UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Um Algoritmo para Definir o Sentido de Ônibus

Vitor de Lima Albuquerque Alves

Orientador

Márcio de Oliveira Barros

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2016

Um Algoritmo para Definir o Sentido de Ônibus

Vitor de Lima Albuquerque Alves

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Prof. Márcio de Oliveira Barros, D.Sc. (UNIRIO)

_____

Profª Geiza Maria Hamazaki da Silva, D Sc. (UNIRIO)

_____

Profª Kate Cerqueira Revoredo, D Sc. (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

DEZEMBRO DE 2016

Agradecimentos

Agradeço aos meus pais pelo esforço, dedicação e carinho durante todos os anos. Sempre priorizando a minha educação e me apoiando. A minha irmã, por sempre ser a principal companheira para todas as ocasiões. Sei que logo estaremos na sua formatura.

Aos colegas da faculdade e todos os professores que participaram de cada etapa dessa importante fase da minha vida. Foram essenciais para o meu crescimento.

Aos meus amigos que estão sempre comigo, seja pessoalmente, seja em pensamento.

Ao Professor Márcio Barros que sempre me ajudou em cada passo e realizou um excelente de trabalho de orientação com muita dedicação e organização. Um muito obrigado.

RESUMO

Com a disponibilização de dados abertos e públicos, soluções que melhoram a qualidade de vida das pessoas podem ser desenvolvidas na forma de sistemas e aplicativos. Uma gama de opções é aberta para a criatividade de pessoas e organizações, que não necessitam ficar reféns das opções disponibilizadas pelo Governo. O objetivo desse projeto é descrever a utilização dos dados públicos abertos, disponibilizados pela Prefeitura da Cidade do Rio de Janeiro, e definir um algoritmo que indique o sentido que está sendo percorrido pelos ônibus de uma linha ao longo do tempo. Com esse algoritmo, esperamos viabilizar os próximos desenvolvimentos para trabalhar com dados de mobilidade urbana e colher informações úteis sobre as linhas, como tempo médio de deslocamento, tempo médio de espera entre dois veículos, entre outros exemplos de informações que podem ser úteis à população. Com essas informações, é possível verificar o desempenho das linhas de ônibus e se ações tomadas no sentido de melhorar o trânsito estão tendo o retorno esperado. No desenvolvimento do algoritmo proposto e do seu conjunto de ferramentas foram utilizados a linguagem de programação Java e o Google Maps API.

Palavras-chave: Algoritmos exatos, Mobilidade urbana, Linhas de ônibus

ABSTRACT

By providing open and public data, solutions that improve people's quality of life can be developed in the form of systems and applications. A range of options is open to the creativity of individuals and organizations, which do not need to comply and accept the options made available by the Government. The objective of this project is to describe the use of open public data, made available by the City of Rio de Janeiro, and to define an algorithm that indicates the direction that buses are traveling over time. With this algorithm, we hope to enable further developments using urban mobility data and collect useful information about the buses lines, such as average travel time, average waiting time between two vehicles, as well as other examples of information that may be useful to the population. With this information it is possible to verify the performance of bus lines and whether actions taken to improve traffic are having the expected return. The Java programming language and Google Maps API were used in the development of the proposed algorithm and its supporting toolset.

Keywords: Exact algorithms, urban mobility, bus lines

Índice

Índice de Figuras

1 Introdução

1.1 Motivação

Estamos vivendo um tempo onde o engajamento da população sobre assuntos políticos está cada vez mais em pauta e o anseio por transparência na política é um dos principais pedidos. Governos tentam atender essa reivindicação com propostas que permitem a consulta de informações públicas. Dentre elas, uma que vem ganhando destaque e pode ser facilmente empregada e manipulada em benefício da população é a disponibilização de todo o tipo de dado que esteja em posse do poder público.

Com a disponibilização de dados abertos e públicos, soluções que melhoram a qualidade de vida das pessoas em geral podem ser desenvolvidas como sistemas e aplicativos. Uma gama de opções é aberta para a criatividade de pessoas e organizações que não necessitam ficar reféns das opções disponibilizadas pelo Governo.

Um dos assuntos políticos mais discutidos e que é diretamente afetados pela disponibilização dos dados públicos é a mobilidade urbana, algo que está sempre em discussões em debates eleitorais, reclamações em conversas entre usuários, notícias em telejornais e revistas, além de outros meios de comunicação. Isso se deve a sua grande importância para a maior parte da população das metrópoles, onde as pessoas podem gastar parte considerável de suas vidas dentro desses meios de transporte público. Por isso, estudos e ações relacionados a essa área sempre são desejados para que a qualidade de vida coletiva seja melhorada.

1.2 Objetivos

O objetivo desse projeto é descrever a utilização dos dados públicos abertos disponibilizados pela Prefeitura da Cidade do Rio de Janeiro e definir um algoritmo que indique o sentido que está sendo percorrido pelos ônibus de uma linha ao longo do tempo. Com esse algoritmo esperamos viabilizar os próximos desenvolvimentos para

trabalhar com esses dados e colher informações úteis sobre as linhas, como tempo médio de deslocamento, tempo médio de espera entre um ônibus e outro, entre outros. Com essas informações é possível verificar o desempenho das linhas de ônibus e se as ações tomadas no sentido de melhorar o trânsito estão tendo o retorno esperado.

1.3 Estrutura do Texto

O presente trabalho está estruturado em capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo II: Neste capítulo introduziremos a situação geral dos dados disponíveis sobre os transportes públicos na cidade do Rio de Janeiro, seus controles, seu funcionamento e seu impacto no cotidiano da população. Também explicaremos a utilização e extração de dados públicos fornecidos pela Prefeitura do Rio de Janeiro;

- Capítulo III: Apresentamos uma análise sobre o algoritmo de definição de sentido do ônibus de uma linha. Detalhamos trechos do algoritmo e descrevemos a lógica para cada um dos trechos;

- Capítulo IV: No quarto capítulo apresentamos os resultados do algoritmo e detalhamos o plano de teste para certificar a sua exatidão. Também nesse capítulo mostramos algumas análises do resultado gerado pelo projeto com base em algumas aplicações como exemplo;

- Capítulo V: Concluímos o estudo neste último capítulo, reunindo as considerações finais, assinalando as contribuições da pesquisa e apresentando algumas sugestões para aprofundamento posterior no projeto apresentado.

2 Dados sobre Transporte Público na Cidade do Rio de Janeiro

Neste capítulo introduziremos a situação geral dos dados disponíveis sobre os transportes públicos na cidade do Rio de Janeiro, seus controles, seu funcionamento e seu impacto no cotidiano da população. Também explicaremos a utilização e extração de dados públicos fornecidos pela Prefeitura do Rio de Janeiro.

2.1 Transporte público

De acordo com Borges (1), o transporte público pode ser definido como um meio de transporte que é proporcionado pelo poder público e que atende a toda população sem distinção. Eles são de suma importância para a produção econômica e o desenvolvimento de uma cidade. Com um transporte público ineficiente ou escasso, a maioria dos trabalhadores não teria condições de se deslocar e, ainda que utilizassem o transporte individual, os engarrafamentos seriam constantes e inviabilizariam a locomoção, causando um impacto negativo em toda a sociedade.

Por se tratar de uma grande metrópole com mais de seis milhões de habitantes no município e onze milhões em sua área metropolitana, os sistemas de transporte em massa são altamente necessários e precisam ser discutidos e melhorados constantemente no Estado do Rio de Janeiro. Por ter uma grande extensão territorial, a região metropolitana possui quatro tipos de sistemas de transporte: metrô, ferrovias, barcas e ônibus.

Os sistemas metroviário, ferroviário e hidroviário são fornecidos e operados por empresas privadas e não possuem uma interligação eficiente e efetiva. Essas iniciativas privadas que fornecem os serviços de transporte acabam trabalhando muitas vezes de forma independente umas das outras. Com isso, a construção de um único sistema

metropolitano de transporte se torna difícil. A falta de comunicação acaba sendo o principal motivo para que isso aconteça.

A Cidade do Rio de Janeiro possui trezentos e seis quilômetros de trilhos e cento e quarenta e três estações divididas entre os sistemas metroviário e ferroviário. A concessionária responsável pela operação e manutenção das linhas do metrô é a MetrôRio, que conseguiu a concessão no ano de 1997. Já a empresa vencedora da licitação sobre a malha ferroviária foi a SuperVia.

Atualmente as três linhas de metrô movimentam diariamente cerca de setecentos e oitenta mil pessoas, sendo considerado o segundo meio de transporte de massa mais utilizado. A construção das linhas se iniciou no ano de 1970 e a partir dessa data não houve grandes investimentos para que estações fossem distribuídas para atender a maior parte da população do município. Os investimentos são demorados e atendem uma pequena parte da população. Embora esses problemas sejam verdade no município carioca, o metrô pode ser considerado o meio de transporte mais eficiente na maioria das grandes metrópoles mundiais.

Através das suas linhas, a malha ferroviária do Estado do Rio de Janeiro transporta setecentas mil pessoas por dia em média. Essas viagens são realizadas através de nove linhas e cento e duas estações, divididas em doze municípios. Os trens responsáveis pela locomoção têm recebido grandes investimentos e, por isso, a grande maioria possui ar refrigerado e condições mínimas para viagens de média e longa distância.

