UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
BACHARELADO EM ENGENHARIA DE SOFTWARE

ERICK BHRENER BARROSO SILVA

UM COMPARATIVO DO ALGORITMO DBSCAN EM AMBIENTES DE MOVIMENTO LIVRE E EM REDE DE RUAS

QUIXADÁ

2016

ERICK BHRENER BARROSO SILVA

UM COMPARATIVO DO ALGORITMO DE DBSCAN EM AMBIENTES DE MOVIMENTO LIVRE E EM REDE DE RUAS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Engenharia de Software da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.
Área de concentração: Computação.

Orientadora: Profª. MSc. Ticiana Linhares Coelho da Silva.

QUIXADÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S579c Silva, Erick Bhrener Barroso.
Um comparativo do algoritmo DBSCAN em ambientes de movimento livre e em rede de ruas / Erick Bhrener Barroso Silva. – 2016.
40 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2016.
Orientação: Prof. Me. Ticiana Linhares Coelho da Silva.

1. Ruas. 2. Sistema de Posicionamento Global. 3. Cluster. 4. Análise por agrupamento. 5. Algoritmos. I. Título.

CDD 005.1

ERICK BHRENER BARROSO SILVA

UM COMPARATIVO DO ALGORITMO DE DBSCAN EM AMBIENTES DE MOVIMENTO LIVRE E EM REDE DE RUAS

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Bacharelado em Engenharia de Software da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Bacharel.
Área de concentração: Computação.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____
Prof.ª MSc. Ticiana Linhares Coelho da Silva
Universidade Federal do Ceará (UFC)

_____
Prof. Msc. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

_____
Prof.ª Dra. Atslands Rego da Rocha
Universidade Federal do Ceará (UFC)

Aos meus pais, Eva e Auricélio, por toda a confiança e suporte dado a mim para chegar até aqui.

AGRADECIMENTOS

Primeiramente a Deus, por tudo que a mim foi dado.

Aos meus pais, Eva Christian e Francisco Auricélio, que me deram a melhor educação, amor e carinho, e são minha base para tudo nessa vida. A minha vó, Raimunda Irene, por tudo que fez por mim, ter me criado, me educado e influenciado na pessoa que eu sou. E a minha irmã, Clarice.

Agradeço especialmente a minha orientadora Ticiana Linhares, por todos os conhecimentos que me ensinou, pelo tempo que dedicou a mim com seu excelente trabalho e por ser uma grande influencia na minha carreira acadêmica e profissional.

Aos meus amigos(as) Thais Luana, Thais Oliveira, Natalie Santos, Jean Zanella, que sempre me motivaram a ser uma pessoa melhor. Aos meus amigos Rogerio, Hinessa e Rafael por todo o suporte e amizade. Todos os meus amigos, irmãos, que adquiri durante minha jornada pelo Ciências sem Fronteiras.

Aos professores participantes da banca examinadora Regis Pires e Atslands Rocha pelo tempo, pelas valiosas colaborações e sugestões.

Ao Venício de Oliveira, por ter me ajudado todas as vezes que eu precisei e seu excelente trabalho para com o curso de Engenharia de Software na secretaria.

A Universidade Federal do Ceará, e a todos os funcionários e docentes que ali trabalham.

A Universidade Cal State Fullerton, California, Estados Unidos, pelos recursos fornecidos para esse trabalho.

“A verdadeira viagem de descobrimento não consiste em procurar novas paisagens, mas em ter novos olhos.”

(Marcel Proust)

RESUMO

Nos últimos anos, em consequência do avanço tecnológico, um grande volume de dados são gerados todos os dias pelas mais diversas tecnologias. Dentre essas tecnologias, encontram-se os dados de navegação de veículos, fornecidos por sensores GPS. Esse volume de dados cresce consideravelmente e é atualmente objetivo de estudo para as mais diversas entidades. Desta forma, existe a necessidade de encontrar soluções efetivas para a extração de informações. Assim, o objetivo desse trabalho consiste na implementação do algoritmo DBSCAN, que utiliza a técnica de clusterização, para comparar dois ambientes diferentes analisando dados geoespaciais gerados por veículos em Beijing, China. O primeiro ambiente, de movimento livre, usa distância euclidiana, e o segundo ambiente utiliza uma Rede de Ruas, que considera as propriedade físicas reais do mapa. Foram realizados testes que comparam os dois ambientes, apresentando a efetividade e deficiências de cada um, velocidade de execução e precisão dos resultados.

Palavras-chave: DBSCAN. Rede de Ruas. Distância Euclidiana.

ABSTRACT

In recent years, as a result of technological advancement, a large volume of data is generated every day by different technologies. Among these technologies are navigation data of vehicles, provided by GPS sensors. The volume of data grows considerably and it is currently a field of study to different kinds of entities. Thus, there is a need for effective solutions for the extraction of information. The objective of this work is the implementation of DBSCAN algorithm, which uses clustering technique, to compare two different environments to analyze geospatial data generated by vehicles in Beijing, China. The first environment, free movement, uses Euclidean distance and the second environment uses a Road Network, which considers the actual physical properties of the map. Tests were conducted to compare the two environments, with the effectiveness and shortcomings of each one, the execution speed, and accuracy of results.

Keywords: DBSCAN. Road Network. Euclidean Distance.

LISTA DE FIGURAS

Figura 1 – Exemplos de um *cluster*, *core point* e *border point*. …..... 17
Figura 2 – Algoritmo *Dijkstra* passo a passo …..... 19
Figura 3 – Densidade dos pontos em Beijing, China …..... 24
Figura 4 – Distância entre dois pontos …..... 25

LISTA DE TABELAS

Tabela 1 – Descrição de hardware …..... 27
Tabela 2 – Descrição de cada janelas de tempo..... 27
Tabela 3 – Tabela de pré-processamento em segundos …..... 28
Tabela 4 – Resultados do teste para *eps*=0.01 e *MinPts*=30 utilizando distância euclidiana …..... 29
Tabela 5 – Resultados do teste para *eps*=0.01 e *MinPts*=30 utilizando distância de rede …..... 29
Tabela 6 – Para *eps*=0.001 e *MinPts*=30 …..... 30
Tabela 7 – Para *eps*=0.002 e *MinPts*=30 …..... 30
Tabela 8 – Para *eps*=0.1 e *MinPts*=30 …..... 31
Tabela 9 – Para *eps*=0.2 e *MinPts*=30 …..... 31
Tabela 10 – Resultados do teste para *eps*=0.02 e *MinPts*=30 utilizando distância euclidiana …..... 32
Tabela 11 – Resultados do teste para *eps*=0.02 e *MinPts*=30 utilizando distância de rede …..... 32
Tabela 12 – Resultados do teste para *eps*=0.02 e *MinPts*=40 utilizando distância euclidiana …..... 33
Tabela 13 – Resultados do teste para *eps*=0.02 e *MinPts*=40 utilizando distância de rede …..... 33

SUMÁRIO

1 **INTRODUÇÃO** 13

2 **OBJETIVOS** 15

2.1 **Objetivo Geral** 15

2.2 **Objetivos Específicos** 15

3 **FUNDAMENTAÇÃO TEÓRICA** 16

3.1 **Rede de Ruas** 16

3.2 **DBSCAN** 16

3.3 **Algoritmos de Cálculo de Menor Caminho** 17

3.3.1 *Dijkstra* 18

3.3.2 *A ** 19

3.4 **Map Matching** 20

4 **TRABALHOS RELACIONADOS** 20

4.1 **Efficient and Distributed DBScan Algorithm Using MapReduce to Detect Density Areas on Traffic Data** 21

