Universidade Federal do Ceará

Campus Quixadá

Curso de Engenharia de Software

Jonas de Sousa Ferreira

UMA ABORDAGEM DE MAP MATCHING DE PONTO PARA PONTO

Quixadá, Ceará

2016

Jonas de Sousa Ferreira

UMA ABORDAGEM DE MAP MATCHING DE PONTO PARA PONTO

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Orientador: Prof MSc. Regis Pires Magalhães

Quixadá, Ceará

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)

F441a Ferreira, Jonas de Sousa.
Uma abordagem de map matching de ponto para ponto / Jonas de Sousa Ferreira. – 2016.
32 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. Regis Pires Magalhães.

1. Ruas. 2. Indexação. 3. Mapas. 4. Algoritmos. I. Título.

CDD 005.1

Jonas de Sousa Ferreira

UMA ABORDAGEM DE MAP MATCHING DE PONTO PARA PONTO

Trabalho de Conclusão de Curso submetido à Coordenação do Curso de Engenharia de Software do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Área de concentração: Computação

Aprovada em: 04 / fevereiro/ 2016

BANCA EXAMINADORA

Prof MSc. Regis Pires Magalhães
(Orientador)
Universidade Federal do Ceará (UFC)

Profª MSc. Ticiana Linhares Coelho da Silva
(Membro)
Universidade Federal do Ceará (UFC)

Profª MSc. Lívia Almada Cruz (Membro)
Universidade Federal do Ceará (UFC)

A minha família principalmente meus pais, meus tios e a Daiana.

Agradecimentos

Agradeço às pessoas que me ajudaram a vencer mais esta etapa da vida. Á Universidade Federal do Ceará - UFC - Campus Quixadá, pela oportunidade de estudos e utilização de suas instalações.

Aos professores da UFC - Campus Quixadá pelo conhecimento compartilhado.

Ao meu orientador Prof. MSc. Regis Pires Magalhães, pelo conhecimento passado, e pela orientação e apoio na realização deste trabalho.

Aos meus amigos Sérgio Filho, Adail Carvalho, João Lucas, Eduardo Gomes, Brendo Colaço, João Marcos e Rogério Carvalho por me suportarem e dividirem o mesmo teto comigo.

Aos meus amigos da turma de 2011.1: João Carlos, Laísa Morais, Rafael Ribeiro, Thiago Pereira, Guylherme Tabosa, Marcilio Valois, Edgar de Almeida, Cleiton Brito, Thiago Levi, Romário Farias, Léuson Mario, Erick Bhrener, Samuel Carvalho, Jefferson Silva, Christyan Moraes e Luis Paulo, que trilharam essa jornada comigo e dos quais nunca vou me esquecer. E aos agregados da turma de 2012.1: Ygor Reis e Matheus Souza também.

À minha futura esposa Daiana Peixoto, pelo amor e compreensão, mesmo nos tempos de difícil compreensão.

Aos meus pais, Adenir e Betânia, aos meus tios, tio Odenilton e tia Uze e à toda minha família por sempre acreditarem na importância do estudo, apoiarem as minhas decisões e pela presença, apesar da distância que nos separava.

"Com Grandes Poderes,
Vem Grandes Responsabilidades."
'Stan Lee'

Resumo

Este trabalho apresenta uma abordagem de *Map Matching* que associa ponto para ponto. Nesta abordagem busca-se, baseado em uma seleção de ponto em um mapa, obter um vértice da rede mais próximo. Assim pretende-se encontrar e implementar uma solução que realize o *Map Matching* de ponto para ponto, utilizando estratégias que possam atender a alguns parâmetros de eficácia e eficiência definidos como: bom tempo de execução, uma boa complexidade e que obtenha uma boa acurácia correspondente ao ponto indicado. Uma das estratégias encontradas para atender alguns dos parâmetros de eficiência foi o uso de índices espaciais. O trabalho também destaca a importância do algoritmo de *Map Matching* quando presente em serviços baseados em localização. Alguns exemplos destes serviços são a descoberta de pontos de interesse mais próximos da localização do usuário, planejamento de rotas, detecção de caminho mais curto, dentre outros.

Palavras-chaves: Map Matching. Rede de Ruas. Índice Espacial.

Abstract

This work presents a Map Matching approach to match point to point. This approach intends to find and implement a solution that performs the Map Matching point-to-point, using strategies that can find suitable parameters defined as: good running time, good complexity and to find a good location corresponding to a given point. One of the strategies found to achieve some of the parameters of effectiveness was the use of spatial indices. The work also highlights the importance of Map Matching algorithm when present in location-based services. Some examples of these services are the discovery of the nearest points of interest aplications user's location, route planning, shortest path detection, among others.

Key-words: Map Matching. Road Network. Spatial Index.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Atualmente é possível encontrar uma grande variedade de serviços baseados em localização que utilizam a posição do usuário como parâmetro de entrada. Serviços baseados em localização são quaisquer serviços que levam em conta a localização geográfica de uma entidade (JUNGLAS; WATSON, 2008). Os serviços baseados em localização utilizam redes de ruas para representar o mundo real. Essas redes são representadas por conjuntos de formas geométricas e pontos. São exemplos de serviços baseados em localização a descoberta de pontos de interesse mais próximos da localização do usuário, planejamento de rotas, detecção de caminho mais curto, dentre outros. Entre todos esses serviços sempre existe a necessidade de determinar-se ponto(s) na rede de ruas. No entanto, a rede de ruas possui uma representação digital que muitas vezes não possui aquele ponto exatamente determinado. Assim, é necessário que os serviços baseados em localização possuam uma etapa que a partir de um ou mais pontos fornecidos como entrada, sejam obtidos um ou mais pontos correspondentes que existam na geometria da rede de ruas subjacente e que tenham localização relativamente próxima dos pontos de entrada correspondentes. Este tipo de problema pode ser solucionado através da utilização de uma técnica chamada *Map Matching* (MM) (BERNSTEIN; KORNHAUSER, 1998).

O objetivo geral de um algoritmo MM é atribuir objetos geográficos às localizações em uma rede de ruas, ou seja, dada uma posição qualquer, o algoritmo busca determinar a localização aproximada de um ou mais pontos em uma rede de ruas. Como exemplos reais de aplicações que utilizam MM encontram-se o *GraphHopper*[1], o *OpenStreetMap*[2] (HAKLAY; WEBER, 2008) e o *Open Source Routing Machine*[3] (LUXEN; VETTER, 2011), dentre outros.

O seguinte exemplo ilustra a relevância da implementação de uma abordagem MM. Supondo que um usuário deseja encontrar os k pontos de interesse mais próximos da sua posição. Ora, se o serviço não possuir a implementação do algoritmo, o usuário terá que fornecer ao serviço um ponto válido da rede de ruas que o serviço utiliza. Obter esse ponto é algo não trivial, visto que a rede de ruas não guarda todos os pontos de um mapa, mas somente os pontos necessários para representar de forma simplificada a geometria das ruas, por questões de economia de espaço e tempo de processamento. Por outro lado, usando uma abordagem de MM, o usuário forneceria um ponto qualquer ao serviço e o algoritmo se responsabilizaria de encontrar e usar um ponto válido na rede de ruas, a ser usado como parâmetro do serviço.

