
Previsão Eficiente do Posicionamento Futuro de Nós em Redes Móveis

Enrique Fynn



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Uberlândia
2015

Enrique Fynn

**Previsão Eficiente do Posicionamento Futuro
de Nós em Redes Móveis**

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Lásaro Jonas Camargos

Coorientador: Rafael Pasquini

Uberlândia

2015

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

F997p Fynn, Enrique, 1989-
2015 Previsão eficiente do posicionamento futuro de nós em redes móveis
/ Enrique Fynn. - 2015.
89 f. : il.

Orientador: Lásaro Jonas Camargos.
Coorientador: Rafael Pasquini.
Dissertação (mestrado) - Universidade Federal de Uberlândia,
Programa de Pós-Graduação em Ciência da Computação.
Inclui bibliografia.

1. Ciência da computação - Teses. 2. Redes remotas (Redes de computadores) - Previsão - Teses. 3. Movimento - Previsão - Teses. 4. Algoritmos de computador - Teses. I. Camargos, Lásaro Jonas. II. Pasquini, Rafael. III. Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Este trabalho é dedicado a minha avó Áurea.

Agradecimentos

A todos que ajudaram direta ou indiretamente na conclusão deste trabalho.

“Se a sua alma não se aperfeiçoa, se seus juízos não se tornam mais lúcidos, melhor fora que o estudante gastasse o tempo a jogar péla, pois ao menos o corpo ele o teria mais ágil. Observai-o de volta após quinze ou dezesseis anos: nada se fará dele; o que trouxe a mais é o grego e o latim, que o fizeram mais tolo e presunçoso do que quando deixou a casa paterna. Devia voltar com o espírito cheio e voltou balofo; incharam-no e continuou vazio.”

(Michel de Montaigne)

Resumo

A habilidade de prever onde nós podem estar em um futuro próximo, pode possibilitar novas aplicações em Redes Ad-Hoc Móveis (MANET). Por exemplo, o conteúdo pode ser gerado para um consumidor em potencial em cenários de computação pervasiva ou congestionamentos de tráfego podem ser previstos e prevenidos. Neste trabalho introduzimos dois algoritmos para previsão de posição futura de nós em uma rede móvel, PheroCast e ToD-Pherocast. O algoritmo de Previsão Baseada em Feromônios (PheroCast) é um algoritmo leve para realizar previsões *online* da posição futura de um nó baseado em seu histórico de movimentação. PheroCast, no entanto, não leva em consideração variações no padrão de movimentação ao longo do dia, usando o histórico do passado da mesma forma. Por exemplo, em um cenário em que o mesmo nó viaja todos os dias de manhã, mas raramente à tarde, PheroCast dará o mesmo ou mais peso para o dado da manhã enquanto estiver prevendo a viagem à tarde, o que provavelmente levaria a uma previsão errada. Devido a tal limitação, desenvolvemos o *Time of Day* Pherocast, ou ToD-Pherocast, uma versão estendida do algoritmo original que leva em consideração o horário da viagem para gerar as previsões, dando mais ênfase à história do movimento em horários similares. Finalmente, apresentamos uma avaliação de desempenho considerando três cenários: (i) previsão da posição de ônibus, cujo comportamento esperado é regular; (ii) previsão de posição de táxis, que aparentemente levaria a baixa taxa de acertos; e (iii) mobilidade de pessoas em relação redes sem fio, que usou rastros coletados pelo grupo de pesquisa do autor. Nossas avaliações mostram que o ToD-PheroCast é até 4.41% melhor que o PheroCast no cenário dos ônibus, em que alcançou acurácia de mais de 85%, e 0.72% melhor no cenário dos táxis, alcançando uma acurácia de até 89.17%. Finalmente, no cenários de previsão de redes sem fio, ToD-PheroCast atingiu 81.02% de acurácia. Esses resultados mostram não só que a previsão da posição é possível nos cenários, mas que pode ser realizada rápida e acuradamente.

Palavras-chave: MANET, VANET, Previsão de Posicionamento Futuro.

Abstract

The ability to predict where nodes might be in the near future may enable several new applications in a mobile ad hoc network (MANET). For example, content may be generated for an approaching potential consumer in pervasive computing scenarios or traffic jams may be predicted and prevented. We introduce PheroCast, a lightweight algorithm to do online predictions of a node's future position based on its previous movement history. PheroCast, however, does not take into account the variations in the movement pattern along the day, using any previous history in the same way. For example, in a scenario where the same node travels every morning, but seldom in the evening, PheroCast would give the same or more weight to the data from the morning when predicting an evening trip, which would likely lead to a wrong prediction. Due to this limitation, we developed the *Time of Day* PheroCast, or ToD-PheroCast, an extended version of the original algorithm which takes the time of the day into account while making predictions, giving more emphasis to the history of movement within similar time windows. Finally, we evaluate the performance in three scenarios: (i) prediction of the position of buses in a metropolis, which are expected to have very regular mobility pattern; (ii) Taxis in a metropolis, which should lead to low accuracy predictions; and (iii) mobility of people interacting with wireless networks, that used traces collected by the author's research group. Our evaluations show that ToD-PheroCast is up to 4.41% better than PheroCast in the bus scenario, in which it achieved over 85% accuracy in its predictions, and 0.72% better in the taxi scenario, in which the algorithm achieved up to 89.17% accuracy. Finally, in the wireless scenario, ToD-PheroCast achieved 81.02% accuracy. These results show that not only forecasting is possible in such scenarios, but that it may be done with high accuracy, online, and in a lightweight manner.

Keywords: MANET, VANET, Future Movement ForeCast.

Lista de ilustrações

Figura 1 – Exemplo de melhoria de qualidade de serviço, sabendo-se a rota de veículos.	24
Figura 2 – Exemplo de comunicação por feromônios em formigas.	31
Figura 3 – Exemplo da trajetória de um nó. Os pontos maiores numerados identificam marcos, e as trajetórias com os relativos símbolos correspondentes na Figura 4.	41
Figura 4 – Exemplo de um <i>log</i> de uma viagem. Para cada viagem, a primeira coluna tem o <i>local</i> onde o nó estava e na segunda coluna a direção que o nó estava seguindo quando a entrada foi adicionada no <i>log</i>	41
Figura 5 – Exemplo de <i>trilha</i> . A localização dos <i>locais</i> foi registrada a cada unidade de tempo. Cada vértice tem uma tupla (id, p) com o identificador do nó e seu peso.	42
Figura 6 – Exemplo de Phero Mapa da Viagem 1, ilustrada na Figura 4. em todos os nós $a = 1, t = 1$, exceto no nó $12E$, onde $a = 1, t = 3$	44
Figura 7 – Pequeno conjunto de viagens de um nó, cada hexágono corresponde a um <i>marco</i> : a casa, a escola e o trabalho	46
Figura 8 – Um excerto do traço do ônibus 1008 seguindo a rota 007. Colunas são data, hora, id, rota, desconhecido e distâncias X e Y de um ponto de referência, em pés (COELHO et al., 2014).	52
Figura 9 – Uma abstração do Phero Mapa da rota 007, inscrito em cada vértice, mostra o tempo de espera médio em segundos.	52
Figura 10 – Diferentes formatos de uma célula	53
Figura 11 – Excerto do log de um ônibus processado.	53
Figura 12 – As entradas mostram um padrão claro de movimentação, como esperado de uma linha de ônibus. Nos eixos x e y, estão representadas respectivamente a longitude e latitude.	54

Figura 13 – Erros nas coordenadas x/y da rota 007 com parâmetros $k = 3$, $g = 600$, $m = 2$ e $\sigma = 0.4$. O erro é mensurado como a diferença, em número de células, entre as previsões e os dados reais.	56
Figura 14 – As entradas mostram os detalhes da cidade de Roma, um padrão diverso, como esperado de um taxi. Nos eixos x e y, estão representadas a longitude e latitude, respectivamente.	58
Figura 15 – Previsões para o táxi 8.	59
Figura 16 – Dados de GPS de um usuário na cidade de Uberlândia, os dados de longitude e latitude foram omitidos para preservar a anonimidade do usuário.	59
Figura 17 – Exemplo abstrato de um Phero Mapa utilizando Conjunto de Serviços Básicos de Identificação (BSSID) como identificadores.	60
Figura 18 – Função de Distribuição Acumulada (CDF) para erros na previsão com diversas configurações de g, k, m e σ	63
Figura 19 – Número de entradas corretas do taxi 8	64
Figura 20 – Número de entradas corretas para todos os taxis	65
Figura 21 – Possíveis locais de interesse de dois indivíduos distintos.	66
Figura 22 – Espaço utilizado pelo grafo, número de veículos e desvio padrão, variando através do tempo para os logs de taxis em Roma.	67
Figura 23 – Espaço utilizado pelo grafo, número de veículos e desvio padrão, variando através do tempo para os logs de ônibus de Seattle.	68
Figura 24 – Tempo levado para prever, número de carros e desvio padrão, variando conforme o tempo nos logs de taxis de Roma.	69
Figura 25 – Tempo levado para prever, número de carros e desvio padrão, variando conforme o tempo nos logs de ôñibus de Seattle.	69
Figura 26 – Arquitetura do aplicativo Android.	85
Figura 27 – Tela do aplicativo.	85

Lista de tabelas

Tabela 1	– Tabela de comparação de estudos de vários modelos de predição de mobilidade (GOHIL, 2014)	38
Tabela 2	– Previsões corretas e incorretas do ônibus 007, para diferentes valores de g, k, m e σ . O cenário $k = 3, g = 1000, m = 2$ e $\sigma = 0.4$ é usado para o patamar.	61
Tabela 3	– Porcentagem de acerto médio dos ônibus, variando o g .	62
Tabela 4	– Previsões corretas e incorretas do táxi 8 para diferentes valores de g, k, m e σ . O cenário $k = 3, g = 100, m = 5$ e $\sigma = 0.4$ é usado para o patamar.	63
Tabela 5	– Porcentagem de acerto médio dos táxis, variando o g .	64
Tabela 6	– Estratégias de desviar ou não de um obstáculo, o estado de Equilíbrio de Nash, onde os dois nós desviam, em negrito.	75

Lista de siglas

ACK *Confirmação*

ACO Otimização da Colônia de Formigas

Anatel Agência Nacional de Telecomunicações

API Interface de Programação de Aplicativos

ARIMA Modelo Auto-regressivo Integrado de Média Móvel

BSSID Conjunto de Serviços Básicos de Identificação

CDF Função de Distribuição Acumulada

DFS Busca em Profundidade

DTN Rede Tolerante a Atraso

EWMA Média Móvel Exponencialmente Ponderada

FGV Fundação Getúlio Vargas

GPS Sistema de Posicionamento Global

HTTP Protocolo de Transferência de Hipertexto

MANET Redes Ad-Hoc Móveis

MPLS Multiprotocolo de Comutação de Etiqueta

N Norte

NMDC Desafio de Dados Móveis da Nokia

OSI Sistema Aberto de Interconexão

PheroCast Previsão Baseada em Feromônios

POSIX Interface Portável entre Sistemas Operacionais

QoS Qualidade de Serviço

SDN Redes Definidas Por Software

SE Sudoeste

SMS Serviço de Mensagens Curtas

SSL Camada de Socket Seguro

SUMO Simulação de Mobilidade Urbana

ToD-PheroCast Previsão Baseada em Feromônios Considerando o Tempo

UFU Universidade Federal de Uberlândia

WDMM Modelo de Movimentação em Dias Úteis

VANET Redes Ad-Hoc Veiculares

Sumário

1	INTRODUÇÃO	21
2	TRABALHOS RELACIONADOS	29
2.1	Média Móvel Exponencialmente Ponderada	29
2.2	Colônia de Formigas	30
2.3	Modelos de Mobilidade	31
2.4	Previsão de posicionamento	33
3	PHEROCAST	39
3.1	Marcos e Locais	39
3.2	Phero Trails	40
3.3	Phero Mapas	41
3.4	Envelhecendo a Informação	43
3.5	Previsão de Posição	46
3.6	ToD-PheroCast	47
3.6.1	Phero Mapas para ToD-PheroCast	48
3.6.2	Previsão de Posição para ToD-PheroCast	48
3.7	Otimizações na Execução	48
4	MATERIAIS E MÉTODOS	51
4.1	Ônibus de Seattle	51
4.1.1	Características do Log	52
4.1.2	Parâmetros de Predição	53
4.1.3	Validade das Suposições	55
4.1.4	Erro	55
4.2	Táxis de Roma	57
4.2.1	Parâmetros de Predição	57
4.3	Cenário Wi-Fi de Uberlândia	57

5	AVALIAÇÃO	61
5.1	Ônibus de Seattle	61
5.2	Táxis de Roma	63
5.3	Cenário Wi-Fi de Uberlândia	65
5.4	Análise de Espaço	66
5.5	Análise do Tempo de Predição	67
6	CONCLUSÕES E TRABALHOS FUTUROS	71
6.1	Trabalhos Futuros	73
6.2	Aplicações	74
	REFERÊNCIAS	77

APÊNDICES 81

APÊNDICE A	– APLICAÇÃO PARA ANÁLISE REAL	83
A.1	Aplicação Android	83
A.2	Servidor na Nuvem	86

Introdução

Dispositivos móveis, como *notebooks*, *smartphones* e sistemas de painel veiculares tornaram-se difundidos em nossa sociedade moderna. Segundo a Agência Nacional de Telecomunicações (Anatel), “O Brasil encerrou junho de 2015 com 282,4 milhões de acessos em operação na telefonia móvel e a densidade de 138,23 acessos por 100 habitantes [...]” (Agência Nacional de Telecomunicações, 2015). Segundo uma pesquisa realizada pela Fundação Getúlio Vargas (FGV), dos telefones celulares, 95% são *smartphones*, i.e., hoje existem mais *smartphones* do que pessoas no Brasil (MEIRELLES, 2015).

O verdadeiro poder dessa mobilidade, no entanto, só pode ser alcançado uma vez que os indivíduos começam a interagir uns com outros e com o ambiente em sua volta, formando uma verdadeira rede móvel *peer-to-peer*. Nesse cenário, os nós podem se mover interagindo com a infraestrutura a sua volta, se disponível, permitindo diversos serviços. Por exemplo, os nós podem confiar uns nos outros para alcançar serviços localizados remotamente, propagar alertas sobre as condições da via, informar sobre um veículo de mais alta prioridade, como uma ambulância, ou para formar redes oportunistas que permitam a comunicação entre nós que nunca estão diretamente conectados. Este é o cenário das Redes Ad-Hoc Móveis (MANET).

Uma MANET é uma rede *ad hoc* autônoma, autorreconfigurável, que consiste de nós independentes que mudam dinamicamente sua conectividade (ROY, 2011). Devido à dinamicidade dos nós, a rede tem natureza inerente *peer-to-peer*, embora possa contar também com uma infraestrutura física. Com o número crescente de aplicativos móveis, as MANET têm se tornado cada vez mais importantes para atender aos requisitos dos usuários de hoje.

Diferentes tipos de nós têm distintos padrões de mobilidade, o que torna difícil estabelecer um mecanismo de roteamento e até mesmo de acesso adequado a todos os cenários. Por isso, apesar do esforço da IEEE (JIANG; DELGROSSI, 2008b), ainda existem poucos protocolos padrão para MANET. A infraestrutura descentralizada e a liberdade dos nós de saírem ou se juntarem à rede a qualquer momento tornam os algoritmos em MANET difíceis de serem construídos. Por exemplo, se selecionarmos um servidor em uma MANET,

mesmo que por consenso, é necessário um mecanismo de previsão do posicionamento, para prevenir que o nó preste a sair da rede seja selecionado como servidor.

As Redes Ad-Hoc Veiculares (VANET) são MANET em que os nós são veículos. Os mesmos conceitos da MANET se aplicam à VANET, contudo, veículos se movimentam mais rápido que pessoas, o que altera a dinâmica da rede e a forma de como resolver alguns dos seus problemas. Neste trabalho, não discutiremos o relacionamento entre nós em uma MANET ou VANET e, sim, um método que é executado somente por um nó, para inferir propriedades sobre sua movimentação. Tal método pode ser estendido para agir cooperativamente. Gerenciar a comunicação entre nós é trabalho de uma aplicação, por exemplo, de roteamento.

Devido à alta mobilidade dos nós em uma VANET e falta de uma estrutura fixa, é desafiador construir uma arquitetura para fazer funcionar serviços dentro de uma MANET. A arquitetura clássica de cliente-servidor, como mencionada anteriormente, não funciona para a maioria dos casos, já que os nós se movem livremente e podem não estar ao alcance do servidor em um dado momento.

Aplicativos, como o *Waze*¹, podem se beneficiar com o uso de algoritmos associados a uma VANET e predição de movimentos dos nós na rede. O *Waze* confia em um serviço centralizado, atualmente, controlado pelo *Google*. Esse programa, e outros que atuam de forma descentralizada, se beneficiariam com uma VANET que usa um algoritmos de previsão *online*, não invasivos e rápidos.

Há diversas aplicações de MANET e, mais especificamente, de VANET. Por exemplo, semáforos podem se tornar inteligentes e controlar o fluxo de carros, de forma a minimizar o tempo de espera, recebendo informações sobre o tráfego dos próprios veículos. Outro exemplo, se diversos carros param bruscamente ou se um deles percebe que colidiu, sinais de emergência podem ser enviados via a VANET para alertar outros motoristas e solicitar auxílio, como proposto por (SURIYAPAIBONWATTANA; POMAVALAI, 2008). Finalmente, as VANET também possibilitam um futuro no qual veículos se comuniquem uns com os outros para que, sem nenhuma ou pouca interferência de humanos e com confiabilidade maior do que a atual, possam se deslocar autonomamente (MOLINA-GIL; CABALLERO-GIL; CABALLERO-GIL, 2014). Além de aplicações de segurança como as mencionadas, VANET permitem o desenvolvimento de aplicações de conforto, como comunicação entre usuários da rede, acesso à Internet e a serviços, como *torrent* (LEE et al., 2006).

Existe um grande potencial para aplicações descentralizadas que funcionam em VANET. Aplicativos comerciais de carona, como o *Uber*, que permite aos usuários escolherem diferentes formas de transporte entre outros usuários do aplicativo; *streaming* de vídeos ou músicas, utilizando tecnologias como implementadas pelo protocolo de *torrent*, como

¹ o atual maior serviço de tráfego que usa informações comunitárias, em que os nós reportam possíveis problemas na pista, editam mapas e, inclusive, conversam com outros nós próximos

o *PopCorn Time* ou *NetFlix*. Aplicações para alertas de trânsito automático, entre outros, podem ser implementadas e otimizadas com algoritmos eficientes de previsão de posicionamento, para garantir, principalmente, alta disponibilidade. Mais aplicações são discutidas na Seção 6.2 deste trabalho.

Para possibilitar serviços, enquanto os nós se movem, a rede tem de se adaptar à nova localização geográfica de cada nó, uma tarefa que impõe muitos desafios. Por exemplo, em um cenário de cliente-servidor, se uma requisição é emitida, a resposta tem que chegar no solicitante mesmo que esse tenha se movido para um lugar diferente. Apesar de existirem diversos protocolos de rede que permitem que nós se movam e ainda continuem conectados, esses protocolos, como o *IEEE 802.11p*, implementado em (TANIUCHI et al., 2009), são meramente reativos i.e., esperam o evento acontecer para reagir e não fornecem a melhor qualidade possível do serviço. Se a posição futura de um nó puder ser estimada com precisão, as trocas de contexto (*handoffs*) podem ser cuidadosamente planejadas, para que a transmissão aconteça sem problemas e garantir, assim, uma melhor qualidade de serviço. Além disso, mesmo se o nó é sempre alcançável e.g. através de uma rede 4G, se os dados puderem ser redirecionados para posições futuras, onde mais opções de conexão estão disponíveis, o dado pode, então, ser entregue por um custo mais baixo.

Para que as aplicações mencionadas se materializem, deve existir uma infraestrutura confiável para a troca de mensagens entre veículos, a despeito de mudanças na topologia da rede. Por exemplo, se o veículo A está se comunicando com o B via um veículo intermediário C, mas C realiza uma mudança de direção, deixando A e B desconectados, uma nova rota entre A e B deve ser estabelecida. Já existem protocolos reativos para realizar essa troca de contexto, como o *IEEE 802.11p* (TANIUCHI et al., 2009).

No exemplo apresentado, que aparece na Figura 1, se a rede pudesse prever a mudança de direção de C e que o veículo D estivesse provavelmente destinado a seguir uma rota paralela às de A e B, então poderia proativamente mover a comunicação para usar D em vez de C. Se tal a reconfiguração da rede, baseada nas previsões de mudanças de rotas, for bem sucedida, a comunicação entre A e B não será interrompida e a reconfiguração passará despercebida pelos usuários. Denominamos este tipo de protocolo, que tenta prever onde os nós estarão e se prepara de antemão à mudança de topologia para minimizar latência e interrupções na troca de mensagens, de *roteamento proativo*.

A mobilidade humana é passível de previsão, uma vez que é previsível em diversas escalas, como mostra (BROCKMANN; HUFNAGEL; GEISEL, 2006). Alguns trabalhos têm explorado essa característica em cenários de rede móvel. Por exemplo, (DOMENICO; LIMA; MUSOLESI, 2012) explora como o histórico dos movimentos de amigos, colhidos através de seus *smartphones*, pode ser usado para melhorar a previsão dos seus movimentos. A técnica utilizada, no entanto, é cara, já que requer uma análise intensa de séries temporais e o conhecimento dos movimentos dos amigos. Acreditamos que esta abordagem exigiria um serviço centralizado. Em (SILVA et al., 2013), um exemplo mais focado

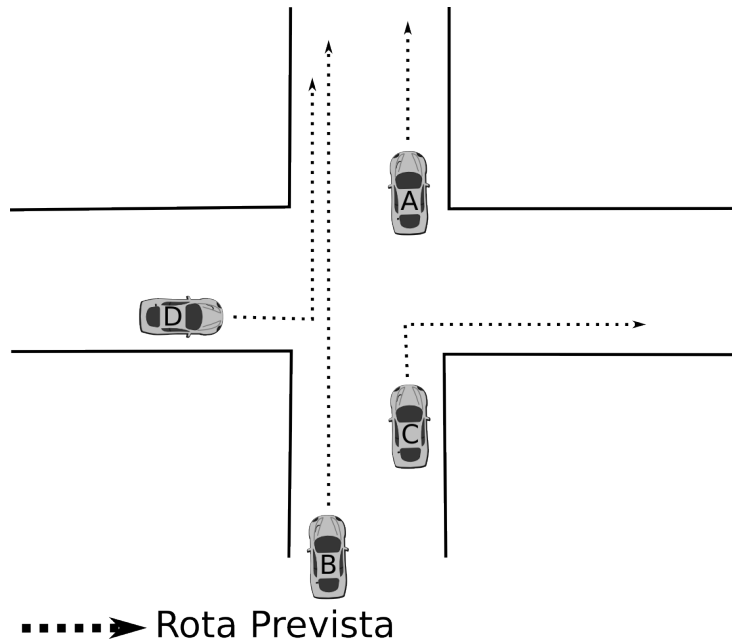


Figura 1 – Exemplo de melhoria de qualidade de serviço, sabendo-se a rota de veículos.

em redes veiculares, ônibus em uma rota bem conhecida se comunicam com elementos da infraestrutura para informar sua posição a um serviço central para que esse possa determinar para onde encaminhar mensagens aos ônibus. Os autores propõem o envio de mensagens replicadas para a última posição conhecida do ônibus, mais os três seguintes pontos de acesso na rota, i.e. nenhuma previsão está sendo realizada.