Embora possua opções suficientes e sustentáveis para um tipo transporte em massa através das águas, a concessionária responsável pelo modal hidroviário disponibiliza apenas quatro linhas através da baía de Guanabara. É o meio de transporte menos utilizado pela população porque tem um custo mais alto e não existem tantos itinerários disponíveis.

Os ônibus dominaram o transporte público na região metropolitana do Rio de Janeiro após o fim da operação dos bondes na década de 1960. São aproximadamente cem milhões de viagens mensais através dos BRTs e ônibus tradicionais. Como comparação, os ônibus transportam cinco vezes mais passageiros que o metrô. Esse desequilíbrio nos torna totalmente dependentes deles. Além disso, o grande aumento do número de veículos particulares está causando uma grande crise de mobilidade na cidade, que atualmente apresenta o maior tempo médio de deslocamento entre as

principais cidades brasileiras. Como podemos ver na Figura 1, é notório que o transporte através dos BRTs e ônibus é o modal que mais afeta a vida da população.

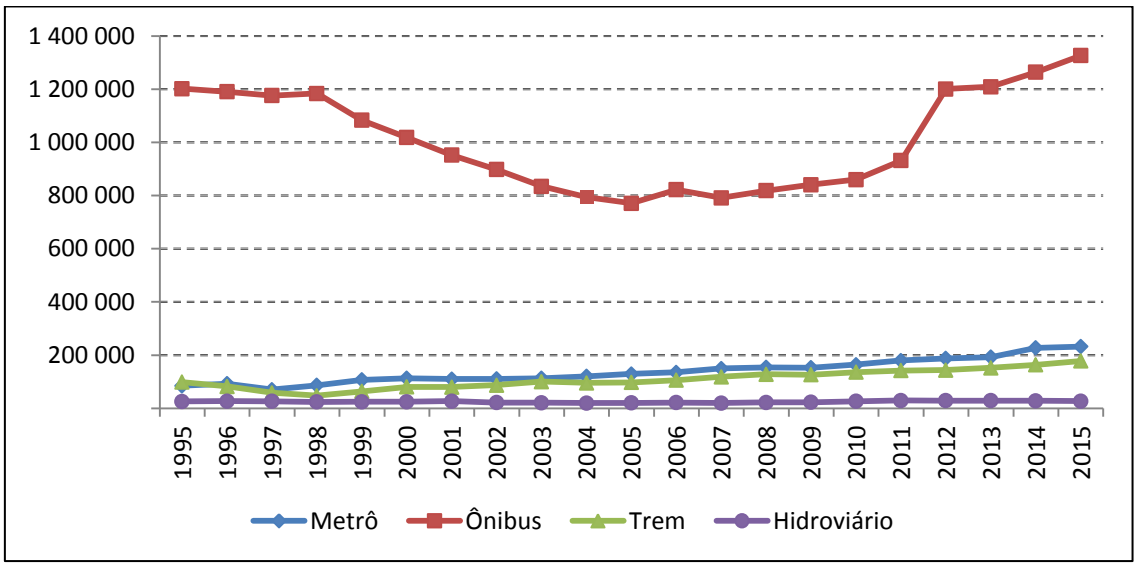

Figura 1: Utilização anual dos meios de transporte no Estado do Rio de Janeiro (dividido por 1000 passageiros)

2.2 Ônibus

Atualmente as linhas de ônibus da cidade do Rio de Janeiro são controladas e coordenadas pelo sindicato Rio Ônibus. Esse sindicato foi fundado em 10 de outubro de 1941 e congrega as empresas que operam o transporte por ônibus do Rio de Janeiro sob o regime de permissão. Está filiado a FETRANSPOR (Federação das Empresas de Transportes de Passageiros do Estado do Rio de Janeiro), que congrega outros nove sindicatos de empresas de ônibus.

Como dito no capítulo anterior, nosso projeto focará no transporte coletivo por ônibus por afetar diretamente um maior número de pessoas que os outros meios, por termos informações insuficientes para a população e também por contarmos com um suporte de dados públicos sobre o modal. Esse suporte é a base para todo o projeto e contém informações gerais de todas as linhas.

Uma das informações fornecidas por esses dados públicos é a posição dos veículos em tempo real. Esses dados são transmitidos através de aparelhos GPS instalados em cada um dos ônibus. GPS (*Global Positioning System*) é um sistema de posicionamento por satélite que fornece ao aparelho receptor a sua localização, assim como informação horária, sob quaisquer condições atmosféricas, a qualquer momento e em qualquer lugar na Terra, desde que o receptor se encontre no campo de visão de três

satélites GPS. Com o aparelho instalado, um ônibus envia a sua posição geográfica, que é formada pela longitude e longitude, e sua velocidade.

Com as informações que nos são disponibilizadas pelo portal da prefeitura, que serão explicadas de forma detalhada no item 2.3, não é possível exibir e extrair tantas informações. Por exemplo, mesmo com todas as posições dos veículos de uma linha em um determinado minuto do dia é difícil definir para alguns desses ônibus quais estão indo de um ponto de partida para outro. Essa informação é de valia para questões importantes como calcular o tempo médio da viagem de um ponto final ao outro, distribuição dos veículos ao longo de um trajeto, tempo decorrido desde o seu ponto final, ou seja, informações que poderiam ser combinadas com ações pontuais para melhorar a prestação do serviço.

2.3 Dados públicos

De acordo com a Open Knowledge Internacional (2), dados são abertos quando qualquer pessoa pode livremente acessá-los, utilizá-los, modificá-los e compartilhá-los para qualquer fim, estando sujeito, no máximo, a exigências que visem preservar sua proveniência e abertura. São publicados em formato aberto e sob uma licença aberta.

Os dados abertos estão se tornando tendência no governo brasileiro e com eles nós conseguimos muitos benefícios. O principal item é o fornecimento de transparência exigido pela população da era da informação. A publicação dos dados permite que a sociedade acompanhe e avalie as ações dos governantes. O compartilhamento de soluções inovadoras, feitas pela própria população para atender a si mesma, é outro item que podemos destacar. Muitos desenvolvedores, organizações e acadêmicos dedicam seu tempo para entregar serviços úteis para muitos cidadãos.

Baseado em oito princípios e seguindo três leis (3), a Prefeitura do Rio de Janeiro disponibilizou uma ferramenta para que desenvolvedores pudessem utilizar dados abertos governamentais em seus sistemas e aplicações. Esses dados são organizados de maneira que permita sua reutilização e, através dos aplicativos e sistemas criados, proporcionar acesso da população aos dados de contas públicas, transporte, saúde, educação e outros.

2.3.1 Dados públicos na prática

Sendo uma forma de democratizar o fornecimento de informações que interessam à população, existem inúmeras formas de melhorar a sua qualidade de vida,

direta ou indiretamente. Nesta seção iremos descrever alguns exemplos práticos de projetos que tiverem suporte de dados públicos.

Aqui na própria Cidade do Rio de Janeiro, os dados dos GPS integrados aos ônibus, juntamente com os dados fornecidos pelo aplicativo *Waze,* foram utilizados para o monitoramento do fluxo de carros no centro da cidade quando o Elevado da Perimetral, uma importante ligação entre os principais entroncamentos rodoviários da cidade, foi interditado para sua implosão. O Pensa (4) foi o responsável por esse estudo e comprovou que a retirada da Perimetral não impactará tanto o trânsito quanto estavam planejando. O seu projeto visava analisar as médias de velocidade (não inclui os veículos parados) e as médias mínimas de velocidade (inclui os veículos parados) dos ônibus e carros para ver qual era a oscilação durante todo o dia com a Perimetral fechada. Como base de comparação, os dados referentes aos dias anteriores ao dia do fechamento já tinham sido extraídos. Realizando a comparação, concluíram que houve uma diminuição de até 15% na velocidade média dos ônibus durante o período da manhã, porém durante o resto do dia a velocidade média não teve alteração.

De acordo com Sulopuisto (5), Helsinki, a capital da Finlândia, é a cidade com maior sucesso na utilização de dados públicos no mundo. Através do seu portal (6), mais de 1000 fontes de dados públicos são fornecidas para seus mais de 600.000 habitantes. Dentre os serviços que atualmente existem, podemos destacar o aplicativo *BlindSquare*. Ele utiliza as informações públicas de serviços e transporte públicos, junto com dados vindo do sistema *Foursquare,* para ajudar pessoas cegas a percorrem a cidade. O funcionamento dele é baseado totalmente no GPS e suas instruções são realizadas por voz ou botões físicos do *smartphone,* para que o usuário não tenha a necessidade de clicar na tela para utilizá-lo. O usuário pode selecionar algum tipo de serviço ou estabelecimento para onde deseja ir e o aplicativo fornece as direções por áudio até o seu destino.

Aplicativos que fornecem informação em tempo real sobre o transporte público de uma determinada região já estão disponíveis em várias partes do mundo. Esses aplicativos melhoram a utilização dos meios de transporte e, consequentemente aumentam a sua utilização, levando assim à redução do trânsito na região. Dentre várias aplicações que são normalmente utilizadas por seus usuários, podemos destacar a "OneBusAway" e a "Donde em Zaragoza".

"Donde en Zaragoza" permite que usuários de iPhone saibam onde está a parada de ônibus mais próxima ou o ponto de wi-fi público.