[1] https://graphhopper.com/
[2] https://openstreetmap.org/
[3] http://project-osrm.org/

O objetivo principal deste trabalho é encontrar e implementar um algoritmo MM de ponto para ponto, ou seja, de um ponto especificado por um usuário para um ponto (vértice) da rede de ruas, utilizando estratégias que possam atender alguns requisitos (ou *threshold*) a serem definidos e incorporá-lo ao *framework Graphast*[4] (MAGALHÃES et al., 2015), haja visto que este *framework* ainda não possui esta funcionalidade.

Graphast é um *framework* que permite construir rapidamente aplicações em redes dependentes de tempo. *Graphast* está disponivel via API pública e dessa forma permite que novos algoritmos sejam incorporados aperfeiçoando seus serviços (MAGALHÃES et al., 2015).

A Seção 2 descreve os principais conceitos utilizados como base de conhecimento para este trabalho. Na Seção 3, encontram-se os trabalhos relacionados que auxiliaram como fonte de pesquisa. A Seção 4 mostra a metodologia deste trabalho. Na Seção 5, são apresentados e discutidos os resultados deste trabalho. A conclusão e os trabalhos futuros são descritos na Seção 6.

[4] http://www.graphast.org

2 Fundamentação Teórica

Nesta seção serão apresentados os conceitos necessários para um completo entendimento deste trabalho e logo em seguida, definições e teoremas necessários ao desenvolvimento do presente trabalho.

Definição 1: Segundo (YIU; MAMOULIS, 2004), uma **rede** é um grafo direcionado $G = \{V, E, W\}$ onde V é o conjunto de vértices, e E é o conjunto de arestas. Uma aresta estabelece a ligação entre dois vértices. W é um conjunto de pesos associados a vértices ou arestas. Um exemplo de rede é mostrado na Figura 1 onde V = {A, B, C, D, E, F}, E = {(A,B), (B,C), (C,D), (E,D), (E,F), (F,A), (B,D), (B,E)}, W = {2, 4, 3, 3, 4, 5, 2, 3}.

Definição 2: Cada aresta representada em uma rede de ruas possui uma **geometria**. Essa geometria pode ser formada por um ou mais pontos. Estes pontos que formam uma geometria são chamados *shape points*. A quantidade desses pontos vai variar de acordo com a complexidade da geometria representada. Um exemplo de geometria é mostrado na Figura 2.

2.1 Rede de Ruas

Segundo (WHITE; BERNSTEIN; KORNHAUSER, 2000) uma rede de ruas pode ser representada por um grafo. As interseções entre as ruas são representadas por vértices do grafo, enquanto que as conexões entre esses vértices é representada através das arestas direcionadas do grafo. A direção das ruas é representada através da direção das arestas, que pode ser unidirecional ou bidirecional. Além disso, vértices e arestas podem ter pesos. O peso de um vértice pode quantificar a popularidade de um ponto de interesse, por exemplo. Já o peso de uma aresta pode representar a distância entre os vértices que ela interliga na rede de rua ou o tempo necessário para percorrer a aresta.

A Figura 3 mostra como uma rede de ruas é representada por um grafo, as arestas representam as ruas e os vértices representam as intersecções entre elas.

2.2 Geometria da Rua

As ruas reais não são formadas apenas por linhas retas. Cada rua possui uma curvatura, algumas são leves outras são bem acentuadas (PINK; HUMMEL, 2008). Esse formato das ruas pode ser representado através de uma geometria, já descrita na Seção 2.1. A geometria usada na rede de rua representa um segmento de rua, assim são necessários pelo menos dois pontos para representar uma geometria (levando em consideração os

vértices da aresta). A quantidade desses pontos vai variar de acordo com a complexidade da curvatura da geometria. A Figura 4 mostra como uma rede de ruas é formada por várias geometrias (os pontos vermelhos representam os pontos da geometria de cada aresta) e como as mesmas ajudam a representar a curvatura de cada rua. Normalmente as consultas que os algoritmos fazem nas redes de ruas não incluem as geometrias, mas considera apenas vértices e arestas. Se os *shape points* fossem usados nesses algoritmos teriam mais dados para processar e isso acarretaria um custo em termos de tempo de execução e uso de memória.

2.3 Estruturas de Indexação Espacial

Uma forma comum de organizar um conjunto de dados, de forma hierárquica e ordenada é através do uso de árvores de pesquisa. As mais simples e conhecidas são as árvores binárias, como: AVL, Red Black e Splay Tree (CORMEN; LEISERSON; RONALD, 1990). Estas árvores têm uma propriedade em comum: elas são balanceadas, ou seja, todos os caminhos desde a raiz (o nó do topo da árvore) até as folhas (nós finais na árvore) possuem o mesmo comprimento. Essas estruturas permitem que operações como a busca por um elemento sejam executadas em tempo logarítmico – O($log(n)$) (CORMEN et al., 2002). Esse tipo de estrutura é empregado para uso em memória principal.

Em comum, árvores binárias são estruturas unidimensionais, isto é, elas pressupõem que a chave de pesquisa seja formada por apenas um atributo ou pela concatenação de vários atributos. Acontece que em sistemas que trabalham com informações multidimensionais (mais de um atributo), estas estruturas não são diretamente aplicáveis.

O acesso a dados multidimensionais pode ser muito mais eficiente através do uso de uma classe de métodos conhecidos como métodos de acesso multidimensionais. Um exemplo de operação que pode se beneficiar deste tipo de indexação são as operações de MM geométricas, que respondem à pergunta: "Qual dos N pontos é o ponto mais próximo do ponto P1 de coordenadas (x1,y1)?". Uma provável solução seria percorrer todos os pontos medindo sua distância ao ponto pesquisado e ao final retornar o ponto mais próximo. Se o número de pontos for muito grande esta solução pode se tornar impraticável pela lentidão do processo. As estruturas de indexação espacial particionam o conjunto de objetos de acordo com sua proximidade espacial, dessa forma reduzindo o conjunto de busca nas operações espaciais. No exemplo mencionado, o método diminuiria o número de pontos testados para uma pequena parcela dos objetos (**??**).

Diversas estruturas ou métodos de indexação são propostos na literatura, como as *R-Trees* (GUTTMAN, 1984) e as *Quad-Trees* (SAMET, 1984), cada uma com suas características e comportamentos específicos. Cada estrutura destas organiza o espaço de forma diferente, o que faz com que cada uma tenha um comportamento diferente em

relação à distribuição e às geometrias dos dados.

Entre os principais métodos de acesso espaciais pode-se citar:

- *R-Tree*: A *R-Tree* é uma árvore balanceada por altura, derivada da *B-Tree*, com ponteiros para objetos espaciais nos nós folhas. Esta propriedade permite que algumas operações, como a busca de um elemento sejam executadas em tempo logarítmico - $O(log(n))$. É uma estrutura de dados hierárquica que utiliza retângulos (chamados de MBRs - *Minimum Bounding Rectangle*) para organizar um conjunto dinâmico de objetos espaciais, de maneira que objetos colocalizados fiquem armazenados próximos uns dos outros e que haja uma redução no espaço de busca a cada nível da árvore (GUTTMAN, 1984).

  Este retângulo é formado a partir da observação dos limites geométricos mínimo e máximo do contorno do objeto, e é expresso pelas coordenadas dos seus pontos inferior esquerdo e superior direito (JR; QUEIROZ, 2005).

  Em uma *R-Tree* todas as folhas aparecem sempre no mesmo nível da árvore. Nós internos contêm a delimitação de retângulos que englobam todos os retângulos dos nós nos níveis inferiores. A Figura 5 mostra a estrutura hierárquica da *R-Tree*. (OLIVEIRA et al., 2011).

