si. Sabemos que para a construção da grade requer tempo linear. A complexidade do particionamento de grafos é de ordem linear com relação ao número de arestas $|E|$ do grafo a ser particionado. Como o grafo é construído a partir das células da grade, temos que $|E| << n$. Assim, podemos afirmar que a complexidade total do G2P é da ordem de $\mathcal{O}(n)$.

Nossa estratégia provê partições balanceadas porque a ferramenta METIS garante que tal característica seja levada em consideração no particionamento. Além disso, nós evitamos que regiões de alta densidade sejam separadas em partições diferentes, já que no particionamento o peso das arestas que são cortadas é minimizado e no nosso modelo de grafo o peso de uma aresta (v_i, u_j) corresponde à densidade da região coberta pelas células da grade representadas pelos vértices u_j e v_i. Isso previne que um *cluster* esteja contido em mais de uma partição e, por consequência, diminui a quantidade de junções que precisam ser feitas no DBSCAN distribuído, que será explicado na Seção 7. Dessa forma, nossa estratégia de particionamento obedece aos critérios enumerados anteriormente. A seguir, nós apresentamos a estratégia de DBSCAN distribuído.

3.4 G2P-DBSCAN distribuído

O processamento do algoritmo DBSCAN de maneira distribuída tem como objetivo tornar possível o agrupamento de grandes volumes de dados de maneira eficiente, obtendo os mesmos resultados que uma execução centralizada do algoritmo. Como mencionado anteriormente, o processamento distribuído de um algoritmo requer que a estratégia de particionamento leve em consideração o balanceamento de carga e as restrições impostas pelo algoritmo para dividir os dados em partições. No caso do algoritmo DBSCAN, em um cenário ideal, nenhum *cluster* deveria ser dividido em mais de uma partição. Porém, é possível que essa situação aconteça, embora o algoritmo de particionamento tente evitá-la. Para o tratamento desses casos, é preciso que exista uma etapa no processamento que verifique se esse fenômeno ocorreu. Caso sejam identificadas situações como essas, uma outra fase de processamento é necessária para corrigi-las. Assim, a distribuição do algoritmo DBSCAN através do paradigma MapReduce é dividida nas seguintes etapas:

- Agrupamento Local
- Encontrar *clusters* que foram divididos durante o particionamento
- Junção dos *clusters* quebrados em partições diferentes

3.4.1 Agrupamento Local

Essa fase consiste em um processo MapReduce que realiza o processo de agrupamento nas partições. Como desejamos que cada partição seja processada de maneira isolada, as funções Map e Reduce precisam, a partir das partições encontradas pela estratégia G2P, organizar os dados de maneira que dados de uma mesma partição sejam processados juntos. As funções Map e Reduce definidas são explicadas a seguir:

1. **Map** Cada ponto do conjunto de dados é descrito como um par $\langle chave, valor \rangle$, onde a *chave* corresponde à partição à qual aquele ponto pertence e $valor$ corresponde ao próprio ponto.
2. **Reduce** Essa função recebe como entrada um conjunto de pares que tem a mesma chave, ou seja, pontos que pertencem a uma mesma partição. A partir disso o algoritmo DBSCAN é aplicado nas partições com o auxílio de uma estrutura de índice k-d-Tree construída para cada partição.

Os Algoritmos 3 e 4 mostram como é feita a implementação das funções Map e Reduce descritas acima, respectivamente. Note que o conjunto de dados processado pelo algoritmo DBSCAN no Algoritmo 4 é o conjunto de pontos P de uma mesma partição.

Algoritmo 3: Agrupamento local - Map

```
Entrada: Conjunto de dados 𝒟
1 Início
2 |  para p ∈ 𝒟 faça
3 |  |  emitir⟨p.particao, (p.Lat, p.Lon)⟩
4 |  fim
5 fim
```

Algoritmo 4: Agrupamento local - Reduce

```
Entrada: Conjunto P de pares ⟨c, v⟩ com mesmo c, minPoints, eps
1 Início
2 |  DBSCAN(P, eps, minPoints)
3 |  Armazena resultados em uma base de dados;
4 fim
```

3.4.2 Encontrar *clusters* candidatos a junção

Essa fase tem como objetivo encontrar *clusters* que podem ter sido quebrados em diferentes partições após a execução da estratégia de particionamento G2P. O Algoritmo 5 mostra como essa verificação é realizada. Para cada *cluster* encontrado, sua forma geométrica convexa é gerada e armazenada, como mostrado nas linhas 2 e 3. Em seguida, nós verificamos as distâncias entre todos os pares de geometria. Se a distância entre duas geometrias G_i e G_j não ultrapassar o valor de *eps*, verificado na linha 6 do algoritmo, o par $\langle C_i, C_j \rangle$ é inserido no conjunto *PC* de *clusters* candidatos a junção. Nos casos em que distância entre dois *clusters* é menor que *eps* há uma grande probabilidade de que esses *clusters* sejam, na verdade, um só, porém dividido pelo particionamento. Duas formas geométricas estarem a uma distância menor que *eps* significa que existe pelo menos um ponto de cada forma geométrica que se alcançam com uma distância menor que *eps*. No algoritmo DBSCAN, isso poderia resultar no agrupamento de todos os pontos que formam as duas formas geométricas como um só *cluster*.

3.4.3 Junção

A fase de junção tem por objetivo verificar e corrigir *clusters* que foram quebrados durante a fase de particionamento. Esse processo acontece através de um trabalho MapReduce, onde os elementos dos *clusters* candidatos são dados como entrada para a verificação. Em grandes conjuntos de dados é possível que vários pares de candidatos sejam encontrados na etapa anterior. Assim, por razões de desempenho, a verificação dos pares é realizada em diferentes nós de computação. As funções Map e Reduce dessa fase são explicadas a seguir:

1. **Map**. A função Map na fase de junção corresponde à função identidade. Os pares $< id, \langle C_i, C_j \rangle >$ de entrada são passados diretamente para a função Reduce, onde $\langle C_i, C_j \rangle$ são *clusters* candidatos a junção e *id* é o menor entre os identificadores desses *clusters*.

2. **Reduce**. Essa função recebe como entrada os mesmos pares passados para a função Map.

Algoritmo 5: Encontrar *clusters* candidatos a se juntar

```
   Entrada: Conjunto de clusters C
   Saída: Conjunto PC de pares de clusters candidatos a junção
 1 Início
 2     para C_i ∈ C faça
 3         Criar a geometria de G_i;
 4     fim
 5     para todas as geometrias G_i e G_j e i <> j faça
 6         se Distancia(G_i, G_j) ≤ eps então
 7             PC ← ⟨C_i, C_j⟩
 8         fim
 9     fim
10     retorna PC;
11 fim
```

Para cada par de candidatos $\langle C_i, C_j \rangle$ é verificado se eles podem de fato se fundir, como mostrado no Algoritmo 6. Caso positivo, esses dois *clusters* são renomeados recebendo como identificador o valor de *id* para que eles passem a representar o mesmo *cluster*, como mostrado na linha 4 do algoritmo.

Algoritmo 6: Segundo MapReduce - REDUCE

```
   Entrada: Conjunto CPID de pares < id, ⟨C_i, C_j⟩ > com clusters candidatos
            a junção
1  Início
2      para ⟨C_i, C_j⟩ ∈ CPID faça
3          se PodeFundir(C_i, C_j) então
4              Renomear C_j e C_i com id em CPID
5          fim
6      fim
7  fim
```

O Algoritmo 7 mostra como é verificado se a junção de dois *clusters* C_i e C_j pode ser realizada. Entre as linhas 4 e 6 é analisado, para cada par de pontos p_i e p_j tal que $p_i \in C_i$ e $p_j \in C_j$, se p_i pertence à vizinhança de p_j. Em caso afirmativo, p_i e p_j são estabelecidos como vizinhos. Essa verificação é realizada com o objetivo de identificar os pontos que estão a uma distância de até *eps* entre os dois *clusters*. Entre as linhas 7 e 12 o algoritmo verifica se algum ponto $p_i \in C_i$ ou $p_j \in C_j$ se tornou *core point*. Essa fase é importante porque C_i e C_j podem se fundir em um só *cluster* se um *core point* $p_i \in C_i$ e outro *core point* $p_j \in C_j$ são alcançáveis por densidade, como podemos ver nas linhas de 12 a 15 do Algoritmo 7. Se a junção entre os dois *clusters* tiver sido realizada, o *cluster* de todo ponto $p \in C_i \cup C_j$ é atualizado para o cluster com menor identificador *i*, como mostrado nas linhas 20 e 21.

Essa estratégia considera a possibilidade de um ponto ruído se tornar *border point* ou *core point* após a junção, diferentemente dos trabalhos relacionados. Essa verificação é feita na linha 23 do algoritmo 7, com a chamada do procedimento **atualizarRuídos()**. Essa função percorre o conjunto de pontos ruídos das partições às quais os *clusters* C_i e C_j pertencem. A vizinhança de cada um desses pontos é verificada e, de acordo com o número de vizinhos, sua classificação pode ser atualizada. Caso um ruído p possua mais que $minPoints$ vizinhos, ele se torna um *core point*. Caso contrário, se houver pelo menos um *core point* na sua vizinhança, o ponto p transforma-se em um *border point*.

3.5 Validação das Junções Realizadas

A seguir nós mostramos que o processamento da junção de *clusters* que foram separados pelo particionamento é correto e produz resultados idênticos ao resultado do agrupamento feita pelo algoritmo de maneira centralizada.

