UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUSTAVO KUHN ANDRIOTTI

Modelagem de Motoristas e Cenários de Escolha de Rota em Simulações de Tráfego Veicular Urbano

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Ana Lúcia Cetertich Bazzan
Orientadora

Porto Alegre, novembro de 2004

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Andriotti, Gustavo Kuhn

Modelagem de Motoristas e Cenários de Escolha de Rota em Simulações de Tráfego Veicular Urbano / Gustavo Kuhn Andriotti. – Porto Alegre: PPGC da UFRGS, 2004.

81 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2004. Orientadora: Ana Lúcia Cetertich Bazzan.

1. Autômato Celular. 2. Modelo Nagel–Schreckenberg. 3. Modelo de Motoristas. 4. Simulação de Trânsito. I. Bazzan, Ana Lúcia Cetertich. II. Título.

Este trabalho é dedicado a Romy Klein.

AGRADECIMENTOS

Agradeço especialmente à minha orientadora, Ana L. C. Bazzan, pela paciência e dedicação durante todos este anos. Preciso também agradecer à minha família que sempre me apoiou e incentivou nos caminhos que escolhi e meu mais sincero obrigado a Romy Klein cuja dedicação e abnegação tornou este trabalho possível.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

NaSch	Nagel–Schreckenberg
BDI	Crenças, Desejos e intenções, do inglês: *Believes, Desires and Intentions*
DRACULA	Sigla para: *Dynamic Route Assignment Combining User Learning and microsimulAtion*
ITS	Sistemas de transporte inteligentes, do inglês: *Intelligent Transportation Systems*
ATIS	Sistema avançado de informações de trajetos, do inglês: *Advanced Traveller Information Sistem*
SMA	Sistemas multiagentes
IVHS	Sistemas inteligentes de veículos em auto-estrada, do inglês: *Intelligent Vehicle Highway Systems*
BJH	Modelo apresentado em (BENJAMIN; JOHNSON; HUI, 1996), nome deriva do nome dos autores: Benjamin, Johnson e Hui
VDR	Modelo apresentado em (BARLOVIC et al., 1998), nome vem do inglês *Velocity-dependent Randomization*
ITSUMO	*Intelligent Transportation System for Urban Mobility*
DRIVER-DFW	*DRIVER Development Framework*
SeSAm	*Shell for Simulated Agent Systems*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho visa apresentar uma metodologia para modelagem de motoristas a serem utilizados em simulações de tráfego veicular discreto. Além da metodologia, será apresentada uma plataforma para implementação de motoristas, chamada *DRIVER-DFW*, baseada neste conceito.

Inicialmente, serão apresentados alguns modelos de movimentação de veículos baseados no modelo de autômato celular Nagel–Schreckenberg. O modelo básico será apresentado juntamente com alguns de seus aperfeiçoamentos, que são os modelos utilizados no simulador *ITSUMO*, que por sua vez é utilizado como base para o trabalho.

Além dos modelos de autômato celular, serão apresentados modelos de planejamento de rota, que se utilizam de várias heurísticas para a tomada de decisão dos motoristas. Destes, selecionou-se um para implementação e demonstração.

Mostradas as etapas para composição do modelo completo de motorista, isto é, movimentação e planejamento, será apresentada a plataforma para implementação de motoristas desenvolvida neste trabalho. Esta separação é a base da plataforma *DRIVER-DFW* que é discutida com mais detalhes para auxiliar a compreensão do seu funcionamento. Além disso, é mostrado como a metodologia é aplicada na plataforma para implementação de motoristas *DRIVER-DFW*.

Por fim, conclui-se que este trabalho apresenta uma alternativa bastante atraente para a implementação de modelos de motoristas, com uma metodologia e uma plataforma de desenvolvimento. Também são apresentadas as diretrizes para dar prosseguimento a este.

Palavras-chave: Autômato Celular, Modelo Nagel–Schreckenberg, Modelo de Motoristas, Simulação de Trânsito.

Driver modelling and route choosing at urban traffic simulations

ABSTRACT

In this work a methodology is presented, which models drivers in discrete traffic simulations. Additionally, a driver development framework, called *DRIVER-DFW*, will be shown.

Initially some vehicle movement models based on the Nagel–Schreckenberg cellular automata will be presented. The basic model, along with some of its improvements, will also be presented. These are the basis of the *ITSUMO* project including the present framework fits in. This simulator is used as simulation engine for the driver development framework *DRIVER-DFW*.

Further, some route choice planning models will be shown. These models use some heuristics for the route choosing decision make. One of these models had been selected for implementation and evaluation.

After showing the steps to create a driver model, i. e., movement and planning, the driver development framework *DRIVER-DFW* will be described. This division on the driver model is the basis of the driver development framework *DRIVER-DFW*, that will be discussed in order to exemplify the use of the framework.

Then is concluded that this work present a valuable alternative to model and implement a driver model, through the methodology and framework presented. Future work is also discussed.

Keywords: Multi-Agent System, Cellular Automaton, Nagel–Schreckenberg Model, Driver's Model, Traffic Simulation.

1 INTRODUÇÃO

Este trabalho tem o tráfego veicular urbano como objeto de estudo. Estudar o tráfego adquire importância na medida em que os problemas a ele relacionados crescem em complexidade e na freqüência com que ocorrem. Por isso, compreender a sua dinâmica pode auxiliar na solução ou atenuação dos problemas que dele emergem. Entre eles está o aumento do estresse, prejuízos financeiros decorrentes do tempo de deslocamento, agressão ao ambiente devido à poluição oriunda da queima de combustível e poluição sonora. Buscar um melhor aproveitamento dos recursos é fundamental para atenuar estes efeitos indesejáveis.

Entre as possibilidades para se atenuar os problemas mencionados pode estar a redução da frota circulante ou alteração na infra-estrutura com o objetivo de otimizar o uso dos recursos. A redução da frota de veículos não parece ser uma realidade à curto prazo, pois, ao analisar, por exemplo, os dados da frota de veículos do Rio Grande do Sul, nota-se que não há sinais de qualquer redução na quantidades de veículos[1]. Sendo assim, otimizar o uso da infra-estrutura[2] existente é a alternativa mais viável na abordagem de alguns problemas relacionados ao trânsito.

Os recursos, principalmente financeiros, são limitados e, portanto, é preciso fazer um planejamento para qualquer mudança na organização da infra-estrutura de transporte, isto é, alterar sentido das ruas, instalar semáforos e outras medidas que não envolvam a criação de novas ruas ou meios de acesso. Este tipo de planejamento pode ser melhor elaborado com o auxílio de um instrumento capaz de fornecer alguma estimativa do impacto de uma determinada alteração. Com esta finalidade, um sistema computacional torna-se uma alternativa bastante atrativa em termos de custos financeiros e de tempo.

Disponibilizar um artefato computacional capaz de fornecer previsões pode auxiliar e otimizar o planejamento de melhorias na infra-estrutura viária. Mas para isso é preciso reproduzir os elementos do trânsito na forma de algoritmos. Sintetizar ruas, sinalização e legislação são tarefas relativamente fáceis quando comparadas com a principal: sintetizar o comportamento dos motoristas.

Resumir em um algoritmo um motorista é, na verdade, fazer uma abstração de parte do comportamento humano, neste caso restrito ao trânsito. É nesta tarefa que este trabalho faz a sua contribuição, isto é, auxiliar o pesquisador na criação de heurísticas capazes de reproduzir parcialmente o comportamento humano relacionado ao trânsito.

Portanto, este trabalho visa fornecer uma ferramenta para implementação de modelos de motoristas no contexto da infra-estrutura de simulação existente. Através dela, o

[1]Informações obtidas ao analisar as estatísticas de licenciamento de veículos no Detran-RS, disponíveis em http://www.detran.rs.gov.br/

[2]Quando se menciona o termo infra-estrutura de trânsito refere-se às estruturas físicas das ruas e faixas disponíveis.

pesquisador poderá rapidamente modelar um motorista sem que isso implique em envolvê-lo nos aspectos de baixo nível do simulador restringindo sua preocupação na implementação do modelo de motorista. Como grande parte das plataformas de implementação, esta também busca reduzir o tempo de desenvolvimento de novos algoritmos e, com isso, o esforço de implementação deverá ser, fundamentalmente, de sintetização do modelo utilizando os recursos da ferramenta.

Um dos objetivos do projeto *ITSUMO*, acrônimo de *Intelligent Transportation System for Urban Mobility* é a construção de um simulador. Este simulador fornece uma infra-estrutura para simulação mas dispõe apenas de um motorista básico, baseado no modelo Nagel–Schreckenberg (NAGEL; SCHRECKENBERG, 1992). Toda e qualquer modificação neste motorista básico implica em um grande esforço de implementação que resulta em uma solução *ad hoc*.

Para suprir esta deficiência elaborou-se uma metodologia para modelagem de motoristas juntamente com uma plataforma para criação e validação de motoristas, chamada *DRIVER-DFW*. Com isso, é fornecida uma solução completa de plataforma de desenvolvimento e validação de modelos de motoristas. A plataforma para implementação de motoristas, *DRIVER-DFW*, se apresenta como uma camada de abstração do simulador. Esta camada permite que aperfeiçoamentos no simulador não interfiram no desenvolvimento de modelos de motoristas e vice-versa.

1.1 Motivação e Objetivos

Congestionamento, acidentes e emissão de poluentes são alguns dos temas que preocupam os pesquisadores de maneira geral. Estes problemas trazem conseqüências negativas aos usuários do trânsito.

Objetivamente, este trabalho relaciona-se com os efeitos diretos sobre os usuários, isto é, consumo de tempo em deslocamento e insatisfação com as condições de trânsito. Buscar um melhor aproveitamento da infra-estrutura existente e permitir que se possam experimentar modificações são os pontos de interesse deste trabalho.

Desta forma, a plataforma que está sendo apresentada permite que se possa desenvolver e experimentar modelos de motoristas. Isto é, permitir que o pesquisador busque o modelo que melhor descreva o comportamento dos agentes de trânsito. Por isso, a criação de um modelo com razoável precisão e acurácia é passo fundamental para permitir a correta previsão do comportamento do trânsito.

Uma vez que os órgãos de planejamento e controle de tráfego estão interessados na satisfação coletiva, ou seja, que um maior número motoristas seja beneficiado, faz-se necessário modelar o comportamento macroscópico. Para atingir tal objetivo foi implementado o modelo Nagel–Schreckenberg de movimentação veicular (NAGEL; SCHRECKENBERG, 1992). Como será visto no capítulo 2, este modelo é pouco fiel ao comportamento individual numa análise microscópica, mas coletivamente torna-se bastante verossímil.

1.1.1 Modelagem de motoristas

A tarefa de modelar um motorista pode ser motivada por várias razões e entre elas está conceber um algoritmo fiel ao comportamento humano ou que cumpra determinadas tarefas. Modelar o comportamento humano pode ser considerada a mais complexa das atividades, enquanto projetar um modelo para realizar uma tarefa específica, sem a necessidade de reproduzir a reação humana, pode apresentar menos dificuldades. Estas duas

categorias de modelos se complementam, dependendo do objetivo do estudo, e interagem para realizar simulações de trânsito.

Independente da categoria do modelo é preciso estabelecer uma metodologia de modelagem. Independente do modelo concebido, ele precisa ser sintetizado em um algoritmo computável. Além disso, este algoritmo precisa ser formulado de forma que seja possível implementá-lo utilizando a infra-estrutura disponível. E com este objetivo, estabeleceu-se uma metodologia de modelagem de qualquer motorista.

A metodologia para modelagem de motoristas proposta neste trabalho consiste em criar dois níveis de abstração distintos: movimentação e planejamento. Estas partes são complementares e necessárias para qualquer modelo que se pretenda implementar. A primeira é responsável por movimentar o veículo e por coordenar a sua interação com os outros veículos. Já a segunda se encarrega de planejar uma rota a ser seguida pelo motorista.

1.1.1.1 Movimentação

A movimentação é uma decisão local de mais baixo nível. Nesta etapa encontra-se o algoritmo de movimentação dos veículos, isto é, qual o próximo movimento do veículo e como ele afeta e é afetado pela movimentação dos outros veículos na sua vizinhança.

Normalmente, a movimentação é dotada de uma forte reatividade e é nela que estão os objetivos mais simples e imediatos. O que fazer para evitar colisões, obter uma maior velocidade e respeitar a sinalização de trânsito são algumas das tarefas pelas quais a decisão de movimentação deve ser responsável. Não se deve colocar nesta parte qualquer tipo de planejamento ou heurística mais complexa pois estas são tarefas da etapa de planejamento.

Como as simulações são discretas é solicitado ao motorista que movimente seu veículo a cada novo passo de simulação. Isto significa que todos os veículos solicitarão uma decisão do seu motorista a cada passo de simulação. Desta forma, o algoritmo precisa ser simples de forma a não comprometer o desempenho do sistema.

É preciso observar que as tomadas de decisão são feitas concorrentemente, isto é, não há qualquer ordem pré-estabelecida de um veículo tomar a sua decisão antes de outro.

1.1.1.2 Planejamento

Esta é a etapa de mais alto nível de um modelo de motorista, conforme a metodologia para modelagem de motoristas propostas neste trabalho. O planejamento de rota deve ser feito nesta parte, pois nela deve ser decidida qual a próxima direção em todos os cruzamentos pelos quais o veículo, associado ao motorista, passa. Como decisões deste tipo ocorrem em menor freqüência é possível que o tempo e recursos consumidos nesta etapa sejam maiores, pois haverá economia destes na etapa de movimentação.

Como comparativo, se um veículo trafega na velocidade máxima permitida dentro do perímetro urbano ($60km/h$, aproximadamente $17m/s$) e a distância média entre cada cruzamento é de $200m$, então o motorista deverá efetuar algum planejamento a cada 11 passos de simulação (que representam $11s$, caso o passo de simulação represente $1s$).

Existe ainda outra possibilidade, que é efetuar o planejamento antes de iniciar a simulação, isto é, pré-planejar a rota antes de usá-la. Este é o caso mais freqüentemente encontrado na literatura especializada. A desvantagem desta abordagem é inviabilizar qualquer reformulação do trajeto caso alguma condição inesperada ocorra, como um congestionamento imprevisto.

1.1.2 Motivação para se modelar um motorista

Quando se modela um motorista, busca-se estudar algum problema relacionado com o trânsito. Dentre estes problemas, destacam-se o planejamento do sistema viário e o fornecimento de algum serviço ao usuário deste. Em todos estes, é necessário reproduzir o comportamento humano, com algum grau de similaridade, em um modelo de motorista.

Independente do interesse, parte-se de um ponto comum que é reproduzir o comportamento humano na movimentação dos veículos. Para isso é preciso que a movimentação destes corresponda, com alguma fidelidade, às ações observadas em motoristas humanos ao conduzirem seus veículos. É também necessário reproduzir a resposta aos estímulos provenientes dos outros veículos. Neste sentido, o algoritmo responsável pela movimentação dos veículos deve respeitar estas restrições.

Quando se deseja utilizar as simulações para planejamento viário é necessário reproduzir o comportamento humano no planejamento. Já na prestação de algum serviço de auxílio aos motoristas, apenas espera-se que o trajeto apresentado respeite algumas restrições.

No caso de planejamento viário, busca-se um modelo que reproduza com certa qualidade o comportamento das pessoas quando efetuam o planejamento de suas rotas. Isto é importante para que seja possível prever o impacto de mudanças estruturais na malha viária. Um exemplo é avaliar o impacto da duplicação de uma determinada faixa ou separar os veículos particulares do transporte coletivo, organização que se observa em várias cidades, inclusive Porto Alegre.

Desta forma, é possível auxiliar na avaliação de mudanças para melhorar as condições de trânsito em regiões que apresentam problemas, ou mesmo o impacto de novos pontos atratores de tráfego, como por exemplo uma indústria ou um centro comercial, para o trânsito na região.

Outro objetivo ao modelar um motorista pode ser o fornecimento de algum serviço aos usuários. Isto está normalmente associado a sistemas automatizados de busca de rota.

Fornecer um caminho alternativo para ir de um ponto à outro dentro da malha viária é o mais evidente deles. Para isso, normalmente, respeita-se alguma restrição na busca da rota. Por exemplo, estas restrições podem ser a passagem por um ponto intermediário no trajeto, encontrar a menor distância ou menor tempo de percurso.

Para fornecer este tipo de serviço não é necessário que se modele um motorista para simular o comportamento humano na etapa de planejamento. Entretanto, é preciso que exista um ambiente de simulação, suficientemente fiel à realidade, que torne a solução fornecida aceitável. Para isto é preciso povoar o sistema com motoristas ruído, isto é, que, coletivamente, reproduzam as condições de ocupação do trânsito apresentadas na realidade.

1.1.2.1 Considerações a respeito da modelagem

Imagina-se que reproduzir o comportamento humano completamente, mesmo que possível, seria complexo demais mesmo numa aplicação restrita. Por isso buscam-se modelos, comparativamente muito mais simples mas que conservem alguma similaridade com a realidade. Pode-se, entretanto, optar entre reproduzir apenas o comportamento macroscópico ou também o comportamento microscópico, que neste caso apresenta maior complexidade.

Os modelos de movimentação são, em sua maioria, empíricos. Isto significa que são feitos com base em observações da realidade (alguns exemplos de modelos em (NAGEL;

SCHRECKENBERG, 1992) e (KNOSPE et al., 2003)). Entretanto, não há uma análise suficientemente precisa que garanta sua fidelidade. São, portanto, fruto de observações e suposições a respeito do mecanismo de inferência humano aplicado ao trânsito. Estas observações se refletem em ajustes finos[3] nos modelos, de forma a se aproximarem das observações.

1.1.3 Objetivo da plataforma para implementação de motoristas *DRIVER-DFW*

A plataforma para implementação de motoristas *DRIVER-DFW* apresentada foi concebida para facilitar o processo de concepção e experimentação de diversos modelos de motorista. Conceber um modelo abstrato de motorista é apenas parte do problema. É preciso uma infra-estrutura de simulação que permita a inclusão deste novo modelo.

Para se obter uma infra-estrutura de simulação que atenda às necessidades particulares de um modelo, pode-se desenvolver um simulador próprio ou adaptar outro já existente. Desenvolver um novo simulador pode ter impacto no tempo entre a concepção do modelo e sua experimentação. Além disso, pode-se, dependendo do modo como ocorreu a concepção, tornar a nova ferramenta de simulação muito especializada, dificultando modificações e aperfeiçoamentos.

Outra alternativa é utilizar uma infra-estrutura existente e adaptá-la às necessidades do pesquisador. Para isso, entretanto, pode ser necessário estudar o funcionamento do simulador escolhido no nível do código fonte, que podem não estar disponíveis ou sob uso restrito.

Para tentar minimizar estes problemas de implementação que a plataforma para implementação de motoristas *DRIVER-DFW* se apresenta como solução. Ela tem como objetivo ser flexível o suficiente para permitir que as mais variadas heurísticas e paradigmas sejam passíveis de implementação. Além disso, ela precisa apresentar uma camada de abstração entre o modelo e o simulador.

É preciso que o mínimo possível de restrições com relação à concepção de um modelo sejam impostas aos usuários. Com isso, os usuários não precisam recorrer a outras soluções para experimentar seus modelos. Além disso, é preciso fornecer o máximo de recursos de forma que não seja necessário efetuar alterações na plataforma ou no simulador a fim de atender às suas necessidades. A plataforma deve fornecer todos os recursos necessários ao modelo de motorista a ser implementado.

Com estes objetivos que pretende-se fornecer uma solução que seja atrativa aos pesquisadores e os desonere da tarefa de implementação de um modelo de motorista. Além disso, esta solução é totalmente gratuita e com poucas restrições de uso (licença *GNU/GPL*).

1.2 Contribuições

Será mostrada uma metodologia associada a uma plataforma para auxiliar no desenvolvimento de algoritmos para a escolha de rota. Percebe-se com os trabalhos apresentados em (KLÜGL; BAZZAN, 2002), (BAZZAN; KLÜGL, 2003), (KLÜGL; BAZZAN; WAHLE, 2003) e em (ROSSETTI et al., 2002) que existe uma necessidade de um método para implementar modelos de motoristas. Este problema torna-se especialmente flagrante quando há a necessidade de formalizar em um algoritmo uma heurística para planejamento e escolha de rota. Com esta metodologia estabelece-se diretrizes para a concepção

[3]Na área de transportes este ajuste fi no chama-se calibragem.

de um modelo de motoristas.

1.3 Organização dos capítulos

Os capítulos deste trabalho estão dispostos da seguinte maneira: no capítulo 2 é apresentado o modelo Nagel–Schreckenberg utilizado em todo o ambiente de simulação. Também são apresentadas as extensões propostas ao modelo, as quais buscam aprimora-lo para torná-lo mais abrangente. Na seqüência tem-se o capítulo 3, em que é apresentado o simulador existente no projeto *ITSUMO* apenas em um nível que permita entender o funcionamento da plataforma para implementação de motoristas.

Alguns modelos de heurísticas utilizados no planejamento de rota são apresentados no capítulo 4. Nele são apresentados dois tipos de heurísticas para o planejamento e escolha de rota. O primeiro utiliza a teoria de jogos, para tentar reproduzir o comportamento humano nesta situação. Já o segundo utiliza a lógica *BDI* para a escolha de rota. Ambos têm em comum a busca por uma heurística capaz de sintetizar o comportamento humano no planejamento e escolha de rota.

No capítulo 5 se apresenta plataforma para implementação de motoristas *DRIVER-DFW* a qual é utilizada para gerar os resultados apresentados no capítulo seguinte. No capítulo 7 são apresentadas as considerações finais a respeito do trabalho. Será mostrada a vantagem em se utilizar a abordagem proposta e os benefícios obtidos. Novas possibilidades se apresentam com este trabalho e algumas delas são expostas como trabalhos futuros.

2 MODELOS DE MOVIMENTAÇÃO MICROSCÓPICAS BASEADAS NO MODELO NAGEL–SCHRECKENBERG

Neste capítulo serão apresentados modelos de movimentação baseados no modelo Nagel–Schreckenberg (NAGEL; SCHRECKENBERG, 1992). Este modelo é um autômato celular que descreve o comportamento microscópico de um veículo. O modelo, entretanto, tem por objetivo reproduzir o comportamento macroscópico.

