COPPE
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
UFRJ

ANÁLISE E PLANEJAMENTO DE ROTAS EM REDES VIÁRIAS

Marcus de Lima Braga

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientadores: Luís Henrique Maciel Kosmalski Costa
Aloysio de Castro Pinto Pedroza

Rio de Janeiro
Janeiro de 2016

ANÁLISE E PLANEJAMENTO DE ROTAS EM REDES VIÁRIAS

Marcus de Lima Braga

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Luís Henrique Maciel Kosmalski Costa, Dr.

Prof. Aloysio de Castro Pinto Pedroza, Dr.

Prof. Célio Vinicius Neves de Albuquerque, Ph.D.

Prof. Pedro Braconnot Velloso, Dr.

Prof. Igor Monteiro Moraes, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2016

Braga, Marcus de Lima

Análise e Planejamento de Rotas em Redes Viárias/Marcus de Lima Braga. – Rio de Janeiro: UFRJ/COPPE, 2016.

XIX, 126 p.: il.; 29, 7cm.

Orientadores: Luís Henrique Maciel Kosmalski Costa
Aloysio de Castro Pinto Pedroza

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2016.

Referências Bibliográficas: p. 118 – 126.

1. Algoritmos Anytime. 2. Algoritmos de Busca Informada. 3. Roteamento Veicular. 4. Redes Veiculares. I. Costa, Luís Henrique Maciel Kosmalski *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Dedico este trabalho acima de tudo a Deus por tudo que Ele é, e a Nossa Senhora do Perpétuo Socorro por seu amor e consolo. Aos meus pais, in memoriam, Cleomir e Marizete Braga. À minha querida esposa Kellem Andrezza. Aos meus queridos irmãos Cleomir e Helen. As minhas tias-mães Elcy e Vânia, ao meu tio Nonato. À minha sogra Helena. A todos os amigos que contribuíram direta e indiretamente para esta pesquisa.

Agradecimentos

Deo Optimo Maximo

A Deus pela realização de mais este sonho, por Sua ajuda na superação dos obstáculos, por sua presença sempre viva em nossas vidas e por nos ter dado Maria Santíssima como mãe, obrigado Senhor! Por todas as amizades feitas nesta Pós-Graduação e por todos aqueles que mesmo distantes estão sempre no meu coração, meus sinceros agradecimentos.

A minha família pelo exemplo, apoio, compreensão, paciência e fortaleza nos momentos mais difíceis da minha vida. Em especial aos meus exemplos de vida, *in memoriam*, Cleomir e Marizete Braga, que não mediram esforços para que nada nos faltasse.

A minha querida esposa Kellem Andrezza pelo amor, companheirismo, tolerância, consolo nos momentos difíceis e incentivos em toda caminhada.

Aos meus queridos irmãos e padrinhos Cleomir e Helen, pelo amor, amizade e por serem verdadeiros irmãos e amigos, muito obrigado por tudo.

As minhas tias-mães Elcy e Vânia, e ao meu tio Nonato, pelo incentivo, amor, amizade e generosidade de sempre.

Aos meus orientadores, Prof. Dr. Luís Henrique Maciel Kosmalski Costa e Prof. Dr. Aloysio de Castro Pinto Pedroza, que me acolheram no doutorado e que acreditaram em mim, me orientaram e motivaram, dando-me a oportunidade de ingressar neste Programa de Pós-Graduação. Tenho-os como modelos de pesquisadores que trabalham arduamente no desenvolvimento da ciência.

A todos os professores do GTA/UFRJ, Otto Duarte, Miguel Campista, Pedro Velloso, que com suas dedicações e competências, fazem deste laboratório um centro de excelência em pesquisas e inovação.

Ao meu querido amigo e irmão de mestrado e de doutorado Alyson de Jesus dos Santos, pelos momentos de alegria, tristeza, certeza e pela amizade. Obrigado por sua ajuda sempre pronta e sincera, muito obrigado.

Aos amigos que fiz no GTA, pela ajuda no meu crescimento como pessoa e profissional. Meus agradecimentos especiais ao Júnior, Margarida de Souza, Professor

JB (João Batista Neto), Fábio Vieira, Rodrigo Couto, Vitor Borges, Lucas Gomes, Dayro Hernandez, Martin Andreoni, Lyno Ferraz, Diogo Menezes, Igor Quintanilha, Leopoldo Maurício, Igor Alvarenga, Eduardo Oliveira, Victor Neto, Tatiana Sciammarella e Dianne Medeiros.

Um agradecimento em especial aos membro do L3I – Université de La Rochelle, pela acolhida durante o período que estive na França, em especial ao Professor Yacine Ghamri-Doudane, pelas valiosas contribuições nesta pesquisa.

Aos amigos Andrews Sobral e Caroline Pacheco, que tornaram nossa estadia em La Rochelle ainda mais inesquecível.

Ao Professor Marcelo Amorim pelas valiosas contribuições nesta pesquisa e na elaboração da revista, meus sinceros agradecimentos.

Aos Professores Valmir Carneiro e Daniel Figueiredo, pela gentileza e disposição em responder as minhas dúvidas durante a concepção desta pesquisa.

A Universidade Federal do Rio de Janeiro (UFRJ) e todos os professores da COPPE, na qual me orgulho por ter vivido grandiosos momentos nesta Pós-Graduação.

A Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) pela concessão da bolsa de doutorado.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela concessão da bolsa de doutorado sanduíche.

Aos Desembargadores do Tribunal de Justiça do Estado do Amazonas, em especial aos seus presidentes Des. João de Jesus Abdala Simões e Desa. Maria das Graças Pessoa Figueiredo pela licença do doutorado.

Agradeço aos professores Célio Albuquerque e Igor Moraes pela presenca na banca examinadora e pelos valiosos comentários sobre a pesquisa.

Agradeço aos funcionários do Programa de Engenharia Elétrica da COPPE/UFRJ, Maurício, Daniele, Rosa e Solange pela presteza no atendimento na secretaria do Programa.

Enfim, para todos aqueles que acreditaram no meu potencial, me incentivaram e torceram para que este sonho se concretizasse.

Meus sinceros agradecimentos!

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ANÁLISE E PLANEJAMENTO DE ROTAS EM REDES VIÁRIAS

Marcus de Lima Braga

Janeiro/2016

Orientadores: Luís Henrique Maciel Kosmalski Costa
Aloysio de Castro Pinto Pedroza

Programa: Engenharia Elétrica

A mobilidade tornou-se um grande desafio em áreas urbanas ao redor do mundo, com congestionamentos frequentes e tempo de deslocamento cada vez maior. Apesar dos recentes avanços no monitoramento da infraestrutura viária por meio dos Sistemas Inteligentes de Transporte (ITS), a dinâmica do sistema e sua instabilidade tornam difícil sua previsão e gerenciamento. Uma questão chave é como, dado o monitoramento de tráfego, um veículo decide mudar dinamicamente sua rota. Neste contexto, analisamos algoritmos da classe *anytime* para calcular a melhor rota entre dois pontos, levando em conta as condições obtidas de um sistema de monitoramento de tráfego. Caso ocorra a interrupção na computação da rota, os algoritmos *anytime* informam rapidamente uma resposta subótima. Isto é possível pela limitação da área de busca. Por outro lado, as rotas produzidas por algoritmos *anytime* são melhoradas progressivamente de acordo com o tempo disponível. Durante a fase de pesquisa, os resultados podem ser recalculados, caso o tráfego sofra alterações. Nesta pesquisa, é analisado o algoritmo *anytime* ITS-ARA* considerando redes viárias reais. Os experimentos utilizaram o Raspberry Pi como plataforma de testes, foram utilizados o fluxo de tráfego da cidade do Rio de Janeiro como medida. Foram medidos tempo de execução, memória consumida, cumprimento da rota, tempo de chegada, velocidade média e distância percorrida. Para diferentes janelas de tempo, os resultados mostram que ITS-ARA* encontra rotas alternativas, que podem ser usadas para reduzir o congestionamento de tráfego.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

ANALYSIS AND ROUTE PLANNING IN ROAD NETWORKS

Marcus de Lima Braga

January/2016

Advisors: Luís Henrique Maciel Kosmalski Costa
Aloysio de Castro Pinto Pedroza

Department: Electrical Engineering

Mobility became a major challenge in urban areas around the world, with frequent congestions and ever growing commuting time. Even with recent advances in the monitoring of road infrastructure through Intelligent Transportation Systems (ITS), the dynamics of such systems make it difficult to forecast and manage the traffic of vehicles. One key issue is how, given traffic monitoring, a vehicle decides to dynamically change its route. In this context, we analyze algorithms of the anytime class to calculate the best route between two spots, taking into account the conditions obtained from a traffic monitoring system. If their computation is interrupted, anytime algorithms inform a suboptimal response. This is made possible by limiting the search area. On the other hand, the routes produced by anytime algorithms are progressively improved according to the time available. During the search phase, the results may be recalculated if the traffic has changed. In this work, we analyze the anytime ITS-ARA* algorithm considering real road maps. We conduct experiments using Raspberry Pi as a test platform and consider the traffic flows from Rio de Janeiro. We evaluate time and memory consumption, route length, arrival time, average velocity and distance travelled. For different time windows, the results show that ITS-ARA* allows finding alternative routes, which can be used to reduce traffic congestion.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

SIGLAS

A* - A-estrela;

AG - Algoritmos Genéticos;

ARA* - *Anytime Repairing A**;

DGPS - Differential Global Positioning System;

DOM - *Document Object Model*;

GPS - *Global Positioning System*;

IA - Inteligência Artificial;

IDE - *Integrated Development Environment*;

IEEE - *Institute of Electrical and Electronic Engineers* - Instituto de Engenheiros Eletricistas e Eletrônicos;

ITS - *Intelligent Transportation System* - Sistemas de Transportes Inteligentes;

LAS - *Local Average Speed*;

RPi - *Raspberry Pi*;

RSSS - *Redes de Sensores Sem Fio*;

SAX - *Simple API for XML*;

SIG - *Sistema de Informação Geográfica*;

SoC - *System on a Chip*;

SPP - *Shortest Path Problem*;

SU - *Street Unit*;

TAS - *Total Average Speed*;

UI - *Unidade de Interseção*;

USB - *Universal Serial Bus*;

VMS - *Variable Message Signs*;

VGI - *Volunteered Geographic Information*;

XML - *eXtensible Markup Language*.

Capítulo 1

Introdução

As grandes cidades do mundo têm sofrido com o acelerado crescimento de suas frotas veiculares, desencadeando um grande desequilíbrio em suas redes viárias, sobretudo na mobilidade urbana com o aumento nos tempos de viagens. A mobilidade é a responsável pelo funcionamento e desenvolvimento nos centros urbanos, é ela que garante o abastecimento e a manutenção da economia. Do ponto de vista da física, o tráfego veicular é um sistema em desequilíbrio de partículas (veículos) que interagem e influenciam uns aos outro, causando uma instabilidade nestas partículas [6]. Falhas mecânicas ou elétricas nos veículos, obras nas vias, mudanças bruscas de faixa são fatores que colaboram para o desequilíbrio do sistema. O ser humano, motorista ou pedestre, com seu comportamento imprevisível, suscetível à fadiga, distração e a substâncias que causam o decréscimo de atenção e reflexos, é outra variável inerente ao sistema e que não pode ser ignorada [7–9].

A ampliação da rede viária e as intervenções promovidas são ações largamente utilizadas e que visam aumentar a eficiência do fluxo dos veículos. No entanto, tais medidas possuem prazo delimitado, seguramente até o próximo crescimento inevitável da demanda [10]. O monitoramento das vias por sensores e câmeras, o uso de placas eletrônicas sobre as condições de tráfego, controle semafórico para favorecer o fluxo nos horário de grande densidade de veículos, são algumas aplicações de Sistemas de Transportes Inteligentes (ITS - *Intelligent Transportation Systems*), que visam auxiliar as Companhias de Engenharia de Tráfego (CET) na administração do trânsito [3]. Contudo, todas essas iniciativas possuem o objetivo de minimizar os atrasos percebidos pelos condutores de veículos, não contemplando o controle e o gerenciamento da rede viária.

Apesar dos esforços de aumentar a utilização do transporte coletivo, os carros ainda são os maiores usuários das vias urbanas colaborando nos conflitos entre seus partícipes (motoristas, ciclistas e pedestres), além de prejuízos de ordem econômica (logística de bens), social (saúde) e ambiental (elevação de CO_2). Portanto, o trânsito tem sido um dos maiores desafios nas grandes cidades em todo o mundo, sendo o

horário do *rush* um dos responsáveis pelo aumento no nível de estresse da população e um componente de incerteza na duração das viagens.

A Tabela 1.1 apresenta o *ranking* com as cinco cidades com os maiores índices de congestionamento no mundo, esses dados foram extraídos do Relatório Anual de 2014 da empresa holandesa de GPS TomTom [5]. Um dos destaques dessa lista é a inclusão de duas cidades brasileiras, Rio de Janeiro e São Paulo, respectivamente em terceiro e quinto lugares. Porém, o que mais chama a atenção são os índices de congestionamento e demora da capital paulista serem inferiores aos da capital fluminense, cerca de 46% e 48 minutos contra os 55% e 59 minutos. Percentuais de congestionamento por turnos (manhã ou noite) e o tamanho da rede viária são outros dados expressos na referida tabela.

Classif.	Cidade	País	Congest. (%)	Manhã (%)	Noite (%)	Demora (min.)	Rede Viária(Km)
1	Moscou	Rússia	74	111	141	76	35.556
2	Istambul	Turquia	62	87	129	66	17.673
3	Rio de Janeiro	Brasil	55	84	115	59	19.434
4	Cidade México	México	54	88	89	53	52.106
5	São Paulo	Brasil	46	66	95	48	46.600

Tabela 1.1: Maiores índices de congestionamentos – GPS TomTom [5].

Encontrar os melhores caminhos, dado uma origem e um destino, tornou-se um problema cotidiano, desde o planejamento logístico de entrega de mercadorias até o planejamento de caminhos [11–16], sistemas de localização em veículos autônomos [17, 18] onde utilizam-se métodos baseados em GPS ou DGPS[1] [1, 19], bem como em sistemas de navegação ou movimentação de braço em robôs [20, 21]. A utilização do GPS nos sistemas de navegação dos automóveis e nos aplicativos dos *smartphones* passou a ser importantes ferramentas de informação para a resolução deste problema.

O uso de aplicativos para escolha de rotas baseadas nas condições de tráfego, tem sido de grande auxílio para os motoristas na fuga dos congestionamentos e na descoberta de novos trajetos. O *Waze* [22] é um aplicativo de rede social que compartilha informações relevantes ao trânsito como acidentes, bloqueios policiais, presença de radares entre outros. Caso o motorista se desvie da rota informada pelo aplicativo, o mesmo realiza o recálculo para uma nova a partir do ponto atual. Dados sobre o fluxo do tráfego são de grande importância na escolha das rotas, desde que estas informações reflitam as condições reais da rede viária no dado instante. Na Figura 1.1 é possível visualizar o sensoriamento do tráfego na ponte que liga a Ilha do Fundão (UFRJ) à Avenida Governador Carlos Lacerda (Linha Amarela). Do lado esquerdo está a representação no *Google Maps* e no lado direito a câmera da CET Fundão. As informações captadas inicialmente (Figura 1.1(a)) em vídeo

[1]O DGPS se refere à *Differential Global Positioning System* que possui uma acurácia de 10 cm.

e mapa eletrônico são coincidentes, porém no momento seguinte isso não é mais verdade (Figura 1.1(b))[2], comprometendo dessa forma a informação divulgada no aplicativo.

(a) Monitoramento às 17:19h.

(b) Monitoramento às 18:11h.

Figura 1.1: Monitoramento de trânsito – CET Fundão em 08/01/2015.

O sensoriamento do tráfego é um importante componente na busca pelo melhor caminho, porém é necessário estabelecer critérios de validade da informação captada, de modo que a mesma possa refletir as mudanças ocasionadas pelo fluxo de veículos. Entende-se que uma rota de alta qualidade, enfatize a minimização dos custos do tempo de viagem, com intuito de proporcionar o equilíbrio entre o conforto e a razoabilidade sempre que possível. Medir qualidade de uma viagem é difícil devido à preferência pessoal de cada condutor veicular e está fora do escopo deste trabalho.

O objetivo desta pesquisa consiste no estudo do planejamento de rotas para auxiliar a mobilidade nas cidades, com intuito de promover um balanceamento de carga na rede viária. Para isto, foram realizados experimentos com planejadores de rotas e

[2]Após adquirir o *Waze* em junho de 2013, o Google o integrou aos seus mapas dois meses depois [23, 24].

seus resultados foram medidos quanto à eficiência computacional destes algoritmos de acordo com: a) tempo de execução; b) memória consumida; c) quantidade de vértices pertencentes ao menor caminho; d) velocidade média; e) média do tempo de chegada; f) média da distância percorrida; g) frequência dos trechos mais utilizados por cada planejador; e h) comportamento estatístico das rotas geradas. São analisados ainda a influência da heurística nos resultados dos rotas geradas, utilizando redes viárias de duas cidades brasileiras, Manaus e Rio de Janeiro.

1.1 Motivação

A mobilidade é um bem vital para as sociedades modernas e tem sido um dos grandes desafios nas áreas urbanas em todo mundo. Mesmo com as soluções advindas do ITS para auxiliar na administração do tráfego, seu efeito prático é somente o de minimizar a percepção de atraso dos condutores, não contemplando o controle e o gerenciamento da rede viária [3].

O uso de rotas alternativas é uma solução buscada por muito condutores para escapar aos grandes congestionamentos. No entanto, não existe nenhuma garantia de sucesso que leve em conta as condições físicas das vias e nem a densidade de veículos em dado momento. Apesar dos avanços tecnológicos empregados nos sistemas de trânsito, ainda não existem técnicas de autogestão que sejam capazes de garantir o equilíbrio do sistema por meio da mobilidade dos veículos na rede viária [25].

As redes veiculares são muito úteis no controle do trânsito. Sistemas de controle de trânsito podem ser divididos em duas grandes funções: o monitoramento e o controle e prevenção de congestionamentos. As redes veiculares podem ser utilizadas para ambas as funções. As redes de comunicação podem ser utilizadas para detectar congestionamentos de forma automatizada e enviar a informação a uma central que, em seguida, divulga a informação [2]. Além disso, as redes veiculares podem também ser utilizadas para o envio da informação de volta aos motoristas que estão no trânsito, enviando sugestões de rotas alternativas. Neste quesito, de acordo com a dinâmica de mudanças das condições do trânsito, novas rotas precisam ser calculadas.

Nesta pesquisa são investigados algoritmos planejadores de rotas em redes viárias, onde são medidos seus desempenhos computacionais quanto à dinamicidade das arestas e a influência da heurística nos resultados, bem como são utilizados dados reais dos fluxos de tráfego da cidade do Rio de Janeiro. Estes algoritmos podem ser utilizados em diferentes aplicações de redes veiculares, onde os veículos são redirecionados por algum motivo, como por exemplo congestionamentos. Naturalmente, outras aplicações para o envio de informação através das redes veiculares no contexto de sistemas de transporte inteligentes ou, de forma mais ampla, cidades inteligentes,

são possíveis, como por exemplo à interação entre veículos elétricos e as redes elétricas inteligentes (*smart grid*). O planejamento de rotas consiste na descoberta do caminho mínimo com base nos estados das arestas com custo associado. O caminho é ótimo, quando a soma de seus custos de transição é mínima, a partir da origem até seu destino, em todas as possíveis combinações [26].

1.2 Definição do Problema

Assim, o problema a ser tratado nesta tese é: Estudar e analisar algoritmos de planejamento de rotas que sejam sensíveis à dinamicidade do trânsito nas redes viárias, de modo a possibilitar um balanceamento de carga nestas redes. Além de testar sua viabilidade em plataforma com características computacionais restritivas.

1.3 Objetivos e Contribuições

Esta pesquisa apresenta a modelagem de redes viárias em grafos, usando uma biblioteca de grafo dinâmico em Java e mapas das cidades extraídos do projeto colaborativo *OpenStreetMap* [27]. A modelagem visa à utilização de uma ferramenta matemática simples, porém poderosa na construção e análise de problemas e consequentemente favorecendo o seu entendimento. O trabalho foi desenvolvido em duas partes, a primeira visando à transformação de mapas viários em grafos, com intuito de proporcionar sua análise e de testar os algoritmos planejadores. Isso foi realizado usando a plataforma Eclipse possibilitando a criação de aplicações derivadas. A segunda parte refere-se à automatização do planejamento de rotas, fazendo uso de algoritmos clássicos e de busca informada que utilizam heurística em seus cálculos. Desta forma, foi realizado o estudo da influência da função heurística nos resultados, além de testes de viabilidade em uma plataforma computacional com limitações de memória e processamento. Foram utilizados mapas viários de duas cidades brasileiras. No segundo experimento, foram utilizados dados reais extraídos do tráfego dos ônibus do Rio de Janeiro. Foram realizadas duas ponderações: a) quanto à distância – cada trecho (aresta) da rede viária utilizada, corresponde ao seu valor real expresso em metros; b) tempo de percurso – os valores foram calculados de acordo com as informações do tráfego real dos ônibus.

Objetivo Geral: Testar e avaliar algoritmos de planejamento de rotas quanto à eficiência, viabilidade computacional, dinâmica e influência heurística. Busca-se ainda demonstrar a promoção do balanceamento de carga nas redes viárias de acordo com a adoção de rotas alternativas geradas pelos algoritmos *Anytime* utilizados.

Objetivos Específicos:

Os objetivos específicos são:

- Investigar a área de planejamento de rotas, analisando desde os sistemas que proveem as informações de tráfego, bem como projetos de sugestão de rotas e suas abordagens utilizadas entre outros.
- Extrair as informações dos mapas viários fornecidos pelo mapa colaborativo *OpenStreetMap*, que estão expressos em uma linguagem de marcação semelhante à *eXtensible Markup Language* (XML). Em seguida, essas informações extraídas irão alimentar um *software* que analisará as características dessas redes e testará algoritmos planejadores.
- Desenvolver uma solução que aplique o planejamento de rotas de acordo com a rede viária alvo e promova um tráfego mais balanceado em suas vias.
- Realizar uma prova de conceito em um dispositivo computacional com restrições de memória e processamento.

São esperadas as seguintes contribuições:

- Utilização de rotas alternativas para promover o balanceamento de carga nas redes viárias;
- Prova de conceito em dispositivo com restrições computacionais;
- Criação de uma ferramenta para o estudo do planejamento de rotas em redes viárias.
- Análise estatística dos algoritmos planejadores de rotas.
- Análise da influência da função heurística nos planejadores de busca informada.

1.4 Metodologia

A metodologia de desenvolvimento da pesquisa (Figura 1.2) consistiu em cumprir as etapas especificadas anteriormente, com intuito de atingir cada objetivo específico descrito na Seção 1.3. Inicialmente foi realizado um levantamento bibliográfico relacionado às tecnologias aplicadas ao planejamento de rotas, algoritmos e projetos empregados e quais técnicas utilizadas no sensoriamento de tráfego, com intuito de verificar o estado da arte dos referidos temas e, por conseguinte adquirir embasamento teórico para o desenvolvimento da pesquisa.

Em seguida, foram estudados meios de adquirir dados do mundo real (mapas viários) e transformá-los para uma abordagem matemática com intuito de se utilizar dessa ferramenta para analisar as propriedades e testar as soluções (algoritmos de planejamento). Nesta etapa foram estudados os arquivos dos mapas das cidades alvo

extraídos do projeto *OpenStreetMap* e expressos em uma linguagem intermediária semelhante ao XML. O desenvolvimento de um *software* para dar suporte aos experimentos foi criado usando a linguagem Java na IDE Eclipse e a biblioteca de grafos dinâmicos *GraphStream* [28]. O planejamento de rotas e a análise de propriedades das redes viárias são algumas das funcionalidades do referido *software*.

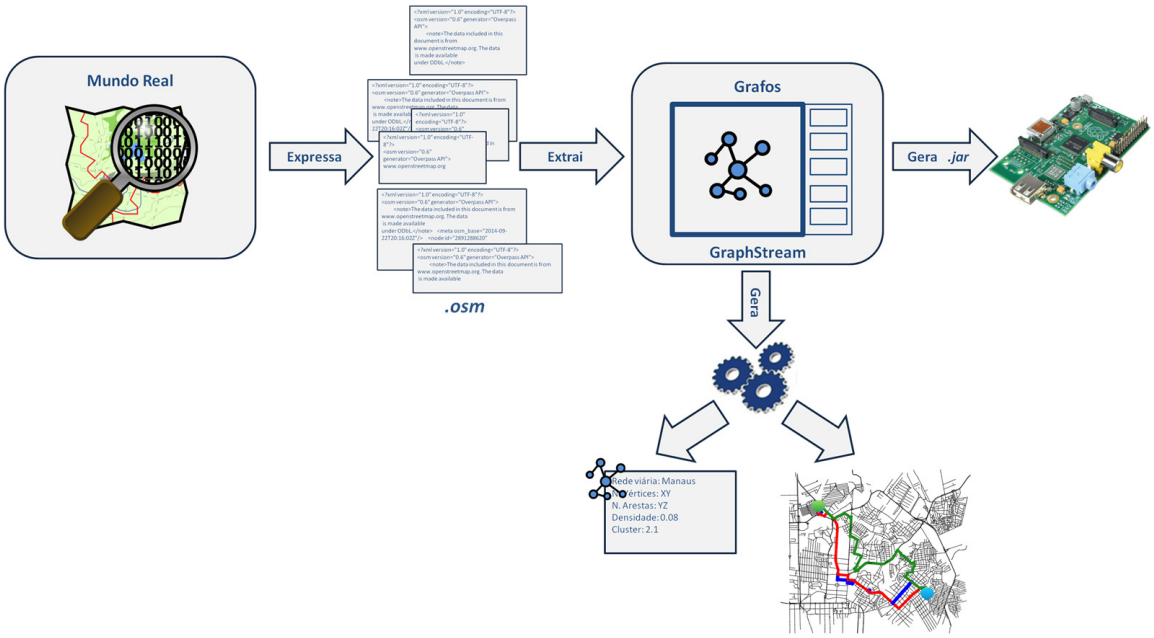


Figura 1.2: Metodologia usada na pesquisa.

Posteriormente, foi criado uma versão embarcada do *software*, com intuito de testar a viabilidade computacional em ambiente com características restritivas de processamento e memória. Foram medidos: o tempo de processamento, memória consumida, quantidade de vértices e arestas e a influência da função heurística. Para isto foi utilizado um Raspberry Pi [29, 30] modelo B com *hardware* semelhante aos *smartphones* básicos, onde os mapas de duas cidades brasileiras foram utilizadas como cenários de testes. Em outro experimento, foram usados informações reais de tráfego e dois tipos de ponderação nas arestas para os cálculos de melhor rota, além do modelo B+ referente ao Raspberry Pi. Para os experimentos finais foi utilizado um *desktop*, com mapaz de sete cidades (brasileiras, europeias e uma estadunidense), onde foi realizado um estudo da variação do coeficiente heurístico utilizado nos algoritmos da classe *Anytime* quanto à influência heurística.

1.5 Organização do Texto

Esta proposta de tese está dividida em sete capítulos. No Capítulo 2 são abordados conceitos pertinentes às áreas de Planejamento de Rotas, Redes Viárias e Teoria dos Grafos, elementos de fundamental importância para compreensão e aplicação

dos resultados obtidos.

No Capítulo 3 são descritos os trabalhos correlatos a esta pesquisa, apresentando sistemas de monitoramento de trânsito, projetos de recomendação de rotas baseadas no trânsito, além de propostas aplicadas à robótica e que estudam algoritmos de busca em mapas viários.

O *software* de planejamento de rotas desenvolvido para o suporte desta pesquisa é descrito no Capítulo 4. São apresentados o estudo do mapa viário colaborativo e a biblioteca de grafos dinâmicos utilizados na criação da referida ferramenta.

No Capítulo 5 são apresentados experimentos em uma plataforma computacional restritiva, com fins de prova de conceito. São realizados experimentos com dados reais sintéticos, utilizando duas ponderações para o cálculo de rota, além da análise da influência heurístca nos resultados e a extração das propriedades das redes viárias utilizadas.

Os estudos da variação do coeficiente heurístico quanto à influência heurística são apresentados no Capítulo 6. São utilizadas sete redes viárias como cenários, sendo que os resultados são analisados quanto aos vértices e tempo de execução em plataforma computacional sem restrições.

Por fim, no Capítulo 7 são apresentadas as principais conclusões sobre os resultados obtidos nesta pesquisa, além de sugestões para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O Capítulo apresenta uma visão geral em planejamento de rotas em redes viárias. Inicialmente é apresentado o problema do planejamento de rotas na Seção 2.1, juntamente com as áreas de atuação relacionadas. Na Seção 2.2 são apresentadas as definições de redes viárias e os algoritmos utilizados nesta pesquisa.

2.1 Problema do Planejamento de Rotas

O problema do planejamento de rotas tem recebido atenção da comunidade científica nas últimas décadas e muitas soluções eficientes têm sido propostas, muitas delas derivadas da teoria dos grafos e seus algoritmos de busca [31–36]. O problema do planejamento de rotas consiste na descoberta do melhor caminho entre dois pontos (origem e destino) com base nos estados das arestas com custo associado. O caminho é ótimo, quando a soma de seus custos de transição é mínima a partir da origem até seu destino em todas as possíveis combinações [26, 37]. O melhor caminho depende do tipo de aplicação, podendo significar o caminho mais curto ou o mais rápido, ou ainda a combinação de vários critérios ou propriedades.

2.1.1 Robótica

O planejamento de rotas é um dos módulos que compõem o sistema de navegação de um robô, sendo o responsável por determinar o trajeto a ser seguido, levando-se em consideração o ambiente em que se encontra e evitando colisões [38].

No contexto da robótica, tem-se que o planejamento de rotas consiste nas configurações q (inicial e final) de localização e orientação, em um espaço de trabalho W e C se refere ao conjunto de todas suas configurações. Dessa forma, dado um robô holonômico (sem restrições de movimento), tem-se que seu espaço de configurações possui três dimensões (x,y,θ) e três graus de liberdade, sendo dois para translação (x,y) e um para orientação e sendo W bidimensional. Dessa forma, o planejamento

de rotas, ocorre no espaço de configuração C e não em W [39].

2.1.2 Veículos

O interesse da indústria em dotar os veículos com automação tem crescido nos últimos 20 anos, onde teve como foco principal a segurança ativa como suporte aos condutores. O *Lane Keeping Aid* (LKA) e o *Adaptive Cruise Control* (ACC) são exemplos de primeiros sistemas desenvolvidos pela indústria automotiva porém com baixa autonomia permitindo a tomada de decisão por parte do condutor [38]. Pesquisas recentes na indústria e na academia tem buscado aumentar a autonomia nos veículos. Estacionamento autônomo [40–42], bem como a completa autonomia dos veículos são tendências de um futuro próximo. Competições promovidas pela *Defense Advanced Research Projects Agency* (DARPA), tais como a *DARPA Grand Challenge* em 2005 [43] e o *DARPA Urban Challenge* em 2007 [44] tem impulsionado a comunidade acadêmica no desenvolvimentos da autonomia de veículos. Outras iniciativas a serem citadas são o carro do Google, o CaRINA (Carro Robótico Inteligente para Navegação Autônoma) da Universidade de São Paulo (USP, São Carlos) [45], que são exemplos de veículos testados nas vias urbanas, onde se utilizam de um conjunto de sensores e técnicas de localização bastante detalhadas do terreno para navegação no ambiente. Na proposta brasileira, a navegação ocorre da seguinte forma, inicialmente um condutor faz o trajeto previamente definido, uma ou mais vezes até que a coleta de dados do ambiente seja completada. Em seguida o veículo navega de modo autônomo, por meio do comparativo entre os dados previamente coletados. Em relação ao planejamento de rotas destas aplicações, ainda existem lacunas quando em relação à natureza reativa do mundo real, onde geralmente os algoritmos aplicados não se adaptar ao problema. Isto demonstra a necessidade da característica adaptativa (sensível ao contexto) ser incorporada aos planejadores de rotas em conjunto com sistemas de tomada de decisão, sensoriamento e controle. O planejamento de rotas em veículos autônomos difere-se em relação aos robôs, pois deve-se levar em consideração fatores humanos, como conforto e segurança dos passageiros, bem como apoio ao condutor [1, 38]. A Figura 2.1 apresenta as áreas de pesquisa que compõem um sistema de navegação para veículos autônomos, onde podem ser vistos as interações entre os componentes do sistema. O sistema de Tomada de Decisão é o reponsável pelo controle do veículo, sendo que suas reações são em tempo real, regido por técnicas e métodos multicritério, tendo como base as informações do fluxo do tráfego, sensores e os demais sistemas. Ressalta-se a área de Planejamento de Rotas e Sistemas de Localização, como sendo elementos cruciais para a movimentação dos veículos autônomos. Sendo portanto, a área de veículos autônomos um outro campo de aplicação desta pesquisa.

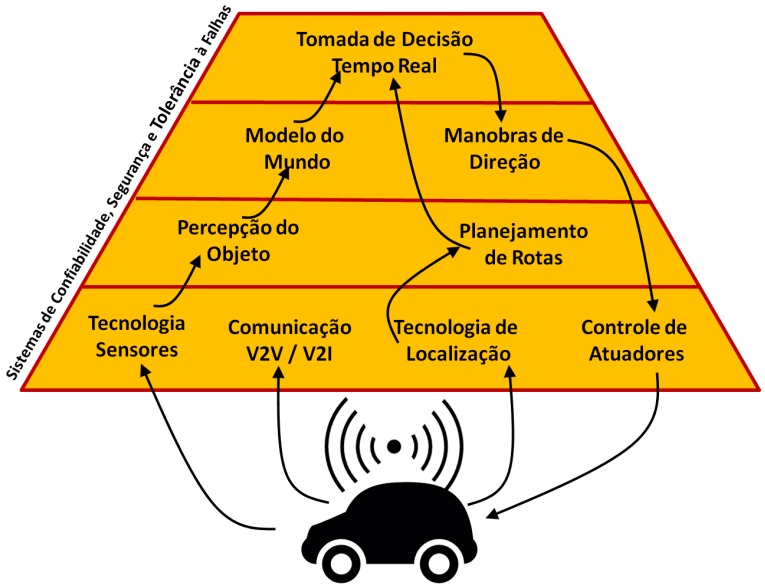


Figura 2.1: Sistema Autônomo adaptado de Furda *et al.* [1].

2.2 Redes Viárias

Uma rede viária pode ser representada como um grafo G finito, formado por um par (V,A) onde V é um conjunto finito não vazio denominado de vértices e A um conjunto de pares não ordenados denominados de Arestas. Temos ainda que o diâmetro de um grafo G é a maior excentricidade de qualquer vértice do grafo, é denotado como $diam(G) = max d_G(a, b) | a, b \in V(G)$ [46]. Desta forma, as arestas representam os seguimentos (trechos) das vias e os vértices correspondem a intersecções da rede viária. A ponderação nas arestas pode ser atribuída por qualquer função, porém para este trabalho, foram utilizados três tipos de ponderação: a) aleatória – onde os valores das arestas na rede viária, variavam de 1 até 80 (Seção 5.1); b) distância – nesta ponderação foram extraídos as distâncias de cada trecho das vias componentes da rede viária (Seção 5.2). Essa ponderação é calculada usando a Fómula de Haversine; e c) tempo de percurso – também pode ser visualizado na Seção 5.2, onde a ponderação é extraída de dados reais do fluxo de tráfego dos ônibus no Rio de Janeiro.

2.2.1 Busca em Redes Viárias

A busca da melhor rota em redes viárias reais, pode ser definida pela determinação do menor caminho entre os pontos de origem e de destino. Embora o algoritmo de Dijkstra tenha um desempenho satisfatório no contexto de grafos, para aplicações do mundo real, Dijkstra é considerado lento para aplicações interativas [32, 47–49]. Sabendo que o tempo de execução do algoritmo em questão, é determinado pela estrutura de dados, mais precisamente por sua fila de prioridade Q, tem-se uma boa

estratégia para melhorar o tempo de execução. Deste modo, para este trabalho foi utilizado uma versão do Dijkstra, da biblioteca *GraphStream*, usando a estrutura de *Heap de Fibonacci*.

Neste trabalho, o planejamento de rotas é formulado como uma variante do problema de otimização do menor caminho (SPP), onde o grafo é representado por uma rede viária do mundo real e cujas arestas possuem função de custo de velocidade por trecho. Essa associação é necessária para melhor ajuste ao sistema de monitoramento proposto por Ribeiro Júnior [2] e a qual este trabalho quer dar continuidade. Desta forma, o referido problema de otimização, na visão de um motorista é dado por dois pontos (vértices em marcações GPS) de origem e destino em uma rede viária de uma cidade (grafo) cujo objetivo é encontrar o caminho mais curto (melhor rota) entre estes pontos utilizando critérios estabelecidos de otimalidade (medidas de qualidade). Sabe-se que o problema pode ser resolvido em tempo linear e no espaço usando algoritmos clássicos como o de Dijkstra, porém para grandes redes viárias ele é demasiadamente lento [32]. Dessa forma é proposto neste trabalho a utilização do algoritmo de busca heurística da classe *Anytime*, o *Anytime Repairing A** (ARA*) ou sua versão otimizada proposta ITS-ARA*, para realização do planejamento de rotas nestas redes viárias. A seguir serão apresentados os algoritmos utilizados nos experimentos e cujos resultados foram comparados neste trabalho.

Algoritmo A*

O A* é um algoritmo proposto por Hart *et al.* [50] amplamente utilizado em Inteligência Artificial (IA) e em jogos eletrônicos. Trata-se de uma extensão do algoritmo de Dijkstra, onde existe uma função heurística $h(x)$ que estima o custo de deslocamento do vértice atual até o vértice destino, acelerando o processo de busca. Isto acontece por meio da limitação de vértices visitados, evitando a avaliação de vértices com menor peso e que provavelmente não comporiam a solução ótima [21]. A distância Euclidiana e a de *Manhattan* são exemplos de heurísticas, a primeira se baseia na distância em linha reta entre dois pontos num plano p_1 em (x_1, y_1) e p_2 em (x_2, y_2), tem-se $\sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$. A segunda refere-se à distância entre dois pontos medidos em um plano cartesiano, onde p_1 em (x_1, y_1) e p_2 em (x_2, y_2), tem-se $|x_1 - x_2| + |y_1 - y_2|$. Neste trabalho, a distância Euclidiana foi utilizada como função heurística nos algoritmos estudados pela simplicidade.

O algoritmo A* é expresso pela função $f(x) = g(x) + h(x)$, onde $g(x)$ refere-se à distância percorrida do vértice inicial ao vértice atual e $h(x)$ refere-se à distância estimada do vértice atual até o vértice destino. O algoritmo trabalha com duas listas temporárias: a de *OPEN* e a de *CLOSED*. A lista *OPEN* contém os vértices que foram submetidos à função heurística $h(x)$, mas que ainda não foram examinados

e expandidos. Esta lista é caracterizada pela prioridade de seus elementos: quanto maior o valor da função heurística atribuída, mais promissor é o vértice. A lista *CLOSED* se caracteriza pelos vértices já examinados e expandidos e que fazem parte da solução.

Algoritmo 1 Algoritmo A*.

```
1: função A*(origem, destino)
2:     limpar todas as listas
3:     adicionar origem em CLOSED
4:     enquanto OPEN não está vazia faça
5:         vérticeAtual := melhorVértice();
6:         se vérticeAtual = destino então
7:             devolve Caminho ótimo encontrado
8:         fim se
9:         remover vérticeAtual de OPEN
10:        adicionar vérticeAtual em CLOSED
11:        descobreVizinhos do vérticeAtual
12:        enquanto todos os vizinhos não visitados faça
13:            Efetuar cálculo dos valores de g(x) e h(x) para cada vizinho
14:            Efetuar cálculo de f(x) = g(x) + h(s)
15:            se vizinho tem melhor f(x) OPEN então
16:                continuar
17:            fim se
18:            se vizinho tem melhor f(x) CLOSED então
19:                continuar
20:            fim se
21:            escolher próximo vizinho a ser visitado
22:            adicionar vizinho em OPEN
23:        fim enquanto
24:    fim enquanto
25: fim função
```

O algoritmo inicia aplicando a função heurística ao vértice origem e em seguida o adiciona à lista *OPEN*, que é uma lista ordenada em função do seu custo estimado (heurística), ou seja, os vértices com menores custos vão para o início da lista (linhas 2-6). Em seguida, o algoritmo verifica se o primeiro vértice na lista *OPEN* é o mais promissor (menor custo) e, em seguida, remove-o da lista *OPEN* e o adiciona para lista *CLOSED* (linhas 7 - 12). A seguir, verifica se o vértice atual é o vértice de destino; se positivo (linha 14), o caminho foi encontrado e deve ser informado. Caso contrário, a função heurística será aplicada aos vértices vizinhos e adicionada à lista *OPEN* de uma forma ordenada (linhas 18-24). O algoritmo verifica se os vértices vizinhos estão dentro da pesquisa e se o caminho entre eles e a origem é menor que o anteriormente encontrado. Então, este vértice encontrado será parte da solução do caminho mais curto (*CLOSED* lista, linhas 8-10).

2.2.2 Algoritmos Anytime

Os algoritmos *Anytime* de planejamento de rotas são oriundos da área de Inteligência Artificial (IA) e têm por característica apresentar resultados subótimos rapidamente e cuja melhora é gradual, caso o tempo seja suficiente o caminho é ótimo [51]. Analogamente ao caminho ótimo, diz-se que um algoritmo de planejamento é completo se encontrar um caminho, caso exista, em tempo finito. Caso não

exista, o algoritmo sempre informa a sua não existência em tempo finito. Consequentemente, um algoritmo de planejamento é ótimo se ele encontra sempre um caminho ótimo [20, 21]. A seguir, serão apresentadas as descrições e os pseudocódigos dos algoritmos estudados neste trabalho.

Anytime Repairing A* – ARA*

O algoritmo ARA* (Algoritmo 2) foi proposto por Maxim Likhachev [21] como uma alternativa ao algoritmo A* na busca baseada em grafo, mais precisamente para o planejamento de trajetórias de robôs. A diferença entre ARA* e A* está no mecanismo de heurística e no refinamento de pesquisa. ARA* tende a encontrar resultados subótimos mais rápido em detrimento à otimalidade do seu antecessor. No entanto, caso o tempo seja suficiente, ARA* encontra resultados ótimos, graças à utilização da técnica *heurística inflada*, que reduz o número de vértices a serem verificados e consequentemente o tempo de execução. Isto é possível devido ao aumento no custo das distâncias $h(x)$, onde os primeiros vértices a serem verificados serão os vértices com menor custo, ignorando os vértices com custo semelhante que poderiam fazer parte do resultado.

A função de ARA* é dada por $f(x) = g(x) + \epsilon * h(x)$, onde $g(x)$ é à distância do vértice x até a origem, $h(x)$ é uma estimativa (heurística) da distância até o vértice final, ϵ é o coeficiente heurístico. Quanto maior o valor de ϵ, menos vértices serão visitados e os resultados são calculados rapidamente. A garantia da otimalidade é alcançada conforme se decrementa o valor de ϵ até um, pois a função passa a ser a mesma de A*. No processo de refinamento de resultados, decremento de ϵ, os valores $f(x)$, $g(x)$ e $h(x)$ são atualizados para cada vértice previamente encontrado.

O algoritmo trabalha com três listas: *OPEN*, *CLOSED* e *INCONS* (linha 27). A lista *OPEN* contém os vértices que foram submetidos à função heurística $h(x)$, mas que ainda não foram examinados e expandidos, trata-se de uma lista de prioridades de maior valor da função heurística. A lista *CLOSED* contém os vértices já examinados e expandidos uma vez na pesquisa. A lista *INCONS* (inconsistente) é responsável por evitar que vértices que tenham sua função $f(x)$ reduzida e que não estejam na lista *OPEN* sejam visitados mais de uma vez. A lista *INCONS* é uma lista temporária até o momento do refinamento é feito e seu conteúdo é adicionado à lista *OPEN*, fazendo com que os vértices já visitados tenham suas funções $f(x)$ reduzidas e sejam novamente avaliados. Isso significa que não é necessário recalcular o caminho mais uma vez, mas somente atualizar os pesos, de modo que uma *"nova rota"* seja concebida. ARA* é composto por três procedimentos: *FValues* responsável pelo cálculo $f(x) = g(x) + \epsilon * h(x)$; *improvePath* que executa o refinamento de pesquisas; e *Main* que gerencia o algoritmo, desde a criação das três listas (*OPEN*, *CLOSED* e *INCONS*) ao decréscimo do coeficiente heurístico ϵ. Em essência, ARA*

Algoritmo 2 Algoritmo ARA*.

```
1: função FVALUES(x)
2:     retorna g(x) + ε * h(x)
3: fim função
4:
5: procedure IMPROVEPATH())
6:     enquanto (key(x_destino) > min_{x∈OPEN}(key(x)) faça
7:         remove x com o menor fvalues(x)from OPEN;
8:         CLOSED = CLOSED ∪ x;
9:         para vizinhos(x) ← 1 até fim faça
10:            se vizinhos(x) nunca visitado por ARA* antes então
11:                g(vizinhos(x)) = ∞
12:            fim se
13:            se g(vizinhos(x)) > g(x) + c (x, vizinhos(x)) então
14:                g(vizinho(x)) = g(x) + h(x, vizinho(x))
15:                se (vizinho(x) ∉ CLOSED) então
16:                    insert/update vizinho(x) in OPEN com fvalues(vizinho(x));
17:                senão
18:                    insert vizinho(x) em INCONS;
19:                fim se
20:            fim se
21:        fim para
22:    fim enquanto
23: fim procedure
24:
25: procedure MAIN())
26:    g(x_destino) = v(x_destino) = ∞; v(x_destino) = ∞;
27:    g(x_origem) = 0; OPEN = CLOSED = INCONS = ∅;
28:    insere x_origem em OPEN com key(x_origem);
29:    improvePath();
30:    informa atual solução ε − suboptimal;
31:    enquanto ε > 1 faça
32:        decrementa ε;
33:        Mova os vértices de INCONS para OPEN
34:        Atualiza as prioridades x ∈ OPEN de acordo com fValues(x);
35:        CLOSED = ∅;
36:        improvePath();
37:        informa atual solução ε − suboptimal;
38:    fim enquanto
39: fim procedure
```

trabalha como se executasse A* várias vezes, seu controle está no decremento do coeficiente heurístico ϵ até 1, condição que garante a otimalidade da solução [20]. O ARA* difere do A*, por não garantir a visitação única a cada vértice durante a pesquisa, isto se deve à superestimação da função heurística causada pela inflação de ϵ.

O procedimento *improvePath* (linhas 5–23) é responsável pela reutilização dos resultados encontrados e que fazem parte da lista *INCONS*. Esta lista é composta por vértices que foram visitados (pertencem à lista *CLOSED*), mas que tiveram sua função heurística reduzida, implicando em uma nova avaliação para verificar se será parte da solução. Esta operação é realizada no procedimento *Main* quando os vértices da lista *INCONS* são movidos para a lista *OPEN*. A função *FValues* é responsável pelo cálculo da heurística função $f(x)$ para cada vértice.