Teorema 1. *Dois clusters C_1 e C_2, encontrados nas partições S_1 e S_2, respectivamente, devem se juntar caso existam dois pontos $p_1 \in C_1$ e $p_2 \in C_2$ que satisfazem as seguintes propriedades:*

Algoritmo 7: CanMerge

Entrada: *Clusters* C_i, C_j candidatos à junção

Início
  para $p_i \in C_i$ **faça**
    para $p_j \in C_j$ **faça**
      se $((p_i \in N_{eps}(p_j))$ **então**
        $adjacentes(p_i, p_j) \leftarrow verdadeiro$
      fim
      se $|N_{eps}(p_i)| \geq minPoints)$ **então**
        $p_i.core \leftarrow verdadeiro$
      fim
      se $|N_{eps}(p_j)| \geq minPoints)$ **então**
        $p_j.core \leftarrow verdadeiro$
      fim
      se $(adjacentes(p_i, p_j) \wedge p_i.core \wedge p_j.core)$ **então**
        $fundir \leftarrow true$
        $sairdolao$
      fim
    fim
  fim
  se $fundir = verdadeiro$ **então**
    para $p \in C_i \cup C_j$ **faça**
      $p.cluster \leftarrow i$
    fim
    atualizarRuidos();
  fim
fim

1. $distancia(p_1, p_2) \leq eps$;
2. $N_{eps}(p_1) \geq minPoints$ *em* $S_1 \cup S_2$;
3. $N_{eps}(p_2) \geq minPoints$ *em* $S_1 \cup S_2$;

Demonstração. Como p_1 e p_2 são *core points* em $S_1 \cup S_2$, conforme estabelecido pelas propriedades 2 e 3, e a distância entre eles é menor que *eps*, podemos afirmar que p_2 é alcançável por densidade (DR) a partir de p_1. Assim, pela propriedade da maximalidade do algoritmo DBSCAN, p_2 também pertence ao mesmo *cluster* que p_1. Pela propriedade da conectividade, quaisquer dois pontos que pertençam ao mesmo *cluster* são conectáveis por densidade. Como p_2 alcança todos os elementos de C_2, o ponto p_1 também alcança por densidade todos os elementos de C_2. Logo, C_1 e C_2 são, na verdade, o mesmo *cluster*.

□

3.6 Conclusão

Nós propomos G2P (*Grid and Graph Partitioning*), uma estratégia de particionamento de dados que considera não somente o balanceamento das partições mas também características dos dados que induzem um melhor particionamento para o algoritmo DBSCAN distribuído. Além disso, propomos também uma estratégia de DBSCAN distribuído que recebe partições de dados e computa, através do paradigma MapReduce, *clusters* baseados em densidade, idênticos aos que seriam encontrados em uma execução centralizada do algoritmo DBSCAN.

4 DBSCAN: ESTRATÉGIAS DE DISTRIBUIÇÃO

4.1 Introdução

Por ser bem conhecido e amplamente utilizado na comunidade, o algoritmo DBSCAN tem sido abordado por diversos trabalhos que se propõem a modificá-lo, seja para domínios de aplicações específicas (CHEN; JI; WANG, 2014), (LIU et al., 2013), (ZHANG; XU; SI, 2013), (BIRANT; KUT, 2007) ou para torná-lo mais escalável e eficiente (GAN; TAO, 2015), (WANG et al., 2015), (SCICLUNA; BOUGANIS, 2014), utilizando como justificativa a sua alta complexidade para a análise de grandes volumes de dados.

Embora o DBSCAN seja capaz de encontrar ruídos e *clusters* de formatos arbitrários no conjunto de dados, outros aspectos, como a capacidade de identificar *clusters* com densidades diferentes ou a visualização de hierarquias nos *clusters*, não são considerados na sua versão original. Ainda, a necessidade de considerar especificidades de diferentes domínios, como o de dados espaço temporais, motivaram o surgimento de novas soluções baseadas no algoritmo DBSCAN que se adequam melhor a esses domínios.

No que diz respeito ao desempenho, soluções centralizadas e distribuídas foram propostas com o objetivo de acelerar o tempo de execução do DBSCAN para análises de grandes quantidades de dados. Nas soluções distribuídas, estratégias de particionamento surgiram a partir da necessidade de dividir os dados em partições balanceadas para garantir a eficiência do processamento.

As estratégias centralizadas que dizem respeito a adaptações do algoritmo para domínios específicos ou mesmo para melhoria de desempenho são discutidas brevemente na Seção 4.2. Visto que a proposta apresentada nessa dissertação consiste em uma versão distribuída do DBSCAN, as estratégias que consideram o processamento distribuído do algoritmo são discutidas em detalhes na Seção 4.3.

4.2 Adaptações centralizadas do DBSCAN

Dentre os trabalhos que adaptam o DBSCAN para domínios específicos, o algoritmo C-DBSCAN (WANG et al., 2014) apresenta uma estratégia para encontrar pontos ideais para localização de paradas de ônibus em uma cidade, baseado em dados de trajetórias de GPSs de táxis. Essa localização é calculada a partir do centro geométrico dos *clusters* encontrados. Já em (CHEN; JI; WANG, 2014) os autores introduzem o algoritmo T-DBSCAN, que se propõe a considerar a dimensão temporal para agrupar dados de trajetórias.

Em (ANDRADE et al., 2013) os autores apresentam o G-DBSCAN, uma versão do algoritmo DBSCAN acelerada por GPU. A solução proposta é composta por dois estágios: 1) Construção de uma estrutura de grafo a partir dos pontos do conjunto de dados, onde cada vértice representa um ponto do conjunto e as arestas são criadas entre pares de pontos que estão a no máximo uma certa distância uns dos outros; 2) Percurso em profundidade do grafo

criado na primeira etapa para encontrar os *clusters*. Ambas as etapas são paralelizadas através de GPUs.

Algumas estratégias, tais como o OPTICS (ANKERST et al., 1999) e o DBCURE (KIM et al., 2014) são capazes de encontrar *clusters* com densidades diferentes em um mesmo conjunto de dados. O OPTICS (*Ordering Points to Identify the Clustering Structure*) consiste em uma versão hierárquica do algoritmo DBSCAN, onde o resultado do processamento é uma hierarquia de *clusters* baseada na distância em que uma certa quantidade de elementos do conjunto de dados é alcançada. Já o DBCURE é uma versão generalizada do DBSCAN, que considera vizinhanças elípticas ao invés de circulares. Os autores propõem ainda no mesmo trabalho sua versão paralela através do paradigma MapReduce, chamada DBCURE-MR.

Em (UNCU et al., 2006) os autores propõem o GRIDBSCAN, um método de agrupamento em três níveis. O primeiro nível seleciona estruturas de grade apropriadas de forma que a densidade seja aproximadamente homogênea em cada grade. No segundo nível as células com densidades similares são mescladas e valores adequados para os parâmetros eps e $minPoints$ são escolhidos para cada grade que restou depois da junção. O terceiro nível corresponde à aplicação do algoritmo DBSCAN no conjunto de dados com os parâmetros identificados. Porém, GRIDBSCAN não é adequado para grandes volumes de dados, visto que seu processamento ocorre todo de maneira centralizada.

4.3 Processamento distribuído

PDBSCAN

Em (XU; JÄGER; KRIEGEL, 1999) é apresentada uma estratégia paralela para computação do algoritmo DBSCAN através de uma arquitetura *shared-nothing*. A estratégia é composta de três passos principais. O primeiro deles consiste na divisão dos dados em partições e na distribuição dessas partições entre as máquinas da infraestrutura. A estratégia de particionamento adotada tem como objetivos alcançar um bom balanceamento de carga entre as partições, minimizar os custos de comunicação entre as máquinas envolvidas no processamento e permitir o acesso distribuído aos dados. O particionamento proposto se baseia na estrutura de índice R*-Tree (BECKMANN et al., 1990), onde suas folhas são divididas em N grupos, de forma que elementos próximos sejam mantidos na mesma partição. A complexidade dessa estratégia de particionamento, portanto, é da mesma ordem de construção de uma árvore R*-Tree ($\mathcal{O}(n)$), onde n corresponde ao número de objetos a serem inseridos na árvore. Além disso, existe ainda um custo computacional de ordem linear para garantir que os elementos espacialmente próximos estejam contidos na mesma partição.

O segundo passo corresponde ao agrupamento das partições através do algoritmo DBSCAN. Um processo mestre coordena o agrupamento nas diferentes máquinas e é responsável pela troca de mensagens entre elas. No terceiro e último passo, os resultados obtidos do agrupamento em cada partição são combinados para corrigir *clusters* que eventualmente foram divididos em partições diferentes no momento do particionamento. É importante ressaltar que essa etapa de junção ocorre de maneira centralizada na estratégia.

Esse trabalho consiste em uma das primeiras abordagens de processamento distribuído do algoritmo DBSCAN. Devido a ausência de ferramentas disponíveis que facilitassem a implementação distribuída de um algoritmo, os autores implementaram toda a estratégia de distribuição, desde a comunicação entre as máquinas até o acesso distribuído aos dados. Com a recente difusão de *frameworks* eficientes que auxiliam a criação de algoritmos paralelos e distribuídos, a reprodução dos resultados obtidos na estratégia PDBSCAN não é considerada pela comunidade para fins de comparação.