Nos Estados Unidos, a ferramenta “OneBusAway” (7) fornece informações em tempo real sobre chegadas de trens e ônibus. É um projeto totalmente open source que iniciou em 2008 com dois estudantes de graduação de Washington. Iniciou com um simples desejo de melhorar a vida dos trabalhadores na região de Puget Sound e, hoje em dia, se tornou um grande projeto aberto pelo mundo que exibe informações em tempo real de sete regiões.

De acordo com Becky Hogge (8), o sistema de transporte do Reino Unido (TfL) começou de forma discreta no cenário dos dados abertos. Porém, após cinco anos da liberação desses dados, várias aplicações já são realidade e são utilizadas por milhões de usuários, gerando uma economia de dezena de milhões. A jornada para esse sucesso começou em 2007, quando *widgets* sobre transporte foram desenvolvidos com a intenção que os usuários pudessem checar os status das linhas do metrô durante os fins de semana. Para que pudessem progredir, em 2009 foi lançada uma seção especial para desenvolvedores web no site da TfL. Nos anos seguintes, novas APIs foram disponibilizadas para localização dos ônibus e trens. Foi possível concluir que foram economizados entre 15 e 42 milhões de libras no desenvolvimento dos aplicativos, ou seja, esse seria o custo para que o governo criasse todas as aplicações.

2.4 O modelo de dados de transporte na Cidade do Rio de Janeiro

Como descrito na seção 2.3, a Prefeitura do Rio de Janeiro possui um portal que disponibiliza as informações necessárias para utilização dos seus dados abertos. Além dessas informações, o portal também permite o acesso aos dados. Como podemos ver na Figura 2 existem 13 grupos de dados no portal.

Figura 2: Portal data.rio

Na seção de transporte e mobilidade do portal data.rio (9), grupos de dados relacionados aos meios de transporte da cidade do Rio de Janeiro são exibidos. Dentre

eles, podemos destacar os dados que definem as paradas das linhas de ônibus, os trajetos das linhas de ônibus e as posições dos ônibus no decorrer do tempo.

A quantidade de dados é grande e para um maior entendimento sobre suas estruturas é necessário definir cada um dos grupos de dados. A seguir, definiremos conceitos básicos e estruturais dos arquivos que mantém estes dados e que servirão de base para este projeto.

2.4.1 Paradas

Ônibus é um meio de transporte que permite a entrada e saída de pessoas no veículo em determinados lugares. Denominamos esses lugares como “Paradas”. Esses lugares são pontos fixos e distribuídos ao longo do trajeto de uma linha de ônibus. Através do portal, temos acesso às paradas de algumas linhas.

As paradas são disponibilizadas em um arquivo CSV por linha de ônibus, ou seja, cada linha de ônibus possui um arquivo único com todas as suas paradas. Cada linha do arquivo representa um ponto de parada dos ônibus da linha selecionada. Esse arquivo fornece os seguintes valores separados por vírgula:

- Linha: atributo que identifica o número da linha;
- Descrição: atributo que identifica o nome da linha;
- Agência: atributo que identifica a agência reguladora responsável pela linha;
- Sequência: atributo numérico que define a ordem das paradas;
- Latitude: atributo que define a latitude da parada;
- Longitude: atributo que define a longitude da parada.

2.4.2 Trajetos

O trajeto de um ônibus é o caminho que deve ser percorrido para que o veículo passe por todos os seus pontos de parada. Os trajetos são únicos e não costumam ter alteração frequente. Para que os dados de um trajeto sejam consumidos por um sistema, é necessário que existam pontos definidos por latitude e longitude e que esses pontos possuam uma ordem lógica.

Os trajetos são disponibilizados através de um arquivo CSV por linha de ônibus, ou seja, cada linha possui um arquivo único com todo o seu trajeto. Cada linha do arquivo representa um ponto do trajeto. Esse arquivo fornece os seguintes valores separados por vírgula:

- Linha: atributo que identifica o número da linha;
- Descrição: atributo que identifica o nome da linha;
- Agência: atributo que identifica a agência que representa a linha;
- Sequência: atributo numérico que define a ordem dos pontos;
- Identificador da trajetória: atributo numérico que identifica a trajetória;
- Latitude: atributo que define a latitude do ponto;
- Longitude: atributo que define a longitude do ponto.

2.4.3 Posições no tempo

Através do GPS existente em cada veículo de uma linha, é possível saber a exata posição do veículo em um determinado momento. O GPS transmite a localização e velocidade de um veículo de tempos em tempos. Essas posições são consolidadas em um arquivo único para todos os ônibus do Rio de Janeiro. Esse arquivo é atualizado a todo o momento e sempre mantém a última localização de cada veículo, ou seja, não é possível saber a localização nos minutos anteriores.

As posições dos veículos são disponibilizadas através de um arquivo CSV único, ou seja, todas as posições de todas as linhas estão contidas em um único arquivo. Cada linha do arquivo representa a posição de um veículo naquele instante de tempo. Esse arquivo fornece os seguintes valores separados por vírgula:

- Data: atributo que identifica o horário em que foi tomada a posição;
- Identificador do veículo: atributo que identifica o veículo;
- Linha: atributo que identifica o número da linha;
- Latitude: atributo que define a latitude da posição;
- Longitude: atributo que define a longitude da posição;
- Velocidade: atributo que define a velocidade no instante da tomada de posição.

2.4.4 Extração

Como o volume de dados que é mantido pelo portal data.rio (9) é pequeno para os itens citados anteriormente, eles podem ser facilmente armazenados e utilizados. Na maioria dos casos, esses arquivos não sofrem nenhum tipo de tratamento antes de sua utilização, porém existe o caso do arquivo referente à posição dos veículos no tempo. Esses dados necessitam de um trabalho de extração e limpeza mais complexo e demorado. O download dos arquivos referente às paradas e trajetos das linhas é simples

e rápido. É necessário frisar que para cada linha existe um arquivo e que para um estudo da linha é necessário tê-los armazenados.

A posição dos veículos é um item vital e provavelmente a parte mais importante do conjunto de dados. Porém suas informações, por estarem consolidadas em um único arquivo, e não separado por linhas de ônibus, como acontece nos outros dois arquivos CSV, não nos permite a sua utilização de maneira simplória. Esse problema também ocorre porque, consumindo os dados do portal em um instante, somente a última posição de um veículo será considerada e não o seu histórico ao longo do dia. Isso apenas nos possibilita estudar as posições em um determinado momento, e não a análise do trajeto realizado pelos veículos ao longo de horas e dias.

Para resolver esse problema, um *job* foi criado. Essa rotina está armazenada em um servidor para que o seu funcionamento ocorra initerruptamente. Seu funcionamento é simples: realiza o download do arquivo de minuto em minuto e vai criando arquivos únicos para cada um desses minutos. Essa rotina gera uma quantidade altíssima de documentos em um dia, cada qual contendo redundâncias, pois muitas vezes os ônibus ficam parados por longos períodos de tempo antes de retomar o movimento. A maneira encontrada para reduzir esta redundância e simplificar o acesso aos dados foi executar um programa que recebesse como entrada esses arquivos e compilasse todos os dados em pastas separadas por linhas de ônibus e datas, retirando as duplicidades de dados como parte deste processamento. Através desses procedimentos conseguimos uma organização que possibilita um trabalho mais rápido e prático com esses dados. Isso acontece porque é muito mais rápido analisar o arquivo de posições de um único dia de uma linha do que analisar um arquivo com dados de linhas e datas que não são necessários para o estudo.

2.4.5 Modelo Conceitual

Com base nos arquivos discutidos anteriormente, podemos fazer o modelo conceitual dos dados para melhor visualização das entidades, relações e atributos que farão parte do projeto (Figura 3). Para as três fontes de dados, foram definidas quatro entidades: Linha, Veículo, Posição e Parada.

A entidade “linha” se refere a um itinerário que pode ter duas paradas finais ou uma, no caso de ser itinerário circular. Cada linha possui um identificador (ex: 107, 455, 457, 413), um nome e uma agência responsável. Para que uma linha possa operar, é necessário que ele possua veículos que realizem o seu trajeto e que pontos de parada

estejam identificados neste trajeto. Veículo é qualquer ônibus que realiza o trajeto definido por uma linha. Ele é identificado por um conjunto de letras, normalmente localizado na lateral de sua carroceria. Um veículo possui um GPS e, portanto, pode ter registrar as posições que percorre ao longo de cada dia. A definição da entidade "parada" pode ser vista no item 2.3.1 e a definição da entidade "posição" no item 2.3.3.