  Após a construção da *R-Tree*, outros algoritmos são utilizados para realizar buscas de objetos na árvore. Estes algoritmos percorrem a árvore recursivamente – de forma semelhante ao que ocorre na operação escolha da subárvore – comparando características dos objetos com um filtro informado e retornando os que o atendem. A busca que será utilizada neste trabalho será uma adaptação do algoritmo de vizinhos mais próximos (*nearest neighbors*), que consiste em retornar os vizinhos mais próximos de um ponto, dado um ponto p e um *threshold* de distância máxima permitida (maxDistance). Todos os objetos que estão distantes deste ponto p em no máximo o tamanho da distância maxDistance são retornados (ROUSSOPOULOS; KELLEY; VINCENT, 1995). Na adaptação deste trabalho, o algoritmo recebe ainda um parâmetro k, que indica o número de objetos mais próximos retornados.

- *R*-Tree*:

  A *R*-Tree* é uma extensão da *R-Tree* que propõe mecanismos para melhorar o tempo de busca na estrutura de dados (BECKMAN, 1990). Estes mecanismos são:

  1. Redução da sobreposição entre os MBRs, que faz com que as buscas na árvore percorram menos nós falsos, ou seja, que não contêm entradas para compor o resultado das consultas (falsos positivos);
  2. Minimização do espaço morto - área do MBR que não é coberta pelos objetos filhos - para evitar que ocorra a sobreposição;

3. Minimização do perímetro do MBR, para deixar o retângulo do MBR o mais quadrado possível, também para reduzir a sobreposição de MBRs;
4. Reinserção de objetos quando ocorre *overflow* em um nó da árvore. Esta técnica escolhe itens menos colocalizados com os demais no nó cheio e tenta uma reinserção dos mesmos na árvore, procurando locais onde eles estejam mais localmente mais próximos e haja espaço para a inserção. Caso isso ocorra, libera-se espaço para a acomodação do item que causaria a divisão, evitando-a (OLIVEIRA et al., 2010).

A R*-Tree é normalmente utilizada como comparativo de outras variações por ter um excelente desempenho. Como ocorre uma reinserção, os custos de processamento são ligeiramente maiores se comparadas aos da *R-Tree*, mas isso é compensado pelas outras vantagens que ela traz, já enunciadas anteriormente (BECKMAN, 1990). Neste trabalho, o algoritmo de busca aplicado na *R-Tree*, também é aplicado na *R*-Tree*.

A Figura 6 mostra a diferença entre a reorganização dos retângulos em três exemplos: no exemplo da esquerda há um nó raiz sobrelotado, ao centro a organização dos retângulos de uma *R-Tree* ainda prejudica a busca de um determinado ponto e à direita uma representação de *R*-Tree*, apresenta uma melhor distribuição dos retângulos (BECKMAN, 1990).

- *Quad-Tree*: A *Quad-Tree* é um tipo de estrutura de dados organizada em árvore, em que cada "nó"ou "tronco"gera sempre (e exatamente) quatro "folhas"(SAMET, 1984).

  Utilizando esta estrutura de dados para realizar a indexação espacial, cada nó corresponderá a uma região quadrada do espaço. Esta região será subdividida, novamente em quatro partes, gerando mais um nível na árvore, e assim sucessivamente, até que haja apenas um ou nenhum objeto geográfico dentro dos quadrados resultantes da subdivisão, conforme mostrado na Figura 7.

  Na Figura 7 existem três níveis (A, B e C) na *Quad-Tree*. A cada um destes níveis corresponderá um quadrado geográfico, conforme apresentado na Figura 8.

A Tabela 1 ilustra uma comparação sobre os tipos de indexação citadas acima. Nela é mostrado como cada tipo de indexação tem seu próprio conjunto de pontos fortes e fracos e tem limitações sobre outras áreas. A partir desta tabela se observará que de acordo com a necessidade e exigência de cada aplicação algum deles irá se destacar dentre os outros.

Tabela 1 – Comparação dos métodos de indexação espacial.

	R-Tree	*R*-Tree*	*Quad-Tree*
Complexidade	A criação dos índices e ajustes são mais fáceis.	A complexidade do algoritmo *R*-Tree* é ligeiramente maior se comparado a complexidade da *R-Tree*.	Ajuste é mais complexo, e definir os valores dos parâmetros de ajuste apropriados podem afetar o desempenho significativamente.
Armazenamento	Menos armazenamento é necessário, comparado a *Quad-Tree*.	Menos armazenamento é necessário, comparado a *Quad-Tree*.	Mais armazenamento é necessário pois requer um nível a mais que a *R-Tree*, para qualquer conjunto de dados.
Carga	Devido a facilidade na criação e nos ajustes, se a aplicação vai trabalhar com consultas a vizinhos mais próximos, a *R-Tree* e suas variações são mais rápidos	Devido a facilidade na criação e nos ajustes, se a aplicação vai trabalhar com consultas a vizinhos mais próximos, a *R-Tree* e suas variações são mais rápidos	Devido à complexidade de criação e ajustes, se a aplicação vai trabalhar com consultas a vizinhos mais próximos, a *Quad-Tree* é mais lenta.
Indicação de uso	Se a aplicação necessitar de muitas atualizações a *R-Tree* e suas variações podem não ser uma boa escolha	Se a aplicação necessitar de muitas atualizações a *R-Tree* e suas variações podem não ser uma boa escolha	Atualizações constantes não afetam a performance da *Quad-Tree* quando comparada a *R-Tree*

Fonte: (SARDADI et al., 2008) | (BECKMAN, 1990).

2.4 *Map Matching*

MM é definido por (JENSEN; TRADIŠAUSKAS, 2009) como um procedimento que atribui objetos geográficos à localizações em uma rede de ruas. Os algoritmos MM têm como objetivo determinar a posição de um objeto geográfico no mapa a partir de uma localização estimada por alguma tecnologia de localização ou fornecida pelo usuário, seja através de um clique no mapa ou de coordenadas geográficas enviadas por alguma tecnologia de localização, tal como GPS. A Figura 9 mostra a localização do usuário e sua respectiva correspondência em uma representação digital de uma rede de ruas.

QUDDUS et al. (2003) classificam os algoritmos MM em três categorias: geométrica, topológica e avançada. Os algoritmos MM da categoria geométrica utilizam apenas a informação geométrica entre a rede de ruas e o ponto usado para realização da correspondência (matching). Segundo BERNSTEIN D.; KORNHAUSER (1998) estes algoritmos se dividem

em três grupos:

- Ponto para ponto: realiza a correspondência da localização estimada com o vértice ou ponto da geometria mais próxima na rede de ruas (BERNSTEIN; KORNHAUSER, 1998). Os algoritmos deste grupo são rápidos e fáceis de implementar, porém apresentam muitas desvantagens quando utilizados na prática, como a sensibilidade à maneira como uma rede de ruas está representada e à quantidade de pontos em um determinado caminho, por exemplo.

- Ponto para curva: corresponde a localização estimada do usuário à curva mais próxima (BERNSTEIN; KORNHAUSER, 1998). Uma implementação simples dessa estratégia consiste em calcular a distância mínima entre a localização estimada e todas as curvas candidatas. Chega a ser perfeitamente adequada à primeira vista, mas também possui algumas desvantagens na prática, pois não leva em consideração os pontos anteriores ou a continuidade e conectividade da rota.