DBSCAN-MR

Em (DAI; LIN, 2012) foi proposta a estratégia DBSCAN-MR, que se trata de uma implementação do DBSCAN utilizando o paradigma MapReduce. Os autores também propuseram uma estratégia de particionamento a partir de uma estrutura de grade, chamada PRBP (Partitioning with Reduced Boundary Points). Contudo, o método PRBP só considera a divisão do espaço em cortes horizontais ou verticais, dificultando a tarefa de balancear a quantidade de pontos por partição. Além disso, dois parâmetros adicionais são requeridos pela estratégia e uma má escolha dos valores para esses parâmetros pode resultar em um particionamento ruim. O primeiro deles, chamado de β, define a quantidade máxima de pontos em uma partição. O segundo parâmetro, θ, representa o desbalanceamento aceitável pela estratégia. A complexidade da estratégia PRBP é da ordem de $\mathcal{O}(n+dn)$, onde d representa a dimensão dos dados e n representa o tamanho do conjunto de dados a ser particionado. Essa complexidade decorre do fato de todo o conjunto de dados precisar ser percorrido para a construção da estrutura de grade, e, também, pelo fato do processo de divisão dos dados em diferentes partições considerar, para cada dimensão, todos os elementos que estão contidos em uma certa porção da grade. No pior caso, essa quantidade corresponde ao conjunto de dados inteiro.

Os autores apresentaram também uma estratégia de junção de *clusters*, uma vez que esses podem ter sido divididos em partições diferentes. A junção é feita através da replicação dos pontos das bordas das partições, requerendo um aumento no espaço de armazenamento necessário.

O DBSCAN-MR é executado em cinco fases. A primeira delas consiste na estratégia de particionamento de dados PRBP que, através de uma estrutura de grade, distribui o conjunto de dados em partições distintas. A cada iteração o espaço é dividido em dois, recursivamente, na direção horizontal ou vertical até que cada partição contenha no máximo β pontos e que a diferença entre os tamanhos das partiçes não ultrapasse o valor do parâmetro θ. Esse tipo de divisão é chamada de BSP (*Binary Space Partitioning*). A divisão é feita buscando minimizar o número de pontos nas bordas das partições. É importante ressaltar que esses pontos são replicados nas partições adjacentes.

Na segunda fase um trabalho MapReduce é executado para agrupar cada partição individualmente de maneira distribuída, o que acontece na função Map. A terceira fase é executada na função Reduce, que recupera os pontos replicados nas partições e os IDs de *clusters* atribuídos a eles. A quarta fase verifica se esses pontos foram agrupados de maneira semelhante nas partições adjacentes, isto é, se eles foram classificados como *core*, *border* ou *noise* em ambas as partições. Em caso positivo, a junção dos dois *clusters* deve ser realizada. A quinta e

última fase associa os elementos dos *clusters* unidos a um único *cluster* para que os mesmos sejam tratados globalmente como um só.

Dentre as desvantagens da estratégia DBSCAN-MR, está o fato do particionamento considerar apenas o método BSP com cortes horizontais ou verticais, que nem sempre é adequado quando *clusters* de formatos arbitrários são considerados. Ainda, o custo computacional necessário para a construção da estrutura de grade da estratégia representa grande parte do tempo gasto pelo particionamento, uma vez que requer que as entradas do conjunto de dados estejam ordenadas de acordo com suas coordenadas. Além disso, a estratégia de junção de *clusters* é executada de maneira centralizada e necessita que as bordas das partições sejam replicadas nas partições adjacentes.

MR-DBSCAN

Em (HE et al., 2011) os autores propuseram a solução MR-DBSCAN, que também utiliza o paradigma MapReduce, além de uma estratégia de particionamento chamada CBP (*Cost Based Partitioning*).

O MR-DBSCAN é executado em 3 etapas. A primeira delas corresponde à etapa de particionamento. De início, um trabalho MapReduce é executado para coletar estatísticas dos dados, ao mesmo tempo que constrói uma estrutura de grade que será utilizada pela função de particionamento. O método CBP considera o custo de recuperar a vizinhança de um ponto como critério de particionamento. Esse custo é definido em (THEODORIDIS; SELLIS, 1996) e estima o desempenho de uma consulta de vizinhança em uma estrutura de índices R-Tree (GUTTMAN, 1984). Essa função é aplicada a cada célula da estrutura de grade a partir das estatísticas coletadas no início do processo. O CBP particiona a grade através do método BSP e se propõe a produzir partições balanceadas em termos de custo, e não no número de pontos absoluto de cada uma. A complexidade da estratégia CBP é, no pior caso, da ordem de $\mathcal{O}(n+nlog(n))$. A porção linear dessa complexidade se corresponde à construção da estrutura de grade, que necessita que todo o conjunto de dados seja percorrido. Após isso, para cada partição, são analisadas as possibilidades onde essa pode ser dividida em duas partições diferentes. Como para cada partição essa divisão pode ocorrer em n localizações, e éssa análise é feita recursivamente para cada partição que é dividida, o pior caso dessa operação corresponde a um tempo da ordem de $\mathcal{O}(nlog(n))$

Na segunda etapa, o DBSCAN é aplicado em cada uma das partições através de uma execução MapReduce. A terceira e última etapa realiza as junções de *clusters* que foram eventualmente divididos em mais de uma partição e se dá por meio de dois trabalhos MapReduce. O primeiro deles identifica os *clusters* que foram divididos no processo de particionamento, e o segundo MapReduce os renomeiam para que sejam tratados globalmente como um só *cluster*. Essa junção é feita através das bordas das partições, que também são replicadas nas partições adjacentes, da mesma forma que em (DAI; LIN, 2012).

Embora a estratégia de particionamento CBP priorize o balanceamento dos custos das partições, o uso do método BSP, que corta o espaço apenas horizontalmente ou verticalmente, limita as possibilidades de divisão.

Mr. Scan

Já em (WELTON; SAMANAS; MILLER, 2013) os autores apresentaram uma estratégia híbrida de computação distribuída do algoritmo DBSCAN, que utiliza uma rede MRNet (*Multicast/Reduction Network*) (ROTH; ARNOLD; MILLER, 2003) para distribuição e *hardware* de alto desempenho, como GPGPUs. A estratégia Mr. Scan é composta de 4 fases. A primeira fase corresponde à fase de particionamento, que também é feita através de uma estrutura de grade. As células da grade são percorridas ao longo do eixo y e x, respectivamente, e são adicionadas a partição corrente até que o tamanho da mesma atinja um limite estabelecido. A complexidade dessa estratégia é da ordem de $\mathcal{O}(n)$. Porém, as partições resultantes desse processo só levam em consideração o balanceamento de carga, mas não consideram a distribuição dos dados. Na segunda fase, o agrupamento é feita em cada uma das partições. A

execução do DBSCAN é feita através de uma implementação *multi-thread* que executa em uma GPGPU. Ainda, uma alteração no algoritmo permite identificar áreas de densidade apenas através da grade construída, reduzindo o número de pontos que serão percorridos pelo DBSCAN. A terceira fase realiza a junção dos *clusters* que foram quebrados em mais de uma partição. Essa junção se dá por meio da replicação das bordas das partições. A estratégia seleciona uma certa quantidade de pontos de cada *cluster* que são capazes de definir se dois clusters devem ou não sofrer junção. A quarta e última etapa tem como funcionalidade a escrita do resultado d agrupamento no sistema de arquivos.

A infraestrutura na qual os experimentos foram executados contava com mais de 18 mil nós de processamento, tendo 32GB de memória RAM e uma placa de vídeo com 6GB por nó, tornando impraticável a reprodução dos resultados alcançados pelos autores. Embora o conjunto de dados dos experimentos fosse consideravelmente volumoso (aproximadamente 6 bilhões de pontos), a infraestrutura utilizada contribuiu de maneira significativa para a obtenção dos resultados apresentados pelos autores.

4.4 Análise comparativa dos trabalhos relacionados

Esta seção apresenta as principais características das soluções propostas para distribuição do algoritmo DBSCAN descritas acima. Dentre esses trabalhos, todos apresentam estratégias de particionamento de dados para o processamento distribuído. Um aspecto comum a essas estratégias é a utilização de estruturas de grade para representar o conjunto de dados. A partir daí, as técnicas de particionamento são aplicadas à essa estrutura. Dentre as técnicas apresentadas, a divisão binária do espaço em cortes horizontais ou verticais é considerada em duas das estratégias (DBSCAN-MR e MR-DBSCAN), tendo como diferença apenas o critério de decisão de onde o espaço será dividido. Na estratégia Mr.Scan, a estrutura de grade é percorrida e as células são acumuladas em uma partição até que um limiar do tamanho máximo da partição seja atingido. Por último, a estratégia PDBSCAN utiliza curvas de Hilbert como critério de agrupamento das células da grade. Embora as curvas de Hilbert apresentem propriedades adequadas para agrupamento (MOON et al., 2001), a distribuição dos dados nas células das grade não é levada em consideração pela estratégia. Já na divisão binária do espaço, o balanceamento das partições, seja por quantidade absoluta de pontos ou por custo de processamento, pode ser prejudicado pelos cortes apenas horizontais ou verticais. Nenhuma das estratégias de particionamento descritas considera a densidade do conjunto de dados como critério para dividir o espaço em partições, e todas replicam os pontos localizados nas bordas das partições.

Na fase de agrupamento individual das partições, também comum a todos os trabalhos, a replicação decorrente do particionamento pode degradar o desempenho da execução do DBSCAN nas partições, uma vez que a quantidade de pontos a serem agrupados é maior. Todos os trabalhos considerados também apresentam uma fase de junção de *clusters*, e se baseiam no agrupamento dos pontos das bordas das partições para decidir se os clusters serão unificados. Essa fase acontece de maneira centralizada em duas das estratégias (PDBSCAN e DBSCAN-MR) e de maneira distribuída nas demais (MR-DBSCAN e Mr. Scan).