Por se tratar de um modelo microscópico pode-se observar a evolução da simulação no nível de detalhe máximo, isto é, o comportamento individual dos veículos. Entretanto, o modelo não garante que neste nível de detalhe a proximidade com a realidade seja mantido. O que o modelo garante é realismo quando se faz observações macroscópicas, isto é, acompanha-se a evolução de um conjunto de veículos.

O modelo baseado em automato celular tem a propriedade de ser eficiente do ponto de vista computacional e de reproduzir, com uma qualidade aceitável, o comportamento macroscópico humano.

Além disto, são apresentadas variantes e extensões (RICKERT et al., 1996; WAGNER; NAGEL; WOLF, 1997; KNOSPE et al., 2003) do modelo Nagel–Schreckenberg básico. As modificações buscam tornar o modelo mais completo e abrangente, melhorando a robustez e possibilitando a utilização em um conjunto mais amplo de situações e cenários.

As análises estatísticas que validam os modelos foram omitidas mas pode ser encontradas em (ANDRIOTTI, 2003).

2.1 Modelo Original

O modelo Nagel–Schreckenberg (NAGEL; SCHRECKENBERG, 1992) propõe uma alternativa aos modelos baseados em dinâmica de fluidos e obtém resultados semelhantes aos apresentados por estes.

Tratando-se de um autômato celular o modelo Nagel–Schreckenberg é, na verdade, um conjunto restrito de regras que rege o comportamento de um veículo que movimenta-se em uma faixa única de auto-estrada, objetivo inicial do modelo. As regras estão listadas abaixo:

1. Aceleração: Se a velocidade v do veículo é inferior à $v_{máxima}$ e a distância até o próximo veículo é superior à $v + 1$, então ele deve acelerar, aumentando sua velocidade $v \leftarrow v + 1$;

2. Desaceleração: Caso a distância até o próximo veículo seja inferior ou igual a sua

velocidade, isto é, $v \geq gap$[1] então o veículo deve reduzir sua velocidade: $v \leftarrow gap$;

3. Aleatoriedade: cada veículo pode, com uma probabilidade $p_{desaceleração}$, reduzir sua velocidade em uma unidade, isto é, $v \leftarrow v - 1$;

4. Movimentação: cada veículo avança v células.

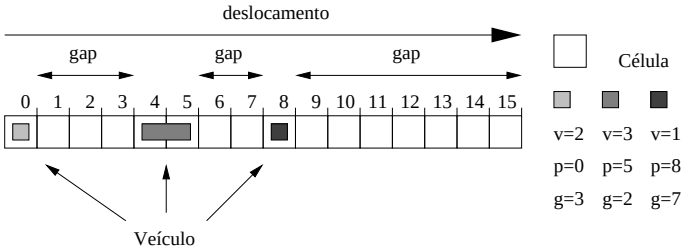

Figura 2.1: Elementos do modelo Nagel–Schreckenberg.

Simplificando, pode-se sintetizar estas regras nos seguintes passos algorítmicos simples (utilizando a tabela de funções[2] 2.2 como referência e o desenho da figura 2.1[3] para melhor entendimento):

$$v(i,t+1) = min(gap(i,t), v(i,t)+1, v_{máxima}(i,t)) \tag{2.1}$$

$$v(i,t+1) = \begin{cases} v(i,t+1) & se\ rand() \geq p_{desaceleração}(i,t) \\ v(i,t+1)-1 & se\ rand() < p_{desaceleração}(i,t) \wedge v(i,t) > 0 \end{cases} \tag{2.2}$$

$$pos(i,t+1) = pos(i,t) + v(i,t+1) \tag{2.3}$$

Esse processo se repete a cada ciclo de simulação para todos os veículos envolvidos. Além disso, trata-se de um modelo discreto, tanto em tempo quanto espaço, isto é, na versão original cada passo de simulação corresponde a 1 segundo do tempo real e cada célula corresponde a 7,5 metros. Os parâmetros originais estão sintetizados na tabela 2.1, mas podem ser diferentes dependendo do modelo.

O passo 2 (equação 2.2) do autômato celular é importante para imprimir um caráter realista na simulação. Sem este parâmetro, a simulação seria totalmente determinística após a fase de acomodação do sistema. Através desta aleatoriedade, é possível simular e observar o ponto de máxima capacidade de uma faixa, isto é, o número máximo de veículos que a faixa comporta antes que ocorram zonas de congestionamento e o sistema passe a apresentar ondas de paradas e arrancadas[4], como vistos em (NAGEL; SCHRECKENBERG, 1992), normalmente observados no ambiente real.

Na figura 2.2 pode-se ver um exemplo de formação de congestionamento onde as velocidades dos veículos indicam como a desaceleração do veículo mais a frente afeta a velocidade dos demais. Esse fenômeno se acentua com o aumento da ocupação da faixa.

[1] gap é o termo empregado para designar a distância inter-veicular, isto é, a distância entre o veículo que se está analisando até o veículo mais próximo, neste caso, o imediatamente a frente, conforme (NAGEL; SCHRECKENBERG, 1992). Na área de transportes este termo é o correspondente à espaçamento.

[2] Originalmente não eram funções mas estão dispostos desta forma para fi carem de acordo com as extensões que utilizam esta notação.

[3] Na fi gura 2.1 existe um veículo com tamanho superior a 1 célula, que não estava previsto no modelo apresentado em (NAGEL; SCHRECKENBERG, 1992), e lá está para fi car claro como é calculado o *gap*.

[4] Expressão original: *stop-start-wave*.

Tabela 2.1: Tabela de unidades de medida adotadas no modelo original.

Parâmetro	unidade
distância	células
célula	7,5 metros
tempo	passo
passo	1 segundo
velocidade	células / passo
velocidade máxima	5 células / passo
incrementos	velocidade

Tabela 2.2: Tabela de funções e parâmetros para movimentação.

Função	Valor retornado
$pos(i,t)$	Posição do veículo i no tempo t.
$gap(i,t)$	Distância entre o veículo i e o veículo imediatamente a frente, no tempo t.
$\textit{visão}(i,t)$	Distância máxima que o veículo i observa a sua frente, normalmente $\textit{visão}(i,t) = v(i,t) + 1$.
$rand()$	Retorna um valor x, tal que $x \in \Re \wedge 0 \leq x < 1$
$p_{\textit{desaceleração}}(i,t)$	Probabilidade do veículo i de desaceleração, tal que $p_{\textit{desaceleração}}(i,t) \in \Re \wedge 0 \leq p_{\textit{desaceleração}} \leq 1$

2.2 Modelo de Rickert para interação entre faixas

A extensão para duas faixas (RICKERT et al., 1996), na verdade, pode ser aplicada para um ambiente com n faixas, pois pode-se analisá-los aos pares. Isto é, aplicando o algoritmo usado a cada par de faixas das n existentes.

O algoritmo proposto parte de uma premissa: não permitir colisões, isto é, a troca de faixa somente ocorre de forma segura. A troca de faixa não pode prejudicar a trajetória dos veículos que já se encontram na faixa de destino, ou seja, a troca de faixa não pode implicar em redução de velocidade por parte dos veículos que se encontram na faixa de destino (no caso de simetria, discutido mais adiante).

Para realizar um passo de simulação são necessários 2 sub-passos: troca de faixa e movimentação linear. A troca de faixa se dá sem qualquer movimentação linear, isto é, existe apenas uma movimentação transversal onde ocorre a troca de faixa. Efetuada a troca de faixa ocorre a movimentação linear, como se as faixas fossem independentes, conforme explicado na seção 2.1.

Existem duas abordagens para o modelo de duas faixas: simétrica e assimétrica. No modelo simétrico, as duas faixas são ocupadas sem qualquer preferência pelos veículos, enquanto no modelo assimétrico, mais realista, os veículos procuram retornar à faixa da direita[5] sempre que possível.

Para ocorrer uma troca de faixa todas as regras abaixo precisam ser satisfeitas. Para as funções, considere a tabela 2.2 e a tabela 2.3:

1. A situação na faixa atual é insatisfatória, isto é, é preciso reduzir a velocidade: $gap(i,t) < \textit{visão}(i,t)$;

[5]No caso de países onde a faixa de menor velocidade, ou preferencial, é a da esquerda, como Inglaterra e Japão, deve-se adaptar a regra tornando a faixa da esquerda a faixa preferencial.

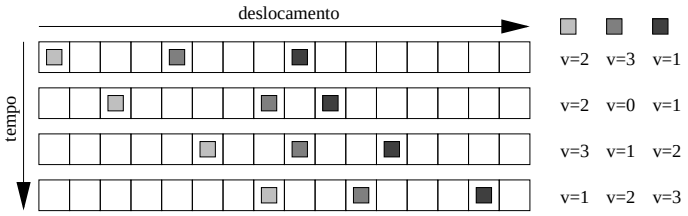

Figura 2.2: Exemplo da formação de uma zona de congestionamento.

Tabela 2.3: Tabela de funções e parâmetros para a troca de faixa.

Função	Valor retornado
$gap_{vd}(i,t)$	Distância entre o veículo i e o veículo imediatamente a frente na faixa desejada.
$\textit{visão}_{vd}(i,t)$	Distância que o veículo i enxerga à frente na faixa desejada, normalmente $gap_{vd}(i,t) = gap(i,t)$.
$\textit{visão}_{vd_{traseiro}}(i,t)$	Distância que o veículo i enxerga atrás na faixa desejada, $\textit{visão}_{vd_{traseiro}}(i,t) = v_{\textit{máxima}}$.
$p_{\textit{mudança}}$	Probabilidade de mudança: $p_{\textit{mudança}(i,t)} \in \Re \wedge 0 \leq p_{\textit{mudança}}(i,t) \leq 1$.

2. Existe espaço na outra faixa, isto é, se for efetuada a troca de faixa o veículo pode seguir sua trajetória: $gap_{vd}(i-1,t) > \textit{visão}_{vd}(i,t)$;

3. Nenhum veículo é obstruído na faixa desejada[6]: $\textit{visão}_{vd}(i-1,t) < gap_{vd}(i-1,t)$[7];

4. O veículo "deseja"[8] realizar a troca de faixa: $rand() < p_{\textit{mudança}}(i,t)$.

O exemplo da figura 2.3 ilustra estes passos, observando o comportamento do veículo *B* nota-se a aplicação das regras (supondo $p_{\textit{mudança}} = 1$). Primeiro o veículo verifica que a situação na faixa atual é insatisfatória ($gap(B,t) < \textit{visão}(B,t)$, $gap(B,t) = 2$ e $\textit{visão}(B,t) = 3$) e existe espaço na outra faixa ($gap_{vd}(B,t) > \textit{visão}_{vd}(B,t)$, $gap_{vd}(B,t) = 2$ e $\textit{visão}_{vd}(B,t) = 3$) mas o veículo *A* seria obstruído com a troca ($\textit{visão}_{vd}(B-1,t) \geq gap_{vd}(B-1,t)$[9], $gap_{vd}(B-1,t) = 1$ e $\textit{visão}_{vd}(B-1,t) = 5$) esse ciclo se repete até o passo 3, onde a ultrapassagem pode ser efetuada.

Desta forma, é possível simular o comportamento do trânsito de duas faixas. Além disso, foi observado que a capacidade das duas faixas é maior que o dobro da capacidade de apenas uma faixa. Isto ocorre pois existem ultrapassagens e a capacidade não fica mais dependente dos veículos mais lentos, além de evitar congestionamentos eventuais. No modelo assimétrico, a capacidade observada na faixa da esquerda é maior que a observada no modelo simétrico, mas a capacidade geral é menor. Isto se deve ao fato da regra 3 ser ignorada no caso assimétrico.

[6]No caso assimétrico, onde procura-se retornar à faixa da direita assim que possível, esta verificação é ignorada quando a troca se dá da esquerda para direita. No caso de troca da direita para esquerda esta regra é sempre válida.

[7]A expressão $\textit{visão}_{vd}(i-1,t)$ refere-se à visão do veículo que ficaria atrás($i-1$) caso o veículo (i) efetuasse a troca de faixa, mesmo raciocínio para $gap_{vd}(i-1,t)$.

[8]Este desejo é expresso na forma de $p_{\textit{mudança}}(i,t)$.

[9]A expressão $\textit{visão}_{vd}(B-1,t)$ refere-se ao veículo que se posicionaria atrás de B, caso este efetuasse a troca de faixa.

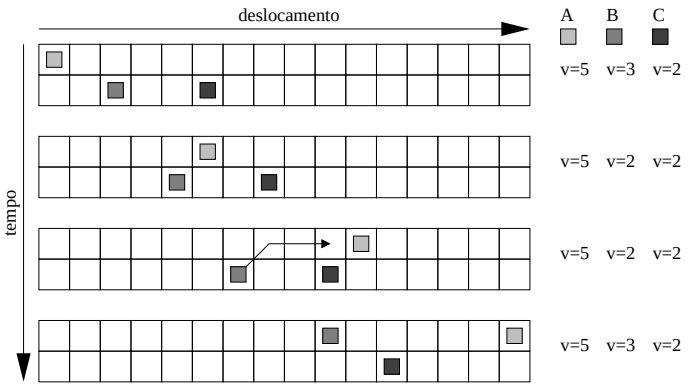

Figura 2.3: Exemplo de comportamento utilizando as regras de troca de faixas.

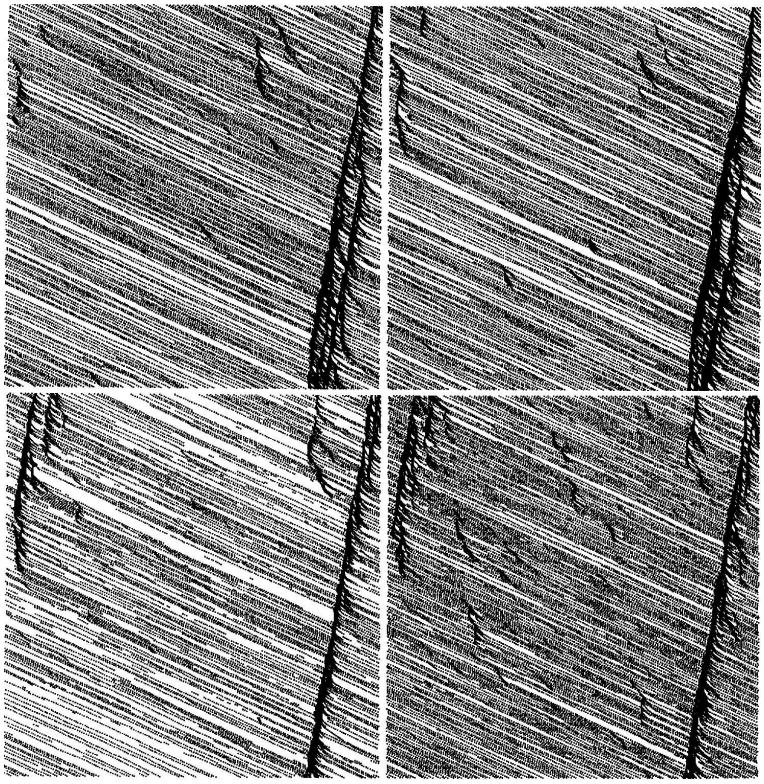

Figura 2.4: Lado esquerdo corresponde à faixa esquerda e o lado direito à faixa direita. Eixo x corresponde ao tempo e o eixo y (orientado de cima para baixo) corresponde ao espaço. No conjunto superior tem-se a aplicação das regras de simetria enquanto no inferior tem-se as regras assimétricas (a faixa da direita é a preferencial). As trocas de faixas são feitas de maneira segura, *visão*$_{vd_{traseiro}}(i) = 5$.

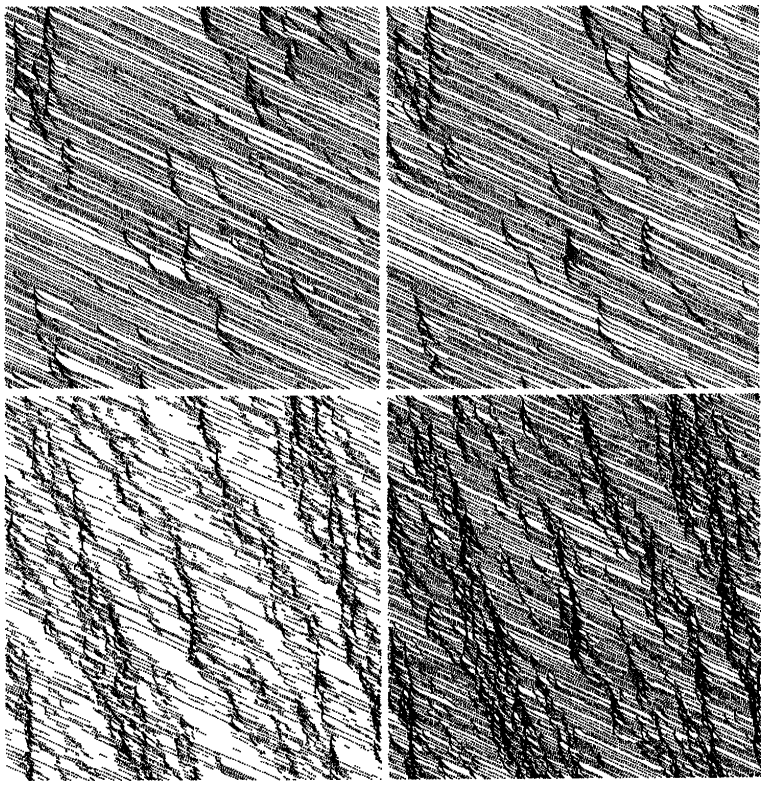

Figura 2.5: Análogo à figura 2.4, porém as trocas de faixas são feitas de maneira insegura, *visão*$_{vd_{traseiro}}(i) = 0$.

É interessante observar na figura 2.4 (extraída de (RICKERT et al., 1996)) o efeito das regras com e sem simetria, mesmo mantendo a regra que assegura a troca entre uma faixa e outra. Esta regra não é respeitada nos diagramas mostrados na figura 2.5, também extraída de (RICKERT et al., 1996).

Nos gráficos das figuras 2.4 e 2.5 o eixo horizontal representa o tempo e o vertical deslocamento, dos veículos. O lado esquerdo representa a faixa esquerda e o lado direito a faixa esquerda. Já os gráficos na parte superior representam a aplicação das regras simétricas e nos inferiores as regras assimétricas. Os "traços"correspondem, de maneira geral, a trajetória dos veículos conforme o tempo avança. Já as áreas escuras representam os congestionamentos.

2.3 Modelo de Wagner

No modelo apresentado em (WAGNER; NAGEL; WOLF, 1997) busca-se uma aproximação entre os dados empíricos e os resultantes das simulações. O conjunto de regras é bastante restrito e eficiente.

Entretanto, não deve buscar qualquer realismo em observações microscópicas (em nenhum modelo baseado no automato celular Nagel–Schreckenberg este é o objetivo) mas, ao contrário, observar o comportamento global.

As regras para troca de faixa tomam como objetivo simular a inversão de densidade entre as faixas, isto é, a faixa da esquerda, após determinada densidade, ser mais usada que a da direita (densidade na faixa da esquerda ser maior que o da direita). As regras, aplicadas antes da movimentação horizontal, são mostradas a seguir:

- Segurança:

  Caso a velocidade máxima do veículo que se aproxima (por trás) na faixa de destino seja maior que o gap resultante da troca de faixa, não há mudança, isto é, só haverá troca de faixa se a esta não provocar uma colisão.

  $v_{\textit{máxima}_{vd}}(i-1,t) \leq gap_{vd}(i-1,t)$

- Trocas da direita para esquerda[10]:

  O incentivo para efetuar a troca da direita para esquerda, tipicamente para ultrapassagem, é a necessidade de reduzir caso o veículo permaneça na mesma faixa.

  Se ($v_{\textit{máxima}}(i,t) > gap(i,t)$) e ($gap_{vd}(i,t) \geq gap(i,t)$) então efetuar a troca de faixa.

- Trocas da esquerda para direita:

  A probabilidade $p_{e\rightarrow d}$ faz o balanço entre o conjunto de regras abaixo, selecionando um deles:

  – Se ($rand() < p_{e\rightarrow d}$) então
    Se ($v(i,t) \leq gap_{vd}(i,t)$) então efetuar a troca de faixa.
  – Caso contrário
    Se ($v_{\textit{máxima}} < gap(i,t) - v_{offset}$) e ($v_{\textit{máxima}} < gap_{vd}(i,t) - v_{offset}$) então efetuar a troca de faixa.

As regras de troca da esquerda para a direita foram concebidas desta forma pois a segunda parte da regra impede a troca de faixa quando a faixa da esquerda está congestionada. Imagine o caso em que a densidade é suficientemente alta na faixa da esquerda, então a cláusula $v_{\textit{máxima}} < gap(i,t) - v_{offset}$ nunca será satisfeita (uma vez que o gap já está menor que $v_{\textit{máxima}}$) e não haverá a troca de faixa. O parâmetro v_{offset} foi introduzido para permitir o ajuste do ponto de inversão da densidade, isto é, quanto maior v_{offset} menor a quantidade de veículos necessária para ocorrer a inversão de densidade. Um esquema deste parâmetros é apresentado na figura 2.6 de (WAGNER; NAGEL; WOLF, 1997).

Além disso, é preciso permitir a troca de faixas (da esquerda para a direita) para o caso em que a densidade torna-se alta. Então o paramento $p_{e\rightarrow d}$ efetua o balanceamento entre a regra geral e outra mais relaxada onde verifica-se, apenas, se existe espaço suficiente na faixa de destino (direita). Este parâmetro, no artigo (WAGNER; NAGEL; WOLF, 1997), é baixo ($p_{e\rightarrow d} = 0.05$) para que não comprometa o efeito da inversão de densidades. A figura 2.7 mostra o gráfico empírico da inversão de densidades e a figura 2.8 mostra os dados coletados de simulações. O efeito do parâmetro $p_{e\rightarrow d}$ pode ser avaliado com o auxilio da figura 2.9 e o efeito de v_{offset}, separadamente, no gráfico da figura 2.10.

Neste modelo, como mencionado anteriormente, o objetivo é simular o comportamento da inversão das densidades. Mas foi observado em (WAGNER; NAGEL; WOLF, 1997) que este modelo possui características que o distanciam da realidade. O volume de troca de faixas em densidades mais altas é superior ao volume observado empiricamente,

[10] Assume-se que a faixa da esquerda é utilizada para ultrapassagens e, portanto, a faixa preferencial (para condução dos veículos) é a da direita.