A centralização vai contra os princípios de uma rede *peer-to-peer*, na qual cada nó é servidor e/ou cliente ao mesmo tempo. Além disso, uma boa predição deve ser *online*, i.e. dados são processados à medida que são coletados; rapidamente e não invasivamente, já que aplicações de natureza anônima podem ser executadas em uma MANET, isso já se mostrou como uma preocupação das especificações para MANET. Mesmo em camadas de baixo nível do modelo Sistema Aberto de Interconexão (OSI), por exemplo, o protocolo IEEE 802.11p (JIANG; DELGROSSI, 2008a), que lida com a camada de dados e foi desenvolvido especificamente para estender o protocolo IEEE 802.11 para redes veiculares, tem endereços MAC aleatórios, diferentemente do 802.11, que são presumidamente fixos e únicos.

Sumarizando, um bom algoritmo de previsão de posicionamento para MANET deve contar com os seguintes requisitos:

- ❑ Rápido: Realizar previsões em um tempo hábil para que aplicativos sejam capazes de tomar decisões.
- ❑ Leve: Capaz de funcionar em sistemas embarcados.
- ❑ Online: Realizar previsões à medida que os dados são lidos.

- ❑ Acurado: Realizar previsões acertadas.
- ❑ Descentralizado: A previsão acontece no nó de forma independente de outros nós ou infraestrutura.
- ❑ Não invasivo: Consultar o mínimo de informações possíveis, mantendo informações sensíveis privadas.

Neste trabalho, lidamos com o problema de previsão da posição futura de um nó, de forma leve e descentralizada, cobrindo os pontos mencionados anteriormente. A nossa principal motivação para o desenvolvimento do algoritmo de previsão tem a ver com a otimização de comunicação em MANET, por meio de uma gestão proativa de Redes Definidas Por Software (SDN) (MCKEOWN, 2009). Mais especificamente, estamos interessados em desenvolver um algoritmo novo de roteamento proativo para VANET. No entanto, essa mesma capacidade pode ser usada em diferentes aplicações, para vários fins. Por exemplo, pode ser usada para otimizar o tráfego em cidades inteligentes, geração de conteúdo sobre medida para clientes *ad loc* ou perceber potenciais caronas para uma cidade mais ecológica e sustentável. Prever onde o nó vai estar no futuro pode também prevenir a propagação de doenças epidêmicas, como mostra (COLIZZA et al., 2007). Compreender padrões individuais de mobilidade pode levar-nos a prever padrões de mobilidade de grupos, sendo que isso pode também ajudar o planejamento urbano (HORNER; O'KELLY, 2001) e minimizar o estrago em um desastre, como um terremoto ou guerra.

O nosso objetivo é desenvolver e validar algoritmos em tempo real de predição de posicionamento futuro de nós em MANET, que sejam eficientes computacionalmente, podendo ser executados continuamente em dispositivos móveis em geral e que tenham taxa de acerto em sua predição boa o suficiente para desenvolvimento de protocolos de roteamento proativo e demais aplicações. Além disso, os dados devem ser tratados de forma pseudônima, procurando-se utilizar o menor número de informações possíveis, sem acesso a, por exemplo, telefonemas ou relações de amizades. Este objetivo se encaixa em um projeto maior, que desenvolverá uma arquitetura de comunicação baseada em SDN (MCKEOWN, 2009)), que explorará o roteamento proativo para integrar VANET, outros tipos de MANET e a Internet.

Apesar da infraestrutura de redes SDN ser *per se* centralizada, a nossa abordagem não utiliza nenhum controle centralizado, isso não impede, contudo, que as duas abordagens possam coexistir.

Aqui, apresentamos uma abordagem para prever futuras posições de um nó em um ambiente móvel i.e., dada a posição atual do nó e um horizonte de tempo, nosso método retorna um conjunto de posições nos quais o nó provavelmente vai estar, dado o horizonte de tempo, com as probabilidades para cada posição. Nosso método assume que o histórico de movimentação de um nó contém, por si só, boas indicações de suas decisões futuras ou, em outras palavras, assumimos que nós tendem a repetir o mesmo padrão regularmente.

Nosso método também é genérico em relação ao que é uma posição. Isto é, nossos algoritmos não necessariamente lidam com coordenadas geográficas, embora isso também seja possível. Por exemplo, uma posição pode significar a identidade de uma antena de rede celular, uma rede sem fio ou qualquer outra abstração. Claramente, a predição é dependente do tipo de informação utilizada como entrada, isto é, para se prever a coordenada geográfica de um nó é necessário que se monte um histórico de onde este nó passou usando um dispositivo como Sistema de Posicionamento Global (GPS). Para cada posição, é necessário que se armazene também o momento em que foi registrada; a qualidade deste *timestamp* afeta o resultado da predição.

O primeiro algoritmo que definimos foi denominado Previsão Baseada em Feromônios (PheroCast), do inglês, *Pheromone Based ForeCast* (COELHO et al., 2014), que faz previsões de posicionamento em um horizonte temporal utilizando uma busca dentro de um grafo direcionado. Em uma primeira avaliação do algoritmo, utilizamos rastros dos ônibus de Seattle, USA, cujos dados não foram particularmente recolhidos para os nossos propósitos e tem baixa qualidade, com alta fragmentação e baixa granularidade. Mesmo assim, verificamos que nosso algoritmo rende boas previsões de posições dos ônibus. Por exemplo, se dividirmos a área de Seattle em uma grade de células de 1.000 pés², PheroCast prevê corretamente a célula que o ônibus vai estar após 30 segundos em 85.4% dos casos. Nesta dissertação, apresentamos uma versão melhorada do algoritmo em (COELHO et al., 2014), bem como uma avaliação estendida do seu desempenho.

Analizamos também o desempenho de PheroCast em rastros de táxis na cidade de Roma, onde esperávamos um baixo desempenho. Contudo, os algoritmos conseguiram uma bom resultado. Mesmo se tratando de veículos sem um hábito aparente, o resultado foi em média 88.45% de previsões certas considerando células de 300 metros², e manteve um desempenho mesmo quando reduzimos o tamanho de cada célula para 100 metros², atingindo 74.58% de acerto. Finalmente, testamos a capacidade do algoritmo de prever qual rede sem fio com sinal mais forte estaria disponível no futuro, utilizando rastros coletados pelo nosso grupo de pesquisa. Nestes testes, o algoritmo obteve acurácia de 81.02%.

Além de melhorias menores feitas no PheroCast, desenvolvemos uma versão do algoritmo que leva em conta informações sobre a hora do dia em que a previsão é feita, denominado Previsão Baseada em Feromônios Considerando o Tempo (ToD-PheroCast), do inglês *Time of Day PheroCast*. Essa informação já está presente nos dados colhidos, derivado do *timestamp*, por exemplo, se utilizado o tempo no padrão de Interface Portável entre Sistemas Operacionais (POSIX), definido como o número de segundos passados desde 1 de Janeiro de 1970, o *timestamp* 1057403597 corresponde a 5/7/2003 às 11:13:17, UTC.

Também avaliamos o desempenho do algoritmo melhorado, ToD-PheroCast, nos três cenários mencionados. O algoritmo teve desempenho melhor em todos, em especial na-

queles em que a previsão é baixa e pode ser melhorada, pois existe uma limitação no algoritmo, que não prevê lugares que nunca foram visitados. Tal limitação contudo estabelece um limite, testado empiricamente nos três cenários, sempre maior que 90% de acerto.

Em resumo, neste trabalho, fazemos as seguintes contribuições:

1. introduzimos o algoritmo PheroCast para prever a posição futura de nós em ambientes móveis (COELHO et al., 2014).
2. introduzimos o algoritmo ToD-PheroCast, que inclui informações temporais na previsão para obter melhores resultados.
3. apresentamos resultados experimentais extensivos que mostram a precisão dos nossos algoritmos.

O restante desta dissertação é organizado da seguinte forma. Discutimos trabalhos relacionados tanto em forma de fundamentação teórica quanto de trabalhos cujos objetivos são similares ao nosso no Capítulo 2. No Capítulo 3, descrevemos nossa estrutura de dados e o algoritmo básico PheroCast. Na Seção 3.5, descrevemos como fazer previsões utilizando a estrutura de dados. Discutimos alguns pontos fracos no algoritmo e apresentamos uma versão melhorada na Seção 3.6. No Capítulo 4, discutimos método de avaliação dos algoritmos e apresentamos os resultados destas avaliações no Capítulo 5. Concluimos este trabalho no Capítulo 6, com especial atenção à discussão de diversos usos dos nossos algoritmos na Seção 6.2.

Trabalhos Relacionados

Neste capítulo, discutiremos conceitos que servem de fundamentação teórica para nosso trabalho, bem como diversos trabalhos com objetivos semelhantes ao nosso, isto é, prever a posição futura de nós em redes móveis.

2.1 Média Móvel Exponencialmente Ponderada

Nossos algoritmos, como será explicado adiante, relacionam a cada local que o nó visitou em determinado momento a média de tempo que esse ficou ali. Para permitir que o algoritmo leve em conta a idade da informação usada no cálculo da previsão, usamos técnicas para ponderar a idade da informação no cálculo desta média, bem como de outras informações. A técnica usada em nossos algoritmos foi a Média Móvel Exponencialmente Ponderada (EWMA), que dá pesos exponencialmente menores para informações mais antigas.

Como veremos a seguir, a EWMA pode ser classificada como um Modelo Auto-regressivo Integrado de Média Móvel (ARIMA), i.e., um modelo utilizado para entender ou fazer previsões sobre uma série temporal. A EWMA possui diversos usos, como a estimativa dos tempos de ida e volta no TCP (FLOYD et al., 2008).

O modelo ARIMA pode ser descrito normalmente como $ARIMA(p, d, q)$, sendo S_t um termo qualquer da nossa série temporal:

- p é o número de termos autorregressivos – no nosso caso $p = 0$, nossa função não utiliza termos autorregressivos para fazer a média.
- d é o número de diferenças dos termos – no nosso caso só existe uma diferença $d = 1 : s_t = S_t - S_{t-1}$, sendo s_t a diferença dos termos. e
- q é o número de elementos passados que serão multiplicados por σ – no nosso caso, $q = 1$, e σ é uma constante que reflete o quanto os dados passados sobreviverão no futuro.

A principal ideia por trás da EWMA é fazer valores mais recentes terem um peso exponencialmente maior a valores menos recentes. Desta forma, o valor mais novo da EWMA tem um pouco de peso a mais que o valor anterior e esse, por sua vez, tem um pouco mais de peso do que o anterior e assim sucessivamente.

Em poucas palavras, a EWMA é uma média construída iterativamente que dá pesos exponencialmente menores para valores mais velhos. Formalmente, dado um conjunto de $t - 1$ valores e sua EWMA S_{t-1} , a EWMA dos $t - 1$ valores originais mais um novo valor v_t , $S_t = \sigma v_t + (1 - \sigma)S_{t-1} = S_{t-1} + \sigma(v_t - S_{t-1})$. Quanto maior o fator σ , $0 \leq \sigma \leq 1$, menor é o peso de valores mais velhos e mais rápido o envelhecimento.

O parâmetro σ denota o quão rápido o feromônio, que será discutido na Seção seguinte, se dissipa no nosso algoritmo.

2.2 Colônia de Formigas

PheroCast foi inspirado por colônias de formigas e estigmergia, i.e., a capacidade de se comunicar indiretamente, modificando elementos no ambiente (DORIGO; BONABEAU; THERAULAZ, 2000).

As colônia de formigas podem ser vistas como sistemas distribuídos, em que formigas, embora sejam seres simples, realizam coletivamente atividades com um alto grau de complexidade, muitas das quais, são impossíveis para uma só formiga, como explorado, no nosso caso, a capacidade de descobrir um caminho ótimo da colônia para uma fonte de comida.

Formigas não mudam muito seus estados internos e suas comunicações baseiam-se no uso e percepção de *feromônios*, um termo genérico que representa essa forma de comunicação, especialmente utilizado pelos insetos, encontrado também em classes mais elevadas de indivíduos, como os bovinos e até mesmo os seres humanos como estudado em (GRAMMER; FINK; NEAVE, 2005), que relaciona feromônios humanos com atração sexual.

Exemplificaremos aqui um algoritmo para entender como funciona uma formiga artificial em busca de alimento. Consideraremos somente este aspecto da formiga para simplificação da nossa colônia.

Suponha que, conforme uma formiga ande, ela deposite um feromônio que pode ser sentido por outra formiga. Chamamos tal feromônio de f , e atribuímos para ele um grau tal que $1 \geq f \geq 0$. À medida que o tempo passa, f se aproxima mais de zero, pois a ação externa degrada o feromônio, assim como a ação do ambiente rarefaz um perfume.

Na figura 2a, as formigas A e B viajam aleatoriamente em volta do obstáculo entre elas e a comida D. A formiga A consegue viajar mais rapidamente, pois escolheu um caminho melhor do que a formiga B, como visto na Figura 2b, assim quando a formiga A volta, B está na metade do caminho. O feromônio depositado pelas formigas começa a

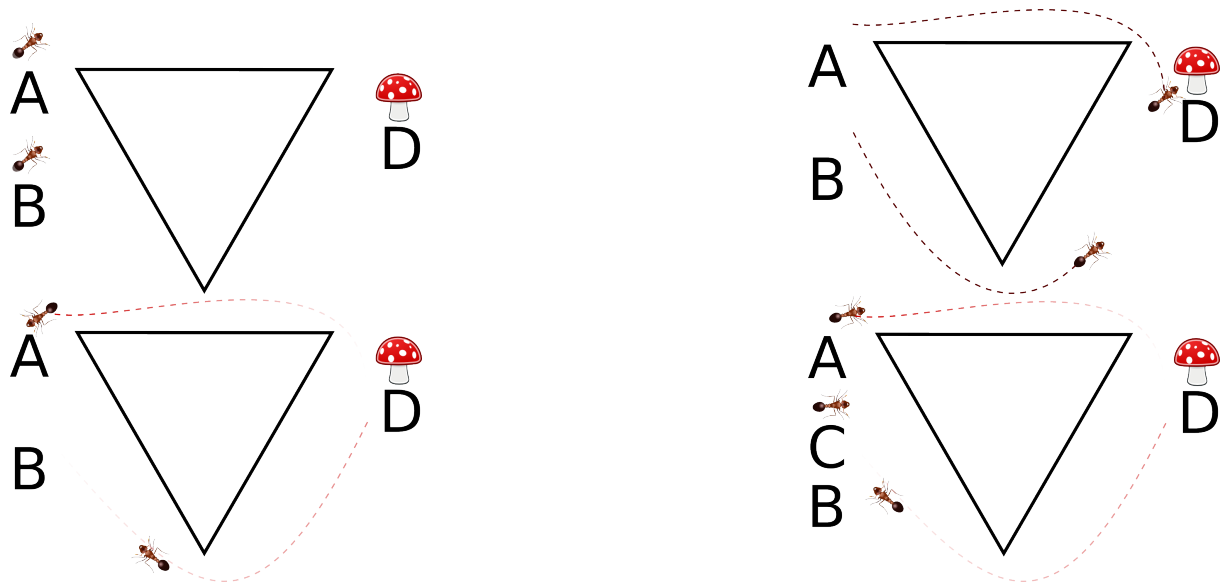


Figura 2 – Exemplo de comunicação por feromônios em formigas.

desaparecer. Uma formiga C chega em seguida, a formiga A começou de volta a viagem para a comida D e C decide ir pelo caminho mais forte, da formiga A. A consequência é que C retroalimenta positivamente (WIENER, 1961) o caminho, obtendo, assim, uma eventual resposta ótima.

A habilidade de formigas encontrarem o melhor caminho de uma série de caminhos possíveis foi estudada em (GOSS et al., 1989), e um modelo matemático estocástico proposto e validado em formigas reais foi proposto. Técnicas de Otimização da Colônia de Formigas (ACO) são usadas principalmente em problemas de otimizações combinatória difíceis em grafos, como para o problema do caixeiro viajante (DORIGO; GAMBARDELLA, 1997), problema do clique máximo ou roteamento de veículos (GAMBARDELLA; TAILLARD; AGAZZI, 1999).

O algoritmo estocástico para movimentação de toda formiga é simples: cada uma delas olha a sua volta e decide pelo caminho que tem mais feromônio, que é degradado pelo ambiente aos poucos.

Enquanto formigas utilizam feromônio para se comunicar umas com as outras, (ToD-)PheroCast possibilita que os nós “deixem marcas” principalmente por eles mesmos e, portanto, podem ser vistos como colônias de formigas independentes, de um só indivíduo. Na Seção 6.2, discutimos algumas das aplicações interessantes que emergem uma vez que as previsões de diversos indivíduos são combinadas.

2.3 Modelos de Mobilidade

Existem, basicamente dois tipos de modelos de mobilidade, sintéticos e reais (ROY, 2011). Modelos sintéticos são gerados a partir de algoritmos e existe uma vasta literatura sobre esses modelos, como veremos a seguir, já modelos reais são coletados de dados

produzidos em ambientes reais, são mais confiáveis, porém são estáticos, diferentemente dos modelos sintéticos, que podem ser gerados quantas vezes se queira, de forma diferente.

A validação dos nossos algoritmos esbarra na necessidade de grandes massas de dados, com rastros de mobilidade de usuários e nós em MANET. Existem alguns repositórios com tais dados que podem ser explorados, como detalhado no Capítulo 4. A maior parte tem alta granularidade de tempo entre os *logs* de posicionamento consecutivos ou cobrem um curto intervalo de tempo de um nó. Este último aspecto é muito importante, pois, para que se possa fazer uma predição adequada, é necessário que se tenham *traces* representativos dos históricos dos usuários, cobrindo a sazonalidade de seus movimentos. Uma forma de se contornar a insuficiência de dados é validar os algoritmos usando rastros sintéticos, gerados a partir de modelos da mobilidade de nós em MANET.

A literatura é rica em modelos de mobilidade, cada um tratando de diferentes domínios de um problema, elaborados para recriar cenários reais. Contudo, a maioria dos modelos encontrados não considera hábitos dos nós, como rotas normalmente seguidas e sazonalidade do movimento. O modelo *Random Way Point*, por exemplo, simula um trajeto aleatório em cada nó, quando o nó chega ao destino escolhe outro caminho aleatório e realiza novo trajeto. Um modelo que mais se aproxima do que seria ideal, no contexto deste trabalho, é o Modelo de Movimentação em Dias Úteis (WDMM) (EKMAN et al., 2008).

Uma pequena variação nos parâmetros do modelo pode ter um grande impacto no algoritmo que se quer observar (CAMP; BOLENG; DAVIES, 2002) e a busca de um modelo que se aproxime mais à realidade é hoje o principal desafio para validação de protocolos de MANET (ROY, 2011). O problema se origina do grande número de variáveis ao simular o comportamento humano, um bom modelo tem de simular acidentes, obras na malha viária, comportamento caótico e demais comportamentos naturais humanos. Tal simulação esbarra na simulação da própria realidade, com informações dependentes umas das outras.

Uma simulação simples levaria inevitavelmente a dados tendenciosos, uma vez que o algoritmo se comportaria de forma irreal em situações quase perfeitas, e.g., simular comportamentos sazonais artificiais, uma variável que é imprescindível ao nosso algoritmo.

Embora tenhamos explorado a construção de rastros sintéticos utilizando matrizes Origem-Destino¹ e explorando a metodologia utilizada para criação do cenário TAPAS (SUMO, 2015), que busca reproduzir com o maior nível de realismo possível, tráfego de carros na área urbana de Köln, na Alemanha; o nosso trabalho esbarrou na necessidade de alto poder computacional para simulações de grandes populações e longos períodos de tempo. Além disso, não contamos com dados completos, como de zoneamento e estatísticas da população. Assim, na versão final do trabalho, não utilizamos rastros sintéticos.

¹ Matrizes Origem-Destino aglomeram dados de viagens de indivíduos para determinadas áreas a , na forma de uma matriz $N \times N$, tal que $a \in N$.

Os modelos sintéticos podem ter aplicações para validação de outros algoritmos, nos quais o padrão de movimentação dos nós não é um fator dominante.

Na versão final do nosso trabalho, optamos por utilizar somente os seguintes modelos reais, discutidos em detalhes no Capítulo 4:

- ❑ táxis de Roma.
- ❑ ônibus em São Francisco.
- ❑ *smartphones* em Uberlândia.

A decisão de utilizar somente rastros reais deve-se ao fato de que, por mais bem moldado, um modelo sintético sempre dá margem para dúvidas no que diz respeito à simulação correta, em especial para o nosso caso, uma vez que existe um grande número de variáveis a serem simuladas e a alteração de uma delas pode impactar drasticamente na execução do algoritmo, já um modelo real, mesmo que intuitivamente ruim para o nosso algoritmo, como o caso do movimento aparentemente errático de um táxi, é suficiente para testar a eficácia do algoritmo em situações adversas.

2.4 Previsão de posicionamento

A mobilidade humana é previsível, como mostrado em (BROCKMANN; HUFNAGEL; GEISEL, 2006), em diversos níveis, e diversos trabalhos exploraram esta previsibilidade no contexto de redes móveis. Esta característica é fundamental para o trabalho apresentado aqui. Isto é, assumimos que o movimento dos nós em um ambiente de redes móveis segue padrões, refletidos no histórico desta movimentação, e propomos métodos para predição da posição de nós em um ambiente de rede móvel usando este histórico.

Em 2011, a empresa de telefonia Nokia divulgou o Desafio de Dados Móveis da Nokia (NMDC) (LAURILA et al., 2012), uma iniciativa de pesquisa que levou à coleta de dados de cerca de duzentos voluntários, incluindo coordenadas de GPS, chamadas telefônicas, Serviço de Mensagens Curtas (SMS), *Bluetooth* e *Wireless*. Os dados foram coletados entre 2009 e 2011, e, posteriormente, divulgados para a comunidade científica juntamente com diversos desafios, incluindo a predição de posicionamento futuro (*Next Place Prediction*).