A estratégia de particionamento da nossa proposta G2P-DBSCAN, chamada G2P, também se baseia em uma estrutura de grade. Contudo, essa grade é transformada em um grafo que reflete a densidade das células através dos pesos dos vértices e das arestas. O problema de particionamento dos dados se transforma então em um problema de particionamento de grafos. Essa estratégia permite que o espaço seja cortado em múltiplas direções, não se limitando apenas às direções vertical e horizontal, diferente das estratégias descritas acima. A partir das complexidades das estratégias acima e de suas características, é possível afirmar que a estratégia G2P apresenta vantagens relacionadas ao desempenho e à qualidade das partições produzidas. Além disso, o G2P-DBSCAN não requer que os dados das bordas das partições sejam replicados. Após o particionamento, é executado um processo MapReduce que realiza o agrupamento nas partições. A fase de junção de *clusters* também é realizada de forma distribuída através de outro processo MapReduce.

A Tabela 4.1 resume as principais características das estratégias de distribuição do algoritmo DBSCAN descritas acima em comparação com a estratégia G2P-DBSCAN.

	MapReduce	**Replicação**	**Tipo de particionamento**	**Junção**
PDBSCAN	Não	Sim	Curvas de Hilbert	Centralizada
DBSCAN-MR	Sim	Sim	BSP linear	Centralizada
MR-DBSCAN	Sim	Sim	BSP linear	Distribuída
Mr. Scan	Não	Sim	BSP linear	Distribuída
G2P-DBSCAN	Sim	Não	Particionamento de grafos	Distribuída

Tabela 4.1: Tabela comparativa das estratégias de distribuição do DBSCAN

4.5 Conclusão

Este capítulo apresentou os principais trabalhos relacionados ao problema de interesse desta dissertação. Esses trabalhos propõem extensões para o algoritmo DBSCAN e foram divididos em duas categorias: aqueles que propõem adaptações centralizadas para o algoritmo e os que apresentam soluções para o seu processamento distribuído. Uma vez que o foco desta dissertação é o processamento distribuído do DBSCAN, nós realizamos uma análise comparativa que mostrou as principais características da nossa proposta e das soluções apresentadas que se encaixam na mesma categoria.

5 ANÁLISE EXPERIMENTAL

5.1 Introdução

Este capítulo descreve o processo de análise experimental do algoritmo G2P-DBSCAN proposto neste trabalho utilizando conjuntos de dados reais. Os dados e a infraestutura utilizados são apresentados nas Seções 5.2 e 5.3, respectivamente.

Para a análise experimental da nossa proposta, diversos aspectos e métricas foram estudados e observados. Primeiramente analisamos o desempenho do G2P-DBSCAN em relação ao DBSCAN centralizado variando o tamanho do conjunto de dados. Em todos os experimentos onde o desempenho dos algoritmos foi analisado, foram realizadas 5 execuções e obtida a média dos tempos de processamento.

Em seguida, realizamos uma análise comparativa do G2P-DBSCAN e da estratégia DBSCAN-MR (DAI; LIN, 2012). O algoritmo DBSCAN-MR foi escolhido para comparação por ser mais semelhante à proposta deste trabalho. As duas soluções usam infraestruturas similares e possuem estratégias de particionamento centralizadas, o que possibilita a comparação direta da eficiência e da qualidade das mesmas. Inicialmente analisamos o tempo de execução e o desbalanceamento da nossa estratégia de particionamento G2P e da estratégia PRBP usada no algoritmo DBSCAN-MR variando o tamanho do conjunto de dados, *eps* e o número de partições k. Posteriormente, analisamos o tempo de execução total das duas soluções para as etapas de particionamento e processamento distribuído, variando o tamanho do conjunto de dados e o valor de *eps*.

5.2 Conjunto de Dados

Foram utilizados dois conjuntos de dados nos experimentos. O primeiro deles, que chamaremos de FOR, diz respeito a dados de tráfego urbano da cidade de Fortaleza. Os pontos foram coletados a partir de radares e dados de GPS no período de Janeiro a Agosto de 2013. Cada entrada do conjunto de dados corresponde ao ID do objeto móvel, o nome da via em que se encontra, sua posição geográfica e a hora exata em que tal ponto foi coletado. No total, o conjunto de dados FOR contém aproximadamente 1 milhão de pontos. Uma amostra desses dados pode ser visualizada na Figura 5.1, onde cada cor corresponde a uma via diferente da cidade. Podemos ver que há concentrações de dados em avenidas mais movimentadas onde as coletas foram feitas. A diferenciação das cores auxilia na identificação de vias diferentes que se cruzam em um ou mais pontos.

O outro conjunto de dados utilizado na experimentação, chamado de YFCC100M (THOMEE et al., 2015) é composto de metadados de fotos do Flickr de usuários do mundo inteiro, provido pelo *Yahoo! Webscope*. Contendo aproximadamente 100 milhões de entradas, dentre as quais 49 milhões possuem geolocalização, esse conjunto de dados possui informações tais como o nome do usuário, detalhes do dispositivo utilizado para capturar a foto, descrição e localização geográfica, além da URL para recuperação da imagem. Na figura 5.2 temos a visu-

Figura 5.1: Amostra do conjunto de dados FOR

alização de uma amostra do conjunto de dados YFCC100M. Cada ponto da figura representa a localização de uma foto da plataforma Flickr. É possível notar a diferença na concentração de fotos em certas regiões do mapa como na Europa e nos Estados Unidos quando comparados à África e à parte central do Brasil. Essas concentrações representam *clusters* dos mais variados níveis.

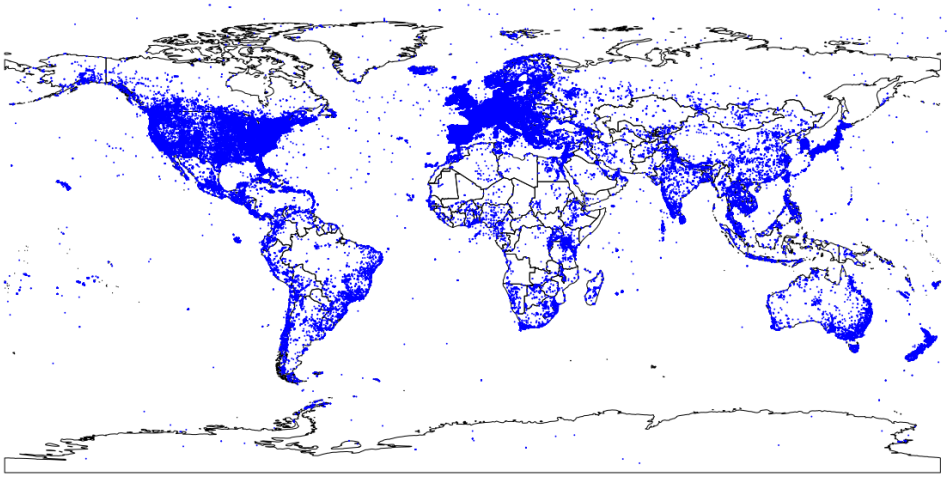


Figura 5.2: Amostra do conjunto de dados YFCC100M

Por razões de tempo de execução e de escalabilidade da versão centralizada do DBSCAN e da estratégia DBSCAN-MR, o número de elementos a serem processados na experimentação envolvendo ambas essas estratégias foi reduzido de acordo com suas limitações. Devido a isso, tais experimentos puderam ser realizados utilizando apenas amostras do conjunto FOR. Por outro lado, na análise das estratégias de particionamento (G2P e PRBP), a quantidade de elementos que ambas as estratégias são capazes de processar é significativamente maior, e,

particularmente, maior que o número de elementos existentes no conjunto de dados FOR. Portanto, na comparação das estratégias de particionamento serão utilizadas amostras do conjunto de dados YFCC100M.

5.3 Infraestrutura

Os experimentos foram executados em uma infraestrutura de nuvem privada na Universidade Federal do Ceará (UFC). No total foram utilizadas 16 máquinas virtuais, tendo como sistema operacional Ubuntu versão 12.04 e contando com 8GB de memória RAM e 4 núcleos de processamento cada. O ambiente contou também com o Hadoop 1.2.1, configurado conforme a Tabela 5.1. Dentre as máquinas virtuais instanciadas, 1 máquina atuou como nó mestre do Hadoop e as outras 15 máquinas como escravos.

Variável de Configuração	Valor
hadoop.tmp.dir	/tmp/hadoop
fs.default.name	hdfs://master:54310
mapred.job.tracker	master:54311
mapreduce.task.timeout	36000000
mapred.child.java.opts	-Xmx8192m
mapred.reduce.tasks	11
dfs.replication	5

Tabela 5.1: Variáveis de configuração do Hadoop definidas nos experimentos.

5.4 G2P-DBSCAN x DBSCAN

No experimento G2P-DBSCAN x DBSCAN realizamos uma comparação do desempenho da estratégia distribuída G2P-DBSCAN proposta nesse trabalho com a versão centralizada do algoritmo DBSCAN. Espera-se que a estratégia G2P-DBSCAN, além de apresentar um melhor desempenho, se mostre adequada para cenários onde o volume de dados a ser analisado aumenta. Para isso foram realizados experimentos com conjuntos de dados de tamanhos diferentes, tornando possível observar o comportamento das estratégias à medida que o volume de dados a ser processado cresce. Utilizamos amostras do conjunto FOR, escolhidas aleatoriamente, e os valores de eps e $minPoints$ foram fixados em 100 e 50, respectivamente. Essa escolha foi feita através de uma série de execuções do algoritmo DBSCAN com valores diferentes para os parâmetros em questão. Dentre essas execuções, foram selecionados os valores que resultaram em melhores agrupamentos.

