UNIFACS UNIVERSIDADE SALVADOR
PROGRAMA DE PÓS-GRADUÇÃO EM SISTEMAS E COMPUTAÇÃO
MESTRADO EM SISTEMAS E COMPUTAÇÃO

ANA MARIA PARANHOS DE AMORIM

TX-TRAJECTORY: UM MODELO PARA REPRESENTAÇÃO DE TRAJETÓRIAS DE TÁXI

Salvador
2013

ANA MARIA PARANHOS DE AMORIM

TX-TRAJECTORY: UM MODELO PARA REPRESENTAÇÃO DE TRAJETÓRIAS DE TÁXI

Dissertação apresentada ao mestrado em Sistemas e Computação, da UNIFACS Universidade Salvador, como requisito parcial para obtenção do grau de Mestre.

Orientador: Prof. Dr. Jorge Alberto Prado de Campos

Salvador
2013

FICHA CATALOGRÁFICA
Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate Internacional Universities

Amorim, Ana Maria Paranhos de

Tx-trajectory: um modelo para representação de trajetórias de táxi. Ana Maria Paranhos de Amorim. – Salvador, 2013.
102 f. : il.

Dissertação apresentada ao Curso de Mestrado em Sistemas e Computação, UNIFACS Universidade Salvador, Laureate Internacional Universities como requisito parcial para obtenção do grau de Mestre.
Orientador: Prof. Dr. Jorge Alberto Prado de Campos.

1. Objetos Móveis. I. Campos, Jorge Alberto Prado de, orient. II. Título.

CDD.004

TERMO DE APROVAÇÃO

ANA MARIA PARANHOS DE AMORIM

TX-TRAJECTORY: UM MODELO PARA REPRESENTAÇÃO DE TRAJETÓRIAS DE TÁXI

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, Universidade Salvador - UNIFACS, pela seguinte banca examinadora:

Jorge Alberto Prado de Campos - Orientador_____
Doutor em Spatial Information Science and Engineering,University of Maine at Orono
UNIFACS Universidade Salvador

Expedito Carlos Lopes _____
Doutor em Ciência da Computação pela Universidade Federal de Campina Grande
Universidade de Campina Grande (UFCG)

Laís Nascimento Salvador _____
Doutora em Engenharia Elétrica pela Universidade de São Paulo (USP)
Universidade de São Paulo (USP)

Salvador, 16 de dezembro de 2013.

Aos meus pais,
filhos e netos
que dão colorido à minha vida!

AGRADECIMENTOS

A Deus, que não pelos meus merecimentos, mas por Sua imensa bondade, tudo me dá. Por Seu auxílio nas minhas escolhas e conforto nas horas difíceis. Agradeço a Ele pelo que sou e por tudo que necessito para viver: saúde, família, amigos e pessoas com quem compartilhar um caminho pelo tempo determinado por Ele.

Ao meu pai (*in memoriam*) e à minha mãe pela dedicação de uma vida inteira. Aos meus filhos Bruno e Taty pelo apoio e incentivo recebido sempre. À minha neta Yasmin, que com sua fala “Vovó você não precisa estudar mais não!”, expressou, do seu modo, seu descontentamento ao sentir diminuir o meu tempo disponível para ela. A João, meu neto mais novo, pelos doces momentos de interrupção durante este trabalho. Quisera eu ter todo o tempo do mundo para estar com eles dois.

Aos meus grandes amigos de trabalho, sempre dispostos a ler um material ou assistir a um ensaio para alguma apresentação. Também aos amigos, que mesmo sem entender muita coisa, por atuarem em outra área, não deixaram de me assistir e incentivar. Todos deram opiniões valiosas.

Aos meus colegas do Mestrado pelos momentos compartilhados. Aos colegas do grupo de pesquisa GANGES da Unifacs, principalmente a Helder pelas sugestões, ideias e incentivo.

Ao meu orientador, Prof. Jorge Campos, por ter-me apresentado uma área de pesquisa tão desafiadora, pela paciência, ensinamentos e dedicação. Além de ter tido a coragem de propor uma pesquisa na área de objetos móveis para uma aluna cuja única imagem desses objetos não passava de um grupo de discos voadores tripulados por pequenos seres estranhos.

Enfim, a todos que contribuíram de alguma forma para a conclusão deste trabalho.

“Mesmo que já tenha feito uma longa caminhada, sempre haverá mais um caminho a percorrer.”

Santo Agostinho

RESUMO

A captura em larga escala de dados sobre a movimentação de objetos móveis tem aumentado a demanda por desenvolvimento de ferramentas geoespaciais para analisar e apresentar as características do comportamento desses objetos nas mais diversas áreas. Sistemas Inteligentes de Transporte, por exemplo, fazem uso intensivo de dados coletados a partir de dispositivos existentes nos veículos para analisar e monitorar as condições das estradas, o fluxo de veículos e de passageiros do sistema de transporte público. A frota de táxi é um importante modal de transporte alternativo e complementar ao sistema público de transporte de massa. Dessa forma, a análise da movimentação dos táxis pode ser usada para capturar informações sobre a condição do tráfego e de entender a um nível mais fino de granularidade a movimentação das pessoas no ambiente urbano. Esta dissertação aborda o problema do mapeamento dos dados da trajetória bruta dos táxis em um modelo de dados mais abstrato e estruturado. O modelo de dados proposto visa criar uma infraestrutura que facilite o desenvolvimento de algoritmos voltados para mineração de dados e para a descoberta de conhecimento sobre a movimentação dos táxis e o comportamento das pessoas que utilizam este meio de transporte como deslocamento. Este trabalho também apresenta uma implementação do modelo conceitual, as estratégias utilizadas para converter dados da trajetória bruta em entidades mais abstratas e semanticamente mais ricas do modelo, além de uma avaliação do modelo através de algumas análises utilizando um conjunto de dados contendo quase um mês de trajetórias de táxis coletadas em uma grande área metropolitana.

Palavras Chaves: Trajetórias de Táxi. Objetos Móveis. Enriquecimento Semântico.

ABSTRACT

The large-scale capture of data about the motion of moving objects has increased the development of geospatial tools to analyze and present the characteristics of these objects' behavior in many different fields. Intelligent Transportation Systems, for instance, make intensive use of data collected from embedded in-vehicle devices to analyze and monitor roads conditions and the flow of vehicles and passengers of the public transportation system. The taxi fleet is an important transport modality complementary to the public transportation system. Thus, analysis of taxis' movements can be used to capture information about the condition of the traffic and to understand at a finer level of granularity the movement of people in an urban environment. This dissertation addresses the problem of mapping taxi raw trajectory data onto a more abstract and structured data model. The proposed data model aims to create an infrastructure to facilitate the implementation of algorithms for data mining and knowledge discovery about taxi movements and people's behavior using this means of transport. This work also presents an implementation of the conceptual model, the strategies used to convert raw trajectory data into more abstract and semantic rich entities of the model, and an evaluation of the model through the realization of some analyses of a dataset containing almost a month of taxis trajectory data collected in a major metropolitan area.

Keywords: Taxi Trajectory. Moving Objects. Semantic Enrichment.

LISTA DE FIGURAS

Figura 2.1 - Padrão geométrico de trajetórias brutas e padrões de deslocamento com semântica associada.....22

Figura 2.2 - Enriquecimento Semântico de Trajetórias. a) dados de uma trajetória bruta; b) dados de uma trajetória com enriquecimento semântico para a área de turismo; c) dados de uma trajetória com enriquecimento semântico para o gerenciamento de transporte.....23

Figura 2.3 - Padrão de projeto para representação de trajetórias.....24

Figura 2.4 - Modelo Episodes.....25

Figura 2.5 - Modelo CONSTAnT 27

Figura 2.6 - Padrões de fluxo de tráfego 29

Figura 2.7 - Relacionamento entre locais de pick-ups e drop-offs.....31

Figura 2.8 - Matriz Origem-Destino 32

Figura 2.9 - Visão geral da aplicação de mapeamento do fluxo de tráfego.....33

Figura 2.10 - Particionamento da cidade de Pequim.....34

Figura 2.11 - Arquitetura do Modelo.....34

Figura 2.12 - Construção das transições a partir das trajetórias 35

Figura 2.13 - Matriz das regiões da cidade 35

Figura 2.14 - Identificação de contorno (skyline).....36

Figura 2.15 - Histograma da renda diária dos motoristas (em Yuan) 38

Figura 2.16 - Comparação - top driver e ordinary driver.....39

Figura 2.17 - Valor fractal (índice de tortuosidade) 40

Figura 2.18 - Sistema de Recomendação 41

Figura 2.19 - Trajetória e Viagem de Táxi 41

Figura 2.20 - Visão geral do sistema de recomendação 42

Figura 2.21 - Recomendação para motoristas 43

Figura 2.22 - Distribuição de renda e razão de ocupação.....44

Figura 3.1 - Dados brutos de uma jornada de trabalho 46

Figura 3.2 - TX-Trajectory – Modelo Conceitual.....49

Figura 3.3 - Jornada de Trabalho – processo de identificação das entidades do modelo 53

Figura 3.4 - Jornada de Trabalho - trajetória representada por entidades do modelo 54

Figura 3.5 - Fluxo do processo de carga dos dados em um BDG.....56

Figura 3.6 - TX-Trajectory – Modelo Relacional.....58

Figura 3.7 - Interface TX-Trajectory Generation Module.....60

Figura 3.8 - Apresentação de uma Working Trajectory e suas entidades constituintes 61

Figura 4.1 - Análise da eficiência do sistema de táxi: a) evolução da Taxa de Eficiência no dia; b) quantidade de motorista trabalhando em cada hora do dia 65

Figura 4.2 - Origem das chamadas em 20/05/2008 de 05:00 – 06:00h. a) pontos com a localização dos *pick-ups*; b) mapa térmico mostrando as regiões com maior demanda de táxi nesse período 67

Figura 4.3 - Trecho de uma aplicação para extração de informação 68

Figura 4.4 - Mapa coroplético mostrando a quantidade de pick-ups e drop-offs nos distritos da área metropolitana de San Francisco 70

Figura 4.5 - Mapa coroplético mostrando a quantidade de drop-offs das viagens de táxi iniciadas no Golden Gate.....71

Figura 4.6 - Mapas mostrando o caminho percorrido nas viagens de táxi iniciadas no Golden Gate Park com destino ao Financial District: a) rotas; b) destaque para os fluxos de convergência.....72

Figura 4.7 - Comparativo da Taxa de Eficiência.....74

Figura 4.8 - Quantidade de táxis trabalhando ao longo do dia.....75

LISTA DE TABELAS

Tabela 1 - Classificação do melhor motorista dentre os 3.000 analisados.....39

SUMÁRIO

1 INTRODUÇÃO 13
1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO 14
1.2 OBJETIVOS 15
1.3 METODOLOGIA 16
1.4 RESULTADOS RELEVANTES 16
1.5 AUDIÊNCIA 17
1.6 ESTRUTURA DO TRABALHO 17
2 TRAJETÓRIA DE OBJETOS MÓVEIS E ESTUDOS DA MOVIMENTAÇÃO DE TÁXIS 19
2.1 SEMÂNTICA DAS TRAJETÓRIAS 20
2.2 ANÁLISE DA MOVIMENTAÇÃO DE TÁXIS 28
3 MODELO TX-TRAJECTORY 45
3.1 DADOS BRUTOS DE TÁXI 45
3.2 DEFINIÇÕES BÁSICAS E ENTIDADES DO MODELO 47
3.2.1 Definições Básicas 47
3.2.2 Modelo Conceitual 48
3.2.3 Mapeamento dos Dados Brutos para Entidades do Modelo 52
3.2.4 Processo de Carga do Modelo em Banco de Dados 55
3.3 COMPARAÇÃO COM OUTROS MODELOS 61
4 APLICAÇÃO E AVALIAÇÃO DO MODELO 64
4.1 ANÁLISES USUAIS DE TRAJETÓRIAS DE TÁXI 64
4.2 ESTUDO DA EFICIÊNCIA E EFICÁCIA DO MODELO *TX-TRAJECTORY* 73
5 CONCLUSÃO 77
REFERÊNCIAS 79
APÊNDICE A – Aplicação TX-Trajectory Generation Module 82

1 INTRODUÇÃO

A evolução da tecnologia e o barateamento dos custos dos dispositivos de posicionamento tornaram a captura de dados, em larga escala, da movimentação dos objetos móveis tecnicamente e economicamente viável. Como consequência, é crescente o número de novas aplicações nas mais diversas áreas voltadas para o entendimento e gerenciamento de fenômenos complexos que envolvem esses objetos. A movimentação de mercadorias pelo mundo, a gestão de tráfego urbano e o monitoramento de aves migratórias são alguns exemplos dessas aplicações (SPACCAPIETRA *et al.*, 2008).

Os Sistemas Inteligentes de Transportes (ou ITS, do inglês *Intelligent Transportation Systems*) abrangem um novo tipo de aplicação projetada para incorporar a tecnologia da informação e comunicação sem fio à infraestrutura de transporte e veículos. ITS fazem uso intensivo de dados coletados a partir de sensores colocados ao longo da rede de transporte e de dispositivos embarcados nos veículos para analisar e monitorar as condições das estradas e do fluxo de veículos e usuários do sistema de transporte (BAZZAN; KLÜGL, 2007).

A frota de táxi é um importante modal de transporte alternativo e complementar ao sistema público de transporte de massa, que faz uso intensivo dos sistemas de posicionamento e comunicação sem fio no apoio às atividades operacionais do setor. Desta forma, a análise de seus movimentos pode ser usada para capturar informações sobre as condições do tráfego e movimentação das pessoas no ambiente urbano, servindo, desta forma, como um importante mecanismo no gerenciamento e análise do sistema como um todo.

Atualmente, a grande maioria das empresas de táxi já utiliza a tecnologia para determinar a localização em tempo real dos veículos da sua frota. Entretanto, para que a grande quantidade de dados, gerados através do uso dessa tecnologia, seja mais bem aproveitada é necessário a utilização de um modelo que permita uma boa organização desses dados. As aplicações voltadas para o estudo da movimentação dos táxis necessitam de informações complementares que podem ser obtidas com o pós-processamento dos dados brutos. Existe, portanto, a necessidade de um modelo de dados que permita adicionar semântica aos dados das trajetórias. Este modelo de dados deverá propiciar a implementação de algoritmos eficientes para a

descoberta de padrões e conhecimento necessários ao entendimento da dinâmica dos objetos em questão[1].

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

O principal objetivo das aplicações de ITS é permitir a familiarização dos usuários com o funcionamento do sistema e de fornecer serviços inovadores para melhorar a coordenação e manutenção do sistema de transporte. Geralmente, as aplicações de ITS trabalham com informações brutas da movimentação dos veículos expressas na forma de tuplas (identificador, localização, tempo) descrevendo o deslocamento de um determinado veículo ao longo do tempo (AMORIM; CAMPOS, 2012). Entretanto, para atender aos requisitos da maioria das aplicações esses dados não são suficientes.

As aplicações de ITS atualmente requerem informações que capturem as características dos movimentos e identifiquem padrões e anomalias no comportamento dos mesmos. Algumas das informações complementares ao deslocamento do objeto tais como: condições das estradas, pontos turísticos, eventos culturais e esportivos, podem ser obtidas com o pós-processamento dos dados brutos da movimentação ou através do cruzamento dessas informações com informações sobre o contexto das aplicações (YAN *et al.*, 2011) (BOGORNY *et al.*, 2011) (YAN, 2009) (ALVARES *et al.*, 2007) e de outras informações obtidas durante o processo de captura dos dados através de sensores especiais ou da interferência direta de uma pessoa.

Atualmente a maioria das empresas e cooperativas de táxi utiliza a tecnologia de transportes inteligentes para determinar a localização instantânea, a situação dos veículos e para enviar o táxi disponível mais próximo ao cliente que está requisitando o serviço. Essa tecnologia, entretanto, gera também uma enorme quantidade de dados com os registros da movimentação dos veículos. Esses dados de movimentação são pouco utilizados para análise, mineração de dados e descoberta de conhecimento. Os motivos que levam a isso são muitos: i) os dados das trajetórias necessitam de uma estrutura mais abstrata, possuem muitos registros redundantes e inconsistentes, e carregam pouca ou nenhuma informação

[1] Esta pesquisa é parte integrante da iniciativa dos pesquisadores do Grupo de Aplicações e Análises Geoespaciais (GANGES), UNIFACS, na área de semântica de trajetórias.

semântica; ii) existem poucos algoritmos adaptados para analisar, extrair e revelar padrões de comportamento deste tipo de objeto; iii) os gestores públicos e das frotas de táxi ainda não despertaram para a utilização dessa rica fonte de informação. Esses motivos serão discutidos detalhadamente no Capítulo dois desse trabalho. Diante deste cenário, identificamos a necessidade de um modelo de dados que permita uma melhor organização dos dados brutos das trajetórias de táxis e que seja capaz de subsidiar uma ampla gama de análises e descoberta de conhecimento sobre esta modalidade de transporte.

A estruturação dos dados brutos das trajetórias de táxis registradas por algum mecanismo de posicionamento é um dos pré-requisitos básicos para a criação da infraestrutura de dados necessária para o desenvolvimento de algoritmos que identifiquem padrões de comportamento das pessoas que utilizam esse meio de transporte. Este trabalho propõe um modelo conceitual e de dados para a representação das trajetórias de táxi. O referido modelo visa facilitar consultas e extração de conhecimento sobre a dinâmica desta modalidade de transporte nas grandes áreas urbanas. Espera-se que através de entidades mais abstratas e que incorporem características relevantes da dinâmica destes veículos seja possível, entre outras coisas, identificar facilmente padrões de comportamento em relação ao movimento dos passageiros pela cidade, suas origens e destinos predominantes e analisar a eficiência do sistema de táxi como um todo.

1.2 OBJETIVOS

O objetivo geral deste trabalho é propor um modelo conceitual e de dados para representar o movimento dos táxis. Como objetivos específicos têm-se: a criação de uma infraestrutura de dados com entidades semanticamente relevantes para o domínio da aplicação; a criação de uma estrutura que facilite a consulta e extração de conhecimento sobre a dinâmica deste modal de transporte nos grandes centros urbanos e que também facilite o desenvolvimento de algoritmos eficientes para a descoberta de padrões e conhecimento necessários ao entendimento da movimentação deste tipo de objeto móvel.

1.3 METODOLOGIA

Para o desenvolvimento do modelo para representação de trajetórias de táxis foi necessário uma revisão de grande parte da literatura pertinente ao estudo de trajetória de objetos móveis, em geral, e das trajetórias de táxis, em particular. A partir dos estudos realizados, procurou-se extrair características de movimentação que se aplicassem aos deslocamentos do objeto em estudo, no caso, táxi.

Com o objetivo de construir o ambiente que permitisse mostrar a aplicação do referido modelo e sua avaliação, abordou-se o problema do mapeamento das informações brutas das trajetórias de táxis para um modelo conceitual genérico. Para realizar esse mapeamento, foi desenvolvido um algoritmo para conversão de dados brutos de trajetória em entidades do modelo conceitual.

Finalmente, como prova de conceito, foram utilizadas ferramentas livres para o desenvolvimento das aplicações voltadas para o armazenamento de dados, para a análise e apresentação dos resultados. Os dados das trajetórias foram armazenados em tabelas no SGBD PostgreSQL com a extensão PostGIS (POSTGIS, 2013). Para o desenvolvimento das aplicações foi utilizado Java com uso do IDE NetBeans. Para a apresentação dos resultados foi utilizado o Quantum GIS (QGIS). O QGIS é um Sistema de Informação Geográfica (SIG) livre e multiplataforma que permite criar, editar, visualizar, analisar e publicar dados espaciais (QGIS, 2013).

1.4 RESULTADOS RELEVANTES

O principal resultado deste trabalho é um modelo conceitual e de dados capaz de representar as trajetórias de táxi. Utilizando uma base de dados construída a partir das trajetórias brutas enriquecidas semanticamente, o modelo proposto foi capaz de auxiliar a consulta e extração de conhecimento. Os exemplos utilizados para comparar o procedimento de extração de conhecimento através do uso do modelo e através da utilização dos dados brutos, sugerem que o modelo se mostrou mais eficiente em relação à utilização dos dados brutos, tanto em tempo de execução quanto em relação à fidelidade dos resultados.

Outro resultado relevante neste trabalho foi o desenvolvimento de um algoritmo para a conversão dos dados brutos das trajetórias dos táxis em entidades do modelo conceitual.

Vale salientar outro resultado bastante relevante deste trabalho, o desenvolvimento do *TX-Trajectory Generation Module*. E também a inauguração no GANGES de uma nova linha de pesquisa voltada para o estudo da movimentação dos componentes do sistema de transporte.

1.5 AUDIÊNCIA

Em função dos temas abordados e estudados neste trabalho, a sua audiência constitui-se de um público diversificado e multidisciplinar. Neste trabalho estão reunidos assuntos e contribuições nas áreas de trajetória de objetos móveis, mais especificamente na análise de deslocamento de táxis. Assim, o público alvo deste trabalho envolve todas as pessoas e pesquisadores interessados em desenvolvimento de algoritmos e ferramentas voltadas à análise da trajetória de objetos móveis, principalmente transporte público e mais especificamente aqueles voltados ao modal de táxi[2].

1.6 ESTRUTURA DO TRABALHO

Esta dissertação está estruturada em cinco capítulos. O Capítulo 2 trata dos estudos relacionados à semântica das trajetórias, isto é, como estruturar os dados brutos da trajetória dos objetos móveis e adicionar informações semânticas que sejam importantes ao domínio da aplicação. Além disso, este Capítulo apresenta trabalhos relacionados à análise da movimentação de táxis.

O Capítulo 3 apresenta o modelo *TX-Trajectory*, as definições básicas utilizadas no desenvolvimento do modelo, o processo de mapeamento dos dados brutos para entidades do modelo e a carga dos dados em um banco de dados com extensão espacial.

O Capítulo 4 faz uma avaliação do modelo e da sua utilização. Para isso, este Capítulo apresenta algumas análises e consultas realizadas através dessa base de dados, construída de acordo com o modelo proposto.

Por fim, o Capítulo 5 apresenta a conclusão do trabalho, descrevendo os pontos fortes e identificando as limitações do modelo. Objetivando a continuidade

[2] Parte deste trabalho foi publicado no GeoInfo 2012 (AMORIM; CAMPOS, 2012) tendo sido selecionado como segundo melhor artigo do simpósio.

deste trabalho, esse capítulo aponta, também, algumas sugestões de trabalhos futuros.

2 TRAJETÓRIA DE OBJETOS MÓVEIS E ESTUDOS DA MOVIMENTAÇÃO DE TÁXIS

Qualquer objeto que possua um dispositivo acoplado capaz de informar a sua localização deixa registrada a sua “pegada” (BOGORNY; BRAZ, 2012). Como resultado da popularização do uso desses dispositivos, existe atualmente uma crescente massa de dados com o registro da movimentação dos mais diversos tipos de objetos, tais como: pessoas, animais e veículos. Esta crescente massa de dados tem aumentando, na mesma proporção, a demanda por novas aplicações e métodos para entender e gerenciar os fenômenos complexos que envolvem esses objetos.

A sequência de “pegadas” deixadas pelos objetos móveis é denominada de trajetória bruta. A representação, armazenamento, consulta, análise, gerenciamento e apresentação das trajetórias brutas têm motivado o desenvolvimento de pesquisas nas mais diversas áreas, dentre elas destacam-se: bancos de dados espaço-temporais (BOGORNY; ENGEL; *et al.*, 2006), geovisualização (ANDRIENKO *et al.*, 2007) (BIADGILGN; BLOK; HUISMAN, 2011), modelos de dados com enriquecimento semântico (ALVARES *et al.*, 2007) (YAN *et al.*, 2011), mineração de dados (BOGORNY *et al.*, 2008) (WESSEL *et al.*, 2009) (LI *et al.*, 2010), *data warehouses* geográficos (MARKETOS, 2009) (LEAL *et al.*, 2011), descoberta de conhecimento e de padrões de comportamento (AKASAPU *et al.*, 2010) (ZHENG *et al.*, 2013) e Sistemas Inteligentes de Transporte (ou ITS do inglês *Intelligent Transportation Systems*) (BAZZAN; KLÜGL, 2007) (SILVA *et al.*, 2005).

O tratamento e a estruturação da informação bruta dos deslocamentos estão entre os maiores problemas relacionados ao processamento e análise de dados sobre a movimentação dos objetos móveis. As trajetórias brutas são usualmente armazenadas na forma de tuplas (*id,x,y,t*), onde *id* identifica o objeto móvel, *x* e *y* correspondem às coordenadas de um local no espaço geográfico e *t* o instante no qual o objeto ocupou este local. Na forma como é apresentada, entretanto, a informação bruta dos deslocamentos dos objetos móveis é de pouca utilidade para a maioria das aplicações. Dentre os principais obstáculos encontrados na utilização de dados brutos, destacam-se: 1) o alto custo computacional e a complexidade das consultas realizadas sobre os mesmos; 2) a dificuldade de identificar padrões no comportamento e no deslocamento dos objetos; 3) a ausência de informações semânticas sobre o deslocamento e 4) a falta de informação da relação do

movimento com a dinâmica de outros objetos nas proximidades que sejam relevantes ao domínio da aplicação.

Neste contexto, identifica-se a necessidade da construção de um modelo que visa minimizar os obstáculos supracitados especificamente para a representação das trajetórias dos táxis. Essa representação deverá ser feita através de entidades que incorporem informações semânticas sobre o deslocamento dos veículos, além de características relevantes do ambiente em que estes veículos trafegam. Acreditamos que o modelo irá facilitar as operações voltadas para a mineração de dados, descoberta de conhecimento e identificação de padrões de deslocamentos. De forma a situar este trabalho em relação às diversas pesquisas envolvendo trajetórias de objetos móveis, em geral, e modelos para representação e análise dos deslocamentos de táxis, em particular, dividimos este capítulo em duas seções. A primeira seção trata dos estudos relacionados à semântica das trajetórias, isto é, como estruturar os dados da trajetória bruta dos objetos móveis e incorporar informações semânticas relevantes ao domínio da aplicação. A segunda seção apresenta trabalhos relacionados à análise da movimentação de táxis. Dentre os principais problemas abordados nos trabalhos que tratam a análise da movimentação de táxis destacam-se: a identificação do padrão de deslocamento dos usuários do sistema; a identificação dos principais pontos de origem e destino e a relação entre essas localidades; a geração de recomendações para motoristas e passageiros; e mecanismos para análise da eficiência dos motoristas.

A motivação para a estruturação deste capítulo em seções aparentemente estanques são duas. De um lado, não foram identificados trabalhos na área de semântica das trajetórias que tratem especificamente do domínio da trajetória de táxis. Por outro lado, não foram encontrados trabalhos na área de análise da movimentação de táxis que tenham formalizado um modelo de dados para este domínio ou que tratem da semântica das trajetórias. Desta forma, acreditamos que a união destas duas linhas de pesquisa, até então dissociadas, possui o potencial de facilitar a análise e o entendimento deste modal de transporte.

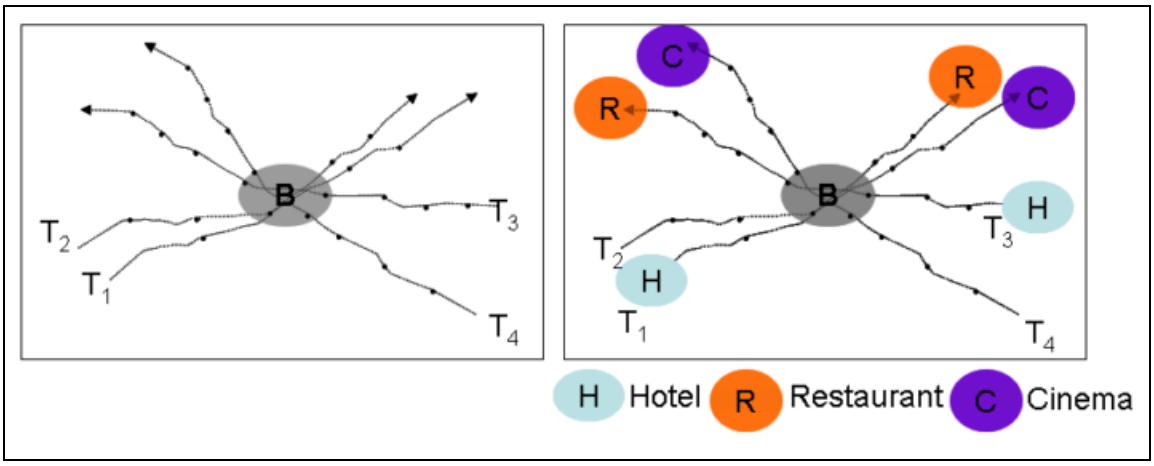
2.1 SEMÂNTICA DAS TRAJETÓRIAS

O processo de tratamento e estruturação dos dados das trajetórias brutas geralmente é acompanhado por algum tipo de tratamento para o enriquecimento

semântico dos dados. Este enriquecimento visa identificar características básicas do movimento (e.g., se o objeto está parado ou em movimento) e acrescentar informações relevantes ao contexto da aplicação (e.g., o motivo de uma parada ou do deslocamento e se o percurso realizado está em conformidade com o padrão esperado).

A utilização de dados brutos de trajetórias na identificação de padrões de comportamento dos objetos móveis dificilmente revela um padrão semântico. Esta utilização, geralmente, revela somente o padrão geométrico do deslocamento. Este tipo de padrão, entretanto, pode não ser significativo em determinados domínios de aplicação (BOGORNY; BRAZ, 2012). Considerando uma base de dados com inúmeros dados brutos de trajetórias, por exemplo, é possível realizar consultas envolvendo somente os aspectos geométricos da trajetória. Estas consultas são realizadas através da extensão espacial da maioria dos SGBDs (Sistema de Gerenciamento de Banco de Dados). Utilizando-se a Figura 2.1 à esquerda, um exemplo de consulta simples retorna todas as trajetórias que passaram por uma determinada região da cidade durante certo período do dia. Se a região B é uma interseção no sistema viário e se o objetivo é conhecer o número de veículos que passam por esta região, esta consulta é suficiente. Entretanto, se o objetivo é conhecer a motivação das pessoas que realizaram tal deslocamento, faz-se necessário algum tipo de procedimento voltado ao enriquecimento semântico das trajetórias. A Figura 2.1 à direita ilustra as mesmas trajetórias retornadas anteriormente, mas com algumas anotações semânticas associadas. Nesta figura é possível identificar que as trajetórias T1 e T3 possuem o mesmo padrão de deslocamento origem-destino, onde o deslocamento tem como origem um hotel e como destino um restaurante. Já as trajetórias T2 e T4 possuem um padrão de deslocamento no qual o objeto vem de algum lugar sem relevância para a aplicação e vai para um cinema.