Em (DOMENICO; LIMA; MUSOLESI, 2012), foi feita uma análise de séries temporais, com rastros de mobilidade de usuários combinados com informações sociais de *smart-phones*, utilizando os dados do NMDC para prever o posicionamento futuro de nós. Uma série temporal é uma sequência de pontos, frequentemente coletados ao longo de um intervalo de tempo. Séries temporais são utilizadas, entre outras coisas, para reconhecimento de padrões, estatística e previsões (ZISSIS; XIDIAS; LEKKAS, 2015). Previsões são feitas, por exemplo, em bolsas de valores, previsão de tornados ou furacões e, como no caso do trabalho apresentado aqui, em diversos outros, previsão de posicionamento.

Os dados analisados por (DOMENICO; LIMA; MUSOLESI, 2012) consistiam de informações de trinta e nove usuários. A partir das séries temporais, o algoritmo infere a rede social dos usuários e usa essa informação para fazer a previsão. Sendo assim, se uma série de amigos se comunica com certa localidade temporal e todos se deslocam em certa direção, então é razoável supor que os mesmos vão para um mesmo destino, possivelmente frequentemente visitado pelo grupo.

Para fornecer previsões precisas, parece razoável que tal abordagem deva ser realizada de uma forma centralizada, na qual os dados dos “amigos” são agregados e correlacionados. Diferentemente, a abordagem que introduzimos aqui apenas considera os rastros de um único usuário e pode ser executada unilateralmente por cada nó na rede móvel. A nossa abordagem é muito leve se comparada à análise de séries temporais em (DOMENICO; LIMA; MUSOLESI, 2012) e bem adequada para dispositivos de baixo processamento e com restrições de energia, como os atuais *smartphones*.

(DOMENICO; LIMA; MUSOLESI, 2012) também aborda a previsão usando rastros de posicionamento de 500 táxis de São Francisco (PIORKOWSKI; SARAFIJANOVIC-DJUKIC; GROSSGLAUSER, 2009). Neste estudo, considera-se apenas a posição dos taxis, já que informação social não é disponibilizada. No cenário, a abordagem em (DOMENICO; LIMA; MUSOLESI, 2012) leva a previsões com um erro de, em média, 0.19 graus, aproximadamente 20km. Quando testado no mesmo conjunto de dados, o ToD-PheroCast, apresentado neste trabalho obteve eficácia máxima de 59.33% em células de 300m, ou seja, o algoritmo acerta em mais da metade das vezes com erro 67 vezes menor do que o proposto por (DOMENICO; LIMA; MUSOLESI, 2012).

Algumas abordagens para previsão da posição futura de nós utilizam Redes Neurais. Uma Rede Neural é uma rede de elementos simples, *neurônios*, a ideia por trás das Redes Neurais é imitar o funcionamento de um sistema nervoso central. As Redes Neurais são amplamente utilizadas para reconhecimento de padrões, o que a torna uma abordagem promissora para aprender sobre padrões de movimentação de nós. Um algoritmo de predição utilizando Redes Neurais é proposto em (CAPKA; BOUTABA, 2004). Nessa abordagem, a trajetória de um nó é marcada conforme ele anda sobre a influência de antenas e o algoritmo de *back propagation* é utilizado para treinar a Rede Neural.

Em (AKOUSH; SAMEH, 2007), os autores utilizam Redes Neurais Bayesianas para prever onde um nó estará, baseado nos seus movimentos passados. O trabalho é focado em utilizar dados das redes *WiFi* para estabelecer identificações dos locais visitados. O principal desafio de se usar Redes Neurais para previsão é saber quantos passos de execução são precisos para convergir a uma solução com um erro aceitável e reconhece-se que a convergência pode ser muito lenta se utilizada na prática.

Para os testes em (AKOUSH; SAMEH, 2007), também foram utilizados os dados de (EAGLE; PENTLAND, 2006). O projeto conta com mais de 500 mil horas de dados contínuos do comportamento de 100 pessoas. Os autores utilizam informações sobre

qual torre celular o indivíduo esteve, aglomeram as células adjacentes e utilizam essa informação para treinar a Rede Neural. Uma vez treinada, ela é capaz de prever onde um indivíduo estará no próximo instante. O dia da semana também é levado em conta, como uma entrada da Rede Neural.

No experimento, foi utilizado um mês para o treinamento e um mês para os testes. Sempre que uma entrada é passada, a Rede Neural tem como saída a próxima célula que o indivíduo estará, uma abordagem similar à nossa, porém, no nosso caso podemos prever onde o nó estará com um horizonte arbitrário de tempo futuro.

O algoritmo foi comparado com técnicas semelhantes, todas utilizando Redes Neurais, em que sua eficácia média foi de 37.6%. Os autores argumentam que um modelo por indivíduo é impraticável em uma grande rede *wireless*, isto acontece devido ao método proposto requerer uma centralização inerente, diferentemente do nosso método, em que cada nó cria seu próprio modelo para análise, podendo assim ser descentralizado.

Em (PAPANDREA; GIORDANO, 2014), é proposto um algoritmo de predição de localização que visa melhorar a precisão de serviços de localização em *smartphones* e minimizar o consumo dos seus recursos, principalmente devido ao uso do GPS (TSUI, 2000). Isto é, neste trabalho, os nós móveis não são apenas um meio para a predição, mas também se beneficiam diretamente da mesma. O algoritmo proposto, contudo, por ser custoso em termos computacionais, é executado uma vez por dia, em período de ociosidade do dispositivo e, portanto, não é adequado aos cenários em que a predição *online* deva ser feita, como no caso de roteamento proativo².

Em (LIU; MAGUIRE, 1995), os autores defendem que não só é preciso agir passivamente conforme as mudanças na topologia da rede, mas antecipar agressivamente o comportamento dos nós, fazendo predições sempre que possível dos seus comportamentos. Assim como os outros estudos em previsão de posicionamento de nós móveis, o algoritmo assume que todos os nós têm um certo grau de regularidade em sua movimentação. Para isso, os autores fazem dois modelos de movimentação, o *Movimento em Círculo* (MC) e *Movimento em Trilha* (MT).

O MC parte da suposição de que sempre que o nó se move para um lugar, eventualmente retorna à posição inicial. Assim pode-se construir uma máquina de estados cíclica, modelada através de uma cadeia de Markov. Argumenta-se que o MC é usado para manter o estado dos movimentos em longo prazo, já o modelo de MT é menos restritivo, em um MC pode-se ter dois ou mais MTs. Além desses dois modelos, foi implementada uma cadeia de Markov, para o caso em que o nó se comporta aleatoriamente, a computação da cadeia de Markov é dispendiosa, portanto os autores reconhecem que ela não deve ser usada frequentemente.

² Em um protocolo proativo, o entendimento da topologia deve ser mantido sempre, para que se saiba informações de antemão. Um grande problema em MANET, já que os nós tem liberdade para se mover, alterando drasticamente a topologia da rede.

Para testar o algoritmo, os autores de (LIU; MAGUIRE, 1995) utilizaram um modelo simulado e obtiveram cerca de 95% de acerto. Embora esse número é consideravelmente alto, os autores utilizaram um modelo simplório, no qual os nós tinham um claro padrão de mobilidade imposto pelo modelo, sem congestionamento e outros ruídos que afetariam drasticamente o modelo. O modelo simula aleatoriedade e o algoritmo consegue resultados acima de 50% para um cenário de até 70% do movimento aleatório, mas uma análise mais detalhada faz-se necessária, especialmente em cenários reais ou simulações mais robustas.

Um estudo sobre trajetórias de 100.000 telefones móveis de usuários anonimizados (GONZALEZ; HIDALGO; BARABASI, 2008), cujas posições foram monitoradas durante um período de seis meses, mostra que os indivíduos tendem a escolher um pequeno conjunto de locais visitados regularmente. De acordo com (GONZALEZ; HIDALGO; BARABASI, 2008) “Para cada usuário pode ser atribuída uma área bem definida, definida por casa e local de trabalho, em que ele ou ela pode ser encontrada na maioria das vezes”. Isto é, a presença de um usuário em um determinado tempo e localização é previsível. Em nosso trabalho, prevemos onde o nó estará a qualquer hora do dia, mesmo que esse esteja em movimento, e dentro de limiares de tempo pequenos.

Uma análise sobre 65 pessoas feita em mais de 10 meses (ZHENG et al., 2008) mostra que vários tipos de movimento podem ser previstos, tais como “De carro” ou “A pé”, apenas olhando para os dados do GPS. Contudo, na análise, usaram-se outras informações, como a topologia viária e a posição de estacionamento, desta forma, sabendo-se a posição de estacionamentos pode-se prever que quando um indivíduo vai a um desses o estado vai mudar para “De carro” ou “A pé”. Sendo assim, o algoritmo proposto pôde prever com 76.2% o uso de quatro meios de transporte.

A melhoria citada nos trabalhos futuros de (ZHENG et al., 2008) é levar em consideração informações relacionadas à hora do dia. Essa informação é utilizada em nosso trabalho para aprimorar o algoritmo proposto, explicado em mais detalhes na Seção 3.6.

Os autores de LOCADIO (KRUMM; HORVITZ, 2004) argumentam que o contexto é tão importante quanto saber a posição de um indivíduo, i.e., saber o que o indivíduo está fazendo, seus objetivos e outros metadados. Tal contexto pode ser utilizado para refinar conteúdos específicos individuais ou acionar comportamentos automáticos.

LOCADIO tenta detectar mobilidade utilizando apenas sinais de rádio *WiFi*, como outros trabalhos mencionados, inclusive o nosso. A razão é o alto consumo energético de dispositivos GPS em *smartphones*, onde comumente são empregados. Conseguiu-se diferenciar entre dois estados, “Parado” e “Em Movimento”, com uma precisão de 87%, analisando a variação da força do sinal de rádio (mais variância significa, provavelmente, mais movimento).

Em LOCADIO, foram feitas 10 viagens para uma predição de posição, com um erro de aproximadamente 1,5 metros. Entretanto, os testes foram todos feitos em um ambiente fechado e controlado, em que a posição geográfica do roteador já era previamente conhe-

cida, isto não ocorre em um cenário real, onde é difícil saber a posição de um roteador qualquer, embora um grande observador passivo, como *Google*, colha abertamente esses dados através dos *smartphones*, que têm instalado o *Android*.

Quando analisamos redes *WiFi* no nosso trabalho, não sabemos previamente a posição dos roteadores; não é feita nenhuma análise para saber a posição geográfica do indivíduo. Contudo, assume-se que roteadores de *WiFi* não se movam, podemos, assim, estabelecer uma posição lógica do indivíduo, sem necessidade de conhecer sua real posição geográfica.

Mais focado em redes veiculares, o trabalho de (SILVA et al., 2013) se concentra em comunicação com ônibus em uma rota bem conhecida. No artigo, os ônibus atualizam um serviço central com a sua posição atual sempre que é realizado um *handshake* com um dispositivo da infraestrutura. Sempre que uma mensagem é enviada para o ônibus, os autores propõem enviar mensagens replicadas a última posição conhecida do ônibus mais três pontos de acesso seguintes na rota do mesmo. Diferentemente do nosso trabalho, não é feita previsão baseada no histórico em (SILVA et al., 2013).

Vislumbramos que a previsão da posição de nós permitirá novos mecanismos de roteamento em VANET, permitindo que as mensagens sejam redirecionadas para onde o nó estará em vez de onde está. (JEONG et al., 2010) tem explorado o mesmo problema, usando rastros de GPS na previsão. Acreditamos que a maioria dos proprietários de GPS raramente planejarão suas viagens dentro de uma área que conhecem bem. Nessas áreas, PheroCast vai utilizar o histórico do usuário na previsão; se uma rota de GPS está disponível, então pode também ser considerada, mas isto não é um requisito.

Na maioria das obras mencionadas, é necessário um treino para que o algoritmo seja treinado e avaliado em outros dados, na maioria das vezes, um subconjunto de todos os dados. Por exemplo, em (ZHENG et al., 2008), 70% dos dados coletados são usados no treinamento e os outros 30%, para avaliar o desempenho do algoritmo. Os algoritmos apresentados neste trabalho, PheroCast e ToD-Pherocast, não necessitam de um período de treinamento. Isto é, os mesmos podem ser usados desde o primeiro momento para a predição, construindo o histórico usado na predição de forma *on-line*. Obviamente, previsões feitas sobre um histórico rico serão potencialmente mais precisas, ficando a cargo do utilizador do algoritmo ignorar as previsões iniciais ou não. A “curva de aprendizagem” dos algoritmos, contudo, é pequena. Em trabalhos futuros, pretendemos desenvolver métodos para detectar quando esta curva é suavizada e determinar a partir de quando a precisão das predições provavelmente alcançou seu pico.

Em (GOHIL, 2014), foi feita uma comparação analítica de técnicas de Mineração de Dados, Modelos de Markov, Redes Neurais, Inteligência Artificial e Lógica Fuzzy. Na Tabela 1, retirada de (GOHIL, 2014), estão sumarizados os principais resultados da análise.

Os dados da Tabela 1 não fornecem um maior detalhe quanto ao erro utilizado por cada método na predição, contudo, podemos ter uma ideia das técnicas e resultados obtidos

Método Utilizado	Ano	Autor	Modelo Proposto	Dataset	Acurácia
Data Mining	2004	[Gotkhan Yavas]	Minerando padrões de mobilidade de usuários por grafos	Somente um usuário	–
	2012	[Thuy Van T. Duong]	Data mining espaço-temporal	Próprio dataset	70%
	2012	[Huiji Gao]	Modelo para capturar a trajetória espaço-temporal	NMDC	50%
	2012	[Le-Hung Tran]	Árvore de decisão específica por usuário	NMDC	61.1%
	2013	[Mr. Mohammad]	Mineração por associação de regra	–	–
	2013	[João Bartolo]	Técnicas de classificação	NMDC	17%
Modelo de Markov	1996	[George Liu]	Modelo de Predição de Movimento	Dados próprios	–
	2013	[Jorge Alvarez-Lozano]	Modelo HMM	4 anos de dados de GPS de 178 usuários	30%
Redes Neurais	2007	[Sherif Akoush Ahmed]	Redes Neurais Bayesianas	Mais de 500 mil horas	–
	2011	[Saikath Bhattacharya]	Redes Neurais de várias camadas	Dados de um usuário	–
	2011	[Maryam Borna]	Estrutura Neuro-Fuzzy adaptativa	Dados próprios	–
	2012	[Velmurugan. L]	Múltiplas técnicas	Dados de Dartmouth College	78.50%
	2012	[Vincent Etter]	Múltiplas técnicas	NMDC	56.22%
	2013	[Smita Parija]	–	–	0.84
Inteligência Artificial	2012	[Jingjing Wang]	SVM com RBF	NMDC	50%
Lógica Fuzzy	2010	[Sajal Saha]	Algoritmo de sistema de interface fuzzy	500 usuários aleatórios	–
	2012	[Ying Zhu]	Método de Machine Learning	NMDC	72%
	2012	N. Mallikharjuna Rao, Prof M. M Naidu	Fuzzy logic e banco de dados fuzzy	–	–

Tabela 1 – Tabela de comparação de estudos de vários modelos de predição de mobilidade (GOHIL, 2014)

nos diversos trabalhos mencionados em (GOHIL, 2014).

PheroCast

Neste capítulo, descrevemos as duas contribuições principais deste trabalho: o algoritmo PheroCast (COELHO et al., 2014) e sua versão estendida, ToD-PheroCast, introduzido na Seção 3.6. Começaremos definindo conceitos básicos para a compreensão do algoritmo e, logo em seguida introduziremos os algoritmos na ordem em que foram concebidos.

Os algoritmos de PheroCast constroem grafos direcionados para representar o caminho de um nó. O grafo sintetiza os *logs* realizados pelo nó sobre seu movimento através de *locais*, onde um “local” pode ter significados diferentes, que em breve discutiremos. Cada *local* visitado é associado a um vértice no grafo. Ao realizar registros frequentes durante um longo período de tempo, o grafo correspondente espelhará os hábitos de mobilidade do nó, que pode então ser utilizado para prever o seu movimento futuro.

3.1 Marcos e Locais

No contexto dos algoritmos de PheroCast, dizemos que um *marco* é uma característica única de uma determinada área e um *local* de um marco é a área mais proeminente em torno do *marco*. Se alguém visita Paris, por exemplo, a Torre Eiffel e o Arco do Triunfo são exemplos perfeitos de *marcos* e a área onde a atenção do turista é mais provável, envolvendo cada monumento, é o *local*. Claramente, alguns *marcos* têm *locais* maiores do que outros, também não existe uma restrição quanto à forma dos *locais*, podendo os mesmos serem irregulares.

Em termos de redes móveis, diferentes *marcos* e *locais* podem ser considerados. Por exemplo, torres de telefonia celular e pontos de acesso sem fio são exemplos claros e seus *locais* são facilmente definidos como a área em que eles fornecem a melhor alternativa de conexão. Outro exemplo que exploraremos mais tarde é o par de coordenadas de latitude e longitude, arredondados até uma certa granularidade. Se estes são os *locais*, então, por exemplo, o *local* do *marco* de referência (10,48) é a superfície dentro das coordenadas

$[(10, 48), (11, 49)[$. Estes exemplos díspares mostram como nossos algoritmos são genéricos em relação às posições da previsão de nós.

3.2 Phero Trails

Os algoritmos de PheroCast rastreiam os nós com base nos *locais* que eles visitam. Para fazer isso, em tempos regulares, os nós guardam o nome dos *marcos* dos *locais* que se encontram no momento. Guardar o nome de um *marco* diferente daquele previamente visitado implica que o nó se moveu, enquanto guardar o mesmo nome do *marco* significa que o nó ficou parado, durante o intervalo registrado, dentro do *local* do *marco*. Claramente, quanto mais fina a granularidade em que são feitos os registros, mais preciso é o rastreamento. Além da posição, os registros também incluem a direção do movimento. Assim que os dados são gerados, são consumidos para atualizar o grafo direcionado que é, em resumo, o histórico do nó, que nomeamos *Phero Mapas*.

Para simplificar a apresentação da estrutura de dados, primeiro convertemos os *logs* em grafos dirigidos simples que representam as viagens individuais de um nó, onde a viagem é, basicamente, um conjunto de entradas sucessivas do *log*. Nós chamamos esses de dígrafos *trilhas*. Cada vértice v em uma trilha contém dois valores, id e p ; nós nos referimos a esses campos do vértice v como $v.id$ e $v.p$ (COELHO et al., 2014):

- $id = (i, d)$: Uma identificação única do vértice v , onde i é o nome do *marco* que esse vértice representa e d é a direção para a qual o nó estava se movendo quando a respectiva entrada foi registrada no *log*. Dois nós com id de mesmo i serão efetivamente iguais, para efeito de comparação, mesmo que o seus d sejam diferentes, e.g., o nó de $id = (1, N)$ e o nó de $id = (1, E)$ representam o mesmo marco;
- p : O peso do vértice, corresponde a quanto de feromônio foi deixado no *marco*, i.e., o número de entradas consecutivas com o mesmo id .

A Figura 3 mostra como foi a trajetória de um veículo, que se move a uma velocidade constante em um espaço contendo alguns *marcos*. A Figura 4 mostra como foram coletados os dados discretos desse veículo que faz três viagens, cada viagem tem um *timestamp*, cujo valor é o número de viagens criadas antes mais 1.

A Figura 5 mostra três exemplos de *trilha*, em um cenário no qual uma entrada de *log* é feita a cada 1 unidade de tempo, por exemplo, a *trilha* de *timestamp* 1 mostra que o nó iniciou uma viagem no *local* 0, movendo-se ao norte, e terminou-a no local 12, a leste e fez três entradas consecutivas no último *local* visitado.

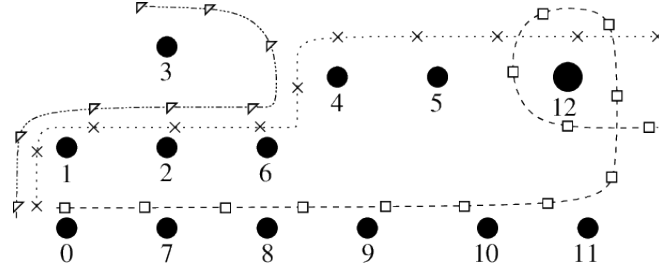


Figura 3 – Exemplo da trajetória de um nó. Os pontos maiores numerados identificam marcos, e as trajetórias com os relativos símbolos correspondentes na Figura 4.

Entrada #	Viagem 1 ×		Viagem 2 □		Viagem 3 ▽	
	Lugar	Direção	Local	Direção	Local	Direção
0	0	N	0	E	0	N
1	1	N	7	E	1	N
2	1	E	8	E	1	E
3	2	E	8	E	2	E
4	6	E	9	E	6	E
5	4	N	10	E	4	N
6	4	E	11	E	3	W
7	5	E	11	N	3	W
8	12	E	12	N		
9	12	E	12	N		
10	12	E	12	W		
11			12	S		
12			12	E		
13			12	E		

Figura 4 – Exemplo de um *log* de uma viagem. Para cada viagem, a primeira coluna tem o *local* onde o nó estava e na segunda coluna a direção que o nó estava seguindo quando a entrada foi adicionada no *log*.

3.3 Phero Mapas

Um *Phero Mapa* é um grafo orientado que resume as viagens do nó móvel, no qual são feitas consultas sobre a sua posição futura esperada. Neste trabalho, vamos nos concentrar em mapas construídos a partir das viagens de um único nó. No entanto, várias aplicações interessantes de mapas combinados com o histórico de nós diferentes podem ser propostas, como brevemente discutido na Seção 6.2.