O procedimento *Main* (linhas 25–39) define os valores de ϵ e inicializa o cálculo da solução subótima. O refinamento da solução é realizado por um laço, onde as operações de atualização heurística nos vértices são realizados por cada diminuição de heurística coeficiente ϵ até um (solução ótima). Com intuito de promover uma

otimização do código por meio da economia na chamada de procedimentos no processo de convergência do coeficiente heurístico, e por conseguinte a generalização do código, foi implementado o algoritmo ITS-ARA* que será descrito a seguir.

ITS-ARA*

O algoritmo ITS-ARA* (Algoritmo 3) é uma versão do ARA* que promove a aceleração da convergência do coeficiente heurístico ($\epsilon = 1$) por meio da diminuição na chamada de três procedimentos (10, 11 e 12): a) *Move states from INCONS into OPEN*; b) *Update the priorities for all* $x \in$ *OPEN according to fvalues(x)*; e c) $CLOSED = \emptyset$. A economia computacional ocorre quando o coeficiente heurístico é decrementado para o valor um, significando que o resultado obtido é ótimo. Outra observação a ser feita é que o ITS-ARA* promove a generalização do algoritmo original, pode-se notar pelas chamadas dos procedimentos *improvePath* e *Atualiza as prioridades* (linhas 29–30 e 36–37 do Algoritmo 2) e que são reescritas nas linhas 6–7. Apesar desta modificação proposta, o algoritmo mantém todas as suas propriedades e funcionalidades. Em outras palavras, ITS-ARA* encontra rapidamente o caminho subótimo, verifica as possíveis alterações nos pesos e caso ocorram, recalcula o caminho ótimo.

Algoritmo 3 Algoritmo ITS-ARA*.

```
1: procedure Main())
2:     g(x_goal = v(x_goal) = ∞; v(x_start) = ∞;
3:     g(x_start = 0; OPEN = CLOSED = INCONS = ∅;
4:     insert x_start into OPEN with fvalues(x_start);
5:     enquanto ε >= 1 faça
6:         improvePath();
7:         publish current ε − suboptimal solution;
8:         decrease ε;
9:         se (ε > 1) então
10:            Move states from INCONS into OPEN
11:            Update the priorities for all x ∈ OPEN according to fvalues(x);
12:            CLOSED = ∅;
13:        fim se
14:    fim enquanto
15: fim procedure
```

Vale ressaltar a importância da aceleração do algoritmo para diversos contextos, dentre os quais podem ser citados o de sistemas com restrições de energia e para veículos autônomos com intuito de favorecer o sistema de tomada de decisão. A seguir serão apresentados os trabalhos relacionados com esta pesquisa.

Capítulo 3

Trabalhos Relacionados

Neste capítulo serão analisados alguns trabalhos relacionados com a solução proposta sendo abordados três aspectos: (1) específicos ao monitoramento de trânsito, onde serão apresentadas duas soluções propostas para área em questão, *COTraMS* [2] e *SINIAV* [52, 53]; e (2) Sugestão de rotas, onde são propostas técnicas de orientação ao motorista para qual rota deve ser tomada, neste sentido serão estudados os trabalhos *UTOSPF* [3], *MobEyes* [54], *Navopt* [55] um *Sistema de Rotas Adaptativas*. Por fim, são estudados dois outros trabalhos experimentais, sendo o primeiro direcionado para área de robótica e o segundo sendo um experimento utilizando uma malha viária. A seguir, os trabalhos correlatos serão descritos, de modo a identificar suas vantagens, desvantagens e características desejáveis para solucionar o problema em questão.

3.1 Sistemas de Monitoramento de Trânsito

Os sistemas de monitoramento de veículos não possuem uma precisão tão acurada, variando entre 10 e 30 metros na localização do veículo na via [56]. A seguir serão apresentadas duas propostas de monitoramento de tráfego.

3.1.1 SINIAV

O Sistema de Identificação Automática de Veículos (SINIAV) [52, 53] é um sistema de identificação veicular que foi criado pela Resolução 212 de 13 de novembro de 2009 do Conselho Nacional de Trânsito (CONTRAN) e modificado pela Portaria 85 de 5 de novembro de 2009, quando foram estabelecidos novos prazos e normas técnicas a serem atendidas para sua implantação em todo o território nacional. O sistema utiliza dados do veículo gravados em uma *tag* que se comunica por *Radio-Frequency IDentification* (RFID) com um sistema associado dotado de antena capaz de ler e validar os dados recebidos. O sistema admite os seguintes três tipos de *tags*:

a) *passivas* que respondem ao sinal enviado pela base transmissora; b) *semi-passivas*, que tem o funcionamento semelhante à anterior, porém são dotadas de bateria interior para dar capacidade de processamento ao microchip e dependem do sinal do leitor para se comunicar; c) *ativas*, que são dotadas de bateria que alimenta o chip e o sinal de comunicação, podendo manter o sinal continuamente independente do sinal de uma antena/leitor e com um alcance maior. Desta forma, os fabricantes de veículos poderão escolher por uma das tecnologias disponíveis, cabendo apenas que as soluções estejam dentro dos padrões estabelecidos nas Normas.

Desde o dia 15 de fevereiro de 2014, Roraima foi o primeiro Estado da federação a utilizar o emplacamento eletrônico, seguindo orientação do Departamento Nacional de Trânsito (Denatran), devido ao Estado possuir a menor frota e com isso ter a possibilidade de diluir os problemas que surgirem para os demais estados quando de suas implantações. A Resolução 433 de 23 de janeiro de 2013, estabelece o prazo limite de conclusão da instalação do sistema para 30 de junho de 2015.

Em sua concepção legal o SINIAV tem por objetivo dotar os órgãos executivos de trânsito com ferramentas modernas e interoperáveis com intuito de auxiliar no planejamento, gestão e fiscalização do trânsito utilizando tecnologia de identificação por radiofrequência [52]. Os principais objetivos do *SINIAV* são: a) Fiscalização urbana e rodoviária; b) Gestão do trânsito; c) Gestão de meios de pagamento e seguro; d) Gestão de transporte de cargas e logística; e) Gestão de trânsito; f) Acompanhamento do Ciclo de Vida dos Veículos.

3.1.2 COTraMS

O *Collaborative and Opportunistic Traffic Monitoring System* (COTraMS) é um sistema oportunístico e colaborativo para monitoramento de trânsito utilizando redes sem-fio IEEE 802.11 infraestruturados proposto por [2]. Diferentemente das propostas de monitoramento automatizado de trânsito atuais, que possuem uma arquitetura centralizada responsável pelo cálculo e divulgação das vias, além de possuir um alto custo de implantação e manutenção, o sistema é descentralizado. O COTraMS é um sistema de monitoramento e divulgação das condições de trânsito, onde as unidades de bordo e acostamento não necessitam estarem interligadas entre si e nem a um ponto central. As informações são trocadas por meio de uma rede IEEE 802.11b/g de modo a atualizar suas tabelas de trechos da via de cada unidade de bordo permitindo um cenário sempre sensoriado. A Figura 3.1 apresenta a arquitetura do COTraMS, onde as informações são obtidas via unidades de bordo e de acostamento para detecção da movimentação do veículo na via, bem como definir e divulgar as condições de cada trecho da via. Devido a sua proposta ser colaborativa, a unidade de bordo é responsável por receber, processar e transmitir os dados sobre

sua movimentação para a unidade de acostamento. O sistema ainda prevê sua utilização em uma arquitetura centralizada, sendo que nesta visão a unidade de bordo transmite os dados para uma central.

As unidades de bordo são constituídas por equipamentos portáteis com *laptops*, *tablets*, *smartphones* ou qualquer equipamento que possua uma interface de rede com o padrão IEEE 802.11 b/g, a fim de executar uma aplicação proposta dentro do veículo que circula na via. A unidade de acostamento espera uma confirmação de que a unidade de bordo está pronta para receber a Tabela de Condições do Trecho (TCT). Ao receber a confirmação a unidade de acostamento envia a TCT e para cada nova unidade de bordo conectada é instanciada uma nova *thread* para uma nova etapa de troca de informações. Uma vez enviada a TCT, a *thread* aguarda a TCT da unidade de bordo para atualização dos valores e em seguida é realizado o cálculo da média harmônica para finalizar o processo de atualização. A seguir são apresentadas as funções componentes do COTraMS:

- **Unidade de Bordo** – é responsável por quatro funções: a) calcular a velocidade dentro de um trecho; b) atualizar a TCT local; c) detectar o melhor momento de envio do TCT para unidade de acostamento; d) enviar o TCT à unidade de acostamento mais próxima, atualizando também os trechos anteriores já que os valores do tempo de vida (*Time to Live* (TTL)) de cada entrada na tabela são maiores. Na Figura 3.1 pode ser visto que um veículo ao passar por um trecho 1, envia as informações referentes à velocidade no trecho para unidade de acostamento 2, onde as informações dos trechos 1 e 4 serão as mais atualizadas. Devido a não haver comunicação entre as unidades de acostamento, os veículos possibilitam a atualização sobre a via, cumprindo o papel dos enlaces de comunicação e possibilitando com que cada unidade de acostamento tenha informação sobre toda a via.

- **Unidade de Acostamento** – divulga sua TCT atual ao cliente (veículo), em ambas as direções, após o término do processo de conexão. Dessa forma, cada unidade de acostamento recebe uma TCT que é comparada com a informação local, e caso o TTL de cada entrada for maior que a informação atual a TCT é atualizada. Para filtrar as informações advindas do envio de informações de vários veículos sobre as condições dos trechos, é feita uma média harmônica simples para o cálculo da condição atual do trecho.

Na Figura 3.2 pode-se visualizar o processo de atualização do TCT, onde a tabela local no instante $t + 1$ é atualizada após a comparação das tabelas no instante t. Sendo que ao detectar o momento que o veículo passa pela unidade de acostamento, o cálculo da velocidade no trecho é feito e em seguida a TCT local é atualizada e enviada para a unidade de acostamento.

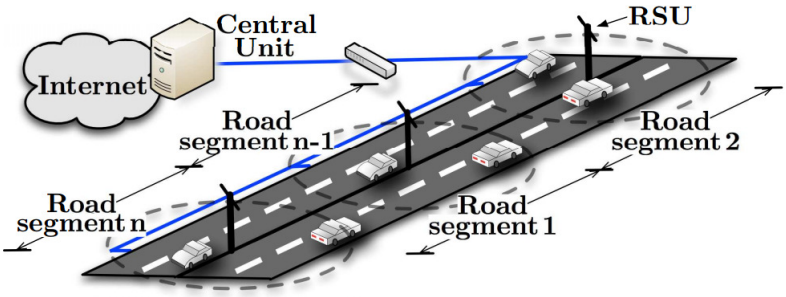


Figura 3.1: COTraMS [2].

TCT Local (instante t)

Trecho	Condição (km/h)	TTL
D1	80	20
D2	70	13
E3	75	12
E4	68	7

TCT recebida

Trecho	Condição (km/h)	TTL
D1	80	20
D2	68	15
E3	60	14
E4	65	10

TCT Local (instante t+1)

Trecho	Condição (km/h)	TTL
D1	80	20
D2	68	15
E3	60	14
E4	65	10

Figura 3.2: Atualização do TCT local [2].

Este trabalho se baseia nesta metodologia de sensoriamento de velocidade por trecho para servir de alimentação das ponderações das arestas para o planejamento de rotas. Isto se deve ao fato de que nesta pesquisa se mapeia as redes viárias como grafos. Desta forma, com intuito de simular estas informações no referida pesquisa, são utilizados dados randômicos com variações de 1 até 80, e dados reais coletadados dos ônibus da cidade do Rio de Janeiro.

3.2 Planejamento de Rotas em Redes Viárias

Os trabalhos estudados nesta seção descrevem propostas de sugestões de rotas com base em informações coletadas na rede viária. Inicialmente os trabalhos e suas metodologias são descritas e na última seção do capítulo são feitas algumas considerações. A seguir serão apresentados os trabalhos e suas respectivas abordagens.

3.2.1 UTOSPF

Em Faez e Khanjary [3] o algoritmo de Dijkstra é utilizado para encontrar o menor caminho baseado nas condições de trânsito coletadas por Redes de Sensores Sem Fio (RSSF) e sistemas ITS (sensores ópticos, indutivos, magnéticos e câmeras de vídeo). As rotas otimizadas são enviadas aos veículos via mensagens, no modo básico. No modo avançado, são previstos veículos equipados com transceptores para proporcionar comunicação direta e interativa.

A estrutura do UTOSPF é visualizada na Figura 3.3, onde a Unidade de Interseção (UI) calcula a Média Total da Velocidade (*Total Average Speed* – TAS) de todas as ruas com base nas Médias Locais de Velocidades (*Local Average Speed – LAS*) de cada rua componente da rota informada. Estas informações são recebidas pelas

Unidades de Rua (*Street Unit – SU*), em seguida as UIs calculam o tempo estimado destas ruas de acordo com seu comprimento e suas TAS. Em seguida, essas informações são enviadas para as demais UIs, de modo que todas tenham o conhecimento umas das outras. Por fim, é utilizado o algoritmo de Dijkstra de caminho mais curto para disseminar as melhores rotas para os motoristas por meio de painéis de mensagens (*Variable Message Signs* – VMS) ou por transceptores especiais instalados nos veículos.

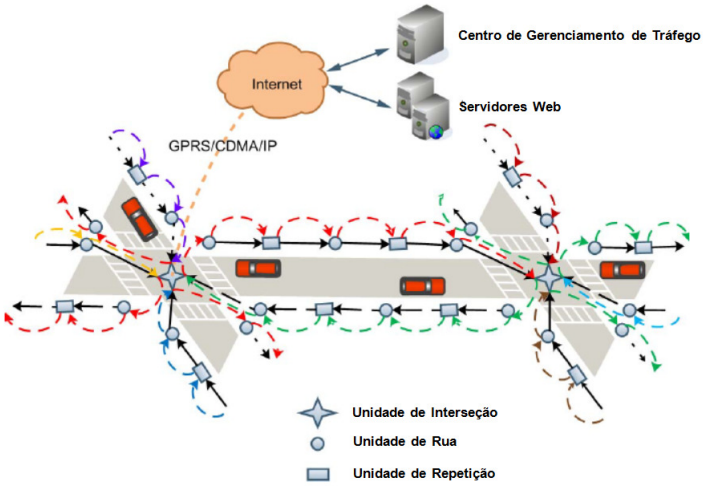


Figura 3.3: Estrutura original do UTOSPF [3].

3.2.2 MobEyes

O sistema *MobEyes* proposto por Lee *et al.* [54] utiliza-se de um algoritmo bio-inspirado para determinação do melhor caminho, usando informações de sensores que captam, classificam e geram periodicamente resumos das informações extraídas do contexto do tráfego de veículos. Estes resumos são disseminados na rede veicular utilizando agentes móveis oportunistas. O algoritmo é bio-inspirado em três comportamentos: o da bactéria *E. coli*, que apresenta modos de locomoção que variam de acordo com nível da concentração de nutrientes; o das formigas (outros insetos sociais), que se utilizam do feromônio para sinalizar aos companheiros o ninho com potenciais conflitos e nos voos de Lévy, que se inspiram no comportamento de voo de alguns grupos de pássaros como os albatrozes quando buscam alimentos.

3.2.3 Navopt

Em Li *et al.* [55] é proposta a utilização do algoritmo de busca A* com uma abordagem adaptativa de tempo real baseada nas condições do tráfego durante o período de um dia: a escolha é feita com base no menor custo para diferentes

horários do dia. A proposta utiliza regras *fuzzy* para adaptação às mudanças no tráfego baseadas no tráfego ou ajustes feitos pelo motorista.

3.3 Estudos Experimentais

Nesta seção são apresentados dois trabalhos experimentais, sendo que o primeiro se baseia em um algoritmo de planejamento de rotas utilizado na robótica, cuja proposta é a melhoria do algoritmo original proposto por Likhachev [21] em seu doutorado. O segundo trabalho se refere a um estudo de algoritmos de busca em uma malha viária, onde são estudados três algoritmos e feitas às devidas considerações. A seguir, os referidos trabalhos são apresentados.

3.3.1 ARA*+

Em Bo Li *et al.* [57], os autores propõem o algoritmo ARA*+ (para robótica), que é uma variação do ARA* que utiliza como estratégia a re-expansão dos estados (vértices visitados) depois do coeficiente heurístico ϵ ser decrementado. De acordo com os resultados dos experimentos, verificou-se que o número de estados expandidos e o tempo de busca para obter o caminho ótimo foi menor em ARA*+ do que em ARA*. A primeira busca de ARA*+ é igual à do ARA*, e como tal encontra rapidamente um caminho subótimo com o valor inicial (ϵ_0) e sem expansão dos estados. Caso haja tempo disponível para continuar a busca, o coeficiente ϵ será decrementado e um novo resultado será encontrado a partir do resultado prévio. A re-expansão dos resultados é permitida para valores de ϵ iguais a 1,2 ou 1, assim o ARA*+ tem menos estados expandidos que seu original (ARA*) e consequentemente menor tempo de busca.

3.3.2 ARA* e AD*

Em Moura *et al.* [58] é apresentado um comparativo dos algoritmos de busca de menor caminho A*, ARA* e *Anytime Dynamic A** (AD*) propostos por Likhachev [21], uma rede rodoviária foi utiliza para os testes. Foi medida a aceleração do ARA* em comparativo com o algoritmo A*, variando-se o coeficiente heurístico ϵ. As quantidades de vértices visitados e as mudanças nos pesos das arestas foram verificadas no segundo experimento comparativo, desta vez entre os algoritmos AD* e A*. Os resultados dos experimentos mostraram uma aceleração máxima de 50 vezes do ARA* sobre o A* no primeiro experimento. Para o segundo experimento foi verificado a alteração nos pesos das arestas (cerca de 4000 arestas modificadas). Observou-se que para mudanças incrementais na região de interesse não foi vantajoso reutilizar os resultados encontrados, apenas 5% se beneficiaram das mudanças.

Porém ao se decrementar os pesos nas arestas, a solução encontrada é 20% menor que a encontrada pelo algoritmo A*.

3.4 Comparativo dos Trabalhos Relacionados

As propostas de [54] e [55], utilizam informações baseadas no tempo para o monitoramento da rede viária, respectivamente, por meio de resumos dos comportamentos dos veículos (posição ou rota tomada) em determinados horários e por informações dadas num período de um dia. Tais abordagens, porém, tendem a ocasionar congestionamentos, pois as informações não refletem as reais condições das vias informadas (por exemplo, obstruída para manutenção ou outro evento), nem tampouco a densidade de veículos nelas. A proposta de [3] é bastante sofisticada e a que exige grandes investimentos e desafios em relação à interoperabilidade dos equipamentos, devido à existência de muitos fabricantes, cada um com seu próprio protocolo de comunicação. Diferentemente de Faez e Khanjary [54] e de Lee *et al.* [55], a presente pesquisa propõe o uso do CoTraMS [2] para o monitoramento do trânsito, por ser uma solução simples, eficiente, com baixo custo quando comparada Li *et al.* [3], e sem desafios quanto à interoperabilidade. No trabalho de Moura *et al.* [58] são estudados os algoritmos ARA* e AD*, bem como viabilidade de seu emprego em uma rede viária. São medidas as acelerações do ARA* em comparação com as do A* variando-se o coeficiente heurístico e, além disso é verificada a influência da variação dos pesos nas arestas na determinação do menor caminho. Os resultados mostraram uma aceleração máxima de 50 vezes do ARA* sobre o A* para o primeiro experimento. No segundo experimento é realizado um comparativo entre os algoritmos AD* e A*, onde foram variados os pesos nas arestas a cada novo par (origem, destino). Verificou-se que para um número de alterações na região de interesse maior que 5% pode ser menos eficiente que recalcular o caminho mínimo com um algoritmo tradicional. Em Bo Li *et al.* [57] é proposto o algoritmo ARA*+, uma variação do ARA* para planejamento de rotas para robôs. A estratégia para aceleração da busca do caminho ótimo é por meio da re-expansão dos estados para um certo valor de ϵ, um parâmetro do algoritmo ARA* já discutido no Capítulo 2, nesta pesquisa é proposto uma otimização do código do ARA* por meio de uma generalização das chamadas de seus procedimentos de planejamento.

Neste trabalho propõe-se a utilização do algoritmo *Anytime* ITS-ARA* para o planejamento de rotas em redes viárias, fazendo uso de sistema de monitoramento de trânsito por trecho, como por exemplo o sistema proposto por [59] que se baseia em IEEE 802.11. Para isto, é feito um estudo comparativo de algoritmos para escolha do menor caminho, dados os pontos de origem e destino. Diferentemente do trabalho de [58] são investigados três algoritmos: o clássico Dijkstra, que calcula o caminho

de custo mínimo entre vértices, o de busca informada A*, que utiliza heurística para encontrar o caminho ótimo e o ARA*, que calcula rapidamente um caminho subótimo, e melhora seu resultado à medida que o tempo é disponibilizado, fazendo uso da técnica de heurística inflada e do refinamento dos vértices visitados para chegar ao objetivo. Os cenários e as medidas utilizadas nos comparativos dos algoritmos são outros diferenciais em relação a [58]. Para o experimento, foram utilizados como cenários os mapas viários de sete cidades e duas plataformas computacionais (sendo uma com características restritivas), onde foram medido tempo de execução, a memória consumida, a quantidade de vértices que compõe os caminhos encontrados e quanto à influência da função heurística nos resultados encontrados. A seguir serão apresentados o *software* desenvolvido para dar suporte nesta pesquisa, bem como seus componentes e funcionalidades.

Capítulo 4

Automatizando o Planejamento de Rotas

O processo de transformação das mapas viários reais em grafos dinâmicos é abordado neste capítulo. São apresentadas as ferramentas utilizadas no processo de transformação de informações do mundo real em um modelo matemático estudado no Capítulo 2. A abordagem proposta tem por premissa a utilização de ferramentas livres para a composição da solução, para isto se optou pela utilização do mapa colaborativo *OpenStreetMap* e da biblioteca de grafos dinâmicos *GraphStream*. A seguir serão apresentadas as referidas ferramentas e o processo de extração da informação do mundo real.

4.1 OpenStreetMap

O OpenSteetMap (OSM) é um projeto livre e colaborativo criado em 2004 por Steve Coast com o objetivo de criar e fornecer dados geográficos de forma gratuita [27, 60–64]. A base de dados é criada e atualizada por meio de mapeamento colaborativo, onde diversos voluntários (colaboradores) reúnem informações geográficas como ruas, avenidas, lugares (edifícios, praias, lanchonetes) captados por meio de dispositivos de GPS. Este movimento colaborativo é chamado de *Volunteered Geographic Information* (VGI) [12–16], onde os dados espaciais digitais são captados, mantidos e compartilhados por cidadãos ao invés de instituições formalmente encarregadas.

4.1.1 Modelo de Dados

O OSM utiliza seu próprio modelo de dados para representar os elementos geográficos por meio da linguagem de marcação *eXtensible Markup Language* (XML). Outra vantagem do projeto é que os mapas podem ser baixados a partir do servidor

com ajuda da OSM API. Basicamente o OSM XML possui três tipos primitivos serão descritos a seguir.

- Nós (*Nodes*)– trata-se do elemento base do modelo, nós se referem a um conjunto de pontos/nós no espaço. É o único tipo primitivo que possui informação sobre a localização, latitude, longitude e altitude, embora esta última seja opcional.
- Caminhos (*Way*) – são formas lineares com limitações, descrevem a ligação entre pelo menos dois pontos. São utilizados para descreverem características lineares como ruas, avenidas, rios e fronteiras entre outros.
- Relações (*Relation*) – são utilizadas para explicar como os dados primitivos se relacionam. Um exemplo de relação é a rota que um veículo toma para chegar a um ponto de destino é constituída pelos diferentes caminhos (way) e nós (pontos de passagem do veículo).

Os tipos primitivos são constituídos por diversos atributos como:

- *user* – refere-se ao nome do usuário que realizou a última modificação na informação;
- *uid* – é o identificador do usuário;
- *timestamp* – trata-se do tempo (data/hora) da última modificação;
- *visible* – se a informação é apagada (invisível);
- *version* – informa a versão da informação;
- *changeset* – informa o número de alterações da informação.

Existem ainda os sub-elementos que constituem as informações expressas em OSM XML e que podem ser vistos na Tabela 4.1 que apresenta suas denominações, descrições e suas utilizações.

Sub-elemento	Usado em	Descrição
Tag	Nós, Caminhos, Relações	Adiciona informações de vários tipos. A informação é armazenada como par de chaves.
Nd	Way	Identifica o nó.
Member	Relação	Identifica o elemento básico.

Tabela 4.1: Sub-elementos básicos do OSM XML.

Um exemplo de descrição de um dado geográfico em código OSM é apresentado na Listagem 4.1 e sua respectiva representação em mapa é visualizada na Figura 4.1.

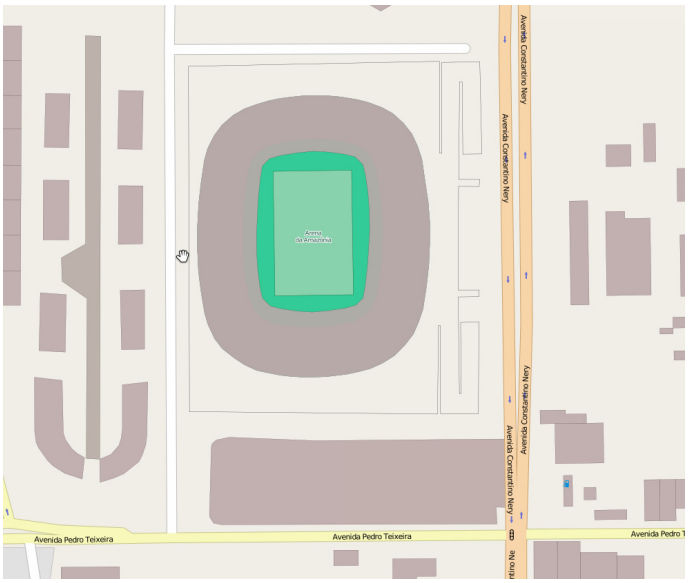


Figura 4.1: Arena da Amazônia – Manaus.

O exemplo em questão refere-se à Arena da Amazônia em Manaus, um dos doze estádios de futebol utilizado na Copa do Mundo de 2014.

O código em *OSM* é apresentado a seguir, de modo que se pode observar nas linhas 5 a 7, que as informações do referido local são explicitadas com os seguintes dados: a) cidade (Manaus); b) CEP (69000000); c) avenida (Avenida Constantino Nery); d) *website* (http://arenadaamazonia.com.br) entre outros.

Arena da Amazônia (Manaus) – exemplo expresso em *.osm*

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <osm version="0.6" generator="Overpass API">
3      <note>The data included in this document is from www.openstreetmap.
           org. The data is made available under ODbL.</note>
4      <meta osm_base="2014-09-22T20:16:02Z"/>
5      <node id="2891288620" lat=''-3.0824684" lon="-60.0281059" version
           =``2" timestamp=``2014-05-31T12:21:20Z" changeset="22655047"
           uid="710590" user="LordOfMaps"/>
6          <tag k="addr:city" v="Manaus"/>
7          <tag k="addr:postcode" v="69000000"/>
8          <tag k="addr:street" v="Avenida Constantino Nery"/>
9          <tag k="2014worldcup" v="venue"/>
10         <tag k="leisure" v="stadium"/>
11         <tag k="name" v="Arena da Amaz\^onia"/>
12         <tag k="note:offset" v="please note, that the roof is above the
               ground-level"/>
13         <tag k="sport" v="soccer"/>
14         <tag k="website" v="http://arenadaamazonia.com.br/"/>
15         <tag k="wikidata" v="Q641527"/>
16         <tag k="wikipedia" v="pt:Arena da Amaz\^onia"/>
17     </node>
```

```
18 </osm>
```

Para o presente trabalho foram utilizados mapas de cidades brasileiras. Foram baixados seus respectivos arquivos *.osm* e que foram importados para uma aplicação construída para análise da rede viária e testes dos algoritmos estudados no trabalho.

4.1.2 Análise de arquivos .OSM

Neste trabalho a linguagem de programação utilizada foi Java devido a sua capacidade de atuar em múltiplas plataformas e possibilidade de se conectar ao ROS (*Robot Operating System*) [65–68] para um possível núcleo de um sistema de planejamento de rotas para estes veículos. Dessa forma, foram utilizadas as APIs (*Application Programming Interface*) da referida linguagem. Para a utilização de arquivos no formato *.OSM*, de modo que se tenha acesso a todas as informações nele contidas, é necessário utilizar algumas ferramentas que permitirão realizar a análise dos dados (*parsing*). A análise dos arquivos *.OSM* é realizada pela biblioteca *Xerces XML*, que tem por ofício o processamento de documentos XML.

4.2 GraphStream

GraphStream é uma biblioteca Java para modelagem e análise de grafos dinâmicos com intuito em usá-los em simulações. A biblioteca foi concebida na ideia do grafo evoluir com o tempo, sendo assim, seu cerne central é que os grafos não são objetos estáticos mas são fluxos de eventos que evoluem com o tempo [69]. A biblioteca é constituída de outras ferramentas complementares que serão descritas a seguir.

4.2.1 Pacotes

A biblioteca *GraphStream* organiza suas funcionalidades em pacotes (*packages*), com intuito de agregar as classes relacionadas a um propósito comum e promover à reutilização de código, reproduzindo desta forma a ideia de biblioteca de código. A seguir serão apresentados os pacotes que constituem a referida biblioteca.

- *Graph Representation* – trata-se do principal pacote fornecido por *GraphStream* e onde estão definidos os métodos e classes dos grafos compartilhados pelos demais pacotes.
- *Graph Generation* – ferramenta de visualização ou simulação de eventos em grafos. É composta ainda por geradores de grafo que compreendem desde grafos regulares, aleatórios, livres de escala, entre outros.

- *Graph I/O* – é responsável pela leitura/escrita dos fluxos de eventos que podem ser gerados no estudo dos grafos dinâmicos, podendo especificar o formato dos mesmos.

- *Graph Theory* – ferramenta que fornece vários algoritmos conhecidos em grafos, propondo sempre que possível, versões de algoritmos para serem usadas em grafos dinâmicos.

- *Graph Visualization* – é a ferramenta dotada de recursos de exibição, que compreendem vários atributos de estilos como cores, traços, setas, ícones a capacidade de nomear vértices e arestas. Outra capacidade presente é a de mudança de automática de *layout* do grafo, podendo se adequar sempre que o grafo evoluir.

A biblioteca *GraphStream* foi escolhida para o desenvolvimento do *software* utilizado na presente pesquisa, devido às suas funcionalidades, além de possuir licença *GNU General Public License*[1]. A seguir é descrito o formato de arquivo pertinente à biblioteca em questão.

4.2.2 Analise de arquivos .DGS

O formato DGS é oriundo da biblioteca *GraphStream* e que permite descrição de grafos estáticos e dinâmicos em forma textual reduzida, isto vale para grandes grafos. A dinamicidade é representada por meio de eventos, como adição, exclusão ou alteração de vértices, arestas ou atributos do grafo.

No DGS, os grafos são composições de vértices e arestas, sendo que tais elementos são providos de atributos como *strings*, números e vetores de *strings* ou numéricos. Um arquivo DGS é composto por *header* e *body*.

Header

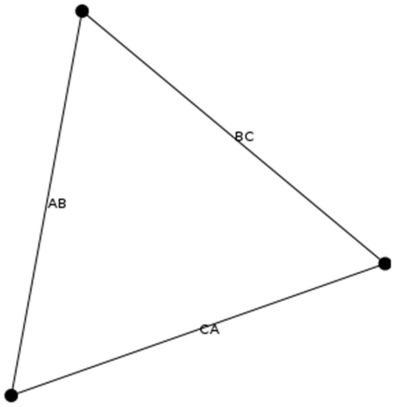
O *header* é composto por duas linhas definindo o nome do grafo e o formato da versão, sendo o comprimento opcional. Na primeira linha é descrita a versão referente ao DGS que atualmente é a quarta (004). A segunda linha contém três informações, o nome do grafo, a quantidade de passos (etapas) e o número total de eventos. Estas últimas informações são somente indicativas, podendo atribuir zero à eles. Contudo, estes campos são referentes à duração do evento, expresso em passos, e a quantidade de eventos respectivamente. A seguir, um exemplo extraído de GraphStream [28], onde um arquivo DGS é apresentado, onde pode-se observar o *header* nas linhas 1 e 2. Na linha 1 é indicada a versão do DGS e na linha 2

[1]Designação da licença para *sotware* livre.

pode ser visto as três informações complementares, o nome do grafo e as quantidades de passos e eventos respectivamente. Pode-se observar o exemplo se refere a um grafo estático, pois em suas informações complementares apresentam zero. A Figura 4.2(b) apresenta a representação do código DGS em *applet* Java.