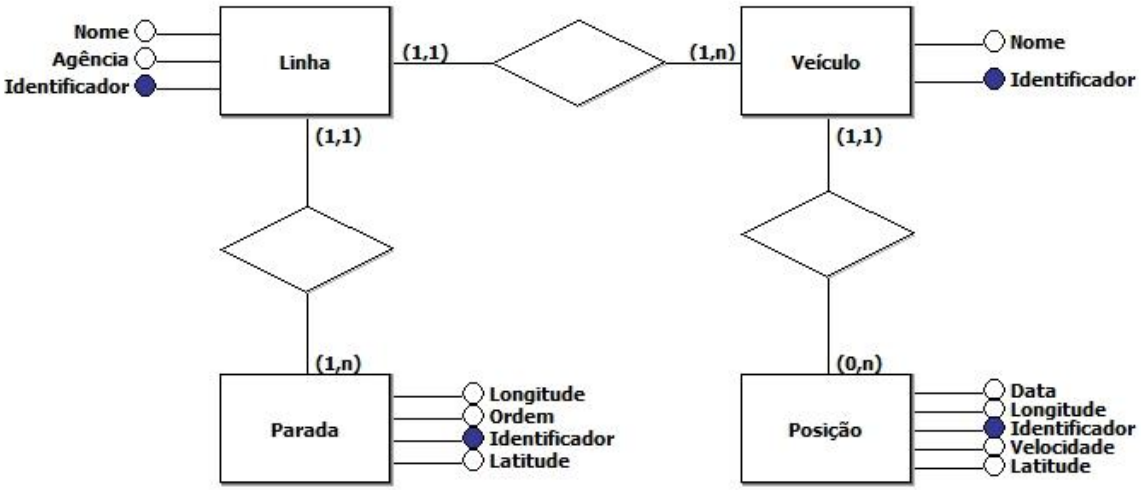

Figura 3: Modelo conceitual dos dados públicos sobre ônibus

2.5 Considerações finais

Nesse capítulo explicamos como funcionam os principais meios de transporte no Rio de Janeiro, como os dados abertos sobre estes meios de transporte podem ser utilizados e como são coletados, e definimos o modelo conceitual dos principais termos ligados ao meio de transporte por ônibus. No próximo capítulo, apresentaremos as principais partes do algoritmo de identificação de posição em rota e explicaremos o seu funcionamento.

3 O Algoritmo de Identificação de Sentido de Veículos

3.1 Introdução

O algoritmo que será descrito neste capítulo tem como objetivo definir o sentido que está sendo percorrido pelos veículos de uma linha de ônibus a cada minuto em que estes veículos estiverem em atividade ao longo de um dia. Para tornar a explicação mais simples, tomaremos como base os conceitos definidos no Capítulo 2 e o algoritmo será apresentado em três fases distintas: a extração dos dados, a estimativa inicial de sentido e a definição final do sentido.

3.2 Extração dos dados

Como visto no capítulo anterior, todos os dados utilizados neste projeto foram retirados do portal de dados livres da Prefeitura da Cidade do Rio de Janeiro, o *data.rio*(9). Esses dados incluem as posições dos ônibus ao longo do tempo, os trajetos das linhas de ônibus e seus pontos de paradas. Todos estes dados são disponibilizados em arquivos CSV e após a leitura destes arquivos, aplicamos um processamento para remover dados imprecisos e duplicados.

A fase de extração dos dados é responsável por acessar o diretório onde os arquivos com as informações das posições dos veículos e dos trajetos das linhas de ônibus ficam armazenadas e realizar uma sequência de processamentos para remover dados imprecisos e incorretos.

Imprecisão é algo que foi notado claramente nas posições dos ônibus. Exemplos de dados imprecisos incluem posições onde é impossível que um ônibus esteja (por exemplo, no mar), posições em que o ônibus não poderia estar considerando as posições que ele ocupou minutos antes (por exemplo, uma posição muito distante das anteriores) e posições que não foram capturadas, seja por falha ou desligamento do GPS que fica

no veículo ou por problemas no servidor que armazena e permite acesso aos dados. Posições de veículos nestas condições são removidas nesta fase e não são utilizadas nas próximas fases do algoritmo.

O arquivo fornecido pela Prefeitura contendo os pontos que definem os trajetos de ida e volta das linhas de ônibus muitas vezes possui pontos duplicados ou muito próximos. Para evitarmos imprecisões sobre as posições dos veículos em relação ao trajeto, assim como o custo de processamento que seria incorrido por conta dos pontos duplicados, descartamos pontos que apareçam em sequência nos trajetos e que estejam muito próximos. Neste caso, a proximidade entre dois pontos foi medida como a diferença entre a latitude do primeiro ponto e a latitude do segundo. Um cálculo similar foi realizado para a longitude. Sempre que a diferença absoluta entre as latitudes e as longitudes for menor do que 0,001, o segundo ponto é descartado.

Como resultado dessa fase, temos a lista de veículos de uma linha armazenada na variável `veiculos`. Cada veículo desta lista tem uma lista de posições que ocupou a cada minuto do dia que está sendo analisado, a que chamaremos de `posicoes`. As posições de um veículo são armazenadas como um vetor com 840 células, sendo uma para cada minuto do dia entre 07:00 e 20:59 horas.

Cada célula do vetor é representada por uma estrutura de dados com os seguintes campos: latitude da coordenada geográfica da posição do veículo no minuto correspondente (se existe posição conhecida neste minuto), longitude desta coordenada (se existe posição conhecida neste minuto), sentido do veículo (que pode ser ida ou volta, mas neste momento está preenchido como desconhecido), índice do ponto mais próximo do trajeto de ida (neste momento, desconhecido) e índice do ponto mais próximo do trajeto de volta (neste momento, também desconhecido).

3.3 Estimativa inicial de sentido

O objetivo dessa fase é identificar os pontos mais próximos nos trajetos de ida e volta de cada posição do veículo ao longo do tempo e preencher estes pontos no vetor `posicoes` do veículo. Naturalmente, não serão calculados os pontos mais próximos para as células do vetor `posicoes` em que a posição do veículo é desconhecida. Em seguida, tentamos identificar o sentido que o veículo está percorrendo a cada minuto, de acordo com a sua proximidade aos pontos dos trajetos de ida e volta. Esta identificação de sentido será aprimorada na terceira fase do algoritmo, quando contará com o contexto do sentido identificado para os minutos anteriores e posteriores.

A rotina que implementa esta fase percorre a lista de `veiculos` e para cada veículo percorre cada entrada do seu vetor de `posicoes`. Para uma entrada do vetor, a rotina calcula a distância da posição atual do veículo a cada ponto do trajeto de ida da linha, armazenando o índice do ponto mais próximo à posição atual do veículo. Um cálculo similar é realizado para o trajeto de volta. Para o cálculo de distância entre a posição atual do veículo a cada ponto dos trajetos da linha foi utilizada a fórmula de Haversine [6]. A fórmula calcula a distância entre dois pontos de uma esfera, pois como estamos trabalhando como coordenadas geográficas não podemos considerar que a ligação entre dois pontos forma uma linha reta, e sim um arco.

É importante frisar que a maior parte das linhas de ônibus possui dois sentidos, ou seja, dois pontos finais, A e B, em lugares distintos e dois trajetos, sendo o primeiro de ida (A → B) e o segundo de volta (B → A). Algumas linhas são circulares, ou seja, possuem um único ponto final e um único trajeto, que começa e termina neste mesmo ponto final. O algoritmo aqui proposto somente se aplica a linhas não circulares, pois no caso das linhas circulares não existe diferença de sentido no percurso dos veículos ao longo do tempo. Sendo assim, nas linhas circulares o algoritmo não é necessário.

Cada ponto de um trajeto possui um índice crescente que demonstra a ordem em que os veículos percorrem o trajeto. Definimos o índice do ponto de partida como zero e os índices dos próximos pontos são incrementados em direção ao ponto final. Os índices dos trajetos de ida e volta são independentes, ou seja, haverá um ponto com índice zero no trajeto de ida e outro no trajeto de volta. Além disso, não há um número máximo de pontos que definem um trajeto: este número varia de linha para linha e pode variar, inclusive, entre os trajetos de ida e volta de uma mesma linha. Guardamos os índices dos pontos na estrutura de dados que representa as posições dos veículos minuto-a-minuto porque é mais eficiente armazenar os índice do que as coordenadas geográficas dos pontos. Além disso, os índices permitem acompanhar a sequência de pontos percorridos pelo veículo (ver terceira fase do algoritmo).

Conhecidos os pontos nos trajetos mais próximos de cada posição do veículo, podemos fazer uma primeira estimativa do sentido que o veículo está percorrendo quando passa naquela posição. Como as posições dos veículos ao longo do tempo raramente se encontram exatamente sobre os pontos que definem os seus trajetos, esta primeira estimativa de sentido poderá ser alterada pela terceira fase do algoritmo, que leva em consideração mais do que o ponto atual do veículo. Na fase atual, escolhemos *Ida* como sentido de uma posição do veículo se esta posição estiver mais próxima do

ponto mais próximo no trajeto de ida e *Volta* se a posição estiver mais próxima do ponto mais próximo no trajeto de volta. Quando a distância entre os pontos mais próximos nos trajetos de ida e volta for menor do que 1 metro, a primeira estimativa de sentido é definida como *Desconhecido*. Tal ocorre quando os caminhos de ida e volta se cruzam ou quando seguem os dois lados de uma mesma avenida.

Como resultado dessa fase, temos o sentido e os índices dos pontos mais próximos nos dois trajetos preenchidos para cada célula do vetor `posicoes` de todos os veículos, ainda que o sentido seja uma primeira estimativa e possa estar marcado como *Desconhecido* em muitas posições do vetor.

3.4 Definição final do sentido

Após as fases de extração de dados e estimativa inicial de sentido, discutiremos a rotina utilizada para definir o sentido final de um veículo ao longo do tempo. Como o vetor `posicoes` já está preenchido com os índice dos pontos mais próximos aos dois trajetos (ida e volta), podemos analisá-lo e definir o sentido final baseado com base nas estimativas iniciais encontradas na fase anterior. Ao longo do texto, exibiremos pseudocódigos para exemplificar o funcionamento das rotinas de forma clara. Esta fase é composta de dois passos, executados em sequência pelas rotinas `realizarTendencias()` e `refinarResultados()`. A execução destas rotinas é coordenada pela rotina `geraPosicaoPelosMinutosDeUmaLinha()`, que recebe como parâmetro uma linha de ônibus, como pode ser observado no Algoritmo 1.