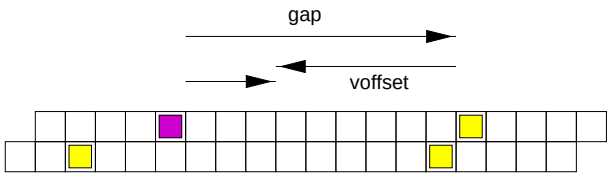

Figura 2.6: Esquema para as regras de troca de faixa.

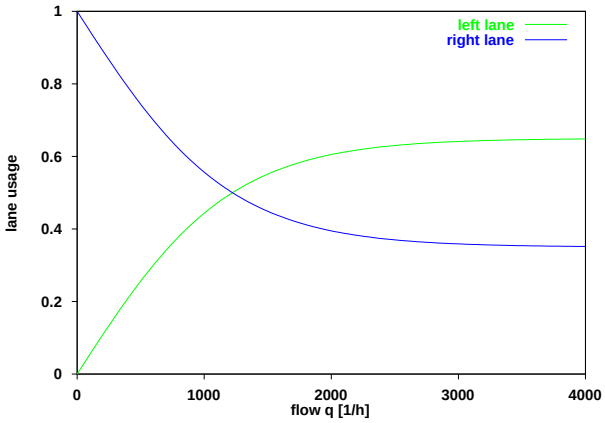

Figura 2.7: Gráfico empírico da inversão de densidades.

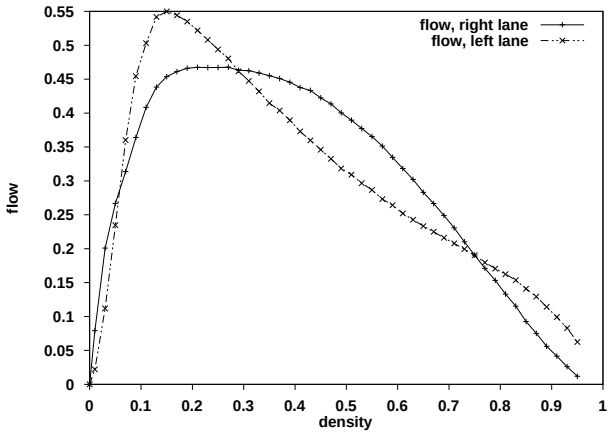

Figura 2.8: Diagrama fundamental, com $v_{offset} = 8$ e $p_{e \to d} = 0.05$, da densidade nas faixas: direita (*usage: right lane*) e esquerda (*usage: left lane*).

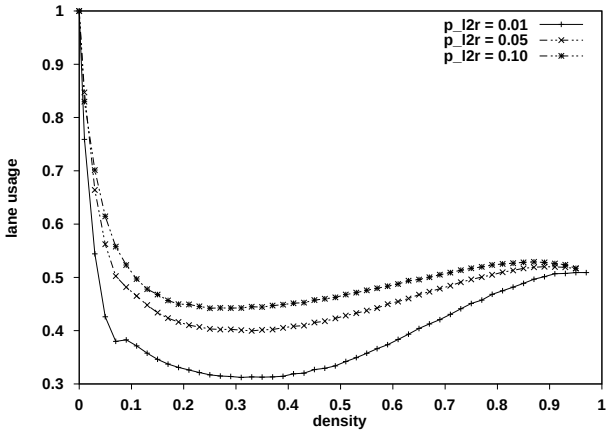

Figura 2.9: Efeito do parâmetro $p_{e \to d}$ (*p_l2r*) no uso da faixa da direita.

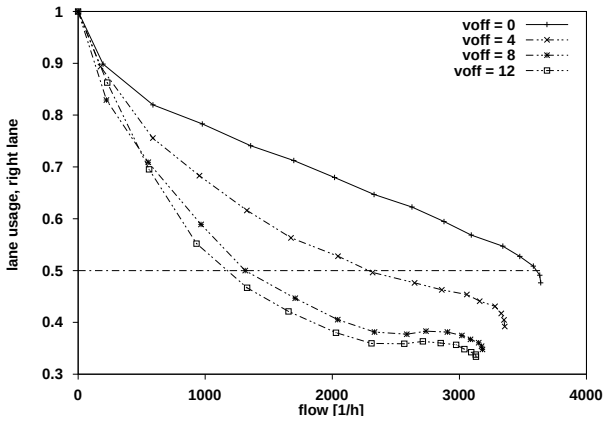

Figura 2.10: Efeito do parâmetro v_{offset} (*voff*) no uso da faixa da direita.

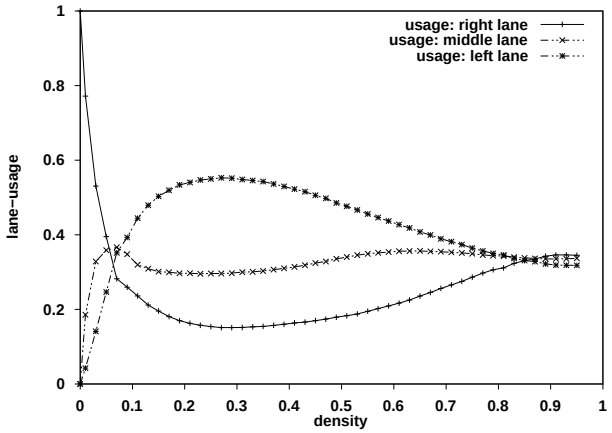

Figura 2.11: Extrapolação do modelo para três faixas.

além do efeito das trocas do tipo "ping-pong" estar presente. Adicionalmente, a capacidade máxima, na faixa da direita é maior que deveria ser, comparativamente, aos valores empíricos.

Por fim, pode-se ver na figura 2.11 uma extrapolação do modelo aplicado a uma simulação com três faixas.

2.4 Aproximação da realidade e consolidação

Buscando uma maior aproximação da realidade, algumas modificações foram propostas em (KNOSPE et al., 2003). Para tanto, várias propriedades foram incorporadas ao modelo a fim de permitir um maior realismo. Além disso, buscou-se incluir comportamentos mais verossímeis: antecipação da velocidade do veículo à frente, manter a velocidade mesmo com *gaps* menores, antecipar a diminuição da velocidade do veículo à frente (evitando colisões), atribuir a veículos parados menor aceleração que os veículos em movimento e permitir que os veículos ajustem sua velocidade a níveis compatíveis com a segurança (sem que colidam com os veículos à frente).

As propriedades adicionais incorporadas foram:

- A função de aleatoriedade da desaceleração obedece à seguinte regra, em um dado tempo t:

Tabela 2.4: Tabela de funções adicionais.

Funções	Valor retornado
$v_{antecipada}(i,t)$	Velocidade esperada pelo veículo i para o veículo à frente.
$gap_{efetivo}(i,t)$	gap esperado por i até o veículo à frente, considerando $v_{antecipada}(i)$.
$freio(i,t)$	Estado da luz de freio de i, que pode estar ligado ou desligado.

Tabela 2.5: Tabela de parâmetros adicionais.

Parâmetro	Significado
gap_{seguro}	Valor considerado seguro para o $gap(i,t)$.
$p_{desaceleração_{freio}}$	Probabilidade de desaceleração caso o veículo à frente esteja reduzindo e seja necessário reduzir.
$p_{desaceleração_0}$	Probabilidade de desaceleração caso a velocidade do veículo seja zero, $v(i,t) = 0$.
$p_{desaceleração_{padrão}}$	Probabilidade de desaceleração nos demais casos.
t_{frente}	Tempo à frente, isto é, o tempo até atingir o próximo veículo. $t_{frente}(i,t) = gap(i,t)/v(i,t)$.
$t_{frente_{seguro}}$	Tempo à frente considerado seguro. $t_{frente_{seguro}} = min(v(i,t), gap_{min})$, onde gap_{min} é um parâmetro da simulação.

$$p_{desaceleração}(i) = \begin{cases} p_{desaceleração_{freio}} & \text{se } freio(i+1,t) = 1 \text{ e } t_{frente} < t_{frente_{seguro}} \\ p_{desaceleração_0} & \text{se } v(i,t) = 0 \text{ e não } (freio(i+1,t) = 1 \text{ e } t_{frente} < t_{frente_{seguro}}) \\ p_{desaceleração_{padrão}} & \text{demais casos} \end{cases}$$

- E o $gap_{efetivo}(i)$ é calculado da seguinte forma:

$$gap_{efetivo}(i,t) = gap(i,t) + max(v_{antecipada}(i,t) - gap_{seguro}, 0)$$

Onde $v_{antecipada}(i,t) = min(gap(i+1,t), v(i+1,t))$, isto é, a previsão da próxima velocidade do veiculo à frente, considerando o modelo original do autômato celular Nagel–Schreckenberg, e o parâmetro gap_{seguro} determina a eficiência da antecipação, isto é, o quão "relaxada" será distância mínima.

O passo de simulação passa a incluir os seguintes sub-passos:

1. Cálculo da probabilidade de desaceleração $p_{desaceleração}(i,t)$:

   $p_{desaceleração} = p_{desaceleração}(i,t)$
   $freio(i,t+1) = 0$

2. Aceleração:

   Se $((freio(i+1,t) = 0)$ *e* $(freio(i,t) = 0))$ *ou* $(t_{frente} \geq t_{frente_{seguro}})$ então $v(i,t+1) = min(v(i,t)+1, v_{máxima})$

3. Redução:

   $v(i, t+1) = min(gap_{efetivo}(i), v(i, t))$

   Se $(v(i, t+1) < v(i, t))$ então $freio(i, t+1) = 1$

4. Aleatoriedade e desaceleração:

   Se $(rand() < p_{desaceleração})$ então $v(i, t+1) = max(v(i, t+1) - 1, 0)$ se $(p_{desaceleração} = p_{desaceleração_{freio}})$ então $freio(i, t+1) = 1$

5. Movimentação

   $pos(i, t+1) = pos(i, t) + v(i, t+1)$

É interessante notar que nesta abordagem as células possuem tamanho de $1,5m$ e isto faz com que os veículos possuam tamanhos maiores que 1 célula[11] e, portanto, a discretização é menor.

Observando os passos descritos pode-se perceber que primeiro calcula-se a probabilidade de desaceleração. Neste cálculo é considerado o estado dos freios e a velocidade atual[12]. Feito isso, o próximo estado dos freios é (pré) marcado como desligado (note que trata-se de $t+1$). Avançando, tem-se a aceleração, que não pode ocorrer se o veículo à frente está reduzindo (luz de freio ligada) ou o próprio veículo está travando, entretanto, se o tempo até o próximo veículo é superior ao tempo de segurança então o veículo tenta se aproximar, isto é, acelerar. No passo de redução, a velocidade de referência é o mínimo entre o $gap_{efetivo}(i)$[13] e a velocidade atual. Caso exista a necessidade de redução, isto é, a velocidade é reduzida em função do veículo à frente, então é preciso ligar a luz de freio. Para desacelerar, característica presente no modelo original, é preciso primeiro verificar se isto é necessário e, o sendo, reduzir efetivamente. Porém, existe a possibilidade de o passo de aceleração ter ultrapassado o limite seguro, $t_{frente} \geq t_{frente_{seguro}}$, e o veículo à frente ($i+1$) não estar travando assim como o este (i), $((freio(i+1, t) = 0)$ *e* $(freio(i, t) = 0))$, então se for feita uma redução é preciso ligar a luz de freio. Por último, é feita a movimentação.

Os parâmetros utilizados em (KNOSPE et al., 2003) estão expressos na tabela 2.6.

As regras propostas para a interação entre as faixas precisa atender alguns pré-requisitos como:

- regras locais simples;
- ser robusta com relação aos veículos lentos, isto é, o fluxo não pode ser regido pelos veículos lentos se eles representam uma pequena fração do total de veículos;
- reproduzir o comportamento empírico observado nas trocas faixas;
- não afetar o comportamento dinâmico dos sistemas de apenas uma faixa.

[11] Adotando o modelo original, descrito na seção 2.1, onde um veículo possui $7,5m$ tem-se que o veículo, agora, ocupa 5 células de comprimento.

[12] Os veículos são numerados com relação à direção de movimentação, isto é, o veículo $n+1$ está à frente de n. Estas referências são válidas somente dentro da faixa, cada faixa possui sua rotulação de veículos.

[13] Este parâmetro é, na realidade, uma previsão simplificada, baseada no modelo original, da próxima posição do veículo à frente, descontando uma distância de segurança (tipicamente, levando em conta o tempo de reação do ser humano).

Tabela 2.6: Tabela de parâmetros de simulação.

Parâmetro	Valor
gap_{min}	6 células
gap_{seguro}	7 células
$p_{desaceleração_{freio}}$	0.94
$p_{desaceleração_{0}}$	0.5
$p_{desaceleração_{padrão}}$	0.1
$v_{máxima}$	20 células/passo
Tamanho do veículo	5 células
Passo	1 segundo

2.4.1 Modelo simétrico

As trocas de faixas, discutidas em capítulo anterior, podem ser simétricas ou assimétricas e parte-se do princípio que os veículos precisam ser incentivados a trocar de faixa, respeitando alguns critérios de segurança. Novamente o movimento dos veículos se dá em duas etapas distintas: primeiro ocorre a troca de faixa (sem qualquer movimentação horizontal) e depois a movimentação. Além disso, aplica-se restrições às interações entre as faixas apenas aos veículos que não estão travando (luz de freio precisa estar desligada) e, é claro, se existe espaço suficiente entre o veículo e o seu sucessor (que segue) e seu precedente[14] (que está à frente), na faixa de destino. Desta forma, as regras para trocas simétricas seguem:

- Incentivo:

  $(freio(i,t) = 0)$ e $(v(i,t) > gap(i,t))$

- Critério de segurança (considerando a faixa de destino):

  $(gap_{efetivo_{destino}}(i+1,t) \geq v(i,t))$ e $(gap_{destino}(i-1,t) \geq v_{destino}(i-1,t))$

Deve-se observar que o critério de segurança refere-se à faixa de destino, isto é, considerando que o veículo será inserido na faixa de destino, calculam-se os valores. Na figura 2.12 (extraída de (KNOSPE et al., 2003)) pode-se ver um esquema que indica as medidas. Novamente pode-se observar, no gráfico da figura 2.13, que o fluxo nas faixas é muito semelhante. Um comparativo do número de trocas entre os diferentes modelos pode ser visto na figura 2.14, onde *NaSch* é o modelo original, *VDR* é o modelo onde a redução da velocidade depende da velocidade, *NaSch + anticipation* considera somente a antecipação da velocidade do veículo à frente e *NaSch + brake lights* considera apenas as regras que regem o acionamento das luzes de freio e conseqüente comportamento. É interessante notar que o uso das luzes de freio reduzem abruptamente o número de interações entre as faixas.

Pode-se implementar assimetria introduzindo o não balanceamento da ocupação das faixas por meio de diferentes tipos de veículos (principalmente com relação às suas velocidades máximas), como carros e caminhões. Infelizmente, esse tipo de não balanceamento pode comprometer as simulações, pois dois veículos lentos podem formar uma "barreira"[15] quando movimentam-se lado à lado nas duas faixas. Este tipo de "barreira"

[14] É considerada a antecipação da velocidade deste no cálculo do espaço.

[15] Expressão original: *plug*.

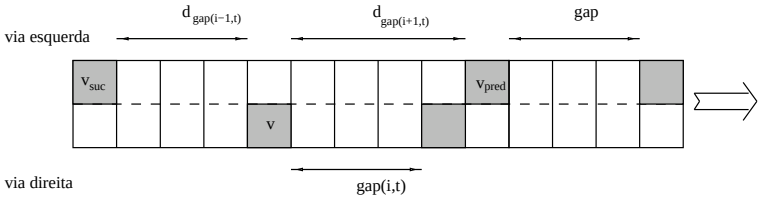

Figura 2.12: Esquema de um segmento de rua.

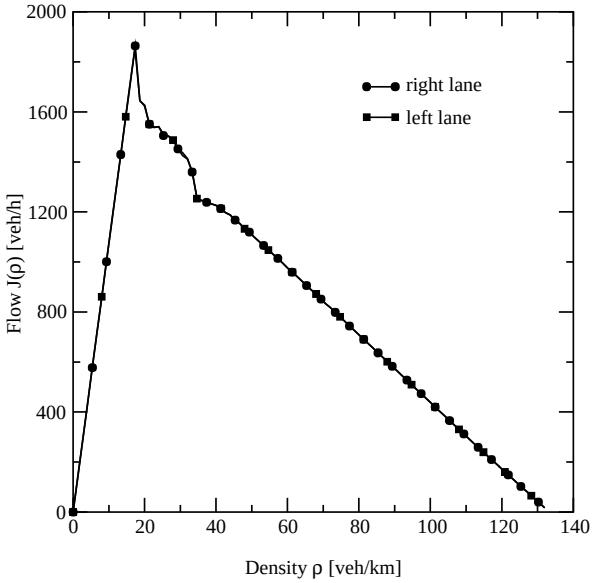

Figura 2.13: Gráfico fundamental para o caso simétrico.

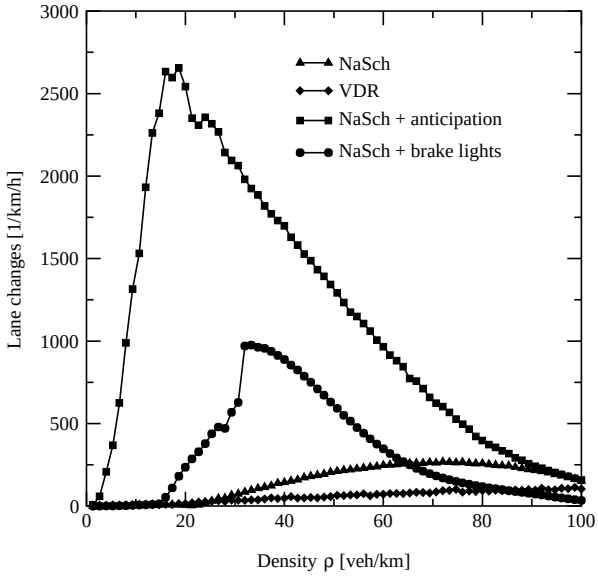

Figura 2.14: Volume de trocas de faixas para os diferentes modelos simétricos.

é bastante estável, pois se desfazem por sucessivas reduções, quando não existe congestionamento. Este tipo de assimetria, mesmo colocando todos os veículos lentos na faixa da direita e os impossibilitando de trocar de faixa, não reproduz a inversão de densidades verificada empiricamente.

2.4.2 Modelo assimétrico

Logo, para obter a inversão de ocupação das faixas é necessário introduzir a assimetria nas regras que regem a troca de faixa. As regras que regem este modelo garantem que os veículos precisam utilizar a faixa da esquerda para efetuar qualquer ultrapassagem, mas a ultrapassagem pela direita pode ocorrer, e o incentivo para trocar da faixa da esquerda para direita é reduzido. As regras do modelo assimétrico seguem:

- Direita para esquerda
  - Incentivo:
    $(freio(i,t) = 0)$ e $(v(i,t) > gap(i,t))$
  - Critério de segurança (considerando a faixa de destino):
    $(gap_{efetivo_{destino}}(i+1,t) \geq v(i,t))$ e $(gap_{destino}(i-1,t) \geq v_{destino}(i-1,t))$
- Esquerda para direita
  - Incentivo:
    $(freio(i,t) = 0)$ e $(t_{frente_{destino}}(i{+}1,t) > 3.0)$ e $((t_{frente}(i,t) > 6.0)$ ou$(v(i,t) > gap(i,t)))$
  - Critério de segurança (considerando a faixa de destino):
    $(gap_{efetivo_{destino}}(i+1,t) \geq v(i,t))$ e $(gap_{destino}(i-1,t) \geq v_{destino}(i-1,t))$

Utilizar os tempos à frente (da faixa atual e de destino) faz com que haja a inversão do uso das faixas. Isto ocorre pois um veículo troca de faixa (da direita para esquerda)

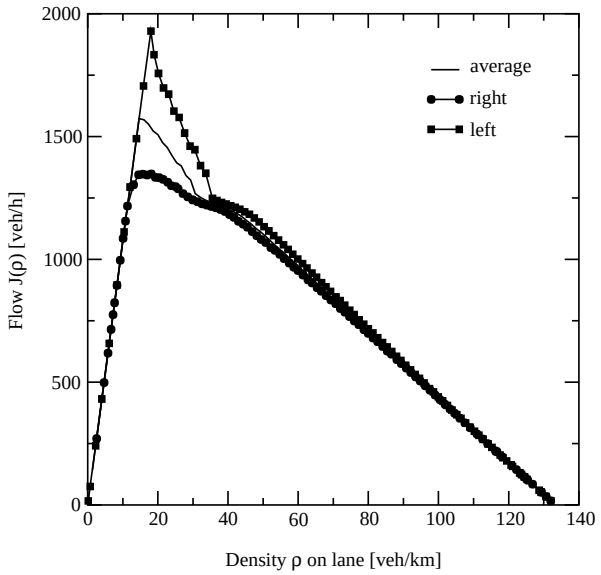

Figura 2.15: Diagrama fundamental para o modelo assimétrico.

apenas se as faixas estiverem suficientemente desocupadas (tempo até o próximo veículo, na própria faixa, superior a 6 segundos e tempo até o próximo veículo na outra faixa, de destino, superior a 3 segundos) ou se for possível efetuar uma ultrapassagem pela direita, que ocorre em densidades mais altas. Pode-se observar o comportamento do fluxo nas faixas para diferentes densidades na figura 2.15 e o uso das faixas na figura 2.16, ambas extraídas de (KNOSPE et al., 2003). É necessário, ainda, restringir o uso das faixas por veículos lentos, caminhões, de forma que estes só possam trafegar na faixa da direita[16].

2.5 Conclusão

Neste capítulo foram apresentados o modelo Nagel–Schreckenberg bem como algumas de suas extensões. Estes foram os modelos nos quais se baseiam o simulador do projeto *ITSUMO* e a plataforma *DRIVER-DFW*. Os experimentos mostrados no capítulo 6 foram realizados utilizando os modelos aqui apresentados.

[16]Isto é necessário devido a formação das "barreiras", discutido na seção anterior.