```
1   DGS004
2   "Tutorial 2" 0 0
3   an "A"
4   an "B"
5   an "C"
6   ae "AB" "A" "B"
7   ae "BC" "B" "C"
8   ae "CA" "C" "A"
9   ce "AB" label="AB"
10  ce "BC" label="BC"
11  ce "CA" label="CA"
```

(a) Código DGS exemplo.



(b) Grafo correspondente.

Figura 4.2: Arquivo DGS e sua representação em grafo.

Body

No *body* estão descritos os eventos que caracterizam a dinamicidade do grafo. Essa dinamicidade é expressa por uma sequência de passos contendo zero ou mais eventos. A ideia de passos remete a noção de tempo no grafo, sendo que os eventos são ordenados e, caso façam parte da mesmo passo, considera-se que a ocorrência é no mesmo tempo. Os eventos podem receber parâmetros, sendo definidos por nome e um valor, sendo que são separados pelos sinais de dois pontos (:) e sinal de igual (=), sendo ambos permitidos. Um exemplo da utilização de parâmetros no DGS é apresentado a seguir.

Um vértice com identificador *b1* e valores de *x=5.23125* e *y=2.45691* como atributos é definido da seguinte forma: *an b1 x=5.23125 y=2.45691*.

O DGS possui comandos que afetam os elementos componentes do grafo e sua dinâmica. A seguir são descritos cada um deles.

- *an* – comando que permite adicionar um vértice, sua sintaxe é definida por um nome (identificador) entre aspas duplas, seguido de outros campos para definição de parâmetros, conforme o exemplo do código DGS exemplo na Figura 4.2(a), nas linhas 3, 4 e 5, adicionam os vértices A, B e C respectivamente;
- *cn* – permite alterar valores de atributos em um vértice. Um exemplo desse comando: *cn b1 x:5 y:1*, podendo ser usado o igual ao invés dos dois pontos;

- *dn* – comando para apagar um vértice. Exemplo, *dn b3*;
- *ae* – mesmo princípio do comando *an* comentado anteriormente, sendo que é voltado para as arestas do grafo. A Figura 4.2(a), nas linhas 6, 7 e 8, adicionam as arestas *AB*, *BC* e *CA*, juntamente com seus respectivos parâmetros em *string*;
- *ce* – altera valores de atributos das arestas. Como por exemplo, *ce X peso=20*, refere-se na alteração do atributo *peso* para o valor 20;
- *de* – apaga uma aresta do grafo, como exemplo tem-se: *de CD*;

4.3 GTAPlanning

Nesta seção é apresentado o *software GTAPlanning*, que se utiliza da teoria grafos na análise e nos testes dos algoritmos de planejamento de rotas, ferramenta construída na IDE Eclipse. A tecnologia, os conceitos utilizados na construção do *software* e as funcionalidades da ferramenta são apresentados a seguir.

4.3.1 Eclipse

O Eclipse é uma plataforma de desenvolvimento integrado (IDE -*Integrated Development Enviroment*) criada pela IBM em 2001 e que hoje é administrada pela comunidade *open-source* com intuito de construir e manter uma plataforma aberta e extensível, capaz de auxiliar no desenvolvimento e gerenciamento de todo o ciclo de vida do *software* [70–72]. A principal características do Eclipse é sua arquitetura baseada em *plug-ins*, onde é possível agregar um pequeno conjunto de serviços por meio de um componente e trabalhar em conjunto com os demais.

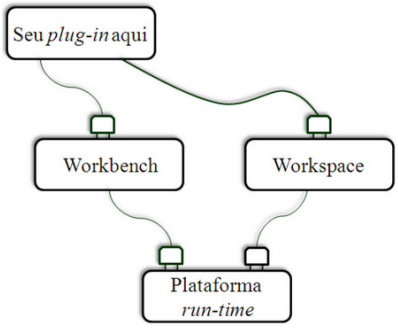


Figura 4.3: *Plug-ins* essenciais do Eclipse (adaptado de Gallardo [4]).

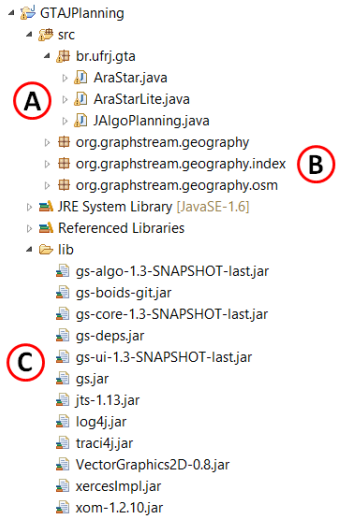
O Eclipse possui dois *plug-ins* que são a base de sua plataforma, pois proveem pontos de extensão para a maioria dos *plug-ins* nativos, são eles: *Workspace* e *Workbench* (Figura 4.3). O primeiro permite a interação do usuário com os recursos da

plataforma, como arquivos e projetos. O segundo permite que os *plug-ins* estendam a interface do Eclipse com menus, barras de ferramentas, criação de janelas entre outros. Na seção seguinte são descritas as bibliotecas utilizadas na construção do *GTAPlanning*.

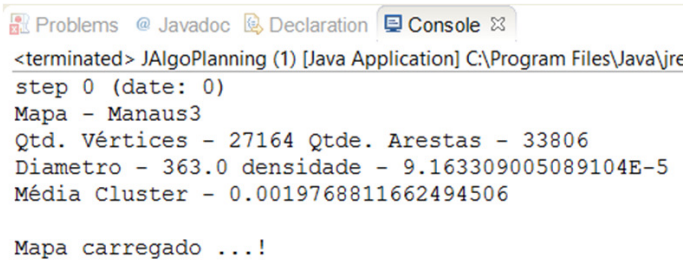
4.3.2 Bibliotecas Utilizadas

O *GTAPlanning* é visualizado na Figura 4.4, onde podem ser vistos os códigos fontes, pacotes e as bibliotecas utilizadas. Na Figura 4.4(a), cada informação é designada por uma letra, sendo a informação expressa em: **A** – relativo ao código fonte da aplicação e os pacotes constituintes, sendo que o primeiro pacote a ser visualizado *br.ufrj.gta* é o da aplicação proposta, nele estão os códigos fontes que atuam na aplicação, sendo dois deles algoritmos de planejamento de rotas (*AraStar.java* e *AraStarLite.java*) e o responsável pelo controle da aplicação *JAlgoPlanning*; **B** – se refere aos pacotes utilizados na extração dos arquivos *.osm*. Os códigos referente ao processo de extração estão presentes nestes pacotes; **C** – são referentes às bibliotecas utilizadas para a funcionalidade da aplicação. Como pode ser visto, o *GTAPlanning* se utilizou da proposta da IDE Eclipse, de agregar pequenos conjuntos de serviços por meio de um componente e trabalhar em conjunto com os demais [72].

O *software* se utiliza da Teoria dos Grafos para extrair algumas informações da rede viária analisada, a Figura 4.4(b) apresenta um exemplo destas informações. As informações extraídas dessa breve análise serão melhor explicadas na seção seguinte. A seguir serão apresentadas as bibliotecas que foram utilizadas na construção da aplicação.



(a) Bibliotecas utilizadas.



(b) Informações geradas no console.

Figura 4.4: GTAPlanning.

- **GraphStream** – as ferramentas utilizadas na construção do *software* foram: a) **gs-algo** – é a biblioteca onde estão definidas as classes de vários algoritmos baseados em grafos. Para utilização desta biblioteca em uma aplicação é necessário adicionar a biblioteca $gs - core$ que será descrita a seguir; b) **gs-core** – nesta biblioteca estão contidas as classes do *GraphStream*. Dessa forma, para utilizá-la em uma aplicação basta adicioná-la; e c) **gs-ui** – esta biblioteca destina-se a proporcionar alternativas de visualizações de grafos a serem usados pelo *GraphStream Viewer*. Nela está contido um renderizador desenvolvido na linguagem *Scala* que suporta propriedades do *Cascading Style Sheets* (CSS) ou simplesmente folha de rosto, que são definidas na biblioteca *GraphStream*, onde atualmente somente visualizações em 2D são implementadas.

- **jts** – se refere à *Java Topology Suite* (JTS), que é uma biblioteca em código aberto que fornece um modelo linear da geometria Euclidiana e é destinada para criação de *softwares* baseados em vetores, sendo bastante usado em Sistemas de Informação Geográfica (SIG) e em aplicações de geometria computacional como o referido projeto.

- **VectorGraphics2D** – biblioteca responsável por fornecer interface de gráficos 2D em Java para vários formatos de arquivos vetoriais como: a) *Encapsulated PostScript* (EPS); b) *Scalable Vector Graphics* (SVG) e c) Portable Document Format (PDF). Ela é usada nos *screenshots* utilizados na pesquisa.

- **xercesImpl** – é uma biblioteca para análise (incluindo DOM[2] e SAX[3], validação e manipulação de XML.

- **xom** – é um novo modelo para trabalhar com arquivos XML que promete corretude[4], simplicidade e desempenho.

As funcionalidades implementadas no *GTAPlanning* serão apresentadas a seguir.

4.3.3 Funcionalidades

Inicialmente, o *GTAPlanning* foi desenvolvido como módulo de planejamento de rotas e testes de planejadores, com intuito de medir o desempenho dos mesmos

[2]O Document Object Model (DOM) é um *parser* que decompõe um documento XML em um modelo hierárquico e o representa na memória, permitindo a leitura, manipulação e modificação do mesmo.

[3]A *Simple API for XML* (SAX) é um *parse* que realiza um processo serial no documento, sendo orientado à eventos e que faz acesso de elemento a elemento do documento XML, mantendo na memória somente as *tags* que estão sendo visitadas, ocasionando um menor consumo de memória.

[4]Se refere ao cumprimento de uma determinada especificação e cumprimento de referidos objetivos.

quanto ao tempo de processamento, memória consumida, quantidade de vértices e arestas. Porém, surgiu a necessidade de caracterizar as redes viárias estudadas quanto à suas características, para isso foi utilizada teoria dos grafos para expressar tais propriedades. O *GTAPlanning* (Figura 4.5) é iniciado a partir da escolha da rede viária a ser testada. Para sua seleção basta acionar o botão *Ler Mapa* referente a leta *A* na respectiva figura e em seguida, uma janela de seleção abrirá para escolha da rede viária (letra B, na mesma figura). Pode-se perceber que na janela de seleção do mapa, é aplicado um filtro nos arquivos, possibilitando apenas mapas descritos em *.dgs* e *.osm*, respectivamente letras *C* e *D*. Conforme será apresentado a seguir.

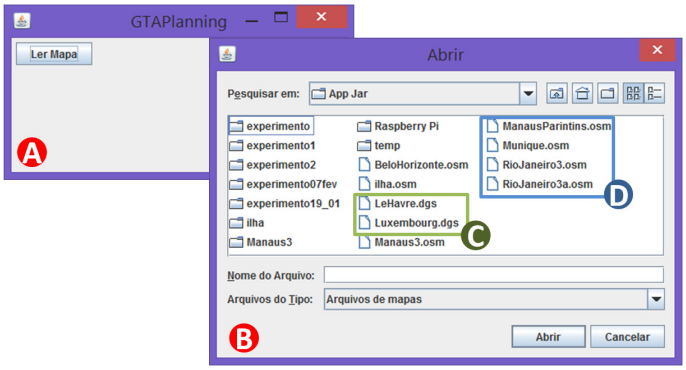


Figura 4.5: GTAPlanning – *Interface*.

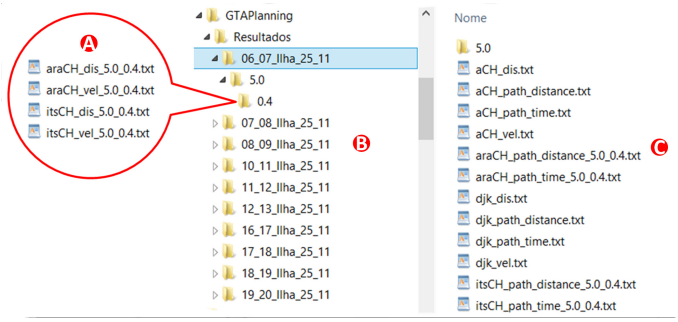


Figura 4.6: Dados gerados nos diretórios.

- *Arquivos de entrada* – atualmente a ferramenta aceita dois tipos de arquivos, o *.osm* e o *.dgs*. O primeiro é extraído do *OpenSteetMap* e foi abordado na Seção 4.1, o segundo é relativo a grafos (estáticos e dinâmicos) abordados na Seção 4.2.2.

- *Arquivos de saída* – A Figura 4.6 o conjunto de informações geradas pelo *sofware*. No referido exemplo, que será apresentado no Capítulo 5, podem ser observadas as dez janelas de tempo produzidas no experimento descritos no referido capítulo pela letra *B*. No diretório selecionado, foram gerados arquivos correspondentes aos algoritmos de Dijkstra ($djk_dis.txt$ e $djk_vel.txt$) e A*

($aCH_dis.txt$ e $aCH_vel.txt$), na letra *C*. Bem como no subdiretório (../5.0/0.4), estão os arquivos gerados para os algoritmos ARA* ($araCH_dis_5.0_0.4.txt$ e $araCH_vel_5.0_0.4.txt$) e ITS-ARA* ($itsCH_dis_5.0_0.4.txt$ e $itsCH_vel_5.0_0.4$) na letra *B* da mesma figura. Os referido arquivos apresentam as seguintes medidas relacionados aos seus respectivos algoritmos: a) tempo de execução; b) memória consumida; c) quantidade de vértices; d) distância percorrida; e) velocidade média; e f) tempo de chegada;

- *Análise de redes viárias* – ao carregar o arquivo referente à rede viária que será usada como cenário, o *software* executa uma análise baseada na Teoria dos Grafos para extrair algumas informações, tais como: a) quantidade de vértices; b) quantidade de arestas; c) diâmetro da rede; d) densidade da rede; e) coeficiente de *cluster*. Tais propriedades ainda não foram utilizadas nos experimentos relatados neste trabalho, mas espera-se que estes possam auxiliar na determinação de uma função de cálculo para encontrar o coeficiente heurístico usado nos algoritmos da classe *Anytime* estudados;

- *Testes de algoritmo de planejamento de rotas* – foi uma das primeiras funcionalidades a serem exploradas, trata-se da extração da medidas com base no sorteio de dois pontos randômicos e a busca pelo menor caminho. Para tais foram implementados dois algoritmos, ARA* e sua versão otimizada ITS-ARA*, além de contar com dois outros algoritmo já implementados da biblioteca *GraphStream*, Dijkstra e o A*. Estes quatro algoritmos constituem a base dos estudos neste trabalho. Inicialmente foram medidos as quatro grandezas já mencionadas: vértices, arestas, tempo e memória consumida. Surgiu em seguida o seguinte questionamento: Qual a influência da função heurística no planejamento de rotas? A questão é respondida em detalhes no Capítulo 5, bem como a variação do coeficiente heurístico.

- *Visualização dos cenários* – é possível visualizar desde o mapa após ser carregado pela aplicação, assim como o estudo de planejamento das rotas, onde são mostrados os pontos de origem e destino marcados por pontos azuis e verdes respectivamente, os trajetos de cada algoritmo são expressos também em cores. Na Figura 4.7 é possível visualizar o resultado do planejamento de rotas dos experimentos realizados na ferramenta, onde o ponto inicial é designado pelo marcador azul e consequentemente o verde se refere ao ponto final. Os planejamentos de rotas são definidos por cores, azul marinho se refere ao algoritmo A*, o vermelho representa o Dijkstra e o ITS-ARA* é representado pelo verde. A rede viária de Manaus é usada como cenário e a ausência da função heurística é utilizada no exemplo.

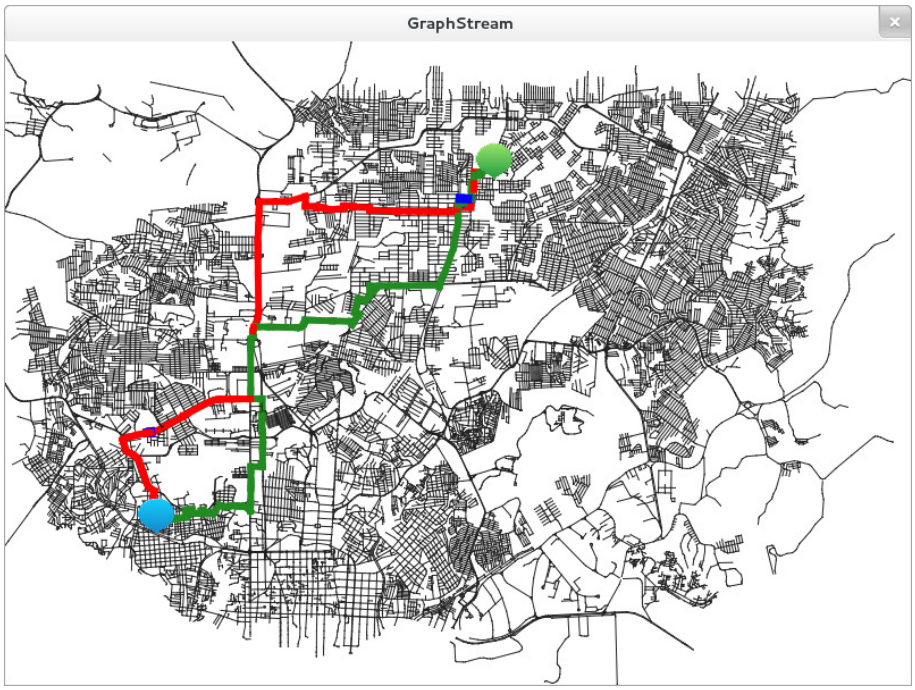


Figura 4.7: GTAPlanning - Planejamento de Rotas.

A visualização dos resultados e análises dos algoritmos e a influência da função heurística, serão apresentados no capítulo seguinte.

Capítulo 5

Planejamento de Rotas com Limitações Computacionais

Este capítulo apresenta o estudo do planejamento de rotas em redes viárias reais brasileiras, onde são analisadas a influência da heurística nos resultados e o estudo de duas ponderações extraídas do fluxo de tráfego real. Para os testes foi utilizado o *GTAPlanning* com intuito de extrair as características do grafo das redes viárias estudadas e as medidas coletadas pelos algoritmos de planejamento de rotas aplicados durante os experimentos. Os mapas utilizados nos experimentos foram extraídos da plataforma colaborativa *OpenStreetMap*. Foram utilizadas duas plataformas computacionais restritivas quanto a memória e processamento, o Raspbery Pi modelo B e B+. Testes e resultados serão apresentados ao final do capítulo.

5.1 Influência da Heurística

Para o experimento foram utilizados mapas das cidades brasileiras de Manaus e Rio de Janeiro extraídos do *OpenStreetMap*. A Tabela 5.1 apresenta as características dos respectivos cenários, quantidade de vértices e arestas, bairros componentes, diâmetro da rede viária e a medida da área de estudo. O presente experimento tem como objetivo testar a viabilidade computacional em uma plataforma típica de sistemas embarcados, mais restrita em termos de capacidade. O Raspberry Pi foi escolhido devido a suas características de *hardware* (memória RAM e processador) se assemelharem à de um *smartphone* básico. A avaliação de desempenho foi feita usando o ambiente de desenvolvimento *Eclipse* com a linguagem de programação Java e a biblioteca de grafos dinâmicos *GraphStream* [69], de onde foram produzidos executáveis *.jar* para geração e coleta dos dados no Raspberry Pi. Para os cenários foram feitas ponderações aleatórias (variando até 80) em suas vias (arestas). No experimento foram utilizadas as implementações dos algoritmos de Dijkstra e A*

provenientes do *GraphStream*. Destaca-se a implementação do primeiro algoritmos por utilizar-se da estrutura de dados *Heap de Fibonacci*, o que o torna mais eficiente para grafos muito grandes, que é o caso das redes viárias. Dessa forma, somente ARA* e ITS-ARA* foram implementados. A metodologia utilizada no experimento é descrita a seguir.

- **Leitura do mapa viário** – as redes viárias são lidas pelo *software* que executa o processo de transformação do arquivo em grafo. Os arquivos lidos são *.osm*. É necessário fazer uma extração da informação expressa em linguagem de marcação, para isto foi adicionado o *plugin org.graphstream.geography* responsável por esta conversão (*parser*). São extraídos quantidade de vértices, arestas e o diâmetro da rede viária. As informações de quantidade de bairros foram tiradas dos mapas sem o auxílio do *software*, bem como o tamanho da área de estudo que foi extraída utilizando o *Java OpenStreetMap Editor* (JOSM).

- **Ponderação aleatória das arestas** – foi criado um procedimento para gerar aleatoriamente o peso das arestas componentes do grafo em estudo. O procedimento é disparado após o grafo ser totalmente carregado no *software*, com intuito de atribuir ponderações com variações de 1 até 80.

- **Testes dos algoritmos de planejamento** – os algoritmos utilizados nos experimentos foram Dijkstra, A*, ARA* e ITS-ARA*, sendo este último uma versão otimizada de seu antecessor. Para todos os experimentos foram sorteados dois pontos, um de origem e outro de destino, os quais são passados como parâmetros para os algoritmos calcularem o planejamento. A cada sorteio e consequente cálculo, as arestas componentes do resultado são ponderadas aleatoriamente e novamente o cálculo é executado. Foram feitas cinquenta repetições para cada par de pontos, sendo que foram feitos três sorteios por par.

- **Resultados dos Testes** – ao fim dos testes de cada algoritmo são gerados arquivos contendo as seguintes medidas: a) tempo de execução; b) memória consumida; c) quantidade de vértices e d) quantidade de arestas componentes do planejamento encontrado. A influência da função heurística nos algoritmos de busca informada também foi medida, sendo portanto apresentados resultados com heurística e sem heurística para os referidos algoritmos de busca informada.

O Raspberry Pi utilizado no experimento foi o modelo B com processador ARM de 700 MHz, 512 MB de RAM e cartão de memória de 4 GB, sistema operacional Raspbian e Java com versão 1.6.0_27. Foram extraídas as médias dos resultados

Rede Viária	Quantidade de Vértices	Quantidade de Arestas	Quantidade de Bairros	Diâmetro
Manaus	4.723	5.645	8	139
Rio de Janeiro	7.146	7.968	14	254

Tabela 5.1: Características dos Cenários usados nos Experimentos.

de todas as medidas com intervalo de confiança de 95%. Estas medidas são importantes para se determinar a viabilidade computacional na referida plataforma e medir a qualidade dos resultados do planejamento, sendo o resultado do número de vértices/arestas a referência de qualidade, quanto mais próximo for seu valor ao gerado pelo algoritmo de Dijkstra, melhor será sua qualidade.

A Figura 5.1 mostra um exemplo da influência da heurística nos caminhos encontrados. As rotas dos algoritmos são designadas por cores, o azul designa Dijkstra, A* é representado por vermelho e a cor verde identifica a rota produzida por ITS-ARA*. Como as diferenças das rotas produzidas por ARA* e ITS-ARA* foram pequenas, optou-se por identificar somente as rotas de ITS-ARA*. Com aplicação da heurística, as rotas geradas são mais evidentes conforme podem ser vistos nas Figuras 5.1(b) e 5.1(d). No entanto, nas Figuras 5.1(a) e 5.1(c) evidenciam a ausência da heurística, fazendo com que as rotas geradas fiquem mais próximas à gerada por Dijkstra. A Tabela 5.2 confirma a observação feita por meio da quantidade de vértices constituintes da rota. Quando a função heurística foi utilizada, os algoritmos A* e ITS-ARA* apresentaram rotas alternativas (Figuras 5.1(b) e 5.1(d)). Os pinos azul e verde são usados para marcar a origem e o destino, respectivamente. O resultado das análises é feito a seguir, mostrando o tempo de execução, memória consumida e número de vértices para cada cenário.

5.1.1 Resultados sem Heurística

Nesta seção a função heurística $h(x)$ não foi considerada para o cálculo do menor caminho nos algoritmos de busca informados como A*, ARA* e ITS-ARA*. Dessa forma, temos a função expressa por $f(x) = g(x)$. Os resultados obtidos quanto ao tempo para ambos os cenários mostraram que A* foi superior aos demais. Porém, levando-se em consideração os intervalos de confiança, pode-se perceber que no cenário de Manaus houve uma grande variação nos algoritmos de busca informada. Deste modo, ITS-ARA* teve o mesmo desempenho que o A*. Para o cenário do Rio de Janeiro, os intervalos de confiança tiveram pouca variação. De modo que percebe-se que A* e ITS-ARA* tiveram o mesmo desempenho, somente no primeiro experimento, sendo que nos demais A* foi o mais rápido.

Para os resultados relativos à memória, Dijkstra foi o mais econômico. Isso se deve ao fato de os algoritmos de busca informada precisarem manter os resultados

(a) Manaus sem heurística. (b) Manaus com heurística.

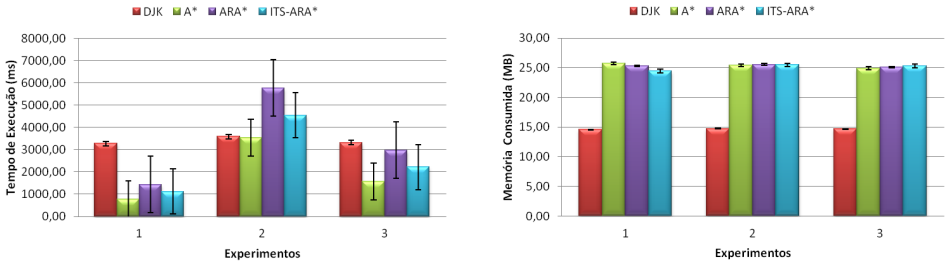
(c) Rio de Janeiro sem heurística. (d) Rio de Janeiro com heurística.

Figura 5.1: Influência da heurística no planejamento de rotas.

encontrados na memória durante as operações de busca. As listas (*OPEN*, *CLOSED* e *INCONS*, encontradas em ARA* e ITS-ARA*) são as estruturas utilizadas para tais operações. A Figura 5.2 exibe os resultados dos tempos gastos e memória consumida nos respectivos cenários. Para o cenário de Manaus (Figura 5.2(a)), em média todos os algoritmos tiveram melhor desempenho em relação ao tempo que no cenário do Rio de Janeiro (Figura 5.2(c)). O gasto de memória também é maior no Rio de Janeiro (Figura 5.2(d)) do que em Manaus (Figura 5.2(b)).

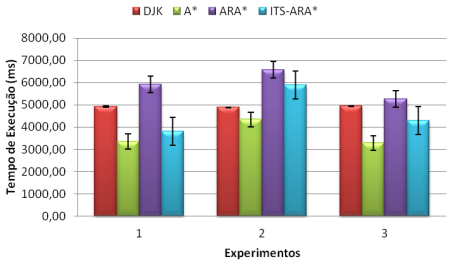
Manaus			
DJK	**A***	**ARA***	**ITS-ARA***
38	38,50	39,02	39,46
82	89,04	88,38	88,38
58	60,46	62,62	60,92
Rio de Janeiro			
DJK	**A***	**ARA***	**ITS-ARA***
129	133,40	135,08	130,48
81	82,20	82,10	84,26
73	73,92	75,64	75,42

Tabela 5.2: Resultados sem Heurística – Quantidade Média de Vértices.

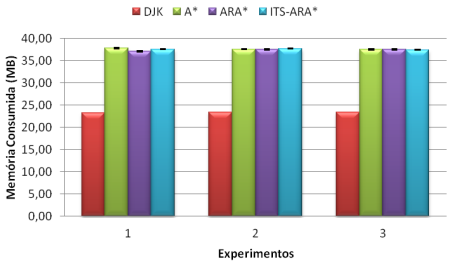