- Curva para curva: realiza a correspondência de uma sequencia de posições estimadas para a curva mais próxima (BERNSTEIN; KORNHAUSER, 1998). Chega a ser melhor do que as correspondências apresentadas anteriormente, porém é mais complexa e seu método consiste em receber como parâmetro um conjunto de posições estimadas em vez de uma única posição estimada.

As três abordagens estão exemplificadas na Figura 10. Nela, os pontos cinzas representam as posições estimadas (por um GPS, por exemplo), os pontos brancos representam os vértices da rede de ruas e os pontos pretos representam a correspondência realizada.

Algoritmos MM topológicos representam uma melhoria dos algoritmos geométricos pois eles utilizam, além da informação geométrica das curvas, a informação topológica da rede. Por fim, os algoritmos MM avançados são aqueles que utilizam estatísticas avançadas, matemática e técnicas de inteligência artificial tais como *Kalman Filter*, o Método de Lógica *Fuzzy* (NAJJAR; BONNIFAIT, 2005) e *Belief Theory* (QUDDUS; NOLAND; OCHIENG, 2006).

No decorrer do trabalho é dada maior ênfase aos algoritmos MM categorizados como geométricos, visto que o objetivo do trabalho é encontrar e implementar uma solução para MM de ponto para ponto, onde baseado na seleção de ponto em um mapa, busca-se obter o vértice mais próximo na rede de ruas.

2.5 *Graphast*

Graphast (MAGALHÃES et al., 2015) é um *framework* que permite construir rapidamente aplicações em redes dependentes de tempo, que são redes onde a cada curva

são definidos valores que representam o custo de ir de um vértice a outro, muito útil para modelos de sistemas de transporte. *Graphast* está disponível via API pública e dessa forma permite que novos algoritmos sejam incorporados aperfeiçoando seus serviços.

Este trabalho busca encontrar e implementar um algoritmo MM de ponto para ponto, de forma que este algoritmo possa ser incorporado ao *Graphast*.

Figura 1 – Rede.

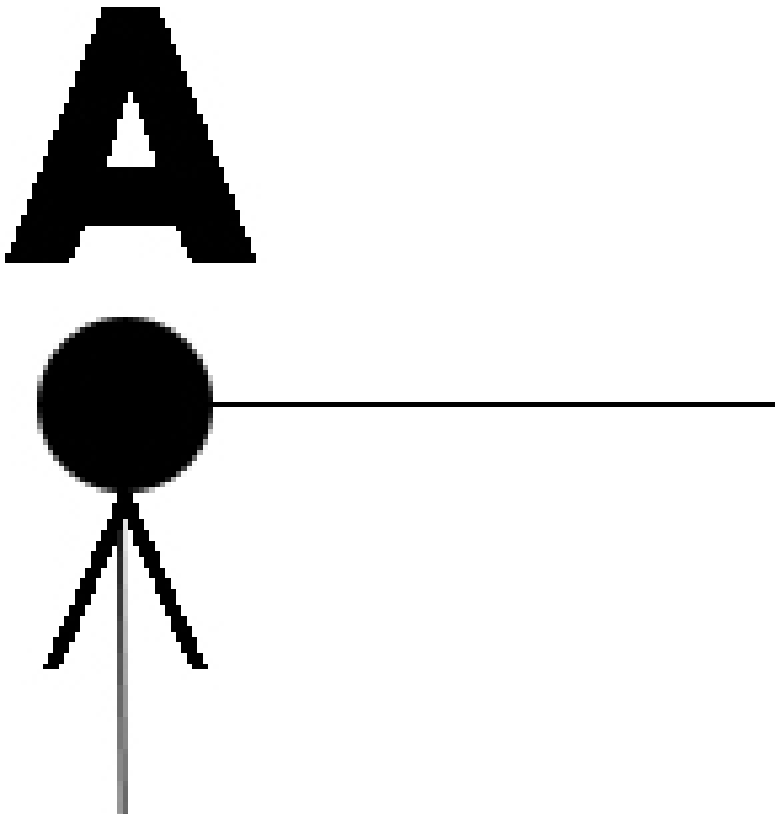
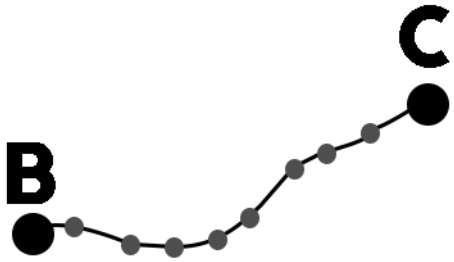
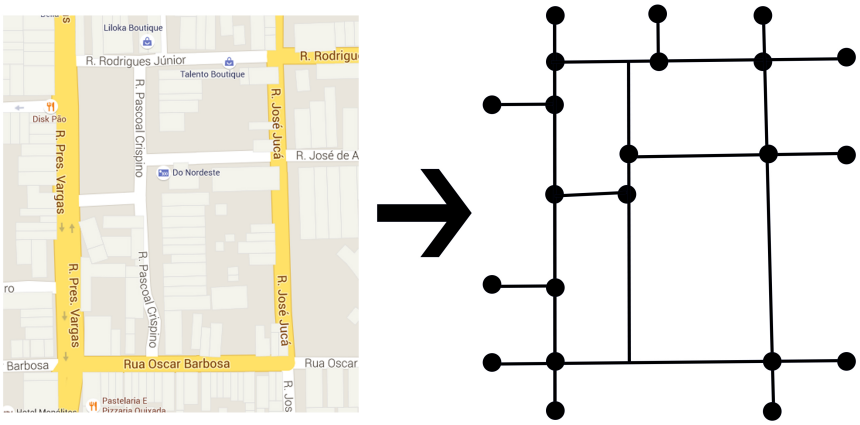


Figura 2 – Geometria.



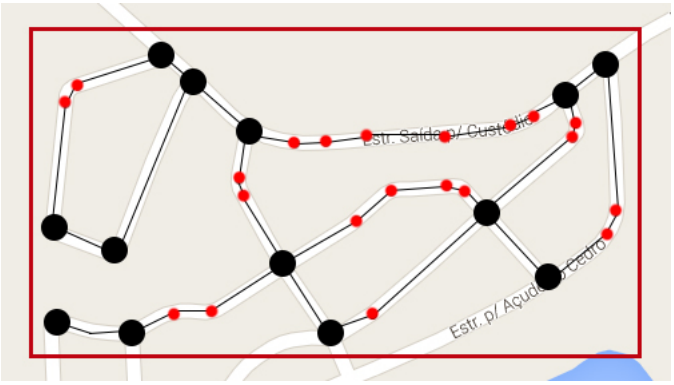
Fonte: Elaborada pelo autor.

Figura 3 – Representação em forma de grafo de um trecho de mapa.