4.2 **NEAT: Road Network Aware Trajectory Clustering** 21

4.3 **NNCluster: An Efficient Clustering Algorithm for Road Network Trajectories** 22

5 **PROCEDIMENTOS METODOLÓGICOS** 23

5.1 **Coleta de Dados** 23

5.2 **Criação da Topologia da Rede** 24

5.3 **Map Matching** 25

5.4 **Implementação do DBSCAN** 25

5.4.1 *Usando distância Euclidiana* 25

5.4.2 *Usando distância na rede de ruas* 26

5.5 **Avaliação e Validação** 26

6 **RESULTADOS** 27

6.1 **Coleta de dados** 27

6.2 **Pré-processamento** 28

6.3 **Desenvolvimento e análise dos resultados** 28

6.3.1 *Testes para MinPts fixo* 29

6.3.2 *Testes para eps fixo* 32

6.3.3 *Análise dos resultados* …..... 33
7 **CONSIDERAÇÕES FINAIS**..... 35
8 **TRABALHOS FUTUROS**..... 36
REFERÊNCIAS 37

1 INTRODUÇÃO

O avanço continuo da tecnologia é um campo ilimitado de estudo e conhecimento. Diariamente, enormes quantidades de dados são gerados por várias sistemas, sejam estes sistemas web, dispositivos móveis (*smartfones*, *smartwatches*, etc.) e dispositivos embarcados, como os usados em carros. Essas dados ficam armazenados em diversas bases de dados, e possuem pouca utilidade nesse estado, pois não passam de meros registros. Porém, existem técnicas que permitem a análise e extração de informações relevantes desses dados. Algumas das principais e mais efetivas técnicas são as de Mineração de Dados.

Segundo Coelho da Silva *et al.* (2013), o processo de mineração de dados é composto por um conjunto de técnicas baseado em modelos capazes de sumarizar dados, extrair novos conhecimentos ou realizar predições com o objetivo de descobrir informações com base em grandes volumes de dados. Exitem várias técnicas de mineração de dados, as principais são categorizadas em classificação, clusterização ou regras de associação. A técnica de clusterização será utilizada neste trabalho para a descoberta de conhecimento em dados de tráfego. Esta técnica, segundo Berkhin (2006), é definida como a divisão dos dados em grupos de objetos similares.

Nesse contexto, as grandes cidades possuem em suas ruas um número considerável de fotos sensores e radares que a todo momento registram informações sobre os veículos nas vias. Além desses registros, os próprios veículos também produzem dados por meio dos sistemas de GPS (Geo Positioning System). Esses dados quando são analisados podem identificar pontos de congestionamento, vias mais utilizadas, etc. Tais informações não servem apenas como fonte de estudo, mas também como fonte base para que órgãos governamentais responsáveis possam realizar melhorias na engenharia de trânsito das cidades.

Os dados produzidos pelos veículos são registrados contendo sua posição geográfica, definida por uma longitude e uma latitude, além da data em que aquela posição foi registrada. Assim, é possível considerar que para um dado intervalo de tempo, todas as posições registradas pelos veículos formam um conjunto de pontos, e um grande aglomerado de pontos representa um congestionamento. Há técnicas de clusterização capazes de descobrir esses aglomerados nos dados, sendo um dos mais importantes algoritmos de clusterização capaz de realizar tal análise o algoritmo DBSCAN (*Density-based Spatial Clustering of Application with Noise*).

O DBSCAN pode encontrar agrupamentos baseando-se na vizinhança dos objetos, onde a densidade associada a um ponto é obtida por meio da contagem do número de pontos

que estão a uma distância desse ponto de no máximo um valor dado como entrada (Gava *et al.*, 2012). Coelho da Silva *et al.* (2013) desenvolveu uma versão distribuída do algoritmo DBSCAN, usando *MapReduce* para encontrar regiões de congestionamentos na cidade de Fortaleza. Esse algoritmo, porém, considerava apenas a distância euclidiana (Gimenes, Gimenes e Opazo, 2005, p. 6) entre os pontos presentes na base de dados. Sabe-se que tal método pode ser impreciso quanto aos resultados finais de seu processamento (Yan e Mamoulis, 2014) para dados de tráfego, pois este não considera as propriedades físicas reais de onde os objetos se encontram. Assim, a opção correta nesse contexto envolve o conceito de Rede de Ruas, em que a distância entre dois pontos é o menor caminho entre eles na Rede. Redes de Ruas são usadas em forma de grafos, considerando nós como sendo cruzamentos ou esquinas e as arestas como cada rua em uma cidade.

Este trabalho propõe a aplicação do algoritmo DBSCAN para Redes de Ruas a fim de provar sua eficacia e precisão comparado as versões que utilizam distância euclidiana. A validação do algoritmo será realizada de duas formas. Primeiramente, executando o algoritmo com diferentes volumes de dados e comparados os resultados entre diferentes intervalos de tempo. Por último, serão comparados os resultados obtidos na versão que utiliza a distância na Rede de Ruas com a versão que utiliza a distância euclidiana.

A próxima seção descreve os objetivos, geral e específicos. A seção 3 trata da fundamentação teórica, ou seja, os conceitos básicos para o entendimento deste trabalho. A seção 4 apresenta os principais trabalhos relacionados. A seção 5 descreve os procedimentos metodológicos. A seção 6 traz os resultados obtidos. A seção 7 apresenta as considerações finais deste trabalho, contendo a conclusão e os trabalhos futuros.

2 OBJETIVOS

2.1 Objetivos Gerais

Comparar as diferentes implementações do algoritmo DBSCAN em ambientes de movimento livre (com distância euclidiana) e em ambientes de movimento em rede de ruas (distância de rede).

2.2 Objetivos Específicos

- Coleta e preparação dos dados;
- Associar cada posição a uma aresta da rede;
- Implementar o algoritmo DBSCAN utilizando distância euclidiana;
- Implementar o algoritmo DBSCAN baseado em uma rede de ruas;
- Realizar experimentos com ambas versões (para movimento em rede e movimento livre) para análise e comparação dos resultados.

3 FUNDAMENTAÇÂO TEÓRICA

Nesta seção serão abordados os conceitos necessários para o entendimento deste trabalho. A seção 2.1 aborda os conceitos sobre Rede de Ruas. Na seção 2.2 o algoritmo DBScan será explicado junto com suas definições. Na seção 2.3 trata sobre algoritmos de cálculo de distância em redes, com enfase no algoritmo Dijkstra, que será tratado na seção 2.3.1. Seção 2.4 explica o que é o processo de *map matching.*

3.1 Rede de Ruas

Uma rede de ruas pode ser definida como um conjunto de ruas que se interligam, como por exemplo, de uma cidade ou de específica região desta. Han, Liu e Omiecinski (2012) definem uma rede de ruas como sendo um grafo *G = (V,E)*, onde *V* é o conjunto de vértices ou nós e em que *E* representa o conjunto das arestas. Os vértices representam cruzamentos entre ruas, início ou fim de cada via e as arestas representam blocos ou quarteirões de ruas. Neste trabalho, cada objeto móvel foi associado a alguma posição de uma aresta na rede.