Cada vértice v no *Phero Mapa* contém, originalmente, três valores:

- $id = (i, d)$ – Identificação única do vértice v , na forma (i, d) , na qual i é o nome do *local* e d a direção (por exemplo, Norte (N) , Sudoeste (SE), etc);
- a – Peso médio do vértice v (peso médio dos vértices com o mesmo identificador nas viagens realizadas no mapa);

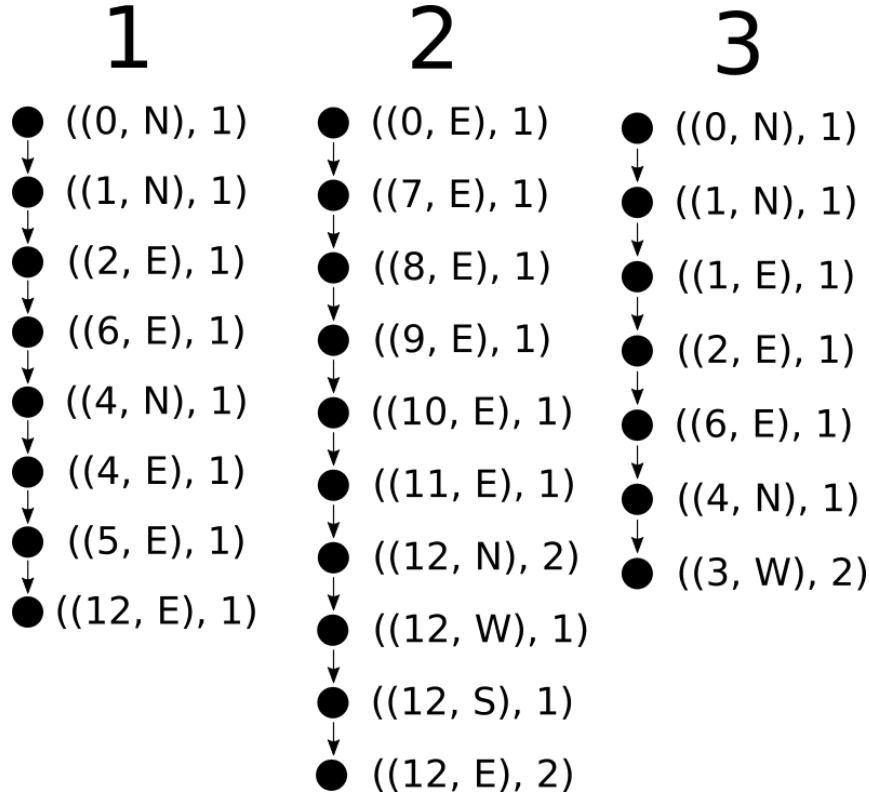


Figura 5 – Exemplo de *trilha*. A localização dos *locais* foi registrada a cada unidade de tempo. Cada vértice tem uma tupla (id, p) com o identificador do nó e seu peso.

□ t – Número de vezes que o *marco* i for visitado.

Dado um conjunto de *Phero Trails*, chamado aqui de *trilhas*, o *Phero Mapa* correspondente tem um vértice para cada um do conjunto de *trilhas*, com o mesmo *id*. Para cada vértice w , o valor de $w.a$ é a média dos valores de p para todo vértice com *id* $w.id$ em todas *trilha* do conjunto e $w.t$ é o número desses vértices.

Formalmente, dado o conjunto $T = \{(V_1, E_1), \dots, (V_n, E_n)\}$, em que cada par (V_t, E_t) , $1 \leq t \leq n$ é uma *trilha* com vértices V_t e arestas E_t , o *Phero Mapa* correspondente $M_T = (V, E)$ é construído como descrito nos Algoritmos 1 e 2. O *Phero Mapa* é construído como se segue:

$$\square \exists w \in V \iff \exists v \in U_T, w.id = v.id$$

$$\square \forall w \in V, w.a = Avg(\{v.p : v \in U_T, v.id = w.id\}, \sigma)$$

$$\square \forall w \in V, w.t = |\{v : v \in U_T, v.id = w.id\}|$$

$$\square \exists (e, e') \in E \iff \exists E_i : \exists (f, f') \in E_i : e.id = f.id \wedge e'.id = f'.id$$

O pseudocódigo do Algoritmo 1 constrói o *Phero Mapa* de acordo com esta definição.

Algoritmo 1 Construindo o Phero Mapa

```

 $V \leftarrow \emptyset$ 
 $E \leftarrow \emptyset$ 
for all phero trail  $(V_t, E_t) \in T$  do
  for all vertex  $v \in V_t$  do
    if  $\neg \exists w \in V : w.id = v.id$  then
       $V \leftarrow V \cup \{(id = v.id, a = v.p, t = 1)\}$ 
    else
       $w.a \leftarrow \frac{w.a * w.t + v.p}{w.t + 1}$ 
       $w.t \leftarrow w.t + 1$ 
    end if
  end for

  for all edge  $(f, f') \in E_t$  do
    if  $\neg \exists (e, e') \in E : e.id = f.id \wedge e'.id = f'.id$  then
       $E \leftarrow E \cup \{(f, f')\}$ 
    end if
  end for
end for

```

A Figura 6 mostra o *Phero Mapa* da Figura 4, com as três viagens conjuntas. A partir de um *Phero Mapa*, pode-se tirar algumas conclusões sobre o movimento de um nó, como no *Phero Mapa* da Figura 6, em que se pode concluir que o nó usa duas rotas diferentes para viajar a partir do *marco* 0 ao 12 e que, em média, permanece dentro do *marco* 8 duas vezes mais do que no *local* 9. Mais interessante é que é possível prever a direção do nó. Por exemplo, a partir do mesmo mapa, se no *local* 6, viajando para o leste, o nó pode estar indo para os *locais* 3 ou 12, mas, uma vez no *local* 4, é provável que seja apenas indo em direção ao 12.

Para conclusões precisas, dependemos de quão certas as informações contidas no mapa são e a sua construção, tal como apresentado no Algoritmo 1, não lida com uma fonte importante de imprecisão: o envelhecimento da informação. Mostramos como corrigir este problema na Seção 3.4, discutiremos também informações importantes da implementação.

3.4 Envelhecendo a Informação

A primeira modificação feita ao Algoritmo 1 foi a ideia de envelhecer a informação, assim como são envelhecidos os rastros de formigas, como descrito na Seção 2.2. Devemos alterar as informações, tal que cada vértice no *Phero Mapa* guarda agora quatro valores:

- $id = (i, d)$ – Identificação única do vértice v , na forma (i, d) , onde i é o nome do *local* e d a direção (por exemplo, N, SE, etc);
- a – Peso do vértice v (peso médio dos vértices com o mesmo identificador nas viagens realizadas no mapa);

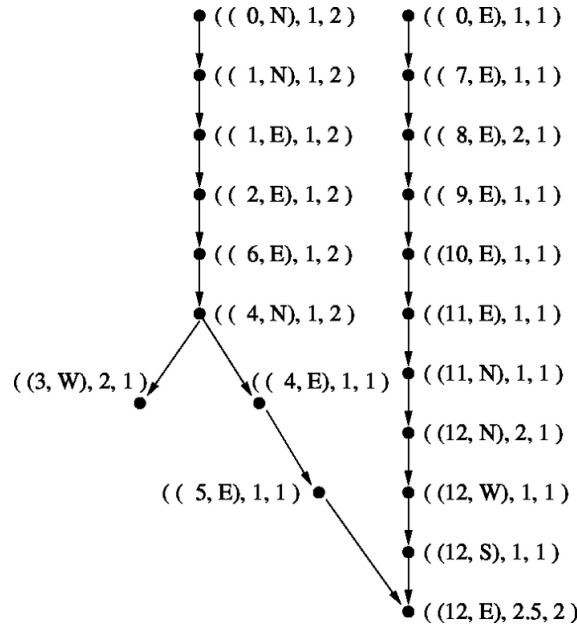


Figura 6 – Exemplo de Phero Mapa da Viagem 1, ilustrada na Figura 4. em todos os nós $a = 1, t = 1$, exceto no nó $12E$, onde $a = 1, t = 3$

- t – Fator de envelhecimento do vértice v , (fator médio de envelhecimento das viagens que continham vértices com o mesmo id , no mapa); e,
- s – *Timestamp* da viagem que causou a atualização do último vértice.

Guardamos um *timestamp* para saber qual foi a última vez que passamos no vértice. Alteramos também o t de cada vértice, tal que, agora, este valor reflete o envelhecimento do vértice, como especificamos abaixo. O Algoritmo 2 constrói o *Phero Mapa* já levando em conta essas modificações.

A fim de levar em conta as alterações no comportamento dos nós, os algoritmos de PheroCast consideram a idade da informação a ser adicionada aos mapas. Fazemos isso através da atribuição de um fator de idade para uma viagem, que decai conforme novas viagens são feitas, até um ponto em que a viagem começa a ser ignorada no mapa. Desta forma, os nós que não são visitados em um longo período terão pouco impacto nas previsões. Nós também envelhecemos a informação sobre quanto tempo um nó permanece dentro de um *local*. Se houver uma mudança no tempo de uma viagem, o grafo, eventualmente, se ajustará aos valores atuais. Em ambos os casos, usamos uma EWMA para realizar o envelhecimento.

Em linhas gerais, para construir o mapa, *trilhas* são processadas na ordem em que são produzidas, i.e. o campo s nos seus vértices. Para cada vértice da *trilha*, um vértice para o mesmo *local* e direção, com o mesmo identificador, é adicionado ao mapa, se ele ainda não existe. Se já existir, ele é ajustado como se segue:

- $w.a$ é atualizado para a EWMA do seu valor atual e $v.p$ é atualizado.

Algoritmo 2 Construindo um Phero Mapa Envelhecendo Arestas(COELHO et al., 2014)

```

 $V \leftarrow \emptyset$ 
 $E \leftarrow \emptyset$ 
for all phero trail  $(V_t, E_t) \in T$ , taken in the order of their timestamp,  $\tau$  do
  for all vertex  $v \in V_t$  do
    if  $\nexists w \in V : w.id = v.id$  then
       $V \leftarrow V \cup \{(id = v.id, s = \tau, a = v.p, t = \sigma)\}$ 
    else
       $w.s \leftarrow \tau$ 
       $w.a \leftarrow w.a + \sigma(v.p - w.a)$ 
       $w.t \leftarrow \sigma + (1 - \sigma)^{\tau - w.s} * w.t$ 
    end if
  end for

  for all edge  $(f, f') \in E_t$  do
    if  $\nexists (e, e') \in E : e.id = f.id \wedge e'.id = f'.id$  then
       $E \leftarrow E \cup \{(f, f')\}$ 
    end if
  end for
end for

```

- $w.t$ é multiplicado por $1 - \sigma$ o número de vezes que houveram *trilhas* entre $w.s$ e a *trilha* que está sendo processada e em seguida somada à σ .

Estas medidas têm o efeito final de modificar o valor de $v.a$ aproximando-o a EWMA dos valores de p para todos os vértices com id $v.id$ em todas as *trilhas* já processadas e $v.t$ o fator de envelhecimento médio das viagens que contêm tais vértices. Finalmente, para cada aresta na *trilha*, uma aresta semelhante é assegurada de existir no mapa, que conecta os vértices correspondentes.

Deve-se notar que este algoritmo não requer um conhecimento a priori de T , ou seja, *trilhas* são processadas da mesma forma em que são geradas e, portanto, nós móveis não precisam armazenar as *trilhas* e podem, a partir do primeiro dia, utilizar o mapa para prever, mesmo se a acurácia provavelmente for baixa no começo.

Uma vez construído, o *Phero Mapa* reflete o histórico do movimento do nó. Considerando o modelo apresentado na Figura 7, vamos supor que um nó viaje de casa para o trabalho e depois para a escola e novamente para casa, todos os dias.

A partir do *Phero Mapa* inferido pelas rotas da Figura 7, pode-se ver que, normalmente, o nó permanece sob a cobertura da antena A1 durante menos de 5 minutos, antes de se mudar para a célula em torno da antena A2. Ele também mostra que, depois de deixar a escola, o nó chega de volta para casa, passando pela antena A3 em 12 minutos, independentemente da forma como se segue. Assim, quando se sabe que ele deixou a escola há 4 minutos, o nó vai ainda levar, em média, 8 minutos para chegar em casa. Quanto mais recentes os *logs* usados para construir o mapa, mais precisa será a informação, exceto em entradas diferentes, mesmo que as rotas tenham tempos muito diferentes. Este

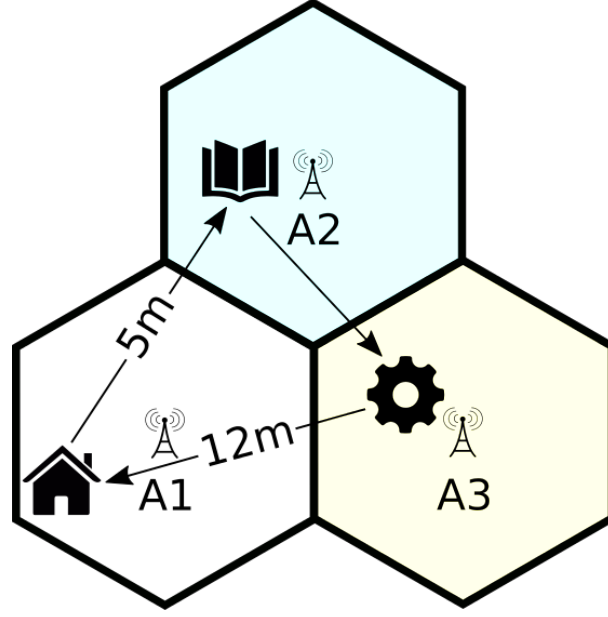


Figura 7 – Pequeno conjunto de viagens de um nó, cada hexágono corresponde a um *marco*: a casa, a escola e o trabalho

é o problema que tentamos resolver através do algoritmo proposto, ToD-PheroCast, que discutiremos na Seção 3.6. Primeiro, vamos explicar como o algoritmo de previsão original funciona.

3.5 Previsão de Posição

A fim de prever a posição futura, tem-se de informar o seu *local* atual (id i e direção d) e por quanto tempo t o nó móvel permanece dentro daquele *local* com aquela direção, bem como especificar para quanto tempo, no futuro h , a previsão deve ser feita. Com esses dados, é preciso responder à seguinte pergunta: quais vértices w são acessíveis a partir do vértice v , $v.id = (i, d)$, com o tempo h , uma vez que t já foi dispendido em v ?

Supondo a sequência que forma um caminho do vértice v_1 para o vértice v_n em um mapa M , representada como $\langle v_1, \dots, v_n \rangle$, possa ser definida como:

$$\square A(\langle v_1, \dots, v_n \rangle) = \sum_{i=1}^n v_i.a; \text{ e}$$

$$\square P_h \text{ como o conjunto de todo } s = \langle v_1, \dots, v_n \rangle \text{ tal que } v_1.id = (i, d) \text{ e } A(s) \leq h + t.$$

Então, um conjunto de vértices alcançáveis a partir de v no tempo h , i.e. a resposta à pergunta indicada no parágrafo anterior é o conjunto $R = \{v_n : \langle v_1, \dots, v_n \rangle \in P_h\}$. O pseudocódigo no Algoritmo 3 descreve uma Busca em Profundidade (DFS) que calcula R .

Algoritmo 3 Previsão da Posição (COELHO et al., 2014)

Require: i, d, t, h
 $R \leftarrow \text{DFS}((V, E), (i, d), t + h)$
function $\text{DFS}((V, E), (i, d), th)$

 Chose $v \in V, v.id = (i, d)$
if $v.a < th$ **then**

 return $\cup_{w \in V, \exists (v, w) \in E} \text{DFS}((V, E), w.id, th - v.a)$
else

 return $\{v\}$
end if
end function

Tão importante quanto determinar R , é determinar a probabilidade de cada vértice v em R ser o nó correspondente ao *local* seguinte a ser visitado, $\mathbf{P}[v]$, que é dada pela equação 1.

$$\forall v \in R, \mathbf{P}[v] = \frac{v.t}{\sum_{w \in R} w.t} \quad (1)$$

Isto é, uma vez que o conjunto de caminhos possíveis é determinado, o campo t é usado para pesá-los uns contra os outros e assim calcular a probabilidade de se usar cada um deles.

Uma vez que a previsão é gerada, pode ser pós-processada de acordo com as necessidades de aplicação. Por exemplo, nós vislumbramos que a previsão possa ser utilizada em uma Rede Tolerante a Atraso (DTN) (FALL, 2003) para decidir para onde enviar mensagens tal que o nó endereçado provavelmente as recolherá. Neste cenário, um roteador de rede pode selecionar os k locais previstos com as maiores probabilidades para encaminhar mensagens; quanto maior o k , melhor cobertura é alcançada, mas máximos são os custos de comunicação, um dilema que a aplicação deve considerar ao utilizar o algoritmo.

3.6 ToD-PheroCast

Como referido na descrição do PheroCast, esse não leva em conta quando (a que horas) a viagem foi feita durante o dia. Tal comportamento pode esconder características importantes dos padrões de mobilidade do usuário e levar a previsões erradas. Por exemplo, considere um nó que ou viaja na manhã de A para B ou viaja a tarde de A para C. Neste contexto, PheroCast provavelmente preverá que sempre que estiver deixando A há uma chance de cinquenta por cento de ir para B ou C. No entanto, se ele presta atenção para o tempo em que a viagem está começando, então poderia decidir se é mais provável que B ou C seja o destino. Cenários como estes são a razão pela qual nós desenvolvemos o ToD-PheroCast.

No ToD-PheroCast, nós dividimos o dia em períodos de atividade. Desta forma, um período de baixa atividade pode começar às 22h e terminar às 6h, quando um período de alta atividade que dura até as 10h começa. Previsões são realizadas primeiramente utilizando a informação do período de tempo e, se não puderem ser feitas, i. e. e não existirem resultados, uma previsão utilizando o grafo igual ao PheroCast original é utilizada.

Existem diferentes formas de armazenar as informações de períodos de tempo. Aqui, nós usamos um mapa para cada período, mais um para a soma de todos. Esta abordagem é de compreensão mais fácil do que armazenar um grafo único, no qual somente um vértice guarda informações para todos os períodos. A desvantagem é o uso do espaço extra, uma análise mais pormenorizada é dada nas Seções 5.4 e 5.5.

Para lidar com períodos do dia, vértices no *Phero Mapa* são estendidos com um campo *time*, o instante em que a entrada correspondente foi feita.

3.6.1 Phero Mapas para ToD-PheroCast

O Algoritmo 4 mostra como os mapas são construídos para a versão melhorada do PheroCast. No algoritmo, $\Pi = (\pi_0, \dots, \pi_{n-1})$ é a sequência de períodos de horas do começo do dia, tal que o i -ésimo período cobre o intervalo $[\pi_i, \pi_{i+1 \bmod(n)}]$

Assumindo que M_i o *Phero Mapa* seja $n = |\Pi|$, cada M_i é referente a um $\pi_i \in \Pi$ e existirá n Phero Mapas. O algoritmo verifica de qual hora do dia o *timestamp* diz respeito, em seguida, verifica qual o Phero Mapa correspondente e o atualiza conforme o Algoritmo 2, utilizando esse mesmo algoritmo. Assim, construímos um *Phero Mapa* geral M_a .

3.6.2 Previsão de Posição para ToD-PheroCast

Uma vez construídos os *Phero Mapas*, para realizar a predição, utilizamos o Algoritmo 5, que funciona semelhantemente ao Algoritmo 3. A diferença é que passamos para o algoritmo o *Phero Mapa* M_i , que corresponde a um dos elementos do conjunto de *Phero Mapas* definidos na Seção anterior. Se o resultado da DFS for vazio, então o algoritmo utiliza M_a para continuar a previsão, como no algoritmo de PheroCast.

3.7 Otimizações na Execução

Para acontecer uma predição qualquer, sem conhecimento prévio no *Phero Mapa* para realizar a DFS, é conveniente que o algoritmo faça uma suposição, já que a predição falhará. As únicas suposições que o algoritmo pode fazer são nós previamente visitados, mas escolher um nó aleatório também levaria, provavelmente, a um erro. A suposição que nos sobra é que o nó permanecerá na mesma posição, então, foi implementado, a partir dessa justificativa, que se nenhuma predição puder ser realizada, o algoritmo supõe que o nó permanece na mesma posição.

Algoritmo 4 Construindo Phero Mapas para ToD-Pherocast.

Let $\Pi = (\pi_0, \dots, \pi_{n-1})$ ▷ ToD periods.
 $\forall i, 0 \leq i < n, M_i = (V_i, E_i) = (\emptyset, \emptyset)$ ▷ ToD Graphs
 $M_a = (V_a, E_a) = (\emptyset, \emptyset)$ ▷ Whole day graph.

for all phero trail $(V_t, E_t) \in T$, with timestamp τ **do**
 for all vertex $v \in V_t$ **do**
 Choose $i : \pi_i \leq v.time < \pi_{i+1} \vee (i = n - 1 \wedge (\pi_i \leq v.time \vee v.time < \pi_0))$
 for all $(V, E) \in \{M_a\} \cup \{M_i\}$ **do** ▷ Update both graphs.
 if $\neg \exists w \in V : w.id = v.id$ **then**
 $V \leftarrow V \cup \{(id = v.id, a = v.p, s = \tau, t = \sigma)\}$
 else
 $w.t \leftarrow \sigma + (1 - \sigma)^{\tau - w.s} * w.t$
 $w.a \leftarrow w.a + \sigma(v.p - w.a)$
 $w.s \leftarrow \tau$
 end if
 end for
end for

for all edge $(f, f') \in E_t$ **do**
 Choose $i : \pi_i \leq v.time < \pi_{i+1} \vee (i = n - 1 \wedge (\pi_i \leq v.time \vee v.time < \pi_0))$
 for all $(V, E) \in \{M_a\} \cup \{M_i\}$ **do** ▷ Update both graphs.
 if $\neg \exists (e, e') \in E : e.id = f.id \wedge e'.id = f'.id$ **then**
 $E \leftarrow E \cup \{(f, f')\}$
 end if
end for

end for

Algoritmo 5 ToD-Pherocast.

Require: i, d, t, h
 Choose $j : \pi_j \leq t < \pi_{j+1} \vee (j = n - 1 \wedge (\pi_j \leq t \vee t < \pi_0))$
 $R \leftarrow \text{DFS}((V_j, E_j), (i, d), t + h)$
if R is empty **then**
 $R \leftarrow \text{DFS}(V_a, E_a, (i, d), t + h)$
end if

function $\text{DFS}((V, E), (i, d), th)$ ▷ Repeated for convenience
 Chose $v \in V, v.id = (i, d)$
 if $v.a < th$ **then**
 return $\cup_{w \in V, \exists (v, w) \in E} \text{DFS}((V, E), w.id, th - v.a)$
 else
 return $\{v\}$
 end if
end function

Materiais e Métodos

Os algoritmos propostos preveem a posição dos nós de uma forma que é indiferente de como a posição em si é definida. Isto é, para os nossos algoritmos, não importa se os *locais* são BSSID de pontos de acesso, nomes de cidades, números de quartos ou coordenadas geográficas grosseiras. Apesar de os algoritmos serem genéricos, a fim de avaliá-los de maneira significativa, fomos restritos a poucos cenários, para os quais existem extensos rastros de mobilidade no mundo real. Nós agora pormenorizamos cada um destes cenários e a maneira como os dados dos rastros foram manipuladas de modo a realizar as avaliações.