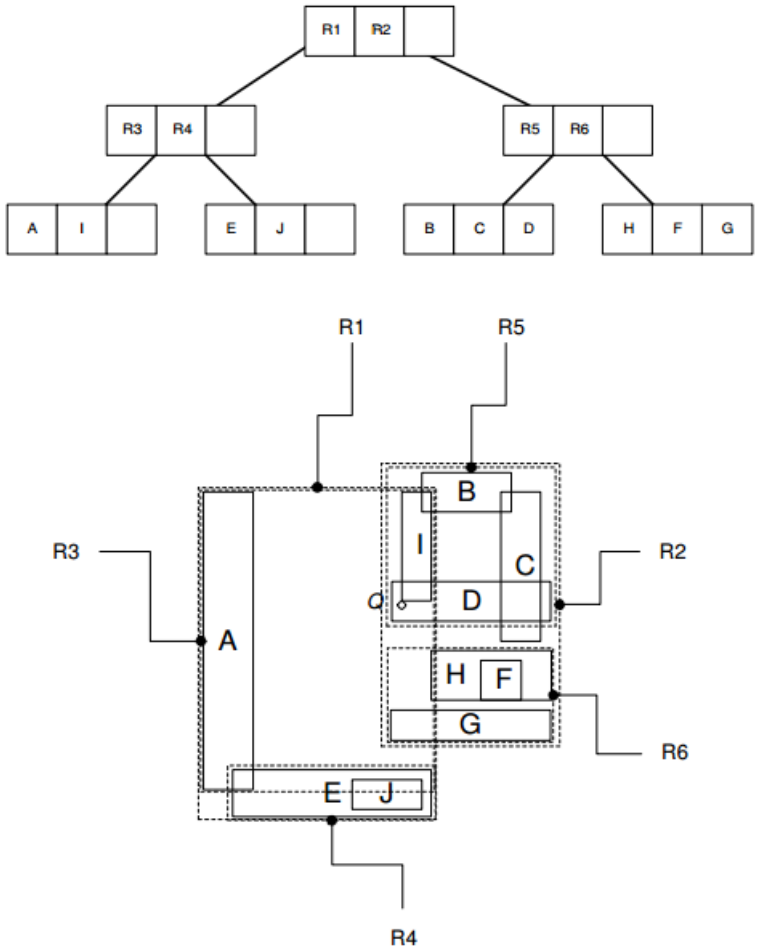
Fonte: Adaptada pelo autor a partir de Google Maps 2015.

Figura 4 – Rede de ruas Geometria da Rua.



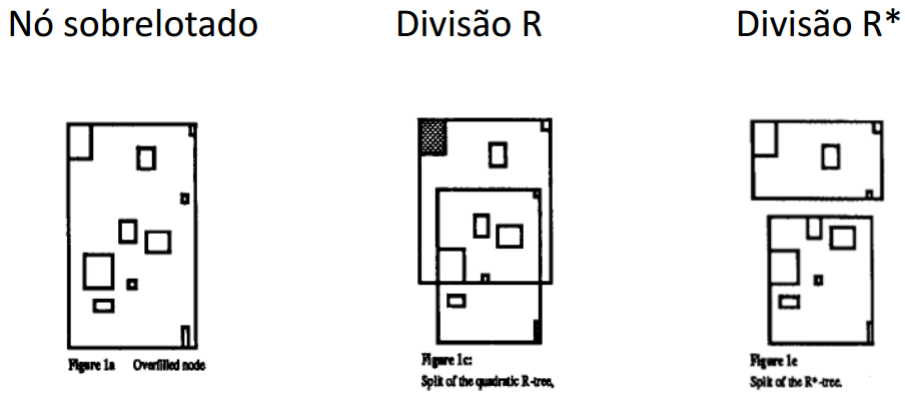
Fonte: Adaptada pelo autor a partir de Google Maps 2015.

Figura 5 – *R-Tree* e retângulos correspondentes.



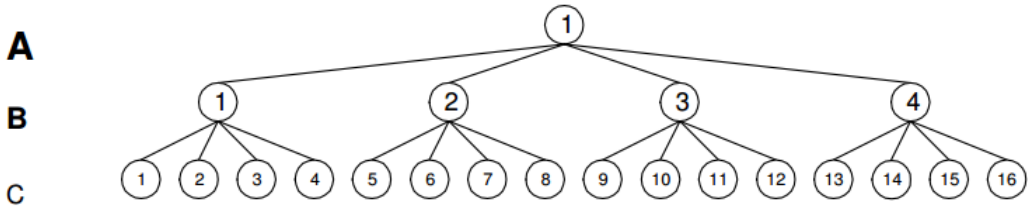
Fonte: (JR; QUEIROZ, 2005).

Figura 6 – Representação comparativa de três divisões: nó superlotado, *R-Tree* e *R*-Tree*.



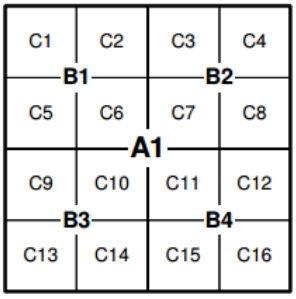
Fonte: (BECKMAN, 1990)

Figura 7 – Organização de uma *Quad-Tree*.



Fonte: (JR; QUEIROZ, 2005).

Figura 8 – Subdivisão do espaço por uma *Quad-Tree*.

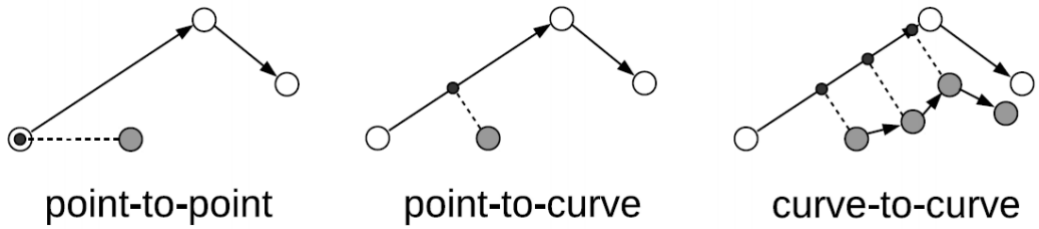


Fonte: (JR; QUEIROZ, 2005).

Figura 9 – Exemplo de MM em uma representação digital de uma rede de ruas.

Fonte: Adaptado pelo autor de *Google* 2015.

Figura 10 – Representação dos tipos de abordagens do MM geométrico.



Fonte: Adaptado pelo autor de *Google* 2015.

3 Trabalhos Relacionados

Nesta seção são descritos trabalhos que influenciaram ou que de algum modo possuem relação com os conceitos envolvidos neste trabalho, além de demonstrar pontos comuns e distintos entre entre eles e este trabalho.

O trabalho de Bernstein e Kornhauser (1998) descreve vários algoritmos e abordagens da técnica MM, são eles: *point-to-point*, *point-to-curve* e *curve-to-curve*. Os algoritmos foram abordados utilizando tanto informações geométricas quanto topológicas.

O trabalho de Bernstein e Kornhauser (1998) assemelha-se a este trabalho na busca de soluções para o problema que envolve a correspondência de pontos para uma geometria em uma rede de ruas, porém possui algumas diferenças. Em primeiro lugar, Bernstein e Kornhauser (1998) supõem que a correspondência *point-to-curve* receba uma sequência de localizações a fim de melhorar a precisão ao MM. Neste trabalho busca-se a solução para o problema em que será recebido apenas uma localização e obtêm-se um ponto correspondente na rede de ruas. Em segundo lugar, Bernstein e Kornhauser (1998) concluem que a abordagem curve-to-curve com informações geométricas e topológicas seja a melhor, assim o objetivo dessa abordagem é realizar a correspondência de um conjunto de pontos de localização do usuário a um conjunto de pontos na rede de ruas com o conhecimento da topologia da rede, porém este trabalho busca encontrar uma solução eficiente para a abordagem *point-to-point*, ou seja, realizando a correspondência de apenas um ponto selecionado no mapa a um vértice na rede. White, Bernstein e Kornhauser (2000) descrevem vários algoritmos e abordagens da técnica MM. O trabalho selecionou quatro algoritmos a serem testados em busca de uma solução para os problemas das mudanças de rotas em cruzamentos em vias urbanas (QUDDUS et al., 2003). A semelhança com este trabalho se deve ao estudo e uso das abordagens de MM, porém se diferencia devido ao uso da abordagem MM para realizar a correspondência de rotas. Neste trabalho o objetivo é relacionar apenas um ponto com uma posição válida na rede de ruas.