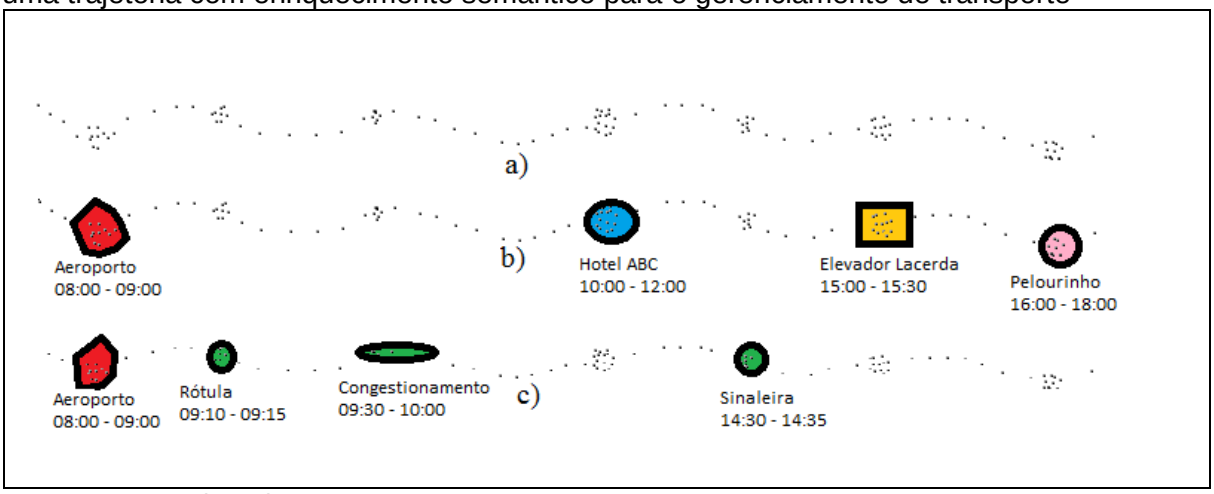
Figura 2.1 - Padrão geométrico de trajetórias brutas e padrões de deslocamento com semântica associada



Fonte: Bogorny (2008).

Outros tipos de enriquecimento semântico requerem uma análise mais detalhada dos dados brutos de trajetória. Muitas vezes é necessário analisar, por exemplo, o padrão espaço-temporal dos pontos da trajetória para identificar e caracterizar o movimento. Objetivando ilustrar este processo de enriquecimento semântico, considere um veículo equipado com algum dispositivo de localização que registra sua trajetória em um ambiente urbano. A coleção de pontos coletada pelo dispositivo pode ser observada na Figura 2.2a. Como a localização do objeto é geralmente capturada pelo dispositivo a uma taxa constante (e.g., uma vez a cada minuto), os pontos da trajetória bruta do objeto móvel descrevem um padrão de aglomeração em trechos da trajetória no qual o objeto está parado e um padrão linear em trechos no qual o objeto está se movimentando. Uma aplicação desenvolvida para o domínio de turismo, por exemplo, pode identificar os locais onde o veículo parou e reconhecer alguns pontos de parada como relevantes para o contexto da aplicação (Figura 2.2b). Já uma aplicação na área de gerenciamento de transportes estaria interessada em associar as paradas a eventos relacionados ao trânsito ou pontos relevantes da malha viária (Figura 2.2c).

Figura 2.2 - Enriquecimento Semântico de Trajetórias. a) dados de uma trajetória bruta; b) dados de uma trajetória com enriquecimento semântico para a área de turismo; c) dados de uma trajetória com enriquecimento semântico para o gerenciamento de transporte



Fonte: Bogorny (2012).

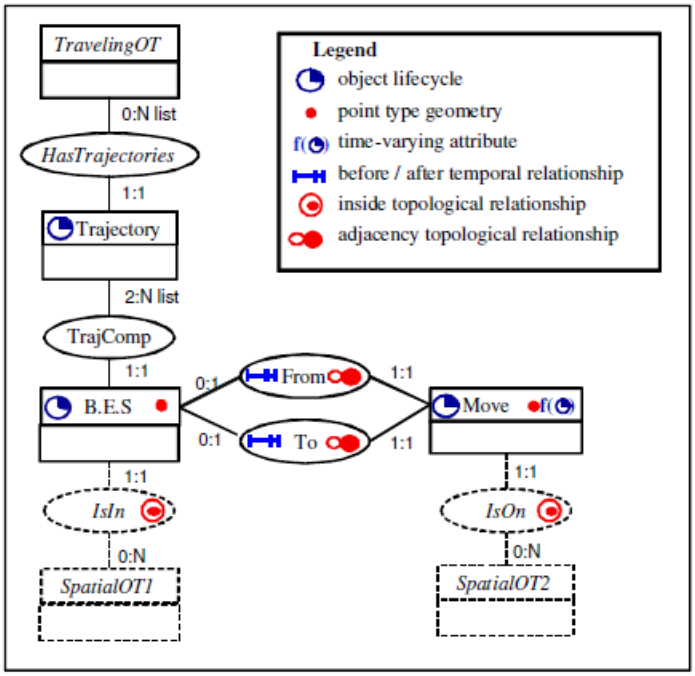
O primeiro modelo desenvolvido com o objetivo de adicionar semântica aos dados brutos foi proposto pelo grupo de pesquisa do professor Stefano Spaccapietra (SPACCAPIETRA *et al.*, 2008). Este modelo é genericamente denominado de modelo de *Stops* e *Moves*. O referido modelo trata trajetórias de objetos móveis como movimentos que correspondem a deslocamentos semanticamente significativos e, por definição, um conceito espaço-temporal. Spaccapietra conceituou uma trajetória como sendo a evolução espaço-temporal de um objeto móvel para atingir um determinado objetivo. A Figura 2.3 ilustra o padrão de projeto utilizado para a representação de trajetórias semanticamente enriquecidas (SPACCAPIETRA *et al.*, 2008).

O padrão de projeto mostrado na Figura 2.3 indica que um objeto móvel (*TravelingOT*) possui zero ou mais trajetórias associadas (*Trajectory*). Cada trajetória, por sua vez, é composta por uma lista de dois ou mais componentes. *TrajComp* modela uma relação de composição entre a trajetória e seus componentes. Os componentes de uma trajetória são os elementos *Begin*, *End*, *Stop* e *Move*. No caso da trajetória ser composta por apenas dois componentes, um deles será *Begin* e o outro *End*.

Semanticamente, Spaccapietra considera uma trajetória como uma lista ordenada de *Stops* e *Moves* que ocorrem entre os instantes *Begin* e *End*. Os elementos *Begin*, *End*, *Stop* são estruturalmente equivalentes e tratados coletivamente no diagrama como *B.E.S.* (Figura 2.3). Os elementos *Begin* e *End*

representam dois pontos espaço-temporais e são utilizados para demarcar os locais e os instantes do início e do final da trajetória, respectivamente. Um *Stop* é uma parte da trajetória que é relevante para a aplicação e onde é considerado que o objeto não se movimentou efetivamente. Diferente do *Stop*, que pressupõe que os objetos ficaram parados por certo período de tempo, a duração do *Begin* e do *End* possui um único *chronon* (unidade de tempo indivisível).

Figura 2.3 - Padrão de projeto para representação de trajetórias



Fonte: Spaccapietra (2008).

O elemento *Move* representa trechos da trajetória onde se observa a movimentação do objeto. Ainda segundo o padrão de projeto da Figura 2.3, um *Move* é delimitado por dois *Stops* consecutivos, pelo *Begin* da trajetória e o primeiro *Stop*, pelo último *Stop* e o *End* da trajetória ou pelo *Begin* e o *End* da trajetória, sendo que esse último caso reflete trajetórias sem elementos do tipo *Stop*.

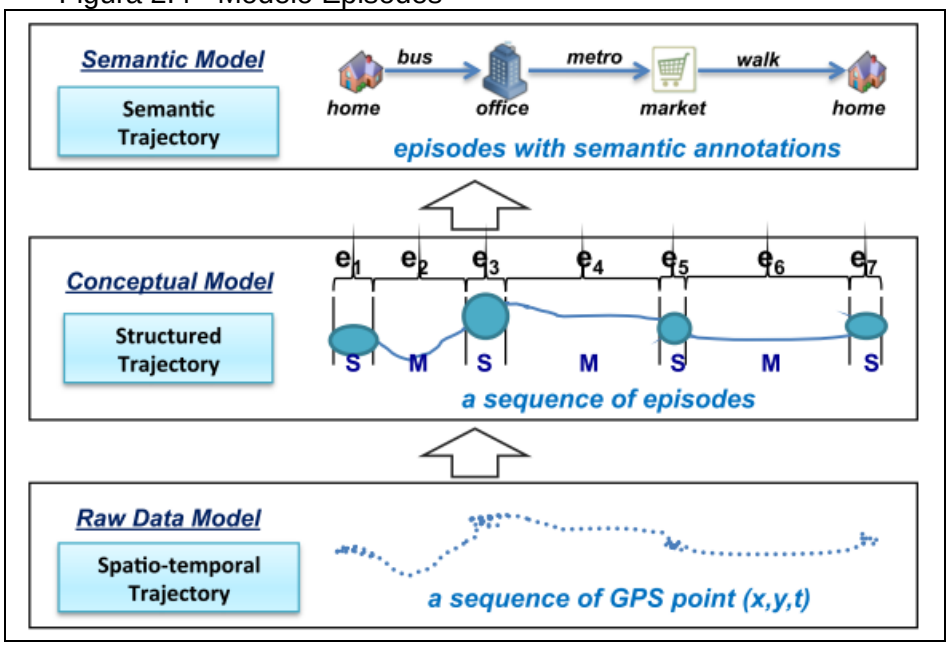
No domínio espacial, os elementos *B.E.S*. são representados por um único ponto (*SpatialOT1*), enquanto o elemento *Move* é representado por uma função de deslocamento ou uma polilinha construída com base nos pontos da trajetória bruta (*SpatialOT2*).

O modelo de *Stops* e *Moves* tem influenciado diversos trabalhos na área de semântica de trajetórias. Estes trabalhos objetivam incorporar níveis crescentes de informações semânticas, suportar consultas de alto nível e servir de base para o

desenvolvimento de métodos de mineração de dados para um amplo espectro de aplicações. Dentre os modelos com estas características destacam-se o modelo *Episodes* (YAN *et al.*, 2012) e *CONSTAnT* (BOGORNY *et al.*, 2013).

O modelo *Episodes* propõe a representação de trajetória de objetos móveis em camadas com crescentes níveis de abstração: dados brutos, conceitual e semântico (Figura 2.4).

Figura 2.4 - Modelo Episodes



Fonte: Yan (2012).

A camada de dados brutos fornece as definições de trajetória a partir da sequência de pontos *(x,y,t)* coletados através de algum dispositivo de posicionamento. Nessa camada é feita a decomposição da sequência de pontos que representam a movimentação do objeto móvel em entidades chamadas trajetória espaço-temporal. Uma trajetória espaço-temporal é um subconjunto dos dados brutos delimitado por dois pontos *(x,y,t)*, denominados *Begin* e *End*.

A camada conceitual corresponde a uma representação mais abstrata e estruturada dos dados brutos. Nesta camada a trajetória espaço-temporal é decomposta em *Stops* e *Moves*. No modelo *Episodes*, os *Stops* e *Moves* são denominados genericamente de episódios. A trajetória espaço-temporal particionada em episódios é denominada trajetória estruturada.

A camada semântica acrescenta informações semânticas aos episódios baseada no contexto e no domínio da aplicação. Estas anotações transformam trajetórias estruturadas em trajetórias semânticas.

Conceitualmente, o modelo *Episodes* é idêntico ao modelo *Stops* e *Moves* (SPACCAPIETRA *et al.*, 2008). O modelo *Episodes* se baseia nos dados brutos da trajetória para produzir uma sequência de episódios (i.e., *Stops* e *Moves*) e utiliza o contexto geográfico e o domínio da aplicação para associar anotações semânticas a estes episódios. Um diferencial relevante no modelo, entretanto, é o de disponibilizar camadas de dados que representam níveis progressivos de abstração da trajetória. Este particionamento favorece a utilização do modelo por diferentes tipos de aplicações, desde aplicações que necessitem da representação da trajetória na perspectiva dos dados brutos até aplicações que utilizem a perspectiva semântica da trajetória.

Em 2013 (BOGORNY *et al.*, 2013), apresentou o *CONSTAnT* (CONceptual model of Semantic TrAjecTories), um modelo de dados conceitual para a representação das trajetórias semânticas dos objetos móveis. O modelo, mostrado na Figura 2.5, surgiu com o objetivo de ser um modelo mais flexível, onde qualquer tipo de informação conceitual pudesse ser acrescentado. Nesse modelo a trajetória semântica é construída por diferentes subtrajetórias semânticas. Essas subtrajetórias são geradas de acordo com os objetivos do objeto móvel, com seu comportamento (ZHENG *et al.*, 2013) e com o meio de transporte utilizado.

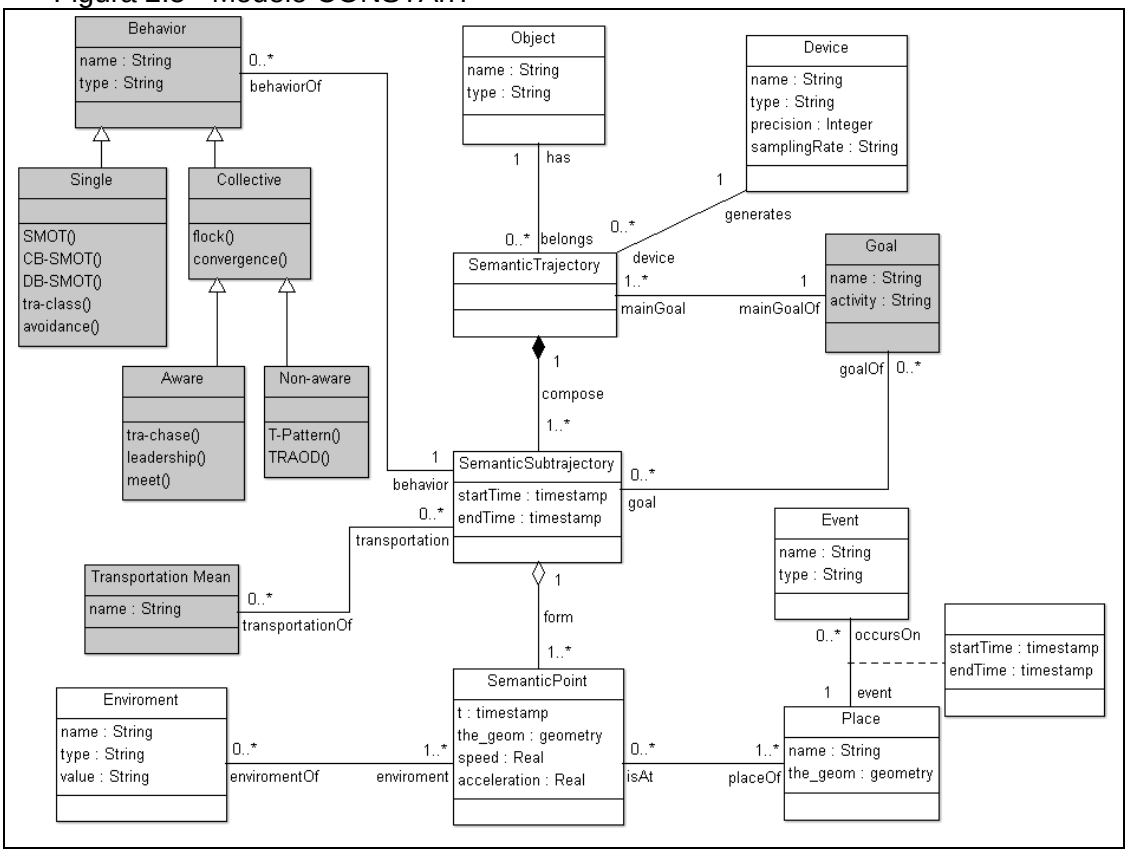
A primeira parte do modelo é formada pelas entidades *Object, Device, SemanticTrajectory, SemanticSubtrajectory, SemanticPoint, Environment, Place* e *Event* (Figura 2.5). A segunda parte do modelo é mais complexa e geralmente necessita de métodos avançados, como os de mineração de dados, para instanciar os objetos. Essa parte do modelo é formada pelas entidades *Goal, TransportationMean, Behavior*. Estas últimas entidades estão destacadas em cinza na Figura 2.5.

A entidade *Object* representa o objeto móvel (e.g., uma pessoa, animal, carro). *Device* é o dispositivo utilizado para coletar a sequência de pontos que forma a trajetória (e.g., um GPS). A entidade *SemanticSubtrajectory* é instanciada de acordo com o domínio da aplicação. Ela pode incorporar o modelo de *Stops* e *Moves*, onde os *stops* e *moves* representam *SemanticSubtrajectory* distintas. Essa

entidade pode, também, ser criada de acordo com o objetivo (*Goal*), comportamento (*Behavior*) e o meio de transporte (*TransportationMean*) utilizado pelo objeto móvel.

O modelo *CONSTAnT* considera que uma trajetória pode ter diferentes comportamentos durante o seu percurso. Para identificar o comportamento referente a cada subtrajetória, o modelo utiliza a entidade *Behavior*. O comportamento do objeto móvel pode ser especializado em comportamentos simples ou coletivo. O comportamento simples diz respeito ao deslocamento de um objeto sem considerar o deslocamento de outros objetos na vizinhança. São informações típicas deste tipo de comportamento as motivações dos diversos *stops* e *moves* realizados ou se o objeto evita passar por uma determinada localidade (*avoidance*) (ALVARES *et al.*, 2011). Os comportamentos coletivos visam identificar padrões de deslocamento de um grupo de objetos. Neste caso, é possível identificar se os objetos desenvolvem, por exemplo, um comportamento de perseguição (*chasing*), isto é, um ou mais objetos perseguem outro determinado objeto (SIQUEIRA; BOGORNY, 2011) ou se um objeto assume a liderança (*leadership*) e coordena a movimentação de outros objetos (LAUBE *et al.*, 2005).

Figura 2.5 - Modelo CONSTAnT



Fonte: Bogorny (2013).

O modelo *CONSTAnT* é bastante amplo e trata dos aspectos que devem ser considerados no enriquecimento semântico da trajetória, qual a relação entre esses aspectos e qual a parte da trajetória bruta que será influenciada por eles. Esses aspectos não foram contemplados em modelos anteriores, como, por exemplo, no modelo de *Stops e Moves*.

Além dos modelos já apresentados, diversos outros trabalhos foram propostos baseados no modelo de *Stops* e *Moves*. Em 2010, (MORENO *et al.*, 2010) realizou uma análise dos *stops* da trajetória considerando não apenas medidas como velocidade, direção e tempo, mas também informações semânticas armazenadas numa base de conhecimento da aplicação para inferir o objetivo de cada parada; em (BOGORNY *et al.*, 2008) foi proposta uma metodologia para a descoberta de conhecimento utilizando trajetórias; em (MORENO; ARANGO, 2011) com a utilização do conceito de trajetória apresentado por Spaccapietra, a trajetória foi incorporada ao *Data Warehouse* como conceito de primeira classe, surgindo um novo tipo de dado.

2.2 ANÁLISE DA MOVIMENTAÇÃO DE TÁXIS

No contexto dos transportes urbanos, o estudo da movimentação de táxis com a finalidade de entender e melhorar o problema da mobilidade urbana é um campo de pesquisa bastante atual. O deslocamento desses veículos nas grandes cidades gera informações que auxiliam no gerenciamento e análise do sistema público de transporte. Além disso, é uma importante fonte de informação no estudo da movimentação das pessoas. A seguir serão apresentados alguns trabalhos que tratam diferentes aspectos deste problema.

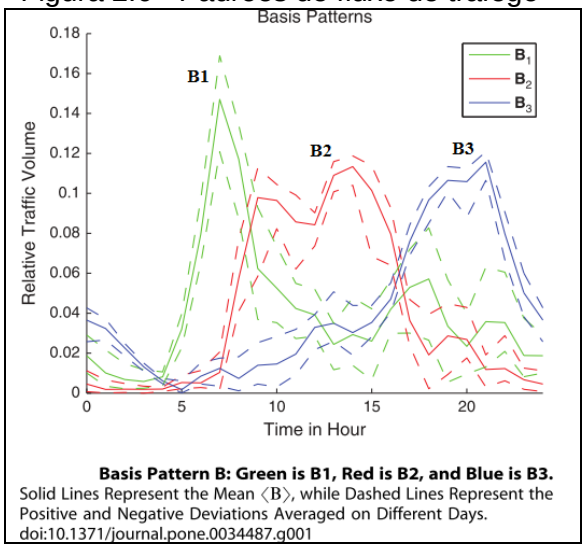
Dentre os principais problemas abordados nos trabalhos que tratam a movimentação de táxis estão: a identificação do padrão de deslocamento dos usuários do sistema em diversos horários do dia e diferentes dias da semana; identificação dos principais pontos de origem e destino e a relação entre essas localidades; recomendação para motoristas e passageiros e o desenvolvimento de ferramentas para medição de eficiência dos motoristas de táxi.

No trabalho apresentado por (PENG *et al.*, 2012) foi feita uma análise da movimentação dos passageiros de táxi em Xangai, na China. Nesse trabalho foram utilizadas somente informações referentes ao local onde o passageiro embarcou e

desembarcou do táxi (i.e., *pick-up* e *drop-off,* respectivamente). Cada par de informações de *pick-up* e *drop-off* foi definido como uma viagem de táxi. Desta forma, os dados do percurso da viagem não foram considerados. Além disso, para facilitar a análise e representação das informações, a área urbana foi dividida em quadrados com 200 metros de lado. Estas regiões são classificadas de acordo com a ocupação predominante (residencial, comercial, lazer, etc.). Esse trabalho identificou que as pessoas utilizam táxi em dias úteis basicamente para três diferentes finalidades: para se deslocarem entre casa e trabalho, entre locais de trabalho ou entre locais que tenham a origem ou o destino diferente de casa ou trabalho, denominado genericamente de outros. Esses três padrões básicos foram utilizados para representar o fluxo de tráfego na cidade.

A análise da movimentação das pessoas pode ser feita através de um gráfico mostrando a variação do volume de tráfego ao longo de um dia útil, de acordo com os três padrões definidos, B1 (casa-trabalho), B2 (trabalho-trabalho) e B3 (outros) (Figura 2.6). Para o padrão B1, o gráfico mostra que o maior volume de tráfego ocorre no período inicial da manhã. Enquanto que para o padrão B2 ocorre um grande volume de tráfego durante todo o período correspondente ao horário normal de trabalho. E finalmente, para o padrão B3 o volume de tráfego cresce no período da noite.

Figura 2.6 - Padrões de fluxo de tráfego



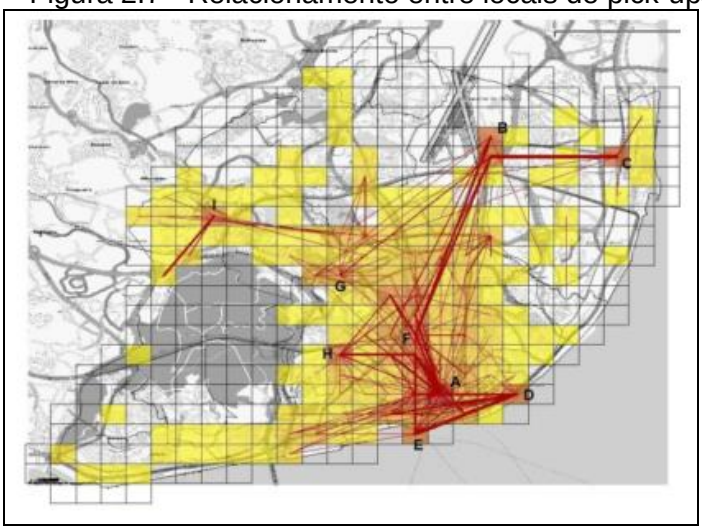
Fonte: Peng (2012).

Análises mostraram que em diferentes dias da semana, mesmo considerando a mesma localização da cidade, existe uma variação no fluxo de tráfego das diferentes categorias (i.e., casa-trabalho, trabalho-trabalho e outros). O motivo desta variação se deve principalmente à incerteza relacionada ao movimento das pessoas. O artigo apresenta ainda uma função de distribuição de probabilidade para o cálculo do desvio relativo do fluxo de tráfego. Essa função de distribuição pode ser usada para prever o fluxo de tráfego na cidade, para planejamento da área urbana (por exemplo, novos empreendimentos residenciais ou empresariais) ou para detectar anomalias no tráfego.

No trabalho de (VELOSO *et al.*, 2011) foram criadas diversas visões com o mapa da cidade para mostrar informações dos locais de *pick-up* e *drop-off* e o relacionamento entre esses locais. Uma distribuição espacial da cidade também foi mostrada com o objetivo de apresentar a distância média percorrida pelos táxis durante o tempo de inatividade. Outras visões mostraram, através de gráficos, a variação do serviço de táxi de acordo com a hora do dia e dia da semana, as viagens de táxi considerando distância percorrida, a duração e a renda, a quantidade de viagens versus o número de táxis trabalhando, a média de tempo gasto e a distância percorrida sem passageiro.

A Figura 2.7 mostra uma visão com o mapa da cidade de Lisboa e linhas conectando os pontos de *drop-off* e o próximo *pick-up*, isto é, a distância linear percorrida pelo motorista à procura de passageiro. A espessura da linha indica um maior relacionamento entre as regiões de *drop-off* e *pick-up*. Um forte relacionamento pode ser observado, por exemplo, entre a localização B-C, D-E, D-A, A-F e F-B. Como o percurso da viagem não foi considerado, mas somente os pontos de origem e destino, a distância considerada foi a distância linear entre esses dois pontos.

Figura 2.7 - Relacionamento entre locais de pick-ups e drop-offs



Fonte: Veloso (2011).

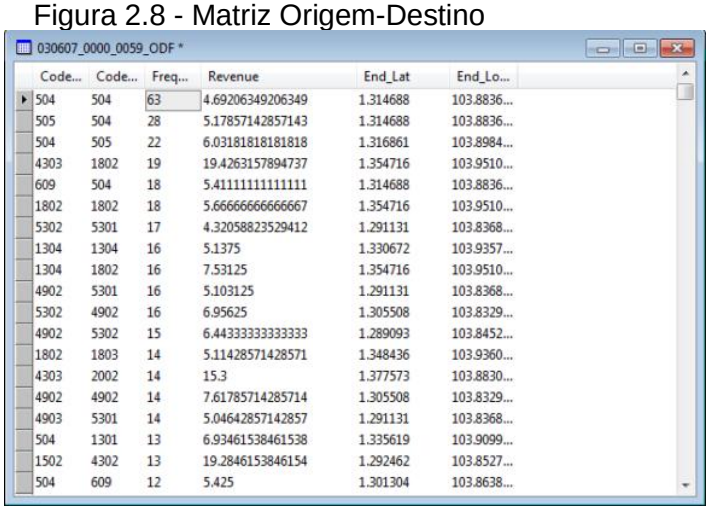
Visando um melhor entendimento dos aspectos relacionados à mobilidade urbana, o trabalho de (VELOSO *et al.*, 2011) analisou a movimentação dos táxis em Lisboa, Portugal. Nesse trabalho foram considerados, também, apenas os dados de origem e destino das corridas de táxi. A análise espaço-temporal e o estudo da previsibilidade das viagens de táxi são as duas principais contribuições desse artigo. Para facilitar a análise das informações, o mapa da cidade foi dividido em quadrados com 500 metros de lado. Neste trabalho foi mostrada a variação espaço-temporal e a concentração dos principais locais de origem e destino dos passageiros de táxis em diferentes períodos do dia, o relacionamento entre essas localidades e os horários de pico de utilização do sistema. Foi analisado o tempo de inatividade dos táxis, isto é, o tempo em que o motorista de táxi fica procurando pelo próximo passageiro, e realizado um estudo de previsibilidade para o próximo local de *pick-up* de acordo com o histórico do fluxo de tráfego no local e horário onde se encontra o táxi. O cálculo da previsão do próximo local de *pick-up*, de acordo com o último *drop-off* realizado, foi feito com a utilização de Pontos de Interesse (POI), além de outras características como dia da semana, hora do dia e condições do tempo.

Em (KAMAROLI *et al.*, 2011) é apresentada uma metodologia para analisar os trajetos dos passageiros de táxi da cidade de Singapura em diversos períodos do dia. A análise foi dividida em dias úteis e em finais de semana. O principal objetivo deste trabalho foi de quantificar, visualizar e examinar o fluxo de táxis considerando

somente informações de origem e destino dos passageiros e valores aferidos em cada viagem de táxi.

Para analisar a correlação entre a demanda de táxis e a área urbana foi utilizada uma matriz com as origens e destinos das viagens de táxi, agrupadas por área urbana. Essa matriz é mostrada na Figura 2.8. O relacionamento entre as origens e os destinos das viagens de táxi, podem revelar as zonas mais rentáveis da cidade do ponto de vista do motorista de táxi. Além disso, a utilização de camadas que indicam o uso da área, como camadas de negócio, educação, diversão, shopping, torna possível a verificação da correlação entre as viagens de táxi e as atividades em curso na cidade.