A tabela 5.2 apresenta os resultados obtidos em segundos. Na infraestrutura utilizada, a versão centralizada do DBSCAN executou sem erros apenas pouco mais de 500 mil pontos. Como esperado, a versão distribuída do algoritmo DBSCAN teve um melhor desempenho, retornando os mesmos resultados de agrupamento que a versão centralizada de maneira bem mais eficiente. O tempo de execução do G2P-DBSCAN foi de até 92% menor que o da

versão original do algoritmo, quando um conjunto de dados de 510 mil pontos foi submetido às duas estratégias.

Tamanho do conjunto de dados	G2P-DBSCAN	DBSCAN
250.000	197	1150
325.000	254	2002
430.000	330	3530
510.000	409	4965

Tabela 5.2: Comparação dos tempos de execução do G2P-DBSCAN e do DBSCAN centralizado (em segundos)

5.4.1 G2P x PRBP

Neste experimento analisamos o desempenho e o desbalanceamento da nossa estratégia de particionamento G2P e da estratégia PRBP usada no algoritmo DBSCAN-MR. Nós definimos como desbalanceamento a diferença entre o tamanho da maior partição encontrada por cada estratégia e o tamanho médio das demais partições. Uma vez que em um processamento distribuído o tempo total de computação é diretamente atrelado ao nó de computação onde o processamento é mais demorado, um maior desbalanceamento implica em um maior tempo total de computação.

Como no experimento anterior, nós variamos o tamanho dos conjuntos de dados para analisar a escalabilidade das duas estratégias. Visto que o parâmetro de entrada eps define o tamanho das células da grade em ambas as estratégias de particionamento analisadas, nós também analisamos diferentes cenários com diversos valores de eps para mensurar a influência do mesmo no balanceamento.

Para o G2P, a quantidade k de partições foi variada entre 2 e 24. Assim, para um certo valor de k, o tamanho ideal de partição é igual ao tamanho do conjunto de dados dividido pelo número de partições. Esse valor é então utilizado para mensurar o desbalanceamento de carga, isto é, o quão maior ou menor cada partição é do que o tamanho ideal. Já no DBSCAN-MR, o parâmetro β, que especifica o número máximo de pontos em uma partição, foi variado de acordo com os tamanhos ideais de partição para os valores de k. Por exemplo, para um conjunto de dados de 1000 elementos e um valor de $k = 2$, o parâmetro β terá valor igual a 500.

Em ambas as estratégias de particionamento, o parâmetro $minPoints$ não exerce influência sobre o processo na construção das estruturas nem nos critérios usados para particionar o espaço. Visto isso, não foram realizados experimentos para observar como o balanceamento de carga e o tempo de execução se comportam com a alteração desse parâmetro.

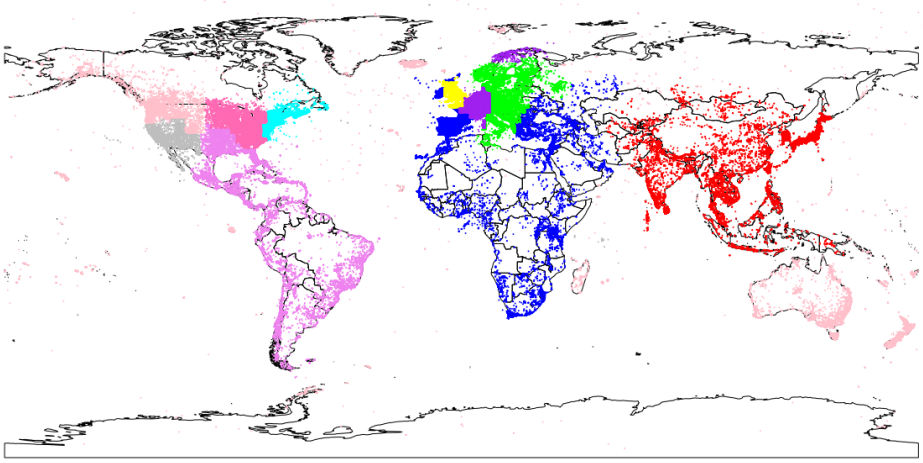
Como a etapa de particionamento contribui apenas com uma pequena parcela do tempo total de processamento de ambas as estratégias (G2P-DBSCAN e DBSCAN-MR), um conjunto de dados maior pode ser utilizado para observação do comportamento de ambas em diversos cenários. Nesse experimento utilizamos amostras do conjunto de dados YFCC100M escolhidas aleatoriamente com tamanho de até 32 milhões de pontos. A Tabela 5.3 mostra os

valores dos parâmetros utilizados na análise experimental. Os valores em negrito são fixados ao variar um certo parâmetro.

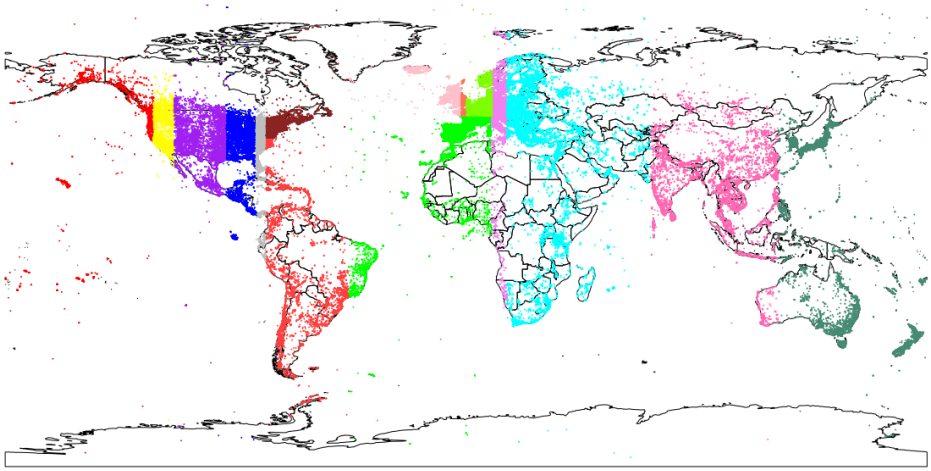
Tamanho do conjunto de dados	1M, 2M, 4M, 8M, 16M, **32M**
eps	0.06, 0.12, 0.25, **0.50**, 1.0
k	2, 3, ... , **12**, ... , 23 24

Tabela 5.3: Valores dos parâmetros utilizados na análise das estratégias de particionamento G2P e PRBP

É importante ressaltar que a estratégia G2P é capaz de cortar o espaço em múltiplas direções. Em outras palavras, ao particionar os dados, o algoritmo não considera somente as direções horizontais e verticais para dividir o espaço em duas partes, como podemos ver na Figura 5.3a. Por outro lado, o a estratégia PRBP considera somente a divisão binária do espaço (BSP) com cortes horizontais e verticais, como mostrado na Figura 5.3b. É possível identificar regiões que aparentam visualmente manter uma certa densidade. Na estratégia G2P essas regiões estão, na maioria das vezes, contidas em uma mesma partição. A estratégia PRBP apresenta diversos



(a) G2P



(b) PRBP

Figura 5.3: Visualização do particionamento no conjunto YFCC100M

cortes verticais do espaço, e poucos horizontais. Em todas as divisões realizadas percebe-se na Figura 5.3b que regiões do espaço que contém uma grande quantidade de pontos são separadas em mais de uma partição.

Variação do *eps*

A figura 5.4 mostra como as soluções se comportam, em relação ao tempo de execução (eixo y) quando o valor de *eps* (eixo x) aumenta. Os valores de *eps* são representados em termos de graus de latitude e longitude. Visto que na estratégia G2P a criação da estrutura de grade é de ordem linear em relação ao número de pontos, e a criação do grafo é linear em relação ao número de células da grade, o tempo de construção do grafo tende a diminuir com o aumento do *eps*, pois o número de células será menor e, por consequência, o grafo a ser particionado também será menor. Embora haja um leve aumento no tempo de execução quando o valor de *eps* é igual a 1.0, tal comportamento se deve à distribuição dos dados e à maneira como o grafo é construído, tornando mais difícil a tarefa de particionar.

Além disso, o tempo de execução do particionamento G2P é significativamente menor que o tempo de processamento da estratégia PRBP. Isso se dá pelo fato da estratégia PRBP construir a estrutura de grade através de fatias. Por exemplo, em uma grade todas as células com mesmo valor para a coordenada x pertencem à mesma fatia. Dessa forma, o algoritmo precisa analisar cada fatia para escolher qual delas será utilizada para a divisão do conjunto de dados.

O desbalanceamento entre as partições aumenta em ambas as estratégias à medida que o valor de *eps* cresce, conforme mostrado na Figura 5.4b. O aumento do valor de *eps* significa também uma diminuição no número de células na grade, já que as células terão tamanhos maiores. Dessa forma, a grade tende a ter uma menor quantidade de células com mais pontos à medida que o valor de *eps* cresce. Isso implica em um pior desbalanceamento dos dados visto que células com menos pontos tem maior probabilidade de serem melhor distribuídas entre as partições.

Apesar do aumento no desbalanceamento da carga com aumento do valor de *eps*, a estratégia G2P se mostrou mais estável. O maior desbalanceamento na estratégia PRBP se dá pelo fato do algoritmo só ser capaz de dividir o espaço de maneira binária com um corte horizontal ou vertical. Como a granulosidade utilizada para o corte do espaço está no nível das células da grade, um menor número de células implica em menos opções de corte para dividir o espaço. Dessa forma, o balanceamento das partições é prejudicado pelo fato do algoritmo priorizar a redução do número de pontos nas bordas.