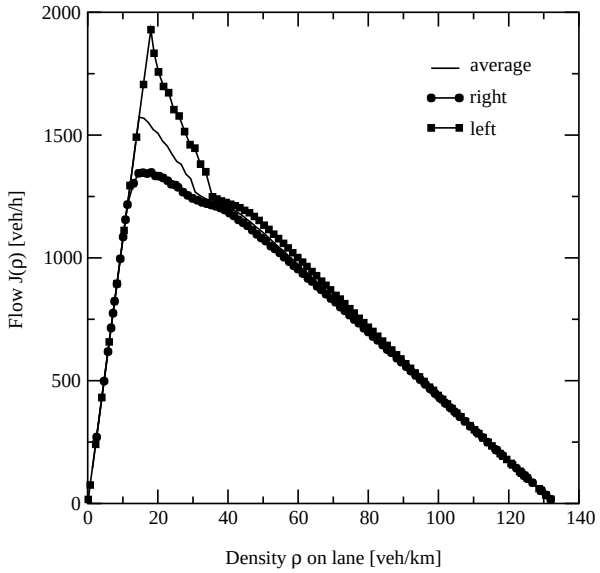

Figura 2.16: Uso das faixas para diferentes densidades.

3 SIMULADOR DO PROJETO *ITSUMO*

A versão preliminar deste simulador foi apresentada em (ANDRIOTTI, 2001) e em (ANDRIOTTI; BAZZAN, 2002) e as versão mais atuais em (SILVA et al., 2004).

3.1 Objetivos do *ITSUMO*

O principal objetivo deste simulador é fornecer uma ferramenta de simulação discreta baseada no modelo Nagel–Schreckenberg (capítulo 2). As razões pelas quais foram escolhidas estas características já foram discutidas na seção 1.1. Além disso, houve a preocupação de disponibilizar um simulador com possibilidade de expansão, isto é, que fosse possível agregar funcionalidades à ele.

Outra característica do *ITSUMO* é permitir que se armazenem as informações topológicas em um banco de dados relacional, baseado na tecnologia *SQL*. Este banco de dados permite que os dados estejam disponíveis ao usuário, para que efetue modificações ou consultas, sem a necessidade de utilizar o simulador para isso. Além disso, ele permite que se utilizem ferramentas próprias para edição dos dados contidos dentro deste banco, sem necessitar o desenvolvimento de ferramentas próprias.

Sintetizando, o *ITSUMO* tem por objetivo fornecer uma ferramenta de simulação discreta, flexível (através do uso de orientação a objetos) e eficiente (com o uso de *C/C++*). Adicionalmente, concede liberdade ao usuário na manipulação dos seus dados, através do uso de um sistema de banco de dados baseado em *SQL*.

É importante observar que existe um contexto maior no qual a solução apresentada está inserida. Todo o ambiente se baseia em simulações de agentes em um mundo discreto. Todo e qualquer uso da ferramenta precisa observar estas restrições para melhor aproveitar os recursos disponíveis.

3.2 Restrições de projeto

Este trabalho foi fundamentado em algumas premissas. Estas premissas impõem algumas restrições à plataforma de implementação de motoristas *DRIVER-DFW*. Entre elas está o fato de se utilizar um simulador discreto.Isto significa que tempo e espaço são discretos.

Utilizar uma ferramenta computacional para análise de problemas é uma alternativa bastante atraente tanto do ponto de vista de modelagem quanto econômico. No ambiente de trânsito não é diferente. Para estudar o comportamento de apenas alguns veículos pode-se utilizar a experimentação, isto é, empregar alguns veículos e motoristas reais é viável. Mas estudar um cenário que envolva um número elevado de veículos em um

cenário mais amplo eleva a dificuldade logística, econômica e de observação através da experimentação.

Outra alternativa seria o uso de um modelo analítico, isto é, descrever o cenário a ser estudado através de um conjunto de equações (contínuas). Entretanto expressões analíticas são bastante restritas quando necessita-se efetuar alterações estruturais no ambiente. Estas alterações normalmente requerem a reformulação do modelo analítico. Modelar qualquer elemento de tráfego, quando for possível usar esta abordagem, exigirá um conhecimento mais elaborado de matemática e análise matemática, restringindo o conjunto de pessoas aptas a utilizar o ambiente. Outro fator de incentivo no uso de simulações são os resultados obtidos com eles. Na literatura científica, em (NAGEL; SCHRECKENBERG, 1992) por exemplo, é mostrado que os resultados das simulações é equivalente, ou superior, aos resultados dos modelos analíticos, isto é, através da descrição matemática do mesmo cenário por meio de um conjunto de equações.

Utilizar simulações discretas permite um aumento de desempenho, sem perda significativa da qualidade dos dados obtidos, ao se observar o comportamento macroscópico. O desempenho é fator relevante quando simulações de uma malha viária grande são necessárias ou mesmo permitir que se utilize a ferramenta de simulação para realizar extrapolações. Um dos usos para as simulações é extrapolar rapidamente, na ordem de segundos, um estado corrente para fornecer uma previsão para os próximas horas, por exemplo.

3.3 Arquitetura

A arquitetura do simulador do projeto *ITSUMO*, pode ser vista através do diagrama de classes da figura 3.1. É interessante notar que a classe responsável por representar o motorista não faz parte da árvore de derivação das demais classes, *TopologyElement*. Esta característica foi adotada pois permite que o motorista tenha a mínima dependência possível em relação aos objetos do simulador.

É importante mostrar quais abstrações estas classes representam. Alguns elementos reais têm sua representação na forma de um objeto de uma determinada classe bastante intuitiva, mas isto não é regra. Um exemplo de como os elementos reais são mapeados dentro das estruturas do simulador pode ser visto na figura 3.2.

Além dos elementos retratados na figura 3.2 há outros, como os objetos do tipo *Source* e do tipo *Sink*. Estes são elementos artificiais que se encarregam de gerar veículos, no caso do *Source*, e de removê-los, no caso do *Sink*, da simulação. Apesar de não estarem relacionados a elementos reais, eles são necessários para conferir dinamismo às simulações.

3.4 Preparação de uma simulação

Para se iniciar uma simulação é necessário preparar o ambiente. Isto significa que é necessário buscar no banco de dados os parâmetros da simulação desejada e criar os objetos apropriados. A ordem em que os eventos ocorrem internamente, de uma maneira bastante simplificada, pode ser acompanhada na relação de ítens a seguir.

1. Estabelece conexão com o banco de dados;

2. Criação do objeto do tipo *Settings*, que irá conter todas as configurações da topologia a ser simulada;

3. Os objetos do tipo *Network*, que contém a malha viária a ser simulada;

4. O núcleo da plataforma de implementação de motoristas *DRIVER-DFW*, apresentada no capítulo 5: *Helper*;

5. Os objetos *Node*, que representam os cruzamentos;

6. Os objetos *Street*, que abstraem o conceito de uma rua ou avenida;

7. Os objetos *TrafficLight*, que representam os semáforos;

8. Os objetos *Source*, que injetam veículos na simulação;

9. Os objetos *Sink*, que absorvem e eliminam os veículos que deixam a simulação;

10. Ajuste dos pesos de cada direção, conforme o cruzamento;

11. Ajuste das possibilidades de direções a serem adotadas nos cruzamentos;

12. Solicitação de atualização dos estados internos do objeto *Helper*, conforme a topologia criada;

13. Liberação do ambiente para o início da simulação.

Se faz necessário detalhar alguns passos mencionados na lista acima, pois sua semântica não é clara. A criação dos objetos é bastante simples e apenas faz a criação do objeto conforme as especificações do banco de dados.

Para o passo que atribui os pesos de cada direção, uma vez que se deseja realizar simulações baseadas na realidade, é necessário reproduzir o fluxo de veículos nas faixas conforme o observado, ou estimando, a partir do cenário real. Para isso, tem-se disponível, em maior ou menor abstração dependendo da fonte dos dados, a quantidade de veículos que seguem para cada um dos possíveis destinos em um cruzamento. Isto fica mais claro com o diagrama da figura 3.3.

Nesta figura apenas os possíveis destinos estão etiquetados. A cada etiqueta está associada uma informação de fluxo, isto é, algum dado que tenha alguma relação com a quantidade de veículos que se dirigiu a algum dos possíveis destinos. O cruzamento pode ser caracterizado como um sistema supostamente fechado, isto é, todos os veículos que "entram" também "saem" e nenhum veículo é inserido ou removido no cruzamento. Com esta propriedade é possível atribuir pesos probabilísticos a cada possível destino de uma faixa e seu respectivo cruzamento.

Esta probabilidade fornece uma orientação à escolha de rota dos motoristas, a fim de reproduzir a distribuição de fluxo desejado. Por exemplo, deseja-se simular o fluxo de uma determinada avenida e para isso os veículos oriundos das ruas transversais precisam, em sua grande parte, desviar sua trajetória para dentro da avenida. É justamente este tipo de informação que está presente nestas probabilidades.

No passo de ajuste dos cruzamentos ocorre a aplicação das orientações e restrições em relação as possibilidades de um veículo em um determinado cruzamento. Um exemplo destas restrições pode ser visto na figura 3.4 onde nem todas as possibilidades são permitidas[1].

[1]Assume-se que estas restrições são imposições das autoridades de trânsito.

No passo de ajuste dos estados internos do objeto *Helper* são feitas as aquisições de informações topológicas. Estes aspectos serão tratados em profundidade na seção 5.2.

Por ora basta saber que neste passo o *Helper* lê a topologia, por varredura nos objetos da simulação, coletando informações. Estas informações são os parâmetros de todos os objetos, além da construção do grafo que representa a topologia. Com isto os elementos da plataforma consultam o *Helper* em vez do simulador em busca de informações à respeito da topologia e seus componentes.

3.5 Passo de simulação

O passo de simulação obedece a seguinte ordem de eventos:

1. Efetuar as ações pendentes nos cruzamentos, objetos do tipo *Node*;
2. Informar aos veículos que solicitem novas decisões de movimentação;
3. Solicitar aos veículos que apliquem as decisões tomadas pelos motoristas;
4. Solicitar que o *Helper* execute seus procedimentos de atualização de estado.

As ações pendentes nos cruzamentos normalmente incluem ativar os elementos que regulam o fluxo, *Source* e *Sink*, e notificar os semáforos da passagem de tempo. No caso dos objetos do tipo *Source*, significa verificar se o passo é adequado para inserir novos veículos na simulação. Já o *Sink* encarrega-se de eliminar os veículos que nele chegaram no último passo de simulação. Os semáforos realizam as suas atualizações de estado, que podem incluir a troca de fase ou troca de plano semafórico.

Como a implementação de algum motorista pode não respeitar as regras de trânsito, primeiro são solicitadas as suas decisões. O simulador em nenhum momento permite colisão de veículos e por isso as decisões são auditadas antes de serem executadas. Feita a verificação das decisões, e possíveis correções, estas são aplicadas aos veículos.

Por fim, o *Helper* é informado da passagem do tempo. Com esta informação o *Helper*, pode efetuar os seus procedimentos temporais. Nesta seção, como já foi dito antes, não serão tratados os detalhes da plataforma para implementação de motoristas apresentada.

3.6 Conclusão

O simulador mostra-se bastante abrangente e flexível. As suas estruturas básicas fornecem a possibilidade de se simular a movimentação veicular em um conjunto bastante amplo de cenários sem perda significativa de fidelidade. Além disso, o fato de ser baseado no modelo Nagel–Schreckenberg lhe confere uma maior eficiência computacional.

Adicionalmente, no simulador do projeto *ITSUMO* não está previsto um modelo de motoristas. Nele apenas o motoristas padrão do modelo Nagel–Schreckenberg foi implementado sem qualquer preocupação em disponibilizar outros tipos de motoristas. Esta deficiência faz com que qualquer modelo de motorista, implementado diretamente no simulador, configure-se numa solução *ad hoc*.

Para solucionar esta deficiência foi criada a plataforma para implementação de motoristas chamada *DRIVER-DFW* e que é apresentada no capítulo 5.

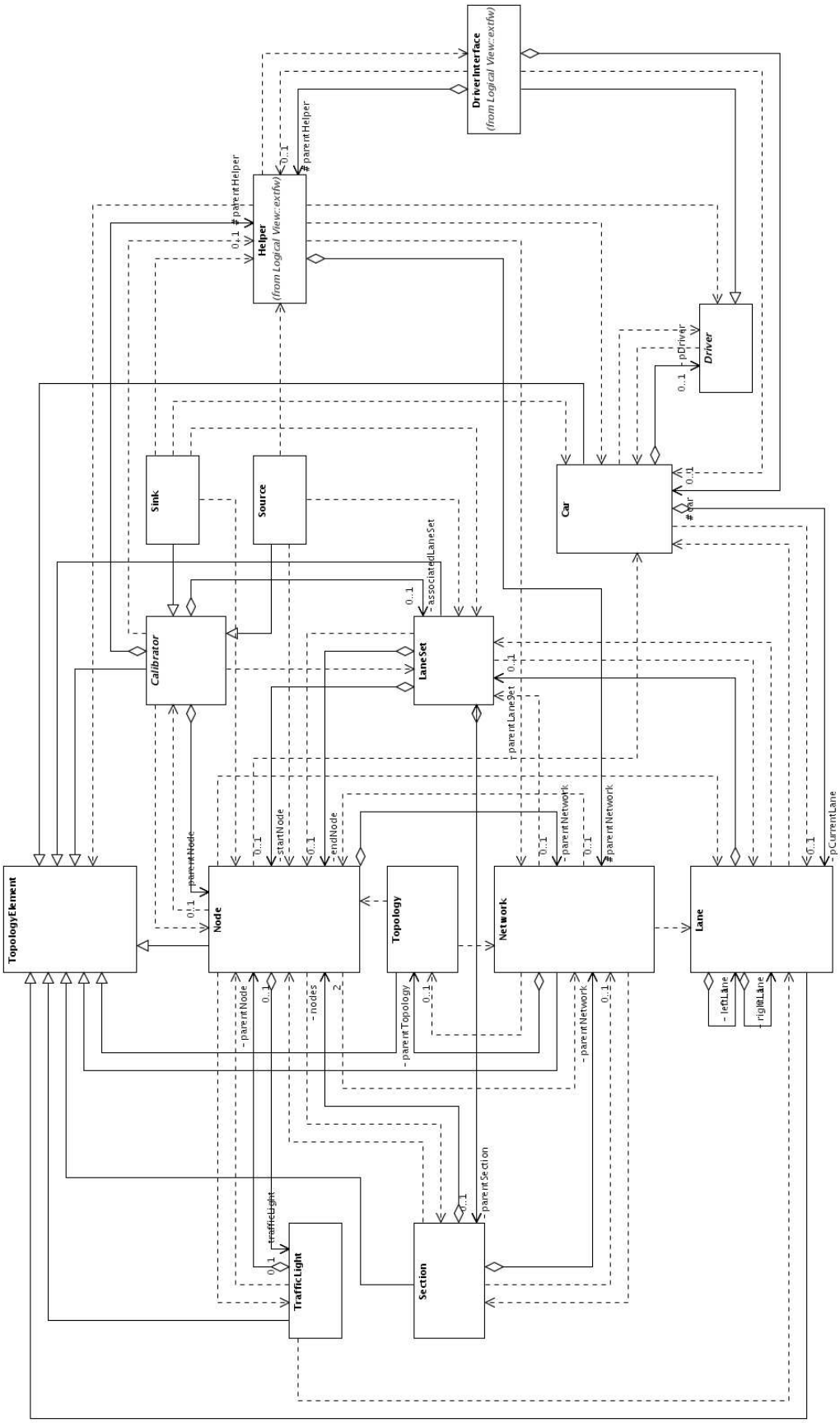

Figura 3.1: Diagrama UML das classes do simulador do projeto *ITSUMO*.

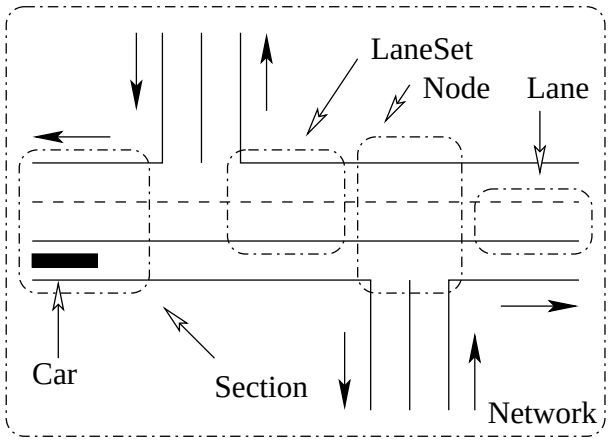

Figura 3.2: Exemplo de como elementos reais são mapeados para as estruturas do simulador.

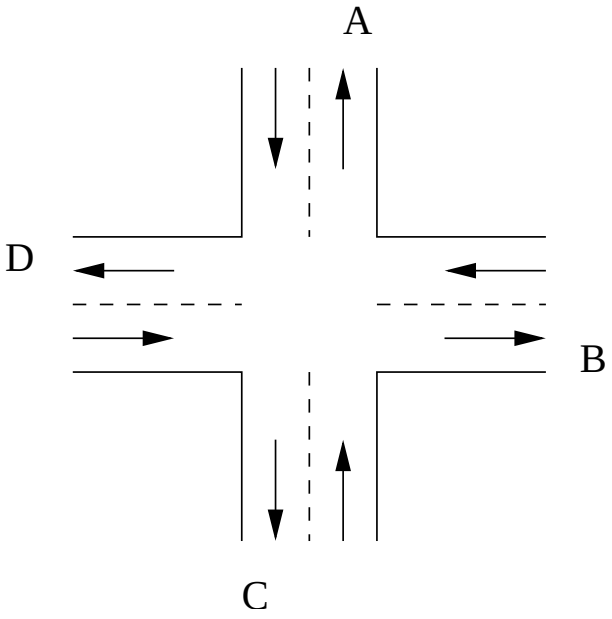

Figura 3.3: Exemplo de cruzamento com suas direções de destino etiquetadas com as letras *A, B, C, D*.

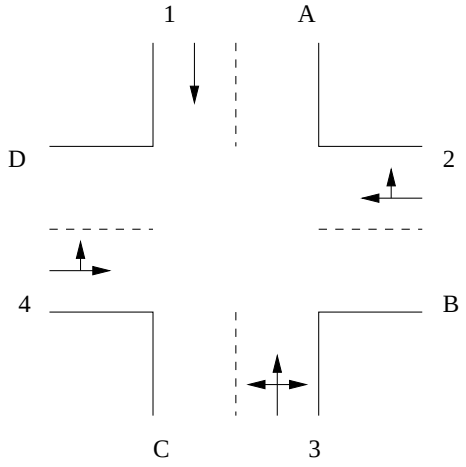

Figura 3.4: Exemplo de restrições em um cruzamento. Destinos etiquetados com as letras *A, B, C, D* e origens etiquetadas com os números *1, 2, 3, 4*.

4 MODELOS DE ESCOLHA E PLANEJAMENTO DE ROTA

No capítulo 2 foram apresentados os modelos de movimentação baseados no modelo Nagel–Schreckenberg. Neste serão apresentados alguns modelos de heurísticas para planejamento e escolha de rota um dos quais será posteriormente referenciado na validação da metodologia para modelagem de motoristas deste trabalho. Estes não são os únicos e será apresentada apenas uma amostra dos muitos modelos disponíveis.

As heurísticas que serão apresentadas buscam simular o comportamento humano no que diz respeito à escolha de rota. São modelos que buscam descobrir os mecanismos que determinam a escolha de um motorista por determinada rota. Busca-se, além disso, verificar que fatores podem influenciar nesta decisão e que efeito produz na decisão final.

Serão vistas duas abordagens diferentes, dentre as muitas existentes, para a escolha de rota. A primeira é baseada na teoria de jogos onde existe um cenário de decisão binário. Neste cenário o motorista deve buscar uma rota eficiente para ir de um ponto ao outro da rede. Na segunda abordagem se mostra o uso da lógica *BDI* para a escolha de rota.

4.1 Modelos de heurísticas de planejamento baseadas na teoria de jogos

Nas três subseções seguintes será visto o emprego de heurísticas baseadas na teoria de jogos para escolha binária de rotas. Esta escolha se dá entre duas rotas distintas que conduzem os motoristas de casa para o trabalho e vice-versa[1].

As simulações apresentadas nestas subseções foram originalmente executadas dentro do ambiente *SeSAm*[2], apresentado em (KLÜGL; PUPPE, 1998).

4.1.1 Aprendizagem de rota por heurística simples

A proposta inicial, apresentada em (WAHLE et al., 2000) e reimplementada em (KLÜGL; BAZZAN, 2002), é fundamentada no princípio de que o comportamento humano no tráfego é baseado na maximização de seus ganhos, ou seja minimizar o tempo de percurso. Além disso, baseia-se no fato de que todos os agentes são racionais e buscam atingir este objetivo e, para isso, precisam ter a habilidade de antecipar a ação dos demais participantes.

O cenário apresentado é bastante simples e o agente motorista precisa escolher entre duas rotas, onde a principal é a mais eficiente (consome-se menos tempo para percorrê-la) que a secundária. O equilíbrio[3] (o tempo consumido na rota principal é mesmo que

[1]Trata-se do cenário chamado, na literatura, de *commuting scenario*.

[2]Disponível em: *http://www.simsesam.de/*.

[3]O equilíbrio no trânsito é descrito pelos princípios de Wardrop (WARDROP, 1952).

o gasto para percorrer a secundária) é atingido quando $1/3$ dos 18 motoristas optam pela rota secundária e $2/3$ a principal. O ganho é dado pela fórmula 4.1.

$$\textit{ganho}(i) = 40 - \begin{cases} 6 + 2 \times \textit{número}_{\textit{principal}} & \textit{se escolheu a rota principal} \\ 12 + 3 \times \textit{número}_{\textit{secundária}} & \textit{se escolheu a rota secundária} \end{cases} \tag{4.1}$$

Para optar entre uma das rotas o agente gera um número aleatório, entre 0 e 1, e compara com a sua heurística, que é a probabilidade de adotar a rota principal. A heurística é calculada pela equação 4.2.

$$\textit{heurística}(i) = \frac{\sum_t \textit{ganho}_{\textit{principal}}(i,t)}{\sum_t \textit{ganho}_{\textit{principal}}(i,t) + \sum_t \textit{ganho}_{\textit{secundária}}(i,t)} \tag{4.2}$$

É importante salientar que existe uma realimentação positiva no *ganho*(i), pois quanto mais um jogador utiliza uma determinada rota mais ele aprende sobre aquela rota. Observe que os agentes são inicializados com *heurística*$(i) = 0.5$, isto é, 50% de probabilidade de escolher a rota principal e 50% de adotar a rota secundária.