Outro trabalho é o *GraphHopper*[1] é uma biblioteca de roteamento de ruas escrito em Java e que fornece uma interface web chamada *GraphHopper Maps*. Por padrão, ele utiliza os dados do *OpenStreetMap* para definir as redes de ruas e os dados do *Shuttle Radar Topography Mission* para definir as informações do relevo. O *GraphHopper* se assemelha a este trabalho pelo fato de utilizar um método de indexação espacial para solucionar o problema de MM.

A fim de mostrar a relevância do algoritmo MM e auxiliar na construção de novos serviços e ferramentas que precisem realizar MM, este trabalho busca implementar no

[1] https://graphhopper.com/

framework Graphast (o qual não possui o algoritmo MM implementado) um algoritmo MM do tipo *point-to-point*, onde a partir de um ponto qualquer recebido, o algoritmo busca o vértice mais próximo na rede.

A Tabela 2 estabelece uma relação deste trabalho com seus trabalhos relacionados mencionados anteriormente.

Tabela 2 – Comparativo entre abordagens dos trabalhos relacionados.

Autor(es)/Ano	Abordagem	Procedimento
(BERNSTEIN; KORNHAUSER, 1998), (WHITE; BERNSTEIN; KORNHAUSER, 2000)	Correspondência *point-to-point*	Realiza a correspondência entre uma posição recebida com o vértice mais próximo.
(BERNSTEIN; KORNHAUSER, 1998), (WHITE; BERNSTEIN; KORNHAUSER, 2000)	Correspondência *point-to-curve*	Realiza a correspondência entre uma posição recebida com a curva/arco mais próximo.
(BERNSTEIN; KORNHAUSER, 1998), (WHITE; BERNSTEIN; KORNHAUSER, 2000)	Correspondência *curve-to-curve*	Realiza a correspondência entre uma posição recebida com a curva/arco mais próximo.
(KARICH, 2016)	Correspondência *point-to-point*	Realiza a correspondência entre uma posição recebida com o vértice mais próximo, utilizando a estrutura de indexação espacial *Quad-Tree*.
Este trabalho	Correspondência *point-to-point*	Realiza a correspondência entre uma posição recebida com o vértice mais próximo, utilizando a estrutura de indexação espacial *R-Tree*/*R*-Tree*.

Fonte: Elaborada pelo autor.

4 Solução *Map Matching*

O objetivo deste trabalho é desenvolver uma solução eficiente de MM onde dado um ponto, o algoritmo deve retornar o vértice ou ponto da geometria mais próximo. Essa solução deve atender aos seguintes parâmetros de eficiência e eficácia: tempo de resposta às consultas de busca, complexidade do algoritmo de busca do tipo de indexação e o retorno deve ser o vértice mais próximo do ponto dado, como dito anteriormente.

Baseado em todas as informações recolhidas, foi identificada uma estratégia para desenvolver a solução MM proposta neste trabalho. Esta estratégia consiste em utilizar uma implementação de indexação espacial dentre as que foram citadas na seção 2.3 e que melhor se adeque a necessidade da proposta do trabalho. O objetivo da utilização de uma implementação de indexação espacial consiste em buscar, de forma otimizada, o vértice que possua a menor distância euclidiana do ponto consultado, ou seja, que esteja mais próximo do ponto escolhido pelo usuário.

No contexto de indexação espacial foi utilizada uma biblioteca de software livre: *R-Tree*[1], desenvolvido por David Moten. Esta biblioteca implementa os métodos de acesso para o índice *R-Tree* e *R*-Tree*. Ela foi selecionada pois está implementada em Java, está bem documentada e encontra-se disponível em um repositório público.

Além dela, a aplicação *GraphHopper*, foi explorada a fim de identificar qual tipo de indexação espacial era utilizado para obter o ponto mais próximo no grafo de um ponto informado. Apesar de seus métodos não poderem ter sido utilizados na solução devido ao forte acoplamento com sua biblioteca, a aplicação pode ser usada para comparação da sua solução com a obtida neste trabalho.

Dessa forma foi possível obter soluções MM utilizando os principais métodos de acesso espaciais citados na seção 2.3.

O Algoritmo 1 mostra o pseudo-código da solução que tem como objetivo localizar no índice espacial o vértice mais próximo do ponto informado. Assumimos que os dados do grafo já estão organizados na *R-Tree*/*R*-Tree* e esta é uma variável global. Assim passamos como entrada apenas a latitude e a longitude do ponto a partir do qual será feita a busca na *R-Tree*/*R*-Tree*. O método Ponto(), recebe como parâmetros uma latitude e uma longitude e retorna um ponto criado com as coordenadas passadas. Outro método utilizado é o PontoMaisPróximo() que recebe como parâmetros o ponto informado, uma quantidade máxima de pontos a serem obtidos e um *threshold* de distância máxima, seu objetivo é percorrer o retângulo que abrange o ponto informado na *R-Tree*/*R*-Tree* e retornar a quantidade máxima determinada de vértices com a menor distância euclidiana

[1] https://github.com/davidmoten/rtree

do ponto informado. Essa busca é realizada apenas dentro do limite estabelecido que foi recebido com o parâmetro da distância máxima. Caso o nenhum ponto dentro desse limite seja encontrado, o limite ao redor do ponto é duplicado e faz-se uma nova busca na árvore com o mesmo propósito. Enquanto não forem encontrados pontos em volta do ponto informado o limite ao redor do ponto é duplicado novamente até que tenha sido retornado o ponto mais próximo. Baseado nos experimentos mostrados no Capítulo 5, foi concluído que para obter um tempo de resposta eficiente do algoritmo, é indicado utilizar uma distância máxima menor ou igual a 10 mt, pois dessa maneira o algoritmo irá percorrer uma área com menos pontos a fim de obter mais rapidamente o ponto correspondente.

Algoritmo 1: Encontre o vértice mais próximo do ponto dado no índice espacial

Entrada: lat, lon
Saída: Vértice no indice espacial mais próximo do ponto dado

```
1  início
2      pontoConsultado ← Ponto(lat, lon)
3      distanciaMaxima ← 0.01
4      quantidadeMaxima ← 1
5      resultado ←
         PontoMaisProximo(pontoConsultado, distanciaMaxima, quantidadeMaxima)
6      while resultado = null do
7          distanciaMaxima ← distanciaMaxima * 2
8          resultado ←
             PontoMaisProximo(pontoConsultado, distanciaMaxima, quantidadeMaxima)
9      end
10 fim
11 retorna resultado
```

O próximo capítulo apresenta os experimentos realizados com as soluções implementadas e seus devidos resultados.

5 Experimentos e Resultados

5.1 Experimentos

Os experimentos foram realizados numa máquina com um processador Intel Core i5 - 2.5 *Gigahertz*, 4 *GigaBytes* de memória principal e um disco rígido de 500 *GigaBytes*. O sistema operacional usado foi o *Windows 10*, executando os serviços básicos.