(a) Manaus – Tempo (ms).

(b) Manaus – Memória (MB).



(c) Rio de Janeiro – Tempo (ms).



(d) Rio de Janeiro – Memória (MB).

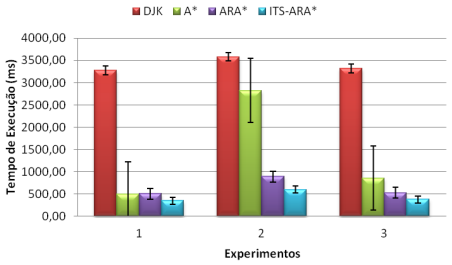
Figura 5.2: Tempo de Execução e Memória Consumida sem Heurística.

A Tabela 5.2 ilustra os resultados sem heurística dos experimentos em termos de quantidade de vértices para os cenários de Manaus e Rio de Janeiro, respectivamente. Pode-se notar que a variação aleatória dos pesos nos caminhos encontrados, foi refletida nas médias decimais dos resultados dos algoritmos A*, ARA* e ITS-ARA* para os dois cenários.

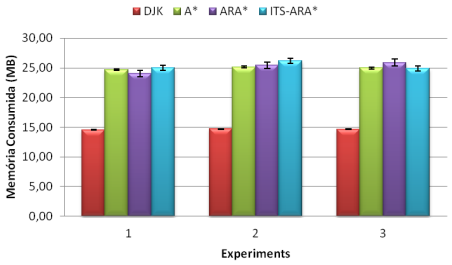
5.1.2 Resultados com Heurística

Em ambos os cenários, é bastante significativo o aumento de desempenho em relação ao tempo, quando a função heurística é adicionada aos algoritmos de busca informada (Figura 5.3). Quando se comparam os cenários de Manaus (Figura 5.3(a)) e Rio de Janeiro (Figura 5.3(c)) aos seus respectivos resultados sem heurística (Figuras 5.2(a) e 5.2(c)) pode-se notar o ganho em relação ao tempo de execução que ambos tiveram com o uso da heurística. No entanto, em relação à memória, os cenários de Manaus e Rio de Janeiro (Figuras 5.3(b) e 5.3(d)) tiveram poucas variações em comparação aos resultados sem heurística (Figuras 5.2(b) e 5.2(d)).

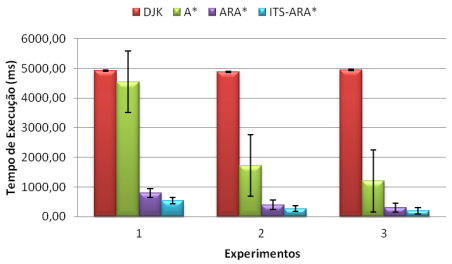
A Tabela 5.3 ilustra os resultados em relação aos caminhos encontrados de Manaus e Rio de Janeiro respectivamente. Pode-se notar que, apesar de encontrarem resultados mais rápido que o Dijkstra, os algoritmos de busca informada não conseguem a mesma eficiência em termos de qualidade das rotas, apresentando valores maiores que os valores de referência (Dijkstra). No entanto, em rotas menores os resultados ficaram próximos da referência, sendo que nos terceiros experimentos de



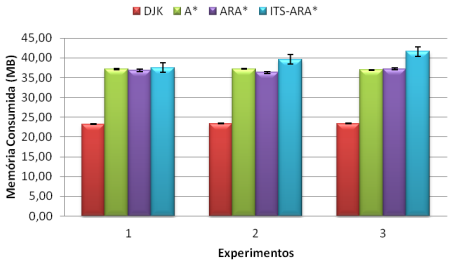
(a) Manaus – Tempo (ms).



(b) Manaus – Memória (MB).



(c) Rio de Janeiro – Tempo (ms).



(d) Rio de Janeiro – Memória (MB).

Figura 5.3: Tempo de Execução e Memória Consumida com Heurística.

Manaus			
DJK	**A***	**ARA***	**ITS-ARA***
38	42	54	54
82	92	129	129
58	64	60	60
Rio de Janeiro			
DJK	**A***	**ARA***	**ITS-ARA***
129	162	190	190
81	87	87	87
73	76	75	75

Tabela 5.3: Resultados com Heurística – Quantidade Média de Vértices.

Manaus e Rio de Janeiro, ARA* e ITS-ARA* obtiveram melhores resultados que A* e próximos aos valores de referência.

5.1.3 Análise de Resultados

Este experimento apresentou uma análise do desempenho dos algoritmos de busca *Anytime* no planejamento de rotas nas redes viárias de Manaus e Rio de Janeiro utilizando a plataforma Raspberry Pi. Foi analisada a influência da função heurística nos cenários propostos. ITS-ARA*, a modificação proposta de ARA*, apresentou resultados relevantes em ambos os cenários quanto ao tempo de execução, principalmente quando se faz uso da função heurística. Para o cenário de Manaus, em média o ITS-ARA* foi mais rápido 7,71 vezes do que o Dijkstra (tempo

DJK/tempo ITS-ARA*), usando 12,96% do seu tempo. Da mesma forma, em relação ao A*, o ITS-ARA* foi 3,16 vezes mais rápido consumindo 68,41% do tempo. E quando comparado com ARA*, o ITS-ARA* foi 1,45 vezes mais rápido e consumindo 31,37% de tempo. Para o Rio de Janeiro, em média: ITS-ARA* foi 14,70 vezes mais rápido que Dijkstra, usando 6,98% de seu tempo. Para o A*, ITS-ARA* foi 7,44 vezes mais rápido, consumindo 13,46% de seu tempo. Em comparação com o ARA*, ITS-ARA* foi 1,49 mais rápido, usando 67,05% de tempo.

Quanto à qualidade das rotas geradas, ITS-ARA* e ARA* tiveram desempenhos iguais com a utilização da função heurística, porém sem seu uso seus resultados ficaram mais próximos dos valores de referência de Dijkstra e sendo que algumas vezes ITS-ARA* superou até mesmo A* nos dois cenários de Manaus e Rio de Janeiro. Este fato chama atenção pois ARA* e ITS-ARA* em sua essência, executam A* n vezes de acordo com o coeficiente heurístico informado, conforme visto no Capítulo 2. Para o cenário de Manaus com uso da heurística, em média ITS-ARA* teve um desempenho de 36,51% maior que Dijkstra e 22,72% maior que A*. No cenário do Rio de Janeiro, os resultados em média do ITS-ARA* em relação ao Dijkstra foram de 24% maiores e 8,30% quando comparado ao A*. Apesar dos resultados com heurística em relação à qualidade das rotas terem sido superiores aos valores de referência, no geral o ITS-ARA* e ARA* mostraram-se bastante flexíveis nos resultados quanto à utilização da função heurística, sendo dessa forma boas alternativas de algoritmos planejadores de rotas.

Apesar de ITS-ARA* ser uma modificação de ARA*, ele mantém as características do original, como agregar a rapidez do A* e a eficiência da reutilização de resultados encontrados, não necessitando de novas buscas quando as ponderações são modificadas. Esta última característica é um diferencial para aplicações em sistemas dinâmicos como as redes viárias. A utilização do Raspberry Pi nos testes foi bastante satisfatória, pois os experimentos demonstram sua completa viabilidade para a construção de um protótipo veicular que utilize IEEE 802.11 no planejamento de rotas.

5.2 Fluxo de Tráfego Real

Para os experimentos a seguir foi utilizada uma parte do mapa do Rio de Janeiro, mais precisamente a Ilha do Governador, com uma área de 40,81 km^2 e 212.574 habitantes (cerca de 3,37 % da população da cidade), e onde está localizado o Aeroporto Internacional Antônio Carlos Jobim.

A metodologia utiliza neste experimento é mostrada na Figura 5.4, onde composta por: a) Rede Viária; b) Fluxo do Tráfego; c) GTAPlanning; e d) Análise dos Dados.

Rede Viária O mapa do Rio de Janeiro foi extraído do *site* OpenStreetMap [73, 74], porém somente a parte que representa a Ilha do Governador foi considerada (Figura 5.5(a)). Este fragmento do mapa contém 7.146 vértices e 7.968 arestas.

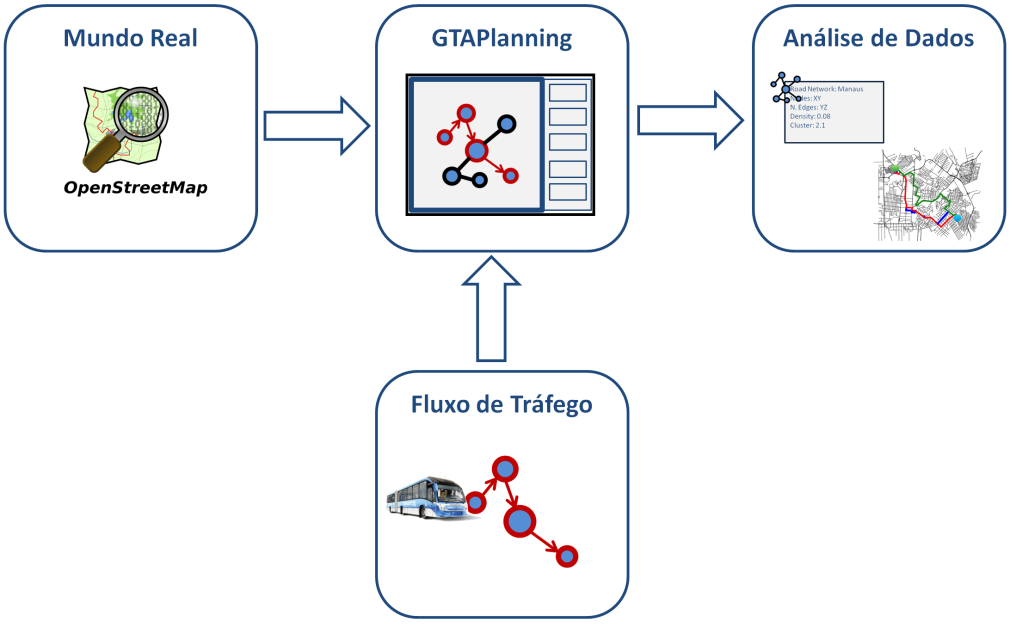
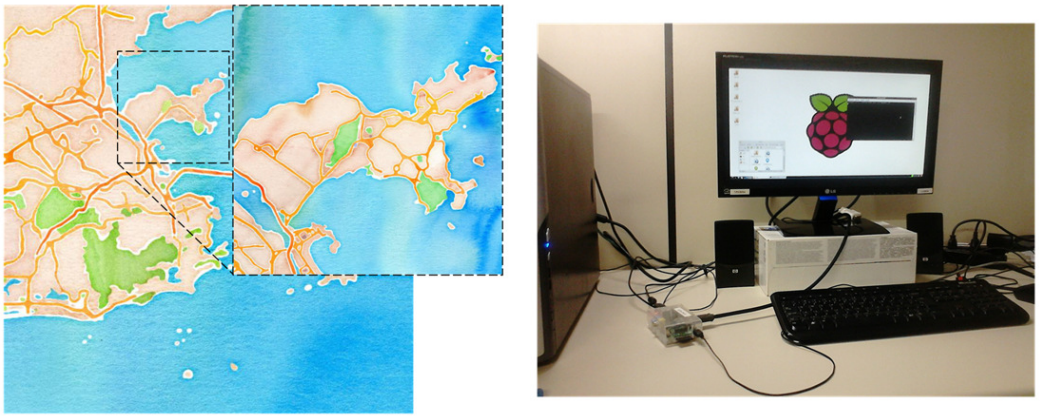


Figura 5.4: Metodologia dos experimentos.



(a) Mapa da Ilha do Governador. (b) Raspberry Pi B+.

Figura 5.5: Mapa e Raspberry Pi usados nos experimentos.

Fluxo do Tráfego Os dados do trânsito foram extraídos dos *traces* de 6.775 ônibus do Rio de Janeiro, em um período de 31 dias, de 1 de Outubro até 1 de Novembro de 2014. Para este experimento, foram utilizados 972.755 dados de GPS, com dez janelas de tempo a partir das 6 h até às 20 h referentes a um dia de coleta. Porém, somente as três janelas de tempo mais significativas foram representadas: 1) de 7 h até 8 horas; 2) de 11 h até 12 h; e 3) de 16 h até 17 h. O formato dos dados

nos arquivos dos *traces* são: data e horário, identificação do ônibus, linha, latitude, longitude e velocidade. Os horários são expressos em horas, minutos e segundos; latitude e longitude são expressos em graus de acordo com suas posições GPS; e a velocidade é expressa em quilômetros/hora.

GTAPlanning É uma ferramenta desenvolvida para análise e testes com algoritmos planejadores de rotas. O *software* é capaz de extrair algumas propriedades relacionadas à grafos, bem como realizar testes de planejamento de rotas dos mapas em questão. Atualmente *GTAPlanning* é capaz de ler mapas expressos em dois formatos *Dynamic GraphStream (DGS) e OSM*, sendo que o primeiro é referente à biblioteca GraphStream utilizada na ferramenta e o segundo é definido pelo *OpenStreetMap*.

Análise dos Dados Para demonstrar a viabilidade computacional dos algoritmos estudados, forma medidos o tempo de execução, memória consumida, quantidade de vértices referentes ao menor caminho encontrado, velocidade média, distância percorrida e tempo de chegada. Os caminhos gerados por cada algoritmo também são analisados em relação à frequência de uso dos trechos e as medidas de estatísticas PDF e CDF. Para o cálculo de menor caminho foram usadas duas ponderações nas arestas: a) quanto à distância, expressa em metros; e b) quanto ao tempo de percurso, ou seja, dada a velocidade no trecho (aresta), e a distância em metros, é calculado o tempo gasto para o deslocamento. Lembrando que a velocidade do trecho é calculada a partir das informações dos *traces* do Rio de Janeiro, e que para o cálculo da distância é realizado com base nos pontos GPS dos trechos e aplicados na *Fórmula de Haversine*. Foram sorteados duzentos pares de pontos (origem e destino) para cada medida, perfazendo um total de 400 resultados por algoritmo. Para os algoritmos de busca informada (A*, ARA* e ITS-ARA*), foi utilizada a distância Euclidiana como heurística e coeficiente heurístico $\varepsilon = 5.0$ para ARA* e ITS-ARA*. O modelo de Raspberry Pi utilizado nos experimentos foi o Model B+ [75] (Figure 5.5(b)), que contém um processador ARM de 700 MHz, 512 MB de RAM um cartão de memória de 4 GB, além do sistema operacional Raspbian e Java embarcado 8.

5.2.1 Ponderação da Distância

Os algoritmos foram avaliados por três diferentes grupos de métricas em três janelas de tempo: a) quanto ao desempenho, tempo de execução e memória consumida são considerados; b) quanto às rotas geradas, distância percorrida, velocidade média e tempo de chegada são estimados com base nas informações do fluxo dos ônibus; d) quanto ao grafo, quantidade de vértices componentes das rotas encontradas, bem

como a frequência com que cada trecho é utilizada por cada algoritmos e suas medidas estatísticas de PDF e CDF. Estas últimas medidas fornecem o comportamento dos algoritmos estudados. Os resultados são apresentados com intervalo de confiança de 95%. Nesta Seção o menor caminho é calculado usando a distância em metros como peso nas arestas. Estas medidas foram extraídas das informações do *OpenStreetMap*, que informa latitude e longitude dos pontos GPS. Conforme mencionado anteriormente, foi usada a *Fórmula de Haversine* para calcular essa medida entre dois pontos em uma superfície esférica de coordenadas.

Figura 5.6: Ilha do Governador – Distância.

A Figura 5.6 mostra um exemplo de planejamento de rotas gerado pelos algoritmos estudados. Os pinos azul e verde são usados para marcar a origem e o destino, respectivamente. Neste exemplo, o resultado de Dijkstra e A* foi o mesmo, bem como ITS-ARA* e ARA* tiveram o mesmo comportamento, conforme esperado. Dessa forma, Dijkstra e A* foram representados pelo traço azul e consequentemente, ITS-ARA* e ARA* pelo traço verde. A Figura 5.7 mostra os mesmos pontos de origem e destino da figura anterior, porém a ponderação utilizada é o tempo de percurso do trecho. Podem-se notar as diferenças entre as duas ponderações, a seguir são apresentados dados que comprovam o comportamento observado.

A Tabela 5.4 mostra os resultados dos algoritmos para cada janela de tempo. Os resultados são expressos em: a) Tempo de execução – média do tempo de execução, dado em milissegundos; b) Memória consumida – média da memória consumida expressa em MB; e c) Vértices – média da quantidade de vértices componentes do caminho encontrado. Pode-se notar que os algoritmos de busca informada (A*, ARA* e ITS-ARA*) foram mais rápidos que o Dijkstra, sendo que em termos gerais

Figura 5.7: Ilha do Governador – Tempo de percurso.

Janela de Tempo	Distância			
	DJK	**A***	**ARA***	**ITS-ARA***
	Tempo de execução (ms)			
07–08 am	366.85	125.04	22.63	21.70
11–12 am	353.62	118.98	23.05	22.71
04–05 pm	353.59	108.43	23.71	22.88
	Memória Consumida (MB)			
07–08 am	159.68	159.77	159.34	159.64
11–12 am	157.39	156.66	156.11	156.22
04–05 pm	157.06	155.90	156.46	156.77
	Vértices			
07–08 am	84.60	84.69	102.00	102.00
11–12 am	83.24	83.60	101.23	101.23
04–05 pm	82.68	82.71	100.87	100.87

Tabela 5.4: Resultados quanto à Distância – Tempo de execução (ms), memória consumida (MB) e quantidade de vértices.

ITS-ARA* foi o mais rápido em todas as janelas de tempo. O ganho médio de ITS-ARA* foi em torno de 93,73% em relação ao algoritmos de Dijkstra, de 80,82 % para o A* e 3,02% em relação ao ARA*. Quanto à memória consumida, os quatro algoritmos tiveram um desempenho bastante próximo, considerando o intervalo de confiança de 95%. Quanto à quantidade de vértices, os resultados mostraram que Dijkstra e A* foram mais eficientes, em média em torno de 17% em relação aos demais.

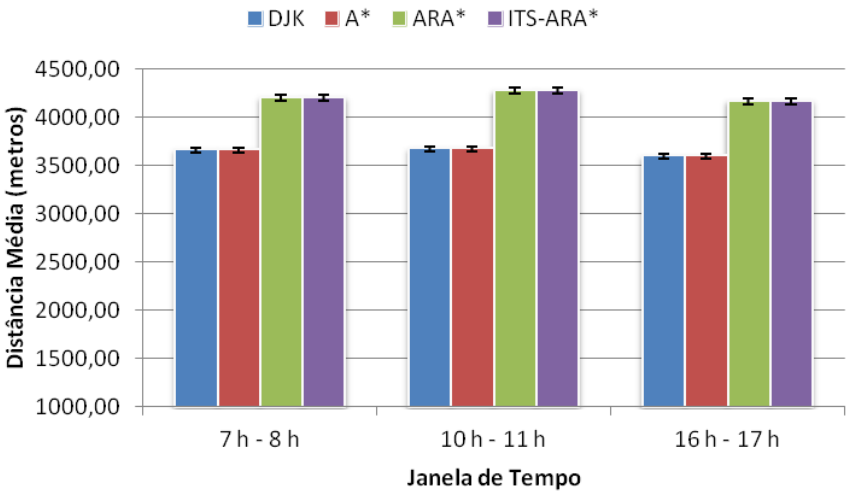


Figura 5.8: Média da Distância Percorrida (metros) – Ponderação de Distância.

A Figura 5.8 apresenta a média da distância percorrida, dentre os menores caminhos, quanto à ponderação da distância expressa em metros. Em princípio, cada algoritmo busca as arestas com menores distâncias para formação de seus resultados (*shortest path*). Em seguida, são extraídas as informações de tempo de percurso da aresta (em minutos) e distância (em metros), de modo que basta aplicar a fórmula para calcular a velocidade.

Os resultados demonstram o melhor desempenho do Dijkstra e do A* em relação à média da distância percorrida: 12,87% (diferença de 541,09 metros) na janela de tempo de 7 horas até às 8 horas da manhã; 14,06% (600,83 metros) na janela de tempo de 11 horas até às 12 horas; e 13,71% (diferença de 571,35 metros) para a janela de 16 horas até às 17 horas.

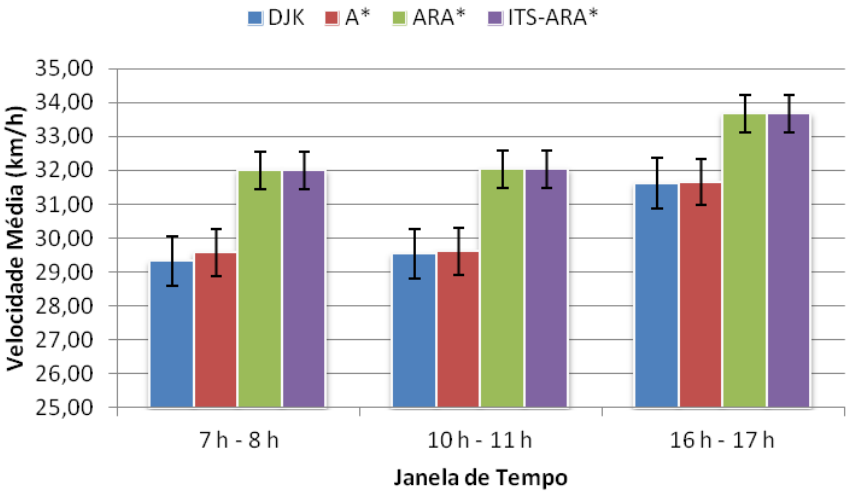


Figura 5.9: Média da Velocidade (km/h) – Ponderação de Distância.

Para a métrica da velocidade média, os algoritmos ITS-ARA* e ARA* obtiveram melhor desempenho em média sobre os demais (Figura 5.9). A diferença sobre os algoritmos de A* e do Dijkstra para as janelas de tempo foram de: 8,39% (2,68 km/h) para 7 hora até às 8 horas; 7,82% (2,51 km/h) para 11 horas até 12 horas; 6,09% (2,05 km/h) 16 horas até às 17 horas.

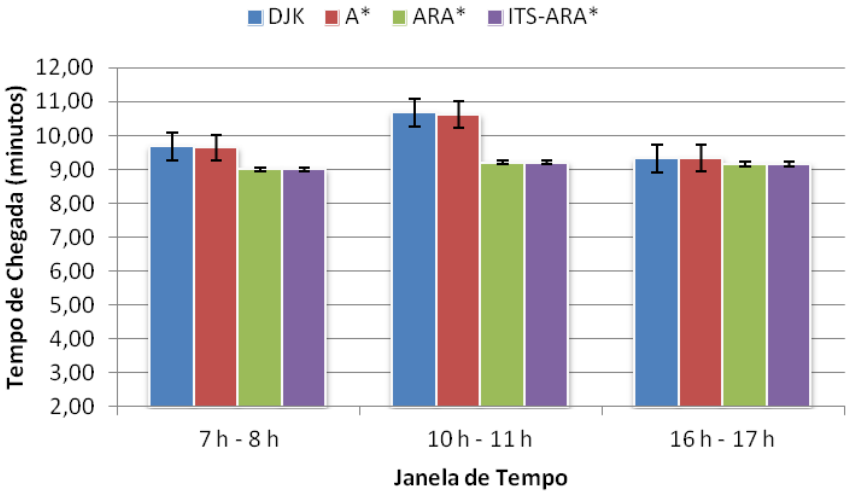


Figura 5.10: Média do Tempo de Chegada (minutos) – Ponderação de Distância.

A média do tempo de chegada nos resultados (Figura 5.10) demonstrou que os algoritmos ITS-ARA* e ARA* encontraram os melhores resultados somente na janela de tempo de 10 horas até às 11 horas, com 13,8% de vantagem sobre os demais, ou seja, 1,47 minutos. Pode-se observar que outros resultados estão dentro do intervalo de confiança. A seguir, são apresentados os resultados dos trechos mais utilizados pelos algoritmos em relação à frequência de uso de cada algoritmo.

Para estes resultados foi feita uma normalização dos identificadores dos caminhos trazidos nos arquivos do *OpenStreetMap*, para melhor apresentação dos mesmos. O processo de normalização deu-se da seguinte forma. Inicialmente todos os resultados de menor caminho foram postos em um único arquivo, ou seja, todos os identificadores de trecho (aresta) foram reunidos. Em seguida foi realizada uma ordenação e uma filtragem dos mesmos identificadores, com intuito de eliminar as repetições. A última etapa deu-se na criação de um novo identificador para cada trecho do mapa na lista. Com base neste arquivo de padronização, foi realizado um mapeamento dos resultados obtidos por cada algoritmo de acordo com a janela de tempo. Por fim, foram extraídas as frequências de cada trecho (aresta) do mapa de acordo com o seu respectivo algoritmo e janela de tempo.

Nos resultados, os algoritmos de Dijkstra e A* tiveram os melhores resultados em relação a distância percorrida conforme esperado, bem como ITS-ARA* e ARA* foram os mais rápido no planejamento de rotas. Os resultados a seguir indicam os

trechos de ruas e de avenidas que foram os mais utilizados nos experimentos. Esta informação é útil para ajudar a prevenir congestionamentos quando técnicas de planejamento de rotas como as apresentadas neste trabalho forem colocadas em prática. Outra informação que pode ser notada, é a observação de rotas alternativas geradas pelos algoritmos ITS-ARA* e ARA* em questão, bem como a diferença nas frequências dos trechos utilizados entre os algoritmos de Dijkstra/A* e ITS-ARA*/ARA*. Raramente os algoritmos possuem a mesma frequência nas três janelas de tempo, sugerindo desta forma a adoção de rotas alternativas, sendo que o identificador mais utilizado foi o trecho 86, com uma maior frequência de 18 para os algoritmos de ITS-ARA*/ARA*, conforme pode ser visualizado na Figura 5.12.

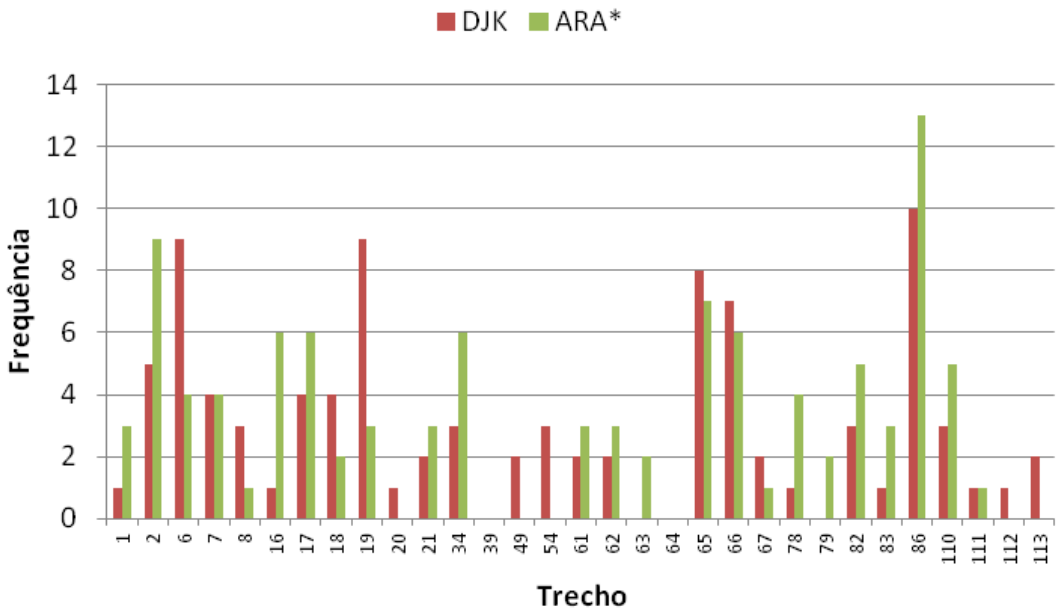


Figura 5.11: Trecho – 7 horas até 8 horas – Ponderação de Distância.

As Figuras 5.11, 5.12 e 5.13 mostram os comportamentos de cada algoritmo estudado de acordo com as frequências dos trechos escolhidos nas diferentes janelas de tempo (de 7 horas até 8 horas, 11 horas até 12 horas e de 16 horas até 17 horas, respectivamente). Pode-se observar que o trecho 6 é o mais utilizado por Dijkstra e A* nas três janelas de tempo estudadas. Fazendo uma correspondência do trecho 6 em questão com sua localização geográfica, tem-se que corresponde a uma parte da estrada da Cacuia até uma parte da Estrada do Galeão, um importante trecho da Ilha do Governador. Analisando o mesmo trecho 6 para os algoritmos de ITS-ARA* e ARA*, observa-se uma diferença na frequência de sua utilização, sendo respectivamente de 4, 5 e 4 vezes nas três janelas de tempo. Outro trecho com comportamento similar é o 19, onde sua frequência inicia com 9, em seguida desce para 5 e finaliza em 9 para os algoritmos de Dijkstra e de A*, para as três janelas de

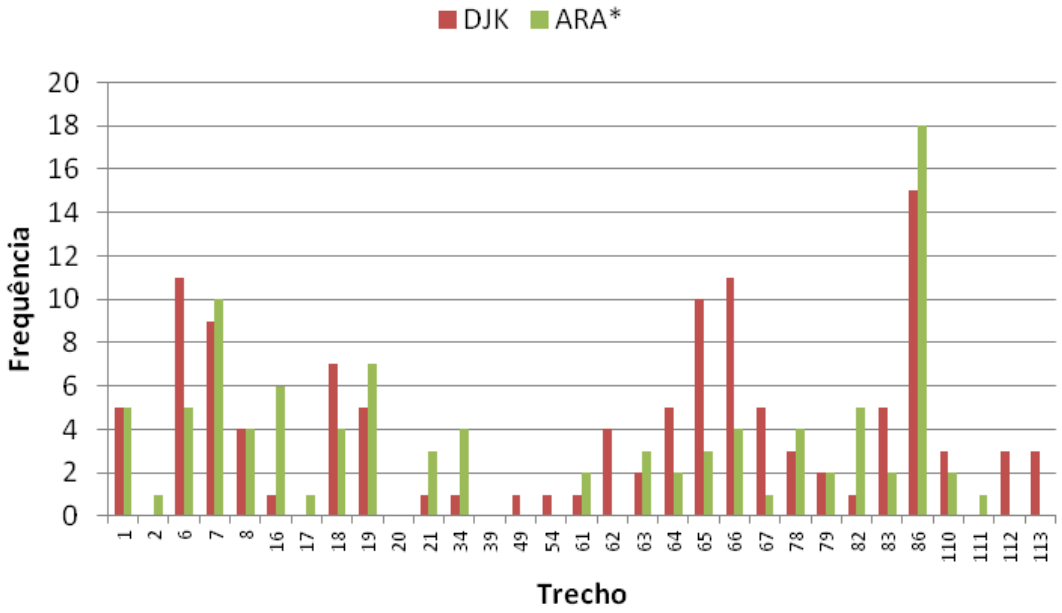


Figura 5.12: Trecho – 11 horas até 12 horas – Ponderação de Distância.

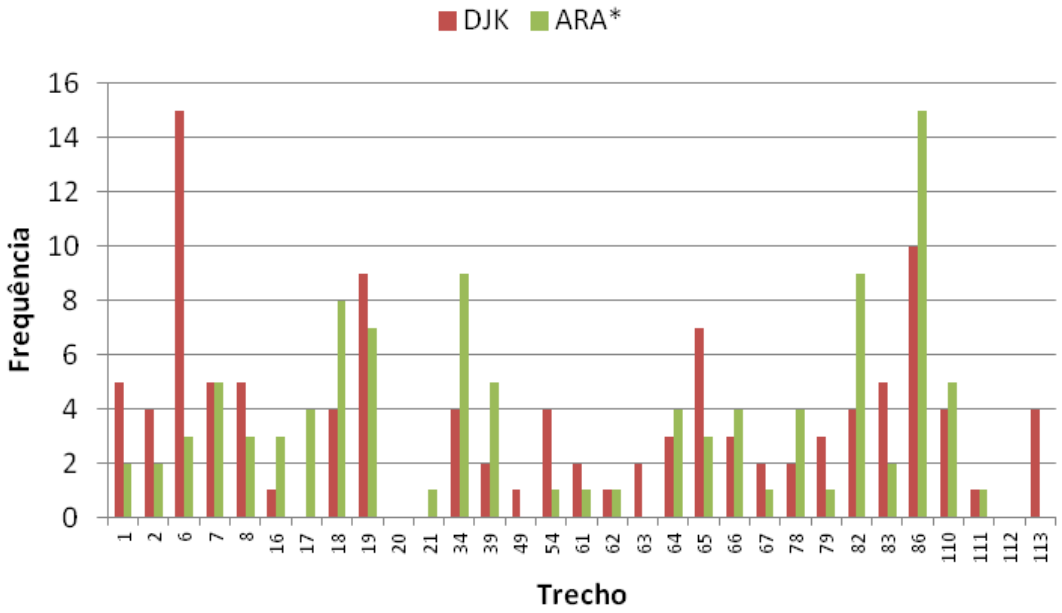


Figura 5.13: Trecho – 16 horas até 17 horas – Ponderação de Distância.

tempo respectivamente. Já para os algoritmos de ITS-ARA* e ARA*, o trecho 19 inicializa em 3 e estabiliza com frequência de 7. O trecho em questão corresponde a uma parte da Avenida Paranapuã, uma importante avenida que liga quatro bairros residenciais da Ilha do Governador, são eles: Bancários, Cocotá, Moneró e Jardim Carioca. Existem também trechos utilizados somente por um par de algoritmos como, por exemplo os trechos 49 e 113, usados somente por Dijkstra e A*, com

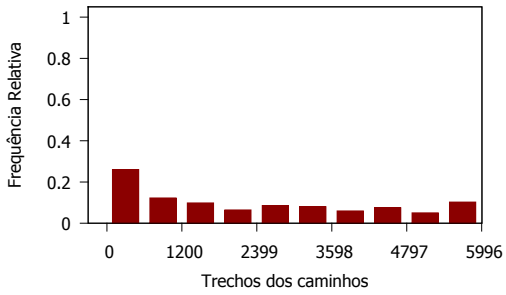
as respectivas frequências em relação às janelas de tempo: de 2, 1 e 1 vez para o trecho 49, que corresponde a uma parte da rua Castorina Faria de Lima no bairro da Portuguesa; e de 2, 3 e 4 vezes para o trecho 113, que corresponde à Rua Costa Dória, bairro Bancários. Outro exemplo a ser citado é o que corresponde ao trecho 86, que referencia a Estrada do Galeão e que foi bastante utilizado pelos algoritmos ITS-ARA* e ARA*, respectivamente para as janelas de tempo com: 13, 18 e 15 vezes.

Com intuito de visualizar o comportamento dos algoritmos de forma mais geral em relação aos caminhos escolhidos por cada um deles, a seguir serão apresentadas as análises das rotas geradas pelos pares de origem e destino sorteados de acordo com sua respectiva janela de tempo. Para isto foram usadas as funções de densidade de probabilidade (*PDF*) para analisar os comportamentos dos algoritmos. Os resultados da PDF foram divididos em 10 intervalos, onde cada intervalo representa a frequência com que cada trecho foi utilizado nos experimentos. As Figuras 5.14, 5.15 e 5.16 apresentam os resultados das PDFs dos algoritmos de acordo com suas respectivas janelas de tempo. Pode-se notar que os resultados dos algoritmos *Anytime* (ITS-ARA* e ARA*) possuem uma distribuição de frequências melhor distribuída em relação aos resultados de Dijkstra e A*, proporcionando dessa forma, equilíbrio ao sistema de rotas. Esta última característica é importante pois privilegia o balanceamento de veicular.

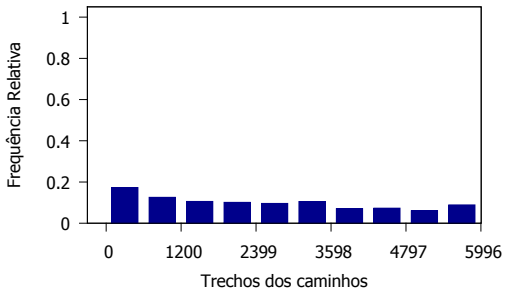
Os resultados de A* e Dijkstra foram praticamente os mesmos em relação às frequências nos intervalos e em suas respectivas janelas de tempo, bem como de ITS-ARA* e ARA*. Esse era um comportamento esperado conforme observado nos resultados anteriormente estudados. Uma característica que pode ser notada nos resultados é a grande frequência na utilização do primeiro intervalo, mais precisamente uso de 25% para os algoritmos de Dijkstra (Figuras 5.14(c), 5.15(c) e 5.16(c)) e A* (Figuras 5.14(a), 5.15(a) e 5.16(a)). No entanto, para o intervalo em questão, os algoritmos ITS-ARA* (Figuras 5.14(d), 5.15(d) e 5.16(d)) e ARA* (Figuras 5.14(b), 5.15(b) e 5.16(b)) o utilizaram 17% nas três janelas de tempo.

Na primeira janela de tempo (de 7h até às 8 h), pode-se observar ainda que os resultados de Dijkstra e A*, tiveram maior concentração nos três primeiros intervalos, mais precisamente 47% e no último intervalo é utilizado com 12%, perfazendo um total de 59% de frequência de utilização. Para os resultados de ITS-ARA* e ARA*, observa-se que as maiores frequências observadas se concentram nos dois primeiros intervalos e nos intervalos de quatro a seis, com percentuais de 29% e 32% respectivamente, correspondendo ao total de 61% da frequência.