O aprendizado significa a reavaliação da probabilidade de escolher a rota principal em função dos ganhos obtidos. Este aprendizado ocorre com a uma freqüência f, que um fator probabilístico que determina quando será efetuada a reavaliação.

A dinâmica da simulação ocorre da seguinte forma:

1. Escolha da rota: sortear um número aleatório, entre 0 e 1. No caso de ser superior à heurística do agente, a rota secundária é escolhida. Caso contrário será adotada a rota principal. De forma mais clara:

$$\textit{rota} = \begin{cases} \textit{principal} & \textit{se } rand() \leq \text{heurística}(i) \\ \textit{secundária} & \textit{se } rand() > \text{heurística}(i) \end{cases} \tag{4.3}$$

2. Reavaliação da heurística: caso seja o momento de reavaliar a estratégia deve-se recalcular a heurística conforme a equação 4.2. Para determinar o momento de reavaliação é utilizada a freqüência de reavaliação f.

A atribuição do ganho dos agentes é realizada a cada passo de simulação e o ganho é distribuído conforme a função *ganho*(i) descrita na equação 4.1.

Os parâmetros $\textit{número}_{\textit{principal}}$ e $\textit{número}_{\textit{secundária}}$ são o número de agentes que adotaram a rota principal e secundária, respectivamente. Existem 18 agentes participando da escolha, a cada rodada, que são submetidos à esta decisão 200 vezes, sendo o equilíbrio atingido quando 12 agentes estão na rota principal e 6 na secundária. Este parâmetros foram adotados em função de configurações semelhantes adotadas no projeto *SURVIVE*[4].

4.1.2 Adaptação da heurística

Com a intenção de melhorar a heurística proposta em (KLÜGL; BAZZAN, 2002) foi proposto em (BAZZAN; KLÜGL, 2003) uma extensão que busca disponibilizar, por parte do simulador (simulando algum serviço de informação de trânsito), uma previsão de ganho aos motoristas. Desta forma, os agentes que utilizam esta informação podem reavaliar suas decisões testando-as e, caso julguem necessário, alterá-las. As regras básicas deste novo algoritmo estão descritas a seguir:

[4]O projeto *SURVIVE* pode ser consultado em *http://www.traffic.uni-duisburg.de/survive/*

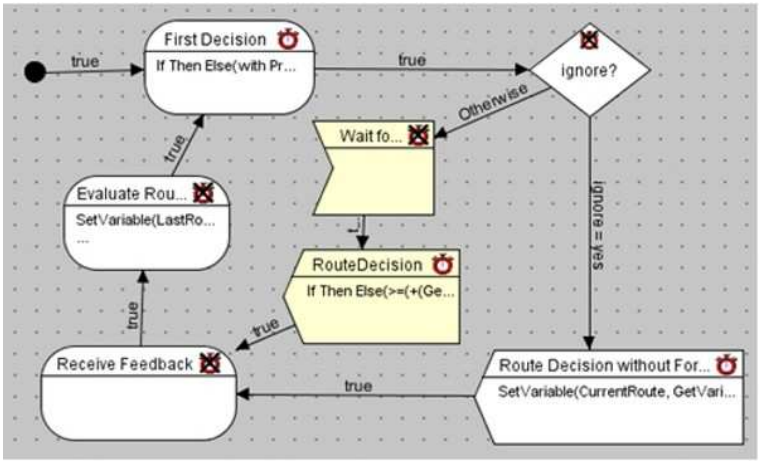

Figura 4.1: Comportamento no motorista, implementado no *SeSAm*.

1. Informa a escolha (preliminar) de rota ao simulador;
2. O simulador recebe a decisão de todos os motoristas e calcula o ganho atribuído a cada rota;
3. O motorista recebe o ganho calculado para a sua decisão;
4. Com base nos ganhos da rota nas rodadas passadas verifica-se, com uma certa tolerância, se o ganho estimado pelo simulador é aceitável, isto é, se está abaixo ou não do ganho esperado[5];
5. Caso o ganho estimado pelo simulador não o satisfaça, a informação de previsão de ganho é ignorada e uma nova escolha é feita; caso contrário, ele mantém a escolha preliminar.

Obviamente nem todos os agentes estão cientes da possibilidade de testar suas decisões. Os motoristas que ignoram a previsão informada pelo simulador, isto é, não testam suas escolhas, são denominados motoristas ignorantes ou de ruído. Um esquema do algoritmo implementado é mostrado na figura 4.1, extraída de (BAZZAN; KLÜGL, 2003).

O comportamento com diferentes quantidades de motoristas "inteligentes" (que reavaliam suas decisões) e diferentes tolerâncias, dentro das quais o motorista não modifica a sua decisão, é mostrado na tabela da figura 4.2, extraída de (BAZZAN; KLÜGL, 2003).

Resumindo, a tabela da figura 4.2 mostra que a maior estabilidade (melhor distribuição dos motoristas) ocorre quando se tem 50% de ignorantes e uma tolerância de 5 pontos no ganho obtido com a avaliação preliminar. É curioso observar que em um ambiente com 100% de agentes "inteligentes" o ambiente se comporta praticamente de forma aleatória. Isto mostra que não é interessante ter um ambiente onde todos estejam sujeitos às mesmas informações.

4.1.3 Diferentes tipos de informações

Em (KLÜGL; BAZZAN; WAHLE, 2003) a idéia apresentada em (BAZZAN; KLÜGL, 2003) é extrapolada no sentido de se fornecer diferentes tipos de informações. É impor-

[5]Como o ganho é baseado em tempo de percurso, interessa a minimização do tempo.

tolerância 1		quantidade de ignorantes	tolerância 1	
motoristas na rota principal	motoristas na rota secundária		heurística final	prêmio esperado
9.11	8.84	0	1.06±0.02	7.40
10.41	7.54	0.25	0.80±0.10	8.78
11.33	6.61	0.5	0.67±0.32	9.25
11.84	6.10	0.75	0.67±0.32	9.77
12.25	5.70	1	0.68±0.42	9.58

tolerância 3		quantidade de ignorantes	tolerância 3	
motoristas na rota principal	motoristas na rota secundária		heurística final	prêmio esperado
11.49	6.45	0	0.73±0.30	8.92
11.48	6.47	0.25	0.70±0.30	9.29
11.84	6.11	0.5	0.68±0.28	9.71
12.01	5.93	0.75	0.67±0.33	9.61

tolerância 5		quantidade de ignorantes	tolerância 5	
motoristas na rota principal	motoristas na rota secundária		heurística final	prêmio esperado
12.34	5.61	0	0.68±0.31	9.23
12.32	5.63	0.25	0.67±0.29	9.29
11.94	6.01	0.5	0.68±0.25	9.83
12.11	5.83	0.75	0.67±0.28	9.27

Figura 4.2: Tabela de dados coletados para diferentes parâmetros de tolerância e quantidade de motoristas ignorantes.

tante salientar que nas simulações aqui apresentadas é utilizado o modelo Nagel–Schreckenberg para a movimentação dos veículos e não uma função de ganho (que retorna o tempo de percurso como a da equação 4.1). Outra diferença é que os motoristas precisam pagar pelas informações que adquirem, cujos os custos estão associados ao grau de dificuldade de seu cálculo. Os agentes dependem de um orçamento limitado. Além disso, do seu ganho é descontado o custo na aquisição da informação. A relação dos tipos de informações disponíveis são:

- Nenhuma informação: o motorista opta por não adquirir qualquer informação e com isso não consome seu capital.
- Congestionamento na rota escolhida: o motorista recebe uma informação binária (presença ou ausência de congestionamento). Foi estipulado que uma rota é dita congestionada se, ao menos, n veículos estão com velocidade zero ao longo do percurso.
- Tempo de viagem na rota escolhida: todos os motoristas têm o seu tempo de viagem computado. Quando um agente solicita esta informação é informado o tempo consumido pelo último veículo na rota em questão.
- Velocidade média na rota escolhida: é informada a velocidade média, calculada em função dos veículos que já trafegaram nas rotas, para as duas rotas. Logo, a rota com velocidade média maior é a mais indicada.
- Densidade na rota escolhida: é informado a densidade, dada em número de veículos por unidade métrica.

A dinâmica da simulação utilizada foi a de distribuir, uniformemente, a probabilidade de comprar alguma das informações (foi incluída a probabilidade de não comprar qualquer informação, isto é, comprar a informação nula) e o *ganho(i)* foi modificado para

contemplar o custo associado à compra de informações. Além disso, os agentes são dotados de orçamento restrito, isto é, pode acontecer de o agente não ter crédito suficiente para comprar nenhuma informação.

$$\textit{ganho}(i) = (T_{\textit{médio}} - T_i) \times F_1 - \textit{custo} \times F_2 \tag{4.4}$$

Onde: $T_{\textit{médio}}$ é o tempo médio calculado em função dos últimos n veículos que realizaram o percurso, T_i é o tempo de viagem do agente i, *custo* é o custo associado às informações adquiridas pelo agente e F_1 e F_2 são fatores de ganho/perda e custo, respectivamente.

Para avaliar a qualidade das informações compradas, isto é, registrar o sucesso relativo à compra da informação é preciso ponderar as próximas compras. O método adotado é o seguinte:

$$\Delta p_{k_i} = \frac{\textit{ganho}_i}{T_{\textit{médio}} - T_{\textit{mínimo}}} \times \begin{cases} p_{k_i} & :\textit{ganho}_i < 0 \\ 1 - p_{k_i} & :\textit{ganho}_i > 0 \end{cases} \tag{4.5}$$

Onde: p_{k_i} é a probabilidade do agente i comprar a informação k, Δp_{k_i} é o aumento/diminuição no peso da informação k para o agente i (os demais pesos são diminuídos/aumentados proporcionalmente), $T_{\textit{mínimo}}$ é o tempo mínimo necessário para percorrer a rota adotada ($T_{\textit{mínimo}} = \textit{tamanho da rota}/v_{\textit{máxima}}$). As simulações são inicializadas com pesos iguais para todas as probabilidades.

A conclusão deste experimento foi que o desempenho dos agentes que optaram por adquirir a informação de velocidade média foi o pior. Além disso, a informação a respeito de congestionamento normalmente informava que as duas rotas estavam congestionadas, sem efeito sobre a tomada de decisão. E, por fim, que o tempo de viagem do último motorista forneceu o melhor ganho aos agentes que compraram esta informação.

4.2 Modelo de heurística de planejamento baseada na lógica *BDI*

Nesta seção é tratado o uso de agentes *BDI* na escolha de rota. O foco da discussão está na escolha entre possíveis rotas, conectando um ponto (fixo) a outro (fixo). Desta forma, busca-se simular o ambiente onde um motorista precisa deslocar-se durante o dia para ir da sua casa para o trabalho e vice-versa.

Inicialmente será apresentado o funcionamento geral da plataforma de implementação e a abordagem utilizada, seguida de alguns aspectos da proposta de implementação e integração com a plataforma de implementação de agentes *BDI AgentSpeak(L)* (RAO, 1996).

4.2.1 Plataforma e proposta

Nos artigos (ROSSETTI et al., 2000) e (ROSSETTI et al., 2000) apresenta-se a proposta de implementação e mostra-se a modelagem dos elementos de trânsito em agentes. Os elementos mais simples podem ser vistos como agentes reativos, entre eles atuadores, detectores (de fluxo) e outros controladores simples. Para modelar os motoristas adotou-se o uso de lógica *BDI* uma vez que estes precisam de maior capacidade de cognição para planejar de forma pró-ativa e avaliar as suas decisões.

Utilizando a abstração de sistemas multiagentes é possível modelar a interação entre o ambiente (representado pela malha viária, isto é, as faixas, ruas, cruzamentos, etc.) e

os motoristas (agentes *BDI*) e sistemas *ITS*[6], como mostrado no esquema da figura 4.3, extraída de (ROSSETTI et al., 2000).

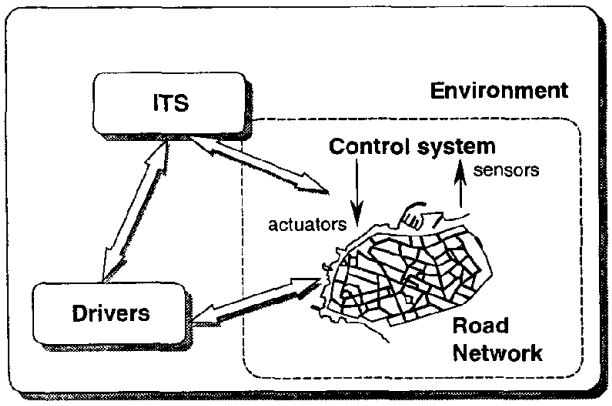

Figura 4.3: Abstração do sistema viário sob a visão de sistema multiagentes.

Os agentes motoristas foram idealizados como agentes baseados em utilidades[7] (da mesma forma que os apresentados na seção anterior), uma vez que os motoristas tomam as suas decisões de acordo com a relação custo / benefício (utilidade). Na interação entre os agentes motoristas não existe dependência direta, e, portanto, não precisam uns dos outros para tomarem suas decisões. Com isso, a sua interação se restringe à percepção mútua (não existe troca de informações/mensagens entre os motoristas).

Com isso os motoristas restringem a sua aquisição de informações àquelas fornecidas pelas interações com os sistemas *ITS* (como condições de tráfego em determinada rua, se esta informação estiver disponível) e com o ambiente, isto é, sua percepção das condições apresentadas no passado.

4.2.1.1 Plataforma DRACULA

A plataforma *DRACULA*[8] aborda a tomada de decisão em dois aspectos: tomada de decisões no dia[9] e dia-a-dia[10] (mostrado na figura 4.4, de (ROSSETTI et al., 2000)). O primeiro módulo trata das decisões durante o percurso, que tem impacto nas percepções do agente a respeito de sua decisão, e o segundo da evolução do estado da malha viária a cada dia.

No módulo identificado por DEMAND os motoristas são representados individualmente e suas decisões são tomadas (rota e horário da partida). Estes agentes (motoristas) são entregues ao módulo de simulação (identificado por SUPPLY) que, efetivamente, simula o ambiente. Terminada a simulação do dia, os agentes são alimentados com as percepções experimentadas naquele dia para formular a decisão do dia seguinte e o ciclo se repete por n dias. Esta dinâmica pode ser melhor visualizada na figura 4.5, também de (ROSSETTI et al., 2000).

Entretanto é necessário estender a plataforma para prever a interação entre os motoristas e os sistemas *ITS*, além de abordar os elementos da simulação como um SMA. Além

[6]Sistemas de transporte inteligentes, do inglês: *Intelligent Transportation Systems*.

[7]Expressão original: *utility-based agents*.

[8]Desenvolvido no Instituto de Estudos de Transportes (Institute for Transport Studies), Universidade de Leeds, Reino Unido.

[9]Expressão original: *within-day decision-making.*

[10]Expressão original: *day-to-day.*

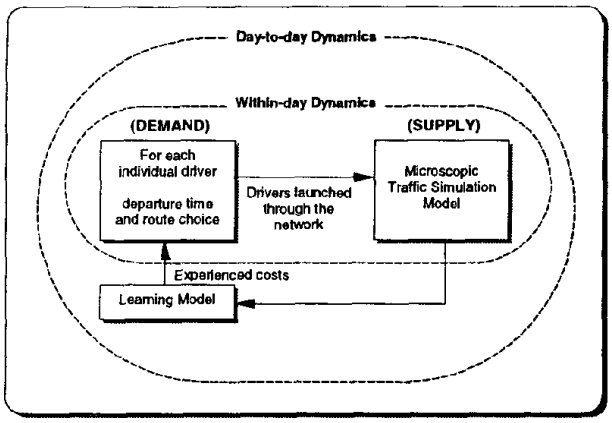

Figura 4.4: Relações básicas no modelo *DRACULA*.

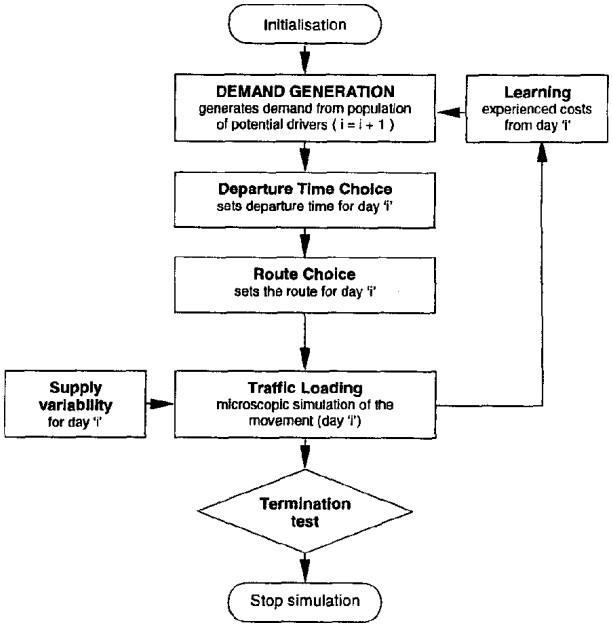

Figura 4.5: Estrutura básica da plataforma *DRACULA*.

disso, a estrutura do agente (motorista) precisa prever a interação com o ambiente e o *ITS*.

4.2.2 Proposta de implementação

A proposta de implementação (ROSSETTI et al., 2002) se utiliza de *AgentSpeak(L)*, apresentada em (RAO, 1996), como método de descrição de agentes *BDI*. Além disso, pretende-se usar a implementação, de *AgentSpeak(L)*, apresentada em (MACHADO; BORDINI, 2001)[11]. Não é foco deste texto apresentar a linguagem *AgentSpeak(L)* que pode ser vista em (RAO, 1996).

O SMA utilizado é descrito por $< N, A, ITS >$ onde N é o conjunto de segmentos[12], A o conjunto de agentes e ITS a tecnologia de *ITS* utilizada. Todo os segmentos são orientados (indicando o sentido da movimentação dos veículos). Cada viagem é especificada pela tupla $< p, z_{org}, z_{dst}, t_{dst}, t_{org}, r >$, onde p representa o propósito da viagem (trabalho, casa, parque, etc.), z_{org} representa o local de origem da viagem, z_{dst} o destino, t_{dst} o tempo desejado para a chegada, t_{org} o tempo agendado para iniciar a viagem e r a rota escolhida. A rota é uma lista de segmentos que o veículo precisa percorrer para ir de z_{org} para z_{dst}.

Os agentes armazenam o estado de cada segmento pelos quais passaram. Esta informação é guardada de forma qualitativa, isto é, o agente lembra somente se o segmento em que transitou apresentou-se em um dos seguintes estados $\{livre, normal, congestionado\}$. O estado $livre$ é atribuído quando o agente atingiu a velocidade desejada, ou superior; $normal$ se a velocidade esteve entre a desejada e um determinado limite, adotado como $20km/h$, e $congestionado$ se a velocidade esteve abaixo do limiar estabelecido.

A definição dos agentes é dada pela tupla $< E, D, P, I, A, S_E, S_O, SI >$ onde E é o conjunto de eventos, D o conjunto de desejos, P são os planos do agente, I as intenções, A o conjunto de ações possíveis, S_E a função de seleção de eventos, S_O a função de seleção das opções de planos possíveis e S_I a função de seleção das intenções. Esta definição se aplica somente aos agentes motoristas. O agente motorista satisfaz seus objetivos quando sua localização atual é igual à z_{dst}. Um exemplo de conjunto de crenças pode ser visto na figura 4.6, extraída de (ROSSETTI et al., 2002).

4.2.2.1 Integração com SIM_Speak

É necessário integrar o ambiente *SIM_Speak*, apresentado em (MACHADO; BORDINI, 2001), com a plataforma *DRACULA*. Adicionalmente, é previsto um sistema *ITS* como parte da simulação. O objetivo de tal plataforma é avaliar o impacto que informações a respeito do trânsito têm no comportamento dos motoristas. O funcionamento da plataforma final, mostrada na figura 4.7 (de (ROSSETTI et al., 2002)), é o seguinte:

- O módulo *MA Initialisation* gera uma população de agentes para a matriz de origens e destinos, matriz *OD*;
- É gerado um conjunto de rotas possíveis a cada agente;
- Após uma execução preliminar da simulação, para que se possa construir um conjunto de crenças, é atribuído a cada agente um conjunto de crenças;

[11] Disponível em : http://www.inf.ufrgs.br/ bordini/SIM_Speak/ .
[12] Cada segmento representa um trecho de uma pista de uma rua que liga um cruzamento à outro.

```
adjacent(h, link1).        adjacent(link1, link2).
adjacent(link2, w).        adjacent(h, link3).
adjacent(link3, link4).    adjacent(link4, link7).
adjacent(link7, w).        adjacent(h, link5).
adjacent(link5, link6).    adjacent(link6, link7).

route(h, w, 30, [link1, link2]).
route(h, w, 45, [link3, link4, link7]).
route(h, w, 35, [link5, link6, link7]).

state(link5, jammed).      purpose(4, work).
state(link3, free).        purpose(5, home).
state(link7, jammed).      purpose(6, leisure).

today(4).                  timeNow(0800).

preTripInformationSystem(user).
enRouteInformationSytem(non_user).
acceptanceWillingness(20).

perceivedArrivalCost(10, work).
usualDepartureTime(h, w, 0815).
expectedTravelTime(h, w, 999).

tripRoute(h, w, [link1, link2], usual).
tripDepartureTime(0815, [link1, link2]).
```

Figura 4.6: Exemplo de conjunto de crenças de um agente motorista.

- Os agentes tomam suas decisões (utilizando ou não o sistema *ITS* disponível, neste caso *ATIS*), que são armazenadas no arquivo *Input MA*;
- A simulação é executada conforme as decisões dos agentes, não é permitida alteração de rota durante a execução e os custos resultantes de cada rota (cada segmento com o seu custo associado) são gerados no arquivo *Output MA*;
- Os agentes atualizam suas crenças, em função dos dados de custos associados a cada segmento da rota selecionada, e estão aptos a formular novas decisões para o dia seguinte.

Com isso é possível utilizar a tecnologia de agentes, utilizando a lógica *BDI*, para estudar o trânsito.

4.3 Conclusão

Foram vistos alguns exemplos de abordagens para modelagem do planejamento de rotas. Estas abordagens diferem entre si tanto no método de representação do conhecimento como na forma como este conhecimento é adquirido. Pode-se perceber, com apenas poucos exemplos, que modelar o comportamento humano no planejamento de rota é uma tarefa mais complexa que a movimentação, vistos no capítulo 2.