Os experimentos realizados foram divididos em dois casos: 1) consulta em uma área pequena (raio = 10 mt) onde são realizadas muitas interações até a obtenção do ponto correspondente (distanciaMaxima >= 10 mt) e 2) consulta em uma área grande (raio = 100 km) a fim de obter o ponto correspondente na primeira interação (distanciaMaxima >= 100 km).

Como parâmetro de medida da eficiência, foi tomado o tempo médio de resposta às consultas espaciais onde foram realizadas três execuções de cada solução. Em cada execução a Java Virtual Machine (JVM) era iniciada e ao terminar a consulta, a JVM era finalizada. Outro parâmetro de medida de eficiência tomado foi a complexidade da busca do tipo de indexação usado na solução e por último o resultado da solução deve ser o vértice ou ponto da geometria mais próximo.

A Tabela 3 mostra os conjuntos de dados que foram utilizados como carga de trabalho dos experimentos.

Tabela 3 – Conjuntos de dados utilizados nos experimentos.

Dados	Mapa	Quantidade de Vértices	Ponto Informado (Lat,Lon)	Fonte
#1	Massachusetts	416.277	42.45918, -71.5052	(GEOFABRIK, 2016)
#2	Suécia	995.759	59.97563, 17.14965	(GEOFABRIK, 2016)
#3	Berlin	65.923	52.50188, 13.41573	(GEOFABRIK, 2016)
#4	Brandenburg	320.715	53.07381, 12.66277	(GEOFABRIK, 2016)

Fonte: Elaborada pelo autor.

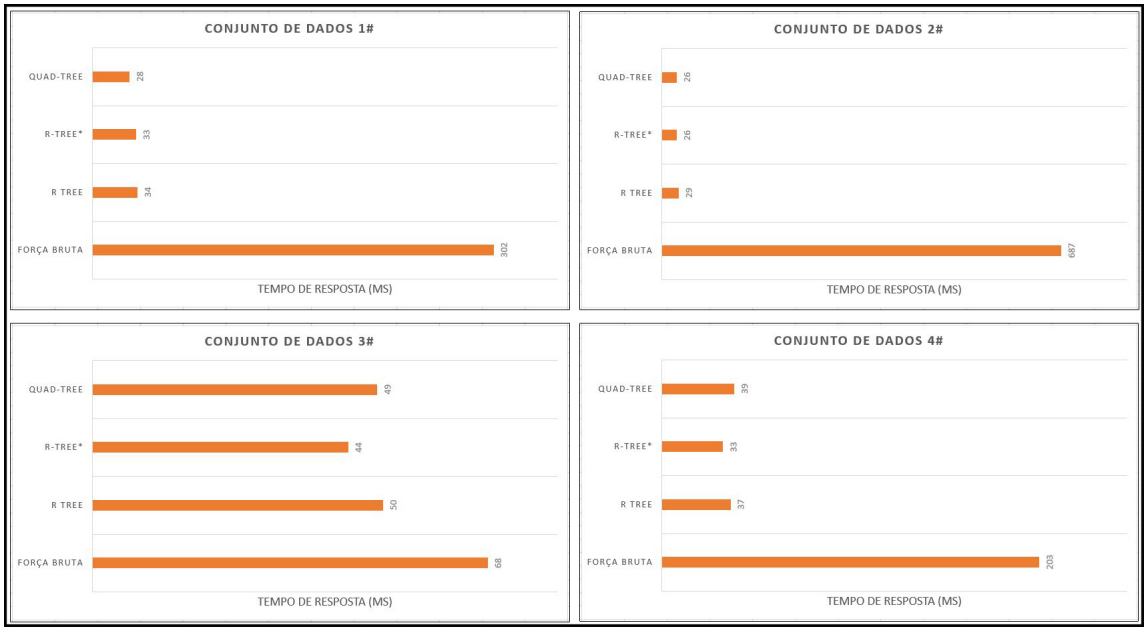
5.2 Resultados

A Figura 11 e a Figura 12 mostram, respectivamente, os resultados obtidos para os experimentos no caso 1 e no caso 2 descritos na seção 5.1. Nelas, são apresentadas o método utilizado, a complexidade do algoritmo de busca da estrutura de indexação utilizada e sua referência. Abaixo é mostrada a média dos tempos de teste das três execuções realizadas sobre cada tipo de indexação correspondente a cada conjunto de dados. Além da solução elaborada neste trabalho (soluções que utilizam *R-Tree* e *R*-Tree* como métodos de indexação espacial) também serão apresentados os resultados do algoritmo de força bruta - que compara a distância euclidiana do ponto informado com todos os pontos que pertencem ao grafo - e com o algoritmo implementado pelo *GraphHopper* que segundo (KARICH, 2016) baseia-se na implementação da estrutura de indexação *Quad-Tree*.

Com base nos experimentos realizados, é possível observar que a abordagem MM desenvolvida com a utilização dos índices espaciais tornaram a consulta mais rápida comparada ao algoritmo de força bruta. Baseado tanto no resultado dos tempos de execuções como na complexidade das soluções que utilizaram índice espacial. Além disso, em todos os experimentos realizados, o valor retornado foi o vértice mais próximo baseado na distância euclidiana.

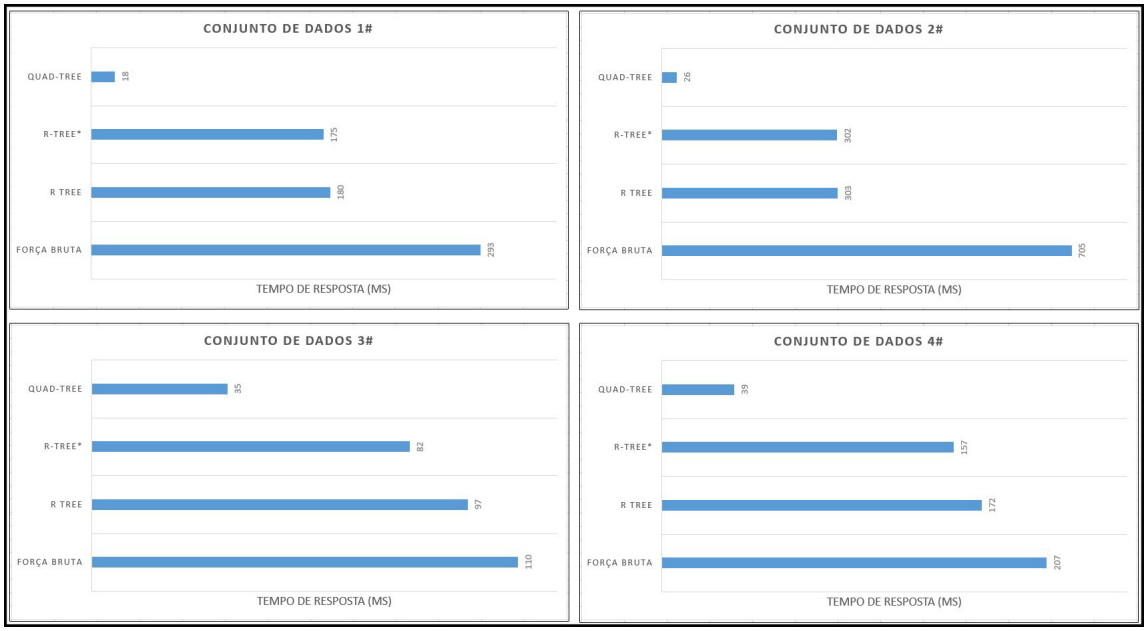
Ambos os experimentos mostraram que o algoritmo desenvolvido neste trabalho mostrou uma grande vantagem em relação ao algoritmo de força bruta. O experimento referente ao caso 1, onde foi realizada uma consulta em uma área pequena a fim de obter muitas interações até a obtenção do ponto correspondente (distanciaMaxima <= 10 mt). Essa estratégia foi mais eficiente comparada ao caso 2, pois a área de busca ao ponto correspondente é reduzida e temos um tempo de resposta melhor. O algoritmo *Quad-Tree*, implementado pelo *GraphHopper*, manteve seu desempenho em ambos os casos, apesar de não ser possível alterar o valor da distância percorrida pelo algoritmo. Como trabalho futuro pretende-se melhorar ainda mais o desempenho da estratégia proposta neste trabalho. Quando temos mais dados, o tempo de resposta do algoritmo de força bruta aumenta, como esperado. Ao contrário do método proposto, que parece vem a ser escalável e não aumenta muito com o aumento da quantidade de dados. É importante destacar que não foi considerado o tempo de criação do índice, pois ele pode ser utilizado nas outras consultas.