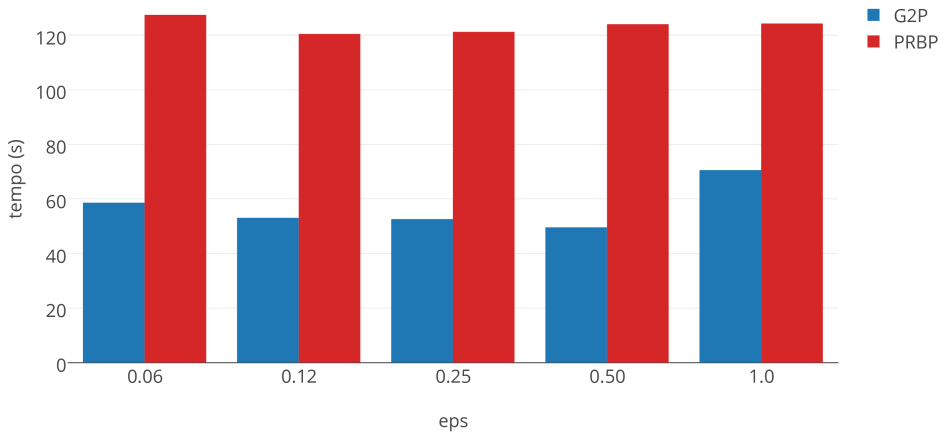
Variação do tamanho do conjunto de dados

Ao variar o tamanho do conjunto de dados podemos analisar tanto a escalabilidade da estratégia quanto a influência da quantidade de pontos no balanceamento de carga. As amostras utilizadas nesse experimentos variaram entre 1.000.000 e 32.000.000 de pontos.

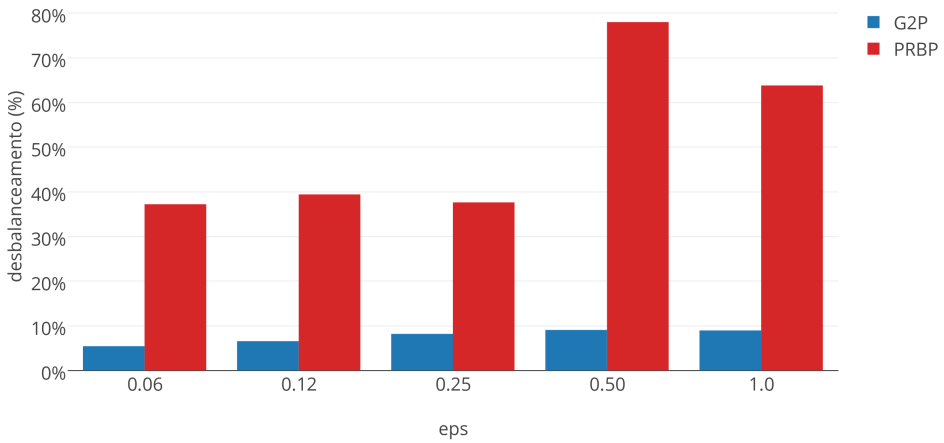
Na Figura 5.5a podemos observar o desempenho das duas abordagens. Embora ambas apresentem desempenhos similares para conjuntos de dados de tamanhos menores, o aumento do tempo de execução à medida que o conjunto de dados cresce é bem maior para a solução PRBP, enquanto que a estratégia G2P apresenta um crescimento menos acentuado. Isso

se dá pela complexidade envolvida na criação das estruturas de dados utilizadas em cada uma das soluções durante o proceso de particionamento dos dados. Como já afirmado anteriormente, a abordagem G2P precisa percorrer o conjunto de dados apenas uma única vez para a criar a estrutura de grade, apresentando assim complexidade linear com relação ao número de pontos. A construção do grafo também ocorre em tempo linear com relação ao número de células da grade. Por outro lado, a estratégia PRBP requer em um primeiro passo a computação de cada fatia da grade individualmente e, posteriormente, o cálculo de metadados sobre essas fatias, que serão utilizados como critério de decisão para o particionamento. Tais diferenças na maneira de computar as estruturas de dados e de decidir onde particionar tornam a estratégia G2P mais escalável que a estratégia PRBP.

A Figura 5.5b mostra que ambas as soluções mantém seu desbalanceamento de carga aproximadamente constante ao variar o tamanho do conjunto de dados a ser particionado.

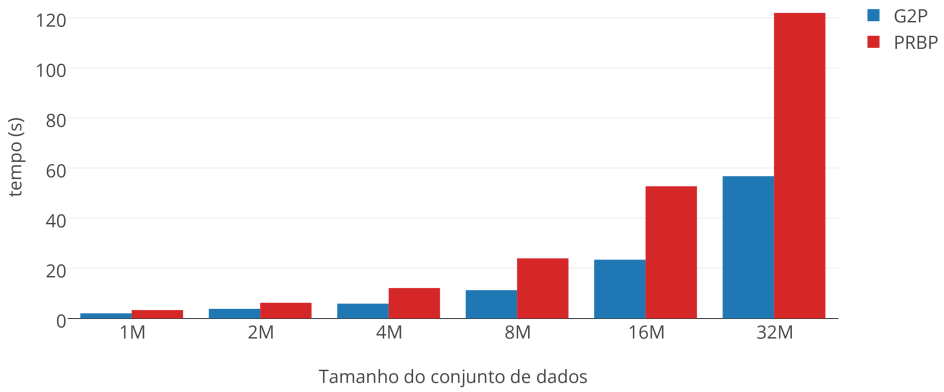


(a) Tempo de execução

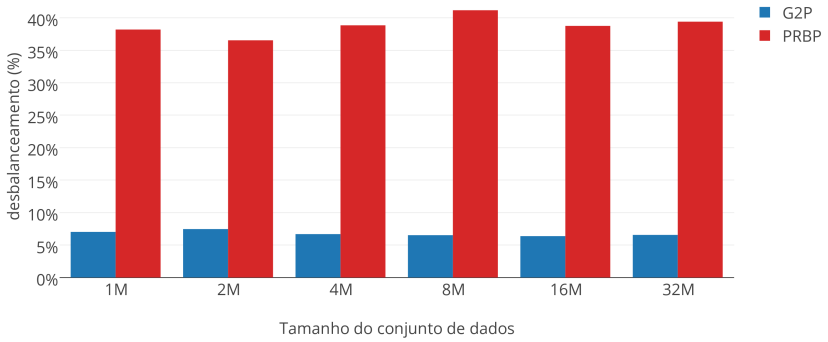


(b) Desbalanceamento

Figura 5.4: Variação dos valores de *eps*



(a) Tempo de execução



(b) Desbalanceamento

Figura 5.5: Variação do tamanho do conjunto de dados

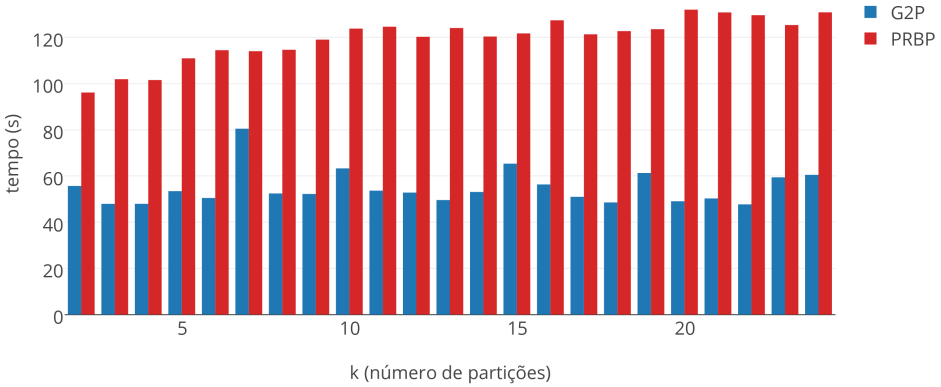
Tendo em vista que as amostras dos conjuntos de dados são coletadas aleatoriamente, é esperado que a distribuição dos dados entre as partições se mantenha similar quando conjuntos de dados de diferentes tamanhos são considerados. Assim, podemos concluir que, para uma dada distribuição de dados, o balanceamento entre as partições não é afetado pelo tamanho da amostra. Não obstante, a estratégia PRBP se mostra menos eficaz ao particionar os dados, uma vez que seu desbalanceamento se mantém em cerca de 40%, enquanto o G2P apresenta em torno de 10% de desbalanceamento.

Variação do número de partições (k)

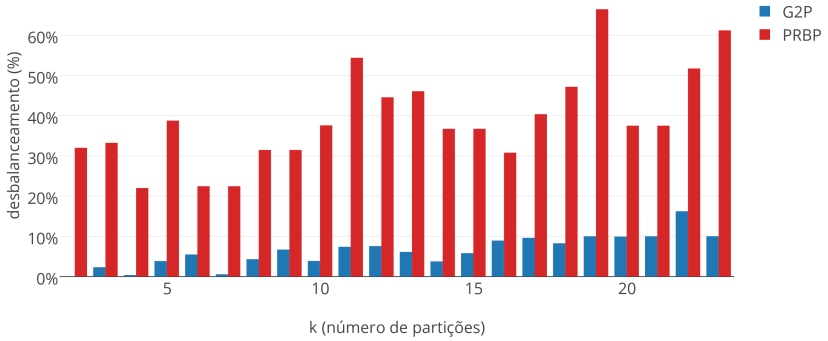
A Figura 5.6a mostra que o desempenho (eixo y) da estratégia G2P não apresenta uma tendência crescente ou decrescente de longo prazo à medida que o valor de k aumenta (eixo x), mas apenas variações irregulares. Todavia, podemos observar que, para todo valor de k considerado, o tempo de execução da nossa estratégia é significativamente inferior ao tempo de processamento do PRBP.