Figura 2.8 - Matriz Origem-Destino

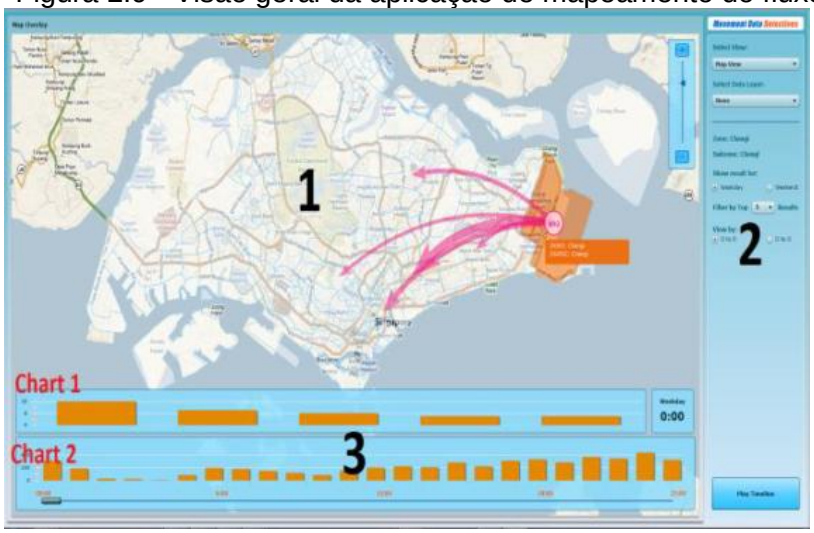


Code...	Code...	Freq...	Revenue	End_Lat	End_Lo...
504	504	63	4.69206349206349	1.314688	103.8836...
505	504	28	5.17857142857143	1.314688	103.8836...
504	505	22	6.03181818181818	1.316861	103.8984...
4303	1802	19	19.4263157894737	1.354716	103.9510...
609	504	18	5.41111111111111	1.314688	103.8836...
1802	1802	18	5.66666666666667	1.354716	103.9510...
5302	5301	17	4.32058823529412	1.291131	103.8368...
1304	1304	16	5.1375	1.330672	103.9357...
1304	1802	16	7.53125	1.354716	103.9510...
4902	5301	16	5.103125	1.291131	103.8368...
5302	4902	16	6.95625	1.305508	103.8329...
4902	5302	15	6.44333333333333	1.289093	103.8452...
1802	1803	14	5.11428571428571	1.348436	103.9360...
4303	2002	14	15.3	1.377573	103.8830...
4902	4902	14	7.61785714285714	1.305508	103.8329...
4903	5301	14	5.04642857142857	1.291131	103.8368...
504	1301	13	6.93461538461538	1.335619	103.9099...
1502	4302	13	19.2846153846154	1.292462	103.8527...
504	609	12	5.425	1.301304	103.8638...

Fonte: Kamaroli (2011).

Uma contribuição interessante no trabalho de (KAMAROLI *et al.*, 2011), é uma aplicação que ajuda a mapear o fluxo do tráfego na cidade. Esta aplicação apresenta uma abordagem que integra geovisualização e informações quantitativas baseada na matriz origem-destino. Uma visão geral da aplicação é mostrada na Figura 2.9. Na parte 1 da figura observa-se a área de geovisualização das informações com os principais fluxos de deslocamento de passageiros ao longo do dia. O painel de controle (parte 2) permite que o usuário selecione as visões ou camadas do mapa que serão utilizadas. Um gráfico mostrando a quantidade de deslocamento dos cinco principais destinos e a movimentação de acordo com a hora do dia pode ser visto na parte 3.

Figura 2.9 - Visão geral da aplicação de mapeamento do fluxo de tráfego

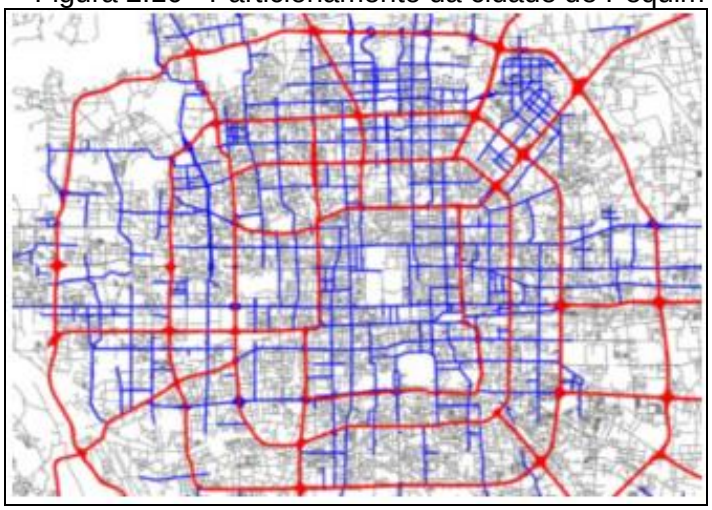


Fonte: Kamaroli (2011).

No trabalho de (ZHENG *et al.*, 2011) o objetivo é detectar falhas no planejamento urbano de Pequim com base em dados obtidos nas análises das trajetórias dos táxis. Os resultados da pesquisa identificaram regiões com problemas de tráfego relevantes e falhas na estrutura de ligação entre essas regiões. Essas informações podem ser utilizadas por gestores urbanos para a construção de uma nova via de ligação entre diferentes pontos da cidade ou para a abertura de uma nova linha de metrô, além de servir, em planos futuros, para lembrar os problemas que não foram considerados em projetos realizados anteriormente.

O método apresentado por (ZHENG *et al.*, 2011) particiona a cidade em regiões disjuntas. Ao invés de realizar uma subdivisão regular do espaço, este trabalho utiliza as principais vias de acesso para promover um particionamento não uniforme. Esse particionamento é mostrado na Figura 2.10, onde as linhas mais grossas representam as principais vias de acesso da cidade e os limites das regiões. As trajetórias diárias dos táxis são projetadas nessas regiões e depois é feita uma análise do relacionamento entre elas. Por fim, são detectados os pares de regiões que apresentam o fluxo de tráfego além da capacidade das vias de ligação.

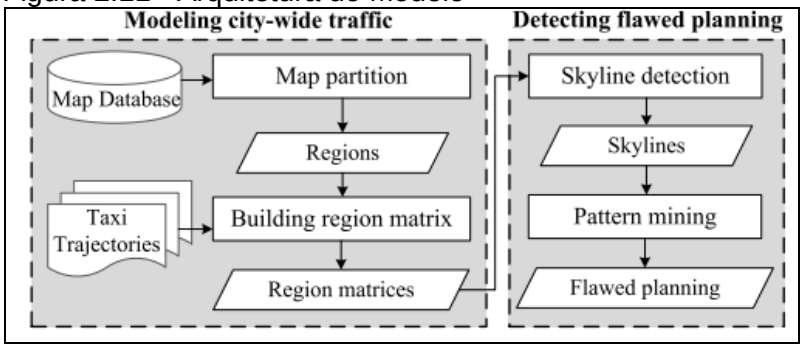
Figura 2.10 - Particionamento da cidade de Pequim



Fonte: Zheng (2011).

O modelo proposto por (ZHENG *et al.*, 2011), mostrado na Figura 2.11, é formado pelo componente de modelagem do tráfego da cidade e pelo componente de detecção de falhas no planejamento.

Figura 2.11 - Arquitetura do Modelo



Fonte: Zheng (2011).

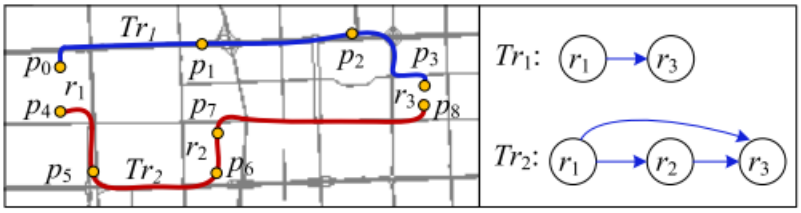
O componente de modelagem do tráfego da cidade particiona o mapa da cidade em regiões, depois constrói uma matriz dessas regiões em diferentes horários e dias da semana. A construção da matriz é feita em três etapas: particionamento temporal, construção das transições e construção da matriz de regiões.

Na etapa do particionamento temporal as trajetórias são separadas em dois grupos: dia útil e dia de repouso (finais de semana e feriados). Em seguida, o dia é segmentado em faixas de horários de acordo com a condição do tráfego da cidade.

A etapa de construção das transições (Figura 2.12) considera apenas as trajetórias com passageiro. Essas trajetórias são projetadas no mapa e depois é feita

a construção das transições entre duas regiões da cidade. Ocorre uma transição da região r_1 em direção à região r_2 se existe uma trajetória, nessa direção, ligando essas duas regiões. Cada par de regiões está associada a três características: 1) volume de tráfego entre duas regiões (i.e., a quantidade de transições $|S|$), 2) velocidade estimada das transições $E(V)$, e 3) relação entre a distância percorrida estimada e a distância euclidiana entre os centroides das duas regiões, θ.

Figura 2.12 - Construção das transições a partir das trajetórias



Fonte: Zheng (2011).

A terceira etapa é a construção da matriz das regiões da cidade. Cada item da matriz é uma tupla do tipo a_{ij} = ($|S|$, $E(V)$, θ) (Figura 2.13). As matrizes são construídas para cada faixa de horário do período a ser analisado. Considere, por exemplo, o estudo do período de um mês com 20 dias úteis e 10 dias não úteis. Considere, ainda, que são estabelecidas 4 faixas de horário para dias úteis e 3 faixas de horário para dias não úteis. Desta forma, será necessário construir 20*4 + 10*3 = 110 matrizes.

Figura 2.13 - Matriz das regiões da cidade

$$M = \begin{array}{c} \\ r_0 \\ r_1 \\ \vdots \\ r_i \\ \vdots \\ r_{n-1} \\ r_n \end{array} \begin{array}{c} \begin{array}{cccccc} r_0 & r_1 & \cdots & r_j & \cdots & r_{n-1} \quad r_n \end{array} \\ \begin{bmatrix} \phi & a_{0,1} & \cdots & & & a_{0,n} \\ a_{1,0} & \phi & \cdots & & & a_{1,n} \\ \cdots & & & & & \\ a_{i,0} & a_{i,1} & \cdots & a_{i,j} & \cdots & a_{i,n} \\ \cdots & & & & & \\ a_{n-1,0} & & \cdots & & \phi & a_{n-1,n} \\ a_{n,0} & & \cdots & & & \phi \end{bmatrix} \end{array}$$

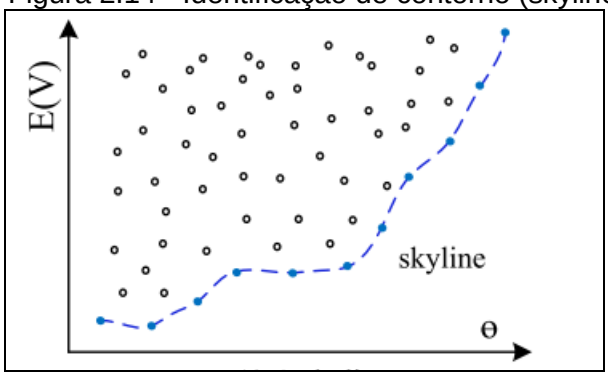
Fonte: Zheng (2011).

O componente de detecção de falhas no planejamento urbano faz a identificação do contorno (*skyline*) de cada matriz e procura por possíveis falhas na conexão entre regiões da cidade utilizando algoritmos de mineração. A análise dos

elementos da tupla (|*S*|, *E(V),* θ) irá determinar as condições do tráfego entre duas regiões e como essas regiões estão conectadas. A propriedade geométrica da conexão entre um par de regiões é determinada por θ. Um par de regiões que possua um valor θ alto informa que as trajetórias realizadas entre essas duas regiões percorreram uma distância maior do que a esperada. Valores de |*S*| e *E(V)* representam características do tráfego. Um valor |*S*| alto e *E(V)* baixo significa um tráfego intenso devido à existência de diversos deslocamentos entre as duas regiões em baixa velocidade. Como o objetivo desse componente é detectar falhas no planejamento urbano, deve-se procurar por pares de regiões com θ alto, *E(V)* baixo e |*S*| alto, indicando que não foi escolhido o caminho mais curto entre as duas regiões, a velocidade de deslocamento é baixa e o volume de tráfego entre as regiões é alto.

Para traçar o *skyline* primeiro são selecionados os pontos da matriz onde o número de transições |*S*| está acima da média, depois são selecionados os pontos de acordo com *E(V)* e θ. A Figura 2.14 mostra um exemplo de *skyline* encontrado de acordo com os pontos de uma matriz construída anteriormente. Pode-se observar que não existe nenhum ponto que tenha simultaneamente *E(V)* menor e θ maior que os pontos que formam o *skyline*.

Figura 2.14 - Identificação de contorno (skyline)



Fonte: Zheng (2011).

No artigo de (GE *et al.*, 2011) foi apresentado um sistema de táxi inteligente capaz de explorar, para uso comercial, os dados gerados pelos táxis da cidade de San Francisco e Nova York. O sistema promete, dentre outras coisas, aumentar a produtividade dos motoristas de táxis com recomendação de rotas, identificar fraudes na condução do veículo e servir de suporte para novas ideias de negócio.

A função de recomendação de rota proposta por (GE *et al.*, 2011) utiliza o conhecimento extraído dos dados históricos dos taxistas que conseguiram um alto índice de ocupação dos seus veículos. Essencialmente a rota recomendada é uma sequência de pontos de *pick-up*. Naturalmente, a melhor rota sugerida será aquela que apresentar a menor distância a ser percorrida até que o próximo passageiro seja encontrado. Como existem muitos pontos de *pick-up* em uma cidade, são necessários algoritmos eficientes para encontrar a melhor rota. Com a análise dos dados históricos dos motoristas, foi observado que existem áreas onde o número de *pick-ups* é maior que em outras áreas. Desta forma, foi proposto um algoritmo de clusterização considerando esses pontos de *pick-up*. Com a análise histórica dos dados de *pick-up* e a partir do entendimento do comportamento dos motoristas, foi desenvolvida também uma função de detecção de fraude na condução do táxi. Para isso foram considerados diversos aspectos, tais como: rota, velocidade, tempo, distância e situação do tráfego.

Em (LIU *et al.*, 2009) é apresentada uma metodologia para analisar o comportamento dos motoristas de táxi em Shenzhen, uma cidade no sul da China. O objetivo é quantificar, visualizar e analisar o comportamento e habilidade dos motoristas de táxi. A habilidade é medida através da renda alcançada, essa habilidade é denominada de "*mobility intelligence*". A renda alcançada está relacionada à rota percorrida pelo táxi de acordo com as condições de tráfego. O motorista é classificado, de acordo com a sua arrecadação, como *top drivers* e *ordinary drivers*. Foi observado que enquanto *ordinary drivers* atuam em locais fixos, os *top drivers* escolhem os locais de acordo com o momento mais oportuno. A Figura 2.15 mostra que a renda diária dos motoristas de táxi segue a distribuição normal, onde a maior parte da arrecadação está entre 700-800 YUAN (*ordinary drivers*) e a outra parte está muito abaixo da média, onde a arrecadação está entre 200-300 YUAN ou muito acima da média com a arrecadação superior a 1.250 YUAN (*top drivers*).

Figura 2.15 - Histograma da renda diária dos motoristas (em Yuan)

Fonte: Liu (2009).

Com o intuito de explicar o motivo dos *top drivers* conseguirem uma arrecadação maior que os *ordinary drivers*, foram feitas diversas análises envolvendo os dois tipos de motoristas. Para representar os *top drivers*, foi selecionado o motorista que conseguiu a maior arrecadação no período analisado e para representar os *ordinary drivers f*oi escolhido de forma aleatória um motorista pertencente a essa classificação. Entre os 3.000 motoristas analisados no período de 25 dias, considerando o valor arrecadado, o *top driver* analisado conseguiu a maior arrecadação em 44% das vezes, esteve entre os três primeiros em 76% das vezes e entre os dez primeiros em 88% das vezes (Tabela 2.1). O resultado obtido pelo *top driver* mostra que ele utiliza uma estratégia para encontrar passageiro, e que os motoristas que conseguirem seguir a mesma estratégia poderão ter uma melhor arrecadação. Portanto, teoricamente, a estratégia do *top driver* poderia definir o modelo a ser seguido pelos outros motoristas.

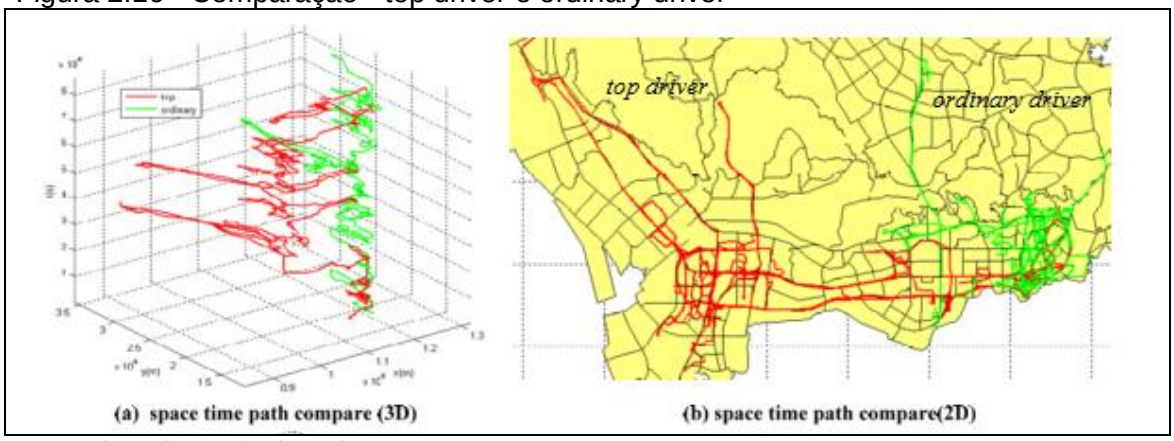
Tabela 1 - Classificação do melhor motorista dentre os 3.000 analisados

Rank	Number	Frequency	Accumulate frequency
1	11	0.44	0.44
2	5	0.2	0.64
3	3	0.12	0.76
8	1	0.04	0.8
9	1	0.04	0.84
10	1	0.04	0.88
>10	3	0.12	1
Sum	25	1	

Fonte: Liu (2009).

A Figura 2.16 permite analisar a configuração espaço-temporal das trajetórias desenvolvidas pelo *top driver* (linha vermelha) e do *ordinary driver* (linha verde). Analisando somente a visão bidimensional (Figura 2.16b) percebe-se que estes motoristas atuam na maior parte do tempo em regiões distintas da cidade. Analisando a configuração espaço-temporal (Figura 2.16a), entretanto, percebe-se que mesmo quando os motoristas trabalham em regiões próximas, estas viagens ocorrem em períodos distintos, sugerindo que existe uma habilidade envolvida.

Figura 2.16 - Comparação - top driver e ordinary driver

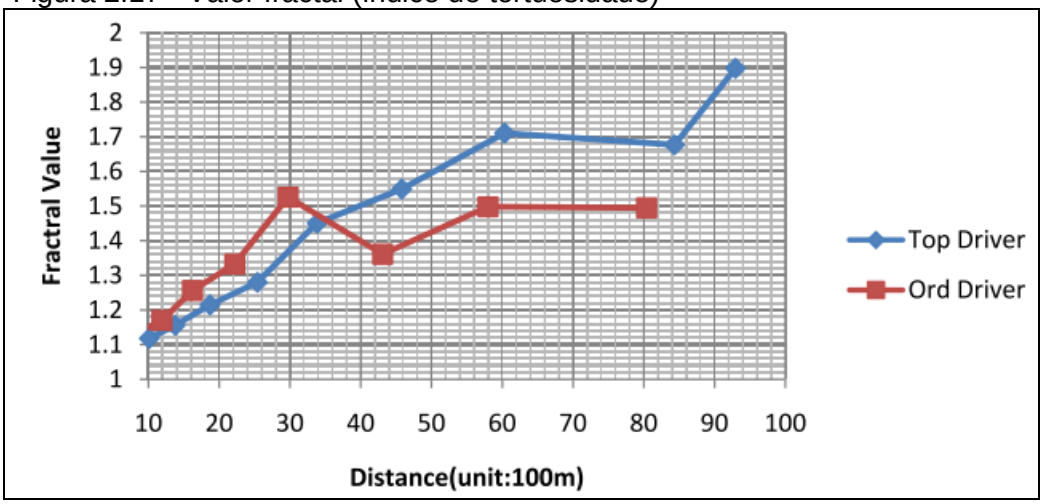


Fonte: Liu, Adaptação (2009).

A ideia de se estudar o comportamento do *top driver* não é a de sugerir que seguindo o caminho realizado por este os demais motoristas irão se tornar *top drivers*. Este comportamento levaria ao aumento de oferta de táxis em determinadas regiões, comprometendo os lucros. O principal é identificar as estratégias utilizadas pelo *top driver* e tentar utilizar a mesma estratégia em diferentes regiões. Para determinar as características de como os motoristas trabalham foi utilizado o

algoritmo de clusterização K-means. Assim, foram obtidos diversos centros de operação de trabalho em diferentes horários para *top drivers* e para *ordinary drivers*. (LIU *et al.*, 2009) mostrou que existe uma clara rotina espaço-temporal para os *top drivers*, onde são consideradas as condições de tráfego e a demanda por táxi. Foi feita uma análise para verificar se os taxistas estavam utilizando o caminho mais curto para levar os passageiros aos seus destinos. Com essa abordagem foi mostrado que em distâncias de até 3,5km os trajetos utilizados pelos *ordinary drivers* eram mais tortuosos que aqueles utilizados pelos *top drivers*, enquanto que para distâncias superiores a 3,5km acontecia o inverso (Figura 2.17). De forma a entender este comportamento, faz-se necessário entender o sistema de tarifação. A tarifa de táxi para distâncias de até 3km é de 12.5 YUAN e em distâncias superiores a tarifa é de 2.4 YUAN/km. Assim, uma boa estratégia é pegar o menor caminho até os 3 primeiros km, e depois tomar o caminho mais longo de maneira a aumentar o valor da viagem. Esta é a estratégia utilizada pelos *top drivers*.

Figura 2.17 - Valor fractal (índice de tortuosidade)

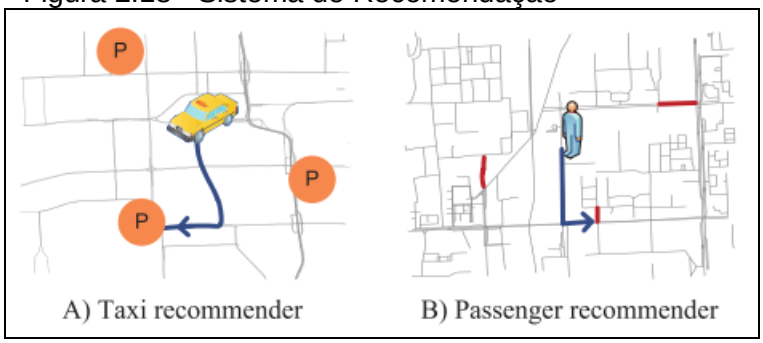


Fonte: Liu (2009).

Em (YUAN *et al.*, 2011) foi apresentado um sistema de recomendação com sugestões para motorista de táxis e passageiros. Para os motoristas de táxis o sistema de recomendação sugere locais e rotas onde a probabilidade de encontrar passageiros é maior, maximizando, desta forma, o lucro e a eficiência do sistema (Figura 2.18A). Para os passageiros são recomendados locais próximos onde eles possam chegar caminhando e encontrar um táxi disponível (Figura 2.18B). Essas

sugestões são baseadas nos padrões de movimentação dos passageiros de táxi (i.e., onde e quando eles costumam entrar e sair dos táxis e na estratégia utilizada pelos motoristas de táxi para conseguir passageiros para transportar). A base de conhecimento dos sistemas de recomendação foi adquirida através da análise de dados históricos das trajetórias dos táxis equipados com GPS.

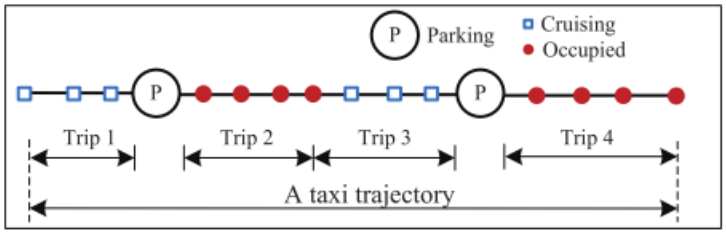
Figura 2.18 - Sistema de Recomendação



Fonte: Yuan (2011).

No trabalho de (YUAN *et al.*, 2011), uma trajetória de táxi (*taxi trajectory*) é definida como uma sequência de pontos de GPS, onde cada ponto é formado pelas seguintes informações: tempo, latitude, longitude, segmento da rodovia e estado de ocupação (ocupado/desocupado). Uma viagem de táxi (*taxi trip*) é uma subtrajetória com o estado *cruising* ou *occupied*. Nesse trabalho foram considerados três estados para o táxi, *occupied* (*O*), *cruising* (*C*) e *parking* (*P*). O táxi foi considerado desocupado tanto no estado *cruising* quanto no estado *parking*. A identificação destes estados necessita de algum tipo de pós-processamento. Veja na Figura 2.19 um exemplo de trajetória de táxi formada por quatro viagens.

Figura 2.19 - Trajetória e Viagem de Táxi

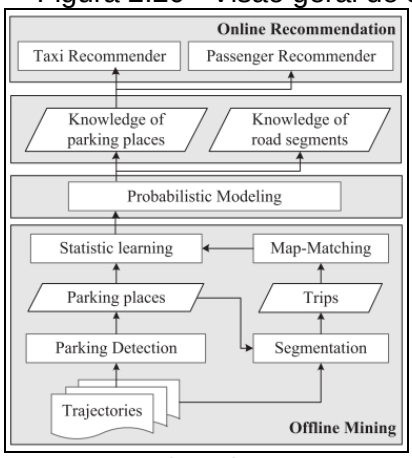


Fonte: Yuan (2011).

O sistema de recomendação é formado por uma camada *offline*, uma camada com o modelo probabilístico e a camada *online* de recomendação. Na camada *offline* as trajetórias são segmentadas, as rodovias utilizadas são identificadas e são

detectados os pontos de táxi. O modelo probabilístico irá integrar o padrão de movimentação dos passageiros com o padrão de movimentação dos taxistas. A Figura 2.20 mostra uma visão geral do sistema.

Figura 2.20 - Visão geral do sistema de recomendação

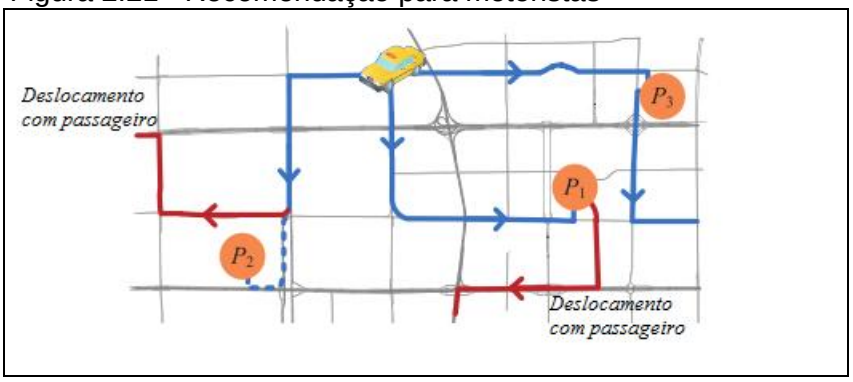


Fonte: Yuan (2011).

O módulo de recomendação para taxistas tem como objetivo informar os melhores pontos de táxi e a rota para chegar a esses locais. Um bom ponto de táxi é aquele onde existe uma alta probabilidade de encontrar um passageiro, tanto no trajeto quanto no local do ponto. Além disso, estando no ponto de táxi, o tempo de espera para pegar o próximo passageiro deverá ser o menor possível. Além disso, o modelo de recomendação para taxistas deverá informar diversas rotas, cada uma delas com o tempo estimado e a probabilidade de encontrar um passageiro até chegar ao ponto de táxi mais próximo, além da distância e tempo estimado do deslocamento com esse passageiro.

A Figura 2.21 mostra um motorista de táxi e as alternativas dos pontos de táxi (P_1, P_2 e P_3) mais próximos a ele. Caso ele escolha se dirigir ao ponto de táxi P_1, existe uma determinada probabilidade para que ele encontre um passageiro ao chegar ao ponto de táxi; caso ele escolha se dirigir ao ponto P_2, existe uma determinada probabilidade para que ele encontre um passageiro mesmo antes de chegar ao ponto de táxi e caso ele escolha o ponto P_3, não existe nenhuma probabilidade para que ele encontre um passageiro, nem mesmo depois de esperar por algum tempo.

Figura 2.21 - Recomendação para motoristas



Fonte: Yuan Adaptação (2011).

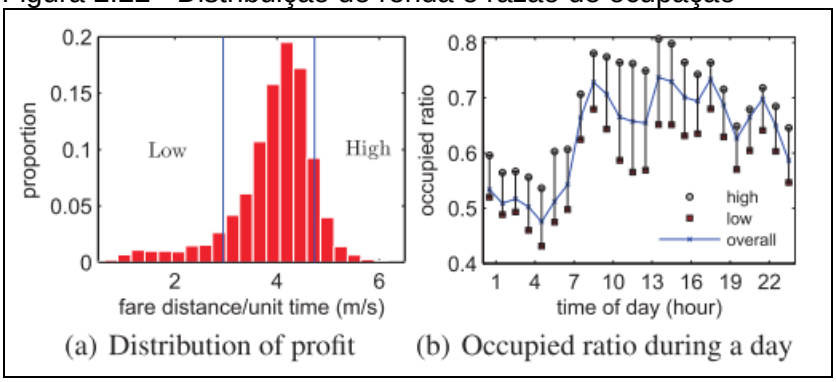
O objetivo do módulo de recomendação para passageiros é informar os táxis disponíveis em locais próximos. Caso o passageiro esteja próximo a um ponto de táxi é sugerido que ele se dirija a este local. Caso contrário, é sugerido que o passageiro de dirija para a via mais próxima onde ele possa encontrar um táxi.

Um tipo de análise proposta por (YUAN *et al.*, 2011) trata da classificação dos motoristas de táxi baseada no resultado financeiro obtido e nas distâncias percorridas. Os autores acreditam que a determinação do lucro de um motorista de táxi depende principalmente de dois fatores. Primeiro, o motorista deve saber os lugares onde pegar rapidamente passageiros em uma determinada hora do dia. Segundo, a extensão das corridas características de cada localidade deve ser conhecida, quanto maior a extensão maior o valor da corrida. Os motoristas sabem, por exemplo, que aeroportos, rodoviárias, *shoppings* e hotéis geralmente têm uma grande demanda por táxi, e têm conhecimento também dos horários de chegada de aviões, trens ou quando termina uma sessão de cinema ou teatro. Então, ao invés de gastar combustível se deslocando pela cidade, os motoristas preferem aguardar pelo melhor momento de sair em busca de passageiro em um ponto de táxi.