Esta diversidade de opções mostra o quão flexível precisa ser a metodologia e a ferramenta de modelagem de motoristas para ser considerado de propósito geral.

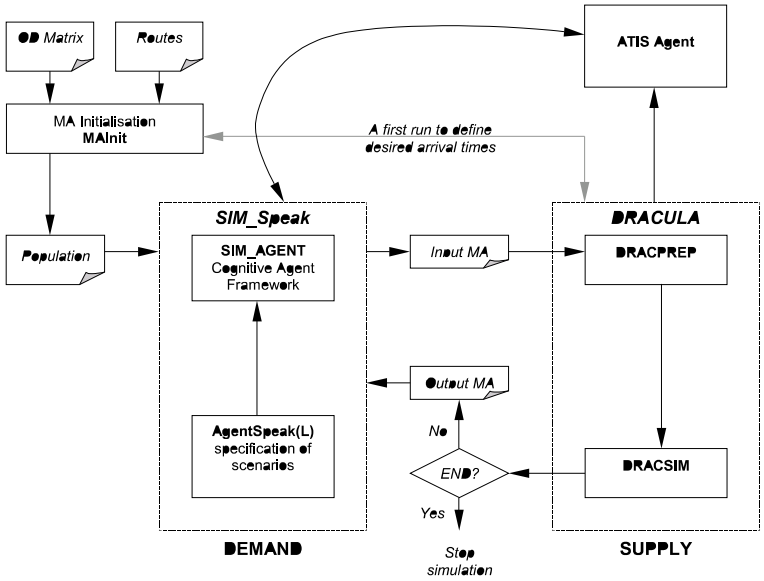

Figura 4.7: Plataforma utilizando *SIM_Speak* junto com o *DRACULA*.

5 *DRIVER-DFW* UMA PLATAFORMA PARA IMPLEMENTAÇÃO DE MODELOS DE MOTORISTAS

Conforme apresentado anteriormente, existia a demanda para se adicionar uma plataforma para a modelagem de motoristas ao simulador microscópico baseado em automato celular (seção 3.6). Além disso, conforme visto nas seções anteriores, esta plataforma deve permitir a modelagem das mais diversas heurísticas a serem aplicadas ao modelo de motorista. Este capítulo trata da arquitetura e funcionamento desta plataforma. Serão apresentados aspectos da implementação, como as tecnologias e técnicas utilizadas[1].

Esta plataforma, como dito anteriormente, se utiliza de um simulador e parte de sua estrutura apresentada em (ANDRIOTTI, 2004).

5.1 Arquitetura da plataforma para implementação de motoristas *DRIVER-DFW*

A plataforma para implementação de motoristas *DRIVER-DFW* visa facilitar a modelagem de motoristas e para tal ferramenta foi desenvolvida uma arquitetura. Esta arquitetura é dotada de uma série de estruturas que visam facilitar o processo de desenvolvimento e interação entre os seus elementos. Nesta seção serão detalhados os motivos para tal arquitetura e como ela funciona.

Explorar as possibilidades desta arquitetura permite que se adote as mais diversas abordagens para os problemas que se apresentarem. É evidente que a tecnologia de agentes é mais adequada para esta arquitetura. Os agentes de controle, se existirem, devem ser implementados nos módulos auxiliares e os agentes que "povoam" o ambiente serão os motoristas. A decisão de adotar tal abordagem cabe ao pesquisador.

Observar o diagrama UML das classes da plataforma para implementação de motoristas auxilia na sua compreensão, este diagrama se encontra da figura 5.1. Neste diagrama tem-se a classe principal, o núcleo da plataforma para implementação de motoristas *DRIVER-DFW*, o *Helper* e uma série de outras estruturas que dele dependem. Outras duas estruturas importantes são os módulos auxiliares, derivados da classe *AbstractHelperModule*, e módulos motoristas, derivados da classe *AbstractDriverModule.*

5.1.1 Classe *Helper*

O núcleo da plataforma para implementação de motoristas *DRIVER-DFW* é a classe *Helper*, que atua como intermediária entre os componentes da *DRIVER-DFW* e os com-

[1]O código fonte está totalmente documentado e através da ferramenta *Doxygen*disponível em http://www.doxygen.org/.

ponentes do simulador. Por ser o ponto de contato, este componente acaba centralizando uma série de tarefas e concentrando as tarefas de controle de praticamente todas as interações entres os elementos da simulação e da *DRIVER-DFW*.

Todas as interações entre os componentes da plataforma ocorrem pelo intermédio do *Helper*. Além destas interações, ele ainda agrega uma série de funcionalidades que servem como utilitários aos usuários da plataforma. Maiores detalhes do *Helper* podem ser encontrados na seção 5.2.

5.1.2 Classe *DriverInterface*

Outro elemento importante é a classe *DriverInterface* que estende a classe *Driver* do simulador[2]. Cada veículo é dotado de um objeto da classe *DriverInterface*. Cada objeto desta classe mantém em seu estado algumas informações de apoio à tomada de decisão do motorista, além do motorista ao qual está vinculado.

Estas informações de apoio, que os objetos da classe *DriverInterface* mantém, são o estado atual e anterior (correspondente ao passo anterior) do veículo além da decisão tomada no passo anterior. Com estas informações o motorista deve ser capaz de formular as decisões mais simples de movimentação e planejamento. Sempre que um objeto do tipo *DriverInterface* perde seu veículo ele é removido (apagado da memória) e seu motorista informado desta remoção. Este procedimento é importante para o controle de uso de memória do sistema.

É importante mostrar como esta classe é acionada pelo simulador. Os métodos mostrados são os chamados pelo simulador a fim de solicitar que os motoristas interajam com os seus veículos. Não é o objetivo explicar todos os tipos de dados envolvidos mas mostrar que estes métodos são pouco significativos para quem não está familiarizado com o código do simulador. Será visto, na seção 5.1.6 como estes métodos são tratados para que chamem os métodos de mais alto nível do motoristas. Os métodos são os seguintes:

```
decision_t takeDecision(Car &carRef)

lane_laneset_pair_t chooseNextLane_Laneset(Car *carRef)
```

5.1.3 Classe *HelperGraph*

O *HelperGraph* é na realidade uma implementação de grafos baseado em lista de incidência. Neste grafo os vértices são os cruzamentos, ou objetos do tipo *Node*. Já os arcos são os segmentos de rua, uma para cada sentido. Isto significa que trata-se, na verdade, de um dígrafo.

Nesta classe encontra-se também uma versão modificada do algoritmo Dijkstra, apresentado em (DIJKSTRA, 1955), para encontrar o menor caminho entre um ponto e outro. Esta funcionalidade está disponível a todos elementos da plataforma.

5.1.4 Classe *ParseConfFile*

Na classe *ParseConfFile* está disponível um leitor de arquivo de configuração. Apesar do arquivo de configuração não estar disponível aos elementos da plataforma, os seus parâmetros estão. Desta forma, qualquer componente da ferramenta pode consultar parâmetros de configuração que sejam de seu interesse e estejam presentes no arquivo.

[2]No jargão de orientação a objetos isto signifi ca que *DriverInterface* é uma classe derivada de *Driver*.

5.1.5 Módulos

Para agregar funcionalidades e novos motoristas se utiliza a tecnologia de módulos dinâmicos. Estes módulos são carregados em tempo de execução e isto lhes confere a propriedade de se alterar o conjunto de módulos sem que seja necessário refazer a plataforma. Assim, o pesquisador pode desenvolver vários módulos, tanto de apoio quanto de motoristas, e agregá-los às simulações apenas alterando parâmetros de configuração da simulação. Isto permite que sejam feitas experimentações e novos desenvolvimentos apenas criando estes elementos, módulos dinâmicos. O uso de módulos e as tecnologias envolvidas são discutidas na seção 5.3.

5.1.6 Classe *AbstractDriverModule*

O módulo motorista, derivado da classe *AbstractDriverModule* e onde deve ser implementado o modelo de motorista a ser simulado, possui uma interface bastante simples. Este módulo precisa responder, resumindo a sua funcionalidade, a dois estímulos externos: decisão de movimentação e decisão de planejamento. A decisão de movimentação, discutida na seção 5.6, é estimulada a cada passo de simulação pelo objeto da classe *DriverInterface* ao qual está vinculado. Os detalhes de funcionamento dos módulos motoristas podem ser encontrados na seção 5.5.

É importante mostrar como a plataforma "traduz" as chamadas de método citadas na seção 5.1.2. Como o objetivo da plataforma é facilitar a implementação as interações do motoristas com o simulador e vice-versa os métodos apresentados na seção 5.1.2 são traduzidos para os seguintes métodos:

```
extfw::driver_decision_t takeLocalDecision(\
                extfw::car_state_t *carCurrentStateRef,\
                extfw::car_state_t *carLastStateRef,\
                extfw::driver_decision_t *driverLastDecisionRef)

extfw::lane_laneset_decision_t chooseNextDestination(\
                extfw::car_state_t *carCurrentStateRef,\
                extfw::car_state_t *carLastStateRef,\
                extfw::driver_decision_t *driverLastDecisionRef)
```

É possível observar que as informações são entregues, aos motoristas, com um tratamento de mais alto nível. Isto cria uma camada de abstração entre o simulador e a plataforma que facilita o seu uso.

5.1.7 Classe *AbstractHelperModule*

Pode-se adicionar funcionalidades à plataforma ou ao simulador através dos módulos auxiliares, derivados da classe *AbstractHelperModule* e discutidos em mais detalhes na seção 5.4. Estes módulos respondem, basicamente, a apenas um estímulo: requisição de informações. Estas requisições podem ser as mais diversas possíveis, como por exemplo calcular o desvio padrão das velocidades de uma determinada faixa ou enviar uma mensagem a todos os agentes presentes.

Os beneficiados mais evidentes com estas funcionalidades são os motoristas. Sempre que alguma funcionalidade se tornar mais genérica, isto é, mais de um agente a necessita, ela pode ser implementada em um módulo auxiliar. Assim, todos os elementos se

beneficiam desta funcionalidade. Os próprios módulos auxiliares podem utilizar a funcionalidade de outros auxiliares para complementar suas tarefas. Isto possibilita uma abordagem descentralizada para resolução de problemas.

5.2 Núcleo da plataforma para implementação de motoristas *DRIVER-DFW*

O núcleo da plataforma para implementação de motoristas *DRIVER-DFW* é representado pela classe *Helper*. Esta classe acumula uma série de funções de controle e algumas funcionalidades. As tarefas mais importantes são a comunicação entre os componentes da ferramenta e os componentes do simulador. É também função sua coordenar e intermediar todas as interações entre os componentes da plataforma, gerir o uso dos módulos e prover um conjunto de funcionalidades mínimas.

É através do *Helper* que os motoristas e módulos auxiliares percebem o ambiente onde estão inseridos. A manutenção de uma visão, padronizada, de todos os elementos do simulador é de sua responsabilidade. Com base nestas informações é feita a tomada de decisão dos motoristas.

A classe *HelperGraph* é responsável por manter o dígrafo que representa a topologia da simulação. O *Helper* provê uma interface de comunicação entre os módulos e este dígrafo. No *Helper* está a responsabilidade de manter este dígrafo consistente.

Outra funcionalidade mínima é fornecer um leitor de arquivo de configurações. Isto é feito pelo *ParseConfFile* que faz a leitura de um arquivo de configurações e mantém os parâmetros especificados, no arquivo de configuração, disponíveis para consulta. Assim como ocorre com o *HelperGraph*, o *Helper* também mantém uma interface de comunicação entre os módulos e o *ParseConfFile*.

5.2.1 Busca de informações topológicas

As informações topológicas correspondem às características da malha viária representadas dentro da simulação. É nelas que se encontram, por exemplo, as definições da quantidade de faixas de uma rua ou quantas ruas atingem um determinado cruzamento.

A alimentação dos registros de informações topológicas ocorre apenas na inicialização da simulação. Como o *Helper* conhece o elemento *Network* do simulador ele efetua uma pesquisa por todas as estruturas topológicas deste. Esta busca é feita através de varredura na estrutura de dependência entre os objetos, uma espécie de árvore de objetos. Com esta varredura ocorre a alimentação das estruturas que servirão de visão para os módulos da plataforma.

Tomou-se o cuidado de não utilizar, na composição desta visão, qualquer tipo de dado dependente das estruturas do simulador. Desta forma, todos os ponteiros são substituídos pelos seus identificadores únicos, por exemplo. Os tipos de dados particulares são substituídos por estruturas padronizadas e fixas dentro da plataforma. Assim, é criada uma abstração entre os módulos e o simulador.

Esta abstração é importante para que modificações no simulador não impliquem em refazer módulos. Permitir que módulos sejam utilizados com relativa independência da versão do simulador é bastante desejável. Desta forma o pesquisador pode efetuar atualizações no simulador sem comprometer o trabalho já realizado nos módulos.

Esta transparência tem um custo computacional. Cada consulta ao simulador deve ser intermediada pelo *Helper* e isto implica no custo de tradução. Esta tradução pode ser simples, apenas conversão de tipos de dados, ou mais trabalhosa como na busca do objeto

ao qual pertence um determinado identificador.

Em favor da transparência e facilidade para o pesquisador optou-se por arcar com os custos extras desta abstração.

5.2.2 Gerência dos módulos

Está no *Helper* a tarefa de gerir os módulos dinâmicos. Para a tarefa de carga dos módulos existe a classe *ModuleLoader*. Esta é a classe responsável por buscar, através do *Helper*, quais módulos devem ser carregados e efetuar sua carga. Caso ocorra algum problema, o usuário é notificado através de uma exceção lançada na detecção da falha.

A função do *ModuleLoader* é efetuar a carga dos módulos e prover meios para o *Helper* manipulá-los. Nesta classe ocorre toda a etapa de manipulação da carga dinâmica e os processos de criação e de destruição de objetos definidos nos módulos. Toda e qualquer criação e remoção de objetos, oriundos dos módulos, deve ocorrer no *ModuleLoader*. Assim sendo, o *Helper* fica responsável apenas por controlar os objetos entregues pelo *ModuleLoader*.

O *ModuleLoader* conhece apenas a classe *ModuleElement*, de onde derivam todos os demais módulos. Além disso, ele tem a capacidade de criar e remover um objeto de determinado módulo, mesmo sem conhecer a classe final à qual o objeto pertence. Claro que esta interface de criação e remoção de objetos é fixa para todos os módulos, independente da implementação que for adotada.

Por isso, se as estruturas mínimas e rígidas dos módulos não forem seguidas, o *ModuleLoader* não poderá efetuar a carga dos módulos. As exigências mínimas incluem a derivação da classe se dar a partir a classe *ModuleElement* e ter disponível a forma fixada de criação e remoção dos objetos. Isto garante que o módulo seja carregado e entregue, sem problemas, ao *Helper*.

Apesar de muito semelhantes, os módulos auxiliares têm tratamento diferenciado do aplicado aos módulos motoristas. O *Helper* cria apenas um objeto de cada módulo auxiliar e intermedia a sua utilização. Já os módulos motoristas podem ter vários objetos ativos, conforme a necessidade, e também são separados em duas categorias. As categorias são motoristas ruído, utilizados pelos objetos *Source* para justamente gerar ruído. Outra categoria são os motoristas do tipo objetivo de estudo, cuja quantidade é especificada em arquivo de configuração. Os módulos auxiliares são detalhados na seção 5.4 e os módulos motoristas na seção 5.5.

5.3 Utilização de módulos dinâmicos

Bibliotecas dinâmicas são a tecnologia adotada para flexibilizar a utilização da arquitetura usando módulos. Esta tecnologia permite que se desenvolva funcionalidades com pouca dependência do programa principal, ou o executável final. De forma simplificada, o programa principal conhece apenas a interface de comunicação e invoca as funcionalidades das bibliotecas através desta interface comum.

Por particularidades do compilador são necessárias duas funções bem específicas em todos os módulos. Estas funções são responsáveis por criar e destruir um objeto. Seu código é bastante simples e apenas cria um novo objeto, conforme o módulo que as acompanha, e devolve o seu ponteiro ao requisitante. Ou, no caso de destruição do objeto, limpa a área de memória ocupada pelo objeto, respeitando seu tipo. A necessidade desta intermediação se dá devido ao chamado *name mangle* do *C++* dentro do *GCC*, que foi o compilador utilizado. Não se conhece, até o momento, nenhum compilador *C++* padrão

que apresente um contorno para esta particularidade.

Apesar desta restrição que força os usuários da plataforma a observarem esta particularidade, foi adotada esta solução. A possibilidade de desenvolver um código fora do simulador, de forma independente, foi um pré-requisito do projeto. Esta característica confere à plataforma uma flexibilidade muito grande. Isto é, funcionalidades podem ser adicionadas ou retiradas, sendo necessário apenas reiniciar as simulações.

Além disso, motoristas podem ser desenvolvidos e testados em diferentes combinações necessitando apenas reiniciar as simulações. Assim, experimentos podem ocorrer paralelamente alterando apenas os parâmetros de inicialização do simulador. Outra facilidade é permitir que se corrija qualquer problema em qualquer módulo sem que isso afete os demais ou mesmo o simulador.

Então, o tempo de desenvolvimento e os custos de manutenção são reduzidos. Diminui-se também a possibilidade de alguma alteração implicar em efeitos colaterais imprevistos. Isto é, algum código afetar alguma outra parte com a qual não está diretamente relacionado. Um exemplo seria uma variável global, definida em um módulo "A", ser modificada em alguma manutenção e isto alterar o comportamento de um módulo "B" de forma inesperada.

Muito se argumentou a respeito das vantagens, mas existem desvantagens. O contraponto está justamente nas questões de desempenho e custos extras. Quando se utiliza uma biblioteca dinâmica o custo em memória aumenta, são inseridos códigos extras para permitir que o programa principal encontre os códigos dos módulos. Outro custo é em relação ao desempenho, pois a primeira carga dos módulos, isto é, quando forem utilizados pela primeira vez, será mais demorada. Isto ocorre pois o programa precisa procurar pelo recurso solicitado e verificar se ele está dentro das especificações. Por este motivo, a carga de todos os objetos oriundos dos módulos é efetuada previamente, exceto a dos motoristas ruídos, que são gerados sob-demanda.

Por fim, é preciso lembrar que é estabelecida uma confiança entre os módulos e os demais componentes da plataforma. Ela estabelece que os módulos irão efetivamente realizar as tarefas as quais se propõe, visto que nenhuma verificação minuciosa é feita nas funcionalidades dos módulos. Por isso, se algum módulo apresentar algum problema crítico, ele pode comprometer toda a simulação. É evidente que este problema se apresentaria independente da utilização de módulos, mas é preciso estabelecer que esta segurança não é oferecida.

5.4 Módulos auxiliares

Como já foi dito antes os módulos auxiliares são responsáveis por estender as funcionalidades da plataforma. É através deles que se deve preencher as lacunas que venham a se apresentar na ferramenta. Para isto foi necessário estabelecer uma interface de comunicação que fosse ao mesmo tempo simples e suficientemente poderosa.

Atingir a simplicidade é necessário para que qualquer usuário seja capaz de utilizar a interface. Ser robusta e poderosa é necessário para que a solução seja útil mesmo para atender às necessidade mais complexas. Por isso, a interface criada é constituída de apenas dois métodos, ou atuadores. Cada método tem sua função bastante específica, um para controle de tempo e outro para atender às requisições externas.

Para tarefas que necessitem de algum controle de tempo existe o método de controle *update*. Através deste método, chamado pelo *Helper* ao final de cada passo de simulação, deve ser feito o controle de tempo do módulo. Para isto, este método recebe como

parâmetro o passo simulação corrente.

Atender às requisições externas é tarefa do método *query*. Assim, toda e qualquer interação com os módulos auxiliares é feito através da chamada do método *query*. Este método possui uma particularidade que lhe confere flexibilidade, onde não se tem um número fixo de parâmetros.

5.4.1 Parâmetros variáveis em *query*

Para quem está familiarizado com programação em *C/C++* este recurso é o mesmo utilizado na função básica *printf*. Para ser mais específico deve-se observar a assinatura[3] do método do *Helper* que intermedia a chamada de *query* no módulo auxiliar:

```
bool queryModule (extfw::module_type_id_t moduleID,\
                  extfw::helper_module_query_id_t queryID, ...)
```

O parâmetro "`...`" é a parte variável do argumento de consulta de um módulo auxiliar. Isto quer dizer que a quantidade de argumentos extras neste método é indeterminado, de zero até quanto se queira (respeitando os limites do compilador).

Além de especificar o módulo a ser consultado, pelo parâmetro `moduleID`, apenas o identificador da consulta, parâmetro `queryID`, precisa ser informado. Esta abordagem é bastante atraente pois deixa a cargo do desenvolvedor especificar a quantidade e os tipos de cada parâmetro necessários a cada uma das consultas as quais ele responde. Por este motivo não são necessários outras formas de comunicação entre o módulo e os elementos externos, que podem ser outros módulos auxiliares ou motoristas.

Apesar de bastante poderoso, este recurso apresenta um problema bastante perigoso: nenhuma verificação é feita em relação aos parâmetros variáveis. O desenvolvedor precisa confiar que quando uma consulta for efetuada ela receberá a quantidade de argumentos necessários nos tipos adequados. Infelizmente não há recursos que permitam realizar qualquer verificação neste sentido. Tudo está baseado na confiança da correta utilização dos recursos.

Levando em consideração estes problemas, acredita-se que o recurso é aceitável. A liberdade e flexibilidade ganhas com esta solução permitem que exista apenas um método de interação com o mundo externo. Este método se mostra suficientemente poderoso para atender às mais diversas finalidades que se queira dar à um módulo auxiliar.

5.4.2 Particularidades dos módulos auxiliares

Os módulos auxiliares possuem uma particularidade em relação ao seu tratamento pelo *Helper*. Este tipo de módulo é criado no início da simulação, na fase de configuração, e é destruído apenas quando a execução do simulador é encerrada. A quantidade de módulos é indefinida, pode-se carregar tantos quantos se queira (desde que exista memória suficiente) mas apenas um objeto de cada módulo é criado.

Além disso, cada módulo é único, isto é, existe um identificador único para cada módulo. Isto permite que exista apenas uma interface de chamada dos módulos auxiliares no *Helper*. E, com isso, qualquer requisição é feita pela mesma interface, independente do módulo a ser consultado.