Algoritmo 1 - Definição final do sentido dos veículos de uma linha de ônibus

```
01  geraPosicaoPelosMinutosDeUmaLinha (Linha linha)
02          Para cada veiculo de uma linha
03                  realizarTendencias();
04                  refinarResultados();
05          Fim Para
06  Fim
```

3.4.1 Realizar tendências

A rotina `realizarTendencias()` controla três variáveis: `tendencia`, `contadorDesconhecidos` e `analiseLocal`. A variável `tendencia` tem como valores possíveis *Ida*, *Volta* e *Desconhecida*, assim como o sentido de um veículo. Ela define a tendência de sentido de um veículo em um determinado momento da movimentação do ônibus. O `contadorDesconhecidos` é uma variável numérica que calcula a quantidade de posições do veículo cujo sentido está definido como *Desconhecido*. A última

variável, `analiseLocal`, é uma lista de posições que auxilia a rotina a definir a tendência quando ela ainda é *Desconhecida*. O resultado final da rotina é a atualização nos valores de sentido de cada posição de um veículo.

A rotina analisa posição por posição de um veículo. Definimos `posicoes[minuto]` como a posição ocupada pelo veículo em um certo minuto do dia (considerando o período entre 07:00 e 20:59 horas). O Algoritmo 2 apresenta o pseudocódigo da rotina `realizarTendencias()`.

Algoritmo 2 – Pseudocódigo da rotina que define a tendência de cada posição do veículo

```
01  realizarTendencias()
02      tendencia = Desconhecido
03      contadorDesconhecidos = 0
04      analiseLocal = []
05      Para cada minuto entre 07:00 e 20:59 horas
06          Se (posicoes[minuto].latitude é nula) ou (posicoes[minuto].longitude é nula)
07              contadorDesconhecidos++
08              Se (contadorDesconhecidos > 3)
09                  tendencia = Desconhecido
10                  Remove todas as entradas de analiseLocal
11                  contadorDesconhecidos = 0
12              Fim Se
13          Senão, Se (tendencia = Ida ou Volta)
14              contadorDesconhecidos = 0
15              Se (estaProximoDoFim() ou houveMudancaDeDirecao())
16                  Inverter tendencia
17              Senão
18                  posicoes[minuto].sentido = tendencia;
19              Fim Se
20          Senão
21              contadorDesconhecidos = 0
22              analiseLocal.adicionar(posicao[minuto]);
23              Se (analiseLocal tiver mais que 5 elementos)
24                  Se (mais de 60% dos sentidos em analiseLocal for Volta)
25                      tendencia = Volta
26                      Define todos valores da analiseLocal como Volta
27                      Remove todas as entradas de analiseLocal
28                  Senão, Se (mais de 60% dos sentidos em analiseLocal for Ida)
29                      tendencia = Ida
30                      Define todos valores da analiseLocal como Ida
31                      Remove todas as entradas de analiseLocal
32                  Senão
33                      Retirar elemento mais antigo da analiseLocal
34                  Fim Se
35              Fim Se
36          Fim Se
37      Fim Para
38  Fim
```

A tendência de sentido de um veículo, armazenada na variável `tendencia`, pode ser descrita como a percepção de sentido no minuto que está sendo analisado. Como antes de analisar as posições de um veículo é impossível prever em qual sentido ele está

se deslocando, definimos que a tendência inicial é *Desconhecida*. Outro momento onde definimos a tendência como *Desconhecida* é quando mais de três posições em minutos consecutivos não possuem uma localização definida. Isso é necessário porque quando existe um grande período sem nenhuma localização não é possível afirmar que não houve uma troca de sentido. Para tratar esses dois casos, utilizamos a lista `analiseLocal`. Ela armazena as posições em minutos sequencias. A cada minuto em que a tendência estiver como *Desconhecia* adicionamos a posição atual a lista `analiseLocal`.

Na linha 5 começa um loop por todos os minutos que são analisados ao longo do dia, assim permitindo passar por todas as posições de um veículo. Neste loop existem três condições relevantes. A primeira, na linha 6, verifica se existe um valor de posição definido para aquele minuto. Caso esta posição não esteja definida, definimos o sentido dela como *Desconhecido* e incrementamos o `contadorDesconhecidos`. Em seguida, verificamos este contado e, se seu valor foi maior que três (três minutos seguidos com posições desconhecidas), colocamos a tendência como *Desconhecido* porque após esse tempo sem dados não podemos manter a tendência. Em seguida, zeramos o contador e limpamos a lista `analiseLocal` para começar uma nova análise.

Na linha 15, a segunda condição trata das posições quando a tendência já estiver definida. Neste caso, zeramos o contador de desconhecidos e verificamos se o veículo está chegando ao final da tendência que está definida (rotina `estaProximoDoFim()`) e se já houve uma mudança de tendência sem ter chegado ao final do trajeto (rotina `houveMudancaDeDirecao()`). Essa última verificação é necessária porque algumas vezes os veículos não chegam ao final de um trajeto para começar o outro. Caso alguma dessas condições seja verdadeira, invertemos o valor da variável `tendencia`, ou seja, passamos de *Ida* para *Volta* ou vice-versa. Do contrário, mantemos a tendência e atualizamos o sentido da posição naquele minuto com a tendência atual.

Por fim, chegamos à última condição. Se a `tendencia` não estiver definida e a posição naquele minuto não for nula, realizamos alguns procedimentos para identificar a tendência. Para tanto, zeramos o contador de desconhecidos e adicionamos a posição naquele minuto à lista `analiseLocal`. Esta lista funciona como um armazenador de posições consecutivas que ainda não possuem uma predominância de sentido, quer dizer, analisando o conjunto de posições não é possível determinar o sentido do veículo quando percorreu estas posições. Caso haja uma predominância de sentido entre as

posições, definimos a `tendencia` com o valor predominante e atualizamos todas as posições da lista para esse sentido. Caso não haja uma predominância entre os valores, eliminamos a posição mais antiga da lista e acrescentando uma nova, de modo a construir uma predominância de sentido ao longo dos próximos minutos.

A rotina `estaProximoDoFim()`, utilizada pela rotina `realizarTendencia()` e apresentada no Algoritmo 3, recebe a tendência e a posição em um minuto como parâmetros para verificar se o ponto mais próximo do trajeto naquela posição é um dos últimos pontos do trajeto, ou seja, se o ônibus está chegando ao fim da linha.

Algoritmo 3 – Pseudocódigo da rotina que verifica se uma posição está próxima do fim de um trajeto

```
01  booleano estaProximoDoFim(posicao, tendencia)
02      Se (tendencia = Ida)
03          diferencaFim = trajetoriaIda.totalPontos-posicao.pegarIndiceTrechoIdaMaisProximo()
04          Se (diferencaFim <= 2)
05              retorna verdadeiro
06          Senão
07              retorna falso;
08          Fim Se
09      Senão
10          diferencaFim = trajetoriaVolta.totalPontos-posicao.pegarIndiceTrechoVoltaMaisProximo()
11          Se (diferencaFim <= 2)
12              retorna verdadeiro
13          Senão
14              retorna falso
15          Fim Se
16      Fim Se
17  Fim
```

A rotina está dividida em dois casos, um quando a `tendencia` é *Ida* e outro quando a `tendencia` é *Volta*. Em ambos os casos, a rotina calcula a diferença entre o total de pontos que forma um trajeto (Ida ou Volta) e o índice do ponto do trajeto mais próximo à posição atual do veículo. Essa diferença é armazenada na variável numérica `diferencaFim`. Caso essa variável seja menor ou igual a 2, a rotina retorna verdadeiro. Caso contrário, o retorno é negativo.

A rotina `houveMudancaDeDirecao()` (Algoritmo 4) recebe o minuto que está sendo analisado e a tendência momentânea e verifica se um veículo começou a mudar o seu trajeto antes de chegar ao fim do trajeto anterior. Para tanto, ela verifica se nas posições anteriores ao minuto recebido houve uma mudança de sentido. A rotina verifica, no máximo, as 10 últimas posições percorridas pelo veículo para que não sejam consideradas posições muito distantes da atual. Em cada posição, se verifica se o índice mais próximo do trajeto da posição atual do veículo (posição no minuto sob análise) é maior ou menor ao índice mais próximo do trajeto da posição sob análise.