Como pode ser visto na Figura 2.22(a), a renda do motorista pode ser medida através da tarifa cobrada, calculada de acordo com a distância percorrida com passageiro em um determinado intervalo de tempo. Com base nessa informação, os motoristas foram classificados em grupos. Os 10% que conseguiram a maior renda formaram o grupo dos *high-profit drivers*, os 10% com a menor renda formaram o grupo dos *low-profit drivers* e os demais formaram o grupo dos *medium drivers*. A Figura 22(b) mostra o índice de ocupação (razão entre a distância percorrida com o

táxi ocupado e a distância total percorrida) ao longo do dia. Para os *high-profit drivers* e os *low-profit drivers* a diferença entre esses dois grupos de motoristas é mais significativa no período entre 10h e 15h (i.e., fora do horário de pico). No horário de pico os táxis se tornam mais escassos, fazendo com que a variação entre o índice de ocupação dos dois grupos de motoristas seja menor.

Figura 2.22 - Distribuição de renda e razão de ocupação



Fonte: Yuan (2011).

Os trabalhos apresentados nesta seção ilustram apenas algumas possibilidades interessantes da utilização dos dados das trajetórias de táxis. As possibilidades são muitas e revelam um interesse crescente na área. Nos trabalhos de (PENG *et al.*, 2012) e (VELOSO *et al.*, 2011) são utilizadas apenas as informações com as localizações de pontos de embarque e desembarque de passageiros. No trabalho de (KAMAROLI *et al.*, 2011) além dessas informações, o valor da corrida também foi utilizado. Em (ZHENG *et al.*, 2011), (GE *et al.*, 2011), (LIU *et al.*, 2009) e (YUAN *et al.*, 2011) são utilizadas as informações da trajetória do táxi e a informação de ocupação do veículo. Alem dessas informações, (LIU *et al.*, 2009) considera o valor da corrida e (YUAN *et al.*, 2011) considera o valor da corrida e as paradas em pontos de táxi.

Considerando todos os trabalhos revisados não identificamos nenhum modelo de dados para representar a movimentação de táxis em ambientes urbanos capaz de suportar uma ampla gama de análises e descoberta de conhecimento. Desta forma, utilizamos o modelo de Spaccapietra como inspiração para o desenvolvimento de um modelo voltado para a representação do movimento dos táxis. O próximo capítulo apresenta detalhes da representação desse modelo e o processo de transformação de dados brutos em entidades do modelo.

3 MODELO TX-TRAJECTORY

Existe na atualidade uma grande demanda por aplicações voltadas para o estudo da mobilidade urbana. O Sistema Inteligente de Transporte (ITS) é uma área de pesquisa voltada para o monitoramento e a análise das condições das estradas e do fluxo dos veículos e usuários do sistema de transporte. A frota de táxi, apesar de não ser considerada como um componente do sistema público de transporte é uma importante modalidade de ITS. Diferente do ônibus, trem e metrô, os táxis não seguem uma rota pré-estabelecida. O táxi busca o passageiro no local onde ele se encontra e leva o passageiro exatamente para o local desejado. Dessa forma, o sistema de táxi tem a capacidade de revelar as verdadeiras necessidades de movimentação do seu usuário, além de mostrar o trânsito da cidade de forma mais abrangente, servindo como um importante mecanismo no gerenciamento e análise do sistema como um todo.

Este capítulo apresenta o modelo *TX-Trajectory* desenvolvido para representar as trajetórias dos táxis. As entidades que compõem este modelo incorporam informações semânticas sobre o deslocamento dos veículos, além de características relevantes do ambiente em que estes veículos trafegam.

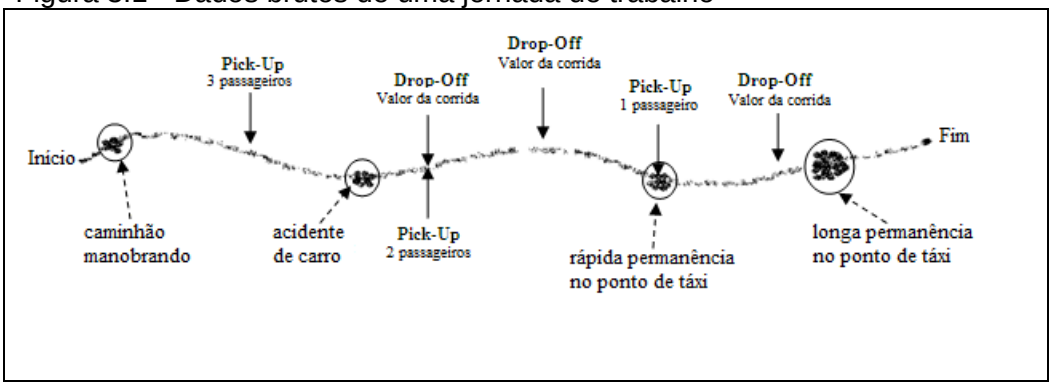
3.1 DADOS BRUTOS DE TÁXI

Para a análise da movimentação de qualquer objeto móvel, o primeiro passo envolve o processo de aquisição das informações brutas do deslocamento. Para ilustrar um processo típico de aquisição de dados sobre a movimentação dos táxis, considere, por exemplo, um cenário onde todos os táxis de uma determinada frota estão equipados com um dispositivo móvel com algum mecanismo de localização embutido e uma aplicação capaz de registrar o percurso das diversas viagens realizadas ao longo do dia e alguns eventos relevantes. O motorista deverá inicializar esta aplicação no início da sua jornada de trabalho e só desligá-la ao final da mesma. Uma vez iniciada, a aplicação começa a coletar e enviar dados sobre a localização do veículo e seu status (i.e., ocupado ou livre). A aplicação possui dois botões: *Pick-up* e *Drop-off*. Toda vez que o motorista pegar um passageiro deverá acionar o botão *Pick-up* e informar o número de passageiros que embarcaram no veículo. Ao final da viagem o botão *Drop-off* deve ser acionado indicando que o táxi

está sem passageiro e disponível para uma nova viagem. Neste instante o motorista deverá informar também o valor cobrado pelo deslocamento. As informações coletadas pela aplicação móvel deverão, idealmente, ser transmitidas em tempo real para uma central de controle. Caso esta funcionalidade não esteja disponível, as informações poderão ser transmitidas ao final da jornada de trabalho do motorista.

O cenário descrito acima já é uma realidade na maioria das empresas e cooperativas de táxi, que usam esse tipo de tecnologia principalmente para saber a localização dos veículos e enviar o táxi disponível mais próximo ao cliente que está requisitando o serviço. A Figura 3.1 mostra os dados brutos gerados pela curta jornada de trabalho de um motorista hipotético. A nuvem de pontos representa dados brutos de localização capturados pelo GPS. As setas contínuas indicam as ações registradas pelo motorista na aplicação móvel embarcada no táxi (*Pick-up* e *Drop-off*). As setas tracejadas indicam alguns eventos externos vivenciados pelo motorista. Estes eventos não foram relatados pelo motorista do veículo, desta forma, eles não fazem parte da base de dados brutos do táxi.

Figura 3.1 - Dados brutos de uma jornada de trabalho



Dados brutos da trajetória do táxi e eventos mostrando pontos de *Pick-up* e *Drop-off* são de pouca utilidade para a maioria das aplicações que têm como objetivo a análise do movimento deste meio de transporte. Do ponto de vista da aplicação, o conhecimento da trajetória, geralmente inclui diversos dados semânticos que complementam os dados brutos (SPACCAPIETRA *et al.*, 2008). Desta forma, no contexto das aplicações voltadas para o estudo da trajetória dos táxis, um modelo conceitual de dados é essencial para representar aspectos mais

relevantes do movimento dos veículos, com a utilização de entidades mais abstratas e semanticamente significativas.

3.2 DEFINIÇÕES BÁSICAS E ENTIDADES DO MODELO

Em 2008 (SPACCAPIETRA *et al.*, 2008) propôs o modelo *Stops* e *Moves,* o primeiro modelo que trata a trajetória de objetos móveis como um conceito espaço-temporal. Nesse modelo uma trajetória é definida como a evolução espaço-temporal do objeto móvel para atingir um determinado objetivo. Inspiradas nesse modelo e baseadas em informações peculiares do domínio da aplicação, serão apresentadas a seguir algumas definições relevantes para o *TX-Trajectory Model*.

3.2.1 Definições Básicas

Definição 1 *Taxi Trajectory (TT)* modela a evolução da posição de um táxi ao longo da vida útil do veículo e é representada por uma lista ordenada de pontos $(p_1, p_2, p_3, ..., p_n)$, onde $p_i = (v\_id_i, d\_id_i, x_i, y_i, t_i, s_i)$, v_id_i é a identificação do veículo$_i$, d_id_i é a identificação do motorista$_i$, x_i e y_i é a localização do objeto no instante t_i, s_i indica o estado de ocupação do veículo e t_i representa o instante que foi realizada a medição, $t_1 < t_2 < t_3 < ... < t_n$.

Definição 2 *Taxi Driver Sub-Trajectory Journey (TDSTJ)* compreende a evolução da posição de um táxi ao longo da jornada de trabalho do seu motorista e é representada por uma lista ordenada de pontos $(p_k, p_{k+1}, p_{k+2}, ..., p_{k+m})$, onde $p_i \subset$ *TT*, $k \geq 1$, $k+m \leq n$ e v_id e d_id são constantes, t_k e t_{k+m} correspondem ao início e ao final do turno de trabalho do motorista (d_id) respectivamente.

Definição 3 *Taxi Driver Sub-Trajectory Journey Full (TDSTJF)* corresponde a um segmento da *Taxi Driver Sub-Trajectory Journey* e compreende o deslocamento do táxi quando existe passageiro ocupando o veículo, sendo representada por uma lista ordenada de pontos $(p_w, p_{w+1}, p_{w+2}, ..., p_{w+n})$, onde $p_i \subset$ *TDSTJ*, $w \geq k$, $w+n \leq k+m$ e s é constante e representa o estado do veículo com passageiro.

Definição 4 *Taxi Driver Sub-Trajectory Journey Empty* corresponde a um segmento da *Taxi Driver Sub-Trajectory Journey* e compreende o deslocamento do táxi quando

não existe passageiro ocupando o veículo, esta entidade é representada por uma lista ordenada de pontos $(p_w, p_{w+1}, p_{w+2}, ..., p_{w+n})$, onde $p_i \subset TDSTJ$, $w \geq k$, $w+n \leq k+m$ e s é constante e representa o estado do veículo sem passageiro.

Definição 5 *Pick-Up Point* é o ponto inicial p_w da *TDSTJF* $(p_w, p_{w+1}, p_{w+2}, ..., p_{w+n})$. *Pick-Up Point* indica o instante e a localização do início de uma *Taxi Driver Sub-Trajectory Journey Full*, isto é, representa o horário e o ponto de origem de uma viagem de táxi com passageiro.

Definição 6 *Drop-Off Point* é o ponto final p_{w+n} da *TDSTJF* $(p_w, p_{w+1}, p_{w+2}, ..., p_{w+n})$. *Drop-Off Point* indica o instante e a localização do final de uma *Taxi Driver Sub-Trajectory Journey Full,* isto é, representa o horário e o ponto de destino de uma viagem de táxi com passageiro.

Definição 7 *Taxi Stand* representa a localização geográfica de um ponto de táxi.

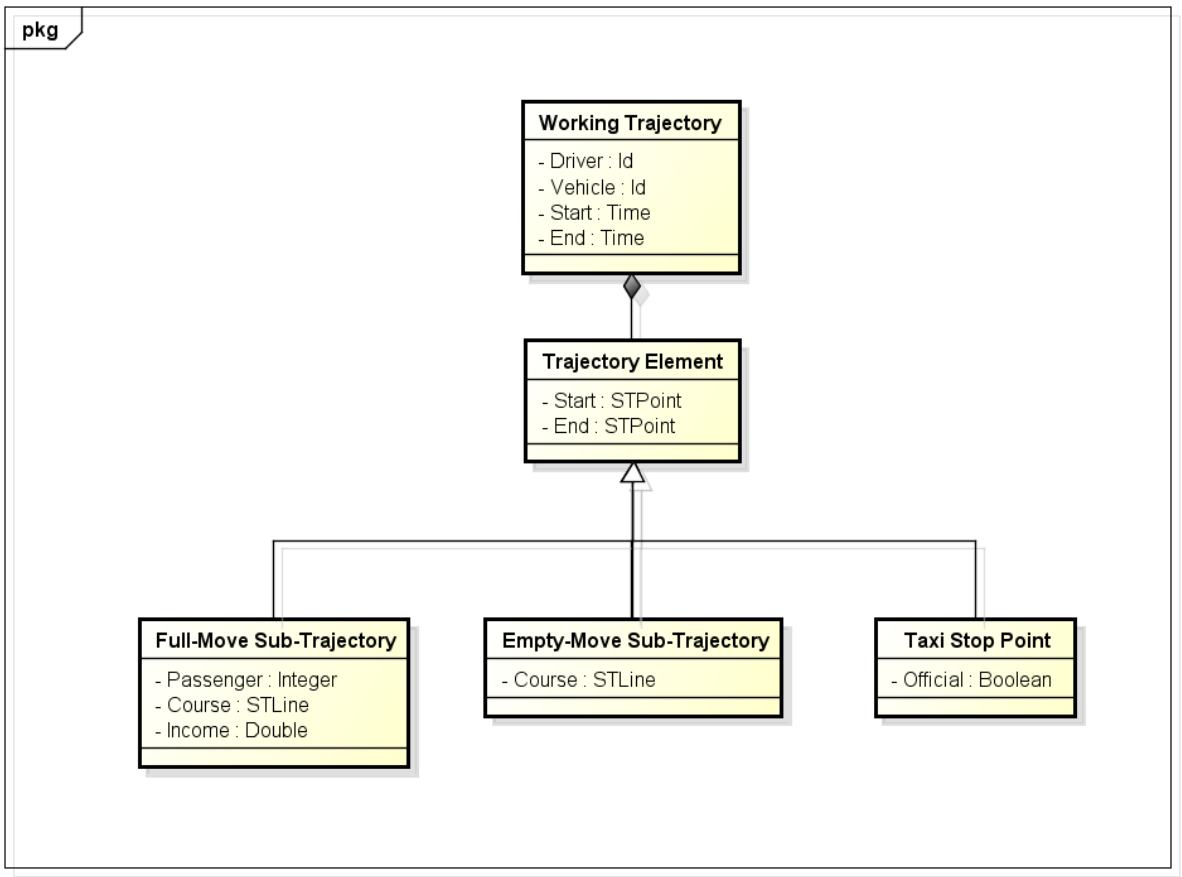
Essas definições são fundamentais para entender o modelo conceitual proposto para representar o movimento dos táxis. As entidades do modelo são apresentadas e discutidas em detalhes na próxima seção.

3.2.2 Modelo Conceitual

As informações brutas da movimentação dos táxis são de pouca utilidade para os algoritmos de mineração de dados e descoberta de conhecimento na maneira como são armazenadas. Objetivando prover uma infraestrutura de dados com entidades semanticamente relevantes para o domínio da aplicação, propomos o modelo *TX-Trajectory.* O referido modelo tem como base a entidade *Working Trajectory* (*WT*). Esta entidade representa uma *Taxi Driver Sub-Trajectory Journey* (Definição 2). A entidade *WT* possui atributos identificando o motorista (*Driver*), o veículo utilizado (*Vehicle*) e dois atributos do tipo *Time* correspondendo ao instante inicial (*Start*) e final (*End*) da jornada de trabalho do motorista de táxi. A combinação veículo-motorista da *Working Trajectory* define o objeto móvel. Esta combinação se faz necessária para identificar situações rotineiras em frotas de táxis, onde diversos motoristas dirigem o mesmo táxi ou em situações onde o motorista trabalha para mais de uma empresa de táxi.

Além dos atributos atômicos mencionados anteriormente (i.e., *Driver, Vehicle, Start, End*), uma *Working Trajectory* possui ainda uma composição de entidades do tipo *Trajectory Element*. Essa entidade, por sua vez, é especializada pelas entidades *Full-Move Sub-Trajectory (FMST)*, *Empty-Move Sub-Trajectory (EMST)* e *Taxi Stop Point (TSP)*. A Figura 3.2 apresenta o modelo *TX-Trajectory* utilizando a linguagem UML *(Unified Modeling Language).*

Figura 3.2 - TX-Trajectory – Modelo Conceitual



O conceito de *Working Trajectory* apresentado no modelo *TX-Trajectory* é baseado no conceito de trajetória definido no modelo de *Stops* e *Moves* (SPACCAPIETRA *et al.*, 2008). Nesse modelo, a trajetória é considerada como uma sequência alternada de *Stops* e *Moves* que ocorrem entre os instantes *Begin* e *End* da trajetória. Diferente da conceituação de Spaccapietra, entretanto, uma *Working Trajectory* é uma combinação de *Full-Move Sub-Trajectory*, *Empty-Move Sub-Trajectory* e *Taxi Stop Point.* As entidades *Full-Move Sub-Trajectory* e *Empty-Move Sub-Trajectory* correspondem aos *Moves* e a entidade *Taxi Stop Point* corresponde ao *Stop*.

A entidade *Full-Move Sub-Trajectory (FMST)* corresponde a um segmento da *Working Trajectory* e representa o deslocamento do táxi quando existe passageiro ocupando o veículo. Uma *FMST* está diretamente relacionada ao nosso conceito de *Taxi Driver Sub-Trajectory Journey Full (*Definição 3*).* Essa entidade possui cinco atributos: um atributo inteiro para indicar o número de passageiros que embarcaram no táxi (*Passenger*); dois atributos do tipo *STPoint* indicando o início (*Start*) e o final (*End*) de uma viagem; um atributo do tipo *STLine* para representação do trajeto da viagem (*Course*) e um atributo double com o valor da viagem (*Income*).

O tipo *STPoint* representa um ponto na dimensão espaço-tempo. Este tipo representa a posição de um objeto móvel e possui um atributo para a localização espacial do objeto e um atributo para associar o instante em que este objeto ocupava esta posição. O tipo *STLine* representa uma coleção arbitrária e não vazia de *STPoints*. É importante salientar que os atributos *Start* e *End* (*STPoints*) herdados da entidade *Trajectory Element* correspondem ao primeiro e último ponto do atributo *Course* (*STLine*). Esta duplicidade de informação visa acelerar as operações sobre os dados que não necessitam de detalhes do deslocamento, mas somente das informações do início e do fim de cada componente da trajetória.

O início da *FMST* corresponde ao ponto em que o passageiro embarcou no veículo. Esse ponto é conhecido como *Pick-Up Point* ou a origem de uma viagem de táxi com passageiro (Definição 5). Já o final da *FMST* corresponde ao ponto em que o passageiro desembarcou do veículo e é conhecido como *Drop-Off Point* ou o destino do passageiro (Definição 6).

Não existe nenhuma dependência espacial entre os atributos *Start* e *End* de uma *Full-Move Sub-Trajectory*, isto é, eles podem representar qualquer ponto no espaço, podendo inclusive ser o mesmo ponto, no caso de uma viagem onde o passageiro retorna ao ponto de embarque. Na dimensão temporal, entretanto, o instante final da trajetória sucede o instante inicial.

A *entidade Empty-Move Sub-Trajectory (EMST)* corresponde a um segmento da *Working Trajectory* e representa o deslocamento do táxi quando não existe passageiro ocupando o veículo. Uma *EMST* corresponde ao nosso conceito de *Taxi Driver Sub-Trajectory Journey Empty (*Definição 4*).* A entidade *EMST* é similar à entidade *Full-Move Sub-Trajectory*, diferindo somente pela ausência dos atributos *Passenger* e *Income*.

O ponto inicial de uma *Empty-Move SubTrajectory* pode ser espacialmente idêntico ao ponto final de uma *Full-Move SubTrajectory* ou de um ponto indicando uma parada em ponto de táxi (*Taxi Stop Point*). De forma similar, o ponto final de uma *Empty-Move SubTrajectory* pode ser espacialmente idêntico ao ponto inicial de uma *Full-Move SubTrajectory* ou de um ponto indicando uma parada em ponto de táxi. No domínio temporal, entretanto, o ponto inicial de uma *Empty-Move SubTrajectory* deve diferir em pelo menos um *chronon* o ponto final de uma *Full-Move SubTrajectory* ou de um ponto de táxi. De forma similar, o ponto final de uma *EMST* deve diferir em pelo menos um *chronon* o ponto inicial de uma *FMST* ou de um ponto de táxi. Os instantes associados ao *Start* e ao *End* da *EMST* podem, ainda, ser iguais, respectivamente, aos instantes *Start* e *End* da *Working Trajectory.*

Um *Taxi Stop Point (TSP)* representa a localização geográfica de um ponto onde o veículo permaneceu parado por um determinado intervalo de tempo à espera de passageiro. Essa entidade possui três atributos: dois atributos do tipo *STPoint* indicando o início da parada (*Start*) e o final da parada (*End*); e um atributo do tipo booleano que indica se o ponto de táxi é oficial ou não. Um ponto de táxi oficial é definido por uma autoridade pública e corresponde à definição de *Taxi Stand* (Definição 7).

A entidade *TSP* não possui uma trajetória associada. A informação espacial armazenada nos atributos *Start* e *End* deve ser a mesma da localização geográfica de um ponto de táxi conhecido. No domínio espacial, a distância entre os pontos de início e fim dá uma ideia aproximada do comprimento da fila. No domínio temporal, os momentos iniciais e finais devem representar instantes distintos, com a diferença entre o instante final e inicial indicando a duração da espera na fila até a chegada de um passageiro. Acreditamos que esta abordagem seja suficiente para atender a maioria das aplicações voltadas para o estudo da movimentação dos táxis.

Para identificar um *Stop* foi adotado o conceito definido por Spaccapietra (i.e., um local onde o objeto tenha permanecido parado por um determinado intervalo de tempo) (SPACCAPIETRA *et al.*, 2008). Portanto, *Pick-Up Point* e *Drop-Off Point* não representam um *Stop* propriamente dito. Estas entidades, com duração de um único *chronon*, são equivalentes ao início e ao fim de uma sub-trajetória do modelo *Stops* e *Moves*. Seguindo ainda a definição de Spaccapietra, um *Stop* é uma situação relevante para a aplicação. No modelo *TX-Trajectory*, existe o interesse em *Stops* que indiquem a parada do motorista em um ponto de táxi para aguardar por

passageiro. O *Stop* decorrente de qualquer outro evento, diferente de uma parada em um ponto de táxi é desconsiderado.

Além das restrições espaço-temporais já mencionadas, a composição das entidades do modelo possui como restrição adicional o fato de não existirem duas *Empty-Move SubTrajectory* consecutivas, elas têm que ser intercaladas por uma *Full-Move SubTrajectory* ou um *Taxi Stop Point*.

De forma a ilustrar o mapeamento das informações coletadas em campo pelas aplicações embarcadas nos veículos, iremos discutir o pequeno exemplo introduzido no início deste capítulo (Figura 3.1). O mapeamento das informações brutas para entidades do modelo não é uma operação trivial. A próxima seção detalha aspectos relevantes nesse mapeamento.

3.2.3 Mapeamento dos Dados Brutos para Entidades do Modelo

Antes do processamento das informações brutas, geralmente é necessária uma limpeza dos dados, onde informações redundantes ou fora do padrão desejado são eliminadas (BOGORNY *et al.*, 2011). Depois desta limpeza, os dados tratados são mapeados para um modelo de dados mais adequado para o domínio da aplicação. Durante o mapeamento dos dados brutos é possível adicionar informações semânticas baseadas em informações do ambiente ou da dinâmica do movimento, como pontos onde o objeto permaneceu parado ou lugares que ele tenha visitado (YAN *et al.*, 2011) (YAN, 2009) (ALVARES *et al.*, 2007).

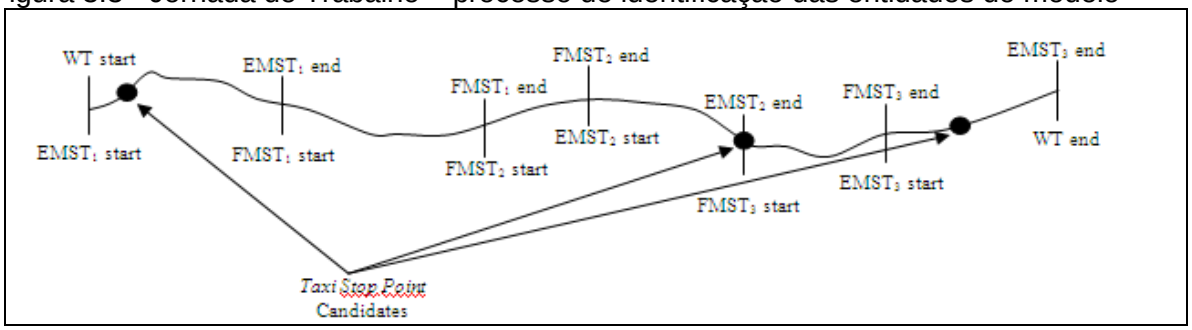
O processo de mapeamento de dados brutos para entidades do *TX-Trajectory Model* é similar ao processo de identificação de *Stops* e *Moves* já tratados em alguns trabalhos (BOGORNY *et al.*, 2011) (PALMA *et al.*, 2008). Na área de análise de trajetórias, o esquema conceitual é construído a partir de dados já existentes, utilizando a técnica de engenharia reversa de banco de dados. Essa técnica foi utilizada por diversos pesquisadores no intuito de desenvolver métodos de instanciação do modelo de S*tops e Moves* considerando diferentes características das trajetórias para encontrar as paradas e os movimentos. Neste contexto, os métodos IB-SMoT (ALVARES *et al.*, 2007), CB-SMoT (PALMA *et al.*, 2008) e DB-SMoT (ROCHA *et al.*, 2010) são os mais conhecidos.

O método IB-SMoT (*intersection-based stops and moves of trajectories*), gera *stops* com base na interseção das trajetórias com objetos geográficos de interesse

para o domínio da aplicação. O segundo algoritmo desenvolvido para a identificação de *stops e moves* foi o CB-SMoT (*clustering-based stops and moves of trajectories*), gera *stops* com base na variação de velocidade da trajetória. O DB-SMoT (*direction-based stops and moves of trajectories*), gera *stops* com base na variação da direção da trajetória.

Tendo como exemplo os dados brutos gerados a partir da jornada de trabalho hipotética ilustrada na seção anterior (Figura 3.1), descreve-se a seguir o processo de mapeamento dos dados brutos para as entidades do modelo *TX-Trajectory*. O primeiro passo no processo é a criação de uma instância da *Working Trajectory*. Nesse momento apenas os atributos atômicos desta entidade são identificados. Fazem parte desses atributos as informações de identificação do motorista e do veículo e as informações de início e fim da trajetória (i.e., *Driver, Vehicle, Start, End*). Depois disso e de acordo com as informações de *Pick-up* e *Drop-off* registradas pelo motorista, é feita a primeira identificação dos trechos da trajetória que representam os *Stops* e *Moves*. Os *Moves* indentificados nessa fase darão origem às *Empty-Move Subtrajectory* e *Full-Move Subtrajectory*. Essas entidades do modelo são criadas com base nos instantes e localizações de *Pick-Up* e *Drop-Off Points* registradas pelo motorista. Os pontos de *Pick-Up* e *Drop-Off* definem os valores dos atributos *Start* e *End* de cada subtrajetória e a coleção de pontos neste intervalo de tempo definem o atributo *Course*, isto é, o percurso de cada subtrajetória. Desta forma, no exemplo da jornada do motorista, com base nos dados da trajetória coletados pela aplicação e nas marcações feitas pelo motorista de táxi, foram criadas três *Empty-Move Subtrajectory*, três *Full-Move Subtrajectory* e três candidatos a *Taxi Stop Point* (Figura 3.3).

Figura 3.3 - Jornada de Trabalho – processo de identificação das entidades do modelo

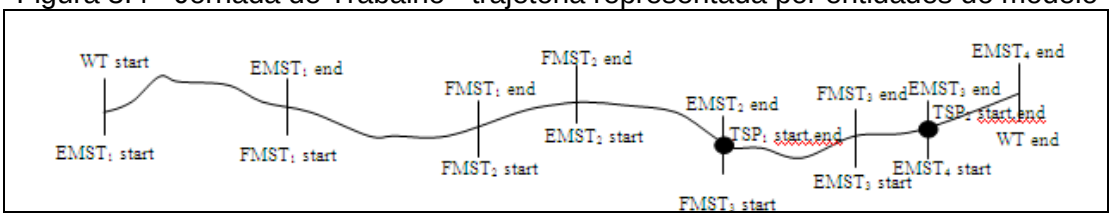


As entidades candidatas a *Taxi Stop Point* são identificadas nos dados brutos através de algoritmos que detectam a ocorrência de paradas do veículo. Esta análise só é realizada em trechos da trajetória sem passageiros. O principal problema na identificação desta entidade é a distinção entre paradas em pontos de táxi e de paradas que ocorrem durante uma *Empty-Move Sub-Trajectory* decorrentes de algum evento externo. Este último tipo de parada pode ser causado, por exemplo, por um engarrafamento ou defeito mecânico do veículo. Esses eventos não são do nosso interesse e por isso não serão explicitamente representados no modelo.

Considera-se inicialmente que todas as paradas realizadas por táxis sem passageiros são, em princípio, paradas em ponto de táxi (*Taxi Stop Point Candidate*). Depois, confronta-se os pontos de parada com a localização de pontos de táxis conhecidos (oficiais ou não). Desta forma, somente os *Taxi Stop Point Candidate* próximos a um *Taxi Stand* (Definição 7) é que são confirmados. Para os dados utilizados em nosso exemplo, o primeiro candidato a *Taxi Stop Point* foi decorrente de um evento externo (i.e., caminhão manobrando na via) e este tipo de evento não é representado no modelo, então esse *Taxi Stop Point* é descartado.

No caso dos dois últimos *Taxi Stop Point* a localização da parada coincide com um ponto de táxi. Desta forma, os dois *Taxi Stop Point Candidate* são confirmados e dão origem às duas entidades do tipo *Taxi Stop Point* (Figura 3.4). Com a criação do último *Taxi Stop Point*, a terceira *Empty-Move Sub-Trajectory* (*EMST*$_3$) foi dividida em duas novas entidades do tipo E*mpty-Move Sub-Trajectory*, intercaladas por um *Taxi Stop Point*.