A Rede de Ruas é obtida através de dados geográficos fornecidos pelo projeto Open Street Map[1]. Os vértices são definidos por *nodes*, as arestas por *edges*. Os dados fornecem qual o vértice inicial e qual o vértice final de cada aresta, o custo ou distância para percorrer do início ao fim (do vértice origem ao vértice destino). Essas informações são importantes para a construção da topologia da rede e para o processo de *Map Matching* explicados mais a frente na seção 3.4.

3.2 DBSCAN

Dentisy-Based Spatial Clustering of Applications with Noise (DBSCAN) é um método de clusterização baseado em densidade. DBSCAN foi primeiro apresentado por Ester *at al*. (1996) e vem sendo largamente utilizado na comunidade científica. Segundo Tran, Drab, e Daszykowski (2012), a complexidade mínima desse algoritmo é $O(n \cdot \log(n))$ e sua performance está diretamente ligada a quantidade de recursos disponíveis para o processamento.

1 www.openstreetmap.org

DBSCAN necessita de dois parâmetros de entrada:

a) *eps*: determina a distância da qual um ponto deve estar de outro para serem vizinhos;

b) *MinPts*: parâmetro que especifica o número mínimo de pontos vizinhos, dentro da distância *eps* definida, que um ponto precisa ter para ser considerado *core point*, e assim, iniciar um novo *cluster*.

Algumas definições importantes do DBSCAN são:

a) *core point*: ponto no qual seu número de vizinhos é maior do que *MinPts*;

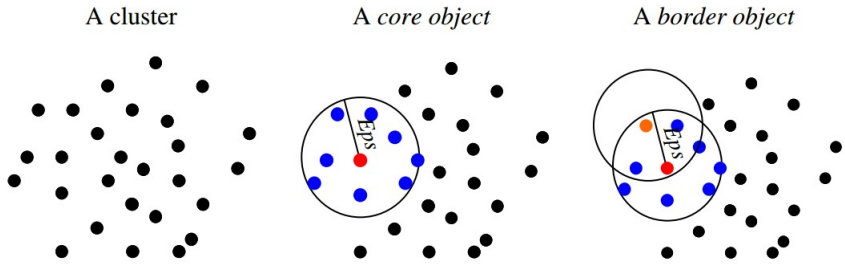
b) *noise point*: ponto que não pertence a nenhum *cluster*;

c) *border point*: ponto que não forma cluster, mas é vizinho de um *core point*.

Para ser um *core point*, o ponto precisa ter um número de vizinhos maior do que o valor *MinPts* fornecido pelo usuário. Um *core point* em conjunto com os pontos vizinhos formam um cluster. Os pontos vizinhos são considerados *border points*. Se após todo o processamento, um ponto não for *core point* ou *border point*, ele é considerado um *noise point*.

A figura 1 traz uma representação gráfica das definições acima. O ponto vermelho representa um *core point*, o laranja um *border point* e os azuis os vizinhos que irão compor um novo *cluster*.

Figura 1: Exemplos de um *cluster*, *core point* e *border point*.



Fonte: Tran, Drab e Daszykowski (2013)

3.3 Algoritmos de Menor Caminho

Os algoritmos de menor caminho são soluções bastante usadas tanto na comunidade acadêmica, quanto para fins comerciais. Segundo Djojo e Karyono (2013),

“Menor caminho é um problema de optimização que vem sendo estudado até hoje pra várias aplicações”.

De acordo com Djojo e Karyono (2013), “[...] menor caminho é o problema de achar o caminho entre dois pontos em um grafo onde a soma dos custos das arestas do caminho possuam o menor valor”. Em uma rede de ruas, faz-se necessário o uso de algoritmos de menor caminho, onde a distância entre dois objetos considera o caminho a ser percorrido por meio das arestas existentes na rede. Dessa forma, não é aplicável usar distância euclidiana onde existe uma rede de ruas.

As seções 2.3.1 e 2.3.2 apresentam os algoritmos de menor caminho mais utilizados, segundo Djojo e Karyono (2013).

3.3.1 Dijkstra

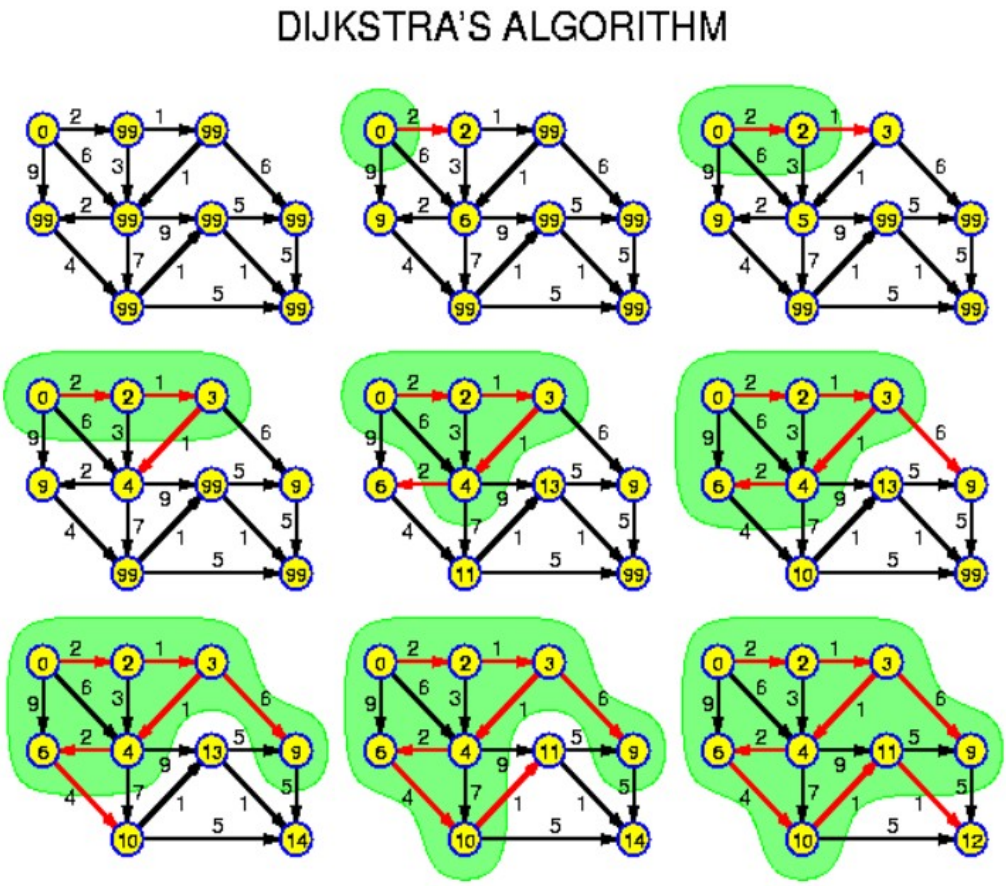
Dijkstra é um algoritmo que encontra o menor caminho entre dois pontos em um grafo. Esse algoritmo é geralmente usado para roteamento e como uma sub-rotina de outros algoritmos para grafos Djojo e Karyono (2013). Jasika *at al*. (2012) define os seguintes passos para o algoritmo:

- Escolhe o vértice raiz do grafo;
- Define um conjunto *S* de vértices e inicia o mesmo como vazio. Ao decorrer do processamento, os vértices que fazem parte do caminho mais curto serão guardados nesse conjunto;
- O vértice raiz recebe um valor da origem e é guardado no conjunto *S*; O valor associado para alcançar a origem é zero.
- A cada iteração é retirado um vértice do conjunto S, sendo este o que apresenta o menor valor de alcance da origem até ele.
- O vértice retirado é expandido e será incluído em S, o vértice vizinho que apresentar menor custo de alcance a partir da origem.
- O processo ocorre até que o vértice de destino seja achado ou que não existam mais vértices.