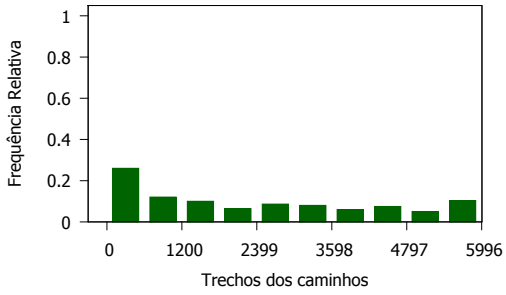
Na segunda janela de tempo, pode-se observar uma diminuição na frequência em comparação com o intervalo anterior, sendo que para os três primeiro intervalos correspondem a utilização de 45% e no último de 10%, perfazendo um total de 55%



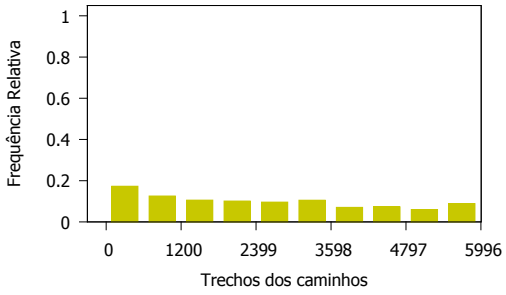
(a) PDF A*.



(b) PDF ARA*.

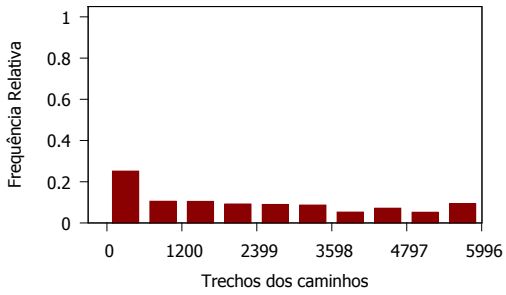


(c) PDF Dijkstra.

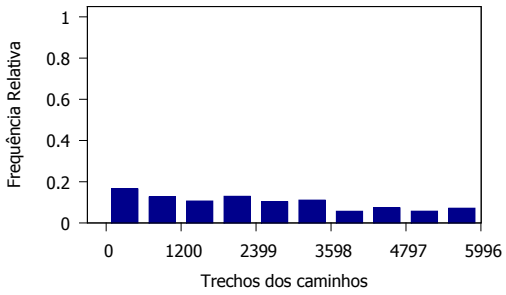


(d) PDF ITS-ARA*.

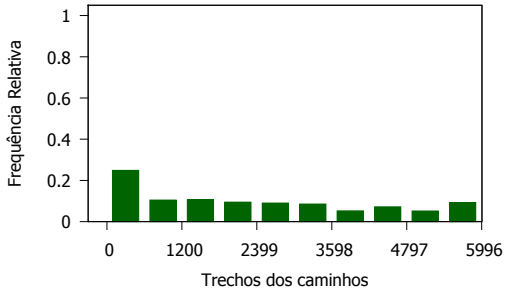
Figura 5.14: PDF algoritmos – 07 h até 08 h – Distância.



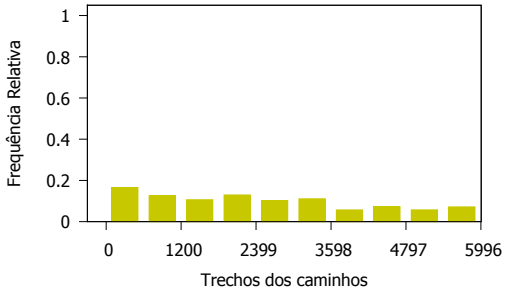
(a) PDF A*.



(b) PDF ARA*.



(c) PDF Dijkstra.

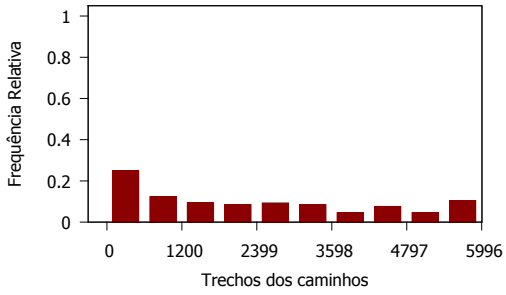


(d) PDF ITS-ARA*.

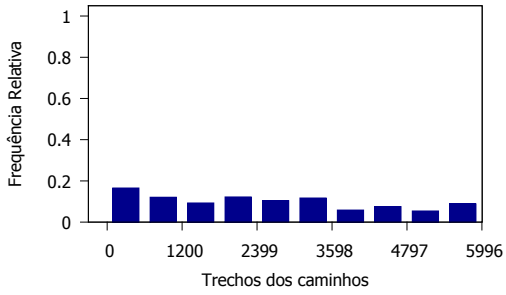
Figura 5.15: PDF algoritmos – 11 h até 12 h – Distância.

da frequência do todo. Para os resultados de ITS-ARA* e ARA*, verificou-se que as maiores concentrações de frequência estão nos dois primeiro intervalos e nos intervalos de três a cinco, respectivos valores percentuais de 29% e 34%, correspondendo ao

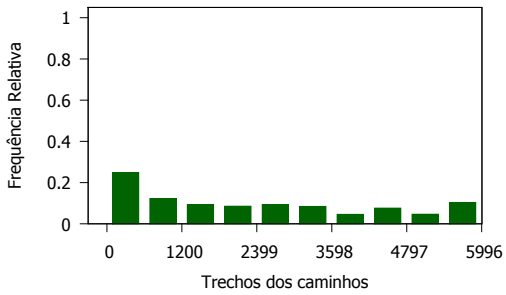
total de 63%. Pode-se observar ainda, que houve um acréscimo de 2% nos intervalos de três a cinco em relação à janela de tempo anterior (de 7h até 8h).



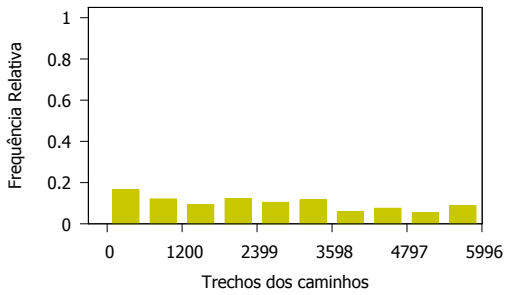
(a) PDF A*.



(b) PDF ARA*.



(c) PDF Dijkstra.



(d) PDF ITS-ARA*.

Figura 5.16: PDF algoritmos – 16 h até 17 h – Distância.

Para a última janela de tempo (de 11 h até às 12 h), observou-se que a maior frequência utilizada por Dijkstra e A* manteve-se nos três primeiros intervalos correspondendo a 47%, bem como ao último intervalo com 10%, perfazendo um total de 57%. Nos resultados de ITS-ARA* e ARA* na última janela de tempo (de 16 h até às 17 h), as concentrações continuaram constantes em relação aos resultados anteriores para os dois intervalos iniciais com 29%. Porém, para os intervalos de três a cinco, houve um decréscimo de dois pontos percentuais, mais precisamente 32%. Desta forma, o resultado total dos intervalos corresponde a 61% da frequência total. Os resultados de análise das CDFs relacionados a este experimento podem ser visualizados no Apêndice B.

5.2.2 Ponderação do Tempo do Percurso

Para os resultados nesta Seção, o menor caminho foi calculado usando os dados das velocidades dos ônibus do Rio de Janeiro, onde foram filtradas as informações pertinentes a Ilha do Governador com base nas coordenadas geográficas. Conforme mencionado anteriormente, para este experimento é utilizado o tempo de percurso por trecho (aresta) como ponderação no planejamento de rotas.

Nestes experimentos foi usado o tempo de percurso, que é o tempo gasto para ir de um ponto a outro, sendo que o mesmo é calculado baseado nas informações de tráfego da cidade do Rio de Janeiro, sendo que cada valor é atribuído como ponderação nas suas respectivas arestas, valor utilizado no cálculo do planejamento de rotas. A Tabela 5.5 apresenta os resultados do planejamento de rotas em relação ao tempo de execução, memória consumida e quantidade de vértices em relação à ponderação de tempo de percurso. O ITS-ARA* foi o mais rápido entre os algoritmos estudados, obtendo ganhos em média de 93,99% em relação ao Dijkstra, 83,77% em relação ao A* e de 3,11% em relação ao ARA*. Porém, para quantidade de vértices e memória consumida, levando-se em consideração o intervalo de confiança, os resultados foram os mesmos.

Janela de Tempo	**Tempo de Percurso**			
	DJK	**A***	**ARA***	**ITS-ARA***
	Tempo de Execução (ms)			
07–08 am	377.92	157.01	23.53	22.53
11–12 am	420.33	146.12	23.95	23.61
04–05 pm	367.85	130.79	24.67	23.77
	Memória Consumida (MB)			
07–08 am	159.64	160.97	158.99	160.01
11–12 am	155.60	157.39	155.90	155.92
04–05 pm	157.29	155.73	156.98	156.42
	Vértices			
07–08 am	118.49	118.49	116.15	116.15
11–12 am	124.35	124.35	123.69	123.69
04–05 pm	121.29	121.29	122.48	122.48

Tabela 5.5: Resultados do tempo de execução (ms), memória consumida (MB) and vértices - Tempo de Percurso.

A Figura 5.17 mostra a distância média percorrida em relação à janela de tempo, pode-se observar que não existe diferença entre os algoritmos nos três intervalos listados. Pode-se notar que a menor distância percorrida encontra-se na primeira janela de tempo (7 horas até às 8 horas), em torno de 5.400 metros. Pode-se notar ainda, que os resultados da distância percorrida nesta ponderação, foram maiores que os resultados da ponderação de distância conforme esperado. Outra observação, considerando os intervalos de confiança nas três janelas de tempo, os algoritmos tiveram o mesmo comportamento, contrariamente aos resultados obtidos utilizando a ponderação da distância.

A velocidade média é outra medida que confirma a tendência descrita anteriormente, onde nas três janelas de tempo os algoritmos tiveram comportamentos similares (Figura 5.18), onde a maior média obtida foi registrada no terceira janela

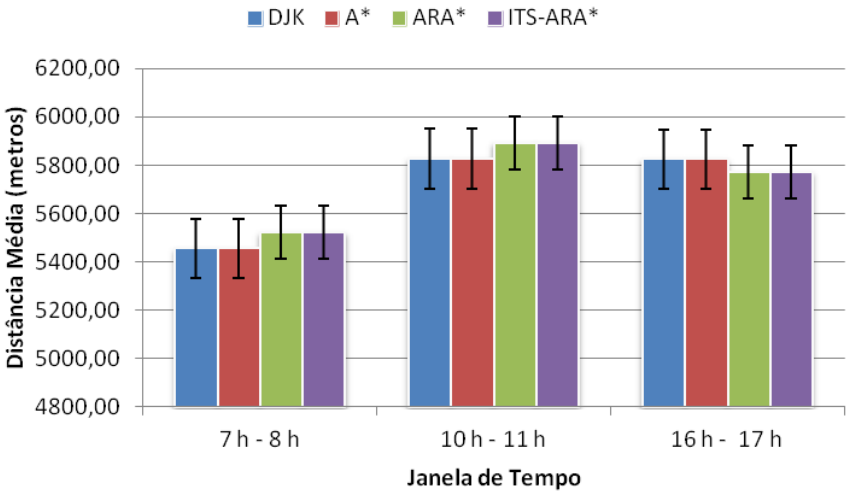


Figura 5.17: Distância Média (metros) – Tempo de Percurso.

de tempo (16 horas até às 17 horas, ou seja, 60 km/h).

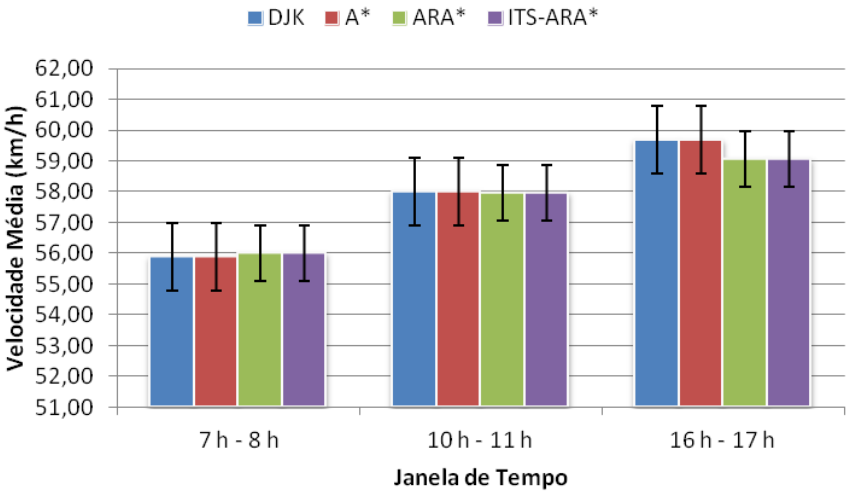


Figura 5.18: Velocidade Média (km/h) – Tempo de Percurso.

A Figura 5.19 mostra a média do tempo de chegada, onde o maior valor registrado foi em torno de 6 minutos. Entretanto, os resultados mostram que não houve diferença entre os algoritmos, levando-se em consideração os intervalos de confiança, sendo que a terceira janela de tempo representa bem este comportamento. Dessa forma, levando-se em consideração este resultado e os demais (distância percorrida e velocidade média), observa-se diferença mínima entre os algoritmos, podendo portanto, a ponderação ser aplicável, além de favorecer o balanceamento no tráfego veicular nas cidades. Pois pode-se usar qualquer um dos algoritmos, que seus resultados serão bastante próximos.

Outra característica que pode ser notada na ponderação de tempo de percurso é a

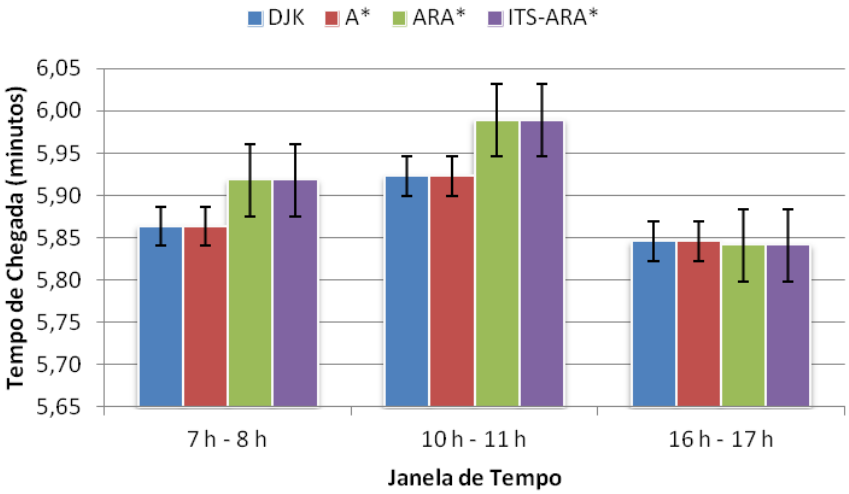


Figura 5.19: Média do Tempo de Chegada (minutos) – Tempo de Percurso.

alta frequência de uso nos trechos por parte dos algoritmos estudados, característica que diferencia da ponderação de distância, sendo que o valor máximo observado foi de 45. Isto se deve à maior uniformidade dos pesos nas arestas, diferentemente do que ocorre na ponderação de distância, ou seja, a variação nos valores quanto ao tempo de percurso é menor que o comprimento nos trechos das vias. Como consequência, os algoritmos têm desempenho similares. Pode-se observar que o Trecho 847 mostra uma interessante diminuição na sua frequência, iniciando em 45 nas duas primeiras janelas (Figuras 5.20 e 5.21), e em seguida decrescendo para 37 de frequência (Figura 5.22). O Trecho 847 corresponde à Rua Astilbe no Jardim Carioca (Ilha do Governador). O Trecho 1006 correspondente a uma parte da Estrada do Galeão, que tem um limite de velocidade de 80 km/h, e apresenta frequência em torno de 45 com pouca variação. O Trecho 1007 apresenta frequências de 45 para as duas primeiras janelas de tempo (7 h até 8 h e de 11 h até 12 h), e por fim apresenta frequência de 35 na última janela (16 h até 17 h).

A seguir as frequência dos trechos escolhidos pelos algoritmos foram analisadas de acordo com as PDFs das respectivas janelas de tempo. Da mesma forma que a ponderação anterior, nesta análise foram considerado dez intervalos com seus respectivos percentuais de frequência de utilização. As Figuras 5.23, 5.24 e 5.25 apresentam os resultados das PDFs dos algoritmos de acordo com suas respectivas janelas de tempo. Pode-se notar que ao contrário da ponderação em relação à distância, os resultados em relação ao tempo de percurso demonstraram que não houve diferenças entre os algoritmos, pois os mesmos apresentaram o mesmo comportamento nas três janelas de tempo, o que corrobora os resultados anteriores.

Na primeira janela de tempo (de 7 h até às 8 h), nota-se que a maior frequência em Dijkstra (Figura 5.23(c)), A* (Figura 5.23(a)), ITS-ARA* (Figura 5.23(d)) e ARA*

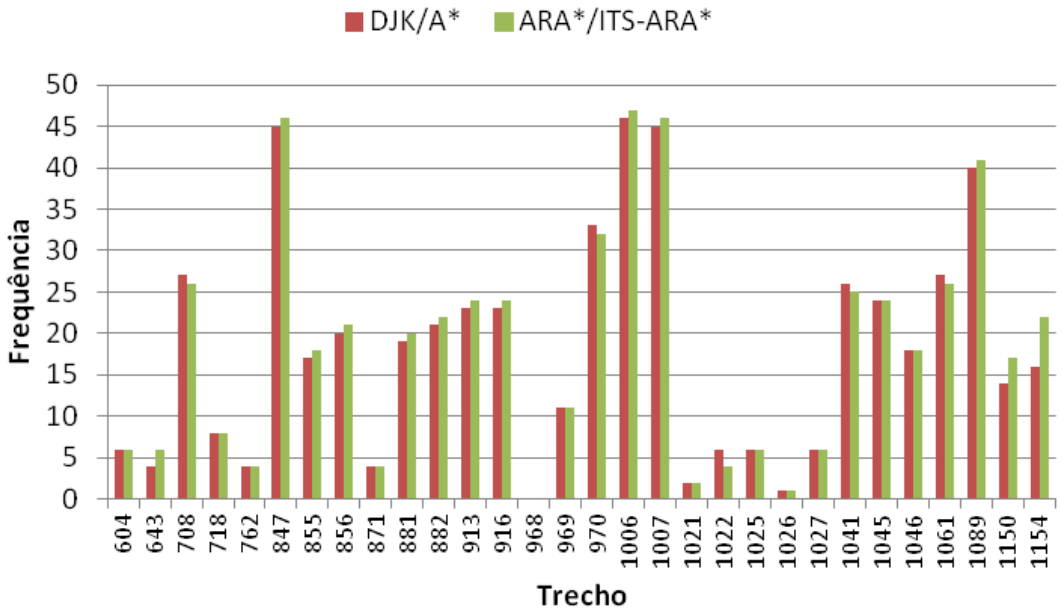


Figura 5.20: Trecho – 7 h até 8 h – Tempo de Percurso.

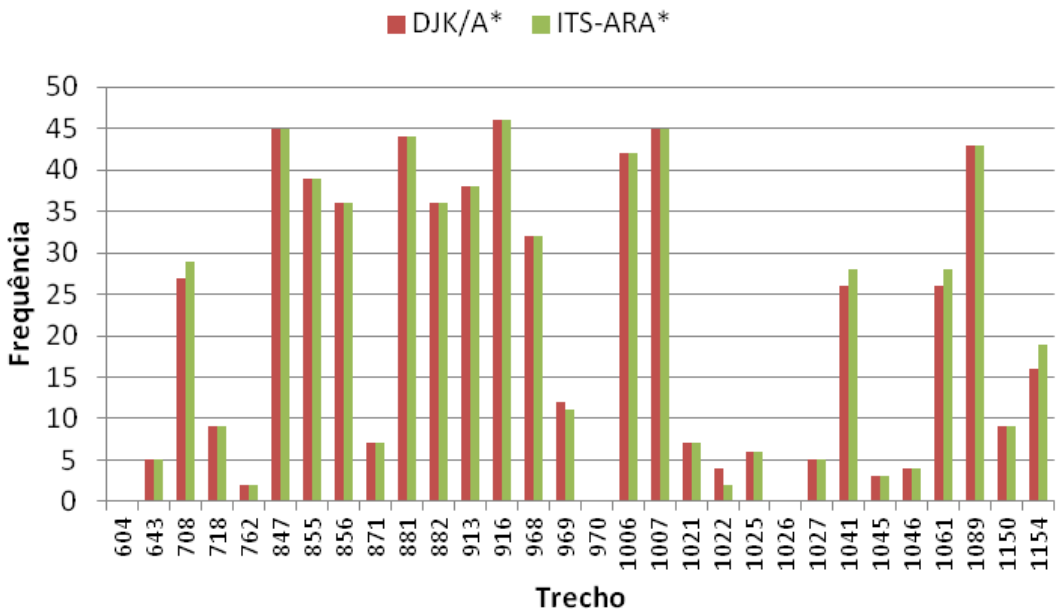


Figura 5.21: Trecho – 11 h até 12 h – Tempo de Percurso.

(Figura 5.23(b)) apresentou-se nos três primeiros intervalos de trechos, perfazendo um total de 50% para o dado intervalo de tempo.

No segundo intervalo de tempo, das 11 h até às 12 h, pode-se notar que houve uma alteração nas frequências correspondentes aos algoritmos em questão, de modo que os intervalos que mais se sobressaem são o primeiro e o terceiro intervalo, que correspondem a 43% da frequência total para Dijkstra (Figura 5.24(c)), A* (Fi-

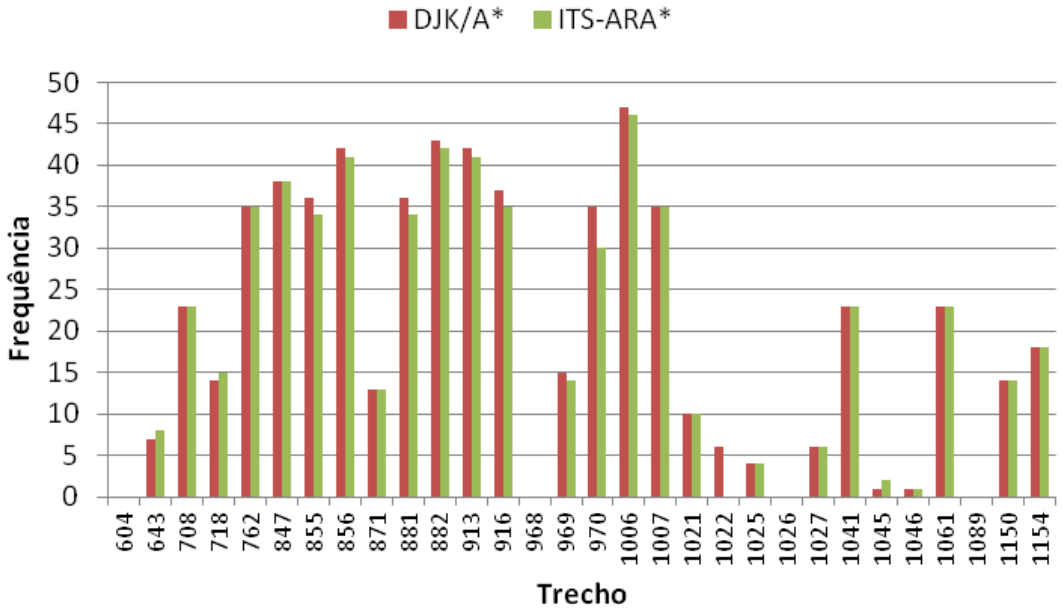
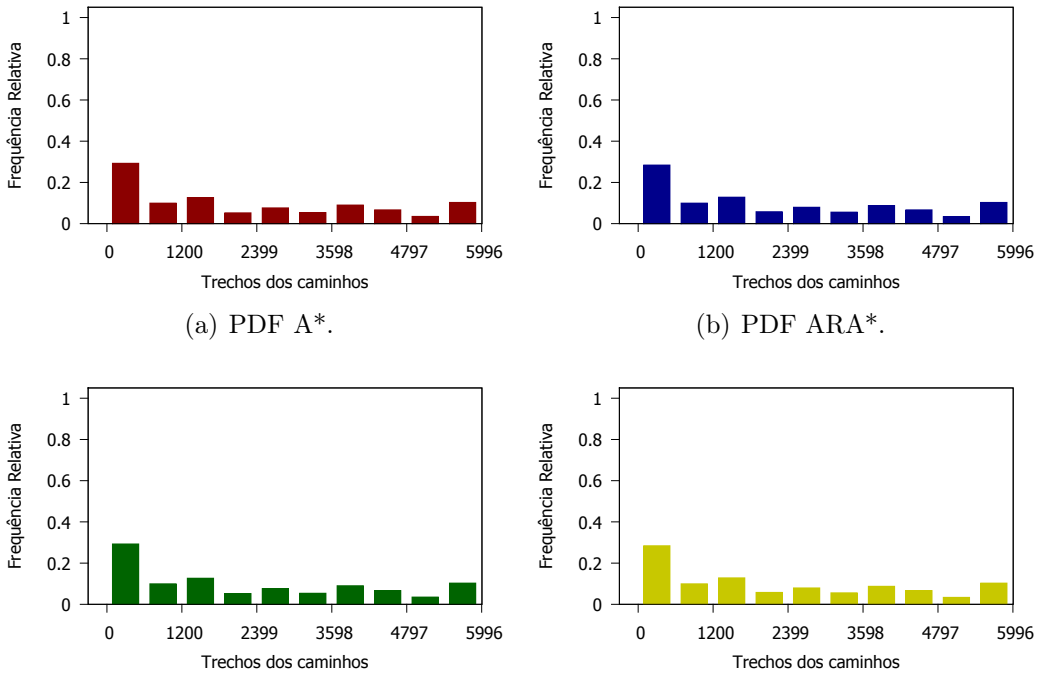


Figura 5.22: Trecho – 16 h até 17 h – Tempo de Percurso.



(a) PDF A*.

(b) PDF ARA*.

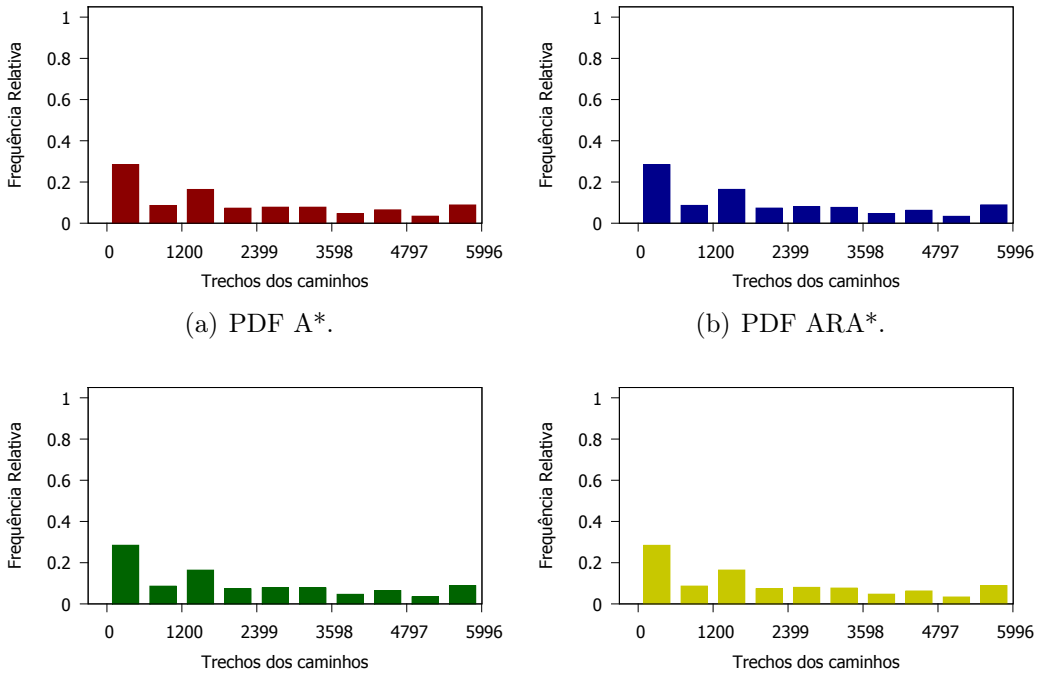
(c) PDF Dijkstra.

(d) PDF ITS-ARA*.

Figura 5.23: PDF algoritmos – 07 h até 08 h – Tempo de Percurso.

gura 5.24(a)), ITS-ARA* (Figura 5.24(d)) e (Figura 5.24(b)).

Na última janela de tempo (de 16 h até às 17 h), nota-se que Dijkstra ((Figura 5.25(c))), A* ((Figura 5.25(a))), ITS-ARA* ((Figura 5.25(d))) e ARA* ((Figura 5.25(b))) tiveram um decréscimo de 3% em relação à janela anterior. Portanto,



(a) PDF A*.

(b) PDF ARA*.

(c) PDF Dijkstra.

(d) PDF ITS-ARA*.

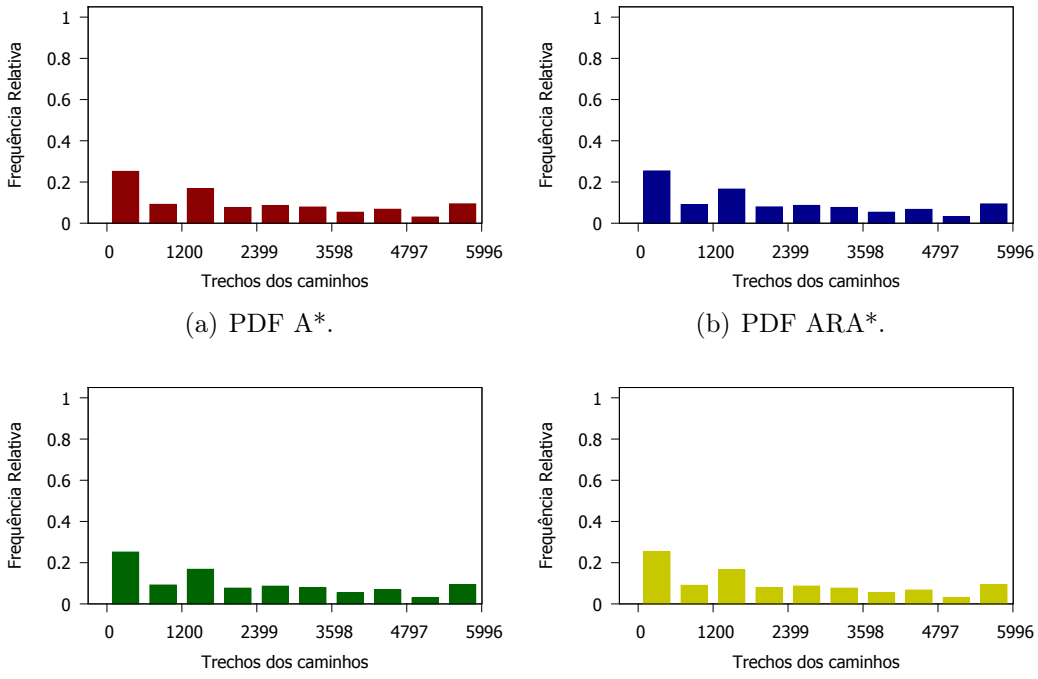
Figura 5.24: PDF algoritmos – 11 h até 12 h – Tempo de Percurso.

o primeiro e o terceiro intervalo correspondem a 40% da frequência dos trechos utilizados no referido intervalo de tempo. Os resultados CDF em relação ao tempo de percurso, estão disponíveis no Anexo B, da mesma forma que a ponderação de distância.

5.2.3 Análise dos Resultados

Este experimento apresentou a análise de desempenho dos algoritmos para o planejamento de rotas na rede viária do Rio de Janeiro, mais precisamente na região da Ilha do Governador, onde foram utilizados dados reais do fluxo de tráfego dos ônibus da referida cidade. Foram utilizados quatro algoritmos de planejamento de rotas (Dijkstra, A*, ITS-ARA* e ARA*) para análise dos resultados. Foi utilizado o computador Raspberry Pi modelo B+ como plataforma de testes, bem como foram utilizadas dois tipos de ponderação nas arestas: distância e tempo de percurso. O mapa da Ilha do Governador foi extraído da plataforma colaborativa *OpenStreetMap*. Em relação ao tempo de execução, em média ITS-ARA* foi o mais rápido entre os algoritmos estudados, cerca de 15,98 vezes mais veloz que Dijkstra, 5,25 vezes que A* e 1,03 mais rápido que ARA*. Para a ponderação de tempo de percurso, os valores foram de 16,69 vezes mais rápido que Dijkstra, 6,22 vezes que A* e 1,03 vezes mais eficiente que ARA*.

Outra característica notada nos resultados foi a heterogeneidade quanto à pon-



(a) PDF A*.

(b) PDF ARA*.

(c) PDF Dijkstra.

(d) PDF ITS-ARA*.

Figura 5.25: PDF algoritmos – 16 h até 17 h – Tempo de Percurso.

deração, enquanto que nos resultados da distância percebe-se claramente que os algoritmos *Anytime* tendem a serem mais distribuídos, favorecendo o equilíbrio do sistema. Na ponderação de tempo de percurso, não houve diferenças entre os resultados dos algoritmos, o que leva a concluir que qualquer algoritmo pode ser utilizado para o respectivo contexto, porém o ITS-ARA* foi o mais rápido no cálculo do melhor caminho.

Mais uma vez, observou-se que os algoritmos ITS-ARA* e ARA*, mantêm a velocidade e a eficiência de A*, agregando o reuso dos resultados encontrados caso ocorra alterações nas ponderações das arestas (trechos), sendo uma característica essencial em ambientes dinâmicos como o trânsito. Os resultados experimentais foram satisfatórios em relação ao desempenho dos algoritmos ITS-ARA* e ARA* na produção de rotas alternativas sensíveis ao contexto, bem como a utilização da plataforma Raspberry Pi no planejamento de rotas em redes viárias.

Capítulo 6

Planejamento de Rotas sem Restrições Computacionais

Nesta seção são estudadas setes redes viárias, sendo três da América do Sul, três europeias e uma da América do Norte. A utilização dos referidos experimentos deveu-se pela busca de características semelhantes para uma possível classificação de tipos de redes viárias. Os mapas utilizados foram das cidades de Manaus, Rio de Janeiro, Belo Horizonte, Le Havre, Munique e Nova Iorque. Os mapas foram extraídos a partir do projeto livre de mapeamento colaborativo *OpenStreetMap* e de arquivos em *DGS* encontrados por meio de pesquisas na Internet. Os testes foram realizados utilizando um computador Intel Core i5-3330 com 3.00 GHz, com 16 GB RAM e HD de 500 GB, a versão Java SE 64 1.8.0_31 e o Sistema Operacional Debian GNU/Linux 7 (wheezy). Os experimentos utilizaram a ferramenta *GTAPlanning* e estudos seguiram a seguinte metodologia:

- **Leitura do mapa viário** – para este experimento foram utilizados arquivos *.dgs* além dos já utilizados *.osm*, sendo que para este formato o *GraphStream* já possui implementações sobre de leitura e interpretação dos mesmos, não necessitando portanto de outros artifícios. Desta forma, é possível selecionar entre os dois formatos permitido, o *.osm* e o *.dgs*. Mais duas propriedades são medidas pelo *software* neste experimento, além das quantidades de vértices, arestas e o diâmetro da rede viária, são introduzidos à média do coeficiente de *clusters* e a densidade do grafo.

- **Ponderação aleatória das arestas** – o procedimento de ponderação é o mesmo utilizado no experimento anterior, com variações são até 80.

- **Testes dos algoritmos de planejamento** – os algoritmos utilizados nos experimentos foram Dijkstra, A*, ARA* e ITS-ARA*, sendo portanto os mesmos do experimento com o Raspberry Pi. No entanto, o coeficiente heurístico (ϵ)

presente nos algoritmos *Anytime* ARA* e ITS-ARA*, é variado de 5,0 até 1,0 em intervalos decrescentes de 0,4, conforme o experimento descrito em Moura *et al.* [58] (Capítulo 3).

- **Resultados dos Testes** – foram gerados no total quarenta e sete arquivos de medidas, sendo quarenta e quatro arquivos de dados referentes aos algoritmos ARA* e ITS-ARA* com a influência da heurística. Cada arquivo de dados contendo as seguintes medidas: a) quantidade de vértices; b) quantidade de arestas; c) tempo de execução. Foram criados três experimentos com pontos de origem e destino diferentes para cada um dos resultados.

As características dos grafos estudados são apresentados na Tabela 6.1. Estas características serão utilizadas futuramente para o desenvolvimento de uma relação entre o grafo estudado e o coeficiente heurístico utilizado nos algoritmos da família *Anytime* estudada, isto se deve pelo fato que na literatura o mesmo precisa ser encontrado por meio de tentativas. É óbvio que no contexto da robótica de onde o ARA* é originário, a determinação de ϵ seja muito difícil de conseguir avaliar, pois não se conhece o ambiente a ser explorado. Porém isto não acontece nas redes viárias, onde o grafo é relativamente conhecido, sendo portanto válido buscar preencher esta lacuna.

Rede Viária	**Qtde Vértices**	**Qtde Arestas**	**Diâmetro**	**Média Coef. de Clusters**
Manaus	27.164	33.806	363	0,0019
Rio de Janeiro	110.036	124.531	1009	0,0022
Belo Horizonte	78.514	98.033	758	0,0050
Luxemburgo	41.640	45.830	503	0,0014
Le Havre	11.734	15.135	148	0,0444
Munique	304.968	354.961	661	0,0018
Nova Iorque	187.242	226.430	791	0,0035

Tabela 6.1: Características dos Grafos medidas nos Experimentos.

Pode-se perceber que mesmo Manaus sendo a segunda menor cidade quanto aos valores de vértices/arestas e diâmetro, ela é a que possui maior densidade em relação às medidas de grafo, sendo portanto a que mais exige o uso de CPU [76]. Rio de Janeiro possui a rede viária com maior diâmetro e terceira maior quanto a vértices e arestas. A seguir serão apresentados os resultados das análises da qualidade das rotas encontradas, dos tempos de execução e das memórias consumidas, bem como a variação do coeficiente heurístico (ϵ). A influência heurística é outro fator componente que foi analisado nos experimentos, foram computados resultados com $f(x) = g(x) + c$[1] e $f(x) = g(x) + \epsilon * h(x)$.

[1]Onde $c = \epsilon * 1$, isto foi feito para captar as variações do coeficiente heurístico nos resultados.

6.1 Análise dos Vértices

Os resultados quanto à qualidade das rotas encontradas e a variação do coeficiente heurístico são apresentados nesta seção. O objetivo desta análise é medir a quantidade de vértices que cada algoritmo planejador produz e compará-los entre si, tendo como referência os resultados do algoritmo de Dijkstra. Em outras palavras, temos que quanto maior aproximação do resultado apresentado por Dijkstra, melhor será o resultado.

6.1.1 Manaus

O experimento apresenta os resultados dos quatro algoritmos estudados na rede viária de Manaus, onde são analisadas as rotas de acordo com a influência da função heurística em cada um dos três pares sorteados aleatoriamente. Os algoritmos A* e Dijkstra, representados respectivamente por A* e DJK nos gráficos, apresentam resultados constantes devido não serem dependentes do coeficiente heurístico. Os resultados sem a influência heurística são visualizados na Figura 6.1, onde se pode notar que os algoritmos ARA* e ITS-ARA*, variam mesmo quando temos a função definida por $f(x) = g(x) + c$. Isto é possível devido à dinamicidade dos pesos nas arestas realizados nos experimentos. Em outras palavras, para cada cálculo de planejamento executado, um novo balanceamento aleatório é realizado na região de interesse da rede. Quanto à qualidade dos algoritmos de busca informada, pode-se notar que ARA* foi o que mais se aproximou dos valores de Dijkstra, mais precisamente nas variações do coeficiente (ϵ) de 3,4 nos Experimentos 1 e 3, de 2,2 no Experimento 2. Em relação ao algoritmo A*, ARA* foi menos eficiente quando seus coeficientes foram 1,0 (Experimentos 1 e 3) e 4,6 (Experimento 2). ITS-ARA* foi menos eficiente que A* nos valores de 3,8 e 4,2 nos Experimentos 1 e 2 respectivamente. Outra característica marcante nos valores dos vértices do experimento é sua variação, expressa por meio de número decimais, fato este que não acontece quando a função heurística é empregada, obtendo dessa forma resultados inteiros.

Os resultados da influência da função heurística no cálculo das rotas são apresentados na Figura 6.2, onde a grande maioria dos resultados de ARA* e sua versão ITS-ARA* estejam distantes dos valores de Dijkstra. Somente quando ϵ foi 1,4 (Experimentos 1 e 3) é que os algoritmos se aproximaram de Dijkstra. Vale ressaltar que ambos tiveram o mesmo comportamento em relação aos vértices encontrados, variando-se o coeficiente heurístico.

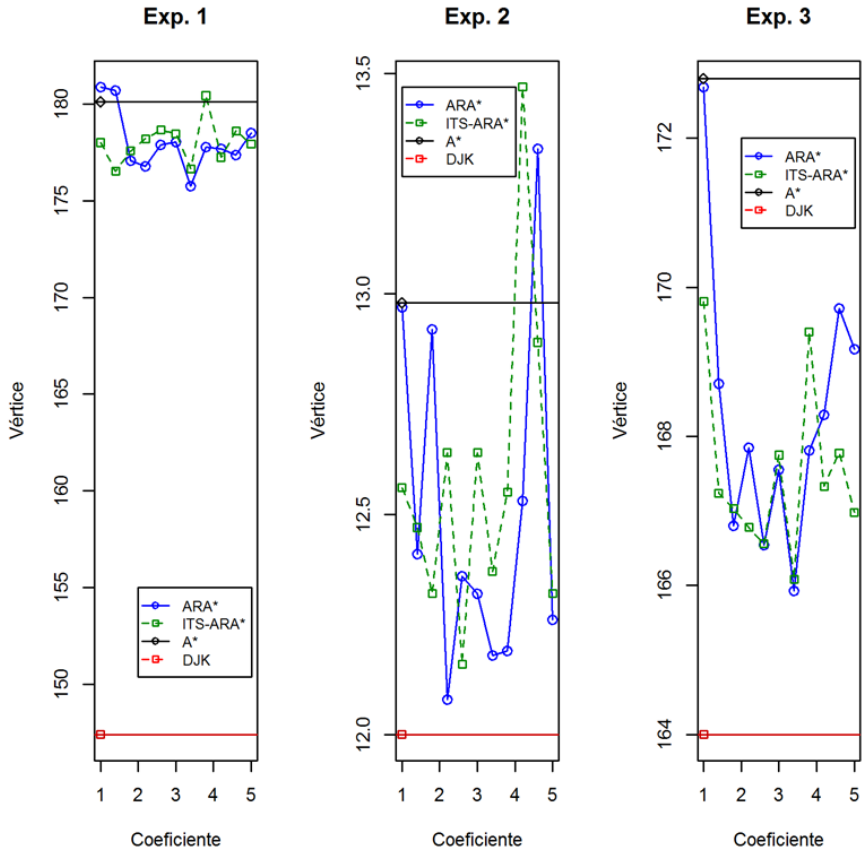


Figura 6.1: Manaus – Análise dos Vértices sem Heurística nos Experimentos.

6.1.2 Rio de Janeiro

A análise dos experimentos para a rede do Rio de Janeiro se deu pela variação do coeficiente heurístico ϵ e pela influência heurística nos resultados, bem com a qualidades das rotas geradas pelos algoritmos nos vértices. Nos resultados sem o uso da heurística (Figura 6.3), é possível notar a variação nos resultados encontrados por ARA* e ITS-ARA*. Mesmo sendo uma versão otimizada de ARA*, os resultados encontrados por ITS-ARA* diferenciam-se do original, sendo que no Experimento 2 foi quem alcançou uma melhor qualidade, devido à sua proximidade de Dijkstra. Ao compararmos os resultados de A* com os de ARA* e ITS-ARA*, percebeu-se que no Experimento 3 foi onde menos conseguiram resultados mais eficientes, mais precisamente cinco para ARA* (intervalos de ϵ de 3,0 a 4,8) e seis para ITS-ARA* (2,6 a 4,8), sendo que em geral o desempenho de ITS-ARA* foi superior ao de seu original, obtendo a maior quantidade de resultados positivos em relação aos resultados de A*.

A adição da heurística é apresentada na Figura 6.4, onde a qualidade dos resultados de ARA* e ITS-ARA* não foram muito significativa, sendo que somente no Experimento 2 obteve um intervalo mais próximos aos valores de referência de

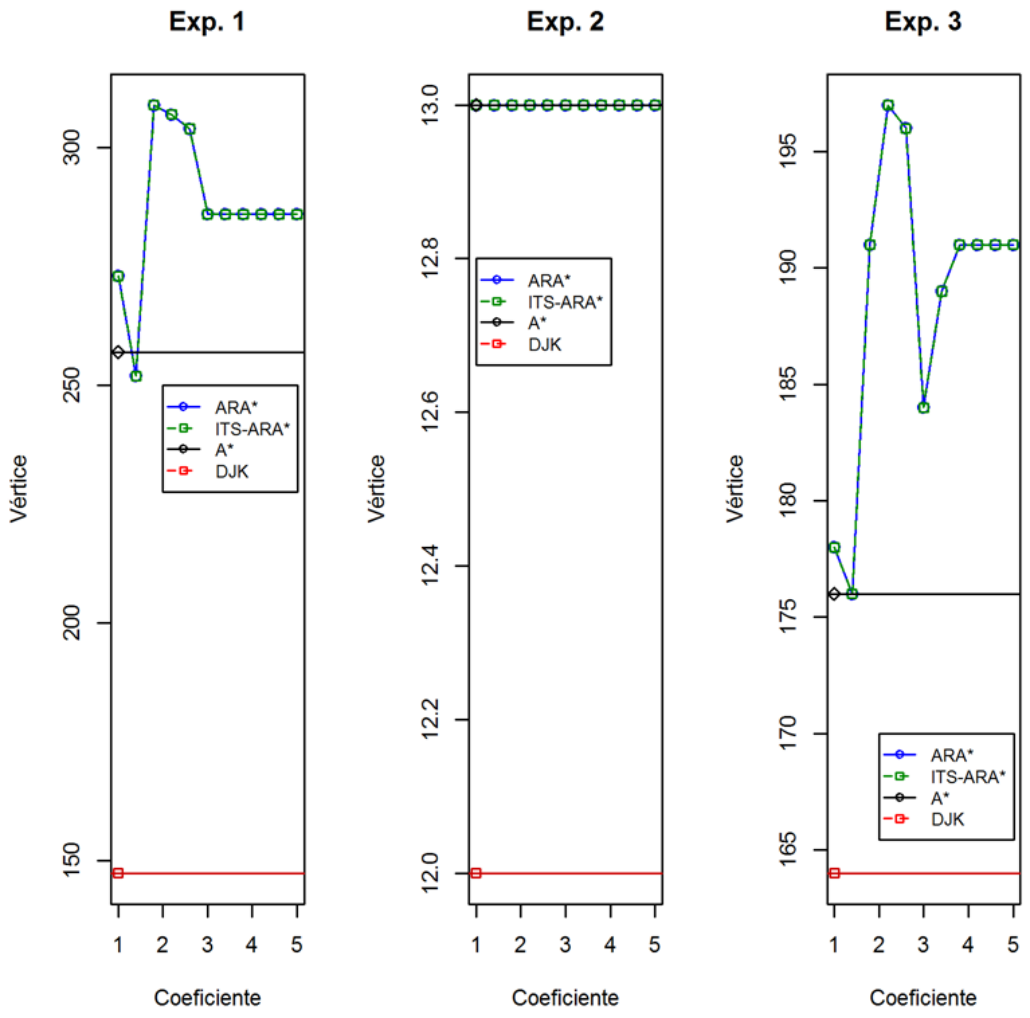


Figura 6.2: Manaus – Análise dos Vértices com Heurística nos Experimentos.

Dijkstra, mais precisamente de 1,0 a 2,6. Ao se comparar com os resultados de A*, percebeu-se que em geral os resultados não foram favoráveis aos algoritmos *Anytime*, o Experimento 3 é o que melhor reproduz essa afirmativa, onde se percebe que em nenhuma variação do coeficiente os valores dos referidos algoritmos conseguiu superar os resultados de A*.