Figura 3.4 - Jornada de Trabalho - trajetória representada por entidades do modelo



A estruturação dos dados brutos das trajetórias de táxi em um modelo orientado a objetos com entidades de primeira classe objetiva facilitar a obtenção de informações importantes para as aplicações voltadas ao estudo desse modal de transporte. Acreditamos que o *TX-Trajectory* irá facilitar a codificação de métodos voltados para a mineração de dados, descoberta de conhecimento e identificação de

padrões de comportamento dos usuários de táxi. De forma a garantir a persistência dos dados e permitir a realização de consultas através de SGBDs (Sistema de Gerenciamento de Banco de Dados), faz-se necessário o mapeamento das entidades do modelo em memória para tabelas de um banco de dados espacial (mapeamento objeto-relacional). Esse mapeamento é discutido na próxima seção.

3.2.4 Processo de Carga do Modelo em Banco de Dados

Objetivando ilustrar o mapeamento dos dados brutos em entidades do modelo e o respectivo mapeamento dessas entidades para o banco de dados foi utilizado dados reais de trajetórias brutas geradas por táxis rodando em San Francisco, Califórnia. Os dados utilizados são de trajetórias brutas geradas por 536 táxis rodando entre 17 de Maio e 10 de Junho de 2008. Estas movimentações geraram aproximadamente 12 milhões de registros. Este conjunto de dados está disponível no *website* do projeto CRAWDAD[3]. Os dados brutos estão na forma de arquivos texto individualizados por motorista de táxi. Cada arquivo possui tuplas com informações de latitude, longitude, hora e status. O campo status indica se o táxi está livre (*empty)* ou ocupado (*full)*. Não há informação sobre o número de passageiros no veículo, nem do valor da corrida. A evolução da posição do táxi ao longo do tempo é coletada em torno de cada minuto.

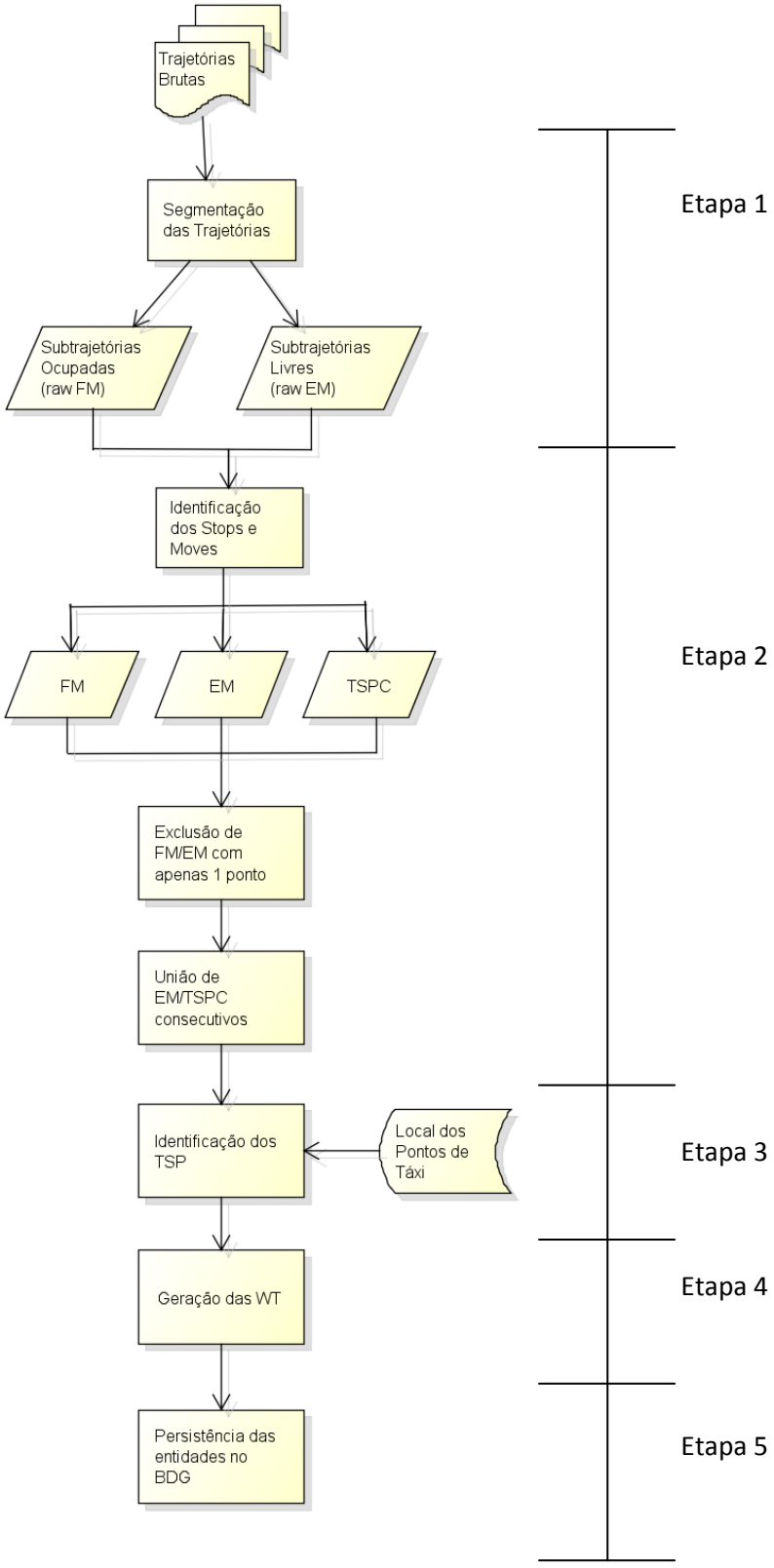
A Figura 3.5 ilustra o fluxo do processo de carga dos dados brutos das trajetórias de táxi em um Banco de Dados Geográfico (BDG) com a utilização do modelo *TX-Trajectory.* Este processo é dividido em cinco etapas. Na primeira etapa, os dados foram divididos em dois subconjuntos, *Raw Full-Move Sub-trajectory (rFMST)* e *Raw Empty-Move Sub-trajectory (rEMST)*, de acordo com o status de ocupação do veículo. Estes subconjuntos intermediários guardam a mesma estrutura dos dados brutos.

Na segunda etapa, apenas as *rEMST* são analisadas. Os dados brutos dessas sub-trajetórias são convertidos em *Stops* e *Moves*. Para isso é utilizado o algoritmo CB-SMoT *(Clustering-Based Stops and Moves of Trajectories)* (PALMA *et al.*, 2008), que foi desenvolvido para identificar *Stops* e *Moves* com base na variação de velocidade. Pesquisas indicam que esse algoritmo produz os melhores resultados para aplicações voltadas ao tráfego de veículos (BOGORNY *et al.*, 2011).

---

[3] http://crawdad.cs.dartmouth.edu/meta.php?name=epfl/mobility

Figura 3.5 - Fluxo do processo de carga dos dados em um BDG



A saída do algoritmo CB-SMoT é uma sequência de *Stops* e *Moves*. Dessa forma, todos os *Moves* são convertidos em entidades *EMST* do modelo *TX-Trajectory*. No entanto, os *Stops* são convertidos em uma entidade temporária chamada *Taxi Stop Point Candidate (TSPc).* Devido à calibragem do algoritmo *CB-SMoT*, identifica-se casos onde um *TSPc* é seguido por outro. Nesses casos, os dois *TSPc* formam um único *TSPc* com o primeiro ponto do primeiro *TSPc* como *Start* e o último ponto do último *TSPc* como *End*.

O tratamento de identificação dos *Stops* e *Moves* dado às *rEMST* não precisa ser aplicado às *rFMST*. O fato de não existir interesse em paradas do táxi enquanto o veículo se encontra com passageiro, permite simplificar o tratamento dos dados brutos e converter *rFMST* diretamente em entidades do tipo *FMST*.

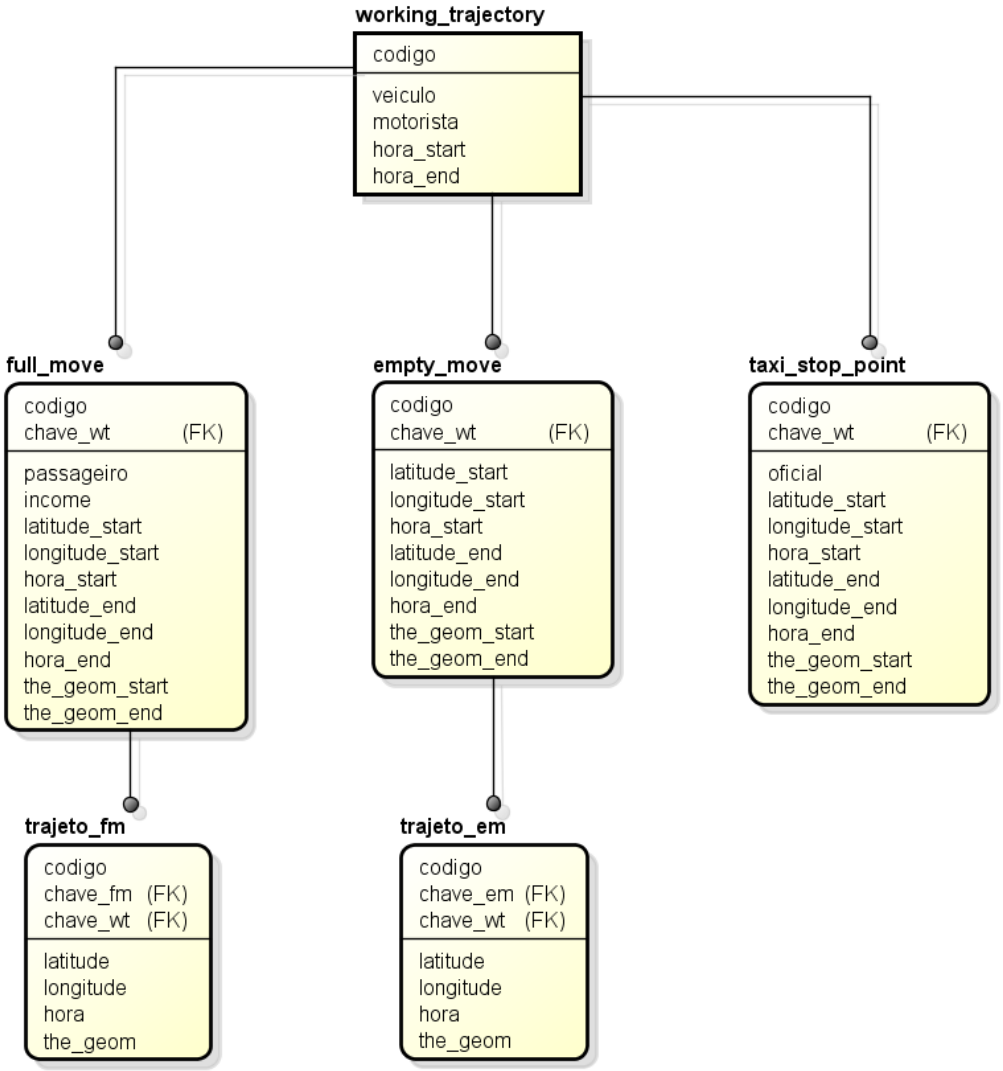
Todos os *Stop*s identificados durante o processamento da *rEMST* são considerados como candidatos a se converterem em entidades do tipo *TSP*. Muitos dos *Stop*s são verdadeiramente paradas em pontos de táxi oficiais e outros são causados por algum evento externo (e.g., engarrafamento) ou paradas em pontos de táxi não oficiais. Para o caso de paradas por motivos externos, considera-se estes *Stop*s como falso positivos. Os *Stop*s falso positivos deveriam ser incorporados a *EMST* mais próxima e não serem tratados como *TSP*. Por uma questão de simplicidade, decidiu-se converter todos *TSPc* para um *TSP* do modelo *TX-Trajectory*. Assim, se o ponto de *Start* ou *End* dessa entidade se encontra dentro de um determinado raio em torno da posição de um ponto de táxi oficial, definiu-se o atributo *Official* da entidade *TSP* como *true*, caso contrário, definiu-se como *false* (etapa 3). O tratamento dos *TSPc* falso positivos ficou como indicação para trabalhos futuros.

Baseado na análise dos dados brutos do projeto CRAWDAD, adotou-se como estratégia o encerramento da jornada de trabalho do motorista toda vez que fosse identificado um período de inatividade (desligamento do GPS) por períodos superiores a duas horas. Entretanto, esta estratégia não funcionou bem e gerou jornadas de trabalho com duração de vários dias, incompatíveis com o esperado para este tipo de atividade. A investigação dessas jornadas de trabalho identificou a existência de diversos pontos de parada de táxi não oficiais com a duração de várias horas. Analisando a localização destas paradas, observou-se que estas eram sempre na mesma localização, isto é, a garagem da empresa da frota de táxis. Desta forma, supôs-se que não era rotina dos motoristas desligar o GPS do veículo

no término da jornada de trabalho. Assim, alterou-se a estratégia para o encerramento da jornada de trabalho do motorista e passou-se a fazer o encerramento toda vez que o motorista se dirigia para a garagem da empresa e a sua jornada já ultrapassava quatro horas de duração (etapa 4).

A última etapa do processo trata do mapeamento das entidades do modelo *TX-Trajectory* para as tabelas do banco de dados espacial PostGIS (POSTGIS, 2013). O modelo *TX-Trajectory* é formado por seis tabelas: *working_trajectory*, *full_move*, *trajeto_fm*, *empty_move*, *trajeto_em* e *taxi_stop_point*. O modelo relacional do *TX-Trajectory* pode ser visto na Figura 3.6.

Figura 3.6 - TX-Trajectory – Modelo Relacional



A tabela *working_trajectory* guarda as jornadas de trabalho dos motoristas. Esta tabela contém informações do nome do motorista, veículo utilizado, data/hora do início e fim da jornada. O atributo *codigo* corresponde à chave primária da tabela e identifica a jornada do motorista. Cada linha da tabela corresponde a uma jornada de trabalho.

A tabela *full_move* guarda os deslocamentos com passageiro. O atributo *codigo* corresponde à chave primária da tabela e identifica a *full_move*. Nesta tabela encontram-se as informações da quantidade de passageiros, o valor da corrida (*income*), o ponto de origem e o ponto de destino do passageiro. Cada jornada de trabalho é formada por diversos deslocamentos com passageiro. O atributo *chave_wt* é uma chave estrangeira e identifica a jornada de trabalho da *full_move*. A tabela *trajeto_fm* guarda o trajeto percorrido no deslocamento com passageiro. Cada linha dessa tabela corresponde a um ponto do deslocamento. O atributo *codigo* corresponde à chave primária da tabela. O atributo *chave_fm* é uma chave estrangeira e identifica a *full_move* que deu origem a esse trajeto. O atributo *chave_wt* também é uma chave estrangeira e identifica a jornada de trabalho da *full_move*. O mesmo relacionamento aqui descrito se aplica às tabelas *empty_move* e *trajeto_em*.

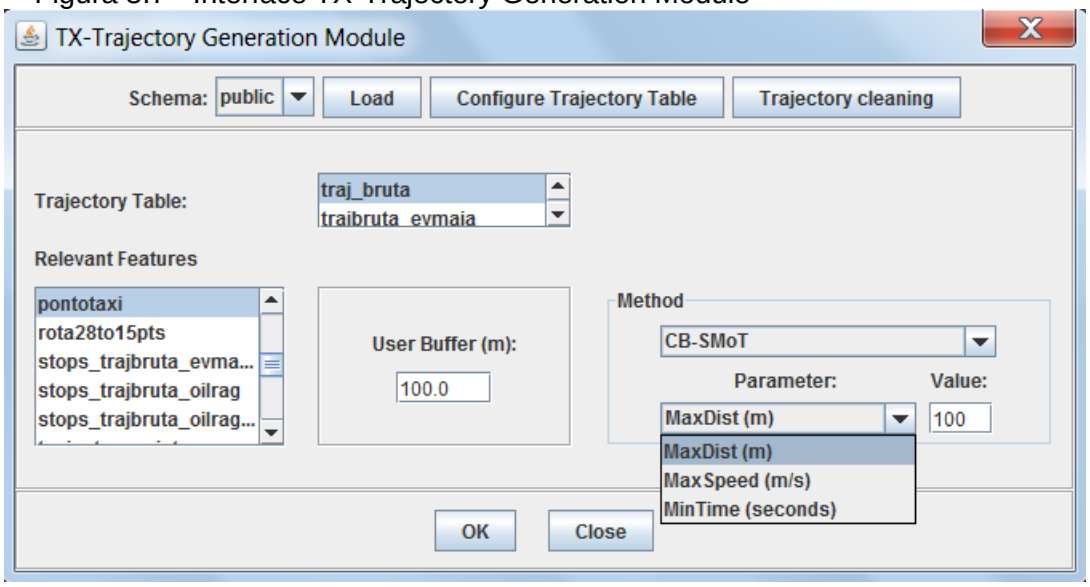
A tabela *taxi_stop_point* guarda os locais onde o motorista parou para aguardar por passageiro. O atributo *codigo* corresponde à chave primária da tabela e identifica o *taxi_stop_point*. O atributo *oficial* indica se esse é um ponto de táxi reconhecido oficialmente ou não. O atributo *chave_wt* é uma chave estrangeira e identifica a jornada de trabalho do *taxi_stop_point*. Os outros atributos indicam o local/hora que o motorista chegou e saiu do ponto de táxi.

Toda tabela espacial precisa ter um atributo para representação da geometria. Esses atributos são formados por tipos geométricos (e.g., ponto, linha, polígono) e são utilizados em bancos de dados geográficos (BDG) para construir consultas espaciais. Nas tabelas *full_move*, *empty_move* e *taxi_stop_point* os atributos espaciais *the_geom_start* e *the_geom_end* são atributos geométricos do tipo ponto. Nas tabelas *trajeto_fm* e *trajeto_em* o atributo *the_geom* também é do tipo ponto e o conjunto de pontos correspondentes a uma *FMST* ou *EMST* representa o percurso da subtrajetória.

Ambas as estruturas de dados (i.e., memória ou banco de dados), servem como base para a implementação de algoritmos e execução de *queries* (consultas) com o objetivo de descobrir padrões e produzir novas visões dos dados.

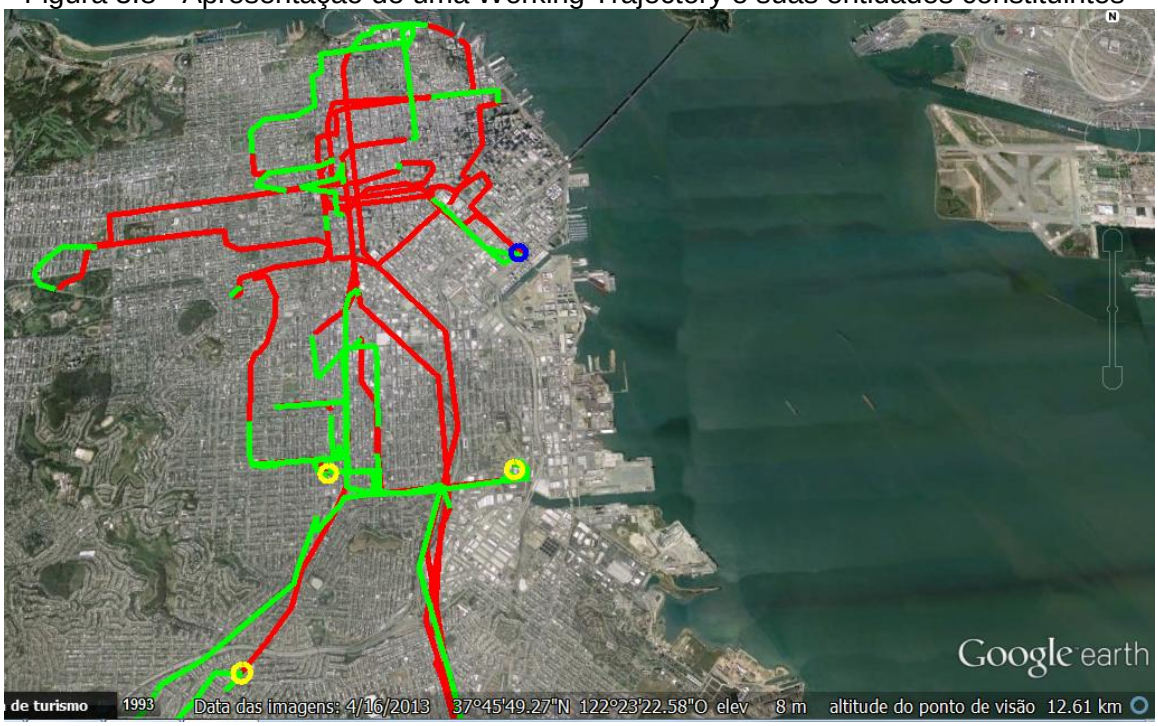
Todas as etapas do processo de mapeamento dos dados brutos das trajetórias para entidades do modelo em memória e persistência em banco de dados foram feitas através de uma aplicação denominada *TX-Trajectory Generation Module* (Figura 3.7). Esta aplicação foi desenvolvida para que a mesma pudesse ser facilmente acoplada à aplicação Weka-STPM (BOGORNY *et al.*, 2011). Weka é uma ferramenta gratuita e de código aberto, voltada para mineração de dados em bancos de dados tradicionais. É muito utilizada em instituições acadêmicas (FRANK *et al.*, 2005). Uma extensão da ferramenta Weka, com suporte a dados espaciais, denominada Weka-GDPM (*Geographic Data Prepocessing Module*) foi proposta em 2006 (BOGORNY; PALMA; *et al.*, 2006). Em 2011 foi proposta uma nova extensão da ferramenta Weka, denominada Weka_STPM (*Semantic Trajectory Preprocessing Module*) com suporte a dados de trajetória, com enriquecimento semântico dos dados, *data mining* em trajetórias semânticas e visualização de padrões de dados de objetos móveis (BOGORNY *et al.*, 2011). O módulo Weka-STPM serviu de modelo para a criação do *TX-Trajectory Generation Module*.

Figura 3.7 - Interface TX-Trajectory Generation Module



Os detalhes da implementação e utilização do *TX-Trajectory Generation Module* estão apresentados no Apêndice B desta dissertação. De forma a ilustrar os dados de saída da aplicação, utilizou-se como entrada um subconjunto dos dados da trajetória bruta de um dos motoristas disponibilizado no projeto CRAWDAD (Figura 3.8). Na Figura, as linhas vermelhas representam os deslocamentos com passageiro (*Full-Move Sub-Trajectory*). As linhas verdes representam os deslocamentos sem passageiro (*Empty-Move Sub-Trajectory*). Os círculos representam as paradas em pontos de táxi (*Taxi Stop Point*), em azul estão representados os pontos de táxi oficiais e em amarelo os pontos de táxi não oficiais. A localização dos pontos de táxi oficiais foi obtida do órgão de transporte da Prefeitura de San Francisco[4].

Figura 3.8 - Apresentação de uma Working Trajectory e suas entidades constituintes



3.3 COMPARAÇÃO COM OUTROS MODELOS

O modelo *TX-Trajectory* foi concebido para servir como um modelo de dados básico para aplicações com interesse em representar e analisar os movimentos dos táxis. Para atingir esse objetivo, o modelo precisa ser abstrato o suficiente para identificar apenas a essência dos movimentos dos táxis e concreto o suficiente para extrair

[4] http://www.sfmta.com/cms/xcust/taxistands.html acessado em março de 2013.

aspectos importantes relativos ao domínio da aplicação. A estrutura dos dados das trajetórias de táxi em entidades mais abstratas cria uma infraestrutura de dados que irá facilitar as operações voltadas para a mineração de dados, descoberta de conhecimento e identificação de padrões de deslocamento. Acreditamos que o modelo desenvolvido atinge o equilíbrio de ser simples e suportar uma ampla variedade de consultas e análises.

Encontrou-se alguns trabalhos que possuem uma forte similaridade com o modelo *TX-Trajectory*. O modelo de (SPACCAPIETRA *et al.*, 2008) serviu de inspiração para o desenvolvimento do *TX-Trajectory Model*. Entretanto, para a representação do movimento de táxis, não se pode utilizar apenas sequências de *Stops* e *Moves.* Apesar de termos identificado que a movimentação dos táxis se caracterizava basicamente por *stops* em pontos de táxi e *moves*, os *moves* precisariam ser identificados em deslocamentos com passageiro e sem passageiro. Além disso, a restrição de alternância entre *Stops* e *Moves* não é verdadeira no domínio de táxi, neste domínio existe uma intercalação de *FMST*, *EMST* e *TSP*.

No trabalho de (YUAN *et al.*, 2011), apesar de o objetivo ser a apresentação de um sistema de recomendação com sugestões para motorista de táxis e passageiros, conseguiu-se identificar a utilização de três entidades: *taxi trajectory, taxi trip* e *parking place*. A entidade *taxi trajectory* corresponde a uma trajetória de táxi e é definida como uma sequência de pontos de GPS, onde cada ponto é formado pelas seguintes informações: tempo, latitude, longitude, segmento da rodovia e estado de ocupação (ocupado/desocupado). A entidade *taxi trip* corresponde a uma viagem de táxi e é definida como uma subtrajetória tendo um único estado, *cruising* ou *occupied*. A entidade *parking place* é definida como sendo um local qualquer onde o motorista aguarda por passageiro, nesse caso o estado da subtrajetória é *parking*. Embora os atributos da entidade *taxi trajectory* tenham sido definidos, não ficou determinado quando começa e termina uma *taxi trajectory,* nem se essa entidade corresponde ou não a uma jornada de trabalho do motorista. Também não foi mostrado como essa entidade é segmentada em *trips* ou se existe alguma restrição para essa segmentação. As entidades *taxi trip* e *parking place* não tiveram seus atributos definidos, além de não ter sido mostrado se existe alguma relação entre essas entidades e a entidade *taxi trajectory*. Esses aspectos que não ficaram suficientemente claros no trabalho de Yuan serviram como itens a serem analisados e verificados no modelo *TX-Trajectory*.

A área de pesquisa semântica em trajetória ainda está no seu início. A semântica na área de trajetórias pode estar, por exemplo, nas atividades realizadas pelo objeto móvel, nos relacionamentos espaço-temporais dessa trajetória com outras trajetórias ou outros objetos. O próximo capítulo aborda a utilização e a aplicação do *TX-Trajectory Model* através de algumas análises e consultas feitas à base de dados construída de acordo com esse modelo.

4 APLICAÇÃO E AVALIAÇÃO DO MODELO

Este capítulo apresenta algumas análises da movimentação dos táxis utilizando o modelo *TX-Trajectory*. As análises aqui apresentadas não são exaustivas e servem apenas para demonstrar o potencial do modelo. O objetivo é mostrar como o modelo *TX-Trajectory* pode ser utilizado para facilitar o processo de extração de conhecimento a partir dos dados das trajetórias de táxi gerados pelo GPS.

A próxima seção apresenta algumas análises, comumente encontradas na literatura, relacionadas à movimentação dos táxis (PENG *et al.*, 2012) (VELOSO *et al.*, 2011) (KAMAROLI *et al.*, 2011) (ZHENG *et al.*, 2011) (GE *et al.*, 2011) (LIU *et al.*, 2009) (YUAN *et al.*, 2011).

A última seção faz uma comparação do procedimento de extração de conhecimento através do uso do modelo *TX-Trajectory* e através do uso dos dados brutos das trajetórias dos táxis.

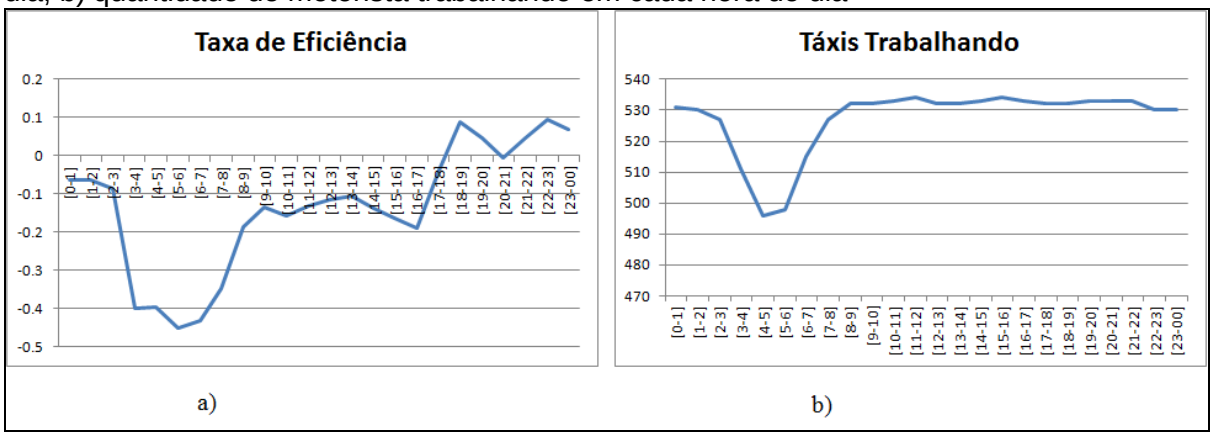
4.1 ANÁLISES USUAIS DE TRAJETÓRIAS DE TÁXI

Uma das análises que se deseja realizar é a medição da eficiência do sistema de táxi. Considerou-se que quanto mais os motoristas rodam com seus veículos ocupados, mais eficiente é o sistema. Diferente de (YUAN *et al.*, 2011) que utiliza a distância percorrida com e sem passageiros como um dos critérios de avaliação do sistema, optou-se por avaliar a eficiência do sistema de acordo com o tempo que o motorista trabalhou com seu veículo ocupado. Acreditamos que a utilização do tempo ocupado/livre seja mais adequada para a medição da eficiência, considerando o trânsito conturbado das grandes metrópoles. Desta forma, objetivando sumarizar o funcionamento do sistema de táxi em um único valor, propôs-se uma métrica para medir a eficiência do sistema chamada de taxa de eficiência (ER, em inglês, *efficiency rate*). A taxa de eficiência é definida em um intervalo de tempo delimitado por dois instantes, *i* e *j*, e considera apenas as características temporais (i.e., duração) das entidades *FMST*, *EMST* e *TSP* (equação 1).

$$ER_{[i,j]} = \frac{\sum \text{duration}(FMST, i, j) - (\sum \text{duration}(EMST, i, j) + \sum \text{duration}(TSP, i, j))}{\sum \text{duration}(FMST, i, j) + \sum \text{duration}(EMST, i, j) + \sum \text{duration}(TSP, i, j)} \quad (1)$$

A função *duration* na equação 1 calcula a duração de uma entidade do modelo dentro do intervalo de tempo [i, j]. Assim, o valor de ER varia no intervalo de -1 a 1, onde 1 significa que todos os táxis estão ocupados durante o intervalo de tempo [i, j] e -1 significa que todos os táxis estavam parados aguardando um passageiro em um ponto de táxi ou em movimento tentando encontrar um passageiro. A Figura 4.1a mostra a evolução da ER considerando todos os motoristas de táxi. A fim de facilitar a interpretação da ER, calculou-se também o número de motoristas de táxi trabalhando a cada hora do dia (Figura 4.1b).