Algoritmo 4 - Pseudocódigo da rotina que verifica se um ônibus está mudando de trajeto sem ter chegado ao final do trajeto anterior

```
01 booleano houveMudancaDeDirecao(minuto, tendencia)
02   contador = 1
03   reducaoDoIndiceDaTendencia = 0
04   indiceIdaMaisProximo = posicao[minuto].pegarIndiceIdaMaisProximo()
05   indiceVoltaMaisProximo = posicao[minuto].pegarIndiceVoltaMaisProximo()
06   Enquanto (contador < 10 e reducaoDoIndiceDaTendencia < 2)
07     Se (tendencia = Ida)
08       indiceIdaMaisProximoAnterior = posicao[minuto - contador].pegarIndiceIdaMaisProximo()
09       Se (indiceIdaMaisProximo < indiceIdaMaisProximoAnterior)
10         reducaoDoIndiceDaTendencia++
11       Fim Se
12     Senão
13      indiceVoltaMaisProximoAnterior = posicao[minuto - contador].pegarIndiceVoltaMaisProximo()
14      Se (indiceVoltaMaisProximo < indiceVoltaMaisProximoAnterior)
15         reducaoDoIndiceDaTendencia++
16     Fim Se
17     contador++
18   Fim Enquanto
19   Se (reducaoDoIndiceDaTendencia == 2)
20     retorna verdadeiro;
21   Senão
22     retorna falso
23 Fim
```

Duas variáveis são responsáveis pelo controle da rotina: `contador` e `reducaoDoIndiceDaTendencia`. O `contador` limita o loop a verificar apenas as últimas 10 posições. Já a variável `reducaoDoIndiceDaTendencia` controla quantas posições do veículo sofreram reduções, ou seja, verifica se o veículo está se afastando do seu destino final. Nas linhas 8 e 13 essas diferenças são calculadas, baseada na posição atual e nas posições anteriores. Caso existam duas reduções nos últimos minutos verificados, considera-se que o veículo mudou de sentido e o retorno é verdadeiro.

3.4.2 Refinar Resultado

Muitas posições percorridas por um veículo não são enviadas pelo GPS e o sentido de percurso nestas posições não é definido pelo algoritmo, permanecendo como *Desconhecido* até este momento. No entanto, os resultados próximos àquela posição permitem considerar que o veículo vinha seguindo um determinado trajeto e assumir que ele continuou neste trajeto ao longo do período coberto por estas posições. Por exemplo, se em um dado minuto não foi possível encontrar a posição do veículo mas as posições anteriores e posteriores a este minuto apontam como se estive no sentido de *Ida*, podemos supor que o seu sentido para os minutos não cobertos também é *Ida*.

A rotina `refinarResultados()`, apresentada no Algoritmo 5, define o sentido das posições que estão com o sentido *Desconhecido*. Ele percorre todas as posições de

um veículo e utiliza as variáveis `inicio` e `fim` para controlar o início e o fim da sequência de posições que estão com o sentido *Desconhecido* para que sejam tratados de forma única. Quando uma sequência de posições nulas é encontrada, uma tendência é definida pela rotina `verificarTendenciaDosProximos()` e as posições têm seu sentido atualizado pela rotina `definirTendenciaParaOsDesconhecidos()`.

Algoritmo 5 - Pseudocódigo da rotina que refina os resultados

```
01  refinarResultados()
02    inicio = 0
03    fim = 0
04    contadorDesconhecido = 0
05    Para cada minuto
06      Se (posicao[minuto] = Desconhecido)
07        Se (contadorDesconhecido = 0)
08           inicio = minuto;
09        Fim Se
10        contadorDesconhecido++
11        fim = minuto
12      Senão
13        Se (contadorDesconhecido > 0)
14          tendencia = verificarTendenciaDosProximos(inicio, fim)
15          definirTendenciaParaOsDesconhecidos(inicio, fim, tendencia)
16        Fim Se
17        contadorDesconhecido = 0
18      Fim Se
19    Fim Para
20  Fim
```

A rotina `verificarTendenciaDosProximos()`, apresentada no Algoritmo 6, é responsável por descobrir qual o sentido das posições nulas. Ela encontra a tendência das posições anteriores (`tendenciaAnteriores`) e posteriores (`tendenciaProximos`) e as compara para definir uma única tendência. Caso as duas sejam iguais, como pode ser visto na linha 5, a tendência de sentido que será retornada pode ser qualquer uma das duas (*Ida* ou *Volta*). Caso as tendências sejam distintas, a prioridade será dada às posições anteriores se seu sentido não estiver definido como *Desconhecido*. Caso esteja, a `tendencia` será definida com o sentido das posições posteriores.

Duas rotinas são a base para a descoberta das tendências de sentido dos pontos anteriores ou posteriores a uma posição. As rotinas `tendenciaDosAnteriores()` e `tendenciaDosProximos()` são semelhantes e trabalham com a mesma premissa: contar a quantidade de posições que vão para o mesmo sentido e retornar o valor da maioria, retirando alguns casos de exceção. O Algoritmo 7 apresenta o pseudocódigo da rotina `tendenciaDosAnteriores()`. Como o pseudocódigo de `tendenciaDosProximos()`

funciona de maneira análoga, apenas a descrição de uma das rotinas é necessária para entender o funcionamento de ambas.

Algoritmo 6 - Pseudocódigo da rotina que verifica qual sentido uma sequência de posições com sentido desconhecido está tomando

```
01  tendencia verificarTendenciaDosProximos(inicio, fim)
02    tendencia = Desconhecido
03    tendenciaProximos = tendenciaDosProximos(fim)
04    tendenciaAnteriores = tendenciaDosAnteriores(inicio)
05    Se (tendenciaProximos = tendenciaAnteriores){
06      tendencia = tendenciaProximos
07    Senão
08      Se (tendenciaAnteriores != Desconhecido)
09        tendencia = tendenciaAnteriores;
10      Senão (tendenciaProximos != Desconhecido)
11        tendencia = tendenciaProximos
12      Fim Se
13    Fim Se
14    retorna tendencia
15  Fim
```

Algoritmo 7 - Pseudocódigo da rotina que verifica qual a tendência de sentido das posições anteriores a uma posição

```
01  tendencia tendenciaDosAnteriores (minutoBase)
02    tendenciaIda = 0
03    tendenciaVolta = 0
04    tendenciaDesconhecida = 0
05    Para cada minuto anterior a minutoBase até minutoBase - 3
06      Se posição[minuto] = Ida
07        tendenciaIda++
08      Senão, Se posição[minuto] = Volta
09        tendenciaVolta++
10      Senão
11        tendenciaDesconhecida++
12      Fim Se
13    Fim Para
14    Se (tendenciaDesconhecida > 1)
15        retorna Desconhecido
16    Senão, Se (tendenciaIda > tendenciaVolta)
17        retorna Ida
18    Senão
19        retorna Volta
20  Fim
```

As variáveis `tendenciaIda`, `tendenciaVolta` e `tendenciaDesconhecida` calculam a quantidade de pontos que vão para cada sentido ou são desconhecidas. A rotina recebe como parâmetro o minuto atual (`minutoBase`), percorre os minutos anteriores (no caso da rotina `tendenciaDosProximos()`, os minutos posteriores são percorridos) e vai incrementando a cada uma das variáveis quando um determinado sentido é encontrado em uma dessas posições, como pode ser visto nas linhas 7, 9 e 11.

O objetivo da rotina é verificar se a maioria das posições segue um mesmo sentido. Se uma tendência de sentido for encontrada, a rotina retorna essa tendência.

3.5 Considerações finais

Nesse capítulo explicamos com pseudocódigos como funciona o algoritmo de identificação de sentido de um veículo. O funcionamento é feito a partir da análise do seu caminho percorrido juntamente com os pontos que formam os seus trajetos. A rotina `realizarTendencias()` é responsável por encontrar os sentidos em cada minuto de deslocamento do ônibus, enquanto `refinarResultados()` trata os valores desconhecidos. No próximo capítulo, apresentaremos as principais dificuldades encontradas no projeto e alguns resultados apresentados pelo algoritmo.

4 Resultados do Algoritmo Proposto

Neste capítulo exibiremos alguns resultados obtidos através da execução do algoritmo de identificação de sentido de veículos. Também mostraremos as dificuldades de efetuar testes com este algoritmo e os métodos que foram utilizados para contornar esses problemas. Exemplos de funcionamento do algoritmo também serão apresentados.

4.1 Aplicação gráfica

O algoritmo descrito no capítulo anterior armazena as coordenadas geográficas e a informação de tempo (horário compreendido entre 07:00 e 20:59 horas) no vetor `posicoes`. Em consequência do grande número de informações manipuladas pelo algoritmo, podemos perceber que estas informações não são facilmente compreendidas em formato textual e muitas não significam nada sem uma representação gráfica. Outro problema da visualização textual é a dificuldade de perceber os acontecimentos naturais na rotina de um ônibus (longas paradas, mudança de sentido sem chegar ao final de um trajeto, erros esporádicos no GPS, entre outros). Isso fez com que desenvolvêssemos uma aplicação Web para que os resultados do algoritmo pudessem ser vistos de forma gráfica. Este recurso permitiu a realização de testes com o algoritmo.

A aplicação Web exibe as posições dos ônibus ao longo do tempo em um mapa. Ela oferece suporte ao uso de posições georeferenciadas (latitude e longitude das posições dos veículos) e diferentes representações (ícones e linha). A exibição de mapas ajuda a identificar com facilidade as posições dos veículos na cidade do Rio de Janeiro; já as linhas e ícones representam os trajetos e os seus pontos de inflexão.

O modelo utilizado no desenvolvimento da aplicação foi uma combinação entre HTML e JavaScript. Para completar o funcionamento e aportar as funcionalidades de itens gráficos e exibição de mapas, utilizamos o Google Maps APIs (10). O Google divide os seus serviços em várias plataformas (Android, iOS, Web), mas o utilizado no nosso projeto foi o Maps JavaScript API. Ele possui uma ampla documentação que nos

permite acessar suas funcionalidades de apresentação de mapas, rotas, trânsito em tempo real e outros recursos que não foram utilizados.