6.1.3 Belo Horizonte

A avaliação dos resultados sem a influência heurística para a rede de Belo Horizonte é apresentado na Figura 6.5. No comparativo dos algoritmos sem a influência da heurística, percebeu-se que o Experimento 3 foi o que apresentou o pior desempenho, tanto no comparativo com Dijkstra, quanto no comparativo com A*, sendo que seu melhor resultado é o apresentando por ITS-ARA* ($\epsilon = 4{,}6$) seguido de ARA* ($\epsilon = 4{,}2$). Nos Experimentos 1 e 2, os algoritmos *Anytime* obtiveram desempenho

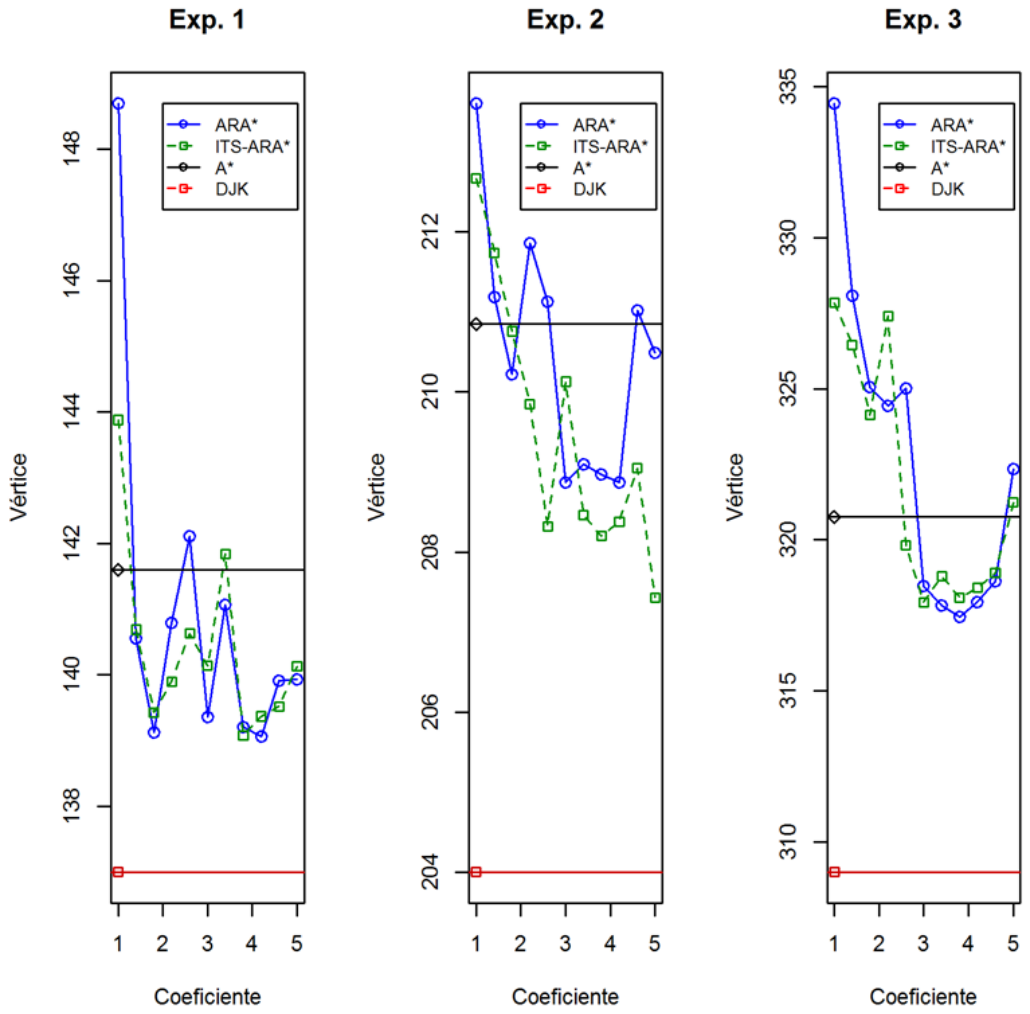


Figura 6.3: Rio de Janeiro – Análise dos Vértices sem Heurística nos Experimentos.

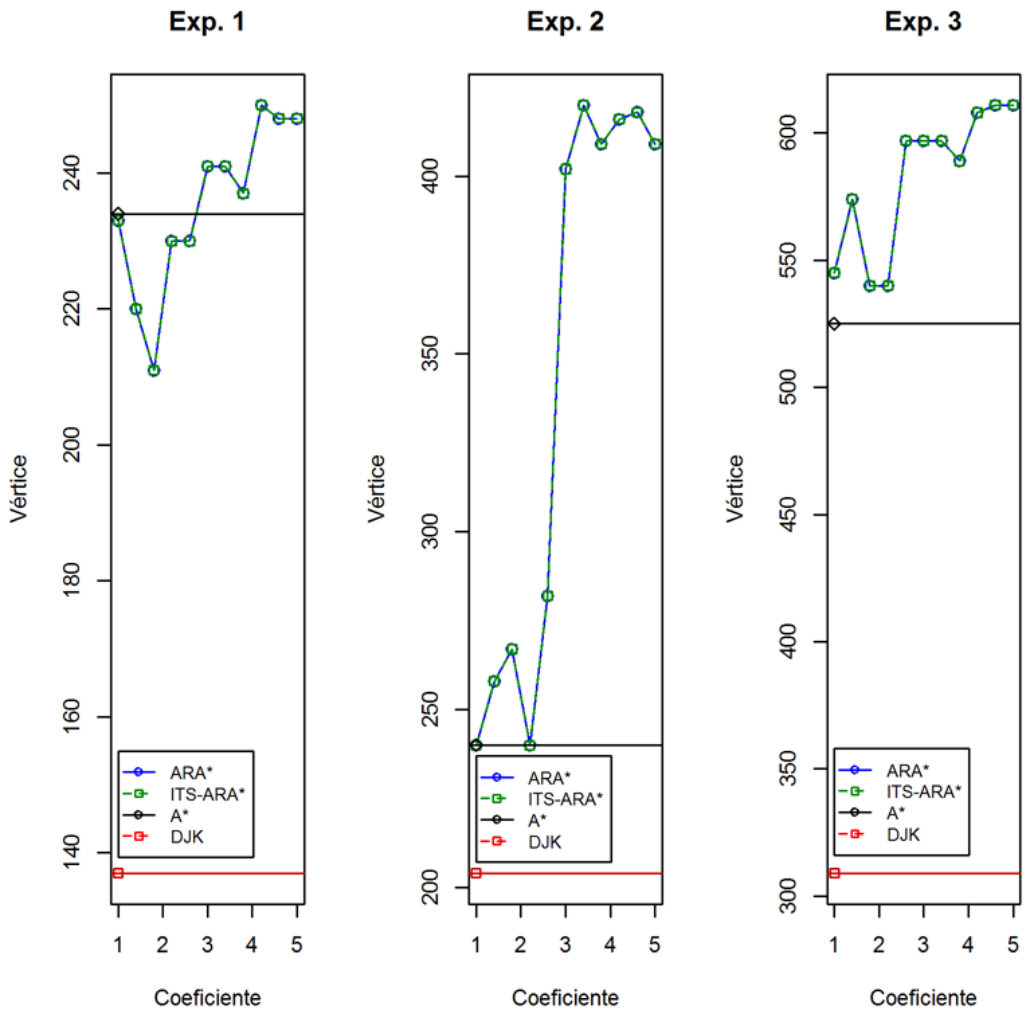


Figura 6.4: Rio de Janeiro – Análise dos Vértices com Heurística nos Experimentos.

melhor, sendo o de ARA* mais significativo que ITS-ARA*.

Nos resultados com adição da heurística (Figura 6.6), percebeu-se que o comportamento dos algoritmos *Anytime*, ARA* e ITS-ARA*, mantiveram os mesmos resultados, sendo que o Experimento 1 ($\epsilon = 1{,}0$) foi o único em que ambos conseguiram um desempenho melhor que A* e consequentemente foi o que mais se aproximou do resultado de Dijkstra. Nos demais Experimentos seus desempenhos não foram tão satisfatórios, sendo que seus resultados mais aproximados foram quando o coeficiente heurístico foi igual a 1,0.

6.1.4 Luxemburgo

A Figura 6.7 apresenta os resultados dos experimentos na rede de Luxemburgo sem heurística, podendo-se perceber que o Experimento 3 foi o que apresentou melhor desempenho nos resultados, sendo que o ARA* foi quem apresentou melhor

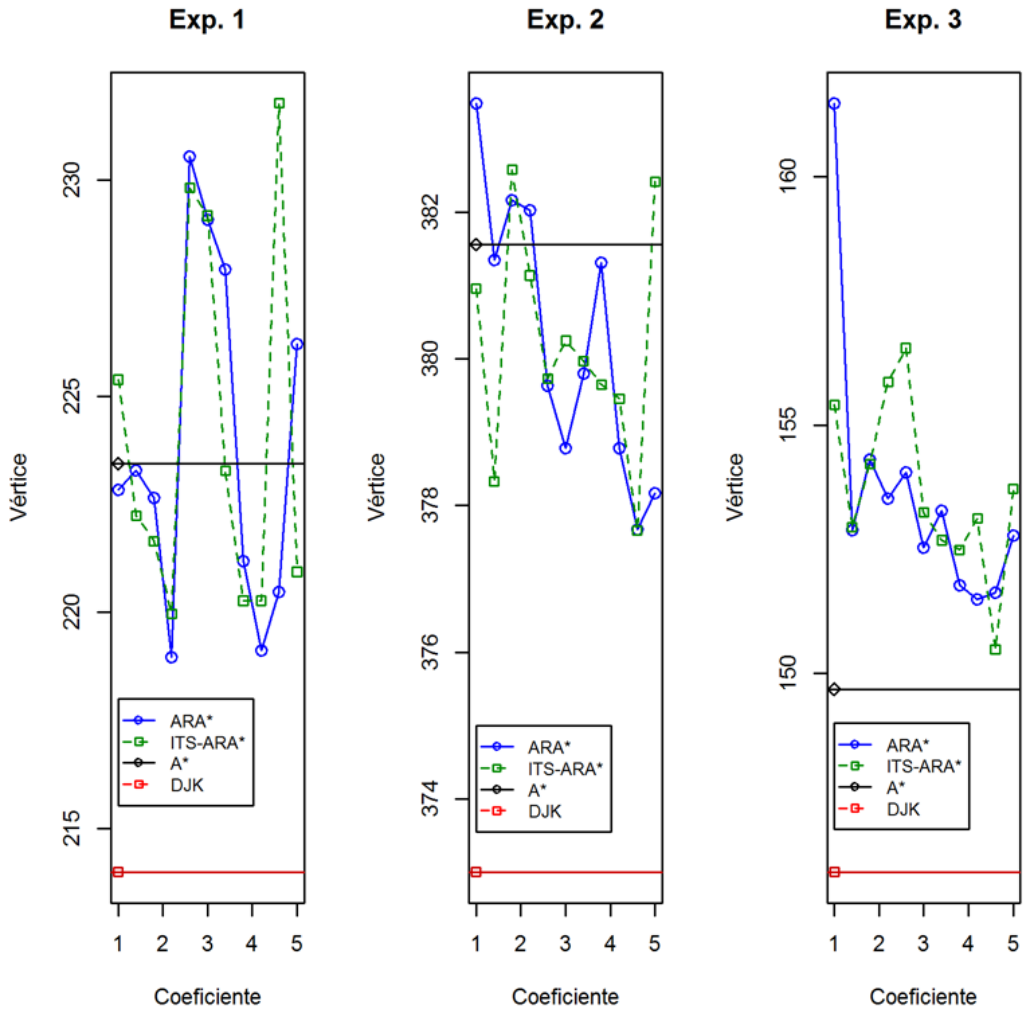


Figura 6.5: Belo Horizonte – Análise dos Vértices sem Heurística nos Experimentos.

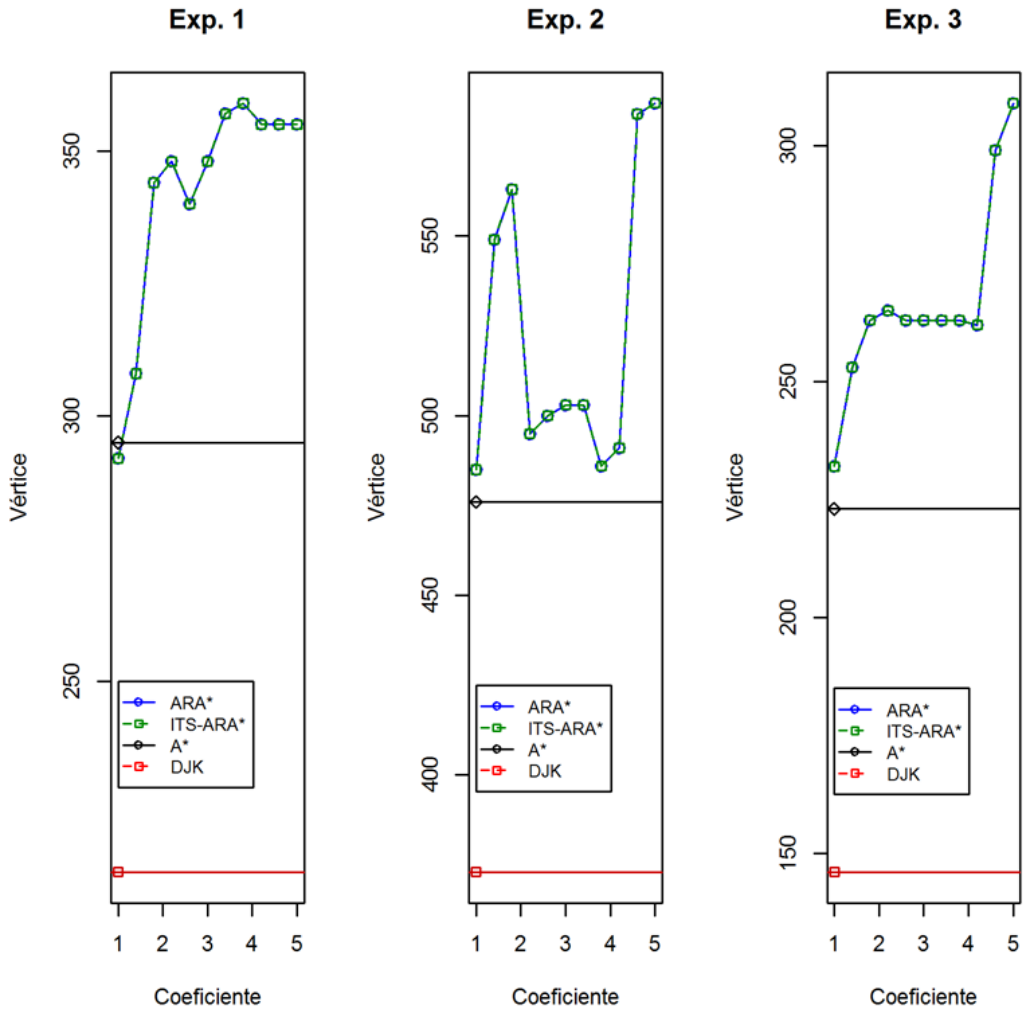


Figura 6.6: Belo Horizonte – Análise dos Vértices com Heurística nos Experimentos.

resultado em comparação com Dijkstra e A*, mais precisamente para ϵ igual a 3,8. O desempenho de ITS-ARA* foi melhor que ARA* no Experimento 3, onde teve resultado mais expressivo para ϵ igual a 4,6. No entanto, no Experimento 2, ARA* foi quem teve melhor êxito nos resultados em comparação com ITS-ARA* e tendo por referência os algoritmos de A* e Dijkstra.

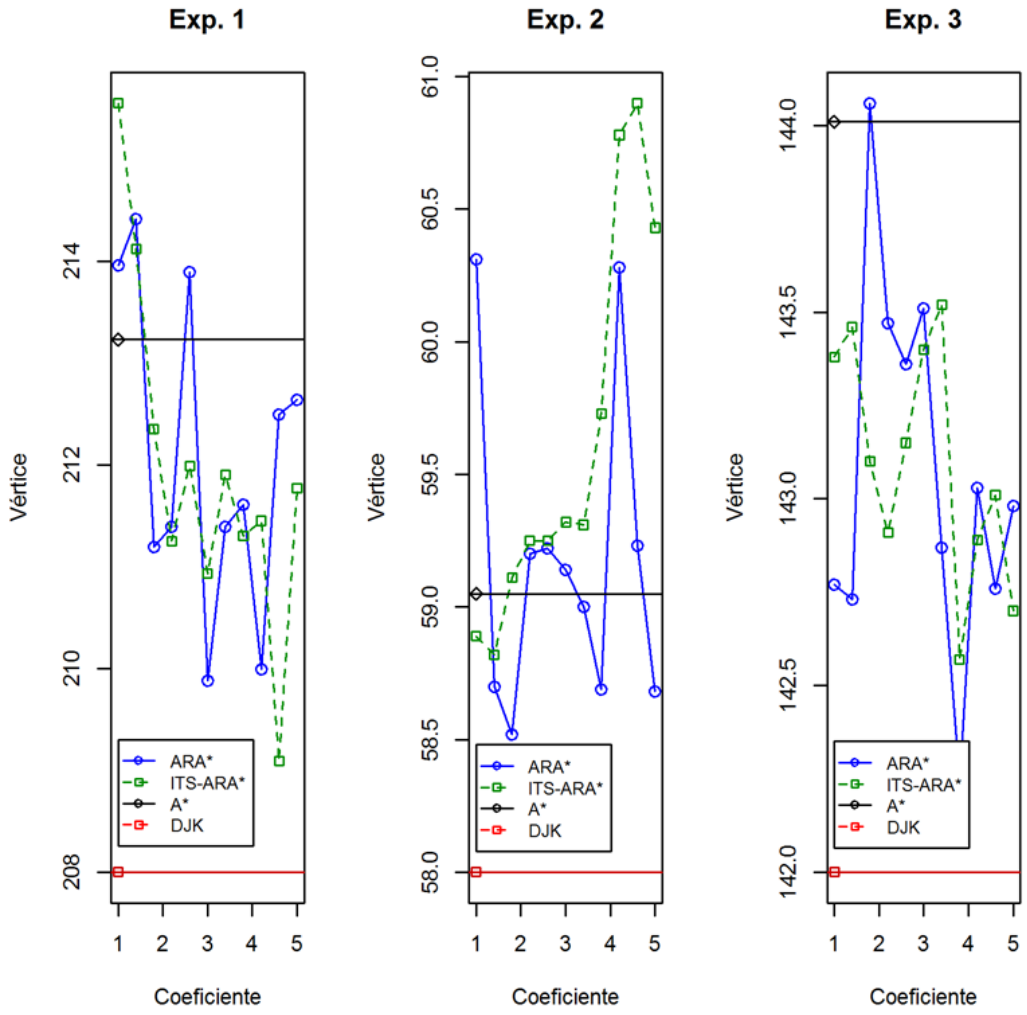


Figura 6.7: Luxemburgo – Análise dos Vértices sem Heurística nos Experimentos.

Nos resultados com utilização da heurística (Figura 6.8), ARA* e ITS-ARA* tiveram seu melhor desempenho no Experimento 1 com coeficiente igual a 1,8. No Experimento 2, pode-se observar que o resultado do algoritmo A* foi mais próximo ao do Dijkstra, isto se deveu a rota encontrada ter sido mais curta em relação às demais. Ainda no referido Experimento, o desempenho dos *Anytime* foi o menos satisfatório, sendo que seu melhor resultado foi para ϵ igual a 1,0. Para o Experimento 3, percebe-se que o resultado mais significativo dos algoritmos em questão, é quando o coeficiente heurístico apresenta o valor de 3,0.

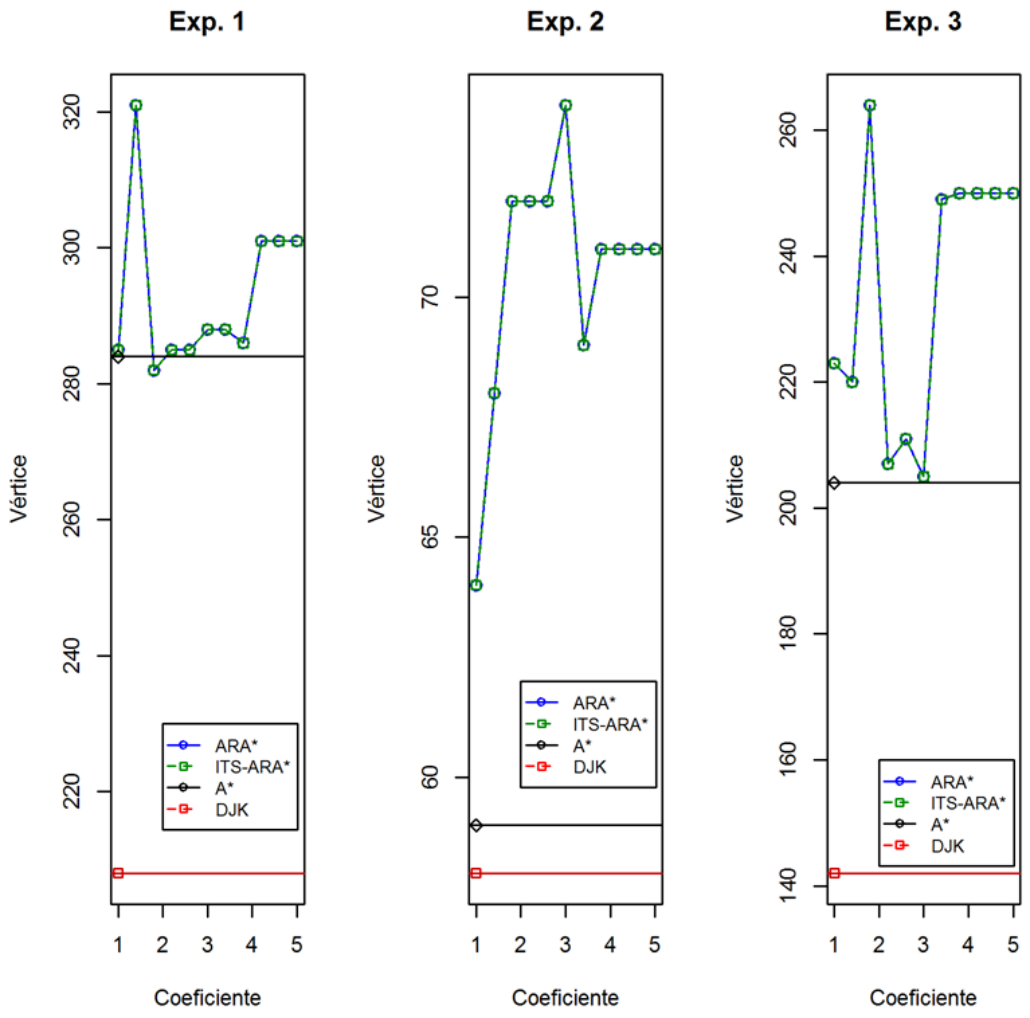


Figura 6.8: Luxemburgo – Análise dos Vértices com Heurística nos Experimentos.

6.1.5 Le Havre

A Figura 6.9 ilustra a ausência da heurística nos resultados dos experimentos, pode-se observar que o Experimento 2 foi o que apresentou resultados menos satisfatórios dos algoritmos *Anytime* em relação aos algoritmos de referência, sendo que ARA* obteve resultados mais significativos em relação a ITS-ARA*. O Experimento 1 foi onde os referidos algoritmos tiveram seus melhores desempenhos, sendo que em geral ARA* obteve resultados mais próximos aos valores de referência. No entanto, para o Experimento 3, apesar da grande variação apresentada pelos algoritmos, ITS-ARA* foi o que teve o melhor resultado para valor de 2,2.

No Experimento 2, para os resultados com heurística (Figura 6.10), ARA* e ITS-ARA* obtiveram seus melhores resultados no experimento, mais precisamente para coeficiente igual a 1,0. Vale ressaltar que A* e Dijkstra obtiveram o mesmo resultado no referido Experimento. No Experimento 1, ARA* e ITS-ARA* não

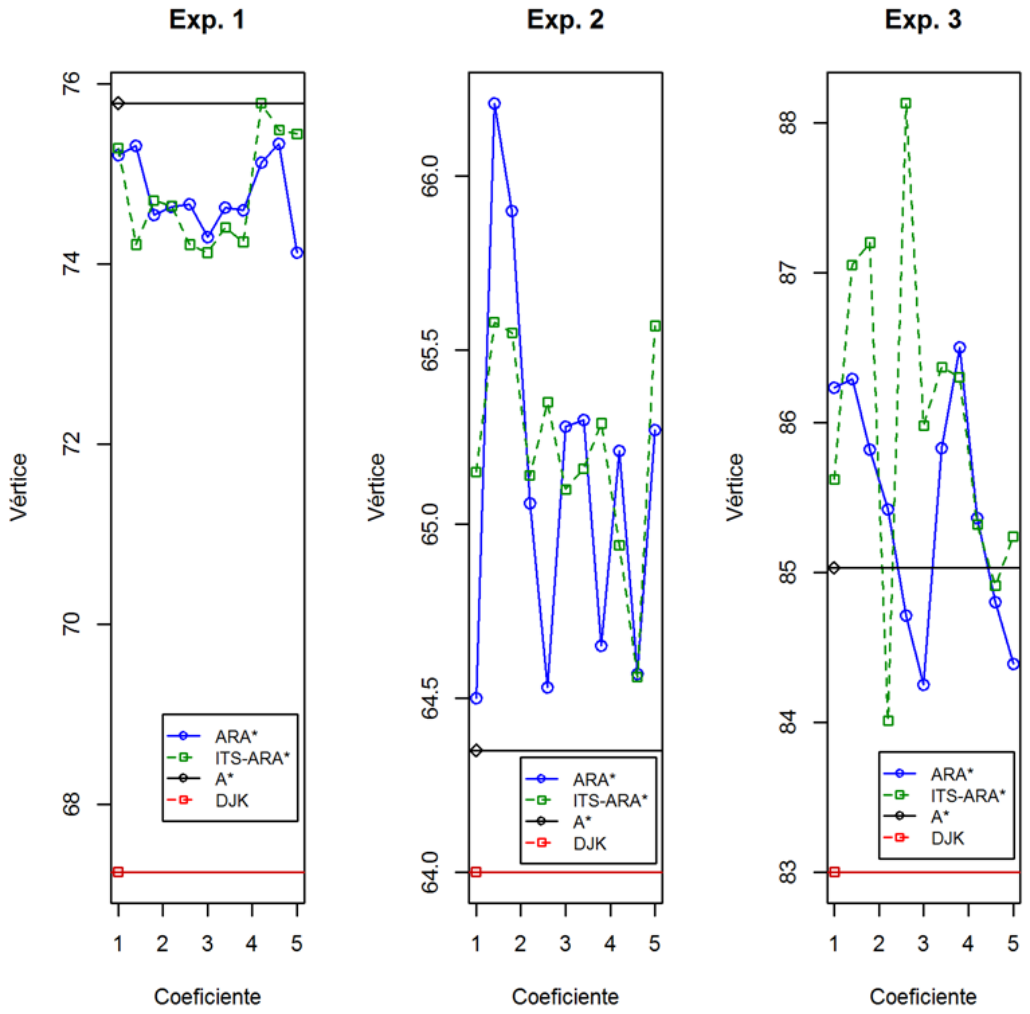


Figura 6.9: Le Havre – Análise dos Vértices sem Heurística nos Experimentos.

obtiveram muito êxito quando comparado com os valores de referência, sendo que suas melhores aproximações foi para ϵ igual a 1,0. No Experimento 3 foi onde os referidos algoritmos obtiveram suas maiores sequências de melhores resultados, mais precisamente no intervalo de 2,4 a 4,2 de ϵ.

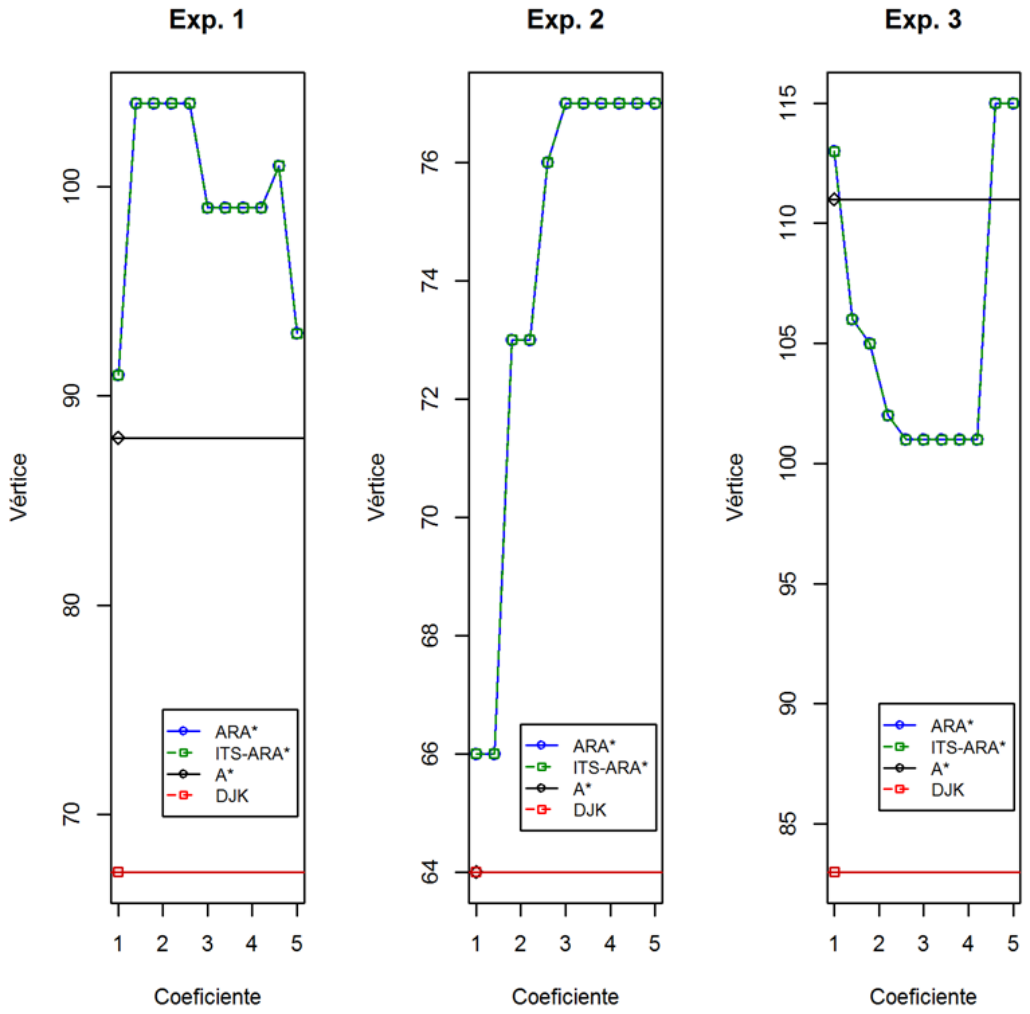


Figura 6.10: Le Havre – Análise dos Vértices com Heurística nos Experimentos.

6.1.6 Munique

Os resultados dos experimentos na rede viária de Munique com ausência da função heurística são visualizado na Figura 6.11, onde todos os Experimentos estudados apresentaram uma constante nos vértice coletados, sendo que no Experimento 1 foi onde ARA* e ITS-ARA* obtiveram menor desempenho em relação aos algoritmos de referência. O Experimento 3 foi onde os referidos algoritmos obtiveram seus melhores resultados, mais precisamente, a diferença foi de oito vértices em comparação com A* e de 12 vértices para o Experimento 2.

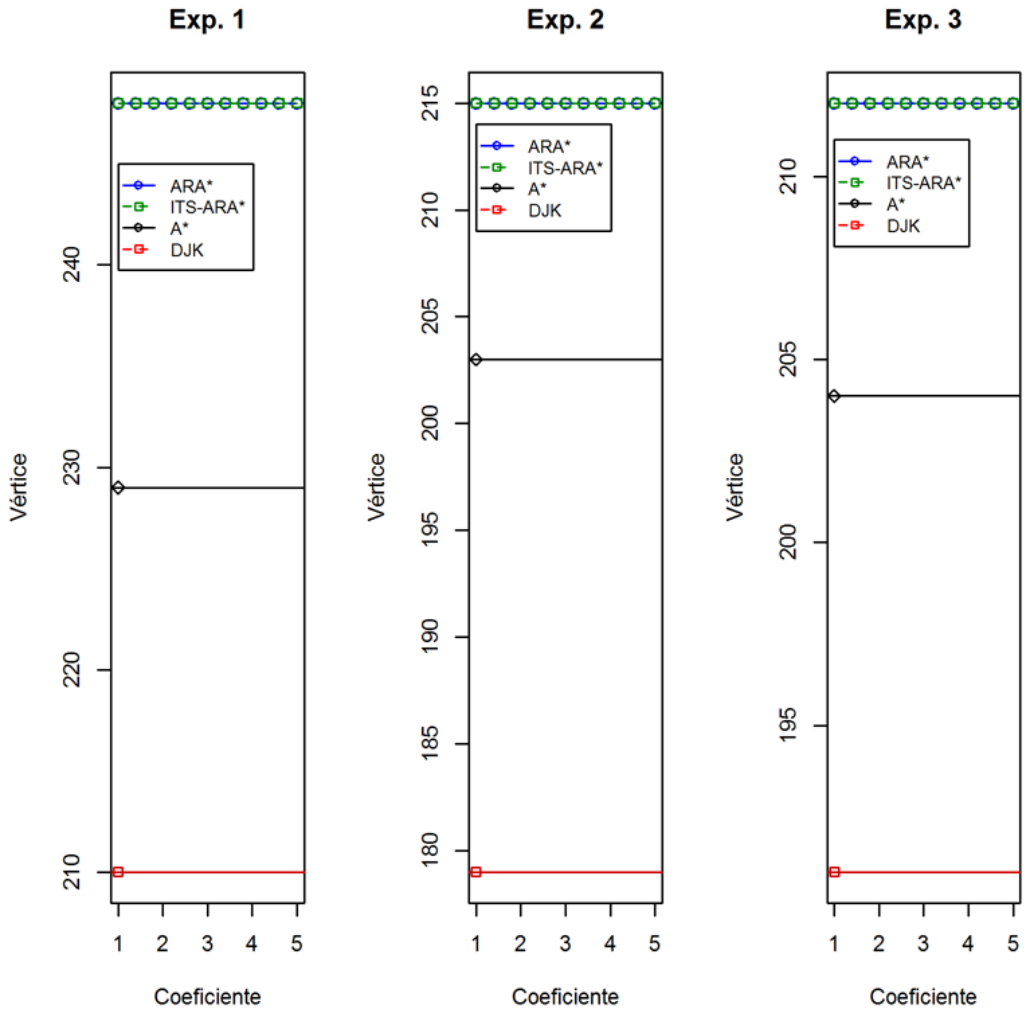


Figura 6.11: Munique – Análise dos Vértices sem Heurística nos Experimentos.

A Figura 6.12 apresenta os resultados com a função heurística adicionada, pode-se observar que o Experimento 3 foi onde os algoritmos *Anytime* obtiveram seus melhores desempenhos, sendo mais precisamente suas diferenças para o valor do Dijkstra foi de 52 vértices contras os 69 vértices no Experimento 2, respectivamente para os valores de 1,4 e de 1,0 para os valores do coeficiente heurístico. No Experimento 3 foi onde os referidos algoritmos não obtiveram tanto sucesso em comparação com os demais Experimentos, sendo que seus melhores resultados se deram para valores de ϵ iguais a um.

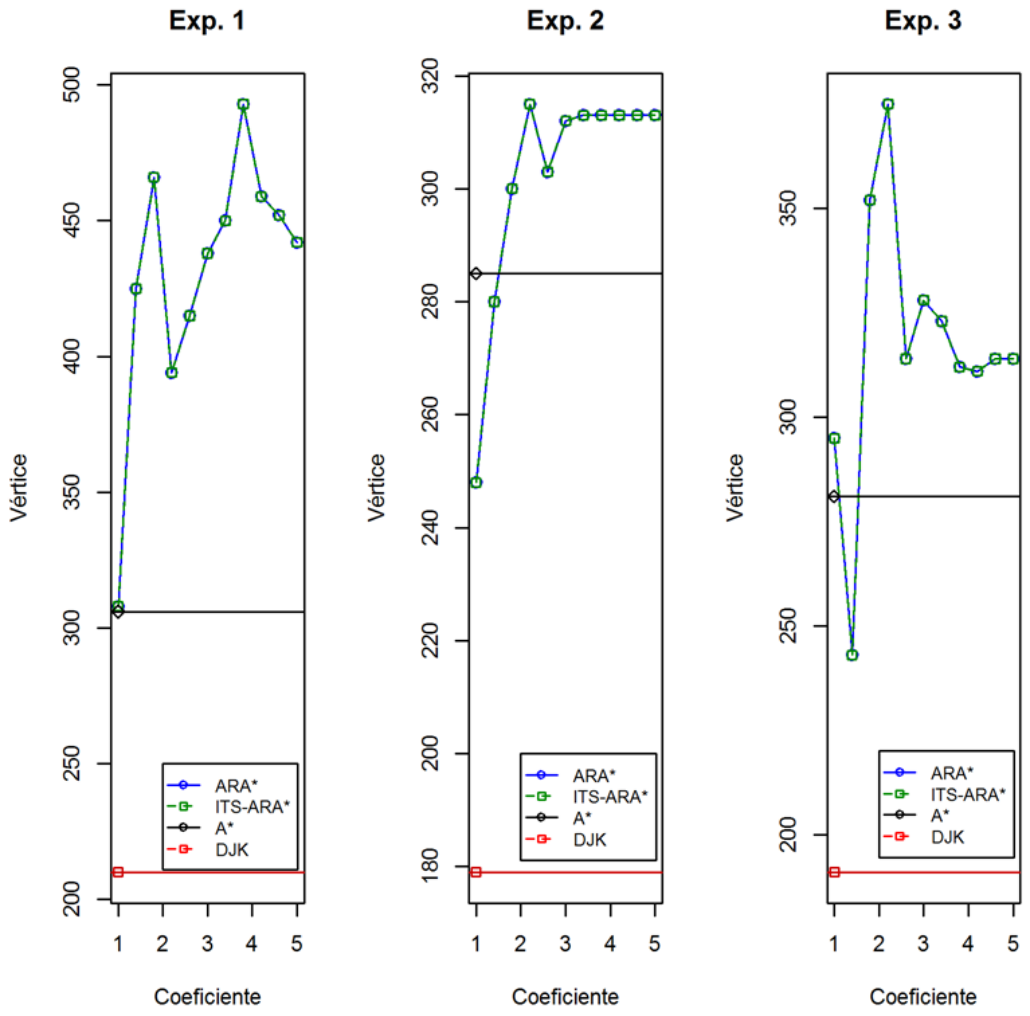


Figura 6.12: Munique – Análise dos Vértices com Heurística nos Experimentos.

6.1.7 Nova Iorque

Os resultados da rede viária de Nova Iorque são visualizados na Figura 6.13 sem a utilização da função heurística e consequentemente a Figura 6.14, apresenta o uso da heurística nos resultados. Observa-se a princípio nos resultados sem o

uso da heurística, que os valores de A* são inferiores aos do Dijkstra, sendo que no Experimento 3, eles estão bastante próximos. Ainda no mesmo Experimento, pode-se observar o comportamento de ARA* e ITS-ARA* como bastante oscilantes, sendo que o primeiro obteve melhor desempenho em comparação com o segundo, mais precisamente para a variação de 4,6 e 1,4 foi o melhor resultado de ITS-ARA*. Outro bom resultado alcançado pelos algoritmos em questão pode ser visualizado no Experimento 1, onde ITS-ARA* obteve melhor resultado em comparação com seu original (em $\epsilon = 3{,}4$). No Experimento 2, ITS-ARA* obteve o melhor e foi o menos eficiente em comparação com seu original, respectivamente nos valores de *epsilon* iguais a 5,0 e 1,0, mais precisamente seu extremos.

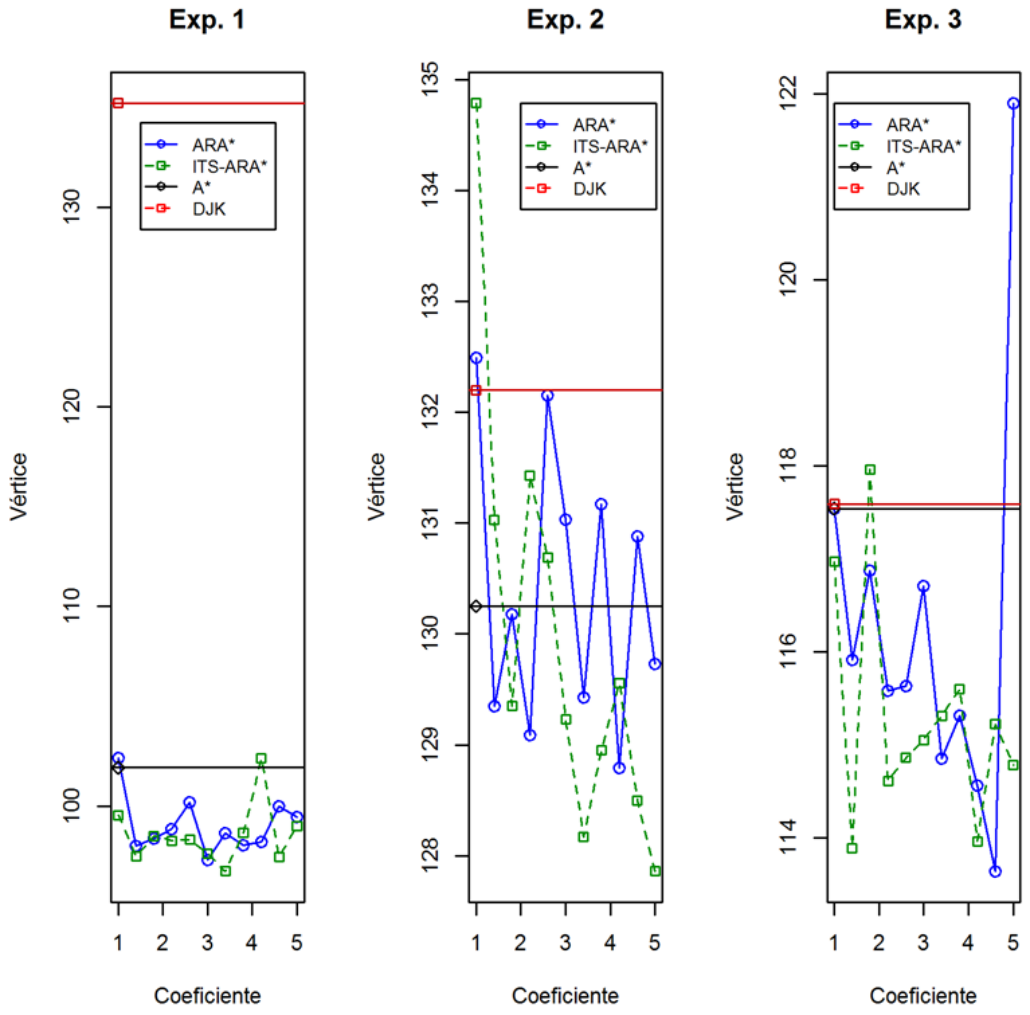


Figura 6.13: Nova Iorque – Análise dos Vértices sem Heurística nos Experimentos.

Com o auxílio da heurística, a superioridade de A* em relação ao Dijkstra foi notada somente no Experimento 1, sendo que ITS-ARA* foi quem obteve o melhor desempenho em relação ao ARA* ($\epsilon = 1{,}0$). Nos Experimentos 2 e 3, os algoritmos em questão demonstraram o mesmo desempenho, sendo que no Experimento 3 seus

resultados foram mais evidentes, pois alcançaram um intervalo mais próximos dos valores de referência (Dijkstra).

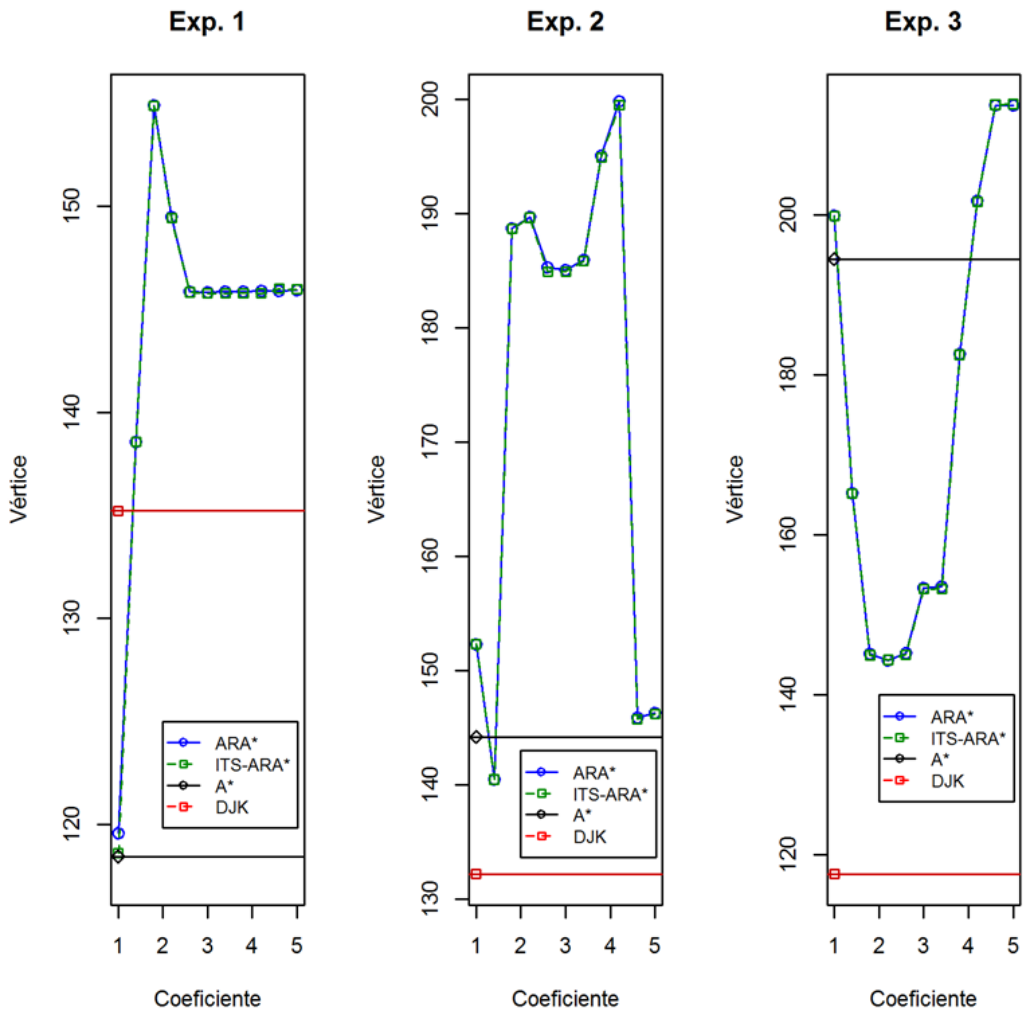


Figura 6.14: Nova Iorque – Análise dos Vértices com Heurística nos Experimentos.

6.2 Análise dos Tempos de Execução

Nesta seção os resultados são avaliados sob dois aspectos: a) tempo de execução pela variação do coeficiente heurístico e b) eficiência do coeficiente heurístico em relação ao tempo. O primeiro aspecto visa estudar o comportamento dos quatro algoritmos quanto ao tempo consumido para encontrar o melhor caminho, são analisados as variações de ϵ em ARA* e ITS-ARA* em comparação com os demais. O segundo aspecto estudado, tem por razão buscar qual o melhor valor encontrado de ϵ para cada rede estudada. Vale ressaltar que na literatura este valor é determinado a partir de testes de campo. Isto é devido à sua área de origem, a robótica. Isto é compreensível pois os Experimentos utilizados não são conhecidos. No entanto,

para a aplicação proposta isto não se aplica, devido se conhecer previamente o grafo a ser estudado. A seguir as redes viárias serão analisadas segundo os dois aspectos mencionados.

6.2.1 Manaus

As análise dos algoritmos de busca nos três Experimentos em relação ao tempo de execução sob a influência da heurística são apresentados a seguir, sendo que o primeiro aspecto a ser estudado tem por referência ausência da heurística, apresentado na Figura 6.15, onde podem ser vistos os quatro algoritmos e seus três experimentos. Pode-se notar que somente no Experimento 2, os algoritmos de busca informada (A*, ARA* e ITS-ARA*) foram mais rápidos que o de Dijkstra, sendo que ARA* e ITS-ARA* tiveram valores próximos ao de A*, mesmo com a variação do coeficiente, excetuando-se somente quando $\epsilon = 5{,}0$. Nos Experimentos 1 e 3, pode-se notar que conforme aumenta a variação de ϵ em ARA* e ITS-ARA*, maior é o tempo de execução.

O uso da heurística pode ser visualizado na Figura 6.16, somente os algoritmos de busca informado são visualizados, devido somente os mesmos sofrerem tal influência. Pode-se notar que somente no Experimento 1, ARA* e ITS-ARA* tem seu pior desempenho em relação a A*, mais precisamente quando ϵ igual a 1,0. Nos Experimentos 1 e 3, ARA* e ITS-ARA* tem seu melhor desempenho para o coeficiente igual a 1,4. No Experimento 2, seus comportamentos são quase paralelos à variação dos coeficientes, sendo que seus menores valores ϵ flutuam entre 1,0 a 1,4 para ARA* e 1,0 a 1,8 para ARA*.

Para o segundo aspecto analisado, será utilizada a Figura 6.17, onde são visualizados a variação dos coeficientes heurísticos em relação ao tempo de execução. As Figuras 6.17(a) e 6.17(b) ilustram o comportamento dos algoritmos ITS-ARA* e ARA* respectivamente, em relação à variação de seus coeficientes heurísticos. Ambos os algoritmos apresentam um crescimento no tempo de execução conforme é variado o coeficiente heurístico.

A influência heurística é visualizada nas Figuras 6.17(c) e 6.17(d), respectivamente para os algoritmos ITS-ARA* e ARA*. Percebe-se que nos Experimentos 1 e 3, ambos os algoritmos alcançam seu melhor desempenho quando ϵ é igual a 1,4.

6.2.2 Rio de Janeiro

A rede viária do Rio de Janeiro é analisada sob o aspecto da variação do coeficiente em relação ao tempo consumido e a influência heurística. Pode-se observar nos resultados sem heurística (Figura 6.18), que A* apresentou melhor desempenho que Dijkstra, assim como ITS-ARA* obteve resultados superiores aos de ARA* em

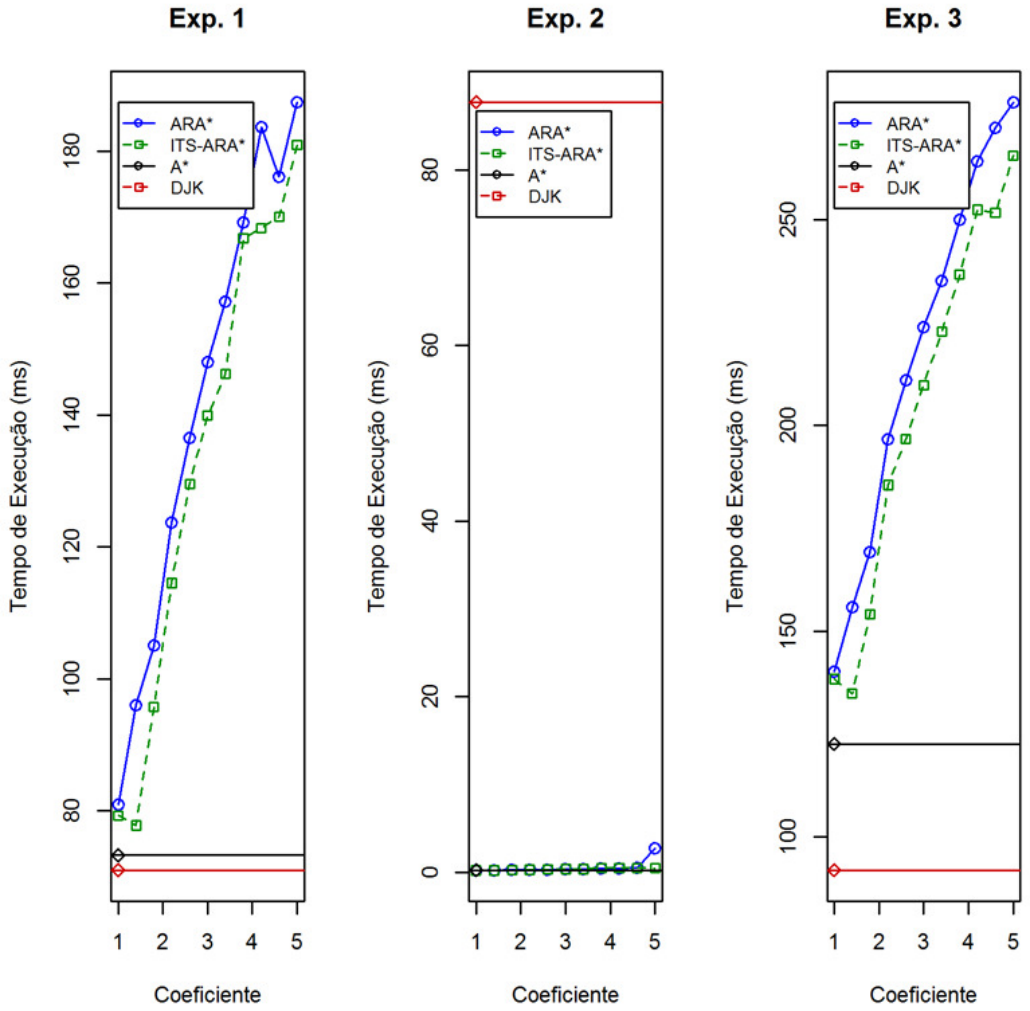


Figura 6.15: Manaus – Análise dos Tempos de Execução (ms) sem Heurística nos Experimentos.

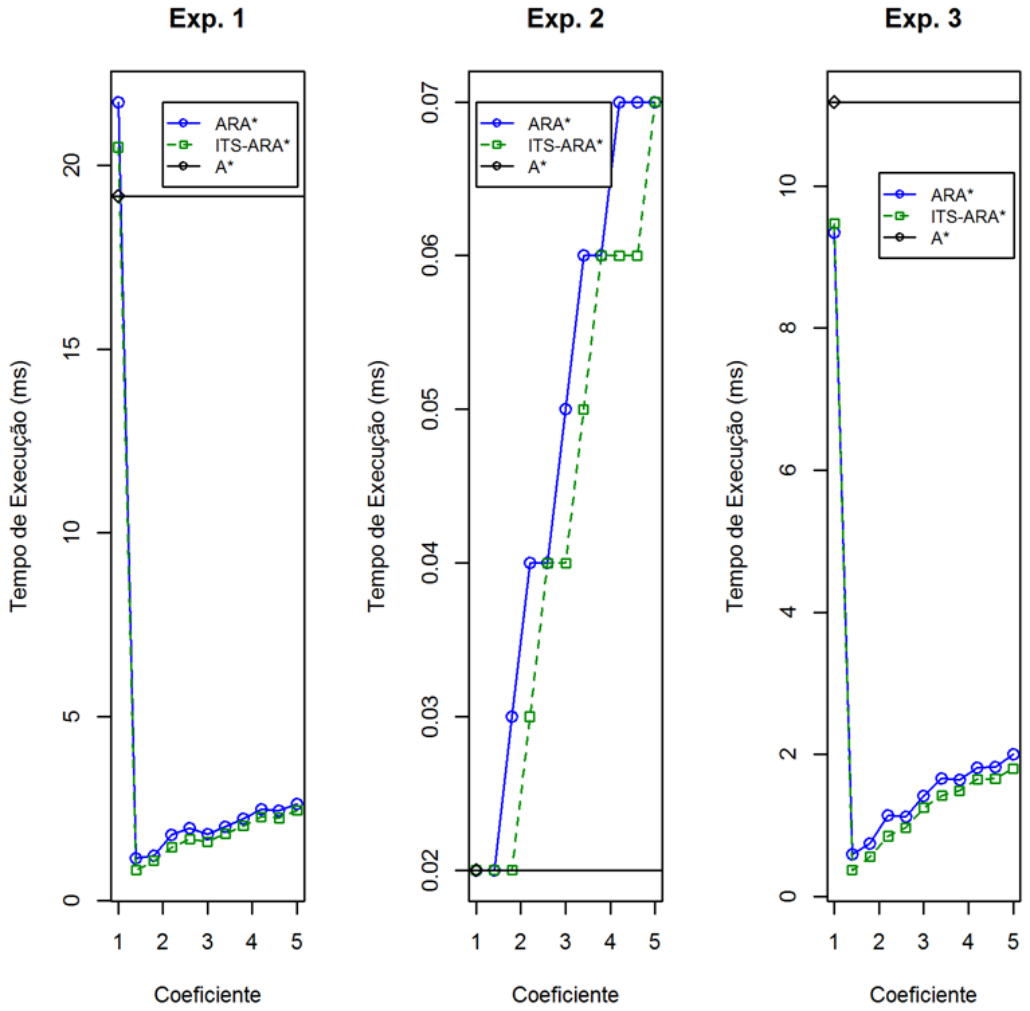
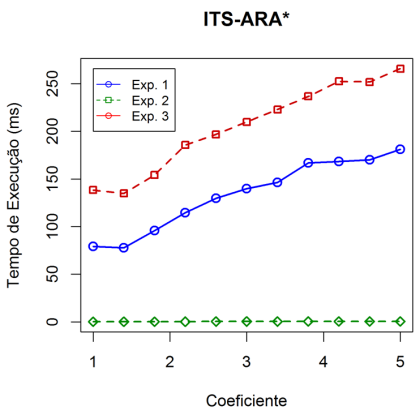
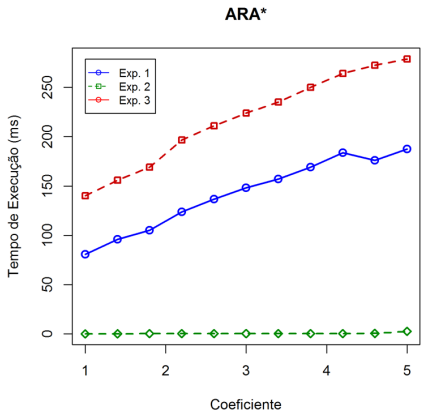


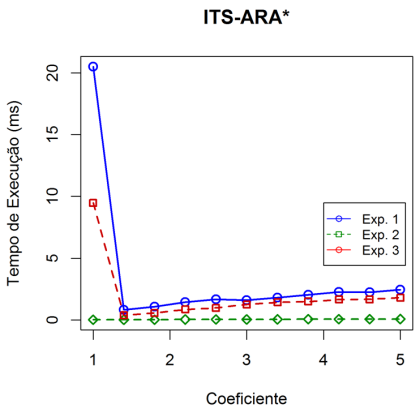
Figura 6.16: Manaus – Análise dos Tempos de Execução (ms) com Heurística nos Experimentos.



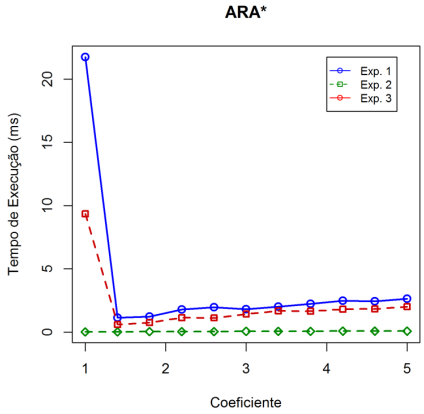
(a) ITS-ARA* sem Heurística.



(b) ARA* com Heurística.



(c) ITS-ARA* com Heurística.



(d) ARA* com Heurística.

Figura 6.17: Manaus - Variação do Coeficiente Heurístico nos Experimentos.

todos os Experimentos, sendo o Experimento 1 o mais evidenciado. Outro ponto a ser ressaltado é a variação do resultados, quanto maior o coeficiente heurístico, maior foi o tempo consumido nos planejamentos de rotas.

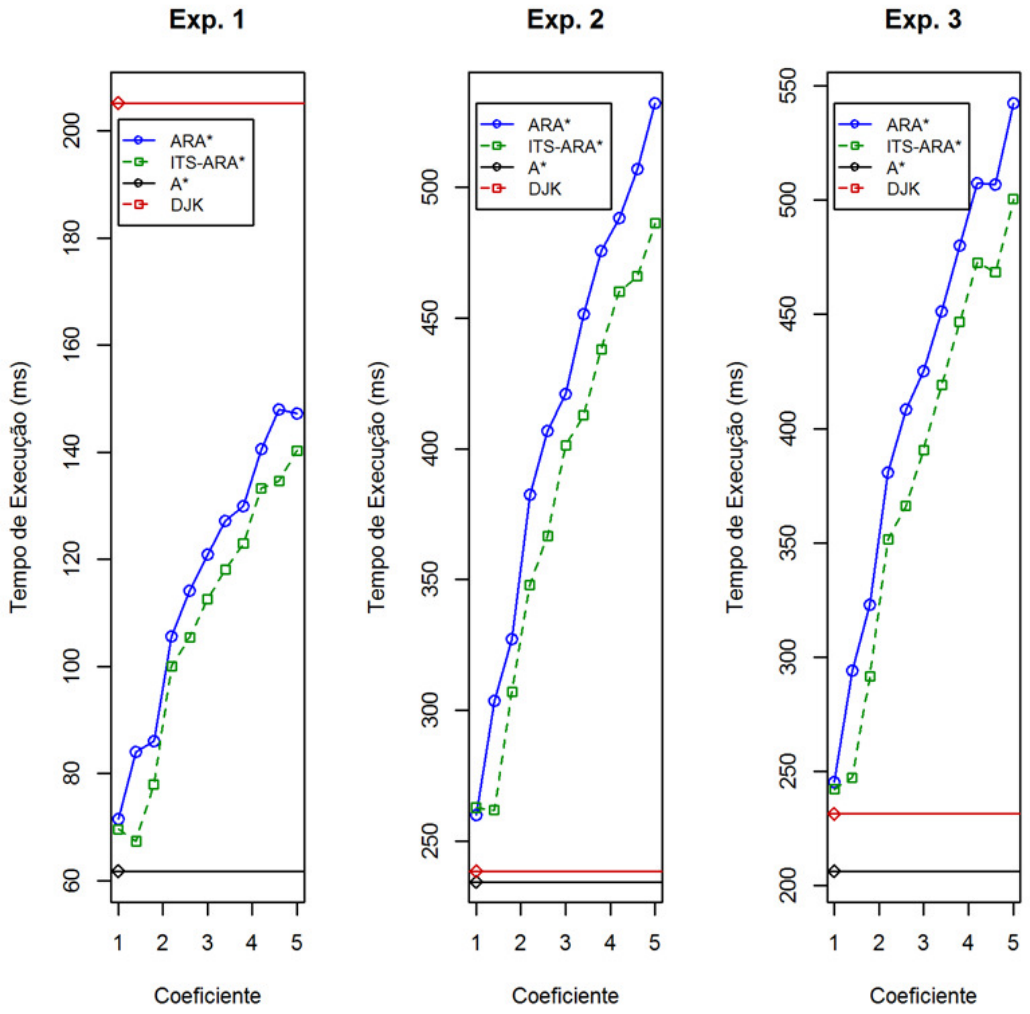


Figura 6.18: Rio de Janeiro – Análise dos Tempos de Execução (ms) sem Heurística nos Experimentos.

Para os resultados com a adição da heurística (Figura 6.19), não foram considerados os valores de Dijkstra, devido o algoritmo não ser influenciado por esta adoção. ITS-ARA* foi o algoritmo de melhor desempenho nos três experimentos estudados, sendo que nos Experimentos 1 e 3, onde obtiveram melhores resultados em comparação com os de referência (A*), mais precisamente nos intervalos de 1,4 a 5,0. Porém, no Experimento 2 é onde fica mais evidenciado a diferença entre os resultados de ARA* e ITS-ARA*, mais precisamente no intervalo de 1,8 a 5,0 do coeficiente heurístico.

Na Figura 6.20 é analisado o desempenho dos algoritmos ARA* e ITS-ARA* em relação ao tempo de execução. Os resultados sem heurística são apresentados

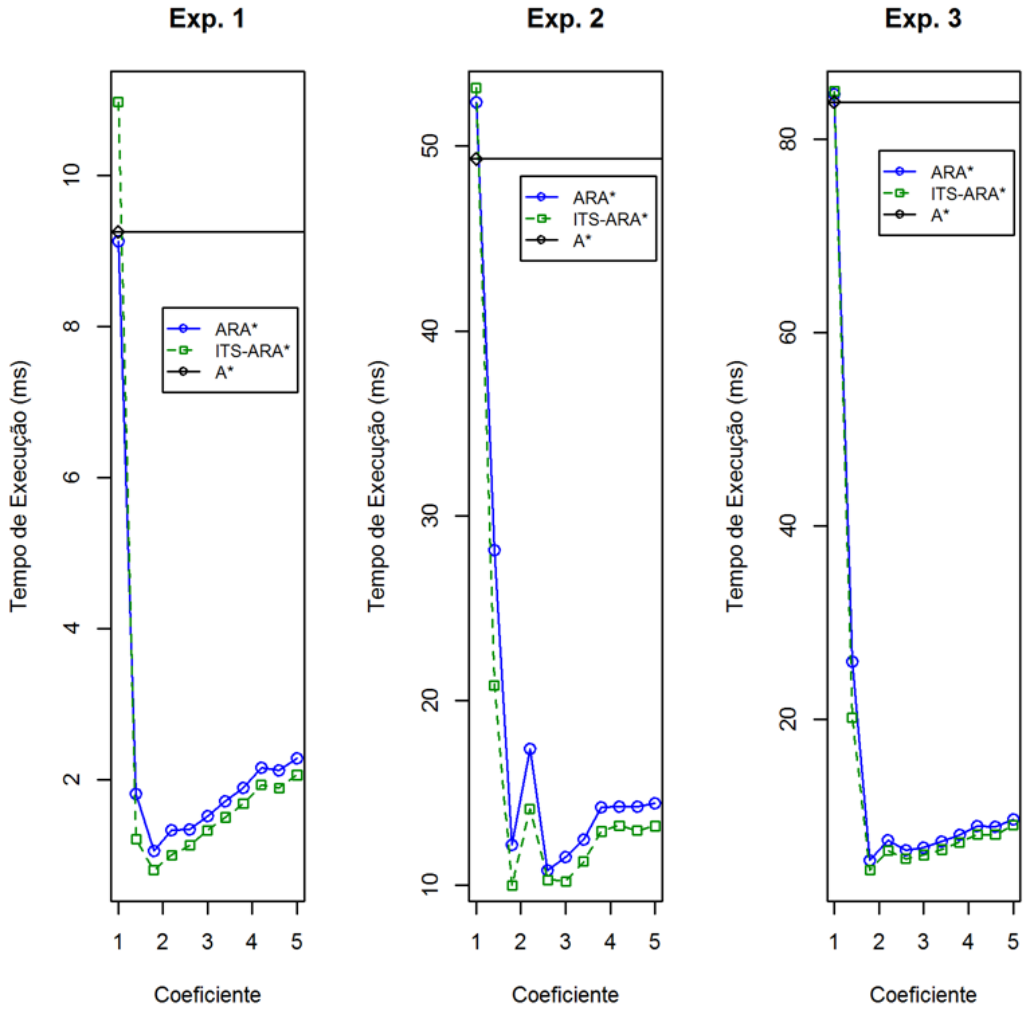
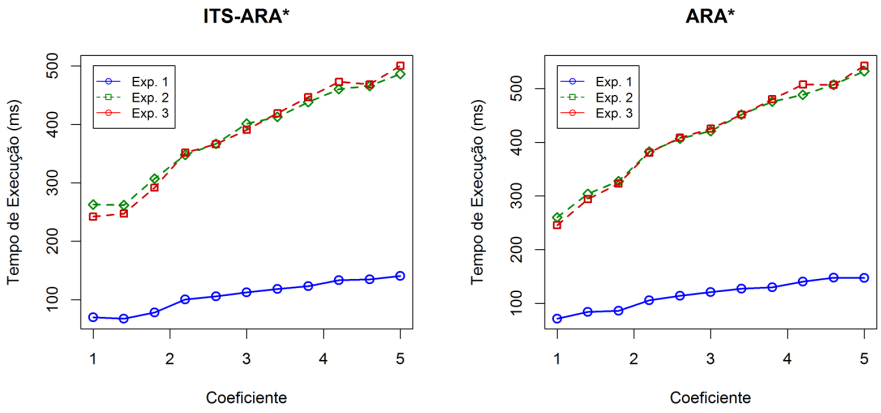


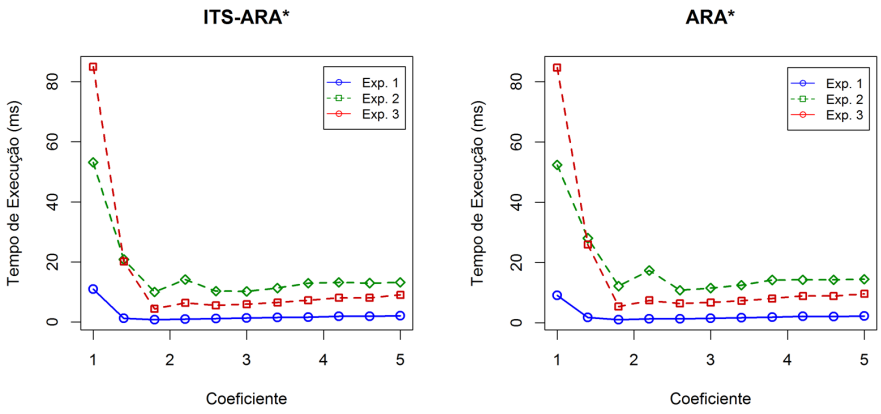
Figura 6.19: Rio de Janeiro – Análise dos Tempos de Execução (ms) com Heurística nos Experimentos.

nas Figuras 6.20(a) e 6.20(b), onde percebe-se que os melhores resultados foram para ϵ igual a 1,0, essa observação serve tanto para ARA* quanto para ITS-ARA*. Para os resultados com heurística (Figuras 6.20(c)e 6.20(d)), os algoritmos ARA* e ITS-ARA* obtiveram melhor desempenho para os valores de ϵ igual a 1,8 para os três experimentos.



(a) ITS-ARA* sem Heurística.

(b) ARA* sem Heurística.



(c) ITS-ARA* com Heurística.

(d) ARA* com Heurística.

Figura 6.20: Rio de Janeiro - Variação do Coeficiente Heurístico nos Experimentos.

6.2.3 Belo Horizonte

Os resultados dos experimento da rede de Belo Horizonte sem heurística podem ser visualizados na Figura 6.21. Pode-se observar nos Experimentos 1 e 2, ITS-ARA* obteve melhor desempenho que ARA*. No entanto, para o Experimento 3, pode-se observar uma alternância dos valores de referência, devido ao valor de A* ter sido menor que o de Dijkstra. Foi no referido Experimento, que ITS-ARA* obteve seu melhor desempenho em relação ao ARA* e aos valores de referência.

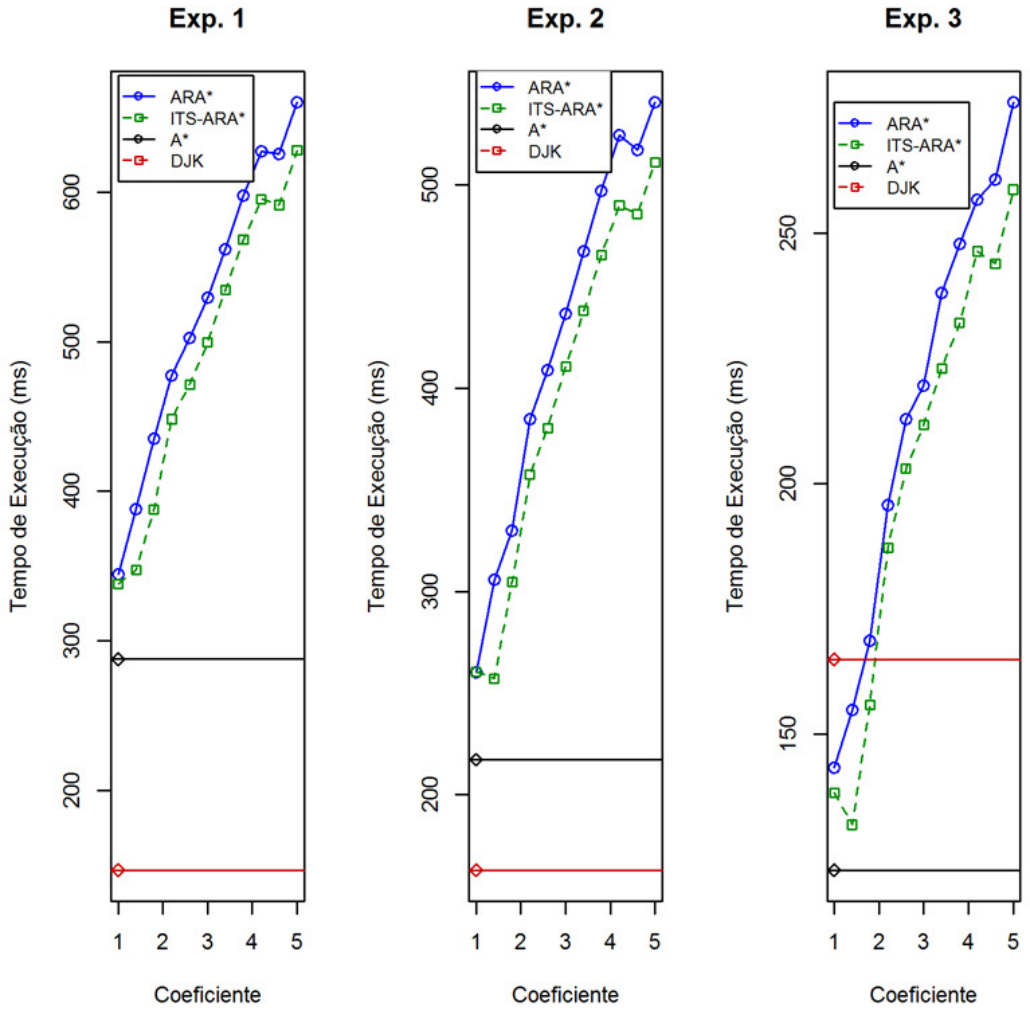


Figura 6.21: Belo Horizonte – Análise dos Tempos de Execução (ms) sem Heurística nos Experimentos.

Nos resultados com o uso da heurística (Figura 6.22), pode-se observar o melhor desempenho de ITS-ARA* em relação à ARA* nos três experimentos, sendo o terceiro o mais expressivo. É possível observar nos resultados que o intervalo de 1,4 a 5,0 do coeficiente heurístico é onde os desempenhos dos referidos algoritmos são melhores, estando abaixo dos valores de referência.

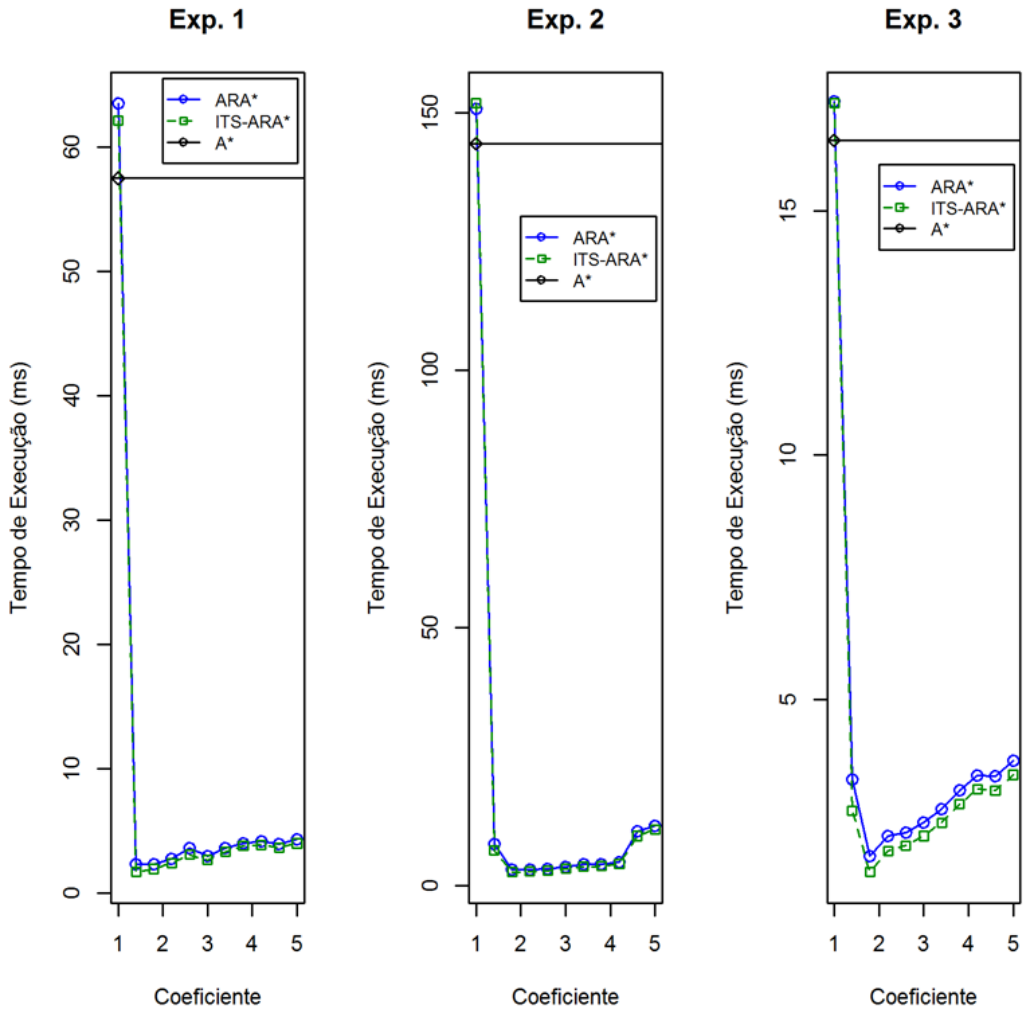
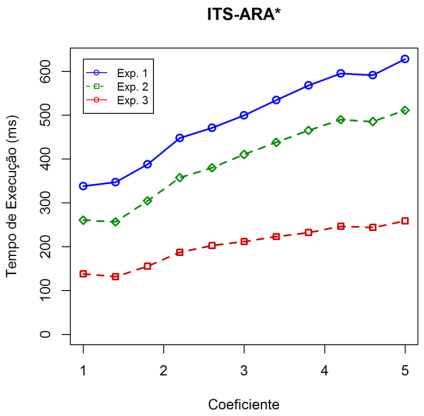


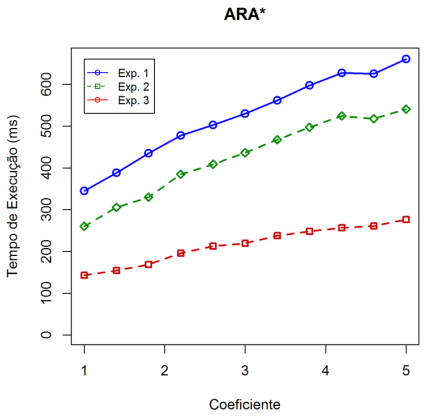
Figura 6.22: Belo Horizonte – Análise dos Tempos de Execução (ms) com Heurística nos Experimentos.

Na Figura 6.23 é apresentado o estudo do desempenho dos algoritmos *Anytime* em relação ao tempo e à heurística. Observou-se que ITS-ARA* (sem heurística) teve melhor desempenho para ϵ igual a 1,4 para os Experimentos 2 e 3 e de ϵ igual a 1,0 para o Experimento 1 (Figura 6.23(a)). Porém, ARA* obteve melhores resultados para o coeficiente heurístico igual a 1,0 em todos os experimentos (Figura 6.23(b)).

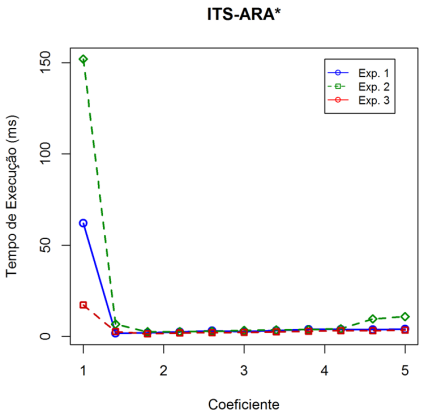
Com o uso da função heurística, visualizado nas Figuras 6.23(c) e 6.23(d), pode-se perceber que ARA* e ITS-ARA* tiveram seus melhores desempenhos com o



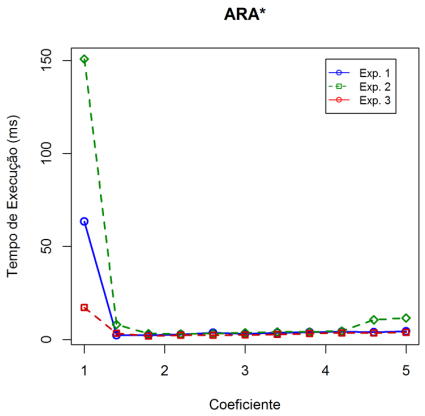
(a) ITS-ARA* sem Heurística.



(b) ARA* sem Heurística.



(c) ITS-ARA* sem Heurística.



(d) ARA* com Heurística.

Figura 6.23: Belo Horizonte - Variação do Coeficiente Heurístico nos Experimentos.

coeficiente igual a 1,4 nos Experimentos 1 e 3, sendo somente o valor de ϵ igual a 1,8 seus melhores desempenhos para o Experimento 2.

6.2.4 Luxemburgo

A análise dos resultados em relação aos tempos de execução da rede viária de Luxemburgo são analisados a seguir, sendo que seus valores sem a influência heurística são mostrados na Figura 6.25, onde ARA* foi superior ao ITS-ARA* somente no Experimento 3, tendo seu melhor resultado para ϵ igual a 1, superando inclusive os valores de referência de A* e Dijkstra. No entanto, para as demais variações do coeficiente ϵ, ITS-ARA* foi o mais efetivo em referência ao ARA* e ao Dijkstra. Para o Experimentos 1 e 2, ITS-ARA* foi superior ao ARA*, sendo superado pelo segundo somente no segundo Experimento nos valores de 1,8 e 2,2 (Experimento 2).

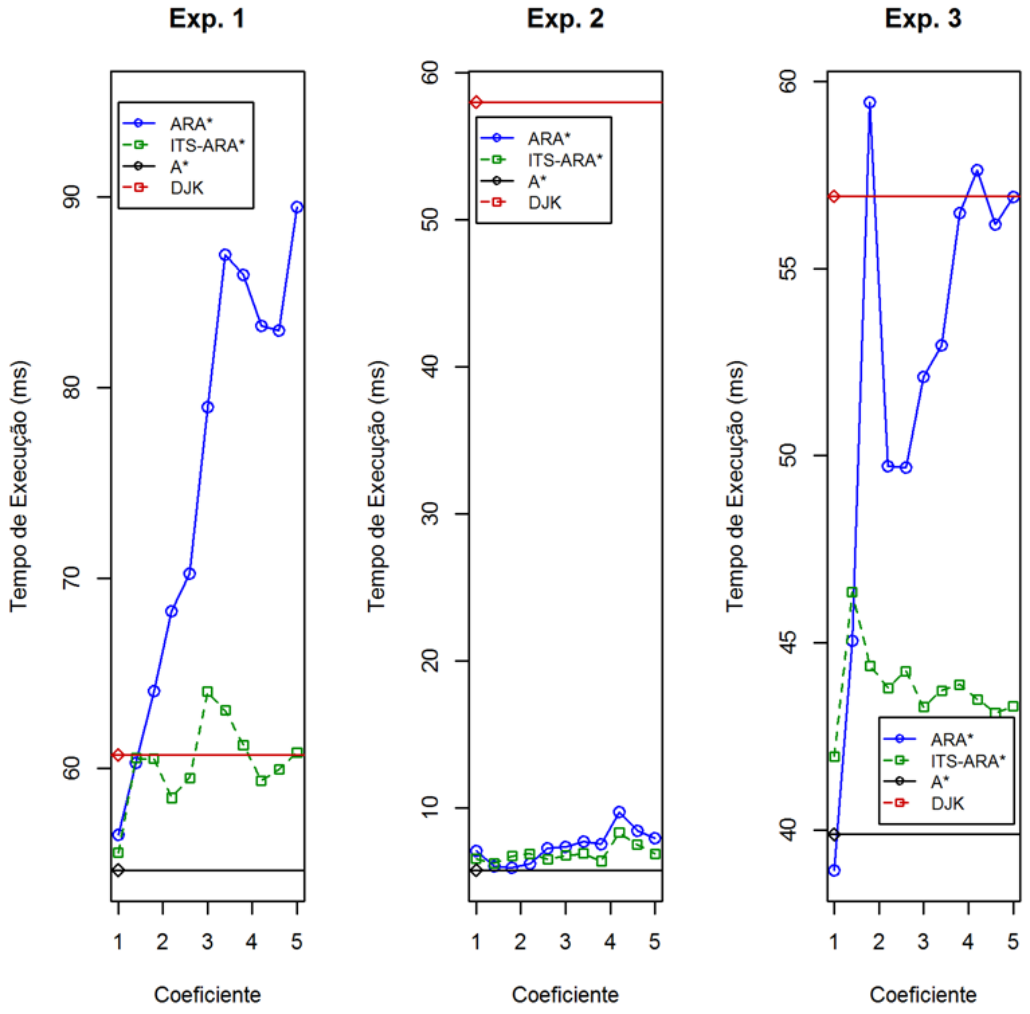


Figura 6.24: Luxemburgo – Análise dos Tempos de Execução (ms) sem Heurística nos Experimentos.

Para analisar os algoritmos quanto ao melhor coeficiente heurístico, observou-

se nas Figuras 6.26(a) e 6.26(b), que ITS-ARA* e ARA* tiveram seus melhores desempenhos para ϵ igual a 1,0 nos Experimentos 1 e 3. Porém para o Experimento 2, seus melhores desempenhos foram para ϵ igual a 1,4.

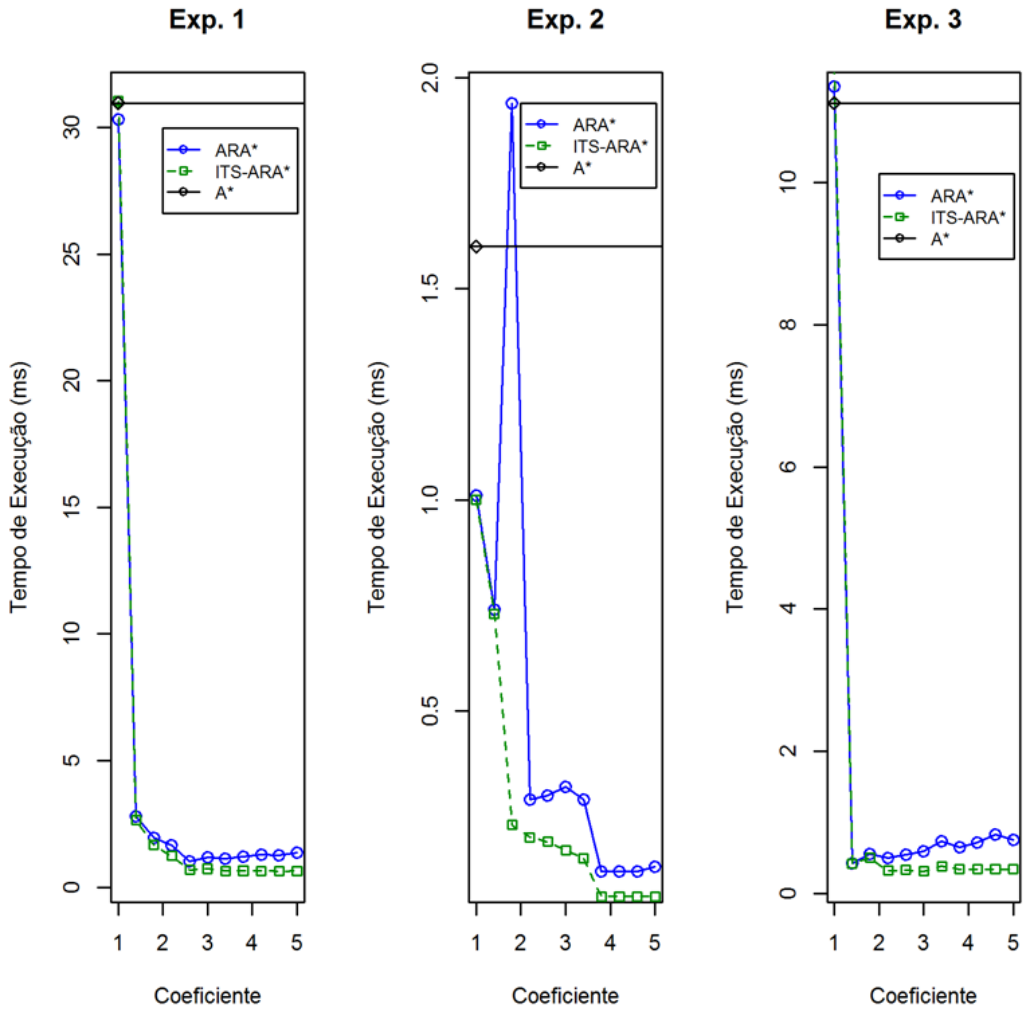
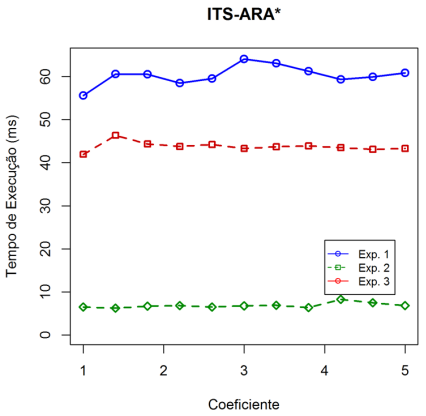
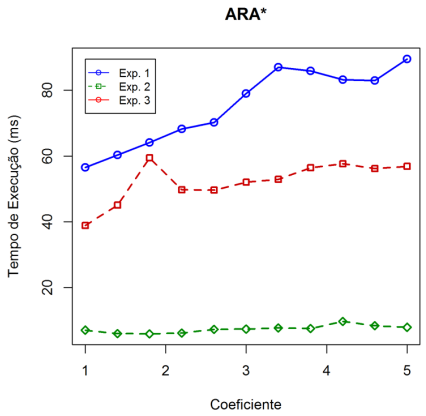


Figura 6.25: Luxemburgo – Análise dos Tempos de Execução (ms) com Heurística nos Experimentos.

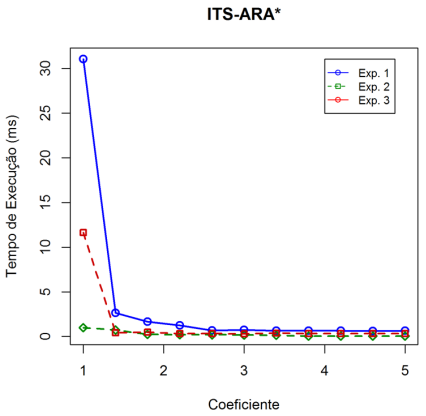
Para os experimentos com heurística, Figuras 6.26(c) e 6.26(d), o melhor desempenho de ITS-ARA* foi ϵ de 4,8 no Experimento 1, e nos intervalos de 3,8 a 5,0 de ϵ para o Experimento 2 e de ϵ igual a 3,0 para o Experimento 3. ARA* obteve seu melhor desempenho no Experimento 1 para ϵ igual a 2,6. No Experimento 2, ARA* foi mais eficiente no intervalo de 3,8 a 4,6 do coeficiente heurístico. E consequentemente, para o Experimento 3, quando coeficiente ϵ de ARA* foi igual a 1,4.



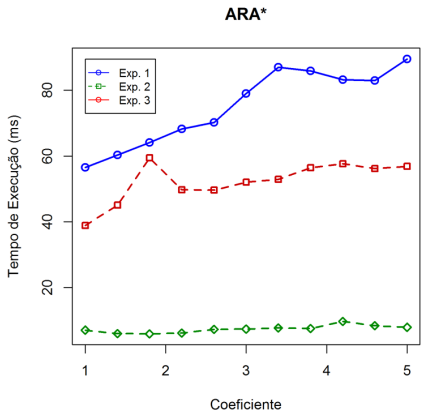
(a) ITS-ARA* sem Heurística.



(b) ARA* sem Heurística.



(c) ITS-ARA* com Heurística.



(d) ARA* com Heurística.

Figura 6.26: Luxemburgo - Variação do Coeficiente Heurístico nos Experimentos.

6.2.5 Le Havre

Os resultados dos experimentos na rede viária de Le Havre sem heurística são visualizados na Figura 6.27, onde se observa nos Experimentos 1 e 3, ITS-ARA* obteve melhor desempenho quando comparado ao ARA*, sendo que no Experimento 1 foi o mais expressivo deles. No Experimento 2, ARA* supera ITS-ARA* somente quando o coeficiente heurístico é igual a 1,0, nos demais ele só se iguala ao ITS-ARA*. Para os resultados com heurística (Figura 6.28), ITS-ARA* continua obtendo resultados melhores que os de ARA*.

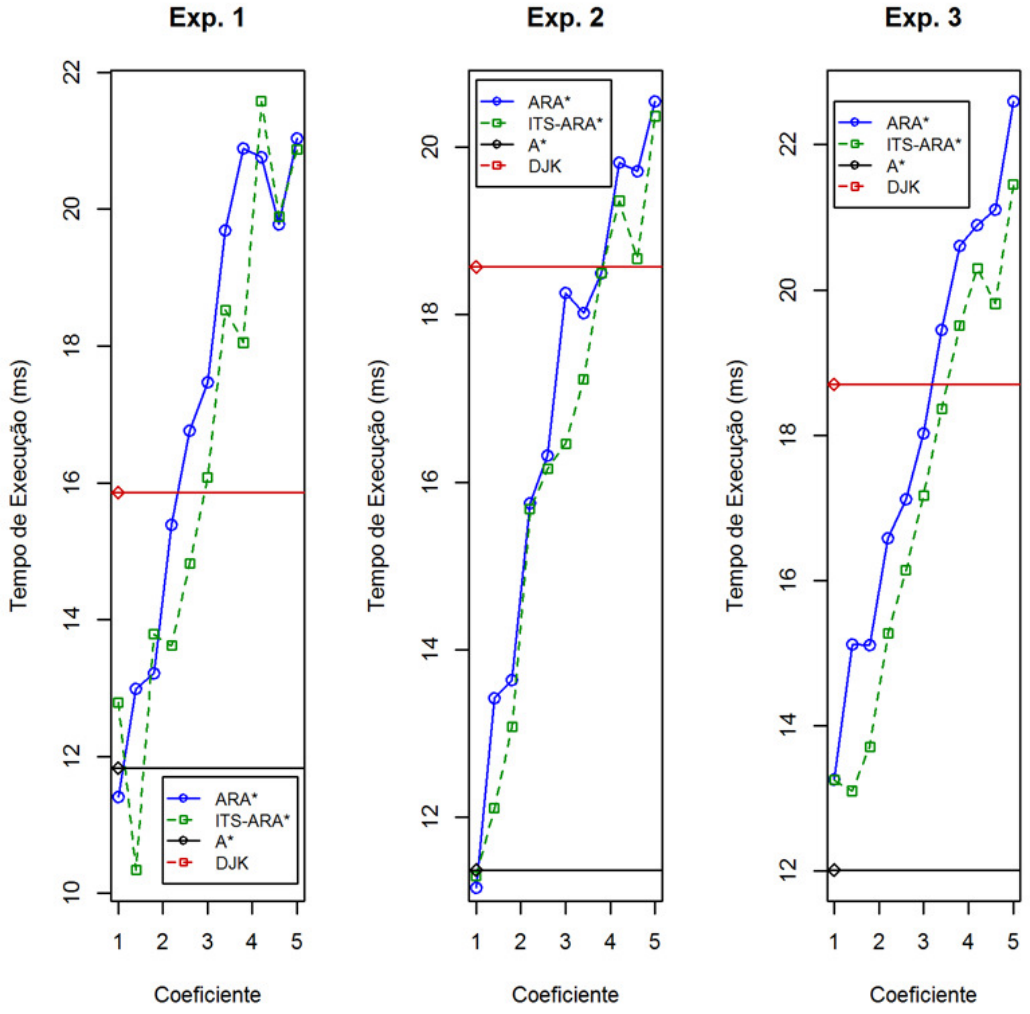


Figura 6.27: Le Havre – Análise dos Tempos de Execução (ms) sem Heurística nos Experimentos.

Na análise dos algoritmos quanto à variação do coeficiente sem heurística (Figuras 6.29(a) e 6.29(b)), para o ITS-ARA* percebeu-se que nos Experimentos 1 e 3, seu melhor desempenho é para o coeficiente ϵ igual a 1,4 e que no Experimento 2 seu melhor desempenho foi para o valor de 1,0. Para ARA*, seus melhores desempenhos foram todos para ϵ igual a 1. Nos resultados com heurística (Figuras 6.29(c)

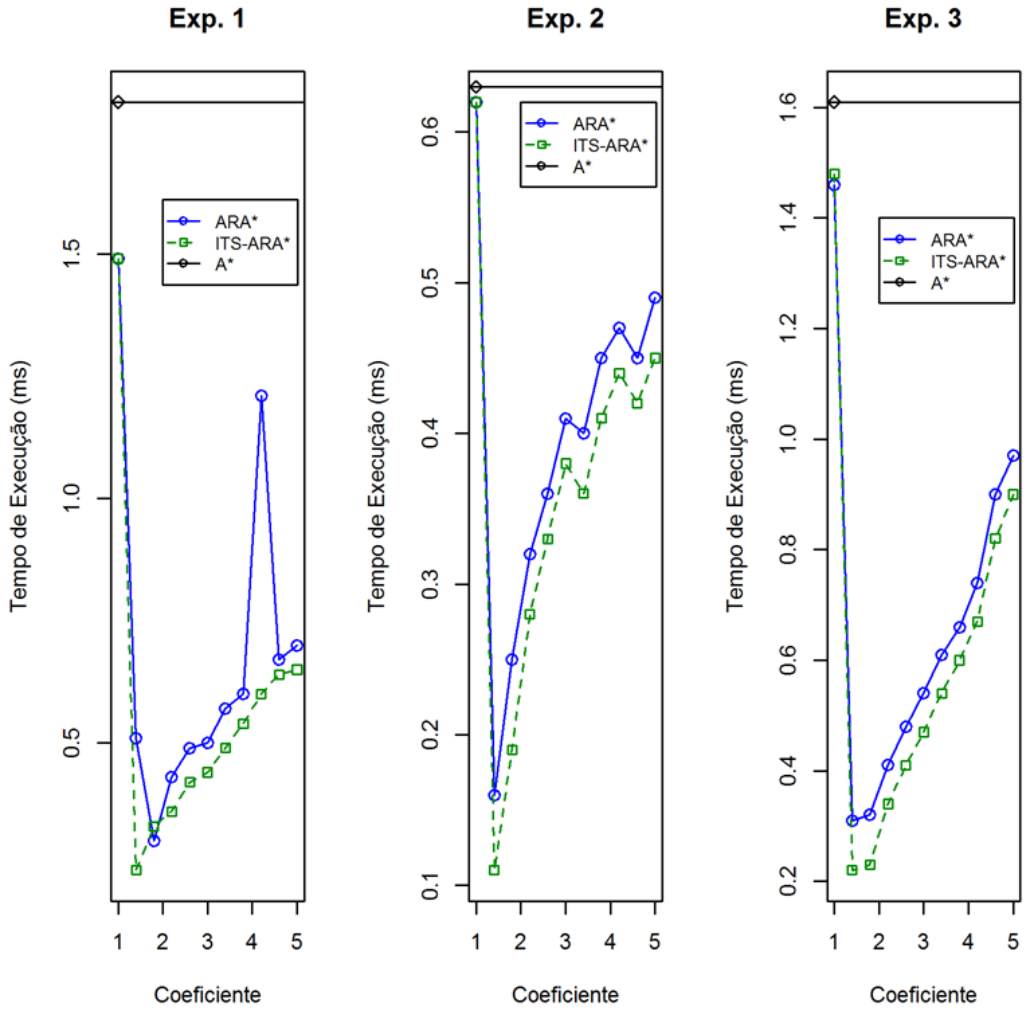
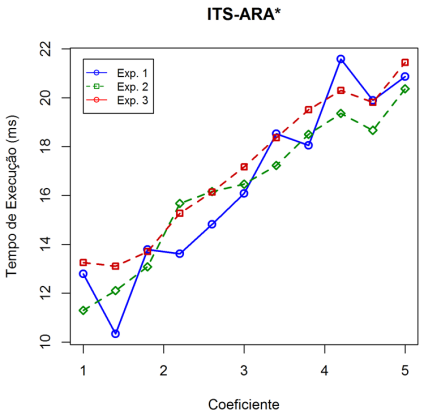
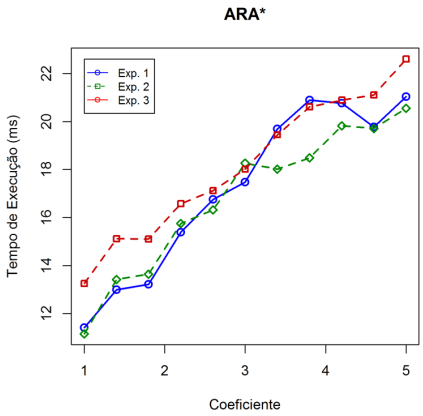


Figura 6.28: Le Havre – Análise dos Tempos de Execução (ms) com Heurística nos Experimentos.

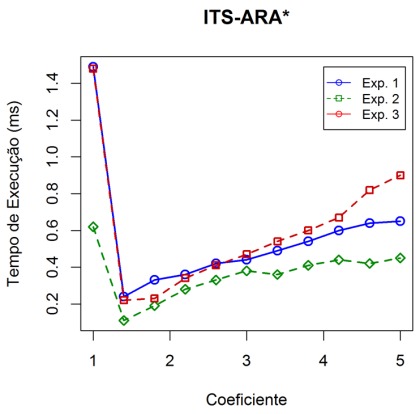
e 6.29(d)), observou-se que ITS-ARA* nos três experimentos, obteve melhor desempenho para o coeficiente igual a 1,4. No caso de ARA*, o Experimento 1 obteve melhor desempenho para ϵ igual a 1,8 e de 1,4 para os Experimentos 2 e 3.



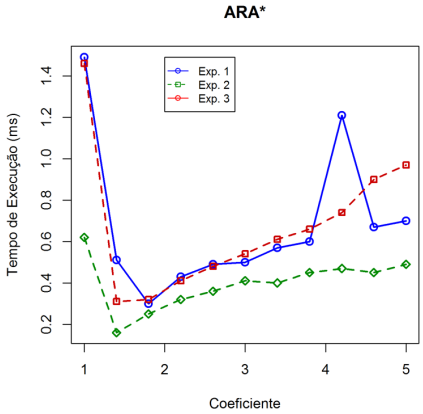
(a) ITS-ARA* sem Heurística.



(b) ARA* sem Heurística.



(c) ITS-ARA* com Heurística.



(d) ARA* com Heurística.

Figura 6.29: Le Havre - Variação do Coeficiente Heurístico nos Experimentos.

6.2.6 Munique

Os resultados sem heurística da rede de Munique podem ser vistos na Figura 6.30 e onde o desempenho do ITS-ARA* foi melhor que ARA* nos três experimentos apresentados, sendo o Experimento 1 o mais expressivo. Outra característica desse experimento é que o Experimento 2 é o único em que o algoritmo de Dijkstra foi mais rápido que A*. Nos resultados com heurística (Figura 6.31), observou-se ainda o melhor desempenho de ITS-ARA* em comparação com seu original.

A análise dos tempos pela variação do coeficiente ϵ pode ser observado nas Figuras 6.32(a) e 6.32(b) para resultados sem heurística, onde em todos os experimentos

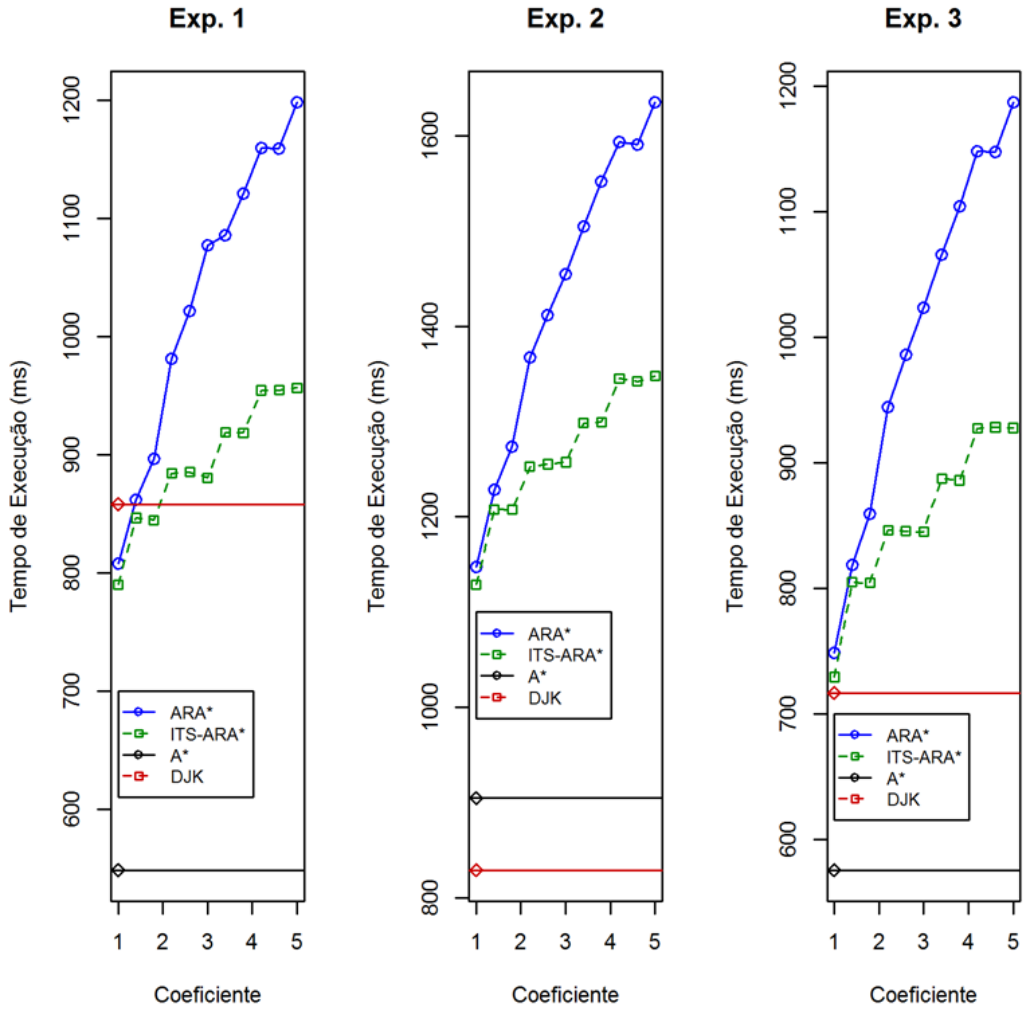


Figura 6.30: Munique – Análise dos Tempos de Execução (ms) sem Heurística nos Experimentos.

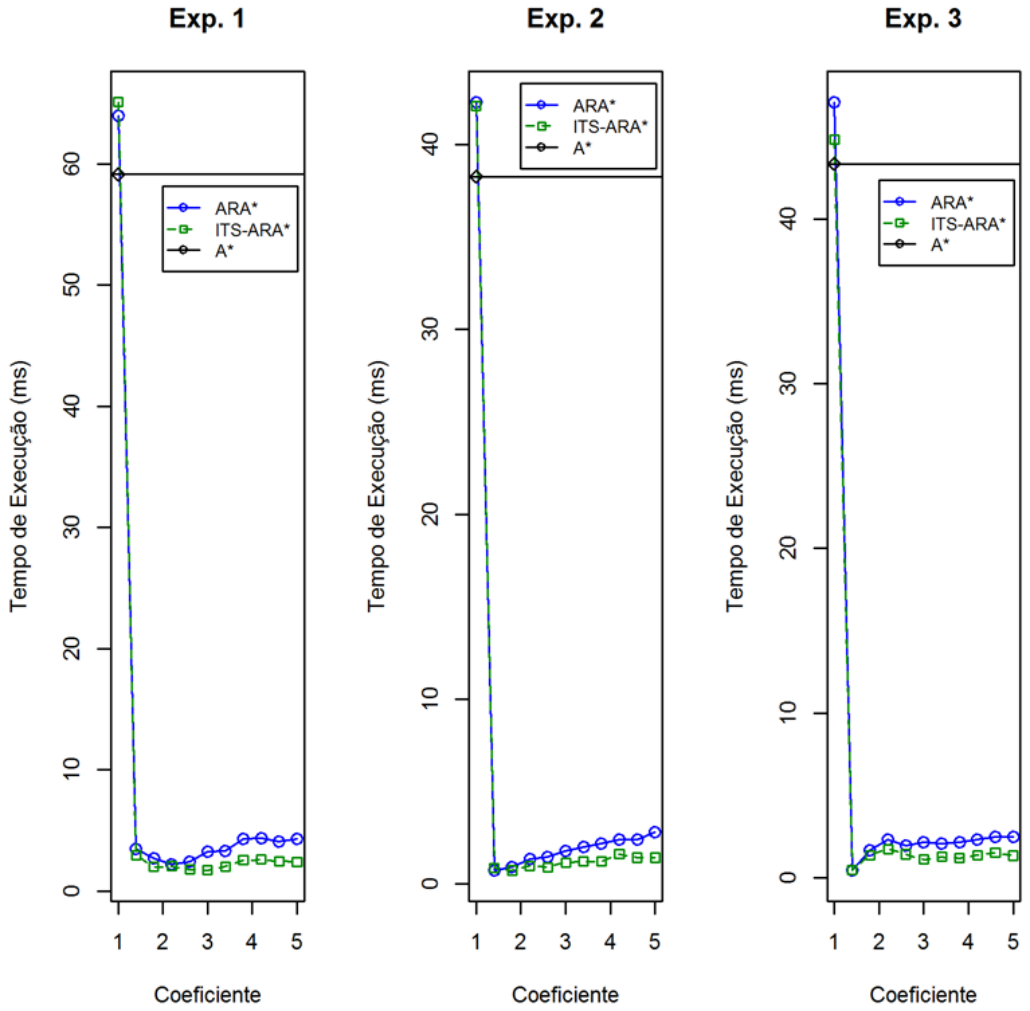
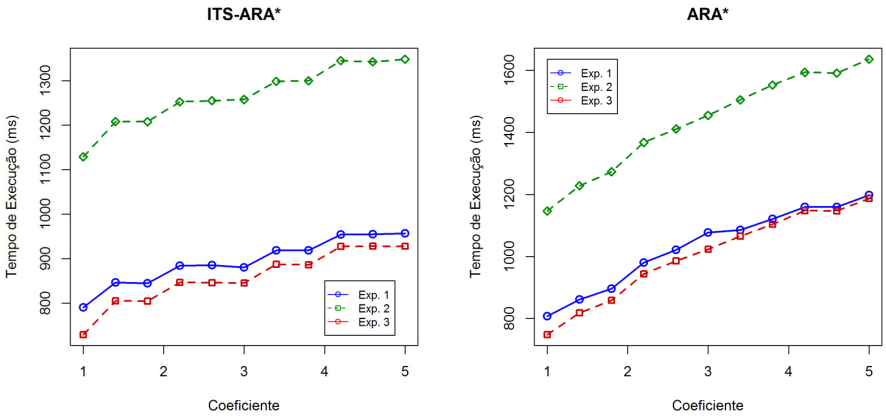