Figura 4.1 - Análise da eficiência do sistema de táxi: a) evolução da Taxa de Eficiência no dia; b) quantidade de motorista trabalhando em cada hora do dia



A demanda por táxi varia no tempo e no espaço de acordo com a necessidade dos cidadãos. Analisando a evolução da ER, pode-se obter algum conhecimento a respeito do funcionamento do sistema. Considerando que uma ER negativa reflete uma baixa procura por táxi, pode ser observado que a demanda começa a diminuir às 23h e atinge o seu mínimo entre 5-6h. Depois disso, uma melhoria da ER reflete um aumento na demanda. Entre 08h e 17h o sistema atinge uma espécie de equilíbrio com ER próxima de zero. Isso significa que durante esse período a probabilidade de encontrar um táxi ocupado ou sem passageiro é quase a mesma. Depois de 5h a demanda por táxi começa a aumentar e alcança o seu máximo em torno de 19h. Provavelmente este crescimento é causado pelas pessoas saindo do trabalho. Depois das 19h a demanda por táxi começa a diminuir, porém volta a crescer entre 22-23h. Esta retomada do crescimento pode ser motivada tanto por pessoas que saem de casa para ir ao cinema, teatro, restaurante ou qualquer outro entretenimento, tanto por pessoas voltando para casa depois do *happy hour*.

O cálculo da ER não leva em conta o número de táxis disponíveis no intervalo de tempo. Assim, uma análise da quantidade de veículos trabalhando ao longo do dia pode ajudar na análise da avaliação do índice de eficiência do sistema de táxi. O número de motoristas de táxi trabalhando é praticamente o mesmo durante todo o dia (Figura 4.1b), exceto no período que coincide com a menor ER (Figura 4.1a). Assim, os resultados das análises dos dois gráficos mostram que a ER só não é pior por volta do amanhecer porque alguns motoristas não trabalham neste período. Essas análises mostram também que os gestores da frota podem optar por diminuir o número de táxis rodando no período da madrugada, com isso o número de veículos rodando sem passageiro iria diminuir e a taxa de eficiência do sistema iria aumentar.

Um dos problemas da utilização da ER para a análise da eficiência do sistema de táxi é que este índice não carrega nenhuma informação espacial, como, por exemplo, quais os locais da cidade onde os táxis estão sendo mais requisitados. No entanto, com o apoio de uma ferramenta SIG – Sistema de Informação Geográfica (ou GIS, do inglês *Geographic Information System)* é possível realizar consultas no banco de dados espacial e produzir novas visões do mapa mostrando a distribuição espacial desses dados. Para a apresentação dos resultados das análises seguintes foi utilizado o Quantum GIS (QGIS, 2013).

Em uma aplicação voltada para o gerenciamento de uma frota de táxi, por exemplo, alguém pode querer saber:

Q1: Quais os locais da cidade que requisitaram táxi no dia 20/05/2008 entre 5-6h da manhã? (Figura 4.2)

Do ponto de vista do motorista de táxi, por exemplo, seria interessante saber o local com maior demanda por táxi no intervalo entre 5-6h, onde o índice de ocupação dos veículos é o mais baixo do dia (Figura 4.1a). A partir da construção de uma consulta SQL que informe a localização espacial do ponto inicial de todas as *FMST* entre 5-6h do dia 20/05/2008 (terça-feira), o sistema pode oferecer uma indicação ao taxista de onde encontrar o próximo passageiro durante esse intervalo de tempo, nesse dia da semana, através da concentração dos pontos no mapa. Para responder a essa pergunta, precisa-se da informação semântica relacionada às

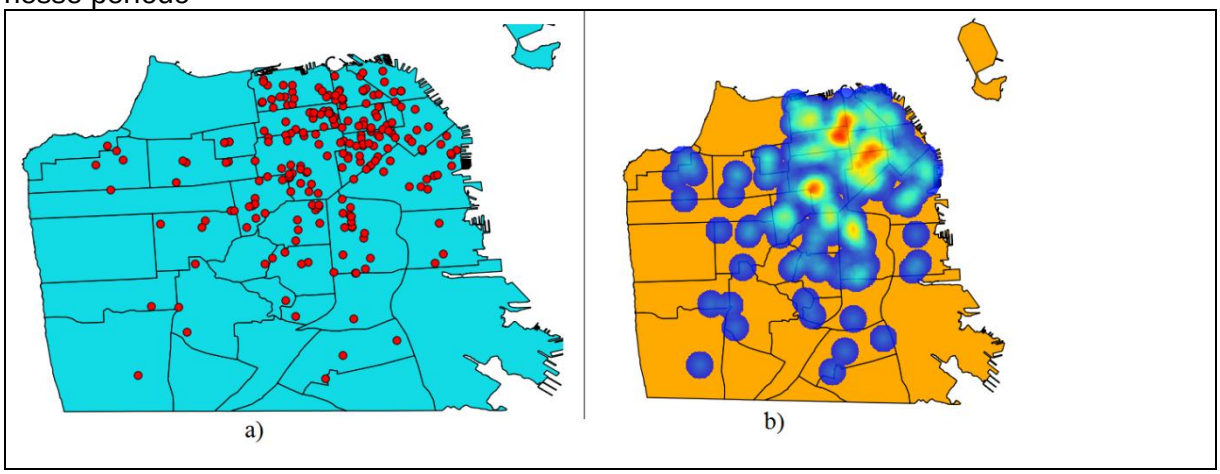
trajetórias de táxi, que identifica onde inicia um deslocamento do veículo com passageiro. O comando SQL poderia ser o seguinte:

```
SELECT * FROM full_move
WHERE to_timestamp(hora_start) >= ‘2008-05-20 05:00:00’ AND
      to_timestamp(hora_start) < ‘2008-05-20 06:00:00’
```

Acessando a base de dados construída com a utilização do modelo, executou-se o comando SQL acima, que retornou 224 ocorrências em 3533ms.

A Figura 4.2 mostra o resultado dessa consulta em dois estilos diferentes: um mapa de ponto com a localização de cada ponto de *pick-up* (Figura 4.2a) e um mapa de calor dando uma ideia das regiões com alta concentração de pontos (Figura 4.2b).

Figura 4.2 - Origem das chamadas em 20/05/2008 de 05:00 – 06:00h. a) pontos com a localização dos *pick-ups*; b) mapa térmico mostrando as regiões com maior demanda de táxi nesse período



Utilizando os dados brutos das trajetórias dos táxis e nossos conhecimentos da linguagem SQL, não conseguimos construir nenhuma consulta SQL capaz de responder os questionamentos aqui apresentados. Para essa primeira análise, apenas com objetivo comparativo, foi desenvolvida uma aplicação para responder a pergunta (Q1) utilizando os dados brutos das trajetórias. Geralmente as informações que se precisa obter em relação às corridas de táxi não estão no nível mostrado pelos dados brutos e sim em nível mais baixo, onde é preciso identificar os deslocamentos com passageiro ou sem passageiro. Além disso, existe uma

distorção no resultado das consultas, já que não existe nenhum tipo de limpeza de dados gerados pelo GPS. A seguir, é mostrado o trecho da aplicação onde são selecionados os dados que satisfazem a condição da pesquisa anterior (Figura 4.3). A aplicação apresenta apenas uma das maneiras de extrair as informações solicitadas.

Figura 4.3 - Trecho de uma aplicação para extração de informação

```
public void loadTelems(int veiculo) throws SQLException {
    Statement s1 = conn.createStatement();
    String sql2 = "SELECT hora,status FROM traj_bruta WHERE veiculo=" + veiculo + " ORDER BY gid Desc ";
    ResultSet rs2 = s1.executeQuery(sql2);
    int xav = -1;
    int status_reg, dia, mes, ano;
    long hora_reg;
    while (rs2.next()) { // para cada ponto da trajetoria ordenado pela hora
        status_reg = rs2.getInt("status");
        hora_reg = rs2.getLong("hora");
        if ((status_reg == 1)) {
            if ((xav == -1)) {  //é uma informação de FullMove
                //verificar se satisfaz a condição procurada
                //verificar se data = 2008-05-20
                Calendar cal = Calendar.getInstance();
                cal.setTimeInMillis(hora_reg * 1000);
                dia = cal.get(Calendar.DAY_OF_MONTH);
                mes = cal.get(Calendar.MONTH);
                ano = cal.get(Calendar.YEAR);
                long hora = cal.get(Calendar.HOUR_OF_DAY);
                if ((mes > 4)||((mes == 4)&&(dia > 20))){
                    break;
                }
                if ((dia == 20) && (mes == 4) && (ano == 2008) && (hora == 9)) {  //20-05-2008 09:**:**
                    cont = cont + 1;
                    System.out.println("cont = " + cont);
                }
                xav = 0;//procurar o próximo FM

            }
        } else {
            xav = -1;
        }

    }
}
```

Outro tipo de análise recorrente no contexto de táxi é aquela que considera apenas os pontos de *pick-up* e *drop-off* da trajetória, isto é, a origem e o destino dos passageiros. A seguir é mostrada uma consulta que apresenta a quantidade de deslocamento, por bairro destino, dos passageiros de táxi. Uma aplicação voltada para a gestão de serviço público de transporte poderia se interessar por uma

informação desse tipo. Como exemplo hipotético, pode-se considerar que em determinado período do ano existe um evento na cidade. Analisando os anos anteriores o gestor pode verificar que o destino das viagens de táxi, durante o período do evento, dobra para um determinado bairro. Esse fato indica que pode haver a necessidade de aumentar a frota de transporte público para essa região durante o período do evento. A pergunta a ser respondida é:

Q2: Qual a quantidade de deslocamento dos usuários de táxi da cidade, totalizado por bairro destino? (Figura 4.4b)

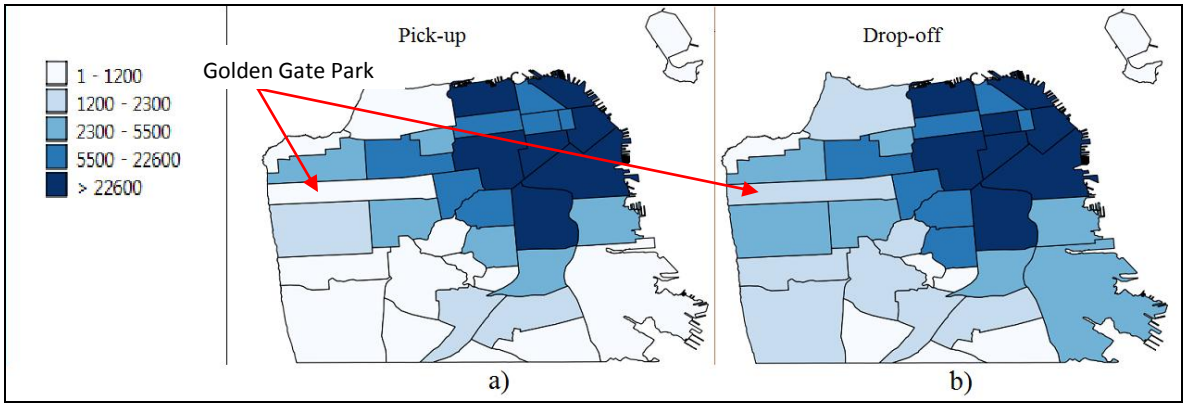
Para responder a essa pergunta, precisa-se apurar a quantidade de deslocamentos com passageiro tendo cada bairro como destino. Para isso será utilizada a tabela ***full_move***, gerada com o uso do modelo, com os deslocamentos com passageiro e outra tabela com a localização dos bairros da cidade de San Francisco. O comando SQL poderia ser o seguinte:

```
CREATE TABLE dropoff_bairro  AS
SELECT count(*) AS qtde, b.gid, b.the_geom, b.neighborho
FROM bairros b, full_move fm
WHERE st_contains(b.the_geom, fm.the_geom_end)
GROUP BY b.gid, b.the_geom, b.neighborho
```

Nesse tipo de análise a trajetória do veículo não é avaliada. Para esse caso foi produzido um mapa de graduação de cores. Dessa forma, consegue-se ter uma ideia das regiões com alta demanda de táxis, *pick-ups* (Figura 4.4a) e os destinos preferidos, *drop-offs* (Figura 4.4b). No entanto esses mapas não podem ser utilizados para estabelecer um padrão de origem-destino, uma vez que é impossível concluir se a maioria das viagens está dentro do mesmo bairro ou indo de um bairro para outro. Porém essas visões podem ser utilizadas como primeiro passo para uma análise mais detalhada de um bairro que apresente um padrão divergente, quando comparado com outros bairros da vizinhança. O *Golden Gate Park*, por exemplo, tem um pequeno número de *pick-up* e *drop-off,* se comparado com os bairros em torno dele (Figura 4.4). Está fora do escopo deste artigo a explicação de tal padrão dissonante, mas este fato pode ser usado por pessoas com algum interesse no

bairro *Golden Gate Park* para obter mais esclarecimentos sobre o uso de táxis nesse bairro.

Figura 4.4 - Mapa coroplético mostrando a quantidade de pick-ups e drop-offs nos distritos da área metropolitana de San Francisco



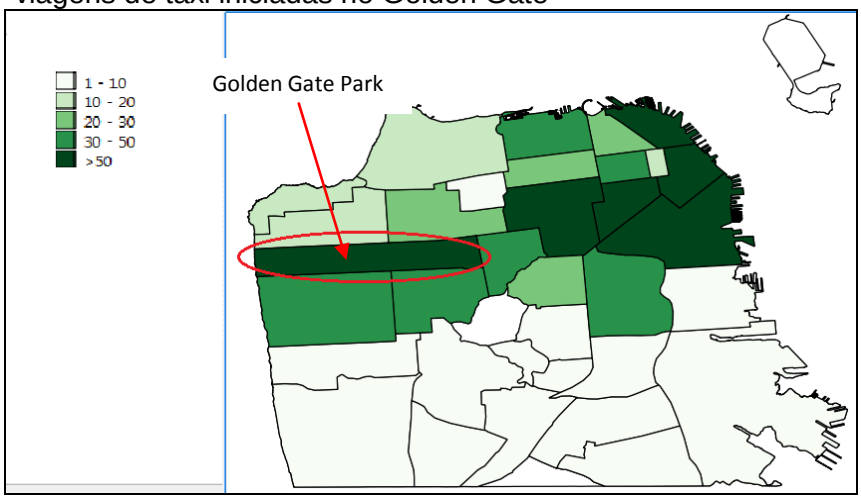
Objetivando aprofundar o conhecimento sobre o padrão de uso de táxi no bairro *Golden Gate Park*, pode-se investigar, por exemplo, quais os bairros de destino das viagens de táxi iniciadas neste bairro. Dando continuidade à demonstração da utilização do modelo, considere, agora, que alguém pode querer saber para onde estão se deslocando as pessoas que tomam táxi no bairro *Golden Gate Park*. A pergunta a ser respondida é:

Q3: Por bairro, qual a quantidade de viagens de táxi cuja origem foi o bairro Golden Gate (gid=28)? (Figura 4.5)

Para responder a essa pergunta, precisa-se apurar, por bairro, a quantidade de deslocamentos com passageiro tendo o bairro *Golden Gate Park* como origem. Executa-se uma consulta SQL contando o número de *drop-offs* das viagens que tiveram como ponto de *pick-up* o *Golden Gate Park.* Este resultado é individualizado por bairro e mostrado em um mapa coroplético (Figura 4.5). O mapa evidencia dois padrões dominantes: viagens dentro do próprio bairro, isto é, começa e termina dentro do *Golden Gate Park*, e viagens com destino à região financeira da cidade. O comando SQL poderia ser o seguinte:

```
CREATE TABLE origem28_destinos AS
SELECT count(*) AS qtde,b.gid,b.the_geom,b.neighborho
FROM   bairros b,full_move fm
WHERE st_contains(b.the_geom,fm.the_geom_end) AND
          st_within(fm.the_geom_start,(SELECT b1.the_geom from bairros b1
WHERE gid=28))
GROUP BY b.gid,b.the_geom,b.neighborho
ORDER BY qtde
```

Figura 4.5 - Mapa coroplético mostrando a quantidade de drop-offs das viagens de táxi iniciadas no Golden Gate



É possível considerar também nas análises a trajetória do veículo, isto é, o percurso do táxi. Imagine que o gestor do sistema de transporte da cidade esteja querendo analisar quais as vias mais utilizadas pelos motoristas no percurso que tem como origem o bairro *Golden Gate Park* e como destino o *Financial District*. A pergunta a ser respondida é:

Q4: Qual o percurso das viagens de táxi que tiveram como origem o bairro *Golden Gate Park* e como destino o *Financial District*? (Figura 4.6a)

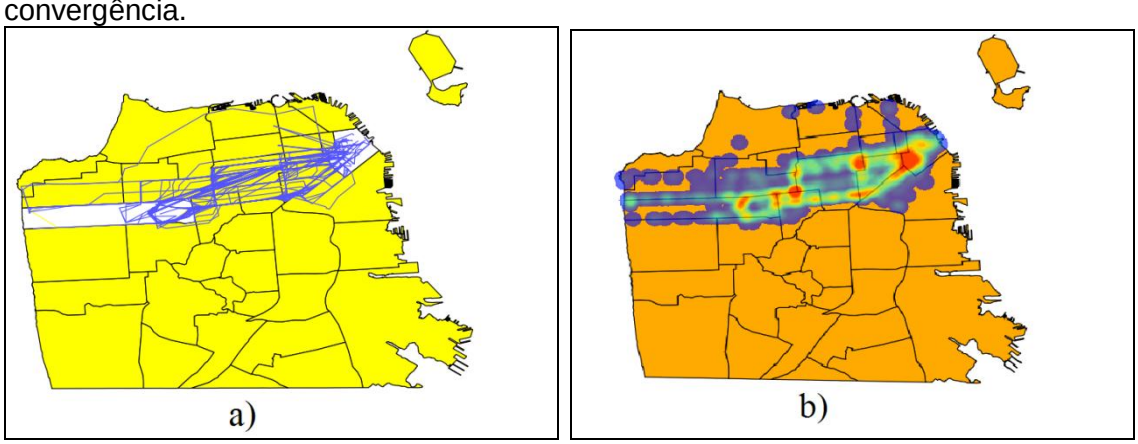
Para responder a esta pergunta precisa-se de uma consulta SQL que selecione todas *FMST* com origem no bairro *Golden Gate Park* e com destino no bairro *Financial District.* Esta informação pode ser representada tanto com um mapa de fluxo (Figura 4.6a) quanto com um mapa de calor, destacando as regiões de convergência do fluxo (Figura 4.6b). Para a construção dessa consulta, primeiro criou-se uma tabela (rota28to15pts) com os pontos dos trajetos das *Full-Move*

selecionadas saindo do *Golden Gate Park* (gid=28) com destino ao *Financial District* (gid=15). Depois, utilizando essa tabela, criou-se uma nova tabela com o caminho percorrido. O comando SQL poderia ser o seguinte:

```
CREATE TABLE rota28to15pts AS
SELECT traj.latitude, traj.longitude, traj.chave_fm FROM trajeto_fm traj
WHERE traj.chave_fm IN
        (SELECT fm.codigo FROM full_move fm WHERE
         st_within(fm.the_geom_start,(SELECT b1.the_geom FROM bairros b1
                                                    WHERE b1.gid=28))
         AND
         st_within(fm.the_geom_end,(SELECT b2.the_geom FROM bairros b2
                                                    WHERE b2.gid=15)))


CREATE TABLE rota28to15line AS
SELECT chave_fm,st_makeline(ST_MakePoint(longitude,latitude))
FROM (SELECT latitude, longitude, chave_fm FROM rota28to15pts) AS rota
GROUP BY chave_fm
```

Figura 4.6 - Mapas mostrando o caminho percorrido nas viagens de táxi iniciadas no Golden Gate Park com destino ao Financial District: a) rotas; b) destaque para os fluxos de convergência.



As análises apresentadas aqui servem apenas como prova de conceito e refletem análises recorrentes na literatura (PENG *et al.*, 2012) (VELOSO *et al.*, 2011) (KAMAROLI *et al.*, 2011) (ZHENG *et al.*, 2011) (GE *et al.*, 2011) (LIU *et al.*, 2009) (YUAN *et al.*, 2011). O método utilizado emprega a elaboração de consultas SQL sobre o modelo de dados armazenado em tabelas de um banco de dados relacional. O objetivo é demonstrar que a maioria das análises usuais pode ser realizada facilmente através do processamento das entidades do modelo *TX-Trajectory*.

A próxima seção realiza uma comparação do procedimento de extração de conhecimento através do uso do modelo *TX-Trajectory* e através do uso dos dados brutos. A intenção é mostrar que a utilização do modelo produz resultados confiáveis e com um menor custo computacional. Os métodos utilizados empregam o desenvolvimento de algoritmos que operam sobre entidades do modelo armazenadas em memória e sobre os dados brutos das trajetórias de táxis também carregados em memória.

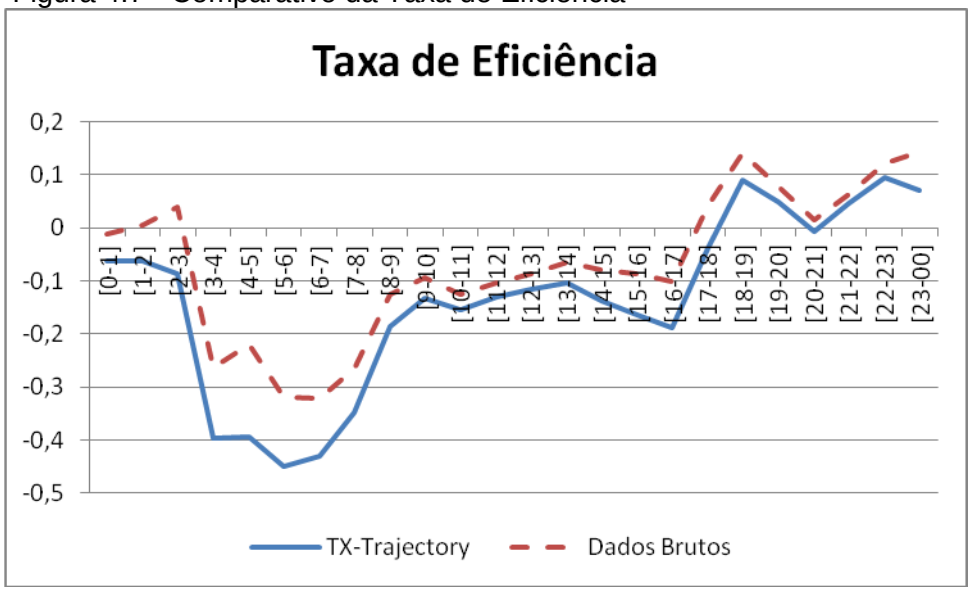
4.2 ESTUDO DA EFICIÊNCIA E EFICÁCIA DO MODELO *TX-TRAJECTORY*

Esta seção apresenta alguns exemplos que visam ilustrar a facilidade para extração de conhecimento quando se utiliza o modelo *TX-Trajectory* em detrimento da utilização dos dados brutos da trajetória. Além disso, mostra que esta extração é feita, na maioria das vezes, de forma mais rápida e produz resultados mais próximos da realidade. É importante salientar que os exemplos apresentados aqui não têm a pretensão de provar formalmente a eficiência e eficácia do modelo, mas serve somente ao propósito de ilustrar estes argumentos.

Objetivando comparar a qualidade dos resultados e a eficiência das análises, desenvolveram-se duas aplicações semelhantes para medir a eficiência do sistema, uma utilizando os dados brutos das trajetórias e outra utilizando os dados armazenados em entidades do modelo *TX-Trajectory*. A aplicação que utilizou os dados brutos executou em 24.460.174.214 nanoseg. A aplicação que utilizou o modelo executou em 13.574.266.443 nanoseg. Desta forma, verificou-se uma diminuição de 44,505% do tempo de execução com a utilização do modelo.

Analisando a qualidade dos resultados obtidos com a execução das duas aplicações (Figura 4.7), verifica-se que existe uma distorção no resultado obtido com a utilização dos dados brutos.

Figura 4.7 - Comparativo da Taxa de Eficiência



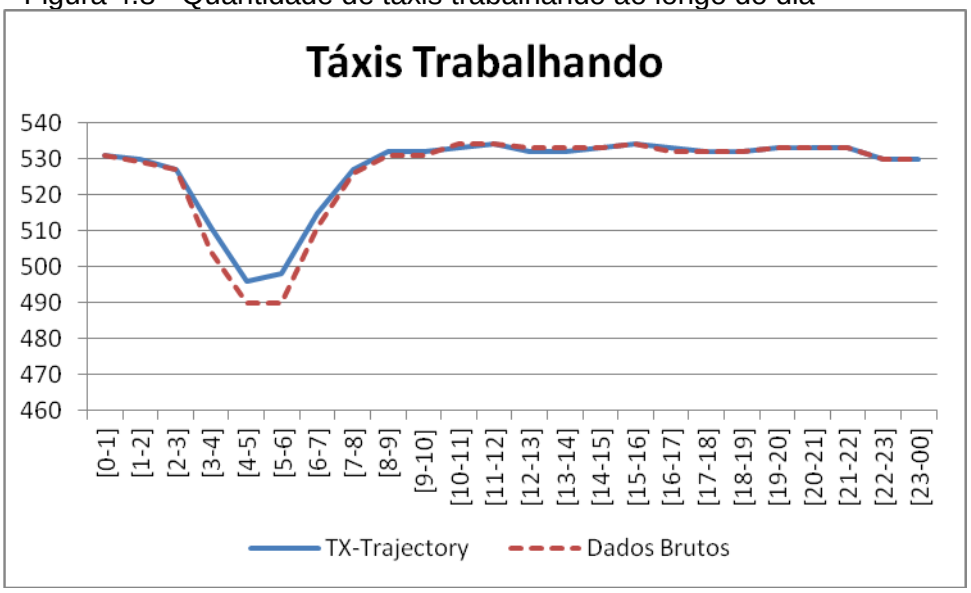
O motivo para esta diferença é que para calcular a taxa de eficiência foram desconsiderados os deslocamentos com duração superior a duas horas. Esta estratégia visa evitar que os táxis que ficaram parados na garagem da frota com o GPS ligado fizessem parte do cálculo da taxa de eficiência. Como os dados brutos só possuem a identificação dos deslocamentos de táxi com passageiro (*Full-Move*) ou sem passageiro (*Empty-Move*) e as paradas em pontos de táxi estão embutidas nos deslocamentos sem passageiro, existem muitas *Empty-Move* com duração superior a duas horas que serão desprezadas. Por outro lado, as *Empty-Move* geradas com a utilização do modelo foram analisadas e os pontos de táxi existentes nesse deslocamento deram origem aos *Taxi Stop Point*. Este procedimento divide um possível deslocamento com duração superior a duas horas em dois outros deslocamentos com duração inferior que não serão mais desprezados.

Analisando a equação para calcular a taxa de eficiência (equação 1), pode-se verificar que para utilizar os dados brutos, como estes não têm a identificação dos pontos de táxi, não pode-se calcular a duração dos *TSP*. Caso exista, por exemplo, *Empty-Move* com duração de 3 horas, ela será desprezada. Por outro lado, utilizando o modelo TX-Trajectory essa *Empty-Move* não será desconsiderada no cálculo já que ela poderá ser quebrada em uma *Empty-Move* com duração de 1,5 horas e um ponto de táxi com duração de 1,5 horas. Dessa forma, o resultado encontrado com a utilização dos dados brutos tende a ser maior que o resultado

encontrado com a utilização do modelo. Esse é o motivo da distorção no resultado encontrado com os dados brutos.

No segundo exemplo a quantidade de motoristas trabalhando na cidade de San Francisco foi medida em cada hora do dia. Da mesma forma que no exemplo anterior, duas aplicações semelhantes foram construídas, uma utilizando os dados brutos das trajetórias e outra utilizando os dados armazenados de acordo com o *TX-Trajectory*. A aplicação que utilizou os dados brutos executou em 12.713.587.023 nanoseg. A aplicação que utilizou o modelo executou em 11.766.520.486 nanoseg. Houve uma diminuição de 7,449% do tempo de execução com a utilização do modelo.

Figura 4.8 - Quantidade de táxis trabalhando ao longo do dia



Analisando o gráfico do resultado obtido com a execução das duas aplicações (Figura 4.8), também nesse exemplo pode-se verificar, que existe uma distorção no resultado mostrado com a utilização dos dados brutos. O motivo dessa distorção é o mesmo apontado no exemplo da Figura 4.7, ou seja, a rejeição dos deslocamentos com duração superior a duas horas no cálculo da quantidade de táxis trabalhando. Isso fez com que alguns deslocamentos correspondentes a *Empty-Move* fossem desconsiderados no cálculo onde os dados brutos foram utilizados, causando a diminuição do número de táxis trabalhando. Como os veículos utilizados nessas *Empty-Moves* não foram contabilizados, o número de táxis encontrados em

determinados períodos foi menor que o encontrado utilizando os dados carregados com a utilização do modelo.

Com aplicações semelhantes foi mostrado que a utilização do modelo apresenta resultados que refletem os dados coletados com maior fidelidade, além de existir um ganho considerável no tempo de execução dessas aplicações.

5 CONCLUSÃO

A crescente quantidade de dados com informações sobre a localização de objetos móveis tem despertado o interesse de pesquisadores no estudo de novas ferramentas, métodos e aplicações voltadas ao tratamento da trajetória desses objetos. Neste contexto, o estudo das trajetórias de táxis corresponde a uma área de pesquisa que tem atraído o interesse da comunidade científica. Ao contrário dos sistemas de ônibus, trem e metrô que possuem rotas e pontos de parada pré-definidos, táxis pegam e levam passageiros para onde eles desejam. Esta capilaridade permite uma determinação precisa para a análise voltada à origem e ao destino do deslocamento das pessoas.