Como podemos ver na Figura 4, a aplicação possui uma área principal (a área do mapa) onde um ícone representa a localização de um ônibus em um minuto do dia. Duas linhas (azul e verde) representam os dois trajetos da linha (ida e volta). A aplicação possui ainda uma área de controle, representada na parte superior da Figura 4. A área de controle exibe o índice da posição atual (minuto) e o horário dessa posição (hora e minuto), assim como botões de *play*, *pause* e *reset*. O botão *play* realiza a troca de posições do veículo minuto-a-minuto de forma automática, enquanto o botão *pause* permite a troca de posições do ônibus de forma manual. O botão *reset* volta para o minuto inicial (07:00 horas) do deslocamento do ônibus. A cada troca de minuto (manual ou automática), o ícone que indica a posição do ônibus se desloca para a posição georeferenciada referente a este minuto. A cor do ícone que representa o ônibus será igual a cor do trajeto em que este se encontra em um determinado minuto do dia.

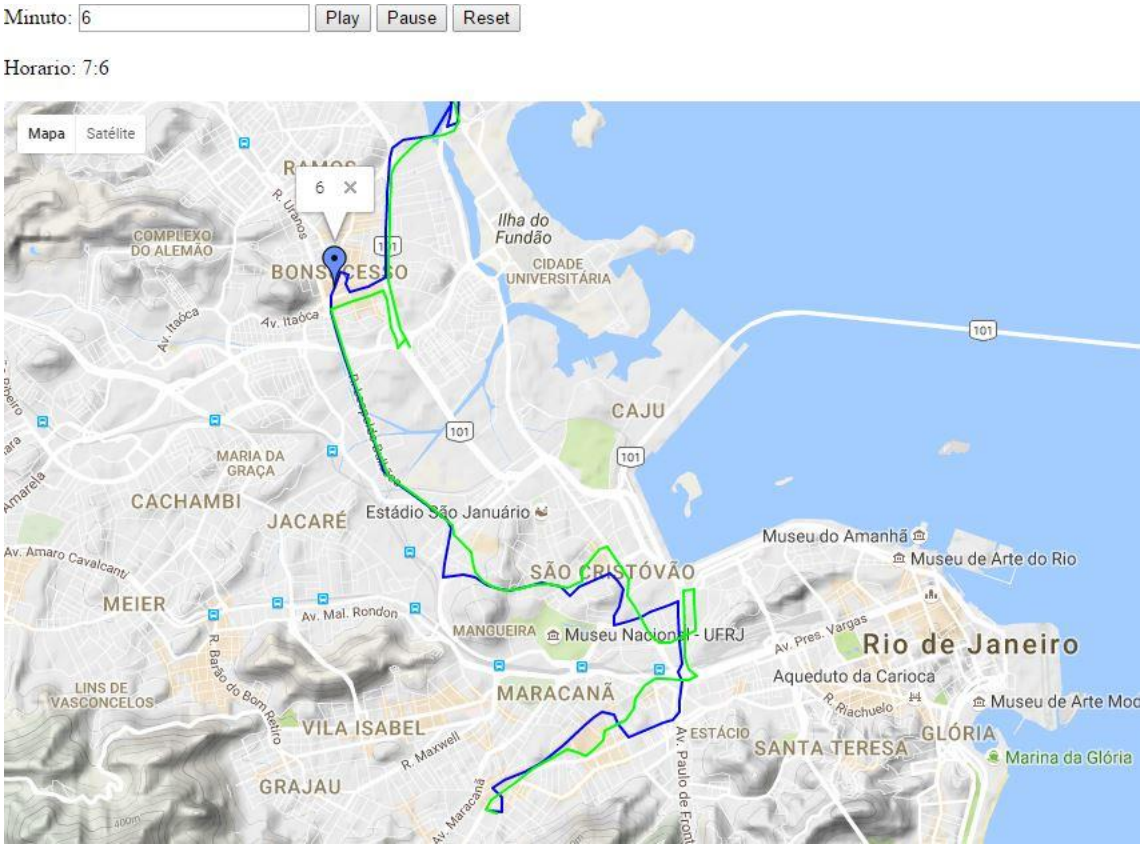

Figura 4: Uma tela da aplicação Web que foi desenvolvida para avaliar o algoritmo de identificação de sentido dos veículos

4.2 Plano de Testes do Algoritmo

Com a utilização da aplicação Web, conseguimos montar um roteiro de teste para avaliar o funcionamento do algoritmo conforme o seu desenvolvimento avançava. Através dela, conseguimos visualizar situações recorrentes em algumas linhas de ônibus e erros que não seriam vistos sem o auxílio da aplicação.

Considerando os dados recebidos do Portal da Prefeitura do Rio de Janeiro, muitas linhas de ônibus estavam disponíveis para executarmos o algoritmo para os seus veículos. Decidimos limitar os testes a algumas linhas e verificar manualmente a qualidade do resultado gerado pelo algoritmo. Para cada uma dessas linhas executamos os testes para alguns veículos no mesmo dia do mês para que não houvesse distorção nos resultados por conta de alteração nos trajetos das linhas.

Nosso objetivo era verificar se os trajetos que estavam indicados pelo algoritmo em cada um dos minutos do caminho percorrido pelo ônibus estavam correspondendo ao trajeto que ele estava seguindo na realidade. Para isso, comparamos os resultados gerados pelo algoritmo de definição de sentido com os sentidos corretos em cada minuto do dia para os veículos envolvidos em nossa análise (chamaremos de *gabarito*). Para elaborar o gabarito foi necessário um trabalho manual de observação e anotação dos sentidos dos veículos com o auxílio da aplicação Web.

Cada ônibus escolhido para o teste foi analisado na aplicação Web e suas informações de direção de trajeto foram anotadas conforme o veículo ia se locomovendo no mapa. Importante frisar que nesse momento nenhuma informação vinda do algoritmo foi considerada, mas apenas os dados coletados da Prefeitura. Foram anotados o minuto de início e o fim de cada viagem (deslocamento do início até o fim de um trajeto) do veículo. As Tabelas 1 e 2 apresentam os gabaritos coletados para as linhas 107 e 292 nas datas 03/09/2015 e 29/07/2015, respectivamente, comparados com os resultados obtidos pelo algoritmo.

Tabela 1 – Resultados observados e gabarito para a linha 107 – Urca x Central

Linha		107				
Veículo		A63521				
Data		03/09/2015				
	Resultado do algoritmo			Gabarito		
	Início	Fim	Trajeto	Início	Fim	Trajeto
1	07:00	08:15	Ida	07:00	08:15	Ida
2	08:16	08:57	Volta	08:16	08:57	Volta
3	08:58	10:10	Ida	08:58	10:10	Ida
4	10:11	11:01	Volta	10:11	11:01	Volta
5	11:02	12:25	Ida	11:02	12:25	Ida
6	12:26	13:22	Volta	12:26	13:20	Volta
7	13:23	14:44	Ida	13:21	14:44	Ida
8	14:45	15:33	Volta	14:45	15:33	Volta
9	15:34	16:31	Ida	15:34	16:31	Ida
10	16:32	17:29	Volta	16:32	17:29	Volta
11	17:30	18:28	Ida	17:30	18:27	Ida
12	18:29	19:29	Volta	18:28	19:27	Volta
13	19:30	20:13	Ida	19:28	20:13	Ida
14	20:14	20:46	Volta	20:14	20:46	Volta
15	20:47	20:58	Ida	20:47	20:58	Ida

Tabela 2 - Resultados observados e gabarito para a linha 292 – Engenho da Rainha x Castelo

Linha		292				
Veículo		B55002				
Data		29/07/2015				
	Resultado			Gabarito		
1	Início	Fim	Trajeto	Início	Fim	Trajeto
2	07:00	07:16	Ida	07:00	07:16	Ida
3	07:17	09:16	Volta	07:17	09:16	Volta
4	09:17	10:23	Ida	09:17	10:23	Ida
5	10:24	11:48	Volta	10:24	11:48	Volta
6	11:49	12:50	Ida	11:49	12:50	Ida
7	12:51	15:12	Volta	12:51	15:06	Volta
8	15:13	16:33	Ida	15:07	16:33	Ida
9	16:34	18:19	Volta	16:34	18:02	Volta
10	18:20	19:29	Ida	18:03	19:29	Ida
11	19:30	20:48	Volta	19:30	20:42	Volta
12	20:49	20:58	Ida	20:49	20:58	Ida

Podemos verificar que nem sempre os resultados obtidos pela verificação manual são iguais aos resultados obtidos pelo algoritmo, como nas linhas 6 e 12 da Tabela 1 e linhas 7 e 9 na Tabela 2. Estas distorções são resultado de localizações desconhecidas (dados não coletados pelo GPS) e da indefinição do sentido dos veículos

na proximidade com o final do trajeto. A diferença nessas viagens selecionadas e descritas acima acontece na mudança de sentido, pois é complexo definir em que momento o veículo terminou um trajeto e começou outro, tendo em vista que ele permanece muitos minutos parado no ponto final.

O número de viagens nos dois casos é igual, o que garante que as viagens estão sendo contabilizadas corretamente. A maioria das viagens identificadas não apresenta diferença significativa, reforçando que a definição de quando um ônibus termina uma viagem e começa outra em um contexto em que a informação nem sempre está totalmente disponível é o principal problema encontrado pelo algoritmo.