4.1 Ônibus de Seattle

Ônibus de transporte público são comuns em cidades ao redor do mundo. Esses ônibus circulam com um horário agendado e, portanto, devem ter padrões de mobilidade muito regulares. Como tal, espera-se que qualquer algoritmo de previsão tenha um bom desempenho na previsão das posições dos ônibus. Assim, este é o primeiro lugar em que nós avaliamos nossos algoritmos de PheroCast. Note que o mesmo cenário descrito aqui também foi utilizado na avaliação do algoritmo PheroCast original (COELHO et al., 2014). A avaliação descrita, no entanto, considera uma versão ligeiramente modificada do PheroCast, discutida no Capítulo 3 e nossa versão melhorada, ToD-PheroCast, apresentada na Seção 3.6.

(JETCHEVA et al., 2003) colheu, durante o mês de Novembro de 2001, os rastros do trânsito dos ônibus de Seattle. Esses rastros indicam quando o registro foi feito, o identificador único do ônibus, o número da rota que o ônibus estava seguindo no momento e sua distância a partir de um ponto de referência. Um pequeno trecho de tal traço é representado na Figura 8.

Como sugerido pelo número de rota no *log* da Figura 8, o mesmo ônibus pode seguir diferentes rotas em momentos variados. Nós aplicamos o algoritmo no amalgamado de ônibus que seguem a mesma rota, para isso, fazemos um pré-processamento dos *logs*,

```

20-11:20:50:06 1008 007 00700572 53313.750000 143677.765625
20-11:20:50:37 1008 007 00700572 52934.000000 144449.687500
20-11:20:51:06 1008 007 00700572 52518.625000 145292.718750
20-11:20:51:37 1008 007 00700572 52136.875000 146063.234375
20-11:20:52:05 1008 007 00700572 51806.125000 146747.406250
20-11:20:52:37 1008 007 00700572 51289.625000 147785.031250

```

Figura 8 – Um excerto do traço do ônibus 1008 seguindo a rota 007. Colunas são data, hora, id, rota, desconhecido e distâncias X e Y de um ponto de referência, em pés (COELHO et al., 2014).

agrupando diferentes ônibus em mesmas rotas. Um pequeno excerto do *Phero Mapa*, construído a partir dos *logs* dos ônibus, pode ser visto na Figura 9.

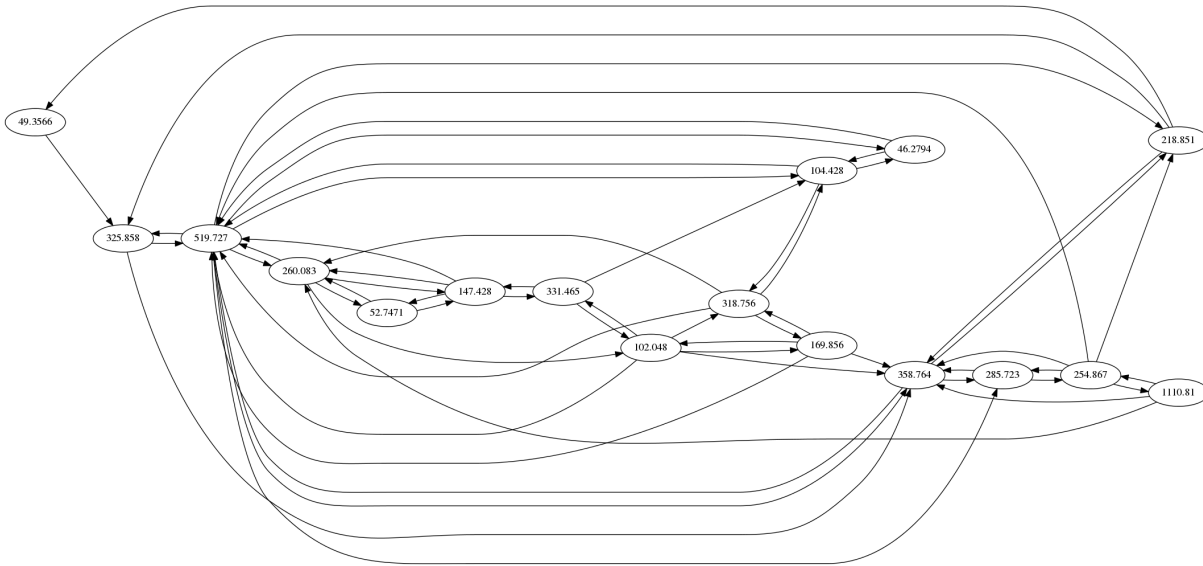
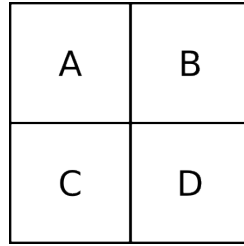


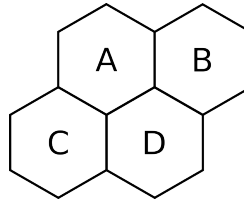
Figura 9 – Uma abstração do Phero Mapa da rota 007, inscrito em cada vértice, mostra o tempo de espera médio em segundos.

4.1.1 Características do Log

Inicialmente, para análise dos algoritmos, dividimos a região de Seattle em uma grande quadrada, tal como apresentado na Figura 10 a, e usamos o algoritmo para prever a célula em que o ônibus estaria. Como se pode ver na Figura 12 a, os ônibus de Seattle se movem em diagonal na maior parte das suas trajetórias. Assim, ao dividir as células em quadrados, existem “saltos” nos rastros. Por exemplo, para se deslocar do ponto D ao ponto A na Figura 10 a, deve-se primeiro ir a um quadrado intermediário (B ou C) e depois para o quadrado desejado. Para reduzir esse comportamento, foi testado também o mesmo *log* com uma malha hexagonal, como pode ser visto na Figura 10 b. Neste caso, existe uma linha na intersecção entre os hexágonos D e A, não havendo hexágono intermediário de um movimento em diagonal. Os resultados foram aproximadamente 10% melhores em todos os cenários testados.



a) Exemplo de uma malha 2x2 quadrada.



b) Exemplo de uma malha 2x2 hexagonal.

Figura 10 – Diferentes formatos de uma célula

Em quase todos os *logs*, existem lacunas entre os dados e erros de GPS. Então, para o algoritmo, o ônibus parece ter viajado para um ponto distante e voltado ao original em um pequeno espaço de tempo. Isto é facilmente corrigido ao estimar a velocidade máxima de um veículo e podando pontos excessivamente distantes no *log*. Outros erros de GPS são difíceis de corrigir; um excerto dos dados do ônibus 007, mostrado na Figura 11, tem um *log* que implica que o ônibus fez 5 km em 38 segundos, i. e. o ônibus se moveu a cerca de 473 km/h, claramente um erro considerando a velocidade dos ônibus desta data em áreas urbanas.

```
19-11:22:35:09 59928.750000 128705.875000
19-11:22:35:50 60017.750000 128051.890625
19-11:22:36:28 53488.250000 143323.484375
19-11:22:36:29 53488.250000 143323.484375
19-11:22:37:08 53479.250000 143341.531250
```

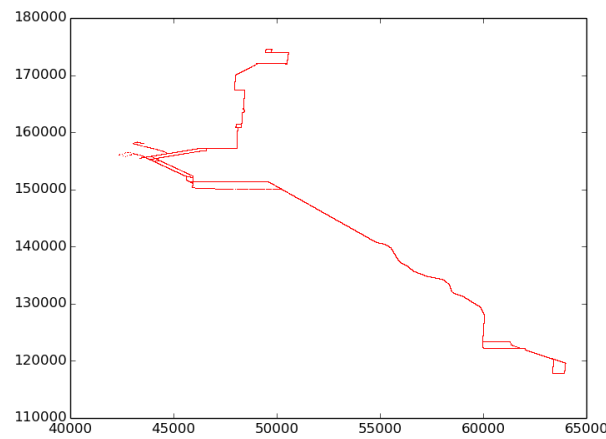
Figura 11 – Excerto do log de um ônibus processado.

4.1.2 Parâmetros de Predição

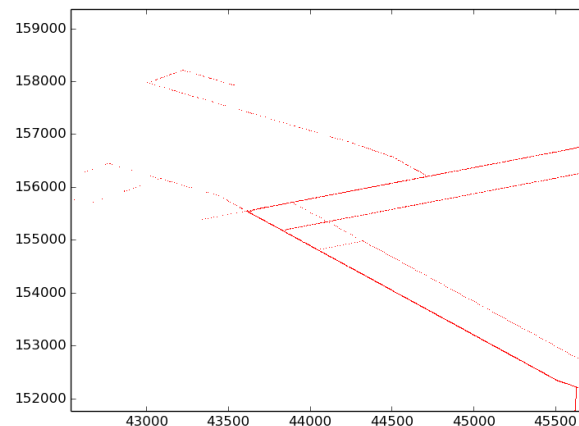
Como pode ser observado na Figura 8, os *logs* foram feitos aproximadamente a cada 30 segundos e, portanto, qualquer previsão dos nossos algoritmos só pode ser feita se respeitado esse horizonte de tempo ou um múltiplo dos mesmos. Além disso, há muitas lacunas nos registros, com entradas consecutivas distantes de alguns minutos ou mais. Tais lacunas tornam, se não impossível, muito difícil de se prever a posição futura.

Durante o experimento, o *log* é processado como se cada entrada tivesse acabado de ser produzida. Apesar de não haver dados sobre os traços informando o intervalo em que

as entradas deveriam ter sido gravadas, identificamos que a maioria das entradas estão a cerca de trinta segundos de diferença da próxima. Para lidar com variações e pequenas falhas no *log*, consideramos qualquer intervalo menor que $m \times 30$ segundos como *uma unidade de tempo*, em que m é um parâmetro de teste, e ignoramos intervalos maiores. Assim, se duas entradas do *log* somam mais do que uma unidade de tempo separadas, então elas não resultam em uma aresta no *Phero Mapa*. Para cada entrada processada, foi utilizado o (ToD-)PheroCast para prever a próxima posição da entrada, se esta for levada em consideração. Desta maneira, ao processar a segunda entrada na Figura 8, estimamos onde o ônibus estaria depois de uma unidade de tempo, cerca de 20:51:07, e verificamos se ele corresponde às coordenadas na terceira entrada do *log*.



a) Todas as entradas de ônibus na rota 007.



b) Zoom do trace

Figura 12 – As entradas mostram um padrão claro de movimentação, como esperado de uma linha de ônibus. Nos eixos x e y, estão representadas respectivamente a longitude e latitude.

As coordenadas dos ônibus no *log* são demasiadamente pequenas para nossos propósitos (1 pé). Nos testes, foram consideradas unidades grosseiras de g pés. Temos o equivalente a uma malha de células quadradas ou hexagonais de g pés² sobrepostas em Seattle. Na Figura 12, por exemplo, foi considerado $g = 300$ pés.

Em resumo, os resultados apresentados são dependentes de quatro parâmetros: k , o número máximo de entradas consideradas a partir da previsão; g , as dimensões de uma célula sobre o mapa; m , o limite para considerar entradas consecutivas no *log*; e σ , o parâmetro para a EWMA.

4.1.3 Validade das Suposições

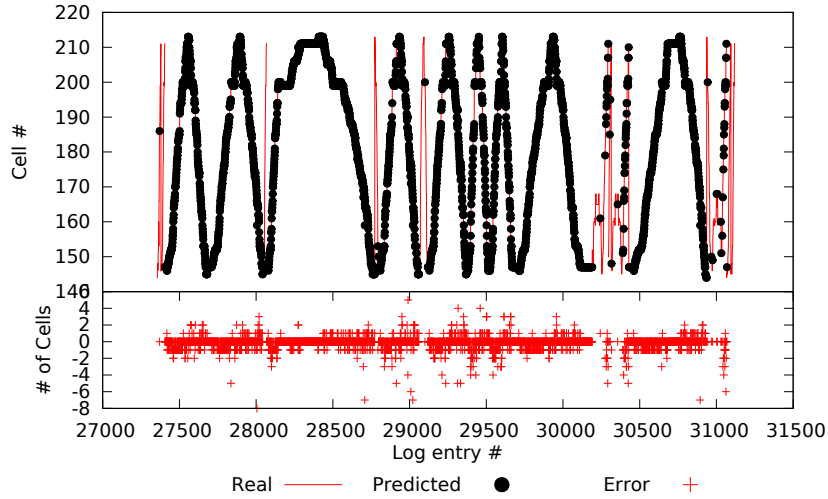
Os resultados que apresentamos aqui são para as 186 viagens ao longo de rota 007, que constituem em 310.206 entradas no *log*. Nós nos concentramos nesta rota, porque é a maior do conjunto de dados dos ônibus, entretanto, os resultados para as outras rotas e rastros são semelhantes. O protótipo utilizado está disponível em <<https://github.com/enriquefynn/pherocast>>, com uma explicação de como os *logs* foram pré-processados.

A partir dos *logs* e da Figura 12(a), é evidente que a suposição de que fizemos com que o nó móvel siga padrões claros é satisfatória, como esperado por ônibus que seguem rotas bem definidas. Um zoom da Figura 12(a) é fornecido na Figura 12(b).

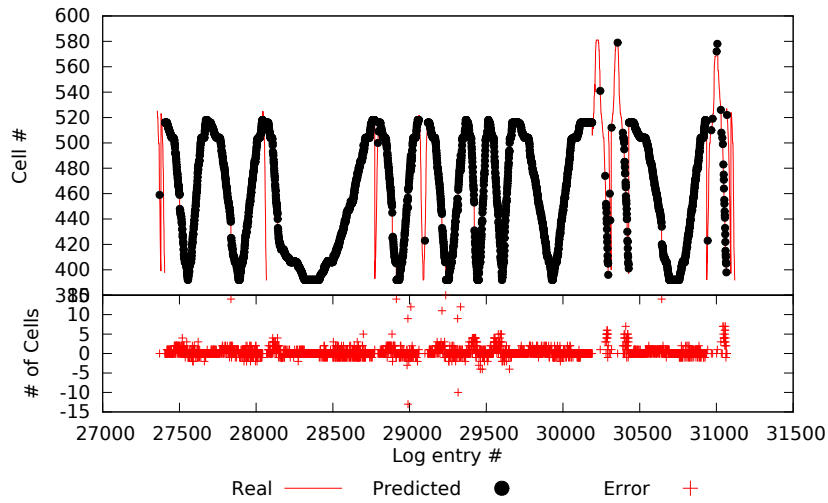
4.1.4 Erro

A fim de medir o erro dos nossos algoritmos, foram consideradas, entre as k previsões selecionadas, a mais próxima da posição real do nó. O erro é a diferença, em número de células, entre a posição real e a posição da previsão.

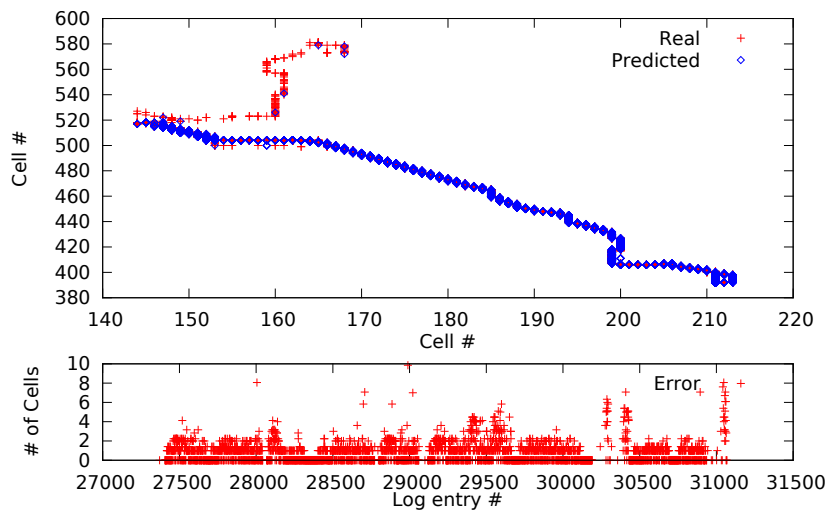
Sendo assim, se considerarmos o cenário com parâmetros $k = 3$, $g = 600$, $m = 2$ e $\sigma = 0,4$. Para tal cenário, a Figura 13(a) mostra a coordenada x de algumas milhares de entradas nos rastros, bem como a melhor previsão. O erro é apresentado na parte inferior da Figura 13(a). Um gráfico semelhante para as coordenadas y e a combinação de coordenadas x e y é apresentado na Figura 13(b) e (c), respectivamente.



a) Erros na coordenada x



b) Erros na coordenada y



c) Erros na coordenada x e y

Figura 13 – Erros nas coordenadas x/y da rota 007 com parâmetros $k = 3$, $g = 600$, $m = 2$ e $\sigma = 0.4$. O erro é mensurado como a diferença, em número de células, entre as previsões e os dados reais.

4.2 Táxis de Roma

Os dados dos táxis de Roma são parte de uma pesquisa feita em (AMICI et al., 2014), na qual a posição de uma frota de 370 táxis foi registrada e mantida à disposição de outros pesquisadores (BRACCIALE et al., 2014). O conjunto de dados contém a localização precisa de cada carro. Os táxis informam sua localização a cada 7 segundos, assim, os dados têm alta granularidade.

A cidade de Roma é um desafio devido à sua malha viária irregular, que difere muito da encontrada na literatura e de congestionamento pesado. Este é o melhor cenário que encontramos devido ao pequeno número de erros em seus *logs* e alta frequência de dados.

Os táxis percorrem um elevado número de caminhos. Como visto na Figura 14, podemos ter uma noção das ruas de Roma a partir de um único táxi. O carro 8 é o que tem mais *logs*, com 27 dias de histórico e 115.983 entradas. Supomos a princípio que a alta movimentação dos táxis prejudicaria significativamente a previsão.

Embora um táxi percorra diferentes caminhos, fica grande parte do tempo parado. Isso possibilita uma previsão mais acurada pelo algoritmo, que, conforme verificado na Figura 15, se comporta relativamente bem. Para o carro 8, a taxa de previsão foi de 70.63%. Levando em conta pequenas células de 100 metros, faremos uma análise mais detalhada no Capítulo 5.

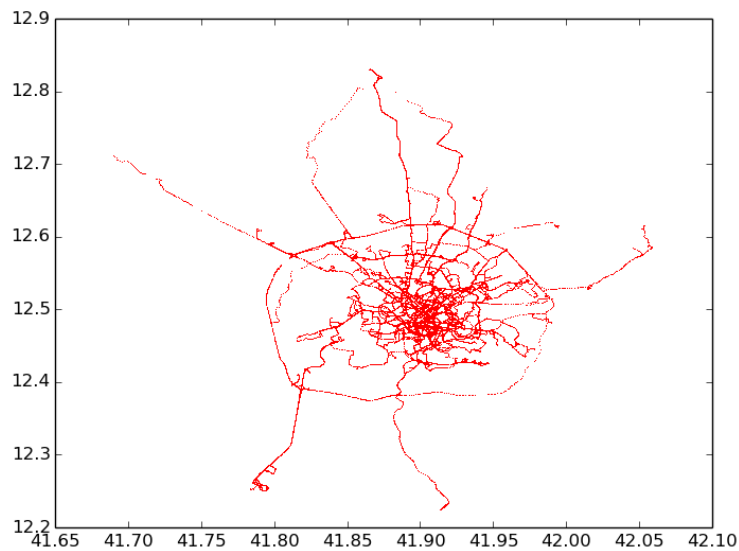
Observamos na Figura 15 que a maior parte das previsões corretas estão localizadas no centro da cidade, onde o táxi faz mais viagens.

4.2.1 Parâmetros de Predição

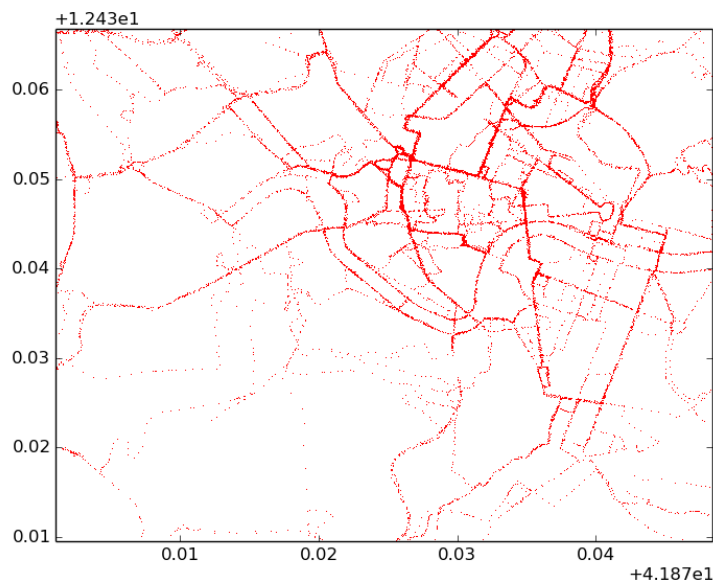
Os parâmetros para o cenários dos táxis de Roma são semelhantes aos vistos na Subseção 4.1.2. As coordenadas são apresentadas em latitude e longitude. Os dados são coletados a cada 15 segundos e não existem muitas lacunas nos dados, que são pré-processados para retirada de dados falhos (AMICI et al., 2014). Os parâmetros para avaliação deste cenário são iguais aos mencionados anteriormente na Subseção 4.1.2.

4.3 Cenário Wi-Fi de Uberlândia

Os telefones móveis são normalmente carregados por indivíduos em suas rotinas, oferecendo uma grande oportunidade para rastrear padrões de mobilidade (ONNELA et al., 2007). Nós desenvolvemos uma aplicação para celulares que tem o sistema operacional *Android* e um serviço na nuvem para rastrear o GPS e *Wi-Fi* dos usuários. O servidor se encontra no Apêndice A. A busca de *Wi-Fi* em telefones é significativamente mais rápida e eficiente do que usando o GPS, pois, através dele, podemos ter uma *posição lógica* baseada nos *ids* de *Wi-Fi* e estimar se o usuário está se movendo ou não, como descrito em (KRUMM; HORVITZ, 2004).



a) Todas as entradas do táxi 8.



b) Zoom do trace

Figura 14 – As entradas mostram os detalhes da cidade de Roma, um padrão diverso, como esperado de um taxi. Nos eixos x e y, estão representadas a longitude e latitude, respectivamente.

Foram observados os dados de 20 usuários do aplicativo, anunciado somente entre colegas e professores da Universidade Federal de Uberlândia (UFU). Os dados foram coletados de forma pseudônima, de modo a não sabermos a identidade real dos usuários.