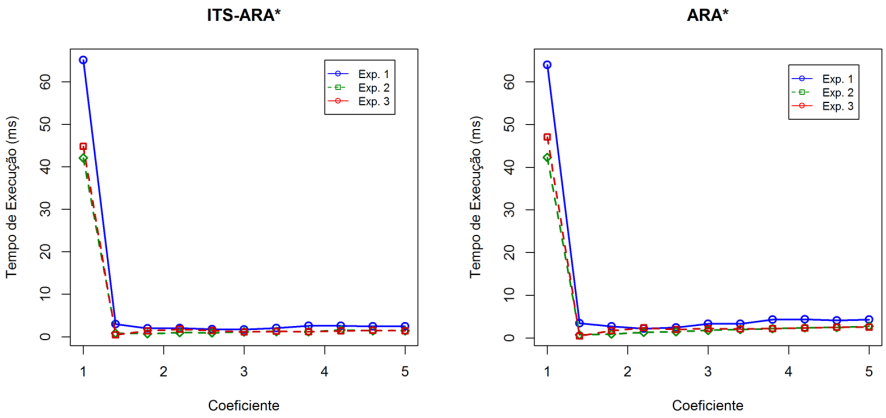
Figura 6.31: Munique – Análise dos Tempos de Execução (ms) com Heurística nos Experimentos.

os melhores desempenhos dos algoritmos se deram para o valor de 1,0. No entanto, ao se comparar com os resultados com heurística (Figuras 6.32(c) e 6.32(d)), observou-se que para os Experimentos 2 e 3 de ARA* e ITS-ARA*, seu melhor coeficiente foi de 1,4 e que para o Experimento 1, seu melhor desempenho foi de 3,0 para ITS-ARA* e de 2,2 para ARA*.



(a) ITS-ARA* sem Heurística.

(b) ARA* sem Heurística.



(c) ITS-ARA* com Heurística.

(d) ARA* com Heurística.

Figura 6.32: Munique - Variação do Coeficiente Heurístico nos Experimentos.

6.2.7 Nova Iorque

A Figura 6.33 ilustra os resultados sem heurística dos tempos de execução para a rede viária de Nova Iorque, assim como seus resultados com o uso da função heurística podem ser visualizados na Figura 6.34. O algoritmo ITS-ARA* apresentou um melhor desempenho nos três Experimentos apresentados, sendo que o Experimento 1 é o que mais evidencia tal fato, pois supera seus valores de referência (A* e Dijkstra). Observa-se também o comportamento do ARA* crescendo de acordo com a variação do coeficiente heurístico, o que não ocorre com ITS-ARA*, que apesar

de variar seus resultados consegue até minimizá-los, conforme observado no Experimento 2. Nos resultados com heurística, observou-se que ITS-ARA* manteve o melhor desempenho em relação ao ARA* nos três experimentos, além de ter sido superior aos valores de referência de A*.

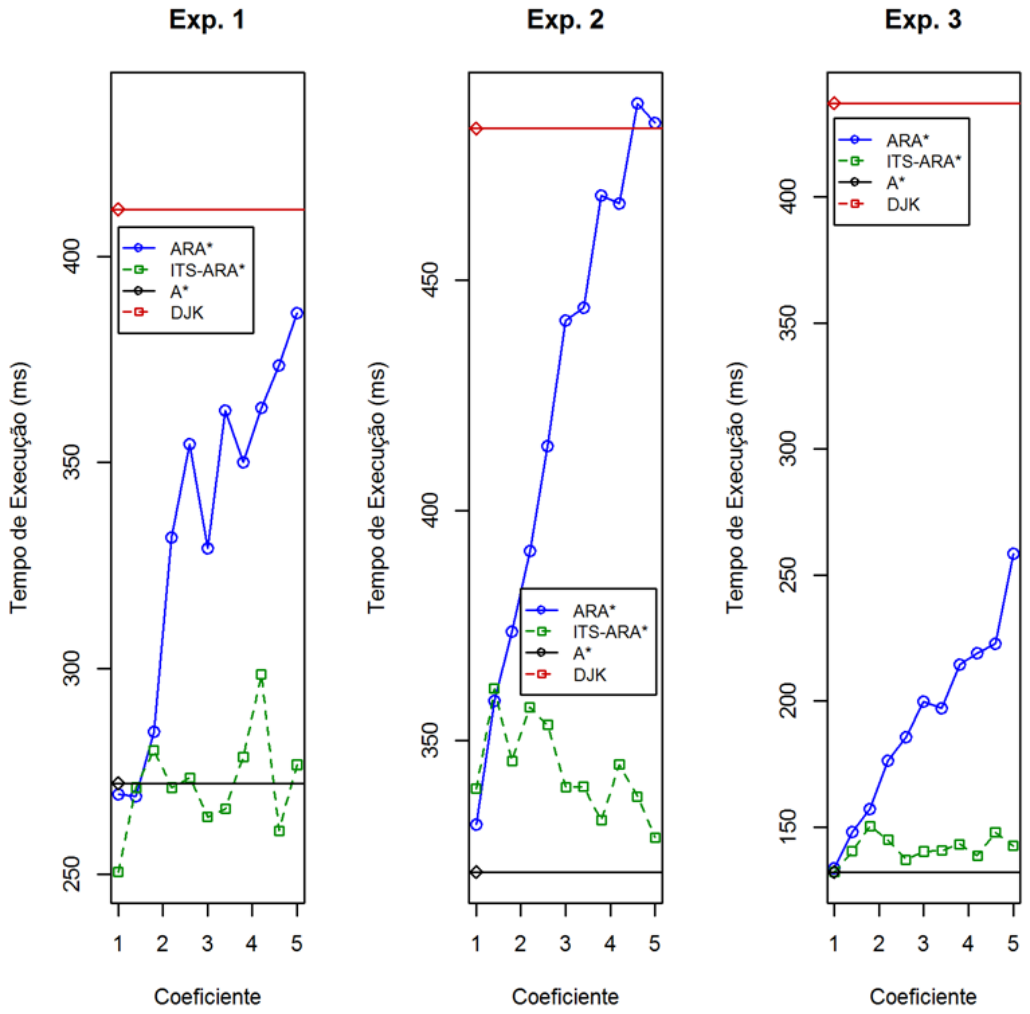


Figura 6.33: Nova Iorque – Análise dos Tempos de Execução (ms) sem Heurística nos Experimentos.

No estudo do desempenho quanto à variação dos coeficiente no tempo (Figura 6.35), observou-se no resultados sem heurística (Figuras 6.35(a) e 6.35(b)) que nos três experimentos o melhor valor do coeficiente heurístico corresponde a 1,0 para os algoritmos ITS-ARA* e de ARA*. Porém para os resultados com heurística (Figuras 6.35(c) e 6.35(d)), o melhor coeficiente de ITS-ARA* variou bastante, como no Experimento 1, onde seus melhores resultados se deram para ϵ igual a 3,4 e 4,2. O melhor coeficiente para o Experimento 2 foi de 4,6. Para o Experimento 3 seu melhor valor se deu com ϵ igual a 1,8. Para os resultados de ARA* temos que para o Experimento 1, seu melhor valor de ϵ foi de 1,8, de 3,0 para o Experimento

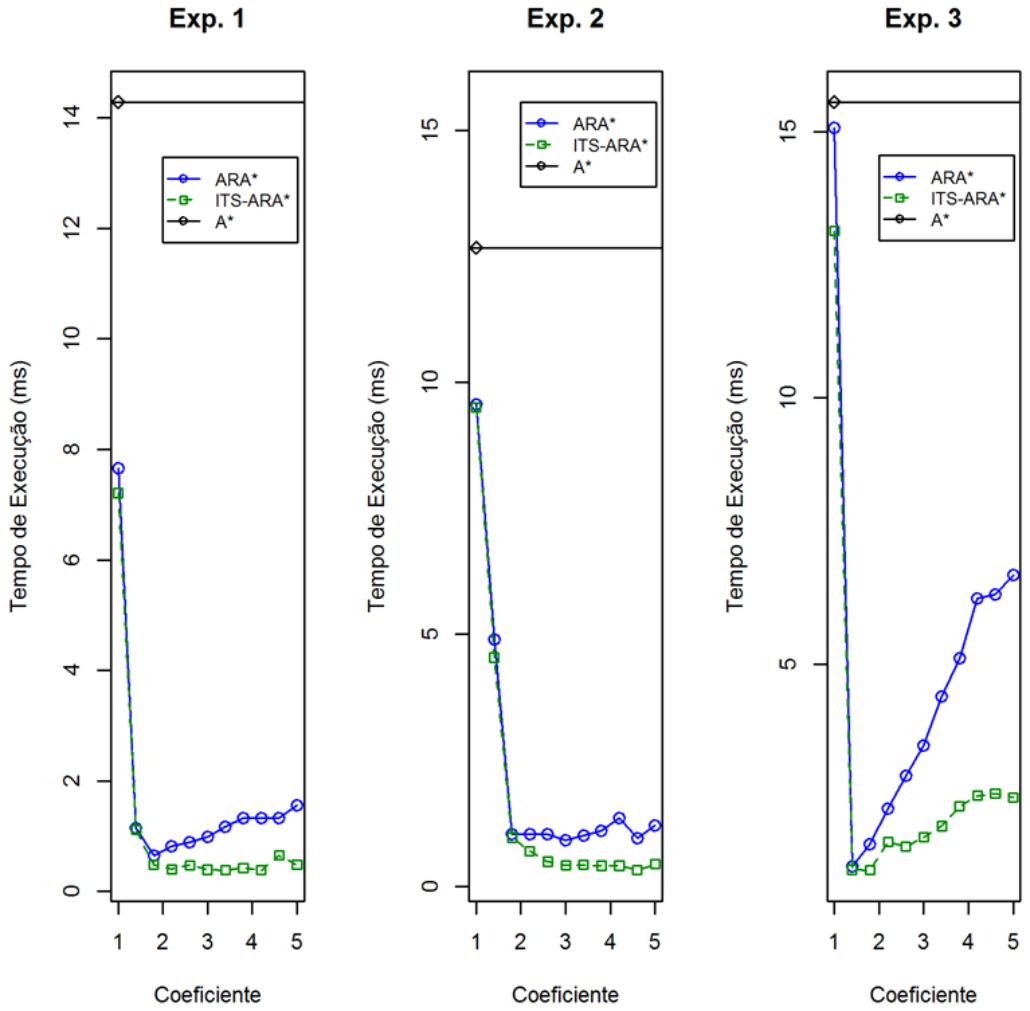
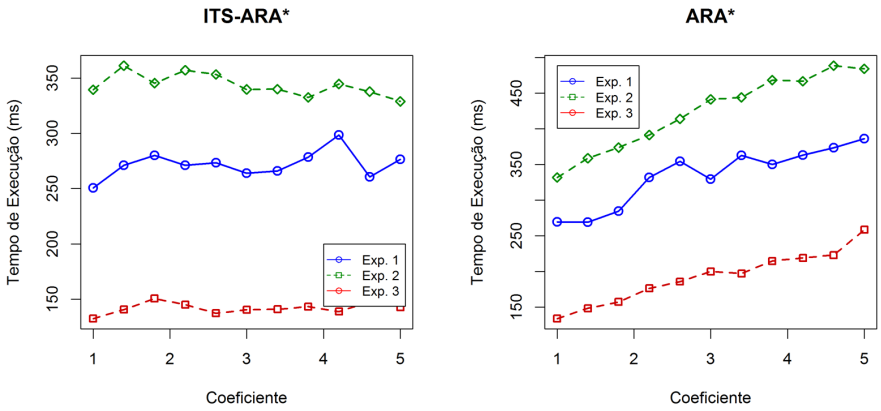


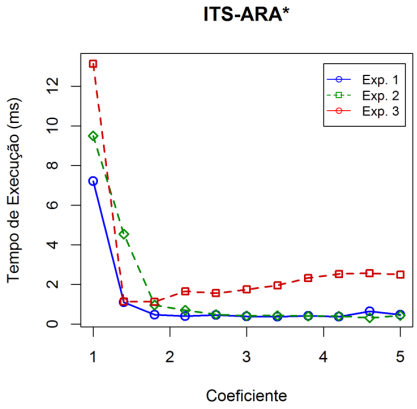
Figura 6.34: Nova Iorque – Análise dos Tempos de Execução (ms) com Heurística nos Experimentos.

2 e de 1,4 para o Experimento 3.

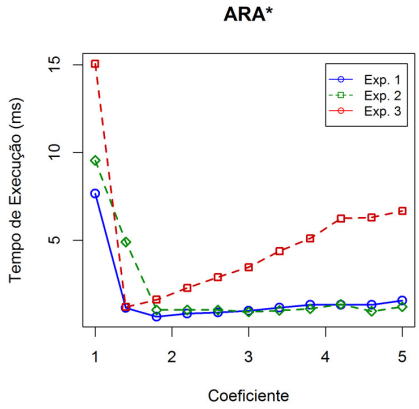


(a) ITS-ARA* sem Heurística.

(b) ARA* sem Heurística.



(c) ITS-ARA* com Heurística.



(d) ARA* com Heurística.

Figura 6.35: Nova Iorque - Variação do Coeficiente Heurístico nos Experimentos.

6.3 Análise de Resultados

Este experimento apresentou dois aspectos da análise de resultados, o primeiro tendo como foco a qualidade dos resultados de acordo com os vértices componentes das rotas encontradas em relação à variação do coeficiente heurístico. Já o segundo teve como foco a análise do tempo quanto à variação do ϵ. Ambas as medidas são importantes na investigação quanto ao melhor valor do coeficiente heurístico ϵ. A influência da heurística nos resultados foi outra observação realizada, podendo-se destacar que na maioria dos experimentos a ausência da heurística evidenciou os resultados de ARA* em relação a sua versão otimizada ITS-ARA*, porém sem destacar nenhum valor do coeficiente heurístico. A cidade de Munique foi a única que em seus resultados apresentou uma função constante para a variação de ϵ. Os

estudos na cidade de Nova Iorque sem uso da heurística, mostraram que os valores encontrados por A* foram menores que os valores de referência (Dijkstra).

Para os resultados com adição da função heurística percebeu-se que ARA* e ITS-ARA* obtiveram os mesmos comportamentos quanto à variação de ϵ em todas as redes viárias estudadas, inclusive em Munique que apresentou comportamento diferente das demais sem o uso da heurística. Porém, somente no Experimento 1 de Nova Iorque mais uma vez o valor de A* foi menor que o valor de referência do Dijkstra. A Tabela 6.2 apresenta os resultados das cidades em relação à variação do coeficiente heurístico. De um modo geral nos resultados sem uso da heurística, notou-se que houveram muitas variações nas quantidades de vértices componentes da rotas, resultando em médias com decimais. No entanto, os resultados de Nova Iorque foram os únicos que este comportamento se repetiu quando da adição da função heurística.

Com Heurística			
Cidades	**Experimento 1** ϵ	**Experimento 2** ϵ	**Experimento 3** ϵ
Manaus	1,4	1,0 até 5,0	1,4
Rio de Janeiro	1,8	1,0 e 2,2	1,8 e 2,2
Belo Horizonte	1,0	1,0	1,0
Luxemburgo	2,2	1,0	3,2
Le Havre	1,0	1,0 e 1,4	2,8 até 4,2
Munique	1,0	1,0	1,4
Nova Iorque	1,0	1,4	2,2

Tabela 6.2: Qualidade do Planejamento quanto ao Coeficiente Heurístico – Resultados com Heurística.

As medidas de tempo em relação à variação de ϵ sem uso da heurística, demonstraram na maioria dos testes a superioridade de ITS-ARA* em relação ao ARA*, sendo somente em Luxemburgo (Experimento 3) a única exceção, onde ARA* teve melhor desempenho nos valores de 1,0 e 1,4, nos demais ITS-ARA* foi superior. Outro dado que chama atenção é o do Experimento 2 referente à Manaus, como a rota sorteada foi relativamente pequena, os tempos variaram de 0,07 ms a 0,02 para os valores de 5,0 até 1,0 respectivamente. Na adição da função heurística, ITS-ARA* e ARA* tiveram comportamentos semelhantes em relação à variação de ϵ, porém sendo o primeiro o que teve melhor desempenho em relação ao original, esse comportamento já era esperado. No entanto, diferente das medidas em relação aos vértices, os resultados apontam para uma certa convergência de melhor resultado quando se varia o coeficiente heurístico ϵ para o algoritmo ITS-ARA*, estes dados podem ser acompanhados na Tabela 6.3. Pode-se observar que as cidades de Rio de Janeiro e Le Havre, indicam 1,8 e 1,4 respectivamente, como valores ideais para o coeficiente heurístico no planejamento de rotas. As cidades de Manaus

e Belo Horizonte apresentam resultados com tendências aos valores ϵ de 1,4 e 1,8 respectivamente. As cidades de Luxemburgo, Munique e Nova Iorque apresentaram intervalos de possíveis valores do coeficiente. Dessa forma, pode-se concluir que é necessário um estudo mais detalhado em relação às propriedades matemáticas das redes viárias, com intuito de verificar possíveis convergência para estes valores de ϵ.

Com Heurística			
Cidades	**Experimento 1** ϵ_1	**Experimento 2** ϵ_2	**Experimento 3** ϵ_3
Manaus	1,4	1,0 e 1,4	1,4
Rio de Janeiro	1,8	1,8	1,8
Belo Horizonte	1,4	1,8	1,8
Luxemburgo	4,6	3,8 até 5,0	3,0
Le Havre	1,4	1,4	1,4
Munique	3,0	1,8	1,4
Nova Iorque	3,4 e 4,2	4,6	1,8

Tabela 6.3: Tempo de Execução quanto ao Coeficiente Heurístico – Resultados com Heurística.

Capítulo 7

Conclusões e Trabalhos Futuros

Este capítulo sintetiza as conclusões desta pesquisa e as direções futuras, sendo que o objetivo é reforçar as contribuições e apontar possíveis direções para o prosseguimento da investigação. Desta forma, as conclusões desta tese são apresentados na Seção 7.1 e em seguida, na Seção 7.2 são apresentados as direções futuras deste trabalho.

7.1 Conclusões

O objetivo principal desta pesquisa foi a construção de uma solução para auxiliar a mobilidade nas cidades por meio da técnica de planejamento de rotas, com intuito de prover um balanceamento de carga nas redes viárias. Esta proposta possibilita duas visões do problema: a) Visão do usuário/condutor - onde o planejamento de rotas auxilia na escolha da melhor alternativa para chegar ao destino desejado, buscando minimizar o desconforto dos congestionamentos; e b) Visão do administrador do trânsito - auxiliando na administração do tráfego por meio da divulgação de informações das vias sensoriadas. Outro ponto a ser destacado, é que este trabalho utiliza-se de dados coletados por qualquer sistema de sensoriamento que capte as velocidades médias por trecho de uma rede viária, como exemplo pode-se citar o sistema COTraMS proposto por Ribeiro Júnior [77], que efetua o monitoramento veicular utilizando a rede IEEE 802.11 b/g.

Nesta pesquisa foram abordados aspectos de planejamento de rotas, análise em redes viárias e teste de viabilidade em uma plataforma computacional com características restritivas. Para isto foi utilizado o Raspberry Pi, nos modelos B e B+. A utilização da referida plataforma foi motivada por prova de conceito em um hipotético problema de contingência, onde a comunicação foi interrompida por algum motivo, de modo que o condutor precise planejar a sua rota em um dispositivo portátil e com capacidade de comunicação, ou seja, um *smartphone*. O Raspberry Pi foi escolhido por possuir as características computacionais semelhantes a um

smartphone padrão, bem como possibilitar a construção de um protótipo de uso mais específico.

Com intuito de viabilizar a referida pesquisa, foi criada uma metodologia a ser utilizada. Em seguida esta metodologia foi automatizada por meio da ferramenta *GTAPlanning*, proporcionando o suporte na análise das características das redes viárias, bem como dos algoritmos de planejadores de rota. Para o desenvolvimento da ferramenta foi feito um estudo sobre Teoria dos Grafos, Planejamento de Rotas e ITS. Foram abordados propriedades matemáticas sobre grafos, estudos de algoritmos de busca clássicos e de uso na robótica (*Anytime*). Em ITS foram definidos conceitos e as melhores práticas adotadas, bem como o panorama de ITS no mundo e sensoriamento do tráfego. Os conceitos apresentados foram de grande valia no desenvolvimento deste trabalho, além de fornecer conhecimento sobre o estado da arte das áreas em questão. Em seguida, esta pesquisa analisou as abordagens de alguns trabalhos correlatos existentes na literatura. Os trabalhos foram avaliados de acordo com os aspectos: a) sistemas de monitoramento de trânsito; b) sugestão de rotas em redes viárias; c) análise de experimental de algoritmos. A partir destas informações foi montado um comparativo entre os trabalhos correlatos.

O *software* construído para dar suporte à presente pesquisa, foi desenvolvido na IDE Eclipse utilizando-se a linguagem Java. A escolha da referida linguagem deve-se ao fato da mesma ser multiplataforma e sem a necessidade de recompilação do *software*, além de ter fácil integração com o sistema operacional para robôs ROS (*Robot Operating System*) [78]. Para a análise das redes viárias e testes dos algoritmos planejadores de rotas realizados pelo *software* de planejamento de rotas, foi realizado um processo composto de três etapas: (1) escolha da cidade em um mapa colaborativo digital, cujas informações são expressas em uma linguagem de marcação *OSM*; (2) extração das informações contida na linguagem de marcação e convertidas em grafos; e (3) análise e planejamento de rotas em redes viárias utilizando conceitos de grafos. Vale ressaltar que foram utilizadas as implementações dos algoritmos de Dijkstra e A* provenientes do *GraphStream*, devido ao primeiro ter uma implementação (*Heap de Fibonacci*) mais eficiente para grafos muito grandes. E a segunda para poupar tempo de implementação. Dessa forma, somente ARA* e ITS-ARA* foram implementados.

A integração do mundo real com a Teoria dos Grafos foi conseguida pelo *GTAPlanning*, onde é possível utilizar o mapa colaborativo *OpenStreetMap* e realizar uma análise das características da rede viária alvo até a análise de algoritmos de planejamento de rotas além de inferir dados reais dos fluxos de trânsito. Para a análise das redes viárias e testes dos algoritmos planejadores de rotas foi realizado um processo composto de três etapas: (1) escolha da cidade em um mapa colaborativo digital, cujas informações são expressas em uma linguagem de marcação *OSM*;

(2) extração das informações *OSM* no software; e (3) análise e planejamento de rotas em redes viárias utilizando dados inseridos.

O primeiro aspecto analisado foi a eficiência computacional dos planejadores de rotas. Nesta etapa foram medidos os tempos de execução, memória consumida e quantidade de vértices e de arestas componentes dos resultados. Foram utilizados nos testes quatro algoritmos de busca, os clássicos Dijkstra e A* e os da classe *Anytime*, ARA* (originário da robótica) e sua versão otimizada ITS-ARA*. Como esses três últimos algoritmos utilizam-se de heurística para determinação dos caminhos, foi realizado o estudo da influência da heurística no planejamento de rotas. A segunda abordagem analisada se refere à análise dos caminhos gerados pelos algoritmos em relação à frequência de utilização dos trechos da rede viária. Para isso foram utilizados dados reais do tráfego dos ônibus do Rio de Janeiro. Foram realizados experimentos em dez janelas de tempo, sendo que os pares de origem e de destino foram sorteados aleatoriamente, em um total de duzentas repetições por janela de tempo. Para esta abordagem, foram utilizadas duas ponderações na rede viária, uma relacionada à distância expressa em metros e outra relacionada ao tempo de percurso. A Fórmula de Haversine foi utilizada para calcular as distâncias a partir das informações extraídas do *OpenStreetMap*. Para a segunda ponderação, seu cálculo foi extraído dos dados de GPS dos ônibus do Rio de Janeiro, levando-se em consideração a velocidade calculada e as informações de distância referentes ao trecho.

Para os testes de viabilidade computacional em uma plataforma com características restritivas foi utilizado um Raspberry Pi, cujo cenário de testes foram duas cidades brasileiras. A influência da heurística no planejamento de rotas, bem como o tempo de execução, memória consumida e a quantidade de vértices e arestas foram medidos. O experimento teve por objetivo ser uma prova de conceito, devido ao *hardware* do *Raspberry Pi* ser bem semelhante aos modelos básicos utilizados em *smartphones*, com isso pretendia-se testar sua viabilidade técnica. Como resultado dos experimentos, observou-se que ITS-ARA* obteve um melhor desempenho que ARA* quanto ao tempo de execução e com e sem o uso da função heurística. Sendo que seus resultados mais expressivos são quando faz uso da heurística. No entanto, quanto à qualidade da rota gerada, ITS-ARA* e ARA* tiveram seus desempenhos bastante próximos sem o uso da heurística. Por outro lado com a utilização da heurística, seus resultados foram os mesmos.

Para o segundo experimento, foram utilizados dados reais extraídos dos fluxos de tráfego da cidade do Rio de Janeiro por meio dos dados coletados pelos GPS dos ônibus da referida cidade. Foram utilizadas duas ponderações no experimento, a da distância e a do tempo de trajeto. Foram avaliados, os tempos de execução, memória consumida, quantidade vértices, bem como os caminhos gerados pelos planejadores

de rotas, velocidade média, tempo de chegada, distância percorrida. O experimento utilizou-se de três janelas de tempo: a) de 07 h até 08 h; b) de 11 h até 12 h; e c) de 16 h até às 17 h. Os resultados foram bastante heterogêneos quanto à utilização das referidas ponderações. Para a ponderação da distância, foi possível observar a frequência dos trechos mais utilizados e seus respectivos comportamentos, onde se pode destacar que os resultados dos algoritmos *anytime* foram mais equilibrados, melhor distribuídos em relação aos resultados de Dijkstra e A*, nas três janelas de tempo estudadas. No entanto para a ponderação tempo de trajeto, os resultados foram bastante semelhantes entre os algoritmos planejadores de rotas, diferenciando-se mais notoriamente pelos tempos de cálculo de rota, sendo que o ITS-ARA* foi o que obteve melhor desempenho entre os demais.

7.2 Trabalhos Futuros

Esta pesquisa resultou no desenvolvimento de uma ferramenta de planejamento de rotas *GTAPlanning*, onde testes com algoritmos planejadores e inferência de dados reais de velocidade de tráfego podem ser inseridos. A seguir, serão descritas possíveis contribuições para o desenvolvimento da pesquisa. Uma possível extensão deste trabalho refere-se a migração do módulo de planejamento de rotas para um aplicativo em *Android*. Esta migração é possível e não tão impactante, pois os aplicativos desenvolvidos para esta plataforma são em sua maioria escritos em Java, que foi a linguagem escolhida para o desenvolvimento desta solução. Um possível complemento desta referida proposta seria a utilização da GPU *Graphics Processing Unit* para acelerar os processo de planejamento de rotas, possivelmente usando uma solução com OpenCL (*Open Computing Language*) e Java [79, 80].

Outra interessante direção é o desenvolvimento de monitoramento e controle de tráfego (Figura 7.1), que auxiliaria às companhias de engenharia de trânsito no gerenciamento e monitoramento das redes viárias. O sistema será descrito a seguir.

- **COTraMS** – Responsável pelo sensoriamento do tráfego veicular, coletando as velocidade médias por trecho;
- **Monitoramento** – Proporciona o acompanhamento em tempo real das condições da rede viária;
- **Fluxo de Dados** – Refere-se a um banco de dados que armazena o fluxo de tráfego diário. Estas informações servirão para estudos de viabilidade técnica, simulações e relatórios de atividades relacionadas ao tráfego;
- **Controle e Recomendações** – Módulo responsável pelo controle do sistema como um todo. A partir dele é possível inferir ponderações na rede viária com

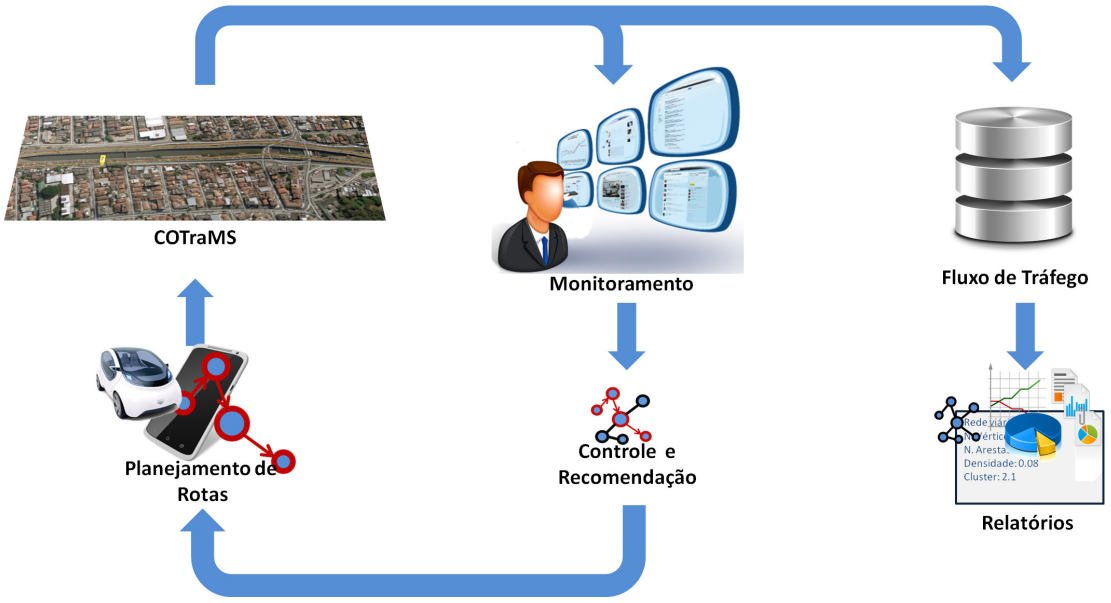


Figura 7.1: Sistema de Monitoramento e Controle de Tráfego.

intuito de favorecer uma situação. Por exemplo, caso haja a necessidade de se fazer um reparo em uma via, ou qualquer outro motivo que seja necessário deixá-la inapta para tráfego de veículos, basta o operador aumentar o peso nesta via para que quando for calculado um planejamento de rotas, tal via seja inacessível para compor uma rota. Situações de emergência, são outro tipo de aplicação deste sistema, ou seja, privilegia-se os veículos para atender a ocorrência, enquanto os demais são direcionados para outras rotas;

- **Planejamento de rotas** – Módulo responsável para informar a melhor rota de acordo com o algoritmo planejador setado;

Outra linha de pesquisa interessante é o estudo das propriedades matemáticas da rede viária, com o intuito de relacionar com o coeficiente heurístico presente nos algoritmos de busca *anytime*. Por fim, uma possível extensão deste trabalho é a implementação do módulo de planejamento de rotas em um veículo autônomo, para isto pode-se utilizar o sistema operacional ROS fazer sua integração com o mesmo. Cabe ressaltar que o referido sistema operacional possui suporte para Java.

Apêndice A

Raspberry Pi

O Raspberry Pi (RPi) é um microcomputador de dimensões reduzidas, do tamanho de um cartão de crédito (85,60 mm X 53,98 mm X 17 mm) e pesando 45 gramas. O *hardware* está integrado em uma única placa de circuito impresso, memória de acesso aleatório (*Random-Access Memory* (RAM)). É desenvolvido pela Fundação Raspberry Pi com o objetivo de estimular o ensino de ciência da computação para as crianças e os jovens. Atualmente esta plataforma está disponível em duas gerações: a) composta por quatro modelos (A e A+; B e B+); e b) Raspberry Pi 2 lançada em fevereiro de 2015. A Tabela A.1 apresenta as diferenças entre os modelos do Raspberry Pi.

	Modelo A	**Modelo A+**	**Modelo B**	**Modelo B+**	**RPi 2**
Lançamento	Fev. 2012	Nov. 2014	Abr. 2012	Jul. 2014	Fev. 2015
Chip	BCM2835	BCM2835	BCM2835	BCM2835	BCM2836
CPU	700 MHz	700 MHz	700 MHz	700 MHz	900 MHz
RAM	256 MB	256 MB	512 MB	512 MB	1 GB
Ethernet	Não	Não	Sim	Sim	Sim
Cartão	SDCard	microSD	SDCard	microSD	microSD
USB	1 porta	1 porta	2 porta	4 porta	4 portas

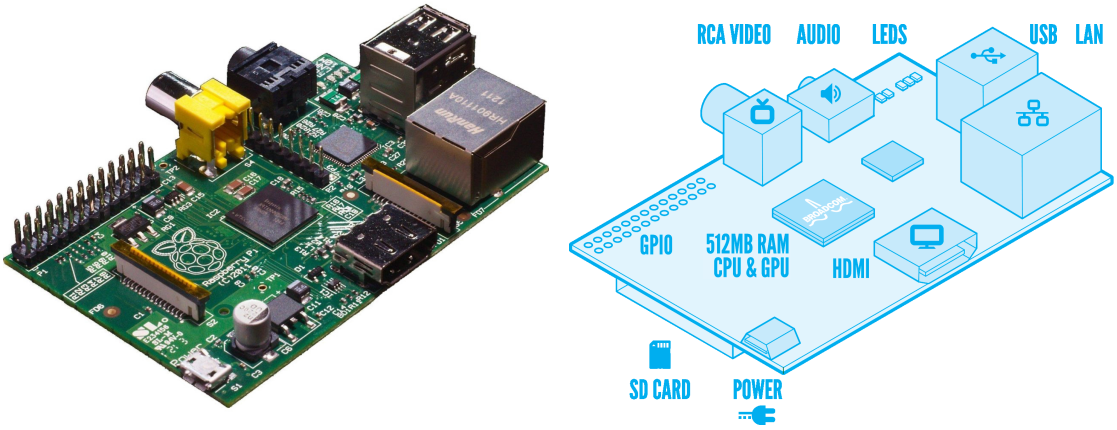
Tabela A.1: Modelos do Raspberry Pi.

Os modelos utilizados nesta pesquisa foram o *B* e o *B+*, sendo que suas principais alterações podem ser destacadas no resumo a seguir:

- 4 portas USB 2.0;
- Redução de consumo de energia entre 0.5W e 1W em comparação com o modelo B
- Áudio foi melhorado
- Substituição do *slot* de cartões SD para uma versão micro SD.

- Aumento no número de pino do GPIO de 26 para 40.