[3] Como é chamada a especificação de qualquer método ou função em *C/C++*.

5.4.3 Módulos disponíveis com a plataforma

São fornecidos juntamente com a plataforma alguns módulos auxiliares já prontos para o uso. Estes módulos, ou utilitários, foram considerados essenciais a qualquer simulação com alguma complexidade além da trivial. Foram, portanto, implementados dois utilitários bastante distintos na sua funcionalidade.

O primeiro é um módulo estatístico. Este módulo disponibiliza uma relação de consultas estatísticas, relativas sempre a uma faixa (ou objeto do tipo *Lane*), enumeradas a seguir:

- Velocidade média da faixa;
- Desvio padrão, das velocidades dos veículos encontrados;
- Quantidade de células livres;
- Densidade de ocupação;
- Lista de veículos com velocidade igual a zero (indicativo de congestionamento);
- Faixa de variação das velocidades, velocidade máxima e mínima observadas nos veículos presentes.

Claro que este conjunto de consultas pode ser estendido, mas para a maioria das necessidades apresentadas foi suficiente. Como já foi dito, pode-se estender estas funcionalidades a outros módulos que implementem outras consultas estatísticas.

Pode-se estender estas mesmas funcionalidades para um conjunto de faixas (objeto do tipo *LaneSet*). Para isso basta alterar a semântica do parâmetro extra para ser interpretado como uma identificação de *LaneSet*. Utilizando as respostas do módulo estatístico se faz necessário adequar os resultados das faixas, *Lanes*, para expressarem o resultado da *LaneSet* que se deseja. Então o trabalho seria obter a lista de faixas que compõe a *LaneSet* e efetuar a consulta individualmente (no módulo já existente), elaborando o resultado final.

Outro módulo disponível é o módulo de comunicação por caixa postal. O método de comunicação implementado é o de caixa postal, sem a necessidade de "contratação" do serviço. Com isso qualquer módulo, tanto auxiliar quanto motorista, pode utilizar o serviço e se comunicar com os outros entes presentes.

O módulo disponibiliza as seguintes funcionalidades:

- Ler a primeira mensagem da fila;
- Ler a primeira mensagem particular (*unicast*);
- Ler a primeira mensagem particular de um remetente em específico;
- Ler a primeira mensagem distribuída por difusão (*broadcast*);
- Enviar uma mensagem para um determinado elemento (objeto de algum módulo);
- Enviar uma mensagem a todos os usuários (*broadcast*);
- Verificar a quantidade de mensagens a serem lidas;

- Verificar a quantidade de mensagens particulares a serem lidas;
- Verificar a quantidade de mensagens não lidas enviadas por um determinado remetente;
- Verificar a quantidade de mensagens, distribuídas por difusão, não lidas.

E cada mensagem é constituída dos seguintes campos:

- Remetente;
- Destinatário;
- Expiração;
- Tamanho da mensagem em caracteres;
- Corpo da mensagem

Como pode ser observado existe um campo que especifica a expiração de uma mensagem. Este campo é necessário para que mensagens não lidas não se acumulem na caixa postal de elementos que não lêem ou não querem ler determinadas mensagens. Por este motivo, a entrega das mensagens não é garantida, isto é, mensagens podem ser perdidas.

Além disso, a classificação das mensagens é por ordem de expiração. Isto é, as primeiras a expirarem devem ser as primeiras a serem lidas. Não é possível especificar expiração indeterminada, as mensagens serão expiradas em algum momento, caso não sejam lidas.

5.5 Módulos de motoristas

Diferentemente dos módulos auxiliares, os motoristas, que também são módulos, são divididos em duas categorias: ruído e objeto de estudo. Os motoristas ruído são os motoristas utilizados pelos objetos do tipo *Source* para povoar a simulação. Já os motoristas objeto de estudo são os motoristas que se quer estudar com maior cuidado e por isso recebem um tratamento diferenciado. Outra diferença entre os módulos motoristas e os módulos auxiliares reside no fato de os módulos motoristas poderem dispor de várias instâncias do mesmo tipo.

Um tipo de motorista pode guiar um ou vários veículos. Existe a possibilidade de se especificar a "cardinalidade" dos veículos dos motoristas. Isto é interessante para permitir uma maior economia de recursos computacionais, caso isso seja possível. Este é o caso dos motoristas que implementam o modelo Nagel–Schreckenberg, no qual um mesmo motorista pode comandar vários veículos. Esta propriedade, poder guiar vários veículos, somente se aplica a motoristas do tipo ruído.

As diferenças entre o ruído e o objeto de estudo é mais extensa. Além da "cardinalidade" dos veículos, existe a propriedade de poder ser eliminado ou não, que somente se aplica a motoristas objeto de estudo. Dependendo do tipo de simulação é necessário manter a instância do motorista "viva" para efetuar alguma consulta ou contabilidade. Isto pode ocorrer mesmo que este não mais participe da simulação. Isto é útil para sinalizar que um motorista não deve ser removido quando seus veículos já percorreram o trajeto planejado.

Como já foi discutido, os módulos motoristas comandam os veículos de forma indireta, através de algum objeto do tipo *DriverInterface* (que implementa a classe abstrata *Driver* do simulador). Então o controle da quantidade de veículos comandados se dá através da quantidade e elementos do tipo *DriverInterface*. Quando uma instância de motoristas fica sem qualquer objeto do tipo *DriverInterface* ela torna-se apta a ser removida.

5.5.1 Preservação de motoristas objeto de estudo

Foi mencionado que motoristas objeto de estudo recebem tratamento diferenciado. Uma destas distinções está no tratamento de exclusão de instâncias. Sempre que um motorista objeto de estudo está apto a ser removido o *Helper* comunica à instância este evento.

Se algum motorista tem a intenção de ser reinserido após deixar a simulação, ele pode fazê-lo. Para isso, o *Helper* questiona a instância do módulo sobre o passo em que deseja ser novamente inserido na simulação. Respondido de forma adequada, o *Helper* coloca a instância numa fila de inclusão, ordenada pelo momento se inserção, que será consumida a medida em que os passos avançam e quando o momento de inserção é atingido, o *Helper* questiona o local, *LaneSet*, que a instância deseja se posicionar e lá o insere.

Outra forma de preservação, sem recolocação, é sinalizá-lo como não removível. Isto faz com que a instância fique ativa e receba atenção do *Helper* informando em que passo se encontra a simulação, única forma de interação efetuada. A instância continua apta a comunicar-se com os módulos auxiliares.

5.5.2 Decisões do motorista

As decisões possíveis aos motoristas são movimentação e planejamento. A movimentação, comentada na seção 5.6, é responsável pela interação de baixo nível. Isto é, decidir a cada passo de simulação que atitude tomar e como reagir aos veículos vizinhos. O planejamento é um algoritmo mais refinado. Nele ocorre a escolha de rota, isto é, que caminho percorrer para ir de um ponto à outro dentro da topologia em que está inserido.

A movimentação é ativada a cada passo de simulação de forma concorrente. Não existe qualquer ordem pré-estabelecida sobre qual motorista toma sua decisão antes de qual motorista, tampouco existe qualquer ordem na aplicação das decisões. Isto deve ser levado em conta no momento de implementar um algoritmo que coordene a movimentação.

No planejamento está a parte mais sofisticada do modelo. Escolher uma rota é responsabilidade do planejamento e as mais diversas heurísticas podem ser aplicadas nesta etapa. Esta etapa será invocada a cada cruzamento, para saber qual direção o veículo deve tomar.

5.6 Metodologia de modelagem de motoristas

Existe praticamente uma ortogonalidade entre as categorias em que se dividiu a tomada de decisão dos motoristas. Isto significa que as ações tomadas em uma decisão têm pouco efeito direto nas ações de outra. Apesar de artificial, esta diferença é necessária pois ajuda a estruturar o modelo de motorista. Notou-se que normalmente a decisão de planejamento somente é necessária em determinados eventos como a aproximação de um cruzamento, na grande maioria dos cenários possíveis. Outra observação importante é a pouca interferência que o planejamento tem sobre a movimentação dos veículos, pois ela limita-se a restringir a movimentação na aproximação do cruzamento. A partir destas constatações a divisão apresentada é quase natural.

A movimentação possui a propriedade de não poder ser planejada com muita antecipação. Isto se deve ao dinamismo do ambiente de trânsito, onde tentativas de antecipação do deslocamento dos vizinhos alguns passos adiante é bastante complicada, se não for inviável. A validade de qualquer decisão de movimentação, dependendo do cenário, acaba no passo seguinte de simulação. Otimizar as decisões de movimentação se restringe, de maneira generalizada, a observar os vizinhos e tentar antecipar suas decisões no passo corrente de simulação.

Esta característica é uma desvantagem e vantagem ao mesmo tempo. Qualquer algoritmo mais elaborado tem pouco ou nenhum efeito sobre a qualidade da decisão final de movimentação. Isto significa que se pode esperar pouca otimização na movimentação, isto é, o ganho fica restrito a algumas poucas células ou passos, dependendo de qual o parâmetro de otimização. Entretanto, o algoritmo tende a ser simples e computacionalmente eficiente.

Por isso a qualidade do modelo está, quase exclusivamente, sob a responsabilidade do planejamento. Normalmente, no planejamento e escolha de rota se dá a validação do modelo de motorista. É nesta etapa do algoritmo que o pesquisador deve colocar seus esforços, uma vez que alguns modelos de movimentação estão disponíveis para uso (através da plataforma *DRIVER-DFW*).

Além da estruturação do modelo do motorista deve-se buscar não sobrecarregar o algoritmo final com funcionalidades desnecessárias. Sempre que alguma funcionalidade se aplica a todas as instâncias de um determinado modelo, e não depende de particularidades do estado interno do motorista, deve estar fora do motorista. Um exemplo bastante simples é supor que o modelo de motorista toma a sua decisão com base na sua velocidade média, experimentada nas rotas escolhidas anteriormente. Neste caso a decisão de qual rota tomar pode ser feita utilizando as funcionalidades já disponíveis na plataforma *DRIVER-DFW*. O motorista apenas armazena as suas velocidades médias, em seus estados internos, e pondera cada etapa do caminho. Feito isto basta solicitar que a plataforma calcule o menor caminho, com base neste pesos.

É preciso, entretanto, bom senso para se aplicar a metodologia corretamente. Apesar de impor algumas restrições ela é bastante flexível, o que pode implicar em uma implementação correta mas ineficiente. A metodologia juntamente com a ferramenta é bastante poderosa e útil quando usada adequadamente.

Sintetizando a metodologia proposta tem-se os seguintes eventos a serem modelados:

Movimentação Responsável pela movimentação regular do veículo e também pelas otimizações locais;

Planejamento Ativado na aproximação de cruzamentos e responsável por estabelecer a rota a ser seguida.

Esta simples proposta serve de base para se organizar a modelagem de um motorista. Baseada nestas premissas de responsabilidades a plataforma *DRIVER-DFW* oferece uma boa ferramenta de modelagem de motoristas.

5.7 Motoristas disponíveis

Dois tipos de motoristas foram implementados utilizando a plataforma para implementação de motoristas *DRIVER-DFW*. Os motoristas implementados se baseiam no

modelo de movimentação de Nagel–Schreckenberg. O primeiro, e mais simples, implementa o modelo apresentado na seção 2.2 e usa as probabilidades apresentadas no capítulo 3 para escolher a sua rota. O mesmo ocorre no segundo motorista, um pouco mais elaborado, implementado com base no modelo apresentado na seção 2.4.

Estes dois motoristas foram implementados de forma a fornecer um modelo de motoristas básico nos quais outros mais elaborados possam se basear. Isto é, criar motoristas mais sofisticados baseados no modelo de movimentação do modelo Nagel–Schreckenberg mas com um algoritmo de planejamento mais elaborado. Nos motoristas implementados a etapa de planejamento apenas faz uma escolha ponderada pelos pesos de cada direção a cada cruzamento, como explicado na seção 3.4.

Com base nestes motoristas simples foram implementados os motoristas apresentados no capítulo 6 (na verdade baseado no modelo de movimentação apresentado na seção 2.2). Isto mostra que é possível estender um modelo já existente a fim de agregar um novo comportamento ou modificar o já existente. Além disso, são dadas as diretrizes para implementação de motoristas baseados na lógica *BDI* na seção 6.6.

5.8 Contribuição a simulações de tráfego urbano

A contribuição deste trabalho à pesquisa de tráfego é o conjunto formado pela metodologia e pela plataforma de implementação. Estes dois componentes são igualmente importantes, pois a existência destes separadamente é pouco significativa. Somente a metodologia teria uma amplitude bastante restrita, pois deixa a cargo do pesquisador criar uma infra-estrutura onde seria possível aplicá-la.

A plataforma vem validar a eficiência e qualidade da metodologia. Com ela é possível demonstrar a aplicabilidade do método e ainda permitir que outros pesquisadores se utilizem dele através da ferramenta.

No próximo capítulo serão apresentados o uso do método aliado à plataforma *DRIVER-DFW*. Também serão apresentados alguns resultados que visam mostrar as vantagens do trabalho apresentado e como ele contribui para a comunidade científica interessada em estudar o tráfego veicular.

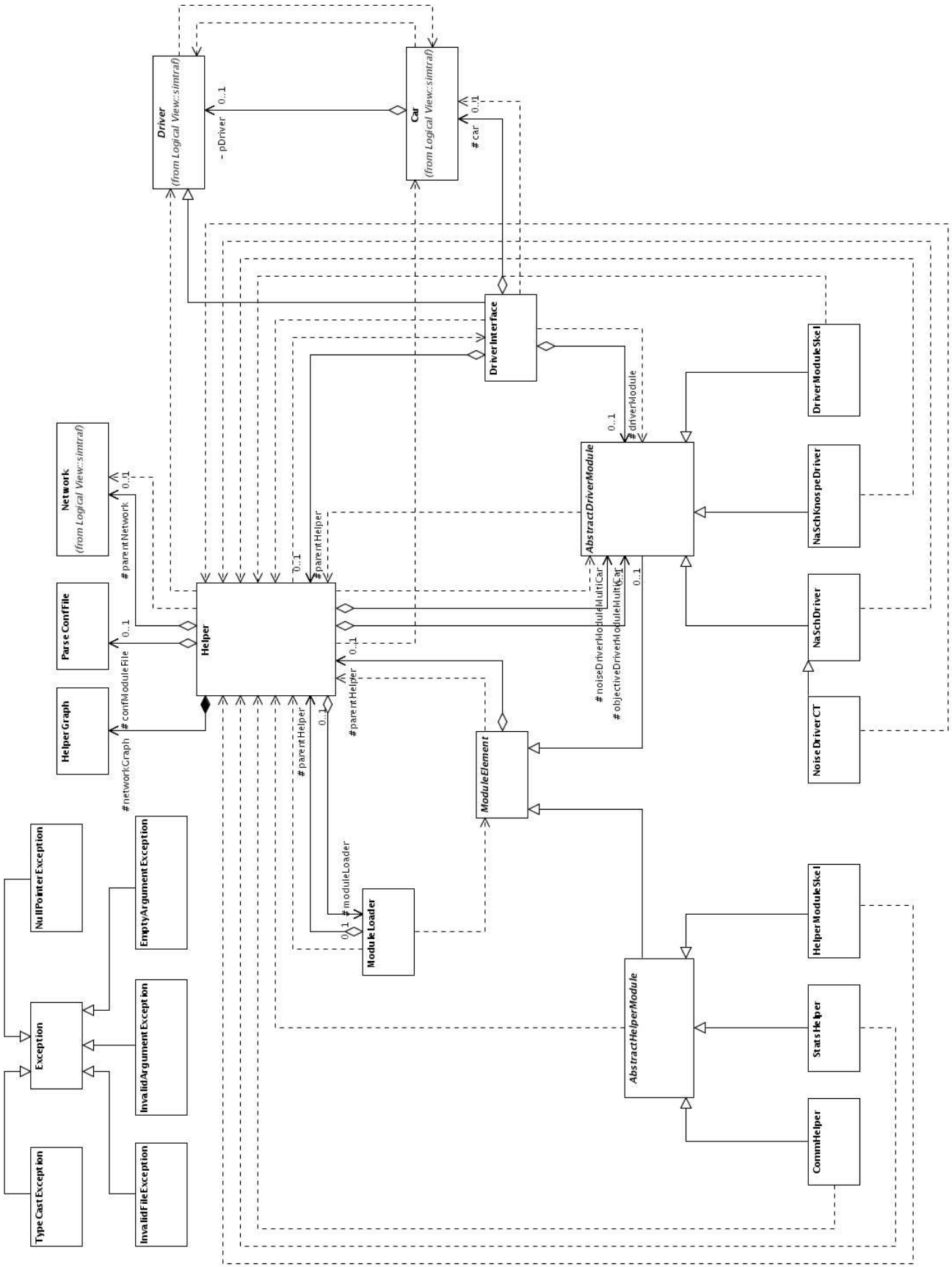

Figura 5.1: Diagrama UML das classes da plataforma para implementação de motoristas *DRIVER-DFW*.

6 DESCRIÇÃO DO USO DA PLATAFORMA E RESULTADOS OBTIDOS

Para validação da metodologia e da plataforma de implementação *DRIVER-DFW* foi escolhido um experimento para simulação que corresponde ao apresentado na seção 4.1.3 e em (KLÜGL; BAZZAN; WAHLE, 2003). Uma modificação neste cenário será apresentada na seção 6.4.

A topologia utilizada está representada na figura 6.1. Nela existem duas rotas, denominadas A e B (identificadas pelas letras correspondentes), que possuem apenas uma faixa (*Lane*), como definido na seção 4.1.3, e com 200 *células* de extensão. A velocidade máxima permitida é de 5 *células/passo*. No nodo mais a esquerda encontra-se um injetor de veículos, ou *Source*, e no mais a direita um sorvedouro de veículos, ou *Sink*, identificados por IN e OUT, respectivamente.

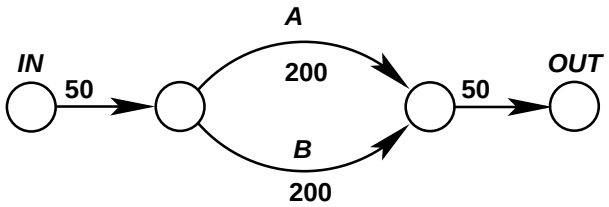

Figura 6.1: Cenário de testes, representação das *LaneSets* e seus respectivos tamanhos, em células.

Todos os motoristas baseiam sua movimentação na extensão para duas faixas do modelo Nagel–Schreckenberg apresentado em (RICKERT et al., 1996) e discutidos na seção 2.2. Foi ajustada uma taxa de desaceleração de 0.25 (probabilidade de 25% para veículo desacelerar). A probabilidade de efetuar uma troca de faixa dentro de uma rua (*LaneSet*) é a máxima, isto é, sempre que possível e vantajoso a troca de faixa deverá ser efetuada.

Todas as simulações foram efetuadas com 50000 passos de simulação (que representam 1 segundo cada) que resultam em um volume de decisões (entre rota A e rota B) que varia entre 850 e 950 para cada agente, dependendo da simulação. Estão presentes 40 motoristas objeto de estudos que percorrem o circuito e são reinseridos. Eles são inseridos apenas no passo 750, o que evita poluir sua coleta de dados com um circuito não populado. Por fim, o tamanho da célula utilizado foi de $5m$.

6.1 Parâmetros de simulação

Como deseja-se validar a plataforma para implementação de motoristas *DRIVER-DFW* buscou-se o cenário mais próximo ao apresentado na seção 4.1.3. Este cenário foi esco-

Tabela 6.1: Tabela de tipos de veículos e suas velocidades máximas permitidas.

Tipo de veículo	velocidade máxima (*células*$/passo$)
corrida	$6 = 108km/h$
esportivo	$5 = 90km/h$
comum	$4 = 72km/h$
transporte	$3 = 54km/h$
carga	$2 = 36km/h$

lhido para reproduzir os resultados apresentados (KLÜGL; BAZZAN; WAHLE, 2003). Algumas modificações, entretanto, foram feitas no cenário original. O número de veículos presentes no cenário é de aproximadamente 120 (foi admitido uma flutuação na quantidade de veículos). A quantidade de motoristas informados, ou motoristas objeto de estudos, foi mantida em 40.

A probabilidade de aprendizagem foi fixada em 0.20. E em outro experimento adicionou-se heterogeneidade de veículos, isto é, os veículos passam a possuir um tipo, que limita a sua velocidade máxima. Os tipos de veículos estão expressos na tabela 6.1.

Para a população de motoristas inteligentes, ou motoristas objeto de estudos, foi assinalado o tipo corrida em todos os experimentos. Foram testadas duas distribuições distintas para a população de motoristas ignorantes, ou ruído:

I Apenas veículos de corrida com 0.50 de probabilidade de escolher a rota A. Este é o cenário original que se encontra descrito na seção 4.1.3.

II Veículos de corrida com 0.20 de probabilidade de escolher a rota A. Veículos de carga com 0.80 de probabilidade de escolher a rota A. Demais veículos com 0.50 de probabilidade de escolher a rota A. Este é um cenário novo em que não há referências para comparações.

Além disso, foram concebidos outros dois tipos de motoristas, chamados experiente e social, que serão explicados na seção 6.4. Desta forma, foram utilizados 3 tipos de motoristas neste mesmo cenário:

Experiente Motorista cuja a decisão se baseia na sua própria experiência, como especificado na seção 6.4.

Social Motorista cuja a decisão se baseia na sua crença a respeito dos outros motoristas, como especificado na seção 6.4.

KBW Motorista cuja a decisão se baseia nas informações adquiridas do centro de informações, como especificado na seção 4.1.3.

6.2 Criando os motoristas

Para criar os motoristas preferiu-se "dividir" o problema em partes. No diagrama de classes da figura 6.2 pode se ver a parte relevante do diagrama UML. Neste diagrama está presente a classe *NaSchDriver* que é a base das demais.