Na primeira versão do aplicativo, foram coletados dados de GPS com alta precisão, como ilustrado na Figura 16. Contudo, essa coleta não é energeticamente viável para

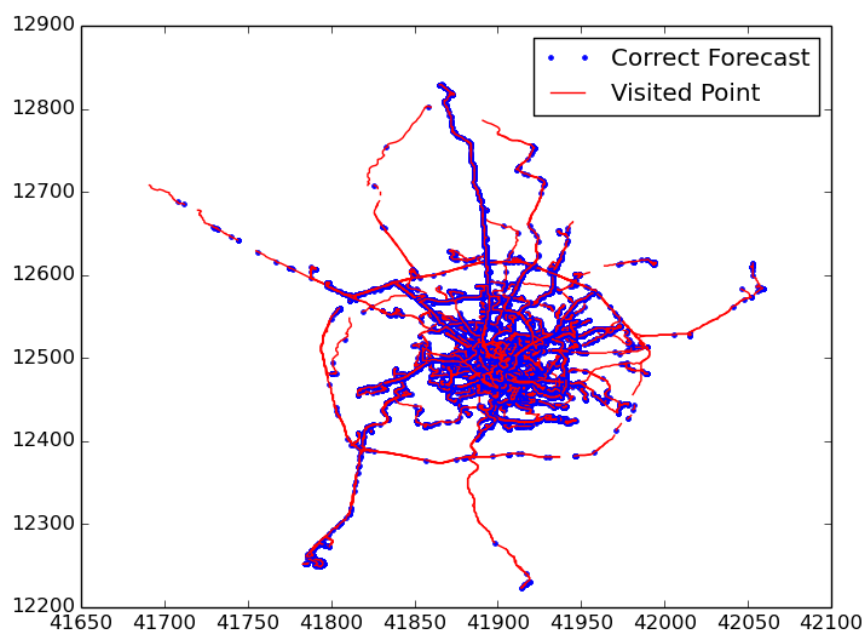


Figura 15 – Previsões para o táxi 8.

um *smartphone* atual e logo foi descartada,, sendo assim, começamos a coletar somente dados de localização grosseiros e informações de redes sem fio, curiosamente, para inferir a posição grosseira o *Google* utiliza as mesmas informações de rede sem fio coletadas pelo

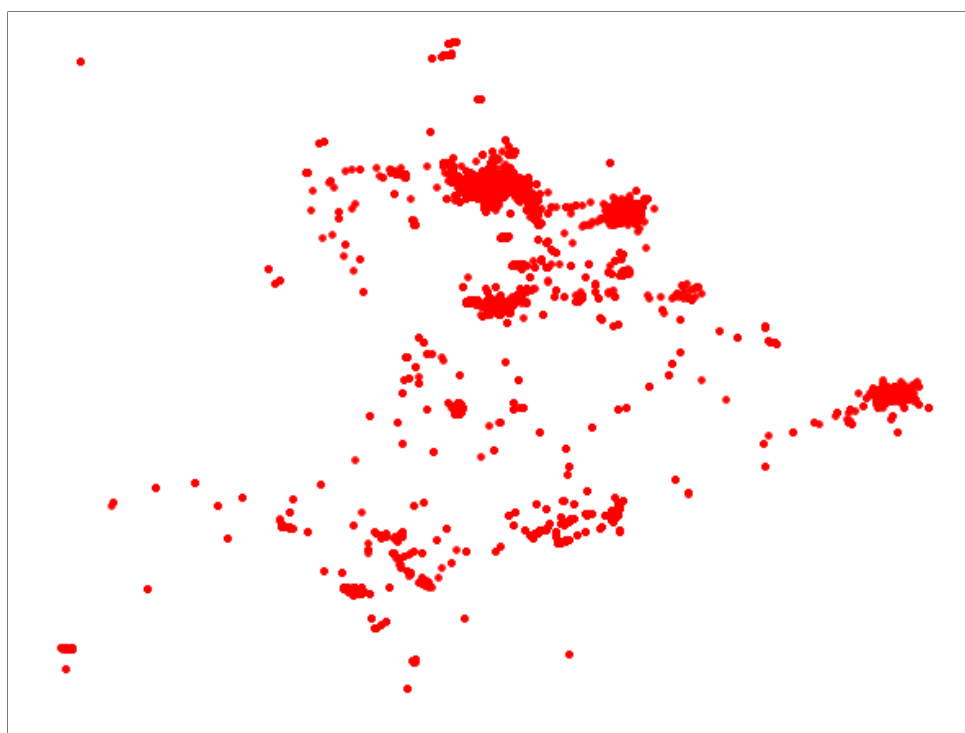


Figura 16 – Dados de GPS de um usuário na cidade de Uberlândia, os dados de longitude e latitude foram omitidos para preservar a anonimidade do usuário.

nosso aplicativo.

As informações de redes sem fio consistem em identificadores únicos, BSSID, com a potência do sinal expressado em decibéis. Um exemplo de *Phero Mapa* é ilustrado na Figura 17, na qual podemos ver como um usuário perambula sobre a influência de diversos roteadores sem fio. É considerado apenas o roteador que tem maior potência em decibéis.

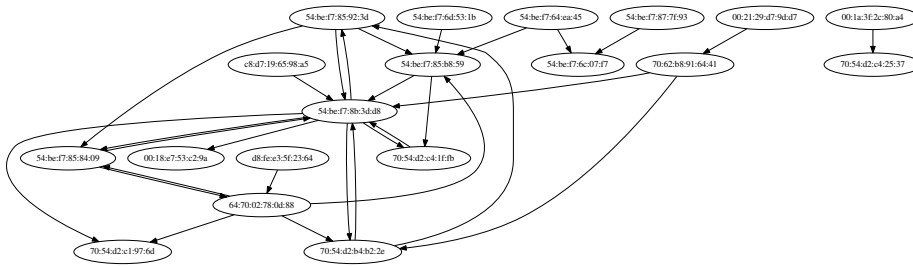


Figura 17 – Exemplo abstrato de um Phero Mapa utilizando BSSID como identificadores.

Avaliação

5.1 Ônibus de Seattle

Para avaliar os nossos métodos, escolhemos utilizar rastros reais obtidos de trabalhos relacionados e um rastro feito por nós mesmos. Para a avaliação, usamos diversos valores de k , g e σ que, respectivamente, significam quantos nós são levados em conta na previsão, tamanho das células e taxa de aperfeiçoamento da EWMA.

Em relação ao ônibus 007, as informações são compiladas na Tabela 2. Na parte superior, apresentamos os resultados ao variar g e fixar k , m e σ . Embora o valor de $k = 3$ pareça arbitrário, ele foi escolhido para refletir as possíveis direções que um ônibus pode seguir a partir de um canto do bloco. Já que o valor de g de 300, 600, ou 1.000 pés correspondem a aproximadamente um, dois e três quarteirões, $k = 3$ deve ser bem adequado nestes casos. Usamos a configuração $g = 1000$ pés como linha de base nas partes média e inferior da tabela.

Os dados mostram que quanto maior o parâmetro g , mais previsões estão corretas.

	$k = 3, g = 1000, m = 2, \sigma = 0.4$	$k = 3, g = 600, m = 2, \sigma = 0.4$	$k = 3, g = 300, m = 2, \sigma = 0.4$
Correto	214102	186700	133258
Incorreto	15101	42503	95945

	$k = 3, g = 1000, m = 2, \sigma = 0.4$	$k = 1, g = 1000, m = 2, \sigma = 0.4$	$k = 5, g = 1000, m = 2, \sigma = 0.4$
Correto	214102	210645	215063
Incorreto	15101	28771	24353

	$k = 3, g = 1000, m = 2, \sigma = 0.4$	$k = 3, g = 1000, m = 1, \sigma = 0.4$	$k = 3, g = 1000, m = 3, \sigma = 0.4$
Correto	214102	87693	253143
Incorreto	15101	3419	20901

	$k = 3, g = 1000, m = 2, \sigma = 0.4$	$k = 3, g = 1000, m = 2, \sigma = 0.9$	$k = 3, g = 1000, m = 2, \sigma = 0.1$
Correto	214102	214181	213999
Incorreto	15101	15022	15204

Tabela 2 – Previsões corretas e incorretas do ônibus 007, para diferentes valores de g , k , m e σ . O cenário $k = 3, g = 1000, m = 2$ e $\sigma = 0.4$ é usado para o patamar.

g	PheroCast	ToD-PheroCast	Diferença
1000 pés	85.40%	87.33%	1.93%
600 pés	64.51%	68.92%	4.41%
300 pés	44.20%	48.26%	4.06%

Tabela 3 – Porcentagem de acerto médio dos ônibus, variando o g .

No entanto, uma série de previsões ainda estão fora, em média, cerca de 0.320892 células, mesmo quando cada célula é de 1000 pés. Acreditamos que isso aconteça devido à granularidade temporal do *log*; 30 segundos pode representar uma grande variedade de deslocamentos, de acordo com a hora do dia, o tráfego e eventos, tais como a carga e descarga de cadeiras de rodas, o que levará a cada nó no *Phero Mapa* a ter informações equivocadas. Esta suspeita é confirmada pelos dados na segunda parte da Tabela 2, na qual variamos o parâmetro k . Quanto maior o k , mais alternativas na previsão são levadas em conta e, portanto, menor é o número de erros.

O parâmetro m desempenha um papel importante no algoritmo, permitindo que dados ruins possam ser ignorados e fazendo com que o algoritmo não arrisque uma previsão. Esse fato é visto na terceira parte da tabela; com $m = 1$, o algoritmo prevê corretamente quase um terço do que preveria com $m = 2$, mas faz apenas um quinto dos erros, alcançando uma precisão de 96.25%. Dependendo da aplicação, pode-se esperar que PheroCast arrisque e, por conseguinte, m deve ser grande. Esperamos que, no entanto, combinado com um g grande, a taxa de sucesso seja muito baixa, pois o *Phero Mapa* será preenchido com nós para a mesma rota, porém não ligados.

Finalmente, a parte inferior da tabela mostra o efeito de diferentes σ . Uma vez que as rotas dos ônibus são muito estáveis, o valor de σ teve pouca influência nos resultados do algoritmo. Testes adicionais são necessários para melhor realçar os efeitos da variação de σ .

Mesmo que todas as configurações levem a erros na previsão, a sua distribuição, apresentada na Figura 18, mostra que a maioria são em torno de uma ou duas células e muito poucas estão acima de cinco células. Em uma rede tolerante à atraso, por exemplo, se o erro é para uma coordenada que ainda vai ser atingida, mesmo se não dentro do horizonte de tempo esperado, a capacidade da rede para entrega de mensagens pode não ser afetada.

Na Tabela 3, vemos como a porcentagem de acerto varia conforme mudamos o tamanho de cada célula g . Os outros parâmetros foram fixos como os da Tabela 2, $k = 3$, $m = 2$ e $\sigma = 0.4$. Para o ToD-PheroCast, utilizamos $\Pi = (0, 4, 8, 11, 14, 17, 19, 21, 23)$. A diferença cai muito quando a previsão se torna muito exata, um indício de que estamos chegando no limite da previsão.

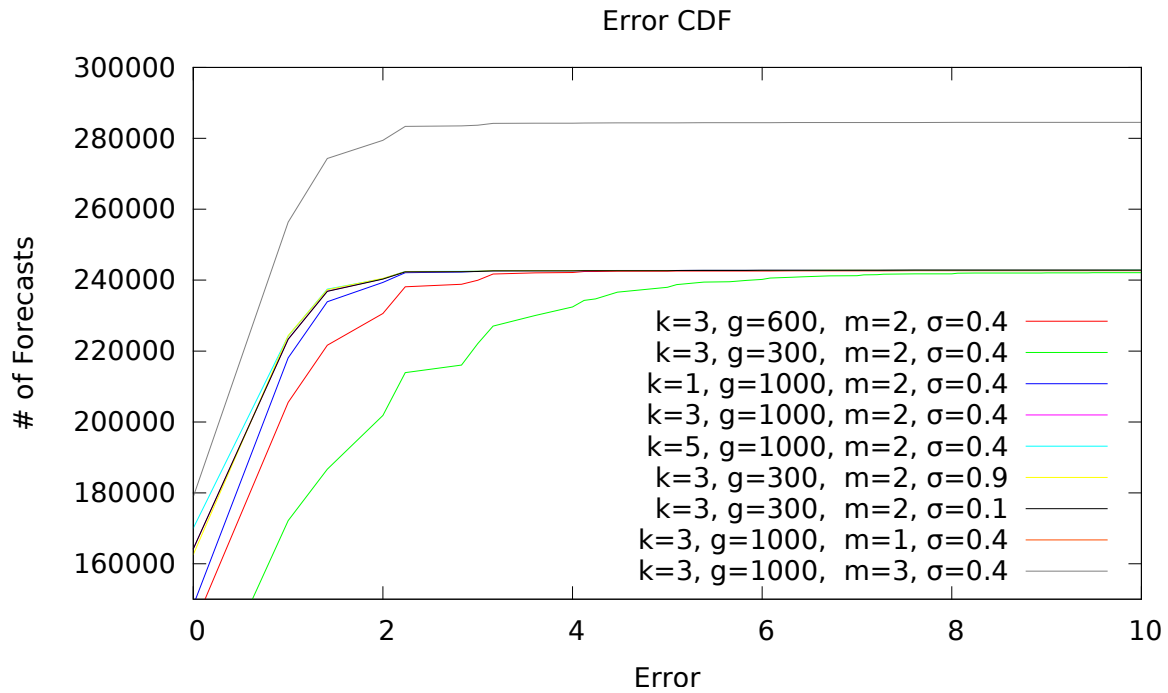


Figura 18 – CDF para erros na previsão com diversas configurações de g, k, m e σ .

	$k = 3, g = 100, m = 5, \sigma = 0.4$	$k = 3, g = 200, m = 2, \sigma = 0.4$	$k = 3, g = 300, m = 2, \sigma = 0.4$
Correto	81573	94426	100730
Incorreto	34537	21684	15380

	$k = 3, g = 100, m = 5, \sigma = 0.4$	$k = 1, g = 100, m = 2, \sigma = 0.4$	$k = 5, g = 100, m = 2, \sigma = 0.4$
Correto	81573	79406	82454
Incorreto	34537	36704	33656

	$k = 3, g = 100, m = 5, \sigma = 0.4$	$k = 3, g = 100, m = 1, \sigma = 0.4$	$k = 3, g = 100, m = 10, \sigma = 0.4$
Correto	81573	79977	81570
Incorreto	34537	33192	34558

	$k = 3, g = 100, m = 5, \sigma = 0.4$	$k = 3, g = 100, m = 2, \sigma = 0.9$	$k = 3, g = 100, m = 2, \sigma = 0.1$
Correto	81573	82680	80700
Incorreto	34537	33430	35410

Tabela 4 – Previsões corretas e incorretas do táxi 8 para diferentes valores de g, k, m e σ . O cenário $k = 3, g = 100, m = 5$ e $\sigma = 0.4$ é usado para o patamar.

5.2 Táxis de Roma

Utilizamos uma configuração parecida para avaliar o cenário dos táxis em Roma. Devido à alta granularidade dos *logs*, escolhemos os valores de $k = 3, m = 2, g = 100m$ e $\sigma = 0.4$ como padrão. Na Tabela 4, vemos como se comporta o táxi 8, do mesmo modo como feito na Tabela 2; os parâmetros de previsão têm as mesmas correlações vistas na Seção 5.1

Na Figura 19, vemos a distribuição de previsões corretas do táxi 8 através do mapa. No mapa de calor, cores mais próximas ao vermelho e do verde indicam, respectivamente,

g	PheroCast	ToD-PheroCast	Diferença
300m	88.45%	89.17%	0.72%
200m	83.87%	84.75%	0.88%
100m	74.58%	75.30%	0.72%

Tabela 5 – Porcentagem de acerto médio dos táxis, variando o g .

menos acertos e mais acertos. Nota-se que a previsão do táxi é melhor no centro de Roma, onde o carro viaja mais. Confirmando a nossa suposição, vemos na Figura 20 o mapa de calor do acerto de todos os táxis do conjunto. A cor varia de forma logarítmica em ambas Figuras 19 e 20.

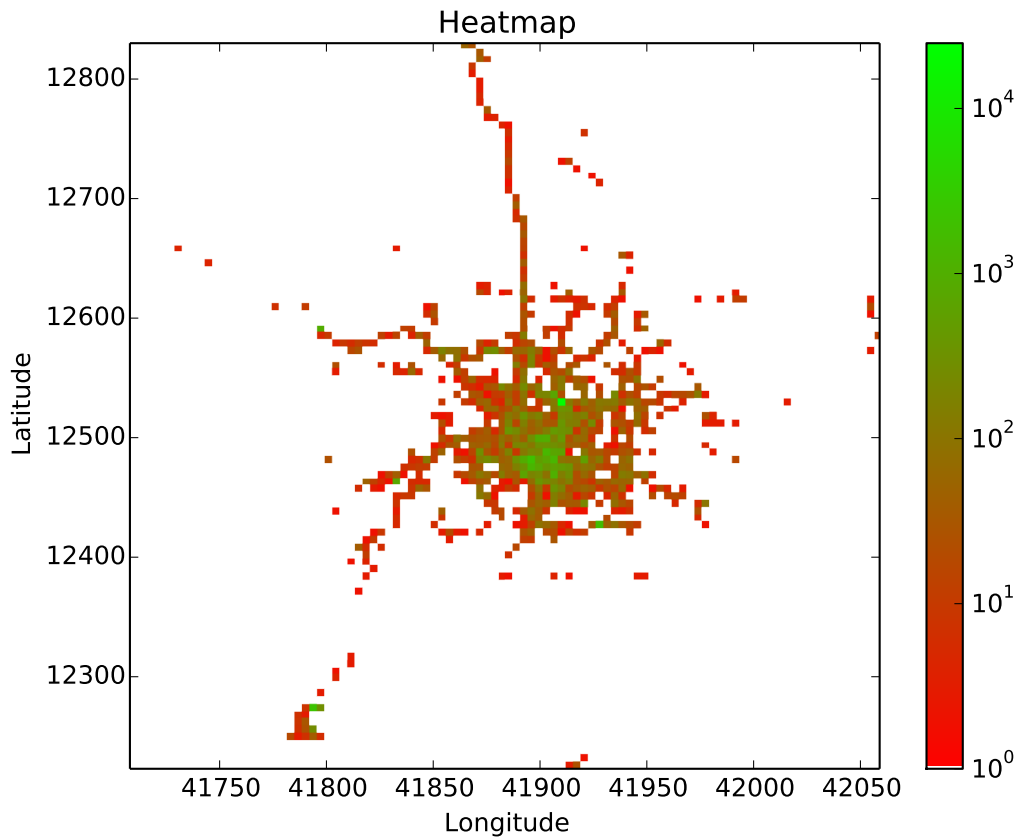


Figura 19 – Número de entradas corretas do taxi 8

Na Tabela 5, temos uma comparação dos dois algoritmos, utilizando como parâmetro $k = 3, m = 2$ e $\sigma = 0.4$. Nos testes do ToD-PheroCast, $\Pi = (0, 4, 8, 11, 14, 17, 19, 21, 23)$, vemos um aumento pouco significativo quando utilizamos o ToD-PheroCast, provavelmente porque os dados já estão próximos ao limite teórico do que o algoritmo pode alcançar, como observado no cenário dos Ônibus de Seattle, quanto menor a previsão, mais o ToD-PheroCast funciona melhor que o PheroCast.

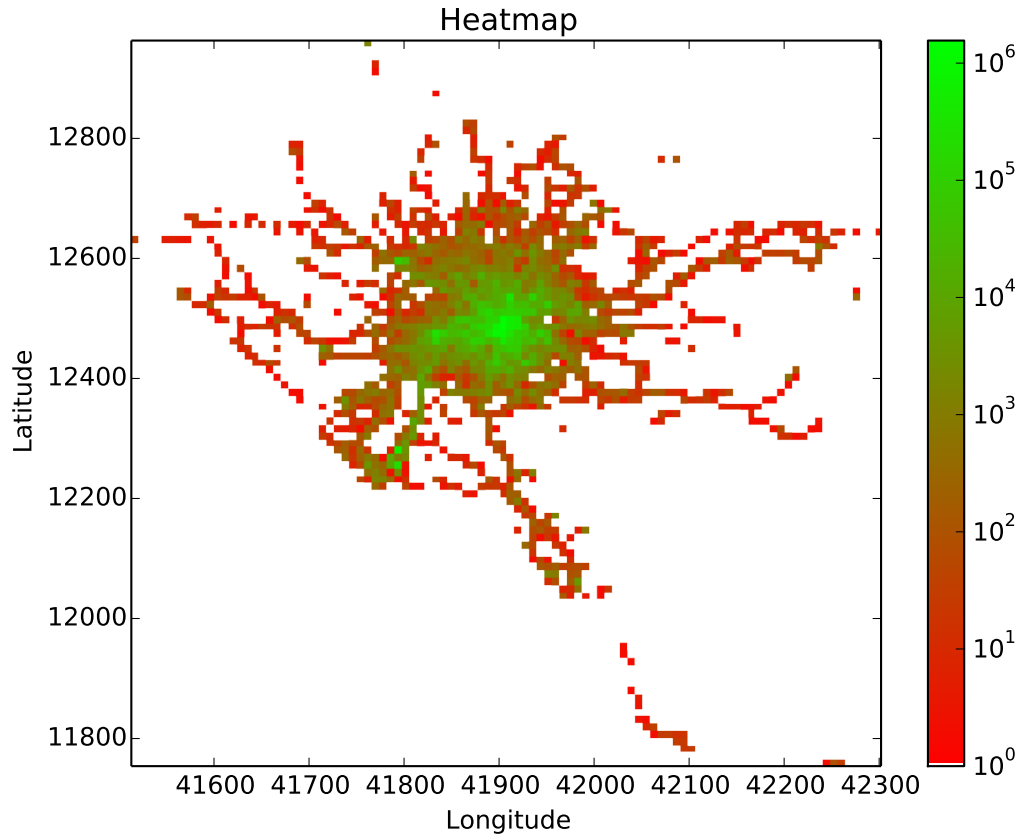


Figura 20 – Número de entradas corretas para todos os taxis

5.3 Cenário Wi-Fi de Uberlândia

Em um cenário real, indivíduos têm a tendência de formar pontos que são mais visitados que outros (GONZALEZ; HIDALGO; BARABASI, 2008), como casa e trabalho. Na Figura 21, vemos todos os pontos coletados de dois indivíduos, observando conglomerados de pontos com possíveis locais de interesses, destacados nas setas de nome *Ponto 1* e *Ponto 2*.

Testamos o ToD-PheroCast no cenário de *Wi-Fi* de Uberlândia, descrito na Seção 4.3. O algoritmo foi testado em 20 usuários, em média, ele conseguiu um acerto de 81.02%, considerando apenas previsões para a rede sem fio mais próxima.

Assim, não existe g e os dados possuem uma consistência melhor do que os outros cenários, embora se o indivíduo deixa, por exemplo, o celular em casa um dia, os dados daquele dia serão completos, porém inconsistentes, pois não expressam a mobilidade do indivíduo.