4.3 Aplicações do algoritmo

Aqui apresentaremos algumas situações ocorridas no dia-a-dia de um ônibus e veremos como elas são tratadas pelo algoritmo. Algumas situações são encontradas frequentemente em uma viagem de uma linha de ônibus, como a chegada de um veículo no seu ponto final e a mudança de trajeto antes de chegar ao fim do trajeto anterior.

4.3.1 Chegada ao ponto final

Nesse exemplo, analisaremos o veículo A63521 da linha 107 no dia 03/09/2015. Na Figura 5 temos uma sequência de posições do veículo em minutos consecutivos. O minuto inicial é o 73 (08:13 horas) e o minuto final é o 76 (08:16 horas). Podemos verificar que, no minuto inicial, o ícone que representa a posição do veículo está com a cor azul, indicando que o sentido identificado para o veículo era de *Ida*. O ponto final de uma linha pode ser visualizado como o término de um trajeto (azul) e início de outro trajeto (verde). Na sequência de posições, vemos que o veículo segue em direção ao fim do trajeto azul (*Ida*) e, logo após a sua passagem pelo ponto final, ele muda de cor e fica verde. Essa mudança indica que o algoritmo identificou que nesse momento o veículo terminou uma viagem e começou outra viagem com sentido contrário.

Figura 5: Deslocamento de um veículo da linha 107

A rotina `estaProximoDoFim()` verifica a chegada no final de um trajeto e está representada no Algoritmo 3. Ela recebe, a cada minuto do deslocamento, a posição do ônibus e o trajeto que ele está seguindo. Nas duas primeiras posições apresentadas na Figura 5, minutos 73 e 74, ela verifica se a posição do veículo está próxima aos dois últimos pontos que formam o trajeto. Ela retorna *falso* porque o veículo ainda está longe desses dois pontos finais. No minuto 75, a rotina reconhece que o ponto mais próximo é um desses dois pontos finais e retorna *verdadeiro*. No algoritmo 2 (linha 15) podemos verificar que caso a rotina `estaProximoDoFim()` retornasse *verdadeiro*, a tendência de sentido seria invertida, assim como o sentido das posições do ônibus a partir dos próximos minutos. E isso é o que ocorre nesse exemplo: como o retorno foi *verdadeiro* no minuto 75, após esse minuto o sentido do ônibus mudou e consequentemente a sua cor também.

4.3.2 Mudança de trajeto antes da chegada ao fim do trajeto anterior

Nesse exemplo, analisaremos o veículo C20045 da linha 380 no dia 03/09/2015. Na Figura 6 temos uma sequência de posições do veículo em minutos consecutivos. O minuto inicial é o 102 (08:41 horas) e o minuto final é de número 113 (08:53 horas). Podemos verificar que, no minuto inicial, o ícone que representa a posição do veículo está identificado com a cor azul, indicando que o sentido identificado para o veículo era de *Ida*. Na sequência de posições, vemos que o veículo sai do traçado para chegar ao fim do trajeto azul (*Ida*). Na posição do minuto 113, ele começa a seguir o trajeto de *Volta* (verde) e, portanto, sua cor também é alterada.

Figura 6: Deslocamento de um veículo da linha 380

A rotina `houveMudancaDeDirecao()` verifica essa mudança de sentido e está representado no Algoritmo 4. Ela recebe cada minuto do deslocamento e o trajeto que o veículo está seguindo. Até a posição 112 ele verifica se a posição do veículo está indo no sentido contrário ao que devia seguir para aquele trajeto e retorna *falso* porque o

veículo ainda não começou a seguir um caminho de retorno. No minuto 113, o veículo começa a seguir o trajeto de *Volta* e a rotina reconhece que ele começou um novo trajeto sem ter terminado o trajeto anterior. Em função disso, a rotina retorna *verdadeiro*. A partir desse momento, os pontos que mantiverem o trajeto de *Volta* terão a cor verde.

4.4 Considerações finais

Nesse capítulo mostramos que não foi simples produzir um roteiro de teste para o algoritmo, dado que não havia um gabarito para servir como base para os testes. Por isso, construímos uma aplicação Web para visualizar o percurso dos veículos e verificar manualmente quando eles trocam de sentido. Assim, conseguimos o horário inicial, horário final e sentido de cada viagem realizada pelos veículos e comparamos com os resultados obtidos pelo algoritmo, chegando a evidências inicias do seu bom funcionamento. Por fim, descrevemos dois casos de difícil tratamento que foram encontrados durante os testes e que foram resolvidos pelo algoritmo proposto.

5 Conclusão

Neste capítulo concluímos a apresentação do algoritmo de identificação de sentido dos ônibus, indicando projetos que podem suceder a partir desse algoritmo e melhorias que poderiam ser realizadas para aprimorar os seus resultados.

5.1 Contribuições

Desenvolvemos um algoritmo que, com base em posições geográficas de um veículo e com pontos que definem seus trajetos, consegue definir o sentido do veículo para cada minuto do dia. Com essa aplicação, acreditamos que é possível utilizar essas informações para poder controlar as frotas de ônibus nas cidades. Essas informações são primordiais para que gestores possam investir em melhoramentos e infraestrutura de suas linhas. O algoritmo consegue uma informação que não é disponibilizada pelo GPS e que impossibilita muitos projetos, pois esse tipo de informação é valiosa e permite que aplicativos de localização em tempo real de ônibus possam saber exatamente qual será o ônibus que passará em um determinado ponto pela cidade.

5.2 Limitações do Estudo e Trabalhos Futuros

O algoritmo apresentado nesse trabalho não funciona com linhas de ônibus que possuem um único trajeto (linhas circulares). Portanto, uma possível melhoria para o algoritmo seria uma implementação para que suportasse um número indefinido de trajetos. Isso permitiria maior liberdade para utilização de ônibus circulares (único ponto de parada) e ônibus com mais de dois pontos de paradas (se houver).

Por ter sido criado para atender as posições dos ônibus da cidade do Rio de Janeiro, se alimentando dos dados fornecidos pelo portal data.rio (9), o algoritmo precisa que os dados possuam uma ordem pré-estabelecida para que possam ser lidos

corretamente. Uma interface simplificada para definir as regras de importação para os diferentes dados seria importante para a aplicação do algoritmo em outras cidades.

Sabendo da dificuldade das linhas de ônibus em definir o tempo de saída dos seus veículos dos pontos finais e a reclamação de seus usuários quanto a demora e lotação dos veículos, um projeto que permitisse o estudo da movimentação dos veículos ao longo do dia para identificar os seus intervalos de chegada aos pontos (tempo entre um veículo e um próximo) é muito importante, pois é uma informação muitas vezes desconhecida pelas empresas responsáveis e poderia gerar uma vantagem competitiva. Esse projeto poderia gerar uma nota de avaliação de uma linha durante um dia. Essa avaliação seria calculada de acordo com o espaçamento entre os veículos, e para isso, é necessário conhecer os ônibus que estão andando no mesmo trajeto. Momentos onde vários ônibus se deslocam muito próximos diminuiriam a nota, enquanto momentos onde os ônibus se deslocam de maneira uniforme melhorariam a sua nota.

6 Referências Bibliográficas

1 BORGES, R. C. N. Sobre a Biblioteca Digital da Câmara dos Deputados. **Biblioteca Digital da Câmara dos Deputados**. Disponivel em: <bd.camara.gov.br/bd/bitstream/handle/bdcamara/1720/definicao_transporte_borges.pdf%3Fsequence%3D4+&cd=4&hl=pt-BR&ct=clnk&gl=br>. Acesso em: 17 Agosto 2016.

2 OPEN KNOWLEDGE INTERNACIONAL. Open Knowledge Internacional. Disponivel em: <https://okfn.org/opendata/how-to-open-data/>. Acesso em: Outubro 2016.

3 PORTAL BRASILEIRO DE DADOS ABERTO. Portal Brasileiro de Dados Aberto. Disponivel em: <http://dados.gov.br/>. Acesso em: Outubro 2016.

4 ESCRITÓRIO DE INTELIGÊNCIA DE DADOS DA PREFEITURA DO RIO DE JANEIRO. Disponivel em: <http://pensa.rio/main/pensa/>. Acesso em: Outubro 2016.

5 SULOPUISTO, O. **CityLab**. Disponivel em: <http://www.citylab.com/tech/2014/04/how-helsinki-mashed-open-data-regionalism/8994/>. Acesso em: Outubro 2016.

6 HELSINKI REGION INFOSHARE. **Helsinki Region Infoshare**. Disponivel em: <http://www.hri.fi/en/>. Acesso em: Outubro 2016.

7 ONEBUSAWAY. **OneBusAway**. Disponivel em: <http://onebusaway.org/>. Acesso em: Outubro 2016.

8 HOGGE, B. Open Data - Six Stories About Impact in the UK. **Omydiar Network**, Outubro 2016. Disponivel em: <https://www.omidyar.com/sites/default/files/file_archive/insights/Open%20Data_Six%20Stories%20About%20Impact%20in%20the%20UK/OpenData_CaseStudies_Report_complete_DIGITAL_102715.pdf>. Acesso em: Outubro 2016.

9 PORTAL DATA.RIO. Portal Data.Rio. **Portal Data.Rio**. Disponivel em: <http://data.rio/>. Acesso em: Outubro 2016.

10 GOOGLE MAPS APIS. **Google Maps APIs**. Disponivel em: <https://developers.google.com/maps/?hl=pt-br>. Acesso em: Novembro 2016.