O desbalanceamento (eixo y) de ambas as estratégias analisadas tende a crescer à

medida que o valor de k (eixo x) aumenta, como mostrado na Figura 5.6b. Uma vez que o particionamento de ambas as soluções é baseado em uma estrutura de grade, a menor unidade de representação do espaço a ser particionado é a célula. Ao aumentar o número k de partições, o tamanho ideal das mesmas diminui. Consequentemente, o número de células necessárias para atingir o tamanho ideal de uma partição é menor, já que o tamanho das células não varia com k. Considerando isso, a distribuição das células em partições de maneira que a soma dos seus tamanhos se aproxime do tamanho ideal se torna mais difícil, visto que esse tamanho tende a se aproximar do tamanho das células.



(a) Tempo de execução



(b) Desbalanceamento

Figura 5.6: Variação do número k de partições

5.4.2 G2P-DBSCAN x DBSCAN-MR

Nesse experimento foi analisado o desempenho das estratégias de processamento distribuído do algoritmo DBSCAN, G2P-DBSCAN e DBSCAN-MR, variando os parâmetros *eps* e o tamanho do conjunto de dados. O tempo de execução considerado inclui o tempo gasto no processo de particionamento, na agrupamento distribuída, na identificação de *clusters* candidatos à junção e na junção propriamente dita.

O conjunto de dados utilizado nesse experimento foi o conjunto FOR. Por razão da escala dos dados desse conjunto, nós variamos os valores de *eps* de 10 a 10000. Nós limitamos o tamanho das amostras utilizadas a até 1.000.000 de pontos devido ao alto tempo de processamento da estratégia DBSCAN-MR. Como exemplo, o tempo de execução dessa estratégia para o maior conjunto de dados considerado foi de aproximadamente 7 horas. Esse resultado foi obtido no experimento mostrado na Figura 5.7 para o valor de *eps* igual 10.000.

Variação do *eps*

A Figura 5.7 mostra o efeito do valor do parâmetro *eps* (eixo x) no tempo de execução das estratégias (eixo y). Ambas apresentam uma degradação de desempenho quando o valor do parâmetro *eps* aumenta. Como o valor de *eps* define o raio onde a vizinhança é considerada, quanto maior for esse raio maior será o número de pontos vizinhos encontrados, e, por consequência, o número de pontos a serem verificados e expandidos. A estratégia G2P-DBSCAN se apresenta mais eficiente quando comparada à estratégia DBSCAN-MR. Uma das razões desse resultado é a diferença de desempenho na etapa de particionamento entre as duas soluções, como mostrado nos experimentos anteriores. Além disso, a etapa de junção de *clusters* acontece de maneira distribuída na estratégia G2P-DBSCAN, diferentemente da estratégia DBSCAN-MR.

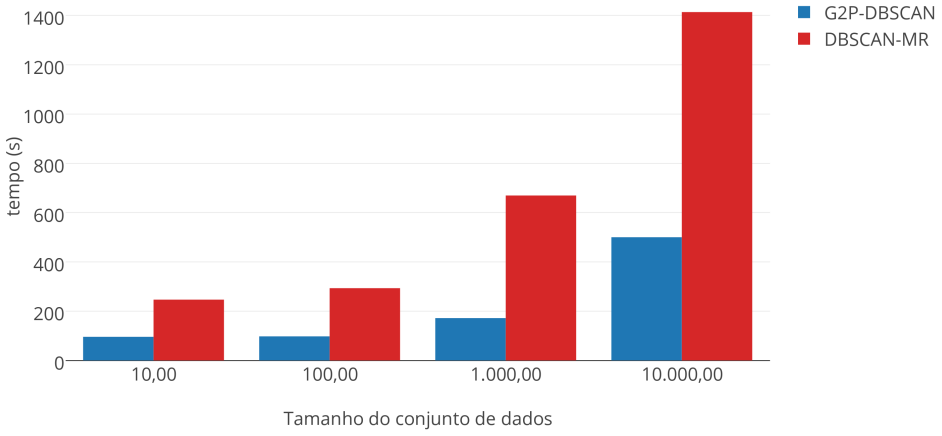


Figura 5.7: Variação do valor de *eps*

Variação do tamanho do conjunto de dados

Da mesma forma que a variação no tamanho do conjunto de dados influenciou no tempo de execução do particionamento, essa variação também teve impacto na estratégia de processamento distribuído do algoritmo DBSCAN. Como podemos ver na Figura 5.8, ambas as estratégias apresentam um aumento no tempo de execução (eixo y) à medida que o tamanho do conjunto de dados (eixo x) aumenta, como esperado. Porém, é possível perceber no gráfico que a estratégia G2P-DBSCAN apresenta um crescimento menos acentuado que a estratégia DBSCAN-MR.

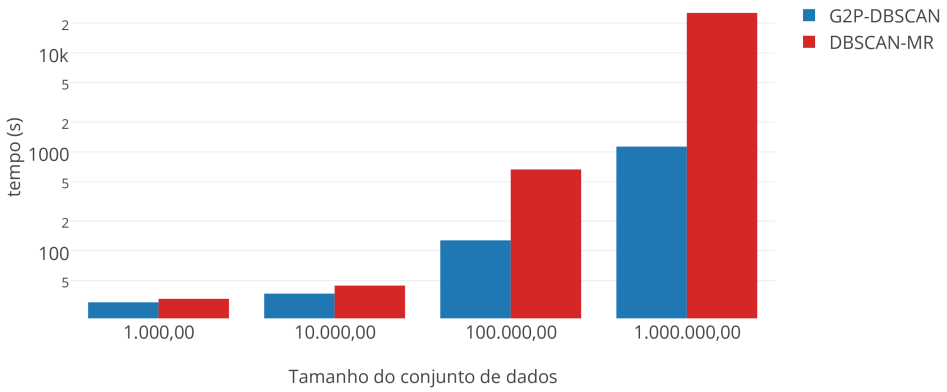


Figura 5.8: Variação tamanho do conjunto de dados

5.5 Conclusão

Neste capítulo foi apresentada uma análise experimental da nossa proposta G2P-DBSCAN usando dados reais. Os resultados dos experimentos mostraram que o G2P-DBSCAN supera de forma consistente o algortimo DBSCAN centralizado em termos de desempenho. Experimentos variando uma série de parâmetros também demonstraram a eficiência da nossa estratégia de particionamento G2P, tanto em termos de balanceamento de carga entre as partições encontradas quanto em desempenho. Ainda, foi feita uma análise comparativa do tempo total de processamento dos algoritmos G2P-DBSCAN e DBSCAN-MR que mostrou que a nossa proposta se comporta melhor que a estratégia DBSCAN-MR em todos os cenários considerados.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 Conclusão

O aumento na quantidade de dados produzidos por dispositivos móveis tem atraído cientistas de diversos domínios com o objetivo de extrair informações a partir desses dados, seja para fins comerciais ou acadêmicos. Porém, tal volume de dados requer que as técnicas de análise sejam capazes de processá-los de maneira eficiente. Ainda, o advento das infraestruturas de computação em nuvem permitiu a difusão das técnicas que se beneficiam de grandes conjuntos de máquinas, uma vez que o princípio de pagar apenas pelo uso possibilita a alocação de máquinas virtuais a baixos custos.

Nesta dissertação nós apresentamos G2P-DBSCAN, uma solução para o processamento do algoritmo de agrupamento baseado em densidade DBSCAN de forma distribuída através do paradigma MapReduce. O G2P-DBSCAN conta com uma estratégia de particionamento dos dados, chamada G2P, que leva em consideração as características do algoritmo DBSCAN e dos dados a serem processados, de forma a auxiliar no processo de agrupamento distribuído. O G2P usa uma estrutura de grade para dividir os elementos do conjunto de dados em células. Em seguida, essa grade é transformada em um grafo que reflete a densidade das células através dos pesos dos vértices e das arestas. O problema de particionamento dos dados é então transformado em um problema de particionamento de grafos. Essa estratégia permite que o espaço seja cortado em múltiplas direções, não se limitando apenas às direções vertical e horizontal, diferentemente de trabalhos relacionados.

Após o particionamento, o G2P-DBSCAN executa um processo MapReduce que realiza o agrupamento nas partições através da execução do algoritmo DBSCAN nos pontos pertencentes a cada uma delas. Uma vez que um mesmo *cluster* pode ter sido quebrado em diferentes partições durante a etapa de particionamento, nós apresentamos também um método para junção de *clusters* que é realizado de forma distribuída através de outro processo MapReduce.

Nós realizamos experimentos utilizando dados reais que mostraram que nossa proposta G2P-DBSCAN é siginificativamente mais rápida que o algoritmo DBSCAN centralizado, como esperado. Experimentos variando uma série de parâmetros também demonstraram que nossa estratégia de particionamento G2P supera a estratégia PRBP usada no algoritmo DBSCAN-MR, tanto em termos de balanceamento de carga entre as partições encontradas quanto em desempenho. Uma análise comparativa do tempo total de processamento dos algoritmos G2P-DBSCAN e DBSCAN-MR também mostrou que a nossa proposta se comporta melhor que a estratégia DBSCAN-MR em todos os cenários avaliados.

6.2 Trabalhos Futuros

Como trabalho futuro, gostaríamos de investigar outras técnicas que possam auxiliar no desempenho de algoritmos de agrupamento distribuídos, garantindo a qualidade dos resul-

tados produzidos. Ao lidar com grandes quantidades de dados, mecanismos que sirvam como pré-processamento para garantir escalabilidade aos algoritmos também devem ser investigados e propostos.