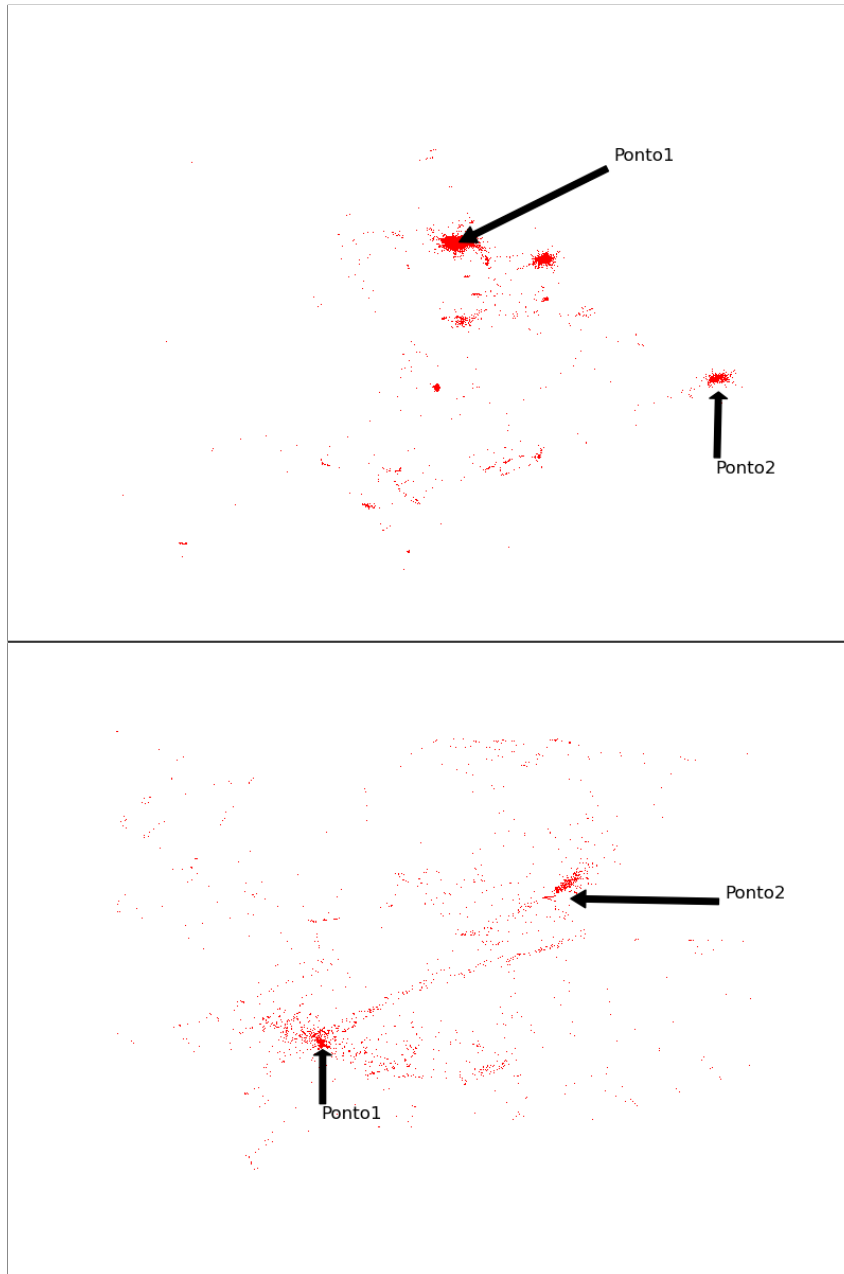


Figura 21 – Possíveis locais de interesse de dois indivíduos distintos.

5.4 Análise de Espaço

Conforme os nós se movem, seus *Phero Mapas* aumentam de tamanho. Nós organizamos a estrutura do grafo de tal forma a manter um baixo custo em tamanho, sendo que cada *local* tem 56 bytes. Quando adicionamos um *local* ao *Phero Mapa* nós adicionamos somente um ponteiro, caso este *local* já tenha sido visitado uma vez, consumindo apenas 8 bytes.

Nas Figuras 22 e 23, temos no eixo y o período das previsões tal que a primeira previsão tem um período $p = 0$, a previsão subsequente tem período $p = p + 1$, o tamanho do *Phero Mapa* desse período (na escala a esquerda) é a média de todos tamanhos de

Phero Mapa do período p dos n carros avaliados (na escala a direita).

Podemos assumir que existe um número finito de pontos no mapa, mesmo que grande. Quando um nó visita todos os *locais*, ele deixará de crescer na razão de 56 bytes e passará a crescer somente na razão de 8 bytes. Eventualmente, quando todas as rotas tiverem sido exploradas, o grafo não crescerá mais. Não é praticável que um nó visite todos os *locais* no mapa, isto é observado empiricamente, como mostra a Figura 22 e 23. Como os *logs* dos veículos são diferentes em tamanho, i. e. alguns têm mais entradas que outros; nós enfatizamos a relação entre o número de veículos e o tamanho médio dos *Phero Mapas* para o período corrente, visto que não podemos fazer conclusões gerais a partir de um subconjunto pequeno de veículos. O baixo desvio padrão mostra que o *Phero Mapa* para cada indivíduo não difere muito da média.

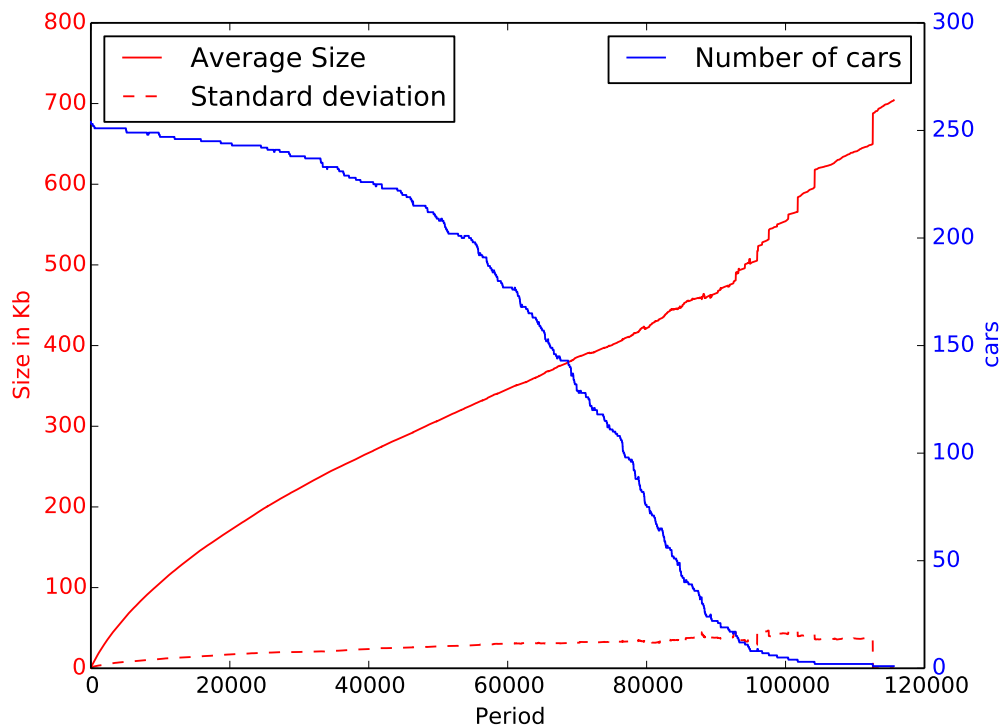


Figura 22 – Espaço utilizado pelo grafo, número de veículos e desvio padrão, variando através do tempo para os logs de taxis em Roma.

5.5 Análise do Tempo de Predição

A análise do tempo gasto para realizar uma predição é difícil devido às características específicas do computador em que a análise foi feita. A mesma análise pode variar bastante de uma configuração de aparelho para outra.

Devido ao tempo para se realizar uma predição ser muito baixo (na ordem de milissegundos), a comutação entre o *kernel* e outros programas do espaço do usuário acabam

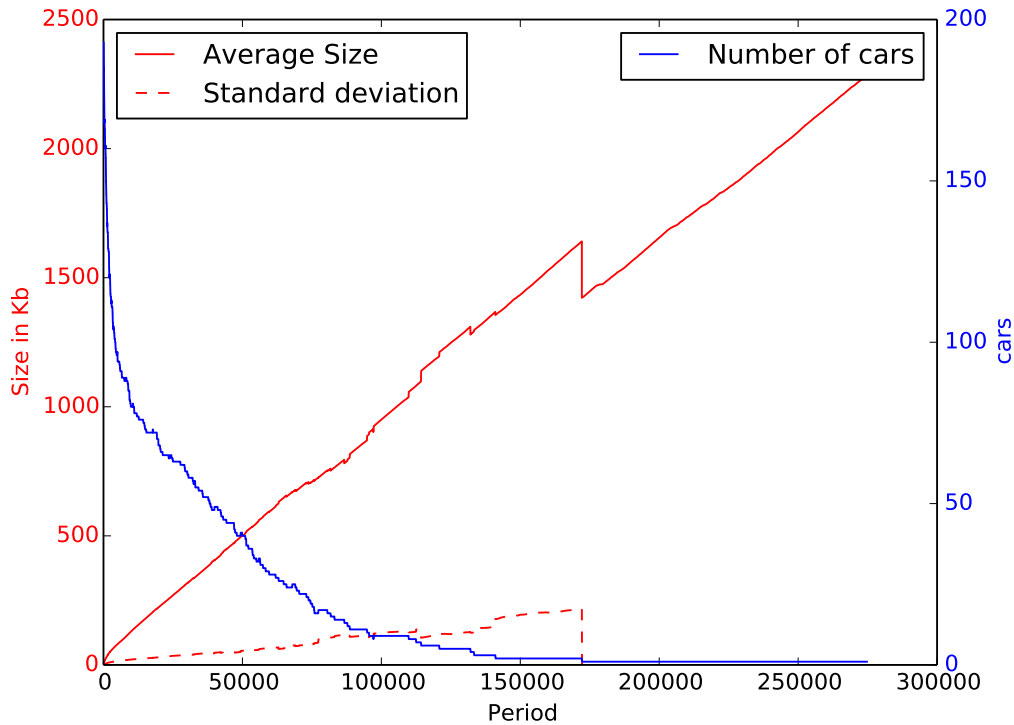


Figura 23 – Espaço utilizado pelo grafo, número de veículos e desvio padrão, variando através do tempo para os logs de ônibus de Seattle.

influenciando mais a predição do que a busca no *Phero Mapa*. Essa busca é o principal gargalo do algoritmo. Considera-se que no horizonte de tempo h existam V vértices alcançáveis, o algoritmo tem complexidade $\mathcal{O}(V)$, quanto maior o horizonte de tempo, mais vértices são alcançáveis.

Nas Figuras 24 e 25, temos no eixo y o período das previsões tal que a primeira previsão tem um período $p = 0$, a previsão subsequente tem período $p = p + 1$, o tempo de previsão desse período (na escala a esquerda) é a média de todas as entradas do período p dos n carros avaliados (na escala a direita).

Como pode ser visto na Figura 24, a velocidade de uma predição varia de 0 ms a pouco mais de 1000 ms, com alguns pontos isolados no final, devido à baixa quantidade de carros. Na Figura 25, temos um cenário melhor, em que o tempo necessário para prever foram todos inferiores a 250 ms.

Nós identificamos um desvio padrão esperado e grande, porque o tempo de prever um nó individual pode ser influenciado por outros programas, incluindo o *kernel*. É importante ressaltar que em um aparelho projetado especificamente para os algoritmos, o tempo seria bem melhor.

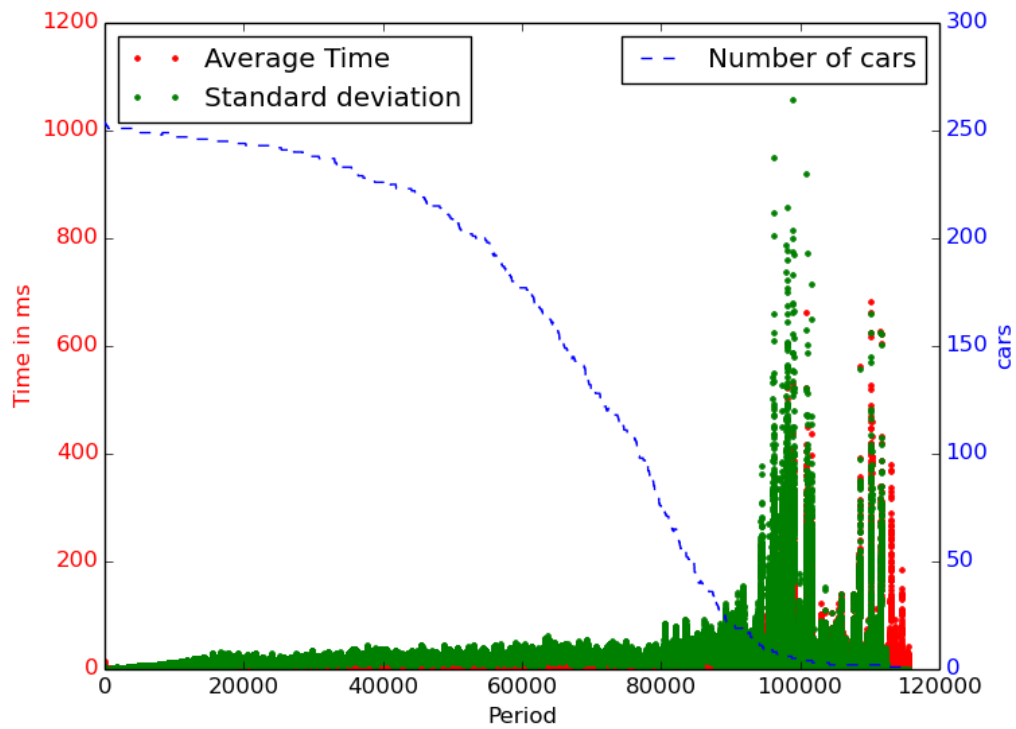


Figura 24 – Tempo levado para prever, número de carros e desvio padrão, variando conforme o tempo nos logs de taxis de Roma.

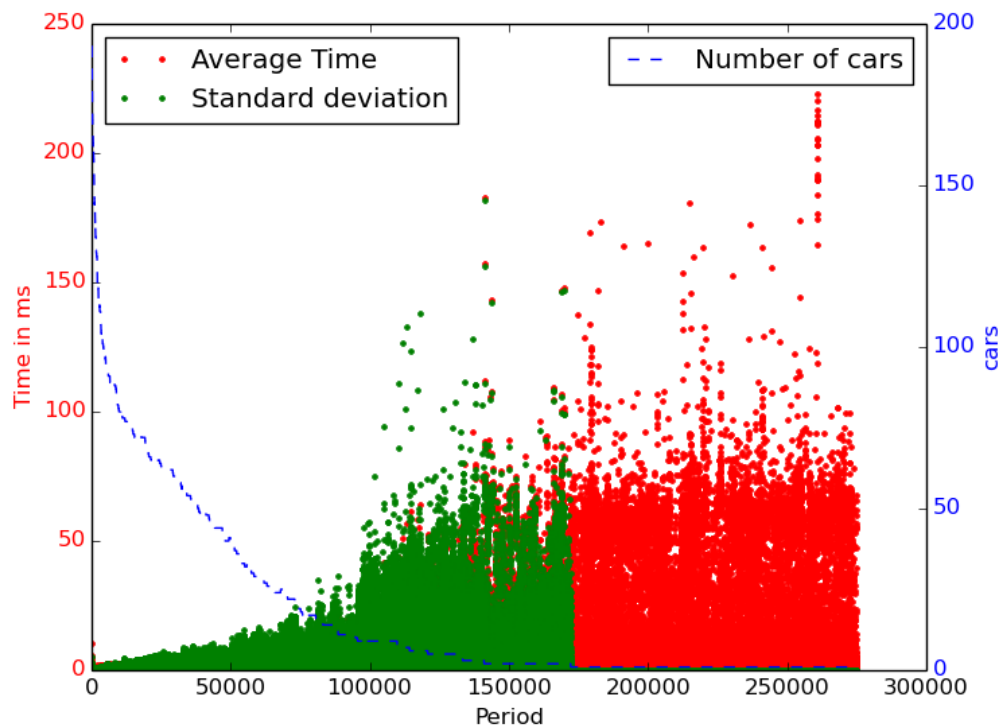


Figura 25 – Tempo levado para prever, número de carros e desvio padrão, variando conforme o tempo nos logs de ônibus de Seattle.

Conclusões e Trabalhos Futuros

Neste trabalho, apresentamos dois algoritmos para prever a localização de nós em redes móveis. Os algoritmos são baseados em uma representação compacta do histórico recente da mobilidade dos nós e inspirado pela abstração de colônia de formigas. Descrevemos também alguns experimentos e mostramos que o algoritmo funciona bem na previsão do local dos ônibus, táxis e pessoas em áreas metropolitanas.

Retomando as propriedades de um bom algoritmo de previsão apresentado na introdução deste trabalho, o algoritmo de previsão necessita das seguintes propriedades:

- ❑ Rápido: Realizar previsões em um tempo hábil para que aplicativos sejam capazes de tomar decisões.
- ❑ Leve: Capaz de funcionar em sistemas embarcados.
- ❑ Online: Realizar previsões à medida que os dados são lidos.
- ❑ Acurado: Realizar previsões acertadas.
- ❑ Descentralizado: A previsão acontece no nó de forma independente de outros nós ou infraestrutura.
- ❑ Não invasivo: Consultar o mínimo de informações possíveis, mantendo informações sensíveis privadas.

Rápido

Os algoritmos apresentados são rápidos, pois cada predição demora, no máximo, alguns segundos para ser executada em nossos testes e, pode-se garantir também um tempo máximo configurável, tornando hábil para a maioria das aplicações tomarem uma decisão. Esse tempo pode ser otimizado se utilizado um *garbage collector* discutido na seção 6.1. Além disso, os algoritmos são precisos, atingindo, nos cenários testados, uma eficácia de mais de 80%.

Leve

Como visto na Seção 5.4, o *Phero Mapa*, única estrutura de dados cujo tamanho se modifica durante a execução do algoritmo, requer no máximo 3Mb nos testes para funcionar corretamente. Este é um tamanho pequeno em comparação a aplicativos embarcados atualmente em celulares, essa estrutura pode ainda ser compactada reduzindo ainda mais o seu tamanho.

Online

O fato do algoritmo ser rápido de ser executado e esse mesmo tempo de execução ser configurável, faz possível o algoritmo ser *online*, assim as previsões são executadas em um tempo máximo, executando previsões enquanto entrega as já solicitadas.

Descentralizado

Nossos algoritmos foram feitos para funcionar de forma descentralizada, independentemente de infraestrutura externa. São necessários poucos dados para a realização de previsões, que acontecem *online* com a coleta de dados, isto é, os dados são processados a medida que coletados, levando a previsões em poucos milissegundos. Finalmente, os algoritmos não são invasivos, pois não precisam compartilhar informação dos usuários com nenhum agente externo ou infraestrutura.

Não invasivo

O algoritmo proposto considera poucas informações para realizar as previsões e como as informações nunca saíam do dispositivo do usuário, informações sensíveis como posição geográfica podem ser criptografadas e só acessíveis à aplicação.

Nossa motivação mais forte para a invenção de PheroCast e ToD-Pherocast foi a necessidade de desenvolvimento de novos mecanismos de roteamento para MANET, em especial para VANET. Especificamente, estamos interessados em adaptar de forma proativa a infraestrutura de rede usando SDN (MCKEOWN, 2009) e permitindo DTN (FALL, 2003) com melhor Qualidade de Serviço (QoS).

Embora a aplicação para um só indivíduo seja interessante, por ser praticável em um cenário real e de forma não invasiva, as aplicações envolvendo múltiplos indivíduos têm o poder de revolucionar o mundo em que vivemos hoje. Vários problemas surgem quando tentamos unificar as previsões ou realizar rotas colaborativas, como confiar na identidade dos indivíduos. Algumas destas possibilidades são discutidas ainda neste capítulo.

Atualmente, estamos trabalhando em várias melhorias para o PheroCast. Por exemplo, estamos estendendo a avaliação do algoritmo em cenários de mobilidade emulados

utilizando o Simulação de Mobilidade Urbana (SUMO) (BEHRISCH et al., 2011). Esperamos que em tal cenário controlado sejamos capazes de medir melhor os efeitos do envelhecimento da estratégia adotada.

6.1 Trabalhos Futuros

Nos próximos passos, planejamos melhorar os algoritmos, incluindo estatísticas de desvio padrão em nossa estrutura de dados do *Phero Mapa*, o que permitiria pesquisas não para um simples conjunto de *locais* que podem ser alcançados, mas por um conjunto de intervalos ao longo dos possíveis percursos. Por exemplo, será possível pesquisar para todos os *locais* alcançáveis em 10 segundos mais ou menos um desvio padrão a partir da localização esperada.

Em (GONZALEZ; HIDALGO; BARABASI, 2008), focou-se em previsões para futuro próximo. Uma questão interessante é fazer previsões para futuros mais distantes, como várias horas ou dias adiante no tempo. Em trabalhos vindouros, pretendemos explorar métodos para usar a informação na forma de *Phero Mapas* para a predição para tais limiares distantes de tempo.

Planejamos investigar como levar em conta mais contextos temporais em que as viagens acontecem, e.g., o dia da semana (DASH et al., 2015), bem como outras informações de contexto. Neste sentido, vamos investigar o uso e a análise de séries temporais mais complexas para determinar se elas podem obter melhores resultados do que a nossa abordagem atual e manter o algoritmo leve o suficiente para ser executado em um sistema *on-board* no carro ou em um *smartphone*.

O *Phero Mapa* não preserva a noção de que, normalmente, diferentes origens levam a destinos distintos, mesmo que as rotas compartilhem alguns *locais*. Assim, um nó tem dois padrões de rota, sendo o primeiro indo do *local* A para o C, passando por B, e o outro indo de A' para C' também passando por B. Ao prever a posição do nó enquanto no local B, o mapa proposto não provê quaisquer meios para diferenciar viagens que começam em A daquelas que começam em A', o que seria essencial para melhorar as previsões.

Finalmente, os dados coletados a partir dos *smartphones* de indivíduos da cidade de Uberlândia serão disponibilizados, da mesma forma que (JETCHEVA et al., 2003) e (BRACCIALE et al., 2014) no *CrawDad* para que a comunidade se beneficie desses dados.

Para manter um baixo *footprint*, um procedimento de *garbage collection* pode ser executado em paralelo com a construção, removendo nós com *timestamp* que são muito antigos do mapa. O processo pode ser otimizado selecionando candidatos para o *garbage collector* durante as pesquisas de previsão.