A Figura 2 apresenta um exemplo de como *Dijkstra* funciona. O primeiro vértice é analisado e recebe um valor de distância igual a zero. Então o algoritmo analisa os vértices vizinhos, guardando as distâncias entre estes até o vértice destino, ou que

não existam mais vértices.

Figura 2: Algoritmo *Dijkstra* passo a passo



Fonte: Jasika *at al*. (2012)

Dijkstra será o algoritmo utilizado nesse trabalho. Djojo e Karyono (2013) apresentaram testes de performance que apresentam a qualidade do algoritmo, quanto ao tempo e custo computacional de processamento, comparado ao algoritmo *A**. Além disso, o algoritmo será usado na como pré-processamento desse trabalho, pois não é o foco do trabalho. No pré-processamento é onde todas as distâncias serão calculadas entre cada vértice para todos os outros para serem acessadas pelo algoritmo *Dijkstra*. O autor preferiu pré-computar essas distâncias, pois é custoso o cálculo do menor caminho durante a execução do algoritmo DBSCAN para o movimento em rede de ruas. Assim, a performance do DBSCAN não será diretamente afetada pelo processamento de *Dijkstra*. Como trabalhos futuros, pretende-se utilizar A*, pois fornece resultados mais rápidos considerando ambientes de *streaming*.

*3.3.2 A**

Segundo Wang *at al*. (2014), “ O algoritmo A* (estrela) é uma extensão do algoritmo *Dijkstra*”. Para AlShawi at al. (2012, pg. 3014), este é um algoritmo baseado em heurísticas altamente eficiente para encontrar o menor caminho. A* usa uma função de avaliação (geralmente denotada por *f(n)*) para guiar e determinar a ordem em que a procura visita cada nó em um grafo. A função de avaliação é dada como a seguir:

$$f(n) = g(n) + h(n)$$

Onde *g(n)* é a distância conhecida entre o vértice de início e o vértice *n* e *h(n)* é o valor estimando de distância entre o vértice *n* e o vértice de destino. A explicação do algoritmo A* não será aprofundada por não ser o foco deste trabalho.

3.4 Map Matching

Map Matching é a processo de determinar em que via um objeto se encontra (Newson e Krumm, 2009). Esse processo é importante nos sistemas de navegação, pois os dados de GPS podem ser imprecisos e esse processo determina onde um veículo se encontra em tempo real e assim pode determinar qual caminho este veículo deve seguir. Um tipo de algoritmo de *Map Matching* que foi utilizado neste trabalho, é o que associa a posição de um objeto a uma aresta existente na rede de ruas. Essa associação é feita pelo cálculo da distância entre o ponto e uma linha, sendo a linha definida pelos vértices origem e destino de uma aresta. A fórmula a seguir representa esse cálculo:

$$distancia(V_1, V_2, (x_0, y_0)) = \frac{(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2 y_1 - y_2 x_1}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \quad (1)$$

A linha é definida por dois pontos $V_1=(x_1,y_1)$ e $V_2=(x_2,y_2)$. (x_0,y_0) é a posição do objeto. Tantos os vértices que definem a aresta, assim como a posição do objeto, são definidos por latitude e longitude. A aresta que estiver mais próxima do ponto, será considerada como a aresta em que o objeto se encontra.

4 TRABALHOS RELACIONADOS

A seguir, são apresentados os principais trabalhos relacionados a este.

4.1 Efficient and Distributed DBScan Algorithm Using MapReduce to Detect Density Areas on Traffic Data

Coelho da Silva *et al.* (2013) apresenta um trabalho em que foi utilizada uma técnica de clusterização para o processamento de dados de tráfego na detecção de congestionamentos. Para isso, uma versão do algoritmo DBSCAN foi desenvolvida, para ser utilizada em um ambiente distribuído e, também, usando o modelo de programação *MapReduce*. Nesse trabalho, foi realizado um experimento com dados de trânsito referentes à cidade de Fortaleza – CE para comprovar a eficiência do algoritmo, assim como sua escalabilidade.

A proposta de Coelho da Silva *et al.* (2013) se assemelha a este trabalho por desenvolver uma versão do algoritmo DBSCAN, em que este busca pontos similares ou vizinhos em uma distância mínima *eps* e para cada conjunto *MinPts* de vizinhos alcançados, novos *clusters* são gerados.

Contudo, a proposta deste trabalho é diferente de Coelho da Silva *et al.* (2013), pois será implementada uma nova versão do algoritmo DBSCAN, que não irá apenas considerar a distância euclidiana entre os dois pontos para formação dos *clusters*, mas também a implementação da distância em rede de ruas como função de similaridade no processo de clusterização.

4.2 NEAT: Road Network Aware Trajectory Clustering

Han, Liu e Omiecinski (2012) apresenta um método de clusterização rápida e efetiva de objetos que se movem em uma rede de ruas. Esse método é divido em três fases de processamento: formação básica de *clusters*, junção de *clusters* e refinamento de *clusters*. Esse trabalho utiliza o algoritmo DBScan em sua terceira fase para o refinamento dos *clusters* formados nas fases anteriores. Este trabalho se assemelha ao de Han, Liu e Omiecinski (2012) por utilizar clusterização por densidade e apresentar uma solução para objetos que se movem ambientes restritos a rede.

No entanto, Han, Liu e Omiecinski (2012) consideram que os objetos móveis

possuem uma trajetória, observando o real deslocamento dos objetos ao longo do tempo. Neste trabalho, não são consideradas as trajetórias dos objetos, pois mudaria o foco do trabalho e são consideradas apenas as posições registradas por cada objeto em um dado intervalo de tempo.

4.3 NNCluster: An Efficient Clustering Algorithm for Road Network Trajectories

Roh e Hwang (2010) apresenta um estudo de como desenvolver um algoritmo de clusterização para dados de trajetória. Essas trajetórias estão relacionadas a rede de ruas e a movimentação de veículos nela. Um dos objetivos apresentados é identificar a proximidade dos objetos na rede de ruas, através do algoritmo de clusterização NNCluster, criado pelos autores. Foram realizados testes utilizando dados de trajetórias reais de Illinois, Estados Unidos. Os resultados mostram a eficiência, velocidade de execução, e qualidade dos clusters formados.

Este trabalho se assemelha ao de Roh e Hwang(2010) na utilização de técnicas de clusterização, o uso de rede de ruas e por utilizar dados geoespaciais. No entanto, o trabalho de Roh e Hwang (2010) utiliza trajetórias e busca aumentar a eficiência no cálculo da proximidade dos objetos situados na mesma via. Neste trabalho, não são consideradas as trajetórias dos objetos, mas apenas as posições registradas por cada objeto.