Figura 11 – Experimentos referente ao caso 1 (distanciaMaxima <= 10 mt)



Fonte: Elaborada pelo autor.

Figura 12 – Experimentos referente ao caso 2 (distanciaMaxima >= 100 mt)



Fonte: Elaborada pelo autor.

6 Conclusão e Trabalhos Futuros

O principal objetivo deste trabalho foi implementar uma solução que realizasse o MM, utilizando estratégias que pudessem atender alguns parâmetros de eficacia definidos como: bom tempo de execução, uma boa complexidade e que buscasse uma boa localização correspondente ao ponto indicado. Uma estratégia encontrada foi o uso de índices espaciais que mostrou-se uma ferramenta de ganho de produtividade excelente, pois trouxe ganho de tempo ao algoritmo e poupou a máquina de executar processos longos e demorados.

Dessa forma também possível compreender a importância de aplicar o MM utilizando tanto informações geométricas quanto informações topológicas, como foi realizado ao se limitar a distância percorrida pelo algoritmo e fazer as comparações de distância baseado na informação geométrica da rede.

A solução encontrada atende a proposta do trabalho mas pode ser melhorada em alguns pontos. Primeiramente, o algoritmo percorre apenas os vértices, de forma que ele não compara a distância do ponto indicado com os pontos da geometria. Assim, como trabalho futuro, pode-se armazenar no índice espacial tanto os vértices quanto os pontos da geometria, maximizando a precisão do ponto correspondente retornado.

Em segundo lugar, a solução proposta não atende a todos os casos. Por exemplo, se houver um obstáculo (ex. rio, muro, ponte, etc.) entre o ponto indicado na rede de ruas e o ponto correspondido, se a consulta se basear apenas na distância euclidiana entre os dois, o dado estaria incorreto pois o ponto correspondente correto deve ser acessível pelo ponto indicado na rede de ruas. Como trabalho futuro, pode-se pesquisar uma forma de calcular o ponto correspondente utilizando não somente a distância euclidiana, uma forma de descobrir essas outras formas seria analisar o *GraphHopper* e seus métodos de busca do ponto mais próximo. Também pode-se usar ou implementar uma solução de *Quad-Tree* genérica e eficiente para que possa ser usada no *Graphast*, caso realmente possua um desempenho melhor que as demais estratégias. Além disso, ainda é preciso incorporar a solução encontrada ao *framework Graphast*.

Referências

BECKMAN, N. Hans-peter begel, ralf schneider. *The R*-tree: An Efficient and Robust Access Method for Points and Rectangles*, ACM, v. 19, n. 2, p. 322–331, 1990.

BERNSTEIN, D.; KORNHAUSER, A. An introduction to map matching for personal navigation assistants. 1998.

CORMEN, L.; LEISERSON, C. E.; RONALD, L. *Rivest (1990) Introduction to Algorithms*. [S.l.]: MIT Press, McGraw Hill, 1990.

CORMEN, T. et al. Algoritmos (tradução da 2a. edição americana). *Editora Campus*, 2002.

GEOFABRIK. 2016. Disponível em: <http://www.geofabrik.de/>.

GUTTMAN, A. *R-trees: a dynamic index structure for spatial searching*. [S.l.]: ACM, 1984. v. 14.

HAKLAY, M.; WEBER, P. Openstreetmap: User-generated street maps. *Pervasive Computing, IEEE*, IEEE, v. 7, n. 4, p. 12–18, 2008.

JENSEN, C. S.; TRADIŠAUSKAS, N. Map matching. In: *Encyclopedia of Database Systems*. [S.l.]: Springer, 2009. p. 1692–1696.

JR, C. D.; QUEIROZ, G. Métodos de acesso para dados espaciais. *Casanova, MA; Câmara, G.; Davis Jr., CA*, p. 213–231, 2005.

JUNGLAS, I. A.; WATSON, R. T. Location-based services. *Communications of the ACM*, ACM, v. 51, n. 3, p. 65–69, 2008.

KARICH, P. 2016. Disponível em: <https://github.com/graphhopper/graphhopper>.

LUXEN, D.; VETTER, C. Real-time routing with openstreetmap data. In: ACM. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. [S.l.], 2011. p. 513–516.

MAGALHÃES, R. et al. Graphast: An extensible framework for building applications on time-dependent networks. In: *Proceedings of the 23Rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. [S.l.: s.n.], 2015.

MOTEN, D. 2016. Disponível em: <https://github.com/davidmoten/rtree>.

NAJJAR, M. E. E.; BONNIFAIT, P. A road-matching method for precise vehicle localization using belief theory and kalman filtering. *Autonomous Robots*, Springer, v. 19, n. 2, p. 173–191, 2005.

OLIVEIRA, T. B. d. et al. Dsi-rtree-um índice r-tree distribuído escalável. Universidade Federal de Goiás, 2010.

OLIVEIRA, T. B. de et al. Dsi-rtree-um índice r-tree escalável distribuído. *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2011.

PINK, O.; HUMMEL, B. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In: IEEE. *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*. [S.l.], 2008. p. 862–867.

QUDDUS, M. A.; NOLAND, R. B.; OCHIENG, W. Y. Effects of navigation sensors and digital map quality on performance of map-matching algorithms. In: *Transportation Research Board 85th Annual Meeting*. [S.l.: s.n.], 2006.

QUDDUS, M. A. et al. A general map matching algorithm for transport telematics applications. *GPS solutions*, Springer, v. 7, n. 3, p. 157–167, 2003.

ROUSSOPOULOS, N.; KELLEY, S.; VINCENT, F. Nearest neighbor queries. In: ACM. *ACM sigmod record*. [S.l.], 1995. v. 24, n. 2, p. 71–79.

SAMET, H. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, ACM, v. 16, n. 2, p. 187–260, 1984.

SARDADI, M. M. et al. Choosing r-tree or quadtree spatial data indexing in one oracle spatial database system to make faster showing geographical map in mobile geographical information system technology. *World Academy of Science Engineering and Technology*, v. 46, p. 249–257, 2008.

WHITE, C. E.; BERNSTEIN, D.; KORNHAUSER, A. L. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 8, n. 1, p. 91–108, 2000.

YIU, M. L.; MAMOULIS, N. Clustering objects on a spatial network. In: ACM. *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. [S.l.], 2004. p. 443–454.