O modelo *TX-Trajectory* apresentado neste trabalho foi desenvolvido para representar e estruturar a trajetória de táxis com a utilização de entidades mais abstratas e semanticamente significativas. O referido modelo tem como base a entidade que representa a jornada de trabalho do motorista. Essa entidade, por sua vez, é especializada pelas entidades que representam os deslocamentos do veículo com passageiro, sem passageiro e as paradas em pontos de táxi. O processo utilizado para converter os dados brutos das trajetórias de táxis em entidades do modelo *TX-Trajectory*, trata e estrutura essas informações brutas do deslocamento, fazendo com que as entidades passem a carregar informações semânticas relativas ao movimento de táxis.

Com o objetivo de avaliar o modelo, utilizou-se dados de trajetórias brutas geradas por 536 táxis na cidade de San Francisco, Califórnia, durante aproximadamente um mês. No processo de análise constatou-se que a utilização do modelo facilita a extração de conhecimento dos dados das trajetórias dos táxis.

A aplicação do modelo propiciou, mediante variadas formas de organização e análise de dados, a visualização de panoramas descritivos e possivelmente elucidativos de questões referentes a áreas de gestão de tráfego e de frotas de táxi. Adicionalmente, realizou-se uma análise comparativa dos procedimentos necessários para extração de conhecimento das trajetórias originais geradas pelo GPS, e das trajetórias enriquecidas semanticamente e organizadas de acordo com o modelo proposto.

Foram identificados alguns pontos relevantes nesta pesquisa que apontam para trabalhos futuros, a saber:

1) *TSP* falso positivo - *Taxi Stop Point* marcado como não-oficial decorrente de algum evento externo, como engarrafamento ou parada em posto de pedágio. Para aumentar a certeza na identificação dessa entidade do modelo pretende-se utilizar a forma geométrica gerada pela aglomeração desses pontos de parada (*convex hull*) e o histórico da ocorrência de outras paradas nas proximidades;
2) Tratamento dos *Stops* que ocorrem nas *FMST* – como para o modelo *TX-Trajectory* só interessa encontrar paradas em pontos de táxi, não foi feita a identificação de *Stops* nas *FMST*. Entretanto, essa identificação pode interessar a outro domínio de aplicação. Além disso, a eliminação de *Stops* não relevantes para a aplicação pode ajudar a diminuir o volume de dados a serem processados;
3) *TX-Trajectory Generation Module* – acrescentar novos métodos para a identificação dos *Stops* nesta interface;
4) Melhorar o algoritmo para a identificação de *Stops* e *Moves* utilizados na interface ou construir um novo algoritmo voltado para trajetórias de táxi;
5) Investigar a estratégia da identificação da quebra de jornada do motorista;
6) Utilizar outras bases de dados para dar carga no modelo.

O modelo *TX-Trajectory*, constitui uma inovadora alternativa para representação, estruturação e análise de dados relacionados com a dinâmica do objeto móvel tipo táxi, viabilizando a simplificação do processo de modelagem das aplicações voltadas para a análise do deslocamento dos táxis em centros urbanos. Esta abordagem abre perspectivas para ampliação dos conhecimentos intrínsecos às trajetórias produzidas por este modal de transporte, incluindo aspectos sociais, políticos, econômicos e ecológicos implicados na temática da gestão dos sistemas de transporte, especialmente, nas grandes cidades.

REFERÊNCIAS

AKASAPU, A. K.; SHARMA, L. K.; RAMAKRISHNA, G. Efficient Trajectory Pattern. **Mining for both Sparse and Dense Dataset**. v. 9, n. 5, p. 45–48, 2010.

ALVARES, L. O.; BOGORNY, V.; KUIJPERS, B.; *et al.* A Model for Enriching Trajectories with Semantic Geographical Information. **ACM-GIS'07**, 2007.

ALVARES, L. O.; LOY, A. M.; RENSO, C.; BOGORNY, V. An algorithm to identify avoidance behavior in moving object trajectories. **Journal of the Brazilian Computer Society**, v. 17, n. 3, p. 193–203, 30 ago 2011.

AMORIM, A. M.; CAMPOS, J. A Conceptual Model for Representation of Taxi Trajectories. In: GEOINFO, 13., 2012, Campos do Jordão. **Proceedings…** 2012.

ANDRIENKO, G.; ANDRIENKO, N.; WROBEL, S. V**isual analytics tools for analysis of movement data**. [S.l.]: [s.n.], 2007.

BAZZAN, A. L. C.; KLÜGL, F. **Sistemas inteligentes de transporte e tráfego** : uma abordagem de tecnologia da informação. [S.l.]: [s.n.], 2007.

BIADGILGN, D.M.; BLOK, C.A ; HUISMAN, O. **Assessing the cartographic visualization of moving objects**. [S.l.]: [s.n.], 2011.

BOGORNY, V. et al. Weka-STPM: a software architecture and prototype for semantic trajectory data mining and visualization. **Transactions in GIS**, v. 15, n. 2, p. 227–248, 4 abr. 2011.

BOGORNY, V.; BRAZ, J. B. **Introdução a trajetórias de objetos móveis**. Joinville: Editora Univille, 2012.

BOGORNY, V.; ENGEL, P. M.; ALVARES, L. O. **GEOARM :** an Interoperable framework to improve geographic data preprocessing and spatial association rule mining. [S.l.]: [s.n.], 2006.

BOGORNY, V.; KUIJPERS, B.; ALVARES, L. O. A Spatio-temporal data mining query language for moving object trajectories. **Technical Report TR-357**, n. i, p. 1–22, 2008.

BOGORNY, V. et al Weka-GDPM: Integrating classical data mining toolkit to geographic information systems. In: SBBD WORKSHOP ON DATA MINING ALGORITHMS AND APLICATIONS (WAAMD 2006), 2006. Florianopolis. **Anais...** 2006.

BOGORNY, V. et al. CONSTAnT - a conceptual data model for semantic trajectories of moving objects. **Transactions in GIS**, n. 295179, p. n/a–n/a, 5 mar.2013.

FRANK, E. *et al.* WEKA: A Machine Learning Workbench for Data Mining. **Springer US. In:** Data Mining and Knowledge Discovery Handbook, p. 1305–14, 2005.

GE, Y. et al. A Taxi business intelligence system. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 17., 2011. **Proceedings …** 2011.

KAMAROLI, N. Q. B. et al. Analysis of taxi movement through flow mapping. **IS415 – Geospatial Analytics for Business Intelligence.**, n. apr., 2011.

LAUBE, P.; IMFELD, S.; WEIBEL, R. Discovering relative motion patterns in groups of moving point objects. **International Journal of Geographical Information Science**, v. 19, p. 639–668, 2005.

LEAL, B. D. C. *et al.* From conceptual modeling to logical representation of trajectories in DBMS-OR and DW systems. **October**, v. 2, n. 3, p. 463–478, 2011.

LI, Z. *et al.* MoveMine : mining moving object databases. **Interactions**, p. 1203–1206, 2010.

LIU, L. et al. U**ncovering taxi driver ' s mobility intelligence through his trace.** USA: SENSEable City Lab, Massachusetts Institute of Technology, 2009.

MARKETOS, G. **Mobility data warehousing and mining**. [S.l.]: [s.n.], 2009.

MORENO, B. N. et al. Looking inside the stops of trajectories of moving objects. **GeoInfo**, p. 9–20, 2010.

MORENO, F.; ARANGO, F. A conceptual trajectory multidimensional model : an application to public transportation. [S.l.]: [s.n.], 2011.

PALMA, A. T. et al. A clustering-based approach for discovering interesting places in trajectories. **Computer**, p. 863–868, 2008.

PENG, C. et al. Collective human mobility pattern from taxi trips in urban area. **PLoS ONE 7(4):e34487. doi:10.1371/journal.pone.0034487.**, v. 7, n. 4, p. 1–8, 2012.

POSTGIS. PostGIS. Último acesso em 20 de novembro de 2013. Disponível em <http://postgis.refractions.net>. 2013.

QGIS. QGIS. [Portal institucional]. Disponível em <http://www.qgis.org> Acesso em: 20 nov. 2013.

ROCHA, J. A. M. R.; TIMES, V. C.; OLIVEIRA, G.; ALVARES, L. O.; BOGORNY, V. DB-SMoT: A direction-based spatio-temporal clustering method. In: IEEE INTERNATIONAL CONFERENCE INTELLIGENT SYSTEMS, 5., 2010. **Proceedings…** 2010.

SILVA, B. C. et. al. ITSUMO : An intelligent transportation system for urban mobility. **Proceedings of the Optimization of Urban Traffic Systems**, p. 224–235, 2005.

SIQUEIRA, F. DE L.; BOGORNY, V. Discovering chasing behavior in moving object trajectories. **Transactions in GIS**, v. 15, n. 5, p. 667–688, 10 out. 2011.

SPACCAPIETRA, S. et al. A conceptual view on trajectories. **Data & Knowledge Engineering**, v. 65, p. p. 126–145, 2008.

VELOSO, M.; PHITHAKKITNUKOON, S.; BENTO, C. Urban Mobility study using taxi traces. In: INTERNATIONAL WORKSHOP ON TRAJECTORY DATA MINING AND ANALYSIS TDMA 11 (2011), 2011. **Proceedings…** 2011.

WESSEL, M.; LUTHER, M.; MÖLLER, R. **What happened to bob**? Semantic data mining of context histories. [S.l.]: [s.n.], 2009.

YAN, Z. Towards semantic trajectory data analysis : a conceptual and computational approach. **VLDB'09**, 2009.

YAN, Z. et. al. SeMiTri : a framework for semantic annotation of heterogeneous trajectories. **EDBT 2011**, p. 259–270, 2011.

YAN, Z.; C et. al. **Semantic trajectories** : mobility data computation and annotation. [S.l.]: [s.n.], 2012.

YUAN, J. et. al. Where to find my next passenger. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING - UBICOMP '11, 13., 2011. **Proceedings…**p. 109, 2011.

ZHENG, K.; ZHENG, Y.; YUAN, N. J.; SHANG, S. On discovery of gathering patterns from trajectories. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING (ICDE), 29., 2013. **Proceedings…**p. 242–253, abr. 2013.

ZHENG, Y.; et. al. Urban Computing with Taxicabs. In: INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING - UBICOMP '11, 13., 2011. **Proceedings…**p. 89–98, 2011.

APÊNDICE A – APLICAÇÃO TX-TRAJECTORY GENERATION MODULE

Para utilizar a aplicação *TX-Trajectory Generation Module*, inicialmente deverá ser informado o esquema do banco de dados utilizado, que no nosso exemplo é *public*. O BDG tem uma tabela chamada *geometry_columns* e todas as tabelas espaciais existentes nesse banco são registradas nessa tabela. Após selecionar o ***Schema*** e pressionar o botão ***Load*** serão mostradas todas as ocorrências da tabela *geometry_columns* no espaço correspondente a ***Trajectory Table*** e ***Relevant Features*** para que seja escolhida, respectivamente, a tabela com as trajetórias brutas e a tabela com as características relevantes de acordo com a aplicação, no nosso caso, os pontos de táxi. Em ***User Buffer (m)*** deverá ser informado o raio de abrangência correspondente ao ponto de táxi. Essa distância irá determinar se o cluster encontrado será ou não considerado como um ponto de táxi oficial. ***CB-SMoT*** indica o método que será utilizado para encontrar os *stops*. ***MaxDist(m)***, ***MaxSpeed(m/s)*** e ***MinTime(seconds)*** são os parâmetros utilizados pelo algoritmo *CB-SMoT*. Após pressionar ***OK***, a aplicação cria as tabelas com as entidades do modelo de acordo com as variáveis informadas.

Supondo que a tabela com as trajetórias brutas seja denominada ***trajbruta*** e as variáveis *MaxDist* (**100**), *MaxSpeed* (**0.9**) e *MinTime* (**300**), a aplicação irá criar as seguintes tabelas:

em_trajbruta_mxd_100_ms_0_9_mt_300 - tabela com os deslocamentos sem passageiro (*Empty-Move Sub-Trajectory*)
tem_trajbruta_mxd_100_ms_0_9_mt_300 - tabela com os trajeto dos deslocamentos sem passageiro
fm_trajbruta_mxd_100_ms_0_9_mt_300 - tabela com os deslocamentos com passageiro (*Full-Move Sub-Trajectory*)
tfm_trajbruta_mxd_100_ms_0_9_mt_300 - tabela com os trajeto dos deslocamentos com passageiro
tsp_trajbruta_mxd_100_ms_0_9_mt_300 - tabela com as paradas nos pontos de táxi (*Taxi Stop Point*)

Essa forma de nomear as tabelas permite que para a mesma tabela de trajetórias brutas, sejam criadas tabelas do modelo com calibragens diferentes, de