O uso de trajetórias modificaria o foco deste trabalho, além de que trajetórias visam estudar o comportamento dos objetos ao longo de seu deslocamento. Para este trabalho, é estudado a comparação de quando se usa distância euclidiana ou quando se usa uma rede de ruas com o algoritmo DBSCAN. A trajetória dos objetos é ignorada, pois são consideradas somente as posições de todos os objetos registradas em intervalo de tempo especifico.

5 PROCEDIMENTOS METODOLÓGICOS

As seções a seguir descrevem os procedimentos metodológicos adotados neste trabalho.

5.1 Coleta dos Dados

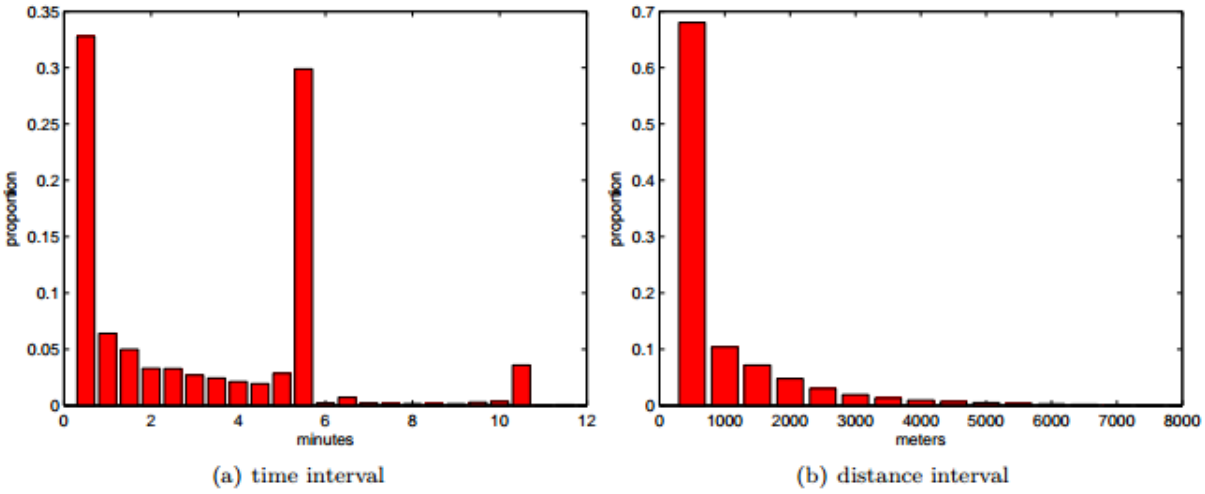
Os dados utilizados nesse trabalho foram fornecidos por Yuan at al. (2010). Os dados são formados por posições geográficas feitas por GPS de 10,357 táxis na cidade de Beijing, China. Essas posições foram coletadas no intervalo entre 2 de Fevereiro e 8 de Fevereiro de 2008. Existem no total mais de 15 milhões de posições registradas e extensão total percorrida pelos táxis chega a 9 milhões de quilômetros.

Cada registro, que é separado por vírgulas, é descrito com as seguintes informações:

taxi id, *date time*, *longitude*, *latitude*;

A Figura 4 apresenta o intervalo entre o tempo e a distância entre dois pontos consecutivos de um mesmo táxi. A média de tempo está em 177 segundos, enquanto a distância está para 623 metros. A Figura 4 (a) apresenta o intervalo de tempo e a Figura 4 (b) o intervalo de cada distância.

Figura 3: Representação dos intervalos de tempo e distância para dois pontos consecutivos

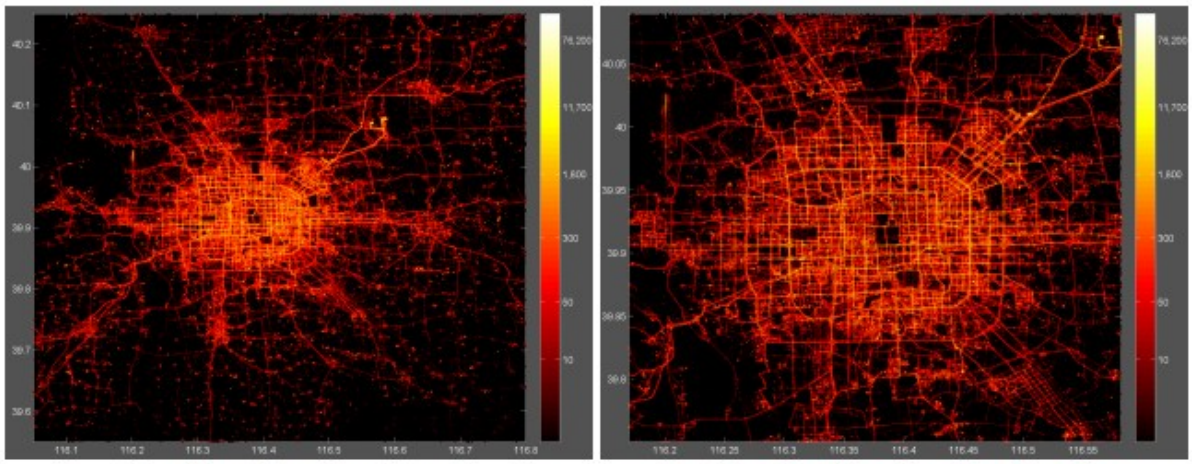


(a) time interval (b) distance interval

Fonte: Yuan *at al*. (2010)

A Figura 5 apresenta a densidade dos pontos diretamente no mapa da cidade de Beijing, China.

Figura 4: Densidade dos pontos em Beijin, China



(a) Data overview in Beijing (b) Within the 5th Ring Road of Beijing

Fonte: Yuan *at al*. (2010)

Neste trabalho, os dados serão separados em janelas de tempo, ou seja, em intervalos iguais de tempos de 7 minutos cada um, pois, de acordo com a Figura 3, é possível perceber que grande parte dos dados foram registrados nos 6 minutos de intervalo. O objetivo de utilizar dados de diferentes intervalos de tempo é perceber que a quantidade de *clusters* pode variar de tempos em tempos (dinamicidade do movimento), como pode ser observado nos experimentos.

5.2 Criação da Topologia da Rede

Nesta etapa, a Rede de Ruas será criada com dados geoespaciais extraídos do projeto *Open Street Map*[2] referentes a cidade de Beijin, China. Esses dados vem no formato *osm* e fornecem informações geográficas de cada via da cidade. Cada via é definida por uma aresta, que possui um *id*, seu custo e quais são os vértices de início e fim. Os vértices por sua vez, são definidos por um *id*, longitude e latitude.

Todos os dados são registrados em um banco de dados PostgreSQL[3]. A extensão PostGIS[4] é utilizada para que o banco possa entender o formato geoespacial dos dados.

2 www.openstreetmap.org
3 https://www.postgresql.org/
4 http://postgis.net/

A extensão pgRouting[5] é utilizada para ligar esses dados e formar a topologia da rede. Dessa forma, cada vértice sabe a qual vértice está ligado. Essas informações são importantes para o cálculo da menor distância na rede, pois o algoritmo precisa saber quais caminhos possíveis de seguir.