Nas Figuras A.1(a) e A.1(b) é possível observar o *hardware* e a representação esquemática do RPi modelo B respectivamente. Todos os modelos da primeira geração são baseados no *System on Chip* (SoC) BCM2835, o mesmo encontrado em muitos *smartphones* no mercado. Este SoC é constituído de um processador ARM com 700Mhz, uma Unidade de Processamento Gráfico (GPU, em inglês *Graphics Processing Unit*) e memória RAM compartilhada com a GPU, de 256 MB para os modelos A e A+, de 512 MB para os Modelos B e B+. Para o modelo de segunda geração, o RPi 2, apresenta um Soc BCM2835, com 1 GB de memória RAM e um processador de quatro núcleos e é capaz de usar o sistema operacional da *Windows 10* da Microsoft. Isto é possível devido ao *boot* ser realizado a partir do cartão microSD.



(a) Placa. (b) Circuito lógico.

Figura A.1: Raspberry Pi .

Os modelos do RPis possuem uma porta micro-usb para alimentação (5V) e a conexão com a rede pode ser pela interface de Ethernet (LAN) ou por meio de um acessório de rede sem fio por USB. A ligação de vídeo pode ser efetuada através de duas formas:

- HDMI (High-Definition Multimedia Interface) – transmissão de vídeo de ala definição.
- RCA (Composite Out) – saída composta bastante limitada em termos de gráficos

O RPi se utiliza de cinco LEDs para o controle dos seus estados, ou seja, cada LED informa sobre as ações que estão sendo executadas no mesmo:

- ACT - D5 (Verde) – acesso ao cartão SD.

- PWR - D6 (Vermelho) — alimentação do Raspberry Pi.
- FDX - D7 (Verde) – ligação de rede (LAN) Full Duplex.
- LNK - D8 (Verde) – atividade com a rede (LAN).
- 100 – D9 (Amarelo) – ligação com a rede (LAN) a 100Mbit/s.

Apêndice B

Resultados CDFs

Os resultados CDFs dos experimentos realizados no Capítulo 5 são apresentados a seguir, de acordo com suas respectivas janelas de tempo e tabelas de coeficientes utilizados na plotagem das distribuições.

B.1 Distância

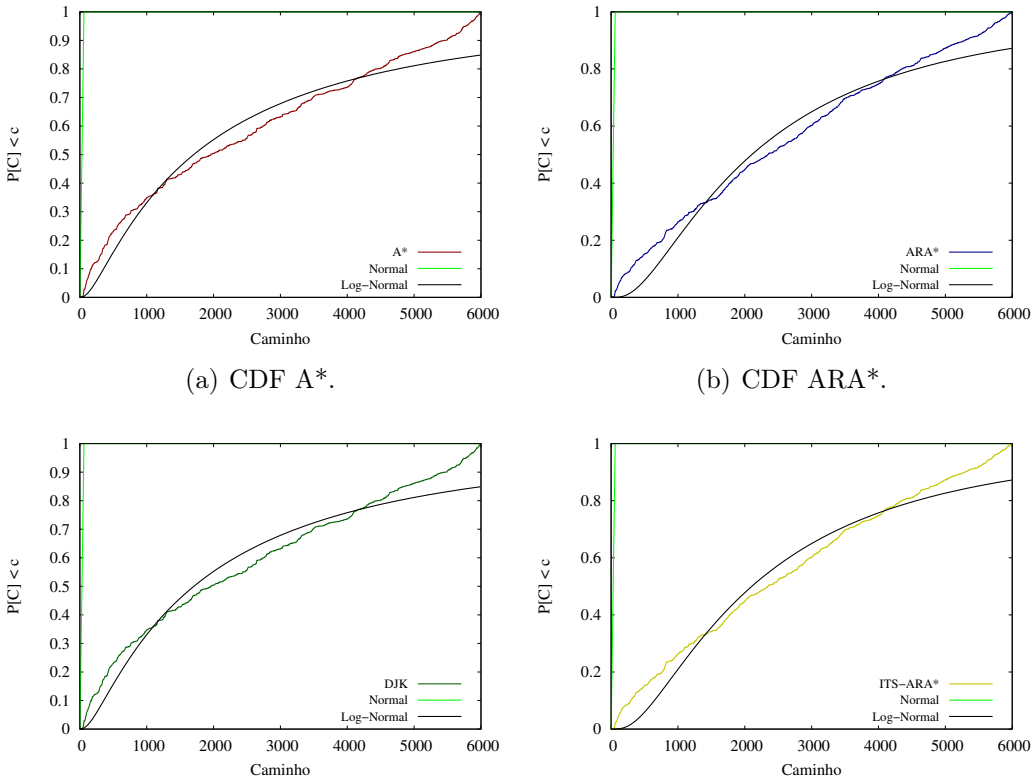
A seguir são apresentados os dados referentes à ponderação da distância de acordo com suas janelas de tempo, a Tabela B.1 mostra os valores (μ,σ) utilizados nas distribuições *log* e *log normal*, sendo esta última a que mais se aproximou do comportamento da CDF.

Janela de Tempo	**Distância**			
	DJK	**A***	**ARA***	**ITS-ARA***
	Parâmetros (μ;σ)			
07–08 h	(7,441;1,218)	(7,439;1,221)	(7,652;9,203)	(7,652;9,203)
11–12 h	(7,464;1,128)	(7,460;1,137)	(7,627;8.715)	(7,627;8,715)
16–17 h	(7,455;1,167)	(7,452;1,170)	(7,668;8,836)	(7,668;8,836)

Tabela B.1: Parâmetros CDFs utilizados nas distribuições – Distância.

B.2 Tempo de Percurso

Nesta Seção são apresentados os resultados CDF com ponderação de tempo de percurso. Da mesma maneira que sua anterior, nota-se que a distribuição que está mais próxima ao valor da CDF em todos os resultados é o *log normal*. A Tabela B.2 mostra os coeficientes (μ,σ).



(a) CDF A*.

(b) CDF ARA*.

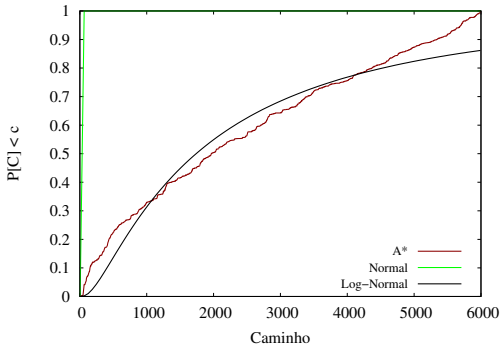
(c) CDF Dijkstra.

(d) CDF ITS-ARA*.

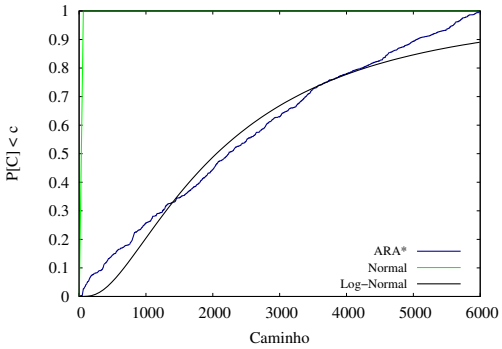
Figura B.1: CDF algoritmos – 07 h até 08 h – Distância.

Janela de Tempo	**Tempo de Percurso**			
	DJK	**A***	**ARA***	**ITS-ARA***
	Parâmetros (μ;σ)			
07–08 h	(7,378;1,246)	(7,378;1,246)	(7,397;1,211)	(7,397;1,211)
11–12 h	(7,359;1,137)	(7,359;1,137)	(7,354;1,137)	(7,354;1,137)
16–17 h	(7,422;1,088)	(7,422;1,088)	(7,418;1,089)	(7,418;1,089)

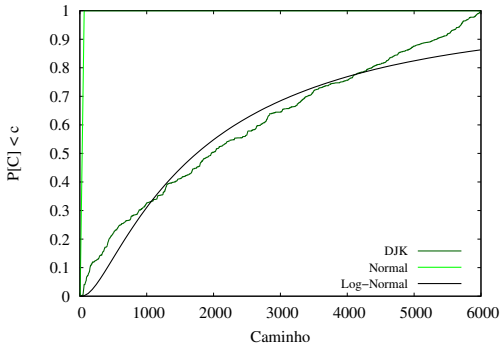
Tabela B.2: Parâmetros CDFs utilizados nas distribuições – Tempo de Percurso.



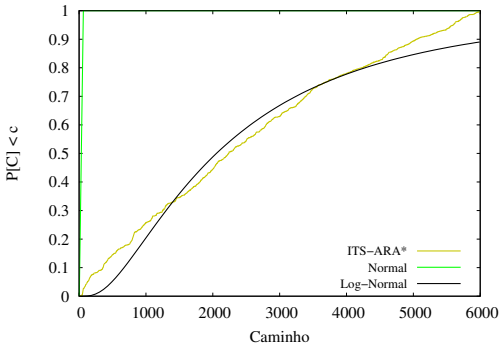
(a) CDF A*.



(b) CDF ARA*.

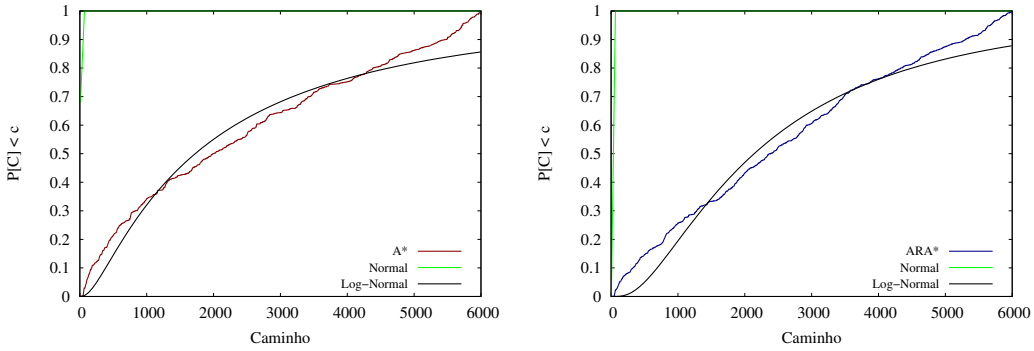


(c) CDF Dijkstra.



(d) CDF ITS-ARA*.

Figura B.2: CDF algoritmos – 11 h até 12 h – Distância.



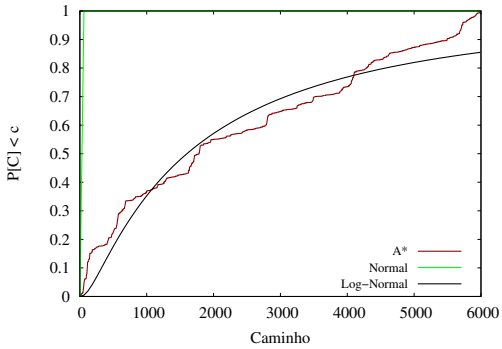
(a) CDF A*.

(b) CDF ARA*.

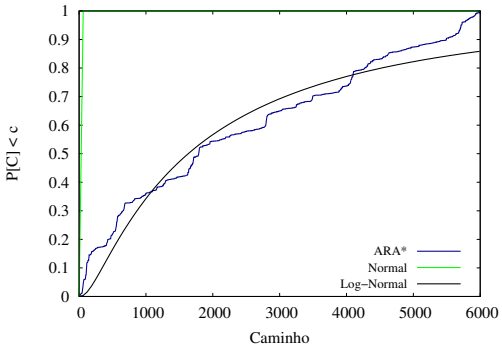
(c) CDF Dijkstra.

(d) CDF ITS-ARA*.

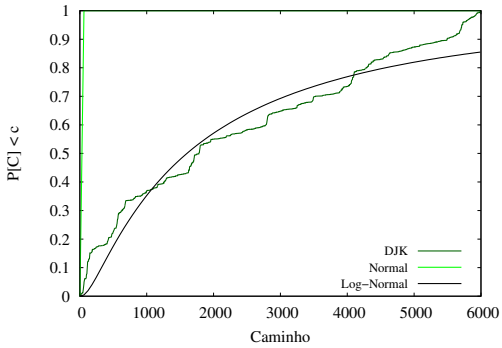
Figura B.3: CDF algoritmos – 16 h até 12 h –Distância.



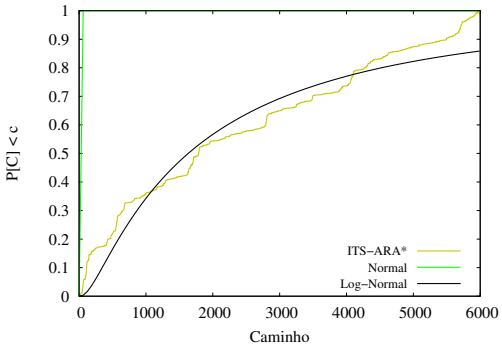
(a) CDF A*.



(b) CDF ARA*.

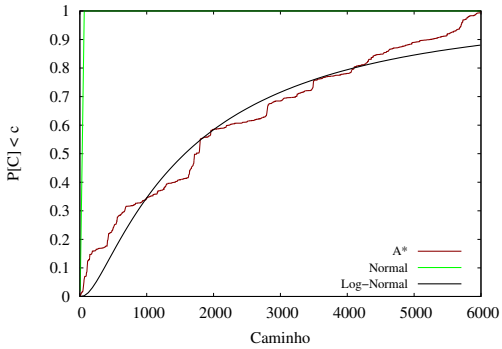


(c) CDF Dijkstra.

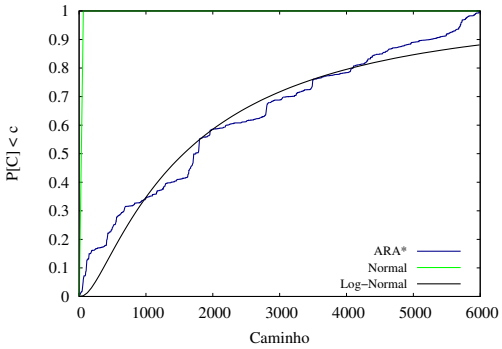


(d) CDF ITS-ARA*.

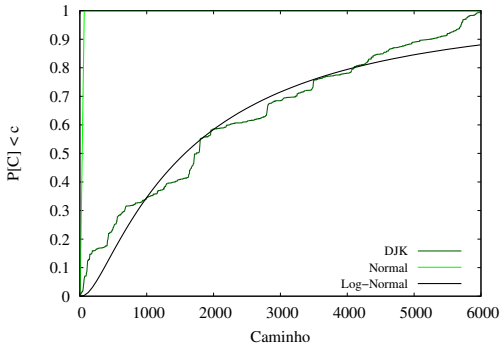
Figura B.4: CDF algoritmos – 07 h até 08 h – Tempo de Percurso.



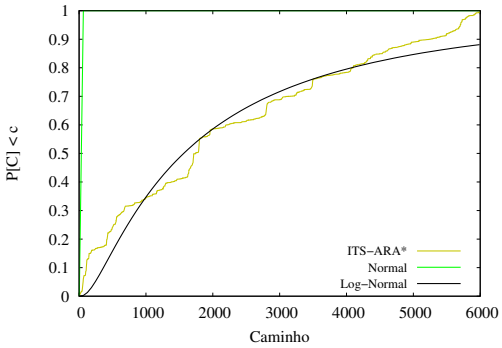
(a) CDF A*.



(b) CDF ARA*.

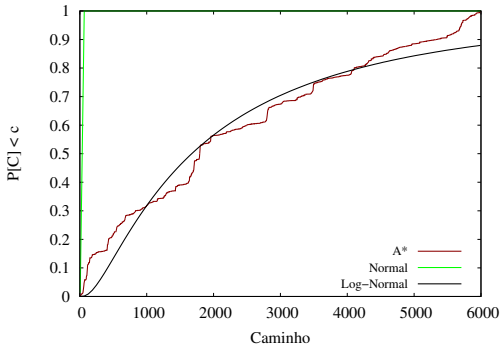


(c) CDF Dijkstra.

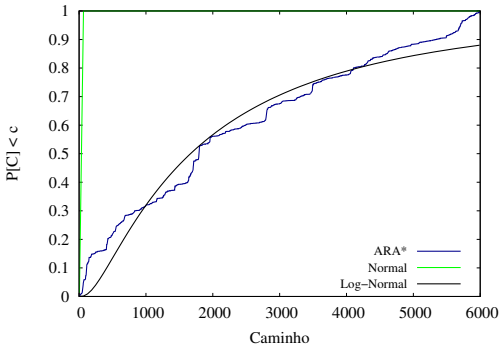


(d) CDF ITS-ARA*.

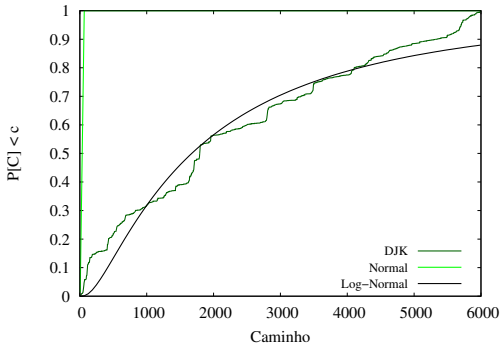
Figura B.5: CDF algoritmos – 11 h até 12 h – Tempo de Percurso.



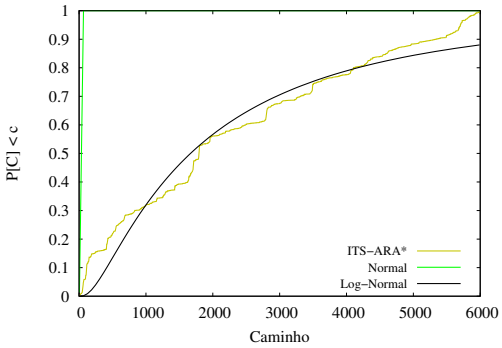
(a) CDF A*.



(b) CDF ARA*.



(c) CDF Dijkstra.



(d) CDF ITS-ARA*.

Figura B.6: CDF algoritmos – 16 h até 12 h – Tempo de Percurso.

Referências Bibliográficas

[1] FURDA, A., VLACIC, L. "Enabling Safe Autonomous Driving in Real-World City Traffic Using Multiple Criteria Decision Making", *IEEE Intelligent Transportation Systems Magazine*, v. 3, n. 1, pp. 4–17, Spring 2011.

[2] RIBEIRO JUNIOR, J., MITRE CAMPISTA, M., COSTA, L. "COTraMS: A Collaborative and Opportunistic Traffic Monitoring System", *Intelligent Transportation Systems, IEEE Transactions*, v. 15, n. 3, pp. 949–958, 2014.

[3] FAEZ, K., KHANJARY, M. "UTOSPF with Waiting Times for Green Light consideration". In: *Systems, Man and Cybernetics*, pp. 4170–4174.

[4] GALLARDO, D. *Developing Eclipse Plug-ins: How to Create, Debug, and Install your Plug-in*. Relatório técnico, S.1, 2002.

[5] TOMTOM. *TomTom Traffic Index - 2013 annual report*. TomTom GPS, jun. 2013. Disponível em: <`http://www.tomtom.com/pt_br/trafficindex/`>.

[6] SUGIYAMA, Y., FUKUI, M., KIKUCHI, M., et al. "Traffic jams without bottleneck - experimental evidence for the physical mechanism of the formation of a jam", *Springer New Journal of Physics*, v. 10, n. 3, pp. 033001, mar. 2008.

[7] IFTEKHAR, L., OLFATI-SABER, R. "Autonomous Driving for Vehicular Networks with Nonlinear Dynamics". In: *IEEE Intelligent Vehicles Symposium (IV)*, pp. 723–729, jun. 2012.

[8] CHIU, Y.-C., BOTTOM, J., MAHUT, M., et al. *Dynamic Traffic Assignment: A Primer*. 1 ed. , Transportation Research Board, 2011.

[9] WU, B.-F., HUANG, H.-Y., CHEN, C.-J., et al. "A vision-based blind spot warning system for daytime and nighttime driver assistance", *Elsevier Computers & Electrical Engineering*, v. 39, n. 3, pp. 846–862, abr. 2013.

[10] ANGIUS, F., REINERI, M., CHIASSERINI, C., et al. "Towards a realistic optimization of urban traffic flows". In: *IEEE Intelligent Transportation Systems*, pp. 1661–1668, 2012.

[11] RUSSELL, S. J., NORVIG, P. *Dynamic Traffic Assignment: A Primer*. 2ª ed. New York, USA, Pearson Education, 2003.

[12] GOODCHILD, M. F. "Citizens as sensors: the world of volunteered geography", *GeoJournal*, v. 69, n. 4, pp. 211–221, jan. 2007.

[13] VIEIRA, J. C. P. *Informação Geogrática Voluntária de Suporte às operações de Bombeiros, INEM e Proteção Civil*. Ph.D. dissertation, Universidade do Minho, Escola de Engenharia, Portugal, 2011.

[14] ROSSIER, L. *Rendering Interactive Maps on Mobile Devices Using Graphics Hardware*. Ph.D. dissertation, Vienna University of Technology, Faculty of Informatics, Viena, Austria, 2012.

[15] NUNES, N. F. *Planeador colaborativo de deslocações de bicicleta em meio urbano*. Ph.D. dissertation, Instituto Superior Técnico, Engenharia Informática e de Computadores, Lisboa, Portugal, 2013.

[16] MCNALLY, O. *Multi-Agent Pathfinding over Real-World Data Using CUDA*. Ph.D. dissertation, University of Dublin, Trinity College, Dublin, Irlanda, 2010.

[17] FIORE, A. G. *A Robust Motion Planning Approach for Autonomous Driving in Urban Areas*. Tese de D.Sc., Department of Aeronautics and Astronautics – Massachusetts Institute of Technology (MIT), Massachusetts, Estados Unidos, 2008.

[18] NEWMAN, P., SIBLEY, G., SMITH, M., et al. "Navigating, Recognizing and Describing Urban Spaces With Vision and Lasers", *The International Journal of Robotics Research*, v. 28, n. 11–12, pp. 1406–1433.

[19] MOHANTA, J. C., PARHI, D. R., PATEL, S. K. "Path planning strategy for autonomous mobile robot navigation using Petri-GA optimisation", *Elsevier Computers & Electrical Engineering*, v. 37, n. 6, pp. 1058–1070, nov. 2011.

[20] LIKHACHEV, M., GORDON, G., THRUN, S. "ARA*: Anytime A* with Provable Bounds on Sub-Optimality". In: *In Advances in Neural Information Processing Systems 16: Proceedings of The 2003 Conference (NIPS-03*. MIT Press, 2004.

[21] LIKHACHEV, M. *Search-based Planning for Large Dynamic Environments*. Tese de Doutorado, School of Computer Science – Carnegie Mellon University, 2005.

[22] LEQUERICA, I., GARCÍA LONGARON, M., RUIZ, P. "Drive and share: efficient provisioning of social networks in vehicular scenarios", *Communications Magazine, IEEE*, v. 48, n. 11, pp. 90–97, November 2010. ISSN: 0163-6804. doi: 10.1109/MCOM.2010.5621973.

[23] EXAME. "Google inicia integração do Waze ao Maps". 2013. http://exame.abril.com.br/tecnologia/noticias/google-inicia-integracao-do-waze-ao-mapshttp://exame.abril.com.br/tecnologia/noticias/google-inicia-integracao-do-waze-ao-maps.

[24] TERRA. "Maps inclui dados do Waze; atualização já chegou ao país". 2013. http://tecnologia.terra.com.br/hardware-e-software/maps-inclui-dados-do-waze-atualizacao-ja-chegou-ao-pais,1e13fb56c7690410VgnCLD2000000dc6eb0aRCRD.html.

[25] KIM, W., GERLA, M. "NAVOPT: Navigator Assisted Vehicular Route OPTimizer". In: *IEEE Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 450–455.

[26] FERGUSON, D., LIKHACHEV, M., STENTZ, A. "A Guide to Heuristicbased Path Planning". .

[27] OPENSTREETMAP. "OpenStreetMap Project". 2013. http://www.openstreetmap.org.

[28] GRAPHSTREAM. "GraphStream Project". 2015. http://graphstream-project.org/.

[29] SANDEEP, V., GOPAL, K., NAVEEN, S., et al. "Globally accessible machine automation using Raspberry pi based on Internet of Things". In: *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, pp. 1144–1147, Aug 2015. doi: 10.1109/ICACCI.2015.7275764.

[30] JANARD, K., MARURNGSITH, W. "Accelerating real-time face detection on a raspberry pi telepresence robot". In: *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on*, pp. 136–141, May 2015. doi: 10.1109/INTECH.2015.7173482.

[31] LAVALLE, S. M. *Planning Algorithms*. New York, NY, USA, Cambridge University Press, 2006. ISBN: 0521862051.

[32] SCHULTES, D. *Route Planning in Road Networks*. Tese de Doutorado, Karlsruhe University, 2008.

[33] DELLING, D., SANDERS, P., SCHULTES, D., et al. "Algorithmics of Large and Complex Networks". Springer-Verlag, cap. Engineering Route Planning Algorithms, pp. 117–139, Berlin, Heidelberg, 2009. ISBN: 978-3-642-02093-3. doi: 10.1007/978-3-642-02094-0_7. Disponível em: <http://dx.doi.org/10.1007/978-3-642-02094-0_7>.

[34] HLINĚNÝ, P., MORIŠ, O. "Scope-based Route Planning". In: *Proceedings of the 19th European Conference on Algorithms*, ESA'11, pp. 445–456, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN: 978-3-642-23718-8. Disponível em: <http://dl.acm.org/citation.cfm?id=2040572.2040622>.

[35] VAN DEN BERG, J., ABBEEL, P., GOLDBERG, K. "LQG-MP: Optimized Path Planning for Robots with Motion Uncertainty and Imperfect State Information", *Int. J. Rob. Res.*, v. 30, n. 7, pp. 895–913, jun. 2011. ISSN: 0278-3649. doi: 10.1177/0278364911406562. Disponível em: <http://dx.doi.org/10.1177/0278364911406562>.

[36] MCNAUGHTON, M. *Parallel Algorithms for Real-time Motion Planning*. Tese de Doutorado, Robotics Institute – Carnegie Mellon University, 2011.

[37] BRAGA, M. L., SANTOS, A. J., PEDROZA, A. C. P., et al. "Planejamento de Rotas com Algoritmos Anytime em Redes Veiculares na Plataforma Raspberry Pi". In: *XXXI Simpósio Brasileiro de Engenharia de Sistemas Computacionais - SBESC'2014*, SBESC '2014, pp. 1–6, 2014.

[38] HULT, R. *Path Planning for Highly Automated Vehicles*. Tese de Mestrado, Chalmers University of Technology, Gotemburgo, Suécia, 2013.

[39] ROMERO, R. A. F., PRESTES, E., OSÓRIO, F., et al. *Robótica Móvel*. 1ª ed. Rio de Janeiro, Brasil, LTC, 2003.

[40] NEJAD, H., DO, Q., SAKAI, R., et al. "Real time localization, path planning and motion control for autonomous parking in cluttered environment with narrow passages". In: *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, pp. 1357–1364, Sept 2012. doi: 10.1109/ITSC.2012.6338723.

[41] PRADO, M. G., MAGALHÃES, A. C., WOLF, D. F., et al. "Detecção de Vagas e Estacionamento Autônomo de Veículos". In: *Simpósio Brasileiro de Automação Inteligente - SBAI 2013*, pp. 1–6, 2013.

[42] CHEN, C., RICKERT, M., KNOLL, A. "Path planning with orientation-aware space exploration guided heuristic search for autonomous parking and maneuvering". In: *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pp. 1148–1153, June 2015. doi: 10.1109/IVS.2015.7225838.

[43] THRUN, S., MONTEMERLO, M., DAHLKAMP, H., et al. "Stanley: The Robot That Won the DARPA Grand Challenge: Research Articles", *J. Robot. Syst.*, v. 23, n. 9, pp. 661–692, set. 2006. ISSN: 0741-2223. doi: 10.1002/rob.v23:9. Disponível em: <http://dx.doi.org/10.1002/rob.v23:9>.

[44] "DARPA Urban Challenge – Measuring Congestion Worldwide - Accessed in November/2015". http://archive.darpa.mil/grandchallenge/, 2015.

[45] FERNANDES, L. C., SOUZA, J. R., PESSIN, G., et al. "CaRINA Intelligent Robotic Car: Architectural design and applications", *Journal of Systems Architecture*, v. 60, n. 4, pp. 372 – 392, 2014.

[46] DA COSTA, P. P. *Teoria de Grafos e suas Aplicações*. Tese de Mestrado, Universidade Estadual Paulista Júlio de Mesquita Filho, Rio Claro, São Paulo, 2011. Instituto de Geociências e Ciências Exatas.

[47] GEISBERGER, R., VETTER, C. "Efficient Routing in Road Networks with Turn Costs". In: Pardalos, P., Rebennack, S. (Eds.), *Experimental Algorithms*, v. 6630, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 100–111, 2011. ISBN: 978-3-642-20661-0. doi: 10.1007/978-3-642-20662-7_9. Disponível em: <http://dx.doi.org/10.1007/978-3-642-20662-7_9>.

[48] DELLING, D., GOLDBERG, A. V., PAJOR, T., et al. "Customizable Route Planning in Road Networks", *Transportation Science*, 2015. doi: 10.1287/trsc.2014.0579. Disponível em: <http://dx.doi.org/10.1287/trsc.2014.0579>.

[49] DIBBELT, J., PAJOR, T., WAGNER, D. "User-Constrained Multimodal Route Planning", *Journal of Experimental Algorithmics*, v. 19, pp. 3.2:1.1–3.2:1.19, abr. 2015. ISSN: 1084-6654. doi: 10.1145/2699886. Disponível em: <http://doi.acm.org/10.1145/2699886>.

[50] HART, P., NILSSON, N., RAPHAEL, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *Systems Science and Cybernetics, IEEE Transactions on*, v. 4, n. 2, pp. 100–107, 1968. ISSN: 0536-1567. doi: 10.1109/TSSC.1968.300136.

[51] ZILBERSTEIN, S. "Using Anytime Algorithms in Intelligent Systems", *AI Magazine*, v. 17, n. 3, pp. 73–83, 1996.

[52] ROJAS, A., DE A. RITO, A. C., RIBEIRO, P. C. M. *Desenvolvimento de Sistemas Complexos aplicáveis aos Sistemas Inteligentes de Transportes (ITS) - O caso SINIAV*. Relatório técnico, Cadernos do IME - Série Informática, jun. 2009.

[53] ROJAS, A. *Geração Dinâmica de Padrões de Movimentação Rodoviária de Cargas e Veículos Baseada em Identificação por Radiofrequência*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, 2012.

[54] LEE, U., MAGISTRETTI, E., GERLA, M., et al. "Bio-Inspired Multi-Agent Collaboration for Urban Monitoring Applications", *Bio-Inspired Computing and Communication*, v. 5151, n. 3, pp. 204–216, 2008.

[55] LI, C., ANAVATTI, S., RAY, T. "Adaptive route guidance system with real-time traffic information". In: *IEEE Intelligent Transportation Systems*, pp. 367–372, 2012.

[56] BOUKERCHE, A., OLIVEIRA, H. A. B. F., NAKAMURA, E. F., et al. "Vehicular Ad Hoc Networks: A New Challenge for Localization-Based Systems", *Comput. Commun.*, v. 31, n. 12, pp. 2838–2849, jul. 2008. ISSN: 0140-3664. doi: 10.1016/j.comcom.2007.12.004. Disponível em: <http://dx.doi.org/10.1016/j.comcom.2007.12.004>.

[57] LI, B., GONG, J., JIANG, Y., et al. "ARA*+: Improved Path Planning Algorithm Based on ARA*". In: *IEEE Web Intelligence and Intelligent Agent Technology*, pp. 361–365, dez. 2012.

[58] MOURA, L., RITT, M., BURIOL, L. S. "Estudo Experimental de Algoritmos em Tempo Real de Caminho Mínimo Ponto a Ponto em Grafos Dinâmicos". In: *Anais do XLII Simpósio Brasileiro de Pesquisa Operacional*, SBPO, 2010.

[59] RIBEIRO JÚNIOR, J. G., M., C., M., M. E., et al. "A Decentralized Traffic Monitoring System Based on Vehicle-to-Infrastructure Communications". In: *Wireless Days (WD), 2013 IFIP*, pp. 1–6, 2013.

[60] RAHMAN, K., ALAM, T., CHOWDHURY, M. "Location based early disaster warning and evacuation system on mobile phones using OpenStreetMap". In: *Open Systems (ICOS), 2012 IEEE Conference on*, pp. 1–6, Oct 2012. doi: 10.1109/ICOS.2012.6417627.

[61] ZHAN, A., CHANG, M., CHEN, Y., et al. "Accurate Caloric Expenditure of Bicyclists Using Cellphones". In: *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, pp. 71–84, New York, NY, USA, 2012. ACM. ISBN: 978-1-4503-1169-4. doi: 10.1145/2426656.2426664. Disponível em: <`http://doi.acm.org/10.1145/2426656.2426664`>.

[62] RAHMIG, C., KLUGE, A. "Digital maps for railway applications based on OpenStreetMap data". In: *Intelligent Transportation Systems - (ITSC), 2013 16th International IEEE Conference on*, pp. 1322–1327, Oct 2013. doi: 10.1109/ITSC.2013.6728414.

[63] DAS, R., PUROHIT, P., ALAM, T., et al. "Location based ATM locator system using OpenStreetMap". In: *Software, Knowledge, Information Management and Applications (SKIMA), 2014 8th International Conference on*, pp. 1–6, Dec 2014. doi: 10.1109/SKIMA.2014.7083518.

[64] IMTEAJ, A., CHOWDHURY, M., MAHAMUD, M. "SmartTravel: An approach to redolant Transportation Guiding Application in context of Bangladesh using OpenStreetMap". In: *Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on*, pp. 1–5, May 2015. doi: 10.1109/ICEEICT.2015.7307405.

[65] RICH, C., PONSLER, B., HOLROYD, A., et al. "Recognizing Engagement in Human-Robot Interaction". In: *Human Robot-Interaction*, 2010.

[66] COHEN, B., CHITTA, S., LIKHACHEV, M. "Search-Based Planning for Manipulation with Motion Primitives". In: *International Conference on Robotics and Automation*, 2010.

[67] KALAKRISHNAN, M., CHITTA, S., THEODOROU, E., et al. "STOMP: Stochastic trajectory optimization for motion planning". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 4569–4574, May 2011.

[68] FOOTE, T. "tf: The transform library". In: *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-

Source Software workshop, pp. 1–6, April 2013. doi: 10.1109/TePRA.2013.6556373.

[69] PIGNÉ, Y., DUTOT, A., GUINAND, F. "GraphStream: A Tool for bridging the gap between Complex Systems and Dynamic Graphs", *CoRR*, v. abs/0803.2093, 2008.

[70] ANISZCZYK, C., GALLARDO, D. "Get started with the Eclipse Platform". jul. 2007.

[71] FERREIRA, M. A. R. *Desenvolvimento de um Plug-in Eclipse para Modelagens de Aplicações Geoespaciais*. Tese de Mestrado, Universidade do Vale do Itajaí, 2009.

[72] BRAGA, M. L. *Geração Automática de Código em Redes de Sensores Sem Fio usando Communicating X-Machine*. Tese de Mestrado, Universidade Federal do Amazonas (UFAM), Manaus, Amazonas, jan. 2012. Programa de Pós-Graduação em Engenharia Elétrica –Faculdade de Tecnologia.

[73] HAKLAY, M., WEBER, P. "OpenStreetMap: User-Generated Street Maps", *IEEE Pervasive Computing*, v. 7, n. 4, pp. 12–18, out. 2008.

[74] HAYAKAWA, T., IMI, Y., ITO, T. "Analysis of Quality of Data in OpenStreetMap". In: *IEEE Commerce and Enterprise Computing CEC*, pp. 131–134, 2012.

[75] VUJOVIC, V., MAKSIMOVIC, M. "Raspberry Pi as a Sensor Web node for home automation", *Elsevier Computers & Electrical Engineering*, v. 44, pp. 153–171, maio 2015.

[76] POZZER, C. "Teoria dos Grafos (Notas de Aula)". `http://www-usr.inf.ufsm.br/~pozzer/disciplinas/ed_7_grafos.pdf`, jun. 2010. [Acessado em 03 de Março 2015].

[77] RIBEIRO JÚNIOR, J. G., MITRE CAMPISTA, M., COSTA, L. "COTraMS: A Collaborative and Opportunistic Traffic Monitoring System", *IEEE Transactions on Intelligent Transportation Systems*, v. 15, n. 3, pp. 949–958, 2014.

[78] QUIGLEY, M., CONLEY, K., GERKEY, B., et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*, 2009.

[79] TABOADA, G. L., RAMOS, S., EXPÓSITO, R. R., et al. "Java in the High Performance Computing arena: Research, practice and experience", *Science of Computer Programming*, v. 78, n. 5, pp. 425–444, 2013. ISSN: 0167-6423. Special section: Principles and Practice of Programming in Java 2009/2010; Special section: Self-Organizing Coordination.

[80] GAREA, S. R. *High Performance Java for Multi-core Systems*. Tese de Doutorado, Department of Electronics and Systems – University of A Coruña, 2013.