acordo com os valores informados para as variáveis usadas no *CB-SMoT*. O código desta aplicação pode ser consultado a seguir.

```
package txStpm;

import javax.swing.*;

import java.awt.Container;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridBagLayout;
import java.awt.GridBagConstraints;
import java.awt.Insets;
import java.awt.event.*;
import java.io.IOException;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.ArrayList;
import java.util.List;

public class TxSTPM extends JDialog {

    private Method[] algs;
    private javax.swing.JComboBox jComboBoxSchema;
    private javax.swing.JList jListTrajectoryTables;
    private javax.swing.JList jListRF;
    private javax.swing.JTextField jTextFieldBuffer;
    private javax.swing.JComboBox jComboBoxMethod;
    private javax.swing.JTextField jTextFieldParam;
    private javax.swing.JComboBox jComboBoxParam;
    public Connection conn;
    private Config config = new Config();
    private Double buffer = 50.0;
    private int table_srid;
    //
    Integer maxDist, minTime;
    Double maxSpeed;
    double lat_reg, long_reg, dist, vel;
    ArrayList<STPoint> taxiPoint = new ArrayList<STPoint>();
    ArrayList<TrajectoryElements> wts = new ArrayList<TrajectoryElements>();
    TrajectoryElements telems = new TrajectoryElements();
    EmptyMove em, em1;
    FullMove fm;
    TaxiStopPoint tsp;
    TaxiStopCandidate tspc;
    int veiculo;

    public TxSTPM(String user, String pass, String url) {
        this.setTitle("TX-Trajectory Generation Module");
        init();
        initComponents();
        try {
            if (user == "") {
                conn = DriverManager.getConnection(url);
            } else {
                conn = DriverManager.getConnection(url, user, pass);
            }

            ((org.postgresql.PGConnection) conn).addDataType("geometry", org.postgis.PGgeometry.class);

            loadSchemas();
```

```
            jComboBoxMethodItemStateChanged(null);
            config.conn = conn;
            config.tid = "veiculo";
            config.time = "hora";
        } catch (Exception e) {
            System.out.println(e.toString());
            JOptionPane.showMessageDialog(this, "Error in conection with DB.");
            dispose();
        }
    }

    /**Load the tables in the DB with geometry columns
     */
    private void loadSchemas() {
        try {
            Statement smnt = conn.createStatement();
            ResultSet rs = smnt.executeQuery("SELECT DISTINCT f_table_schema FROM geometry_columns");
            while (rs.next()) {
                jComboBoxSchema.addItem(rs.getString(1));
            }
        } catch (SQLException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(this, "Failed loading schemas");
        }
    }

    private void init() {
        algs = new Method[1];

        int i = 0;
        /**CB-SMOT:
         * Used to clusterize and find slow-speed periods in a trajectory.
         */
        algs[i] = new Method() {

            public String toString() {
                return "CB-SMoT";
            }
        };
        algs[0].param.add(new Parameter("MaxDist (m)", Parameter.Type.INT, new Integer(100)));
        algs[0].param.add(new Parameter("MaxSpeed (m/s)", Parameter.Type.DOUBLE, new Double(0.9))); //54m/s
        algs[0].param.add(new Parameter("MinTime (seconds)", Parameter.Type.INT, new Integer(300)));
//5minutos
        maxDist = (Integer) algs[0].param.elementAt(0).value;
        maxSpeed = (Double) algs[0].param.elementAt(1).value;
        minTime = (Integer) algs[0].param.elementAt(2).value;
    }

//------------------------------------------------------------------------------------------------------------------
// INTERFACE BEGINS
//------------------------------------------------------------------------------------------------------------------
    private void initComponents() {

        //Mounts the layout
        Container container = getContentPane();
        container.setLayout(new BorderLayout());

        //PANEL LOAD SCHEMA
        JPanel panelSchema = new JPanel();
        panelSchema.setBorder(BorderFactory.createEtchedBorder());
        JLabel schemaLabel = new JLabel("Schema:");
        panelSchema.add(schemaLabel, BorderLayout.CENTER);
        jComboBoxSchema = new JComboBox();
        panelSchema.add(jComboBoxSchema, BorderLayout.CENTER);

        JButton Load = new javax.swing.JButton("Load");
        Load.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        LoadActionPerformed(evt);
    }
});
panelSchema.add(Load);

JButton configure = new javax.swing.JButton("Configure Trajectory Table");
panelSchema.add(configure);

JButton filter = new javax.swing.JButton("Trajectory cleaning");
panelSchema.add(filter);

container.add(panelSchema, BorderLayout.NORTH);

//PANEL CONTENT
JPanel panelContent = new JPanel();
panelContent.setBorder(BorderFactory.createEtchedBorder());

GridBagLayout gridbag = new GridBagLayout();
panelContent.setLayout(gridbag);
GridBagConstraints c = new GridBagConstraints();

c.fill = GridBagConstraints.BOTH;
c.insets = new Insets(5, 10, 5, 10);

//Config of trajectory Table
c.gridx = 0;
c.gridy = 0;
JLabel jLabel1 = new javax.swing.JLabel("Trajectory Table: ");
panelContent.add(jLabel1, c);

c.gridx = 2;
c.gridwidth = 2;
c.weightx = 1.0;
JScrollPane sc1 = new javax.swing.JScrollPane();
DefaultListModel modelsc = new DefaultListModel();
jListTrajectoryTables = new JList(modelsc);
jListTrajectoryTables.setVisibleRowCount(2);
jListTrajectoryTables.setFixedCellWidth(2);
sc1.setViewportView(jListTrajectoryTables);
panelContent.add(sc1, c);

//Relevant Features
c.gridx = 0;
c.gridy = 1;
c.gridheight = 2;
JLabel jLabel2 = new javax.swing.JLabel();
jLabel2.setText("Relevant Features");
panelContent.add(jLabel2, c);

c.gridy = 3;
c.gridwidth = 3;
c.weightx = 1.0;
JScrollPane jScrollPane1 = new javax.swing.JScrollPane();
DefaultListModel modelRF = new DefaultListModel();
jListRF = new JList(modelRF);
jListRF.setVisibleRowCount(4);
jListRF.setFixedCellWidth(4);
jScrollPane1.setViewportView(jListRF);
panelContent.add(jScrollPane1, c);
validate();

//Buffer
JPanel bufPanel = new JPanel();
bufPanel.setBorder(BorderFactory.createEtchedBorder());
GridBagLayout gbag = new GridBagLayout();
```

```
bufPanel.setLayout(gbag);
GridBagConstraints c2 = new GridBagConstraints();
c2.insets = new Insets(5, 5, 5, 5);

c2.gridx = 0;
c2.gridy = 0;
JLabel jLabel3 = new javax.swing.JLabel("User Buffer (m):");
bufPanel.add(jLabel3, c2);

c2.gridy = 2;
c2.gridheight = 2;
jTextFieldBuffer = new JTextField();
jTextFieldBuffer.setPreferredSize(new Dimension(60, 20));
jTextFieldBuffer.setText("100.0");
bufPanel.add(jTextFieldBuffer, c2);

c.gridx = 3;
c.gridy = 3;//2;
c.gridheight = 2;
c.gridwidth = 2;
panelContent.add(bufPanel, c);

//Method Panel
JPanel panelMethod = new JPanel();
panelMethod.setBorder(BorderFactory.createTitledBorder("Method"));
gbag = new GridBagLayout();
panelMethod.setLayout(gbag);
c2 = new GridBagConstraints();
c2.insets = new Insets(3, 3, 3, 3);

c2.gridx = 0;
c2.gridy = 0;
c2.gridwidth = 6;
jComboBoxMethod = new JComboBox(algs);
jComboBoxMethod.setPreferredSize(new Dimension(210, 20));
jComboBoxMethod.addItemListener(new java.awt.event.ItemListener() {

    public void itemStateChanged(java.awt.event.ItemEvent evt) {
        jComboBoxMethodItemStateChanged(evt);
    }
});
panelMethod.add(jComboBoxMethod, c2);

c2.gridx = 0;
c2.gridy = 2;
c2.gridwidth = 3;
JLabel jLabel4 = new javax.swing.JLabel("Parameter: ");
panelMethod.add(jLabel4, c2);

c2.gridx = 3;
JLabel jLabel5 = new javax.swing.JLabel("Value: ");
panelMethod.add(jLabel5, c2);

c2.gridx = 3;
c2.gridy = 3;
jTextFieldParam = new JTextField();
jTextFieldParam.setPreferredSize(new Dimension(40, 20));
jTextFieldParam.addFocusListener(new java.awt.event.FocusAdapter() {

    public void focusLost(java.awt.event.FocusEvent evt) {
        jTextFieldParamFocusLost(evt);
    }
});
panelMethod.add(jTextFieldParam, c2);

c2.gridx = 0;
c2.gridy = 3;
```

```
    jComboBoxParam = new JComboBox();
    jComboBoxParam.setPreferredSize(new Dimension(160, 20));
    jComboBoxParam.addItemListener(new java.awt.event.ItemListener() {

        public void itemStateChanged(java.awt.event.ItemEvent evt) {
            jComboBoxParamItemStateChanged(evt);
        }
    });
    panelMethod.add(jComboBoxParam, c2);

    c.gridx = 5;
    c.gridy = 3;
    panelContent.add(panelMethod, c);
    container.add(panelContent, BorderLayout.CENTER);

    JPanel panelDown = new JPanel();
    panelDown.setBorder(BorderFactory.createEtchedBorder());

    gridbag = new GridBagLayout();
    panelDown.setLayout(gridbag);
    c = new GridBagConstraints();

    c.fill = GridBagConstraints.BOTH;
    c.insets = new Insets(10, 10, 10, 10);

    JButton jButtonOK = new javax.swing.JButton("OK");
    jButtonOK.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            try {
                OKActionPerformed(evt);
            } catch (IOException e) {
                e.printStackTrace();
                System.out.println(e.getMessage());
            } catch (SQLException e) {
                e.printStackTrace();
                System.out.println(e.getMessage());
            }
        }
    });

    c.gridx = 3;
    c.gridy = 0;
    c.gridwidth = 2;
    panelDown.add(jButtonOK, c);

    JButton jButtonCancel = new javax.swing.JButton("Close");
    jButtonCancel.addActionListener(new java.awt.event.ActionListener() {

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButtonCancelActionPerformed(evt);
        }
    });
    c.gridx = 5;
    c.gridy = 0;
    c.gridwidth = 2;
    panelDown.add(jButtonCancel, c);

    container.add(panelDown, BorderLayout.SOUTH);

    setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    this.pack();
    jComboBoxSchema.requestFocusInWindow();
    this.setMaximumSize(new Dimension(600, 360));
    this.setSize(680, 360);
    this.setVisible(true);
}
```

```
    private void LoadActionPerformed(ActionEvent evt) {
        try { //load the tables in a list of auxiliary strings
            Statement s = conn.createStatement();
            ResultSet vTableName = s.executeQuery("SELECT f_table_name as tableName,type "
                    + "FROM geometry_columns "
                    + "WHERE f_table_schema=trim('" + (String) jComboBoxSchema.getSelectedItem() + "') "
                    + "ORDER BY tableName");
            DefaultListModel model = (DefaultListModel) jListTrajectoryTables.getModel();
            model.removeAllElements();
            DefaultListModel model2 = (DefaultListModel) jListRF.getModel();
            model2.removeAllElements();
            while (vTableName.next()) {
                model2.addElement(new AssociatedParameter(vTableName.getString("tableName"),
vTableName.getString("type")));
                model.addElement(new String(vTableName.getString(1)));
            }
        } catch (Exception vErro) {
            vErro.printStackTrace();
        }

    }

    private void jComboBoxMethodItemStateChanged(java.awt.event.ItemEvent evt) {
        Method alg = (Method) jComboBoxMethod.getSelectedItem();
        jComboBoxParam.removeAllItems();
        for (int i = 0; i < alg.param.size(); i++) {
            jComboBoxParam.addItem(alg.param.elementAt(i));
        }

        jComboBoxParam.setEnabled(true);
        jTextFieldParam.setEnabled(true);
    }

    @SuppressWarnings("static-access")
    private void jTextFieldParamFocusLost(java.awt.event.FocusEvent evt) {
        Parameter p = (Parameter) jComboBoxParam.getSelectedItem();
        try {
            if (p.type.DOUBLE == p.type) {
                p.value = new Double(jTextFieldParam.getText());
            } else {
                p.value = new Integer(jTextFieldParam.getText());
            }
        } catch (NumberFormatException e) {
            JOptionPane.showMessageDialog(this, "Parameter value invalid!");
        }
    }

    @SuppressWarnings("static-access")
    private void jComboBoxParamItemStateChanged(java.awt.event.ItemEvent evt) {
        Parameter p = (Parameter) evt.getItem();
        if (p.type == p.type.DOUBLE) {
            jTextFieldParam.setText(((Double) p.value).toString());
        } else {
            jTextFieldParam.setText(((Integer) p.value).toString());
        }
    }

    private void OKActionPerformed(ActionEvent evt) throws IOException, SQLException {
        maxDist = (Integer) algs[0].param.elementAt(0).value;
        maxSpeed = (Double) algs[0].param.elementAt(1).value;
        minTime = (Integer) algs[0].param.elementAt(2).value;
        if (checkBufferState()) {
            System.out.println("Buffer of " + buffer + " saved.");
        } else {
            JOptionPane.showMessageDialog(this, "Buffer expects a number.");
            return;
```

```
        }

        if (jListRF.getSelectedIndex() == -1
                && jComboBoxMethod.getSelectedItem().toString().compareTo("CB-SMoT") != 0) {
            JOptionPane.showMessageDialog(this, "Select one or more relevant features.");
            return;
        }

        if (jListTrajectoryTables.getSelectedIndex() == -1) {
            JOptionPane.showMessageDialog(this, "Select one or more trajectory table.");
            return;
        }

        Object[] objs = jListTrajectoryTables.getSelectedValues();
        String[] str = new String[objs.length];
        for (int i = 0; i < objs.length; i++) {
            str[i] = (String) objs[i];
        }

        //controls if SRID of RFs are different from trajectories...
        // ALL the trajectories should have the SAME srid
        // it is checked ahead in the foreach.
        config.table = str[0];
        String error = checkSRIDs();
        if (error.compareTo("") != 0) {
            JOptionPane.showMessageDialog(this, error);
            return;
        }
        for (int count = 0; count < str.length;) {
            java.util.Date tempo, fim, ini = new java.util.Date();
            config.table = str[count];
            try {
                //trajectory srid has to be the same of all the other trajectory-tables
                Statement sn = conn.createStatement();
                ResultSet rsn = sn.executeQuery("select srid from geometry_columns where f_table_name='" +
config.table + "'");
                rsn.next();
                if (table_srid != rsn.getInt("srid")) {
                    throw new Exception("SRID imcompatibles. Trajectory table " + config.table + " should be changed.");
                }
                //enf of srid checking
                System.out.println("Creating tables...");
                createTables();
                System.out.println("Processing the trajectories...");

                loadTrajectories(config.table);
                fim = new java.util.Date();
                tempo = new java.util.Date(fim.getTime() - ini.getTime());
                count++;
                if (count == str.length) {
                    System.out.println("Processing time: " + tempo.getTime() + " ms");
                    JOptionPane.showMessageDialog(this, "Operation finished succesfully.");
                }

            } catch (Exception e) {
                e.printStackTrace();
                JOptionPane.showMessageDialog(this, "Error during operation");
                System.out.println("Error: \n" + e.getMessage());
            } finally {
            }
        }
        Runtime.getRuntime().gc();
    }

    private void jButtonCancelActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            conn.close();
```

```
    } catch (SQLException sqle) {
        System.out.println("Erro na conexão ao Bando de Dados : "
                + sqle.getMessage());
    } finally {
        this.dispose();
    }
}

private boolean checkBufferState() {
    try {
        buffer = Double.valueOf(jTextFieldBuffer.getText());
        return true;

    } catch (NumberFormatException e) {
        jTextFieldBuffer.setText("50.0");
        return false;
    }
}

private String checkSRIDs() {  //ok

    Statement sn;
    try {
        //getting trajectory SRID
        sn = conn.createStatement();
        String sql1 = "SELECT srid FROM geometry_columns WHERE f_table_name='" + config.table + "'";
        ResultSet rsn = sn.executeQuery(sql1);
        rsn.next();
        table_srid = rsn.getInt("srid");
        //getting all the RFs
        Object[] objs = jListRF.getSelectedValues();
        AssociatedParameter[] relevantFeatures = new AssociatedParameter[objs.length];
        for (int i = 0; i < objs.length; i++) {
            relevantFeatures[i] = (AssociatedParameter) objs[i];
        }
        //comparing their SRIDs with the trajectory
        for (AssociatedParameter a : relevantFeatures) {
            sn = conn.createStatement();
            String sql2 = "SELECT srid FROM geometry_columns WHERE f_table_name='" + a.name + "'";
            rsn = sn.executeQuery(sql2);
            rsn.next();
            if (table_srid != rsn.getInt("srid")) {
                return "Error in the SRID of table: " + a.name;
            }
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
    return "";
}

private void createTables() throws SQLException {
    Statement s1 = conn.createStatement(); //WK - Working Trajectory
    Statement s2 = conn.createStatement(); //FM - Full Move
    Statement s3 = conn.createStatement(); //EM - Empty Move
    Statement s4 = conn.createStatement(); //TSP - Taxi Stop Point
    Statement s5 = conn.createStatement(); //TFM - Trajeto FM
    Statement s6 = conn.createStatement(); //TEM - Trajeto EM

    String nmTableWK = nameTableWK(config.table);
    String nmTableFM = nameTableFM(config.table);
    String nmTableEM = nameTableEM(config.table);
    String nmTableTSP = nameTableTSP(config.table);
    String nmTableTFM = nameTableTFM(config.table);
    String nmTableTEM = nameTableTEM(config.table);
```

```
        TxSTPM.setCurrentNameTableWK(nmTableWK);
        TxSTPM.setCurrentNameTableFM(nmTableFM);
        TxSTPM.setCurrentNameTableEM(nmTableEM);
        TxSTPM.setCurrentNameTableTSP(nmTableTSP);
        TxSTPM.setCurrentNameTableTFM(nmTableTFM);
        TxSTPM.setCurrentNameTableTEM(nmTableTEM);

        // WK - WorkingTrajectory table
        System.out.println("\t\tWK - WorkingTrajectory table...");
        try {
           s1.execute("DROP TABLE " + TxSTPM.getCurrentNameTableWK());
        } catch (SQLException ex) {
        } finally {
           StringBuffer sql = new StringBuffer("CREATE TABLE " + TxSTPM.getCurrentNameTableWK() + " ("
                 + "    codigo serial NOT NULL,"
                 + "    veiculo integer NOT NULL,"
                 + "    motorista character varying NOT NULL,"
                 + "    hora_start bigint NOT NULL,"
                 + "    hora_end bigint NOT NULL,");

           sql.append("    CONSTRAINT " + TxSTPM.getCurrentNameTableWK() + "_key PRIMARY KEY (codigo)"
                 + ") WITHOUT OIDS;");

           s1.execute(sql.toString());

        }

        // FM - FullMove table
        System.out.println("\t\tFM - FullMove table...");
        try {
           s2.execute("DROP TABLE " + TxSTPM.getCurrentNameTableFM());
           s2.execute("DELETE FROM geometry_columns WHERE f_table_name = '" +
TxSTPM.getCurrentNameTableFM() + "'");
        } catch (SQLException ex) {
        } finally {
           StringBuffer sql = new StringBuffer("CREATE TABLE " + TxSTPM.getCurrentNameTableFM() + " ("
                 + "    codigo serial NOT NULL,"
                 + "    passageiro integer NOT NULL,"
                 + "    latitude_start double precision NOT NULL,"
                 + "    longitude_start double precision NOT NULL,"
                 + "    hora_start bigint NOT NULL,"
                 + "    latitude_end double precision NOT NULL,"
                 + "    longitude_end double precision NOT NULL,"
                 + "    hora_end bigint NOT NULL,"
                 + "    chave_wt integer NOT NULL,");

           sql.append("    CONSTRAINT " + TxSTPM.getCurrentNameTableFM() + "_key PRIMARY KEY (codigo)"
                 + ") WITHOUT OIDS;");

           s2.execute(sql.toString());

           s2.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableFM() + "',
'the_geom_start'," + table_srid + ", 'POINT', 2)");
           try {
              s2.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableFM() + " DROP CONSTRAINT
enforce_geotype_the_geom_start");
           } catch (SQLException ex) {
              try {
                 s2.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableFM() + " DROP CONSTRAINT
\"$2\"");
              } catch (SQLException e) {
                 ex.printStackTrace();
              }
           }
           s2.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableFM() + "',
'the_geom_end'," + table_srid + ", 'POINT', 2)");
```

```
            try {
                s2.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableFM() + " DROP CONSTRAINT
enforce_geotype_the_geom_end");
            } catch (SQLException ex) {
                try {
                    s2.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableFM() + " DROP CONSTRAINT
\"$2\"");
                } catch (SQLException e) {
                    ex.printStackTrace();
                }
            }
        }
        // EM - EmptyMove table
        System.out.println("\t\tEM - EmptyMove table...");
        try {
            s3.execute("DROP TABLE " + TxSTPM.getCurrentNameTableEM());
            s3.execute("DELETE FROM geometry_columns WHERE f_table_name = '" +
TxSTPM.getCurrentNameTableEM() + "'");
        } catch (SQLException ex) {
        } finally {
            StringBuffer sql = new StringBuffer("CREATE TABLE " + TxSTPM.getCurrentNameTableEM() + " ("
                    + "    codigo serial NOT NULL,"
                    + "    latitude_start double precision NOT NULL,"
                    + "    longitude_start double precision NOT NULL,"
                    + "    hora_start bigint NOT NULL,"
                    + "    latitude_end double precision NOT NULL,"
                    + "    longitude_end double precision NOT NULL,"
                    + "    hora_end bigint NOT NULL,"
                    + "    chave_wt integer NOT NULL,");

            sql.append("    CONSTRAINT " + TxSTPM.getCurrentNameTableEM() + "_key PRIMARY KEY (codigo)"
                    + ") WITHOUT OIDS;");

            s3.execute(sql.toString());

            s3.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableEM() + "',
'the_geom_start'," + table_srid + ", 'POINT', 2)");
            try {
                s3.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableEM() + " DROP CONSTRAINT
enforce_geotype_the_geom_start");
            } catch (SQLException ex) {
                try {
                    s3.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableEM() + " DROP CONSTRAINT
\"$2\"");
                } catch (SQLException e) {
                    ex.printStackTrace();
                }
            }
            s3.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableEM() + "',
'the_geom_end'," + table_srid + ", 'POINT', 2)");
            try {
                s3.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableEM() + " DROP CONSTRAINT
enforce_geotype_the_geom_end");
            } catch (SQLException ex) {
                try {
                    s3.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableEM() + " DROP CONSTRAINT
\"$2\"");
                } catch (SQLException e) {
                    ex.printStackTrace();
                }
            }
        }
        // TSP - TaxiStopPoint table
        System.out.println("\t\tTSP - TaxiStopPoint table...");
        try {
            s4.execute("DROP TABLE " + TxSTPM.getCurrentNameTableTSP());
```

```
            s4.execute("DELETE FROM geometry_columns WHERE f_table_name = '" +
TxSTPM.getCurrentNameTableTSP() + "'");
        } catch (SQLException ex) {
        } finally {
            StringBuffer sql = new StringBuffer("CREATE TABLE " + TxSTPM.getCurrentNameTableTSP() + " ("
                    + "    codigo serial NOT NULL,"
                    + "    oficial boolean NOT NULL,"
                    + "    latitude_start double precision NOT NULL,"
                    + "    longitude_start double precision NOT NULL,"
                    + "    hora_start bigint NOT NULL,"
                    + "    latitude_end double precision NOT NULL,"
                    + "    longitude_end double precision NOT NULL,"
                    + "    hora_end bigint NOT NULL,"
                    + "    chave_wt integer NOT NULL,");

            sql.append("    CONSTRAINT " + TxSTPM.getCurrentNameTableTSP() + "_key PRIMARY KEY (codigo)"
                    + ") WITHOUT OIDS;");

            s4.execute(sql.toString());

            s4.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableTSP() + "',
'the_geom_start'," + table_srid + ", 'POINT', 2)");
            try {
                s4.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTSP() + " DROP CONSTRAINT
enforce_geotype_the_geom_start");
            } catch (SQLException ex) {
                try {
                    s4.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTSP() + " DROP CONSTRAINT
\"$2\"");
                } catch (SQLException e) {
                    ex.printStackTrace();
                }
            }
            s4.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableTSP() + "',
'the_geom_end'," + table_srid + ", 'POINT', 2)");
            try {
                s4.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTSP() + " DROP CONSTRAINT
enforce_geotype_the_geom_end");
            } catch (SQLException ex) {
                try {
                    s4.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTSP() + " DROP CONSTRAINT
\"$2\"");
                } catch (SQLException e) {
                    ex.printStackTrace();
                }
            }
        }

        // TFM - TrajetoFullMove table
        System.out.println("\t\tTFM - TrajetoFullMove table...");
        try {
            s5.execute("DROP TABLE " + TxSTPM.getCurrentNameTableTFM());
            s5.execute("DELETE FROM geometry_columns WHERE f_table_name = '" +
TxSTPM.getCurrentNameTableTFM() + "'");
        } catch (SQLException ex) {
        } finally {
            StringBuffer sql = new StringBuffer("CREATE TABLE " + TxSTPM.getCurrentNameTableTFM() + " ("
                    + "    codigo serial NOT NULL,"
                    + "    latitude double precision NOT NULL,"
                    + "    longitude double precision NOT NULL,"
                    + "    hora bigint NOT NULL,"
                    + "    chave_wt integer NOT NULL,"
                    + "    chave_fm integer NOT NULL,");

            sql.append("    CONSTRAINT " + TxSTPM.getCurrentNameTableTFM() + "_key PRIMARY KEY (codigo)"
                    + ") WITHOUT OIDS;");
```

```
            s5.execute(sql.toString());

            s5.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableTFM() + "', 'the_geom'," +
table_srid + ", 'POINT', 2)");
            try {
               s5.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTFM() + " DROP CONSTRAINT
enforce_geotype_the_geom");
            } catch (SQLException ex) {
               try {
                  s5.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTFM() + " DROP CONSTRAINT
\"$2\"");
               } catch (SQLException e) {
                  ex.printStackTrace();
               }
            }
         }
         // TEM - TrajetoEmptyMove table
         System.out.println("\t\tTEM - TrajetoEmptyMove table...");
         try {
            s6.execute("DROP TABLE " + TxSTPM.getCurrentNameTableTEM());
            s6.execute("DELETE FROM geometry_columns WHERE f_table_name = '" +
TxSTPM.getCurrentNameTableTEM() + "'");
         } catch (SQLException ex) {
         } finally {
            StringBuffer sql = new StringBuffer("CREATE TABLE " + TxSTPM.getCurrentNameTableTEM() + " ("
                  + "    codigo serial NOT NULL,"
                  + "    latitude double precision NOT NULL,"
                  + "    longitude double precision NOT NULL,"
                  + "    hora bigint NOT NULL,"
                  + "    chave_wt integer NOT NULL,"
                  + "    chave_em integer NOT NULL,");

            sql.append("    CONSTRAINT " + TxSTPM.getCurrentNameTableTEM() + "_key PRIMARY KEY (codigo)"
                  + ") WITHOUT OIDS;");

            s6.execute(sql.toString());

            s6.execute("SELECT AddGeometryColumn('" + TxSTPM.getCurrentNameTableTEM() + "', 'the_geom'," +
table_srid + ", 'POINT', 2)");
            try {
               s6.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTEM() + " DROP CONSTRAINT
enforce_geotype_the_geom");
            } catch (SQLException ex) {
               try {
                  s6.execute("ALTER TABLE " + TxSTPM.getCurrentNameTableTEM() + " DROP CONSTRAINT
\"$2\"");
               } catch (SQLException e) {
                  ex.printStackTrace();
               }
            }
         }

      }

      public String formatNameParameter(String nm) {
         System.out.println("Parm = " + nm);
         if (nm.trim().equalsIgnoreCase("MaxAvgSpeed")) {
            return "as";
         } else if (nm.trim().equalsIgnoreCase("MinAvgSpeed")) {
            return "as";
         } else if (nm.trim().equalsIgnoreCase("MinTime (seconds)")) {
            return "mt";
         } else if (nm.trim().equalsIgnoreCase("MaxSpeed (m/s)")) {
            return "ms";
         } else if (nm.trim().equalsIgnoreCase("MinDirChange (degrees)")) {
            return "md";
         } else if (nm.trim().equalsIgnoreCase("MaxTolerance (points)")) {
```

```
            return "mt";
        } else if (nm.trim().equalsIgnoreCase("MinTimeVar (seconds)")) {
            return "mtv";
        } else if (nm.trim().equalsIgnoreCase("MinTimeSpeed (seconds)")) {
            return "mts";
        } else if (nm.trim().equalsIgnoreCase("MaxDist (m)")) {
            return "mxd";
        }
        return nm;
    }

    private void loadTrajectories(String tableTraj) throws SQLException, IOException {
        loadTaxiPoint();
        Method method = (Method) jComboBoxMethod.getSelectedItem();
        Statement s = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
        // selects the trajectory-tid to be processed
        String sql = "SELECT " + config.tid + " as tid, count(*) FROM " + config.table + " GROUP BY " + config.tid;
        ResultSet rs = s.executeQuery(sql);
        //for each table trajectory...
        while (rs.next()) {
            veiculo = rs.getInt("tid");
            String meth = method.toString();
            // CB-SMOT
            if (meth.startsWith("CB-SMoT")) {
                //select the points of the trajectory in sequential time
                loadTelems(tableTraj, veiculo);
                //identificar trechos da EmptyMove sem deslocamento e considerar como TSPC ==> CB-SMoT
                procuraStops();
                //identificar EM/FM com apenas 1 ponto e excluir
                limpezaEMFM();
                //identificar EM/EM e juntar em 1 só
                agregaEM();
                //identificar TSPC/TSPC e juntar em 1 só
                agregaTSPC();
                //identificar os verdadeiros TSP
                //calcular a distância dos TSPC até os pontos de táxi; se < 100 é TSP
                loadTSP();
                //gerar as WKs
                geraWK();
            }
        }

    }
//-----------------------------------------------------------------------------------------------------------------
// INTERFACE ENDS
//-----------------------------------------------------------------------------------------------------------------
    private static String currentNameTableWK;
    private static String currentNameTableFM;
    private static String currentNameTableEM;
    private static String currentNameTableTSP;
    private static String currentNameTableTFM;
    private static String currentNameTableTEM;

    public String nameTableWK(String wk) {
        return "wk_".concat(wk.concat(parametersClusterStr()));
    }

    public String nameTableFM(String fm) {
        return "fm_".concat(fm.concat(parametersClusterStr()));
    }

    public String nameTableEM(String em) {
        return "em_".concat(em.concat(parametersClusterStr()));
    }

    public String nameTableTSP(String tsp) {
```

```
        return "tsp_".concat(tsp.concat(parametersClusterStr()));
    }

    public String nameTableTFM(String tfm) {
        return "tfm_".concat(tfm.concat(parametersClusterStr()));
    }

    public String nameTableTEM(String tem) {
        return "tem_".concat(tem.concat(parametersClusterStr()));
    }

    public String parametersClusterStr() {
        StringBuilder str = new StringBuilder();
        for (Parameter param : parametersCluster()) {
            if (param.name != null && !param.name.equals("") && param.value != null) {

str.append("_").append(formatNameParameter(param.name)).append("_").append(param.value.toString());
            }
        }
        String str1 = str.toString().toLowerCase().replace(".", "_").replace(",", "_").replace(" ", "").replace("(degrees)",
"").replace("(seconds)", "").replace("(points)", "");
        return str1;
    }

    public List<Parameter> parametersCluster() {
        List<Parameter> list = new ArrayList<Parameter>();
        int tamParans = 0;
        while (tamParans < this.jComboBoxParam.getModel().getSize()) {
            Parameter param = (Parameter) this.jComboBoxParam.getModel().getElementAt(tamParans);
            list.add(param);
            tamParans++;
        }
        return list;
    }

    private static void setCurrentNameTableWK(String currentNameTableWK) {
        TxSTPM.currentNameTableWK = currentNameTableWK;
    }

    private static void setCurrentNameTableFM(String currentNameTableFM) {
        TxSTPM.currentNameTableFM = currentNameTableFM;
    }

    private static void setCurrentNameTableEM(String currentNameTableEM) {
        TxSTPM.currentNameTableEM = currentNameTableEM;
    }

    private static void setCurrentNameTableTSP(String currentNameTableTSP) {
        TxSTPM.currentNameTableTSP = currentNameTableTSP;
    }

    private static void setCurrentNameTableTFM(String currentNameTableTFM) {
        TxSTPM.currentNameTableTFM = currentNameTableTFM;
    }

    private static void setCurrentNameTableTEM(String currentNameTableTEM) {
        TxSTPM.currentNameTableTEM = currentNameTableTEM;
    }

    public static String getCurrentNameTableWK() {
        if (currentNameTableWK == null || currentNameTableWK.equals("")) {
            return "wk";
        }
        return currentNameTableWK;
    }

    public static String getCurrentNameTableFM() {
```

```
    if (currentNameTableFM == null || currentNameTableFM.equals("")) {
        return "fm";
    }
    return currentNameTableFM;
}

public static String getCurrentNameTableEM() {
    if (currentNameTableEM == null || currentNameTableEM.equals("")) {
        return "em";
    }
    return currentNameTableEM;
}

public static String getCurrentNameTableTSP() {
    if (currentNameTableTSP == null || currentNameTableTSP.equals("")) {
        return "tsp";
    }
    return currentNameTableTSP;
}

public static String getCurrentNameTableTFM() {
    if (currentNameTableTFM == null || currentNameTableTFM.equals("")) {
        return "tfm";
    }
    return currentNameTableTFM;
}

public static String getCurrentNameTableTEM() {
    if (currentNameTableTEM == null || currentNameTableTEM.equals("")) {
        return "tem";
    }
    return currentNameTableTEM;
}

public void loadTaxiPoint() {
    STPoint ponto;
    Statement sn;
    try {
        //getting all the RFs
        Object[] objs = jListRF.getSelectedValues();
        AssociatedParameter[] relevantFeatures = new AssociatedParameter[objs.length];
        for (int i = 0; i < objs.length; i++) {
            relevantFeatures[i] = (AssociatedParameter) objs[i];
        }
        //select nas tab de RFs
        for (AssociatedParameter a : relevantFeatures) {
            sn = conn.createStatement();
            String sql2 = "SELECT y, x, the_geom FROM " + a.name;
            ResultSet rsn = sn.executeQuery(sql2);
            rsn = sn.executeQuery(sql2);
            while (rsn.next()) { // para cada ponto de taxi
                lat_reg = rsn.getDouble("y");
                long_reg = rsn.getDouble("x");
                org.postgis.PGgeometry geom_reg = (org.postgis.PGgeometry) rsn.getObject("the_geom");
                ponto = new STPoint(lat_reg, long_reg, 0, geom_reg);
                taxiPoint.add(ponto);
            }
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }

}

public void loadTSP() {
    boolean achou;
```

```
        for (int i = 0; i < (telems.size()); i++) {
            TrajectoryElement te = (TrajectoryElement) telems.get(i);
            if (te instanceof TaxiStopCandidate) {
                achou = false;
                for (int j = 0; j < (taxiPoint.size()); j++) {
                    dist = telems.get(i).getStart().distance(taxiPoint.get(j));
                    if (dist < buffer) {
                        achou = true;
                    }
                }
                tsp = new TaxiStopPoint();
                tsp.setStart(te.getStart());
                tsp.setEnd(te.getEnd());
                if (!(achou)) { //TSPC não é um ponto de táxi
                    tsp.setOficial(false);
                } else {
                    tsp.setOficial(true);
                }
                telems.insere(tsp, i);
                telems.exclui(i + 1);
            }
        }

    }

    public void loadTelems(String tableTraj, int veiculo) throws SQLException {
        Statement s1 = conn.createStatement();
        String sql2 = "SELECT " + config.time + " as time,latitude,longitude,status,the_geom,gid FROM " + tableTraj
+ " WHERE " + config.tid + "=" + veiculo + " ORDER BY " + config.time + " DESC ";
        ResultSet rs2 = s1.executeQuery(sql2);
        boolean inicio = true;
        int xav = -1;
        int status_reg;
        long hora_reg;
        while (rs2.next()) { // para cada ponto da trajetoria ordenado pelo tempo
            lat_reg = rs2.getDouble("latitude");
            long_reg = rs2.getDouble("longitude");
            status_reg = rs2.getInt("status");
            hora_reg = rs2.getLong("time");
            org.postgis.PGgeometry geom_reg = (org.postgis.PGgeometry) rs2.getObject("the_geom");
            if (inicio) {
                telems = new TrajectoryElements();
                telems.setDriver(1);
                telems.setVehicle(1);
                telems.setStart(hora_reg);
                telems.setEnd(hora_reg);
                telems.setGeom(geom_reg);
            }
            if (status_reg == 0) {  //é uma informação de EmptyMove
                if (xav == 0) { //continuação da EmptyMove
                    em.setCourse(lat_reg, long_reg, hora_reg, geom_reg);
                    em.setStart(lat_reg, long_reg, hora_reg, geom_reg);
                    telems.setStart(hora_reg);
                } else {  //nova EmptyMove
                    if (xav == 1) { //finalizou uma FullMove
                        telems.insere(fm);
                    }
                    em = new EmptyMove();
                    em.setStart(lat_reg, long_reg, hora_reg, geom_reg);
                    em.setEnd(lat_reg, long_reg, hora_reg, geom_reg);
                    em.setCourse(lat_reg, long_reg, hora_reg, geom_reg);
                    xav = 0;
                }
                telems.setStart(hora_reg);
            }
            if (status_reg == 1) { //é uma informação de FullMove
                if (xav == 1) {  //continuação da FullMove
```

```
                fm.setCourse(lat_reg, long_reg, hora_reg, geom_reg);
                fm.setStart(lat_reg, long_reg, hora_reg, geom_reg);
                telems.setStart(hora_reg);
            } else {  //nova FullMove
                if (xav == 0) { //finalizou uma EmptyMove
                    telems.insere(em);
                }
                fm = new FullMove();
                fm.setPassenger(1);
                fm.setStart(lat_reg, long_reg, hora_reg, geom_reg);
                fm.setEnd(lat_reg, long_reg, hora_reg, geom_reg);
                fm.setCourse(lat_reg, long_reg, hora_reg, geom_reg);
                xav = 1;
            }
            telems.setStart(hora_reg);
        }
        inicio = false;
    }
    if (xav == 0) {
        telems.insere(em);
    } else {
        telems.insere(fm);
    }
}

public void procuraStops() {
    //identificar trechos da EmptyMove sem deslocamento e considerar como TSPC ==> CB-SMoT
    ArrayList neighborhood = new ArrayList();
    ArrayList neighborhoodMaior = new ArrayList();
    STLine trecho;
    int qtd = 0;
    for (int n = 0; n < (telems.size()); n++) {
        TrajectoryElement te = (TrajectoryElement) telems.get(n);
        if (te instanceof EmptyMove) {
            em = (EmptyMove) te;
            if (em.size() > 1) { //encontrar Eps-linear-neighborhood de p ==> num.max de pontos da EM que distem
delta-d de p (delta-d = 100m)
                qtd = 0;
                for (int i = 0; i < (em.size() - 1); i++) {
                    neighborhood.add(i);
                    for (int j = i - 1; j >= 0; j--) { // 0..i-1
                        dist = em.getCourse().getElement(i).distance(em.getCourse().getElement(j));
                        if (dist <= maxDist) { //distancia maxima entre os pontos do cluster
                            vel = em.getCourse().getElement(j).velocidade(dist, em.getCourse().getElement(i));
                            if (vel <= maxSpeed) {
                                neighborhood.add(j);
                            } else {
                                break;
                            }
                        }
                    }
                    for (int j = i + 1; j < em.size(); j++) { // i+1..(em.size()-1)
                        dist = em.getCourse().getElement(i).distance(em.getCourse().getElement(j));
                        if (dist <= maxDist) {//100) {  //distancia maxima entre os pontos do cluster
                            vel = em.getCourse().getElement(i).velocidade(dist, em.getCourse().getElement(j));
                            if (vel <= maxSpeed) {
                                neighborhood.add(j);
                            } else {
                                break;
                            }
                        }
                    }
                    if (neighborhood.size() > 1 && (neighborhood.size() > neighborhoodMaior.size())) {
                        neighborhoodMaior.clear();
                        for (int k = 0; k < neighborhood.size(); k++) {
                            neighborhoodMaior.add(neighborhood.get(k));
                        }
```

```
                    }
                    neighborhood.clear();
                }
            } //fim encontrar Eps-linear-neighborhood de p
            //verificar se point é um core point
            if (neighborhoodMaior.size() > 0) {
                int first = (Integer) neighborhoodMaior.get(0);
                int last = (Integer) neighborhoodMaior.get((neighborhoodMaior.size()) - 1);
                long hora_ini = em.getCourse().getElement(first).getHora();
                long hora_fim = em.getCourse().getElement(last).getHora();
                if ((hora_fim - hora_ini) >= minTime) { // é um STOP - tempo mínimo do primeiro ao ultimo ponto do
cluster
                    //quebrar a EM em EM/TSPC(trecho contendo neighborhoodMaior)
                    if (first > 0) {
                        //0..first ==> EM
                        trecho = em.getTrecho(0, first);
                        em1 = new EmptyMove();
                        em1.setStart(em.getCourse().getElement(0));
                        em1.setEnd(em.getCourse().getElement(first));
                        em1.setCourse(trecho);
                        qtd = qtd + 1;
                        telems.insere(em1, n + qtd);
                    }
                    //first..last ==> TSPC
                    trecho = em.getTrecho(first, last);
                    em1 = new EmptyMove();
                    em1.setStart(em.getCourse().getElement(first));
                    em1.setEnd(em.getCourse().getElement(last));
                    em1.setCourse(trecho);
                    qtd = qtd + 1;
                    tspc = new TaxiStopCandidate(em1);
                    tspc.setStart(em1.getStart());
                    tspc.setEnd(em1.getEnd());
                    telems.insere(tspc, n + qtd);
                    if (last < ((em.size()) - 1)) {
                        //last..em.size()-1 ==> EM
                        trecho = em.getTrecho(last, em.size() - 1);
                        em1 = new EmptyMove();
                        em1.setStart(em.getCourse().getElement(last));
                        em1.setEnd(em.getCourse().getElement(em.size() - 1));
                        em1.setCourse(trecho);
                        qtd = qtd + 1;
                        telems.insere(em1, n + qtd);
                    } //excluir a EM original
                    telems.exclui(n);
                    n = n + (qtd - 1);
                } //fim-se do tratamento do STOP
            } //fim-se do tratamento do LNEps
            neighborhoodMaior.clear();
        } //fim-se do tratamento dessa EM
    } //end-for terminou de analisar todas as EM

}

public void limpezaEMFM() {
    //identificar EM/FM com apenas 1 ponto e excluir
    for (int n = 0; n < (telems.size()); n++) {
        TrajectoryElement te = (TrajectoryElement) telems.get(n);
        if (te instanceof EmptyMove) {
            em = (EmptyMove) te;
            if (em.size() == 1) {
                telems.exclui(n);
                n = n - 1;
            }
        } else {
            if (te instanceof FullMove) {
                fm = (FullMove) te;
```

```
                if (fm.size() == 1) {
                    telems.exclui(n);
                    n = n - 1;
                }
            }
        }
    }
}

public void agregaEM() {
    //identificar EM/EM e juntar em 1 só
    for (int i = 0; i < (telems.size() - 1); i++) {
        TrajectoryElement te1 = (TrajectoryElement) telems.get(i);
        if (te1 instanceof EmptyMove) {
            em1 = new EmptyMove();
            em1 = (EmptyMove) te1;
            if (telems.getNext(i) instanceof EmptyMove) {
                TrajectoryElement te2 = (TrajectoryElement) telems.getNext(i);
                EmptyMove em2 = new EmptyMove();
                em2 = (EmptyMove) te2;
                //atualizar ponto final da EM com o ponto final da EM seguinte;
                em1.setEnd(em2.getEnd());
                //incluir em EM os elementos da EM seguinte
                em1.setCourse(em2.getCourse());
                //exclui EM seguinte
                telems.exclui(i + 1);
                i = i - 1;
            }
        }
    }
}

public void agregaTSPC() {
    //identificar TSPC/TSPC e juntar em 1 só
    for (int i = 0; i < (telems.size() - 1); i++) {
        TrajectoryElement te1 = (TrajectoryElement) telems.get(i);
        if (te1 instanceof TaxiStopCandidate) {
            TaxiStopCandidate tsc1 = new TaxiStopCandidate();
            tsc1 = (TaxiStopCandidate) te1;
            if (telems.getNext(i) instanceof TaxiStopCandidate) {
                TrajectoryElement te2 = (TrajectoryElement) telems.getNext(i);
                TaxiStopCandidate tsc2 = new TaxiStopCandidate();
                tsc2 = (TaxiStopCandidate) te2;
                //atualizar ponto final do TSPC com o ponto final do TSPC seguinte;
                tsc1.setEnd(tsc2.getEnd());
                tsc1.setCourse(tsc2.getCourse());
                //exclui TSPC seguinte
                telems.exclui(i + 1);
                i = i - 1;
            }
        }
    }
}

public void geraWK() {
    //gerar as WKs
    String motorista = "motorista";
    TrajectoryElements telems2 = new TrajectoryElements();
    STPoint garagem = new STPoint(37.75136, -122.395, 0, null);
    long hora_inicio = 0;
    long hora_fim = 0;
    long jornada;
    int pos;
    for (int i = 0; i < (telems.size()); i++) {
        telems2 = new TrajectoryElements();
        pos = 0;
        for (int j = i; j < (telems.size()); j++) {
```

```
                TrajectoryElement te = (TrajectoryElement) telems.get(j);
                if (hora_inicio == 0) {
                    hora_inicio = te.getStart().getHora();
                }
                dist = te.getEnd().distance(garagem);
                jornada = (te.getEnd().getHora() - hora_inicio);
                if ((dist < 200) && (jornada > 14400))// garagem e jornada > 4h
                {
                    telems2.insere(te, pos);
                    hora_fim = te.getEnd().getHora();
                    pos = pos + 1;
                    //String nome = f1 + "\\" + parte1 + i + "CB-SMoT.kml";
                    //telems2.writeKML(nome);
                    telems2.setDriver(1);
                    telems2.setVehicle(veiculo);
                    telems2.setStart(hora_inicio);
                    telems2.setEnd(hora_fim);
                    wts.add(telems2);
                    PopulaBD populaBD = new PopulaBD();
                    populaBD.incluirDados(telems2, motorista, conn);
                    telems2 = new TrajectoryElements();
                    if (j < ((telems.size()) - 1)) {
                        hora_inicio = telems.getNext(j).getStart().getHora();
                    }
                    i = j;
                    j = telems.size();
                } else {
                    telems2.insere(te, pos);
                    hora_fim = te.getEnd().getHora();
                    pos = pos + 1;
                }
                if (j == telems.size() - 1) {
                    i = j;
                }
            }
        }
        if (!(telems2.size() == 0)) {
            //String nome = f1 + "\\" + parte1 + "CB-SMoT" + telems.size() + ".kml";
            //telems2.writeKML(nome);
            telems2.setDriver(1);
            telems2.setVehicle(veiculo);
            telems2.setStart(hora_inicio);
            telems2.setEnd(hora_fim);
            wts.add(telems2);
            PopulaBD populaBD = new PopulaBD();
            populaBD.incluirDados(telems2, motorista, conn);
        }

    }

    public static void main(String args[]) {
        String url = "jdbc:postgresql://localhost:54321/";
        String db = "taxi_trajectories";
        String user = "postgres";
        String pass = "postgres";
        System.out.println("antes ");
        new TxSTPM(user, pass, url + db);
        System.out.println("depois ");
    }
}
```