5.3 Map Matching

O processo de *Map Matching* é feito após a coleta dos dados e a criação da topologia da rede. Os dados possuem uma posição geográfica definida por GPS, porém, ocorre dessas posições não serem precisas e podem apontar para locais que não são zonas de tráfego. Assim, esse passo relaciona cada posição registrada à aresta mais próxima na rede de ruas. Essa associação ocorre pelo uso da fórmula apresentada na seção 3.4 (1). Os pontos são então guardados no banco de dados contendo o *id* de qual aresta ele pertence, o vértice inicial e o vértice final desta, além da distância de deslocamento entre o vértice inicial (conhecido como offset) e a posição em que esse ponto se encontra na aresta.

5.4 Implementação do DBSCAN

DBSCAN foi implementado em duas versões diferentes. A diferença entre ambas está na forma de cálculo de distância adotado. Uma versão utiliza a distância euclidiana e a outra utiliza a distância em uma rede de ruas. As duas versões foram testadas com diferentes valores de *eps* e *MinPts*, e em todos os casos o tempo de processamento foi reportado.

Os passos a seguir descrevem as duas formas de cálculo de distância que será utilizada no algoritmo DBSCAN. Para ambas versões, foi usada a mesma implementação do algoritmo DBSCAN.

5.4.1 Usando distância Euclidiana

A primeira parte desse trabalho, o algoritmo DBSCAN utiliza a distância Euclidiana, ou seja, a distância em linha reta entre dois pontos. Essa distância será obtida através do uso da fórmula (2).

5 http://pgrouting.org/

$$distância(p,q)=\sqrt{(q_1-p_1)^2+(q_2-p_2)^2} \quad (2)$$

Onde *p* e *q* são dois pontos distintos. (p_1, p_2) representam a posição do ponto *p* e (q_1, q_2) representam a posição de *q*, ambas posições definidas por latitude e longitude. Os dados usados nesse trabalho são definidos por sua longitude e latitude. A rede de ruas é ignorada aqui, pois o cálculo da distância é considera uma linha reta entre os pontos.

Se a distância entre um ponto *p* e *q* for menor ou igual que o valor de *eps*, o ponto *p* é vizinho do ponto *q*. Se *p* possuir mais vizinhos do que o número *MinPts*, será considerado um *core point* e deve ser expandido para que o cluster contenha seus vizinhos.

5.4.2 Usando distância na rede de ruas

Nessa etapa, o algoritmo de menor caminho *Dijkstra* foi utilizado para o cálculo das distâncias como pré-processamento da execução do algoritmo DBSCAN. Para realizar este cálculo, a partir de um vértice inicial, todos as distâncias entre os demais vértices da topologia da rede foram analisadas pelo algoritmo *Dijkstra*. Nesse trabalho, cada vértice foi considerado como origem e foi computada a menor distância dele na rede para todos os outros. Todos os valores de custo são armazenados em uma lista de adjacência que é acessada pela implementação do DBSCAN para rede de ruas.

Para acessar a distância entre dois pontos devem ser fornecidas a qual aresta cada um pertence. Cada aresta, como definido anteriormente, é composta por um vértice inicial e um vértice final. Os vértices iniciais dos dois pontos são usados como parâmetros para descobrir qual o menor caminho. Se nenhum caminho for encontrado ou a distância for maior do que o valor *eps*, esse ponto não é considerado vizinho.

5.5 Avaliação e Validação

Nesta etapa serão realizados os experimentos em formas de testes para avaliar e validar a comparação entre as versões implementadas. Serão avaliadas a qualidade e precisão dos resultados obtidos para cada uma das versões. Os resultados serão validados através da comparação dos resultados obtidos de cada versão em cada teste, apresentando a diferença entre ambas, quanto ao número de *clusters* e ao número de *noise points*.

6 RESULTADOS

Esta seção apresenta os resultados obtidos através de experimentos realizados com as duas versões do DBSCAN, uma que utiliza a distância euclidiana e outra que utiliza distância na rede de ruas considerando o menor caminho.

Todos os testes foram realizados na mesma máquina, cuja configuração do hardware está descrita na Tabela 1. O algoritmo DBSCAN foi implementado na linguagem Java versão 1.8. O ambiente de desenvolvimento utilizado foi a IDE Eclipse, versão *Mars* 2.

Tabela 1: Descrição de hardware

Hardware	**Configuração**
Memória RAM	16gb
Processador	Intel© Core i7-4510U CPU @ 2,00GHz
Sistema Operacional	Linux Mint 17.1 Cinnamon 64-bit

Fonte: Elaborada pelo autor

6.1 Coleta de dados

Todos os dados foram previamente coletados. Estes foram separados em 5 janelas de tempo contíguas. A primeira janela de tempo foi formada utilizando dados registrados no dia 4 de Fevereiro de 2008, a partir das 17h:30m deste dia. Cada janela de tempo possui 7 minutos de intervalo. Esta separação em janelas de tempo foi escolhida pois será possível identificar a movimentação e mudança dos *clusters* ao longo do mapa.

A Tabela 2 descreve a quantidade de pontos para cada janela de tempo criada.

Tabela 2: Descrição de cada janela de tempo

Identificação	**Número de pontos**
1	28200
2	26088
3	28304
4	24416
5	29256

Fonte: criada pelo autor

6.2 Pré-processamento

O pré-processamento corresponde a computação das distâncias em rede de todos os vértices para todos os vértices na rede. A Tabela 3 apresenta o tempo de pré-processamento de cada janela de tempo na versão que utiliza o algoritmo *Dijkstra* para o cálculo da menor distância na rede de ruas.

Tabela 3: Tempo de pré-processamento em segundos.

Janela de tempo	***Dijkstra***
1	52
2	51
3	51
4	52
5	54

Fonte: elabora pelo autor

Não é necessário o pré-processamento quando a distância é euclidiana, pois ela pode ser computada ao longo do algoritmo de DBSCAN em tempo O(1). No entanto, a computação do caminho mais curto entre dois vértices na rede pode ser custoso, neste caso, o autor preferiu realizar essa computação como pré-processamento.

6.3 Desenvolvimento e análise dos resultados

Após o pré-processamento, é realizada a execução do algoritmo DBSCAN. A execução ocorreu isoladamente, sem interrupções do usuário, e em todos os casos, não foi utilizada nenhuma forma de paralelismo, como *threads*.

Foram realizados testes para ambas as versões de implementação. A primeira fase de testes, descrita na Seção 6.3.1, foi realizada com valor fixo para *MinPts* e valores diferentes para *eps* como entrada para o algoritmo DBSCAN. A segunda fase de testes, descrita na Seção 6.3.2, foi realizada com valor fixo para eps e valores diferentes para MinPts como entrada. A Seção 6.3.3 traz uma análise sobre os resultados de dos testes.

6.3.1 Testes para MinPts fixo

Todos os testes abaixo foram realizados com valor de MinPts igual a 30.

A Tabela 4 apresenta os resultados do primeiro teste realizado para a versão com distância euclidiana para as 5 janelas de tempo. O valor de entrada para *eps* foi 0.01 e para *MinPts* foi 30. O tempo de processamento é apresentado em segundos.

Tabela 4: Resultados do teste para *eps*= 0.01 e *MinPts*=30 utilizando distância euclidiana

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise Points***
1	7.77	12	2144
2	7.53	15	2146
3	7.66	13	1994
4	6	12	2011
5	7.66	14	2223

Fonte: elaborada pelo autor

A Tabela 5 apresenta os resultados do primeiro teste realizado para a versão com distância de rede. Os valor de entrada para *eps* foi 0.01 e para *MinPts* foi 30. O tempo de processamento está em segundos.