Como uma otimização final, as diferentes potências de $1-\sigma$, necessárias nos cálculos da EWMA, são pré-calculadas e armazenadas em uma tabela de referência rápida. Valores

menores que um limiar pré-definido são consideradas zero, descartando eficazmente a informação que é muito antiga.

6.2 Aplicações

Há muitos cenários diferentes em que o PheroCast ou alguma variação dele pode ser útil. Vamos discutir aqui alguns exemplos.

Rotas de GPS pré-calculadas

Se PheroCast estiver incorporado em um sistema veicular, como um computador de bordo equipado com GPS, então, sempre que o GPS é utilizado para calcular uma rota, ela pode ser utilizada pelo PheroCast para melhorar as previsões. Com base em nossa própria experiência, conjecturamos que isto é válido para os outros condutores, sempre que dirigindo em uma área bem conhecida, as sugestões do GPS importarão muito pouco. Em áreas menos conhecidas, no entanto, a sugestão do GPS provavelmente serão seguidas. Assim, PheroCast, pode considerar a sugestão de GPS como uma trilha ordinária, que terá pouco peso nas áreas conhecidas, mas será muito importante para complementar a pouca informação no mapa das zonas menos conhecidas.

Roteamento

Como referido anteriormente, estamos interessados em usar o PheroCast na otimização de roteamento de VANET. Uma vez que um nó sabe onde está indo, este pode anexar tais informações para dados de saída de modo que eles sejam utilizados pela rede para rotear respostas de volta para o nó. Desta forma, a previsão pode ser usada para criar fluxos em OpenFlow (MCKEOWN et al., 2008) ou reservar canais no Multiprotocolo de Comutação de Etiqueta (MPLS)(ROSEN; VISWANATHAN; CALLON, 2001).

Com um melhor roteamento, todas as aplicações de VANET melhoram, pode-se fazer *streaming* de áudio/vídeo com veículos ao redor, que viajam para uma mesma localidade, espalhando informações necessárias para reduzir o número de acidentes. Se um acidente for presenciado, com um clique, pode-se alertar todos os condutores em volta assim, bem como os serviços de emergência e a ambulância, por exemplo, pode ir mais bem preparada se souber a priori de informações, como quantidade de feridos, tipo de acidente ou condições de acesso ao local.

Planejamento de Tráfego em Cidades Inteligentes

Em uma cidade inteligente, os nós poderiam informar periodicamente suas posições e previsões para alguma entidade comum. Desta forma, a entidade pode então correlacionar

	Nó 2 Acidenta	Nó 2 desvia
Nó 1 Acidenta	-10/-10	-10/0
Nó 1 desvia	0/-10	-5/-5

Tabela 6 – Estratégias de desviar ou não de um obstáculo, o estado de Equilíbrio de Nash, onde os dois nós desviam, em negrito.

a informação para estimar e prever o tráfego. Isso permitiria detecta e prevenir engarrafamentos, entre outras ações. Além disso, se os nós estão tomando rotas incomuns, outros nós, com padrões de mobilidade semelhantes, podem ser aconselhados a tomar desvios antes que eles atinjam a área com anomalia.

O mesmo mecanismo pode ser usado em áreas não previamente (ou recentemente) visitadas pelo nó, ou seja, as previsões podem ser realizadas com base nos padrões de mobilidade média da área.

Veículos Prioritários

Ao saber e atestar a rota de um veículo prioritário, como uma ambulância, o algoritmo pode refazer a rota do nó, a fim de deixar o veículo com mais prioridade em uma rota de maior velocidade. Os sinais podem ser retransmitidos para coordenar uma sequência de nós, mantendo sempre a velocidade da ambulância alta e constante, sem muita influência à velocidade média de cada nó.

Luzes de Freio Eletrônicas

Faz-se difícil tornar os sistemas veiculares completamente autônomos. Vamos supor que exista um buraco em uma pista congestionada e um nó, que não tinha conhecimento desse buraco, passa agora por ele. A rede pode realizar o consenso de que este nó deva passar no buraco, acidentando-se sozinho para não prejudicar os demais nós da VANET. A decisão, embora boa globalmente, é ruim para o nó que se acidenta de fato. Se modelarmos a situação como um jogo com 2 nós, existe somente um estado com o Equilíbrio de Nash¹, o que ambos tentam desviar do obstáculo, como visto na Tabela 6.

Veículos podem saber se outro veículo adiante freiou mesmo sem vê-lo, o que permite ao automóvel (autômoto ou não) reagir em caso de uma freiada brusca, como ocorre em acidentes, conforme o exemplo mencionado anteriormente, o condutor pode colocar outros veículos em perigo. Ao entender padrões de posicionamento, podemos antever caminhos de outros veículos e prepará-los com rotas para minimizar os danos do acidente.

¹ Uma situação em que, em um jogo não cooperativo envolvendo dois ou mais jogadores, nenhum jogador tem a ganhar, mudando sua estratégia unilateralmente. (OSBORNE; RUBINSTEIN, 1994)

Referências

- Agência Nacional de Telecomunicações. 2015. <http://www.anatel.gov.br/dados/index.php?option=com_content&view=article&id=283>. [Online; acessado em 31-July-2015].
- AKOUSH, S.; SAMEH, A. Movement prediction using bayesian learning for neural networks. In: IEEE. **Systems and Networks Communications, 2007. ICSNC 2007. Second International Conference on**. [S.l.], 2007. p. 6–6.
- AMICI, R. et al. Performance assessment of an epidemic protocol in vanet using real traces. **Procedia Computer Science**, Elsevier, v. 40, p. 92–99, 2014.
- BEHRISCH, M. et al. Sumo - simulation of urban mobility: An overview. In: **SIMUL 2011, The Third International Conference on Advances in System Simulation**. Barcelona, Spain: [s.n.], 2011.
- BRACCIALE, L. et al. **CRAWDAD data set roma/taxi (v. 2014-07-17)**. 2014. Downloaded from <http://crawdad.org/roma/taxi/>.
- BROCKMANN, D.; HUFNAGEL, L.; GEISEL, T. The scaling laws of human travel. **Nature**, Nature Publishing Group, v. 439, n. 7075, p. 462–465, 2006.
- CAMP, T.; BOLENG, J.; DAVIES, V. A survey of mobility models for ad hoc network research. **Wireless communications and mobile computing**, Wiley Online Library, v. 2, n. 5, p. 483–502, 2002.
- CAPKA, J.; BOUTABA, R. Mobility prediction in wireless networks using neural networks. In: **Management of Multimedia Networks and Services**. [S.l.]: Springer, 2004. p. 320–333.
- COELHO, P. et al. Node position forecast in manet with pherocast. In: **Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on**. [S.l.: s.n.], 2014. p. 73–80. ISSN 1550-445X.
- COLIZZA, V. et al. Modeling the worldwide spread of pandemic influenza: baseline case and containment interventions. **PLoS medicine**, Public Library of Science, v. 4, n. 1, p. e13, 2007.
- DASH, M. et al. Next place prediction by understanding mobility patterns. In: IEEE. **Pervasive Computing and Communication Workshops (PerCom Workshops), 2015 IEEE International Conference on**. [S.l.], 2015. p. 469–474.

DOMENICO, M. D.; LIMA, A.; MUSOLESI, M. Interdependence and predictability of human mobility and social interactions. **arXiv preprint arXiv:1210.2376**, 2012.

DORIGO, M.; BONABEAU, E.; THERAULAZ, G. Ant algorithms and stigmergy. **Future Generation Computer Systems**, v. 16, n. 8, p. 851 – 871, 2000. ISSN 0167-739X. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0167739X0000042X>>.

DORIGO, M.; GAMBARDELLA, L. M. Ant colony system: A cooperative learning approach to the traveling salesman problem. **Evolutionary Computation, IEEE Transactions on**, IEEE, v. 1, n. 1, p. 53–66, 1997.

EAGLE, N.; PENTLAND, A. Reality mining: sensing complex social systems. **Personal and ubiquitous computing**, Springer-Verlag, v. 10, n. 4, p. 255–268, 2006.

EKMAN, F. et al. Working day movement model. In: **Proceedings of the 1st ACM SIGMOBILE Workshop on Mobility Models**. New York, NY, USA: ACM, 2008. (MobilityModels '08), p. 33–40. ISBN 978-1-60558-111-8. Disponível em: <<http://doi.acm.org/10.1145/1374688.1374695>>.

FALL, K. A delay-tolerant network architecture for challenged internets. In: ACM. **Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications**. [S.l.], 2003. p. 27–34.

FLOYD, S. et al. **TCP Friendly Rate Control (TFRC): Protocol Specification**. IETF, 2008. RFC 5348 (Proposed Standard). (Request for Comments, 5348). Disponível em: <<http://www.ietf.org/rfc/rfc5348.txt>>.

GAMBARDELLA, L. M.; TAILLARD, É.; AGAZZI, G. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In: CITESEER. **New ideas in optimization**. [S.l.], 1999.

GOHIL, M. H. Mobile location prophecy: An analytical review. **International Journal of Computer Networks and Wireless Communications**, v. 4, n. 5, p. 327–333, 2014.

GONZALEZ, M. C.; HIDALGO, C. A.; BARABASI, A.-L. Understanding individual human mobility patterns. **Nature**, Nature Publishing Group, v. 453, n. 7196, p. 779–782, 2008.

GOSS, S. et al. Self-organized shortcuts in the argentine ant. **Naturwissenschaften**, Springer, v. 76, n. 12, p. 579–581, 1989.

GRAMMER, K.; FINK, B.; NEAVE, N. Human pheromones and sexual attraction. **European journal of obstetrics & gynecology and reproductive biology**, Elsevier, v. 118, n. 2, p. 135–142, 2005.

HORNER, M. W.; O'KELLY, M. E. Embedding economies of scale concepts for hub network design. **Journal of Transport Geography**, Elsevier, v. 9, n. 4, p. 255–265, 2001.

JEONG, J. et al. Tsf: Trajectory-based statistical forwarding for infrastructure-to-vehicle data delivery in vehicular networks. In: IEEE. **Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on**. [S.l.], 2010. p. 557–566.

JETCHEVA, J. G. et al. **CRAWDAD data set rice/ad_hoc_city (v. 2003-09-11)**. 2003. Downloaded from http://crawdad.cs.dartmouth.edu/rice/ad_hoc_city.

JIANG, D.; DELGROSSI, L. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In: **IEEE. Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE**. [S.l.], 2008. p. 2036–2040.

_____. Ieee 802.11p: Towards an international standard for wireless access in vehicular environments. In: **Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE**. [S.l.: s.n.], 2008. p. 2036–2040. ISSN 1550-2252.

KRUMM, J.; HORVITZ, E. Locadio: Inferring motion and location from wi-fi signal strengths. In: **MobiQuitous**. [S.l.: s.n.], 2004. p. 4–13.

LAURILA, J. K. et al. The mobile data challenge: Big data for mobile computing research. In: **Pervasive Computing**. [S.l.: s.n.], 2012.

LEE, U. et al. Code torrent: Content distribution using network coding in vanet. In: **Proceedings of the 1st International Workshop on Decentralized Resource Sharing in Mobile Computing and Networking**. New York, NY, USA: ACM, 2006. (MobiShare '06), p. 1–5. ISBN 1-59593-558-4. Disponível em: <<http://doi.acm.org/10.1145/1161252.1161254>>.

LIU, G. Y.; MAGUIRE, G. A predictive mobility management algorithm for wireless mobile computing and communications. In: **IEEE. Universal Personal Communications. 1995. Record., 1995 Fourth IEEE International Conference on**. [S.l.], 1995. p. 268–272.

MCKEOWN, N. Software-defined networking. **INFOCOM keynote talk, Apr**, 2009.

MCKEOWN, N. et al. Openflow: Enabling innovation in campus networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008. ISSN 0146-4833. Disponível em: <<http://doi.acm.org/10.1145/1355734.1355746>>.

MEIRELLES, F. 2015. <<http://eaesp.fgvsp.br/sites/eaesp.fgvsp.br/files/arquivos/pesti-gvcia2015ppt.pdf>>.

MOLINA-GIL, J.; CABALLERO-GIL, P.; CABALLERO-GIL, C. Countermeasures to avoid noncooperation in fully self-organized vanets. **The Scientific World Journal**, 2014.

ONNELA, J.-P. et al. Structure and tie strengths in mobile communication networks. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 104, n. 18, p. 7332–7336, 2007.

OSBORNE, M. J.; RUBINSTEIN, A. **A course in game theory**. [S.l.]: MIT press, 1994.

PAPANDREA, M.; GIORDANO, S. Location prediction and mobility modelling for enhanced localization solution. **Journal of Ambient Intelligence and Humanized Computing**, Springer Berlin Heidelberg, v. 5, n. 3, p. 279–295, 2014. ISSN 1868-5137. Disponível em: <<http://dx.doi.org/10.1007/s12652-013-0175-x>>.

- PIORKOWSKI, M.; SARAFIJANOVIC-DJUKIC, N.; GROSSGLAUSER, M. **CRAWDAD data set epfl/mobility (v. 2009-02-24)**. 2009. Downloaded from <http://crawdad.org/epfl/mobility/>.
- ROSEN, E.; VISWANATHAN, A.; CALLON, R. **Multiprotocol Label Switching Architecture**. IETF, 2001. RFC 3031 (Proposed Standard). (Request for Comments, 3031). Disponível em: <<http://www.ietf.org/rfc/rfc3031.txt>>.
- ROY, R. R. **Handbook of Mobile Ad Hoc Networks For Mobility Models**. 1st. ed. [S.l.]: Springer, 2011.
- SILVA, V. B. da et al. A trajectory-based approach to improve delivery in drive-thru internet scenarios. In: **Workshop on Emerging Vehicular Networks: V2V/V2I and Railroad Communications, IEEE International Conference on Communications – ICC 2013**. [S.l.: s.n.], 2013.
- SUMO. 2015. <<http://sourceforge.net/apps/mediawiki/sumo/index.php?title=TAPASCologne>>. [Online; acessado em 04-Aug-2015].
- SURIYAPAIBONWATTANA, K.; POMAVALAI, C. An effective safety alert broadcast algorithm for vanet. In: **Communications and Information Technologies, 2008. ISCIT 2008. International Symposium on**. [S.l.: s.n.], 2008. p. 247–250.
- TANIUCHI, K. et al. Ieee 802.21: Media independent handover: Features, applicability, and realization. **Communications Magazine, IEEE**, v. 47, n. 1, p. 112–120, January 2009. ISSN 0163-6804.
- TSUI, J. B.-Y. **Fundamentals of global positioning system receivers**. [S.l.]: Wiley-Interscience, 2000.
- WIENER, N. **Cybernetics or Control and Communication in the Animal and the Machine**. [S.l.]: MIT press, 1961. v. 25.
- ZHENG, Y. et al. Understanding mobility based on gps data. In: **ACM. Proceedings of the 10th international conference on Ubiquitous computing**. [S.l.], 2008. p. 312–321.
- ZISSIS, D.; XIDIAS, E. K.; LEKKAS, D. Real-time vessel behavior prediction. **Evolving Systems**, Springer, p. 1–12, 2015.

Apêndices

Aplicação para Análise Real

Todos os aplicativos descritos são *Software Livre* e podem ser utilizados para qualquer aplicação não-comercial, desde que seguindo as diretivas da licença *BSD (3-Clause)* que se encontra anexada a cada repositório dos aplicativos.

A.1 Aplicação Android

A aplicação *Android*, foi feita para funcionar em qualquer *smart-phone Android*, independente da versão, pois a aplicação é leve e utiliza informações presentes desde as primeiras versões do *Android*. O programa foi feito utilizando a Interface de Programação de Aplicativos (API) nativa em Java e está disponível em <<https://github.com/enriquefynn/datagather-client>> sobre a licença BSD.

A arquitetura básica é mostrada na Figura 26. Assim que o programa inicia, uma tela como na Figura 27 aparece, esta é a única tela do aplicativo. Nela são encontrados os dados:

1. O número de registros gravados no momento.
2. Dados sobre a sincronização com o servidor.
3. Último erro encontrado ao mandar os dados ao servidor.
4. Botão com os termos de uso.

O aplicativo inicia um *Service*, que continua gravando os dados mesmo que o aplicativo tenha parado de funcionar, assim se o aplicativo, eventualmente, for iniciado novamente, será apresentada a tela da Figura 27, com os dados atualizados do *Service*.

O aplicativo inicia sempre quando o celular é ligado, e como na Figura 26, ele começa a gravar os dados assim que possível, os dados são:

- Dados de posicionamento grosseiros (obtidos a partir de metadados e não do GPS):

- Longitude.
- Latitude.
- *timestamp* informando quando foi feita a coleta dos dados.

□ Dados sobre a redes *wireless*, tais que para cada rede:

- BSSID.
- Potência da rede.
- *timestamp* informando quando foi feita a coleta dos dados.

Assim que são coletados um número maior ou igual a 500 dados, espaçados de 15 em 15 segundos, ou seja, aproximadamente 2 horas de dados o aplicativo tenta enviar os dados ao servidor, como o volume de dados pode ser alto, ele espera que haja uma conexão *WiFi* e não usa o pacote de dados, pois isso poderia custar muito ao usuário.

Antes de enviar os dados, o aplicativo precisa se autenticar no servidor, por meio de uma conexão segura utilizando Camada de Socket Seguro (SSL), para isso, na primeira vez que é iniciado o aplicativo, este cria um usuário e senha pseudônimos, essas são as credenciais do usuário no servidor, garantindo assim que o servidor não saiba quem é o usuário pelo seu nome.

Uma vez autenticado, o aplicativo envia os dados em lotes de 500 entradas, até que tudo tenha sido enviado, caso ocorra um erro, o lote não é enviado e tenta-se de novo, mais tarde, enquanto isso, novos dados são coletados em paralelo, sem interrupção, a cada 15 segundos.

Uma vez que os dados estão no servidor, não há necessidade para do aplicativo manter esses dados no celular, então, ele apaga esses últimos 500 dados que foram enviados assim que o servidor responde com uma *Confirmação* (ACK) para o envio dos dados.

Todos os dados são guardados no celular em um banco de dados *MySQL* nativo ao *Android*.

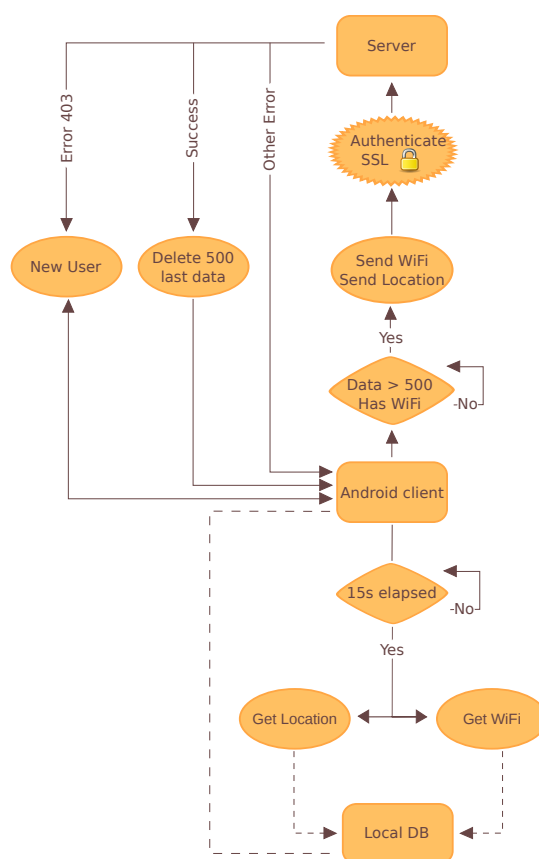


Figura 26 – Arquitetura do aplicativo Android.

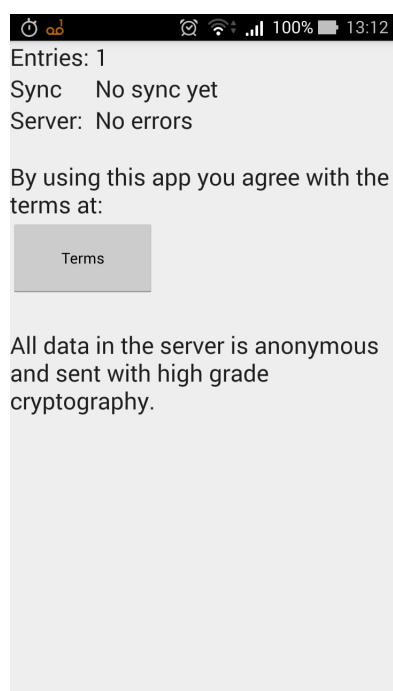


Figura 27 – Tela do aplicativo.

A.2 Servidor na Nuvem

O servidor foi feito utilizando a linguagem de programação *Javascript*, com o *Nodejs* como *framework*, é, como o aplicativo *Android*, um *Software Livre*, e seu código se encontra em `<https://github.com/enriquefynn/datagather>`

O servidor é executado na *Amazon*, que fornece um serviço de hospedagem na nuvem. Os dados são gravados de forma pseudônimas, já que não é possível associar o *id* do usuário com seu nome real, contudo, se o usuário não estiver utilizando um *proxy* para a conexão com o servidor, este pode associar, teoricamente, o usuário com seu endereço físico de *ip*.

O servidor armazena os dados em um banco de dados não relacional, mais especificamente em uma instância do *MongoDB*. A cada duas horas os dados são apagados do banco de dados e salvos em arquivos de texto separados para cada usuário, mantendo as buscas mais rápidas no banco de dados.

As rotas do servidor são definidas da seguinte forma, com os métodos do Protocolo de Transferência de Hipertexto (HTTP):

□ GET

- `/me`: Retorna status 403 se não autenticado ou dados do usuário, *cookies* precisam ser utilizados e autenticação feita.
- `/lastLocation`: Retorna os dados da última localização coletada:

```
{timestamp: "último unix epoch wifi timestamp"}
```

- `/lastWifi`: Retorna os dados da última Wifi coletada:

```
{timestamp: "último unix epoch wifi timestamp"}
```

□ POST

- `/auth`: Autentica ou cria um novo usuário com os dados do *json*:

```
{  
  username: "usuário",  
  password: "senha"  
}
```

Se houver um erro, retorna status 403.

- `/addLocation`: Adiciona dados de localização, conforme o *json*:

```
[{  
  lat: "Latitude Decimal",  
  lon: "Longitude Decimal",  
  timestamp: "unix epoch timestamp"  
}]
```

Retorna {status: Done} e status 200 se correto.

- /addWifi: Adiciona dados de *WiFi*, conforme o *json*:

```
[{  
  name: "bssid1:potência1, bssid2:potência2, ...",  
  timestamp: "unix epoch timestamp"  
}]
```

Retorna {status: Done} e status 200 se correto.