Um aspecto ainda inexplorado pelos trabalhos existentes na literatura é o ajuste dos parâmetros do algoritmo DBSCAN de maneira incremental. Para um cenário estabelecido, com um conjunto de dados e valores dos parâmetros $minPoints$ e eps, o armazenamento do resultado do agrupamento para esses valores em estruturas de dados adequadas poderia ser utilizado em futuras agrupamentos, de forma a evitar que algumas operações sejas executadas novamente. Assim, o estudo de diversas configurações de parâmetros para grandes volumes de dados se tornaria mais simples e poderia ser realizado com um baixo custo de processamento e em menos tempo.

Outro desafio consiste na aplicação de técnicas de particionamento de dados e distribuição apropriadas para outros algoritmos de análise de dados, de maneira que esses possam se beneficiar da divisão realizada nos dados para uma computação mais eficiente. Embora alguns algoritmos de agrupamento já possuam suas versões distribuídas, diversos outros continuam sendo aplicados apenas em suas versões originais. Por último, o uso de outros modelos de distribuição, além do MapReduce, para algoritmos de análise de dados, dá abertura para o surgimento de novas estratégias de particionamento e distribuição de processamento, que podem tirar proveito desses modelo para abordagens mais eficientes.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, G. et al. G-DBSCAN: A GPU accelerated algorithm for density-based clustering. In: *Proceedings of the International Conference on Computational Science, ICCS 2013, Barcelona, Spain, 5-7 June, 2013*. [S.l.: s.n.], 2013. p. 369–378.

ANKERST, M. et al. OPTICS: ordering points to identify the clustering structure. In: DELIS, A.; FALOUTSOS, C.; GHANDEHARIZADEH, S. (Ed.). *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*. [S.l.]: ACM Press, 1999. p. 49–60. ISBN 1-58113-084-8.

BECKMANN, N. et al. The r*-tree: An efficient and robust access method for points and rectangles. In: GARCIA-MOLINA, H.; JAGADISH, H. V. (Ed.). *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990*. ACM Press, 1990. p. 322–331. Disponível em: <http://doi.acm.org/10.1145/93597.98741>.

BIRANT, D.; KUT, A. St-dbscan: An algorithm for clustering spatialtemporal data. *Data and Knowledge Engineering*, v. 60, n. 1, p. 208 – 221, 2007. Intelligent Data Mining.

BULUÇ, A. et al. Recent advances in graph partitioning. *CoRR*, abs/1311.3144, 2013.

CHEN, W.; JI, M.; WANG, J. T-DBSCAN: A spatiotemporal density clustering for GPS trajectory segmentation. *iJOE*, v. 10, n. 6, p. 19–24, 2014.

DAI, B.; LIN, I. Efficient map/reduce-based DBSCAN algorithm with optimized data partition. In: CHANG, R. (Ed.). *2012 IEEE Fifth International Conference on Cloud Computing, Honolulu, HI, USA, June 24-29, 2012*. [S.l.]: IEEE, 2012. p. 59–66. ISBN 978-1-4673-2892-0.

DEAN, J.; GHEMAWAT, S. Mapreduce: Simplified data processing on large clusters. In: BREWER, E. A.; CHEN, P. (Ed.). *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*. [S.l.]: USENIX Association, 2004. p. 137–150.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: SIMOUDIS, E.; HAN, J.; FAYYAD, U. M. (Ed.). [S.l.]: AAAI Press, 1996. p. 226–231. ISBN 1-57735-004-9.

GAN, J.; TAO, Y. DBSCAN revisited: Mis-claim, un-fixability, and approximation. In: SELLIS, T.; DAVIDSON, S. B.; IVES, Z. G. (Ed.). *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. [S.l.]: ACM, 2015. p. 519–530. ISBN 978-1-4503-2758-9.

GAREY, M. R.; JOHNSON, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

GIANNOTTI, F. et al. Unveiling the complexity of human mobility by querying and mining massive trajectory data. *VLDB J.*, v. 20, n. 5, p. 695–719, 2011. Disponível em: <http://dx.doi.org/10.1007/s00778-011-0244-8>.

GUTTMAN, A. R-trees: A dynamic index structure for spatial searching. In: YORMARK, B. (Ed.). *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*. ACM Press, 1984. p. 47–57. Disponível em: <http://doi.acm.org/10.1145/602259.602266>.

HAN, J.; KAMBER, M. *Data Mining: Concepts and Techniques*. [S.l.]: Morgan Kaufmann, 2000. ISBN 1-55860-489-8.

HE, Y. et al. MR-DBSCAN: an efficient parallel density-based clustering algorithm using mapreduce. In: *IEEE 17th International Conference on Parallel and Distributed Systems, ICPADS 2011, Tainan, Taiwan, December 7-9, 2011*. [S.l.]: IEEE, 2011. p. 473–480. ISBN 978-1-4577-1875-5.

ISLAM, A. K. M. T. et al. Mapreduce based parallel gene selection method. *Appl. Intell.*, v. 42, n. 2, p. 147–156, 2015.

KARYPIS, G.; KUMAR, V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Scientific Computing*, v. 20, n. 1, p. 359–392, 1998.

KIM, Y. et al. DBCURE-MR: an efficient density-based clustering algorithm for large data using mapreduce. *Inf. Syst.*, v. 42, p. 15–35, 2014.

LIU, H. et al. Landmark FN-DBSCAN: an efficient density-based clustering algorithm with fuzzy neighborhood. *JACIII*, v. 17, n. 1, p. 60–73, 2013.

MOON, B. et al. Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Trans. Knowl. Data Eng.*, v. 13, n. 1, p. 124–141, 2001.

RÍO, S. del et al. A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. *Int. J. Computational Intelligence Systems*, v. 8, n. 3, p. 422–437, 2015.

ROTH, P. C.; ARNOLD, D. C.; MILLER, B. P. Mrnet: A software-based multicast/reduction network for scalable tools. In: *Proceedings of the ACM/IEEE SC2003 Conference on High Performance Networking and Computing, 15-21 November 2003, Phoenix, AZ, USA, CD-Rom*. ACM, 2003. p. 21. ISBN 1-58113-695-1. Disponível em: <http://doi.acm.org/10.1145/1048935.1050172>.

SCHADT, E. E. et al. Computational solutions to large-scale data management and analysis. *Nat Rev Genet*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 11, n. 9, p. 647–657, set. 2010. ISSN 1471-0064.

SCICLUNA, N.; BOUGANIS, C. Fpga-based parallel DBSCAN architecture. In: GOEHRINGER, D. et al. (Ed.). *Reconfigurable Computing: Architectures, Tools, and Applications - 10th International Symposium, ARC 2014, Vilamoura, Portugal, April 14-16, 2014. Proceedings*. [S.l.]: Springer, 2014. (Lecture Notes in Computer Science, v. 8405), p. 1–12. ISBN 978-3-319-05959-4.

SOUSA, F. R. et al. Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. In: SBC. *SBBD*. [S.l.], 2010. p. 101–130.

SVENDSEN, M.; MUKHERJEE, A. P.; TIRTHAPURA, S. Mining maximal cliques from a large graph using mapreduce: Tackling highly uneven subproblem sizes. *J. Parallel Distrib. Comput.*, v. 79, p. 104–114, 2015.

THEODORIDIS, Y.; SELLIS, T. K. A model for the prediction of r-tree performance. In: HULL, R. (Ed.). *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, 1996, Montreal, Canada*. [S.l.]: ACM Press, 1996. p. 161–171.

THOMEE, B. et al. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

UNCU, Ö. et al. GRIDBSCAN: grid density-based spatial clustering of applications with noise. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, October 8-11, 2006*. [S.l.]: IEEE, 2006. p. 2976–2981. ISBN 1-4244-0099-6.

W, K.; SHEN, L. *Method of minimizing the interconnection cost of linked objects*. [S.l.]: Google Patents, 1971. US Patent 3,617,714.

WANG, B. et al. Design and optimization of DBSCAN algorithm based on CUDA. *CoRR*, abs/1506.02226, 2015.

WANG, W. et al. A C-DBSCAN algorithm for determining bus-stop locations based on taxi GPS data. In: LUO, X.; YU, J. X.; LI, Z. (Ed.). *Advanced Data Mining and Applications - 10th International Conference, ADMA 2014, Guilin, China, December 19-21, 2014. Proceedings*. [S.l.]: Springer, 2014. (Lecture Notes in Computer Science, v. 8933), p. 293–304. ISBN 978-3-319-14716-1.

WELTON, B.; SAMANAS, E.; MILLER, B. P. Mr. scan: extreme scale density-based clustering using a tree-based network of GPGPU nodes. In: GROPP, W.; MATSUOKA, S. (Ed.). *International Conference for High Performance Computing, Networking, Storage and Analysis, SC'13, Denver, CO, USA - November 17 - 21, 2013*. [S.l.]: ACM, 2013. p. 84:1–84:11. ISBN 978-1-4503-2378-9.

WHITE, T. *Hadoop: The Definitive Guide*. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2009. ISBN 0596521979, 9780596521974.

XU, X.; JÄGER, J.; KRIEGEL, H. A fast parallel clustering algorithm for large spatial databases. *Data Min. Knowl. Discov.*, v. 3, n. 3, p. 263–290, 1999.

ZHANG, L.; XU, Z.; SI, F. GCMDDBSCAN: multi-density DBSCAN based on grid and contribution. In: *IEEE 11th International Conference on Dependable, Autonomic and Secure Computing, DASC 2013, Chengdu, China, December 21-22, 2013*. IEEE, 2013. p. 502–507. ISBN 978-1-4799-3380-8. Disponível em: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6843220>.