Tabela 5: Resultados do teste para *eps*=0.01 e *MinPts*=30 utilizando distância de rede

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise Points***
1	141	9	27773
2	132	9	25760
3	156	13	27873
4	118	7	24209
5	269	11	28826

Fonte: elaborada pelo autor

O tempo de processamento para distância de rede está bem maior que para distância euclidiana. Isso é devido a implementação realizada pelo autor, em que foi utilizada uma lista de adjacência, de tal sorte que ao buscar a distância entre um vértice origem e

destino, a lista do vértice origem é percorrida até ser encontrado o custo associado para alcançar o vértice destino. A melhoria dessa implementação pode ser realizada utilizando uma matriz e é considerada como trabalho futuro. Perceba como a quantidade de *outliers* é bem maior para o DBSCAN usando distância de rede, pois é considerado o caminho entre os pontos na rede de ruas.

Os testes conseguintes visam mostrar resultados quando a versão com distância euclidiana recebe um valor *eps* com 3 casas decimais a direita, por exemplo, 0.001. Ou seja, um raio menor de cobertura de pontos. Para distância em rede de ruas, o valor de *eps* irá utilizar apenas uma casa decimal a menos na esquerda, por exemplo, 0.1. A intenção é aumentar o raio de cobertura e verificar os resultados quanto o número de *clusters* e *noise points* encontrados.

A Tabela 6 apresenta os resultados para o teste da versão com distância euclidiana com *eps*=0.001 e *MinPts*=30. A Tabela 7 apresenta os resultados com *eps*=0.002 e *MinPts*=30.

Tabela 6: Para *eps*=0.001 e *MinPts*=30.

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	12	14	27249
2	9.8	13	25444
3	12	11	27622
4	10	9	23991
5	11.82	15	28459

Fonte: elabora pelo autor

Tabela 7: Para *eps*=0.002 e *MinPts*=30

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	10.3	84	20359
2	9.71	82	19871
3	11	93	20866
4	8.54	74	18975
5	10.64	94	20615

Fonte: elabora pelo autor

Os resultados dos testes apresentados nas Tabelas 10 e 11 provam as afirmações feitas anteriormente sobre a relação entre o valor de eps, o número total de *clusters* e *noise points*. A Tabela 6 mostra um baixo valor de *clusters* e um altíssimo valor de *noise points*, resultado de um valor eps baixo. Por outro lado, a Tabela 7 apresenta um aumento no valor de *clusters* e a diminuição dos noise points, pois teve um valor eps maior.

Os dois resultados abaixo foram extraídos da versão com distância na rede de ruas. A Tabela 8 apresenta o resultado para o valor *eps*=0.1 e *MinPts*=30, enquanto a Tabela 9 teve como entrada *eps*=0.2 e *MinPts*=30.

Tabela 8: Para *eps*=0.1 e *MinPts*=30

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	180	86	8174
2	158	72	8249
3	180	73	8122
4	152	85	8598
5	206	82	7967

Fonte: elabora pelo autor

Tabela 9: Para *eps*=0.2 e *MinPts*=30

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	165	26	3721
2	164	30	3797
3	199	30	3727
4	106	32	3708
5	167	33	3644

Fonte: elabora pelo autor

Foi possível perceber através dos resultados para a versão com distância na rede de ruas, a diminuição do número de *noise points*, com aumento do valor *eps*, já que são alcançados mais objetos com o aumento do valor de *eps* (mais objetos podem ser clusterizados). O mesmo padrão pode ser observado na distância euclidiana.

6.3.2 Testes para eps fixo

A segunda etapa de testes teve como entrada *eps*=0.02 em todas as entradas e para ambas as versões. Neste caso, foi realizada a variação do valor de *MinPts*. A Tabela 10 mostra os resultados com distância euclidiana, enquanto a Tabela 11 mostra resultados com distância na rede de ruas.

Tabela 10: Resultados do teste para *eps*=0.02 e *MinPts*=30 utilizando distância euclidiana

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	14	7	961
2	12	3	1058
3	15	7	1058
4	11	4	977
5	17	4	1187

Fonte: elaborada pelo autor

Tabela 11: Resultados do teste para *eps*=0.02 e *MinPts*=30 utilizando distância de rede

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	138	54	25870
2	114	41	24528
3	137	51	26000
4	103	31	23192
5	207	65	26388

Fonte: elaborada pelo autor

Os testes abaixo foram realizados com *MinPts*=40. *Eps* continua com valor fixo igual a 0.02. A Tabela 12 mostra resultados para a versão com distância euclidiana e a Tabela 13 para a versão com distância em rede de ruas.

Tabela 12: Resultados do teste para *eps*=0.02 e *MinPts*=40 utilizando distância euclidiana

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	22	7	1133
2	11	2	1158
3	13	3	1181
4	10	3	1078
5	16	3	1308

Fonte: elaborada pelo autor

Tabela 13: Resultados do teste para *eps*=0.02 e *MinPts*=40 utilizando distância de rede

Janela de tempo	**Tempo de processamento**	**Total de *clusters***	***Noise points***
1	131	12	27322
2	138	12	25418
3	159	9	28062
4	111	7	24307
5	176	20	28013

Fonte: elaborada pelo autor

6.3.3 Análise dos Resultados

O aumento de *eps* nos primeiros testes aumentaram o número de *clusters* e diminuíram o número de *noise points*, como esperado. A razão para essas mudanças é explicada pelo maior raio de cobertura de cada ponto. A distância percorrida dentro da rede de ruas acarretou em um número maior de pontos vizinhos. E assim, na fase de expansão de *clusters* do algoritmo DBSCAN, vizinhos de vizinhos foram encontrados, ocasionando na expansão dos *clusters* encontrados.

Após todos os testes realizados para ambas as versões, foi possível encontrar a diferença na precisão dos resultados, eficiência de cada versão e o custo em tempo. A primeira observação quanto a diferença de resultados, observando os testes realizados, demonstram que o algoritmo DBSCAN com distância em rede de ruas fornece um resultado mais verídico, pois foram considerados os limites físicos do mapa real através da rede de ruas. Enquanto para a distância euclidiana, tais limites são ignorados. A eficiência é comprovada pela variação de valores nos testes. Padrões foram encontrados, como, por exemplo, que existe uma relação

inversamente proporcional entre *eps* e o número de *noise points*, ou seja, quanto maior *eps* for, menor o número de *noise points*.

O tempo de execução das duas versões diferem consideravelmente. Ignorados os custos do pré-processamento dos dados, a versão com distância euclidiana foi mais rápida comparada a versão com rede de ruas. A diferença de tempo deve-se ao fato da forma de como a distâncias são acessadas. O cálculo utilizado na distância euclidiana utiliza a fórmula (2), e tem o custo de tempo O(1). Para a distância em rede de ruas, o cálculo é feito procurando pelo menor valor na distância na lista de adjacências formada no pré-processamento. Esse caminho precisa ser verificado toda vez que dois pontos são comparados e isto tem um custo computacional maior, pois é necessário percorrer essa lista todas as vezes.

7 CONSIDERAÇÕES FINAIS

Este trabalho apresentou a implementação do algoritmo DBSCAN em uma Rede de Ruas e em movimento livre. Também apresentou uma comparação entre outra implementação, que utiliza distância euclidiana e a distância de rede para diferentes variações de *eps* e *MinPts*. Os dados utilizados foram registrados na cidade de Beijing, China. Os dados foram preprocessados e separados em janelas de tempo. Foram realizados testes que comprovam a validade do algoritmo e sua eficiência.

A realização de testes permitiu a observação e identificação de padrões na execução do DBSCAN. As variações dos valores de *eps* e *MinPts* apresentavam resultados parecidos em cada uma das versões durante a sequência testes, porém, a versão com a distância em rede de ruas teve maior custo de tempo para ser processada, o que já era esperado, pois foi utilizada uma lista de adjacências.

8 TRABALHOS FUTUROS

O algoritmo utilizado nesse trabalho foi DBSCAN com utilização de distância euclidiana para movimento livre e com distância de rede para movimentos restrito a rede de ruas. Na busca para obter melhores resultados, é possível aperfeiçoar o algoritmo diminuindo o tempo de execução utilizando técnicas para paralelização de processamento, por meio de ambientes distribuídos ou uso melhores recursos de hardware. A diminuição no tempo de execução pode tornar o algoritmo utilizável em tempo real, processando *streaming* de dados.

Outras maneiras de melhoria seria o uso de um algoritmo do cálculo de distância na rede de ruas menos custoso em tempo, por exemplo, o algoritmo A*.

O uso do modelo de programação *Map Reduce* também é considerada uma melhoria, tendo em vista que o mesmo ajuda a encontrar dados com propriedades iguais em um ambiente distribuído. O *framework* Apache Spark também representa uma melhoria, diminuindo o tempo de processamento dos dados auxiliando o DBSCAN.

REFERÊNCIAS

ALSHAWI, Imad S. et al. Lifetime enhancement in wireless sensor networks using fuzzy approach and A-star algorithm. **IEEE Sensors journal**, v. 12, n. 10, p. 3010-3018, 2012.

BARROS, Edson A. R.; PAMBOUKIAN, Sergio V. D., ZAMBONI, Lincoln C.. **ALGORITMO DE DIJKSTRA: APOIO DIDÁTICO E MULTIDISCIPLINAR NA IMPLEMENTAÇÃO, SIMULAÇÃO E UTILIZAÇÃO COMPUTACIONAL**. International Conference on Engineering and Computer Education, 2007, São Paulo, BRASIL.

BERKHIN, Pavel. **A Survey of Clustering Data Mining Techniques**. Springer Berlin Heidelberg, p 25-71, 2006.

BORAH, B.; BHATTACHARYYA, D. K. A clustering technique using density difference. In: **2007 International Conference on Signal Processing, Communications and Networking**. IEEE, 2007. p. 585-588.

CHEN, J.; LAI, Caefing; MENG, X.; Xu, Jianliang; HU, Haibo. **Clustering Moving Objects in Spatial Networks**. 2007 12th international conference on Database systems for advanced applications Pages 611-623. IEEE.

DAI, B.-R.; LIN, I.-C. (2012). **Efficient map/reduce-based DBScan algorithm with optimized data partition**. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pages 59–66. IEEE.

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

DJOJO, Michael Alexander; KARYONO, Kanisius. Computational load analysis of Dijkstra, A*, and Floyd-Warshall algorithms in mesh network. In:**Robotics, Biomimetics, and Intelligent Computational Systems (ROBIONETICS), 2013 IEEE International Conference on**. IEEE, 2013. p. 104-108.

ESTER, Martin; Kriegel, Hans-Peter; Sander, J.; Xu, Xiaowei. **A density-based algorithm for discovering clusters in large spatial databases with noise**. In: Kdd. 1996. p. 226-231.

FUHAO, Zhang; JIPING, Liu. An algorithm of shortest path based on Dijkstra for huge data. In: **Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on**. IEEE, 2009. p. 244-247.

GAVA, E. M. ; FELIPPE, G. ; MADEIRA, K. ; PALHANO, M. B. ; GARCIA, M. C. M. ; MARTINS, João P. ; SIMÕES, Priscyla. W. T. A .. **O Algoritmo Density-Based Spatial Clustering of Applications With Noise (DBSCAN) na Clusterização dos Indicadores de Dados Ambientais**. Anais SULCOMP, Vol. 6, No 1, 2012.

GIMENES, Fátima M. Pegorini; GIMENES, Régio M. Toesca; OPAZO, Miguel A. U.. **Os Processos de Integração Econômica pela Óptica da Análise Estatística de Agrupamento**. Pesquisa & Debate. Revista do Programa de Estudos Pós-Graduados em Economia Política, São Paulo, v. 16, n. 1(27), pp. 154-178, nov. 2005.

HAN, Binh; LIU, Ling; OMIECINSKI, Edward. Neat: Road network aware trajectory clustering. In: **Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on**. IEEE, 2012. p. 142-151.

JASIKA, Nadira et al. Dijkstra's shortest path algorithm serial and parallel execution performance analysis. In: **MIPRO, 2012 proceedings of the 35th international convention**. IEEE, 2012. p. 1811-1815.

NEWSON, Paul; KRUMM, John. Hidden Markov map matching through noise and sparseness. In: **Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems**. ACM, 2009. p. 336-343.

ROH, Gook-Pil; HWANG, Seung-won. Nncluster: An efficient clustering algorithm for road network trajectories. In: **International Conference on Database Systems for Advanced Applications**. Springer Berlin Heidelberg, 2010. p. 47-61.

SILVA, Ticiana L. C. da; MAGALHÃES, Regis P.; NETO, Antônio C. A.. **Efficient and Distributed DBScan Algorithm Using MapReduce to Detect Density Areas on Traffic Data**. Federal University of Ceara, Fortaleza-CE, 2013.

SOUSA, F. R. C.; MOREIRA, L. O.; MACÊDO, J. A. F.; MACHADO, J. C.. **Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. Simpósio Brasileiro de Banco de Dados** - SBBD 2010. Recife - PE, 2010.

SOUSA, Flávio R. C.; SILVA, Ticiana L. C. da ; MACÊDO, José Antônio F. de; MACHADO, Javam C.. **Análise em Big Data e um Estudo de Caso utilizando Ambientes de Computação em Nuvem**. Simpósio Brasileiro de Banco de Dados - SBBD 2013. RecifePE, 2013.

TRAN, Thanh N.; DRAB, Klaudia; DASZYKOWSKI, Michal. Revised DBSCAN algorithm to cluster data with dense adjacent clusters.**Chemometrics and Intelligent Laboratory Systems**, v. 120, p. 92-96, 2013.

YADAV, Jyoti Rani; SOMAYAJULU, Durvasula VLN; KRISHNA, P. Radha. A Scalable Algorithm for Discovering Topologies in Social Networks. In: **2014 IEEE International Conference on Data Mining Workshop**. IEEE, 2014. p. 818-827.

YIU, Man Lung; MAMOULIS, Nikos. **Clustering Objects on a Spatial Network**. International conference on Management of data, Pages 443-454, New York, NY, 2004.

YUAN, Jing et al. T-drive: driving directions based on taxi trajectories. In:**Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems**. ACM, 2010. p. 99-108.