Na classe *NaSchDriver* está o modelo Nagel–Schreckenberg básico explicado na seção 5.7. Nos motoristas representados pela classe *NoiseDriverCT* foi colocado a informação de tipos, isto é, modificou-se o modelo presente em *NaSchDriver* para atribuir um "rótulo"

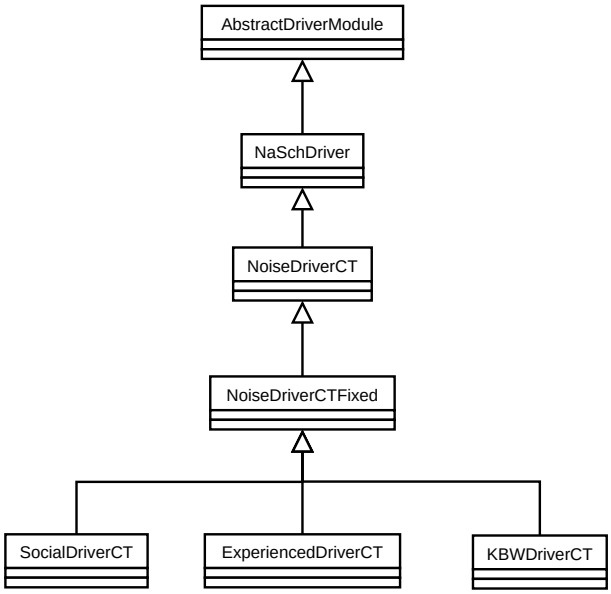

Figura 6.2: Parte do diagrama UML das classes dos Motoristas.

que identifique o tipo a ele associado. Foi feita uma modificação no algoritmo de cálculo da velocidade, herdado de *NaSchDriver*, para consultar o "rótulo" de tipo do veículo e aplicar as restrições de limites de velocidades.

Feito isto foi necessário especializar os motoristas no cenário de teste, figura 6.1. Foi então criado o modelo *NoiseDriverCTFixed* que é uma extensão do modelo presente em *NoiseDriverCT*. Neste modelo foi introduzido o algoritmo de escolha de rota, entre A e B, baseado em um parâmetro probabilístico, que foi chamado de heurística, que representa a probabilidade do veículo escolher a rota A. Em *NoiseDriverCTFixed* este algoritmo retorna sempre 0.5.

Com o motorista base foram feitas as especializações conforme o motorista que se deseja. Em *ExperiencedDriverCT* foi colocado o modelo experiente, seção 6.4. Já em *SocialDriverCT* está o modelo de motorista chamado social, seção 6.4. Por último em *KBWDriverCT* está o modelo especificado na seção 4.1.3.

Note que todos derivam de uma mesma raíz, *NoiseDriverCTFixed*. Em todos os "filhos" de *NoiseDriverCTFixed* o algoritmo de escolha de rota foi implementado de forma a atuar sobre o parâmetro chamado heurística. O detalhamento desta implementação pode ser visto no anexo A.

6.3 Escolha de rota baseada na aquisição de informações

Para implementar este ambiente partiu-se do motorista básico, definido na seção 6.2. Foi alterado o algoritmo de cálculo da probabilidade (de escolha de rota) para comunicar-se com o centro de controle e efetuar a compra de informações e computo dos ganhos (detalhes na seção 4.1.3).

Os custos das informações estão expressos na tabela 6.2.

6.3.1 Resultados

Para produzir estes e os demais resultados foi adotado o seguinte método: caso uma decisão tenha sido tomada no passo 15 e outra no passo 76 o valor do ganho, equação 4.4,

Tabela 6.2: Tabela do custos para aquisição de informações.

Tipo de informação	custo
congestionamento	1
velocidade média	2
tempo da última viagem	2
densidade	3

dos passos 16 à 76 são o mesmo do passo 76 e assim por diante. Com isso, computou-se a média dos ganhos obtidos para traçar o gráfico de ganho por passos.

As figuras 6.3, 6.4, 6.5 e 6.6, referentes ao experimento I, procuram mostrar o ganho médio obtido pelos agentes que adquiriram um determinado tipo de informação ao longo do tempo.

Pode-se perceber que a compra da informação de velocidade média, figura 6.4, constitui uma desvantagem, pois apresenta o pior desempenho (como mencionado em (KLÜGL; BAZZAN; WAHLE, 2003)). Este comportamento foi o mesmo observado em 4.1.3.

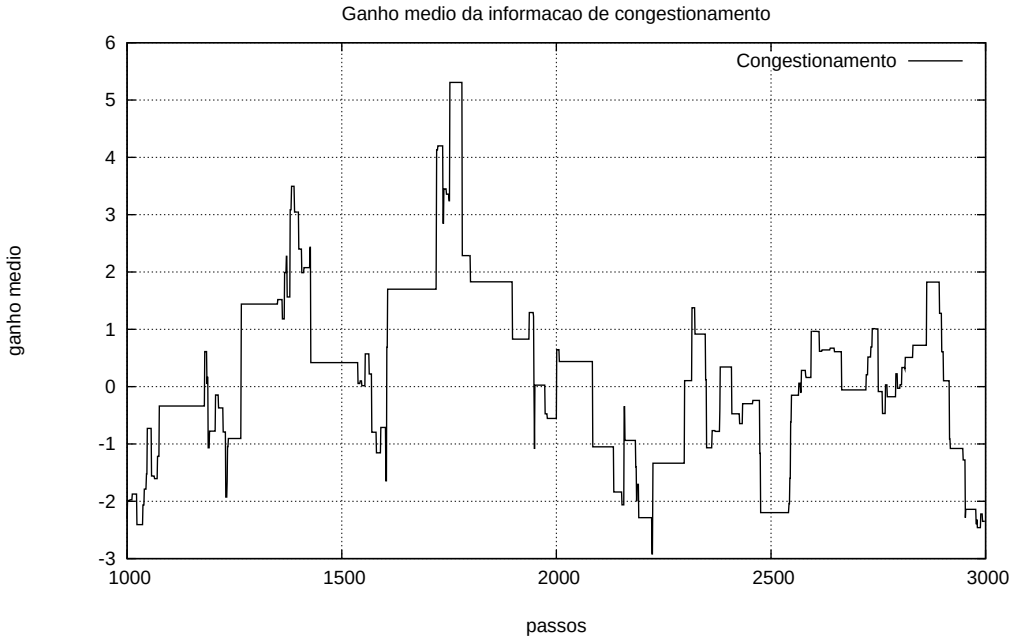

Figura 6.3: Ganho médio obtido pelos agentes que adquiriram a informação de congestionamento, no experimento I.

6.4 Decisão baseada na experiência ou no ambiente

Neste segundo experimento buscou-se verificar o que é mais relevante: experiência ou como se comportam os outros motoristas, considerando seu tipo. Para isso, foi necessário definir o que é a experiência e como ela afeta a tomada de decisão, o mesmo ocorrendo com a avaliação dos outros motoristas.

A experiência foi definida como:

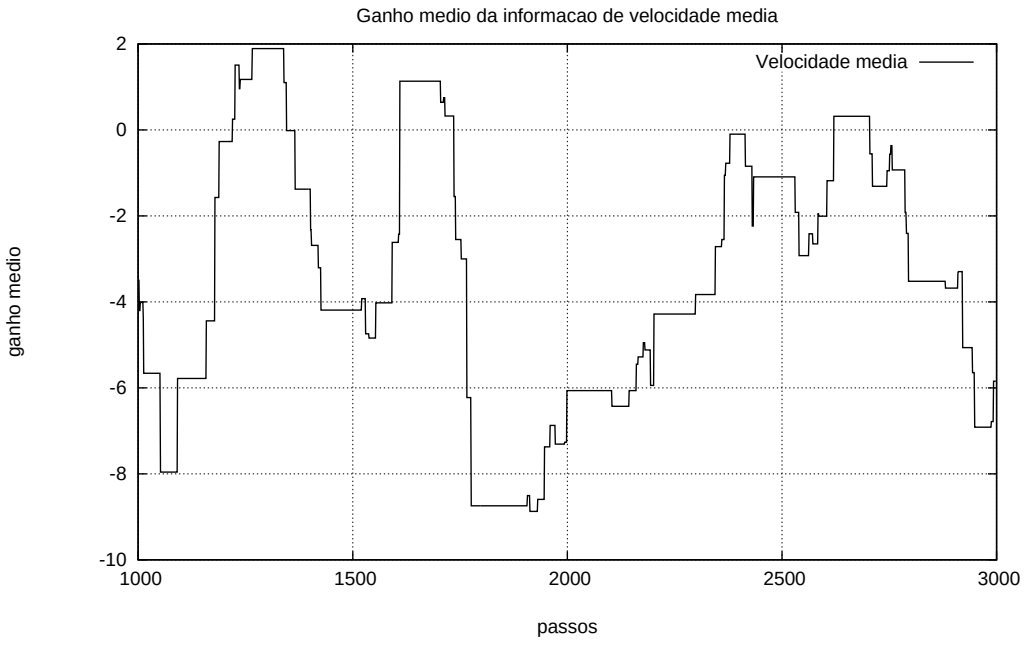

Figura 6.4: Ganho médio obtido pelos agentes que adquiriram a informação de velocidade média, no experimento I.

$$prob_A(Ag_i) = \frac{\overline{t_{R_B}}}{\overline{t_{R_A}} + \overline{t_{R_B}}} \tag{6.1}$$

Onde $prob_A(Ag_i)$ é a probabilidade do agente i escolher a rota A. Já $\overline{t_{R_A}}$ e $\overline{t_{R_B}}$ representam a média dos tempos experimentados nas rotas A e B, respectivamente. Esta média é calculada com base nas últimas n (tomado como 10) vezes em que o motorista percorreu o circuito. Caso alguma das rotas não tenha sido visitada nas últimas n ocorrências o último tempo experimentado será utilizado. Este tempo é sempre guardado para permitir o cálculo da probabilidade.

Já o comportamento do motorista chamado social é baseado na sua crença a respeito do comportamento dos tipos presentes na simulação e suas preferências a respeito deles. A crença a respeito do comportamento dos demais motoristas é sintetizada na equação 6.2, onde as probabilidades $prob_A(tipo)$ são inicializadas com 0.5. Para calcular a sua heurística, baseado nas suas crenças, foi utilizada a equação 6.3.

$$prob_A(tipo) = \begin{cases} prob_A(tipo) - prob_A(tipo)/encontros_A(tipo) & \text{se } tipo \text{ em } B \\ prob_A(tipo) + (1 - prob_A(tipo)/encontros_A(tipo) & \text{se } tipo \text{ em } A \end{cases} \tag{6.2}$$

A equação 6.2 expressa o que o agente acredita ser a probabilidade dos veículos do tipo $tipo$ escolher a rota A. Para efetuar tal cálculo o agente percebe os outros veículos à sua volta e armazena nas suas ocorrências durante o percurso em uma das rotas, atualizando sua crença.

Além destas informações o agente precisa ter alguma crença a respeito do que ele considera atraente ou não. Isto é, quais os tipos de veículos precisam ser "imitados" e quais precisam ser "evitados".

Então a probabilidade do agente escolher a rota A é dada pela equação 6.3.

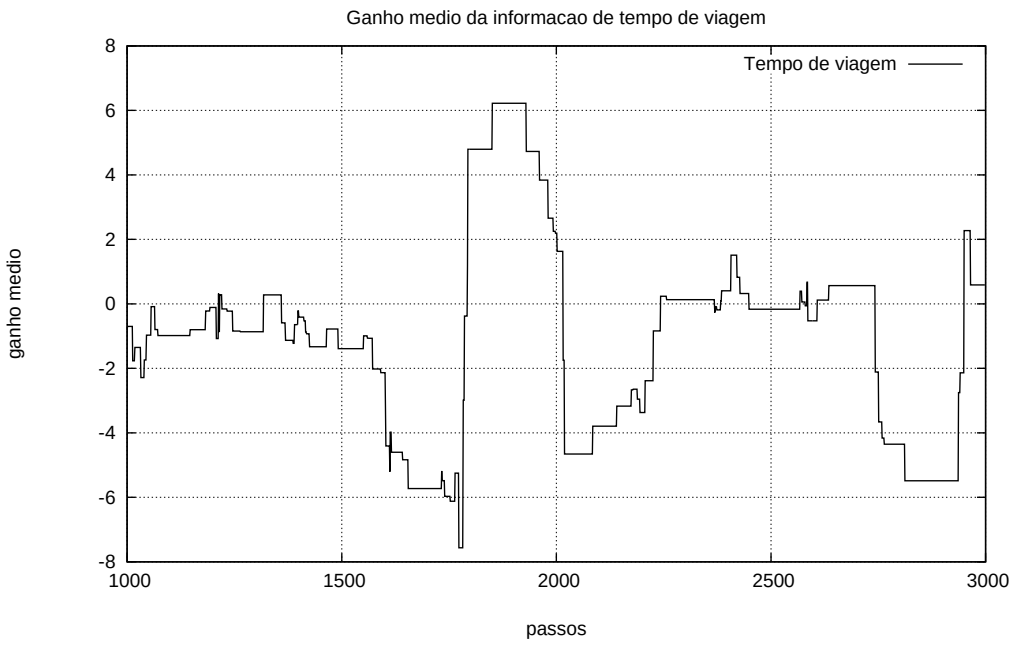

Figura 6.5: Ganho médio obtido pelos agentes que adquiriram a informação de tempo de percurso, no experimento I.

$$prob_A(Ag_i) = \sum_{tipo=carga}^{corrida} \begin{cases} prob_A(tipo) & \text{se } tipo \text{ é atraente} \\ 1 - prob_A(tipo) & \text{se } tipo \text{ não for atraente} \end{cases} \tag{6.3}$$

Caso o agente não tenha uma "opinião" a respeito de algum tipo este não será considerado.

Nos experimentos realizados o agente, chamado social, considera atraentes os tipos corrida e esportivo e não atraentes os tipos carga e transporte.

6.4.1 Resultados

Para comparar os dois motoristas, experiente e social, são apresentados dois tipos de gráficos. Um deles é o de velocidade média dos agentes, isto é, é mostrada a variação da velocidade média ao longo do tempo para os dois tipos de agentes comparados. Já o segundo tipo de gráfico apresenta a proporção entre quantidade de agentes que adotou a rota A em função do total de agentes. Isto, de certa forma, exprime a média do parâmetro chamado de heurística ao longo do tempo.

A proporção entre as quantidades de escolhas pela rota A em função do total não apresenta maiores diferenças (figura 6.7), mas é possível observar que a velocidade média do agente social (figura 6.8) é levemente superior à do agente experiente. Com isso pode-se observar uma vantagem do algoritmo de escolha de rota do agente social em relação ao experiente.

6.5 Comparando os vários experimentos

Mesclando os ambientes pode-se questionar o quão significativas são as informações disponibilizadas no primeiro experimento. Para verificar isso comparou-se o desempenho

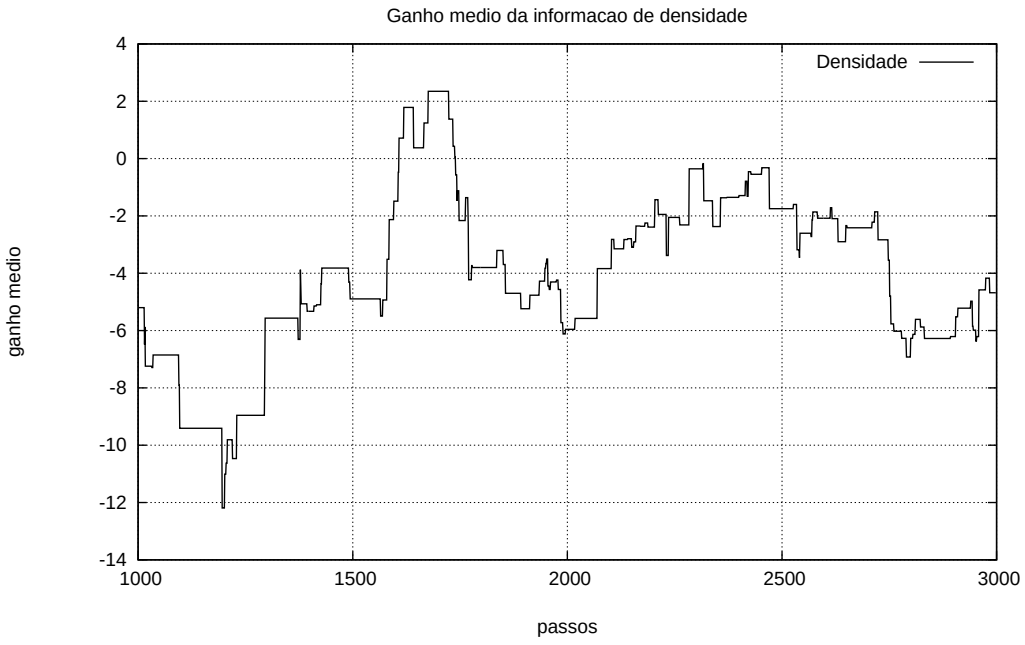

Figura 6.6: Ganho médio obtido pelos agentes que adquiriram a informação de densidade, no experimento I.

dos três tipos de motoristas com relação ao ganho. Foi suposto que os motoristas experiente e social não adquiriram qualquer informação e isto implica que de seu ganho não foi subtraído qualquer custo.

6.5.1 Resultados comparativos

Não é possível inferir qual algoritmo para escolha de rota é o melhor apenas observado a velocidade média dos agentes (figura 6.9), mas observa-se que o agente chamado experiente tem um desempenho inferior. Por outro lado se for observado o comportamento dos agentes no ítem ganho (figura 6.10), supondo que dos agentes social e experiente não é subtraído qualquer custo de seus ganhos, o agente KBW tem um desempenho inferior aos demais, pois precisa adquirir informações e pagar por elas, conforme a tabela de custos 6.2.

6.6 Diretrizes para a modelagem de motoristas *BDI*

Para modelar um motorista que se utiliza da lógica *BDI* a linguagem de descrição *AgentSpeak(L)* (RAO, 1996) é bastante atraente. A linguagem de descrição *AgentSpeak(L)* estabelece que um agente *BDI* é composto de crenças e planos. Já as intenções são geradas automaticamente através de eventos de disparo (*trigger event*), isto é, o interpretador *AgentSpeak(L)* se encarrega de gerar as intenções conforme os estímulos que o agente recebe do ambiente.

Para implementar a proposta (ROSSETTI et al., 2002) (seção 4.2.2) seriam necessários alguns esforços para disponibilizar o ferramental mínimo para a sua implementação. Inicialmente seria necessário criar o interpretador *AgentSpeak(L)*, ou fazer a integração com o trabalho realizado em (MACHADO; BORDINI, 2001). Este interpretador precisa de uma interface com a plataforma para implementação de motoristas *DRIVER-DFW* para

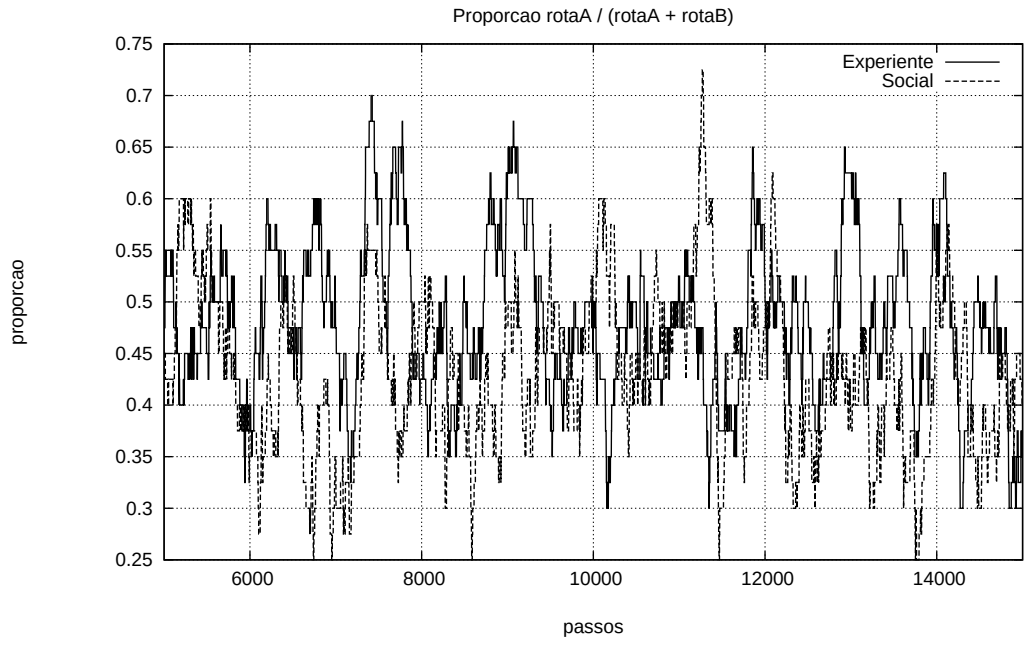

Figura 6.7: Proporção de escolha da rota A referente aos motoristas Experiente e Social no experimento II.

receber os estímulos da simulação e enviar as decisões, através das intenções geradas pelo interpretador *AgentSpeak(L)*.

Supondo que o interpretador pode transmitir aos agentes *BDI* (descritos na forma de *AgentSpeak(L)*) os estímulos da simulação, isto é, o estado das faixas por onde ele está trafegando. Além disso, quais os possíveis cruzamentos (pode-se utilizar o grafo descrito na seção 5.1.3) e momento para a tomada de decisão (solicitação de próxima direção por parte do veículo, descrito nas seções 5.5 e 5.6). As intenções de horário de partida e local de início do trajeto já estão previstas no arcabouço de motoristas disponível na plataforma para implementação de motoristas *DRIVER-DFW*. Já o sistema *ATIS*, previsto no trabalho (ROSSETTI et al., 2002), precisa ser implementado para fornecer as sugestões de rotas e atribuir as origens e destinos aos agentes.

Para a escolha de rota, que é feita antes de iniciar o percurso, é preciso um tratamento diferenciado, ou seja, informar a sua decisão ao simulador a medida que o veículo necessitar.

No momento em que o veículo começa a percorrer a rota escolhida é necessário que a interface do motorista com o agente *BDI* o estimule com os dados necessários neste caso com as condições observadas nas várias etapas da rota (informando, conforme estipulado em (ROSSETTI et al., 2002), se o trecho estava livre, normal ou congestionado). Estas informações deverão ser usadas pelo interpretador *AgentSpeak(L)* para ajustar as crenças do agente.

As diretrizes definidas acima são bastante precisas mas apresentam um alto grau de abstração. Existe uma série de barreiras a serem transpostas para concretizar este projeto e os mais contundentes são o interpretador *AgentSpeak(L)* e a integração através de um módulo motoristas, isto é, fornecer um módulo motorista que interaja com o interpretador *AgentSpeak(L)* e posso transmitir as ações do agentes ao simulador. Além disso, o interpretador precisará ser especializado para modelagem de motoristas de forma que as

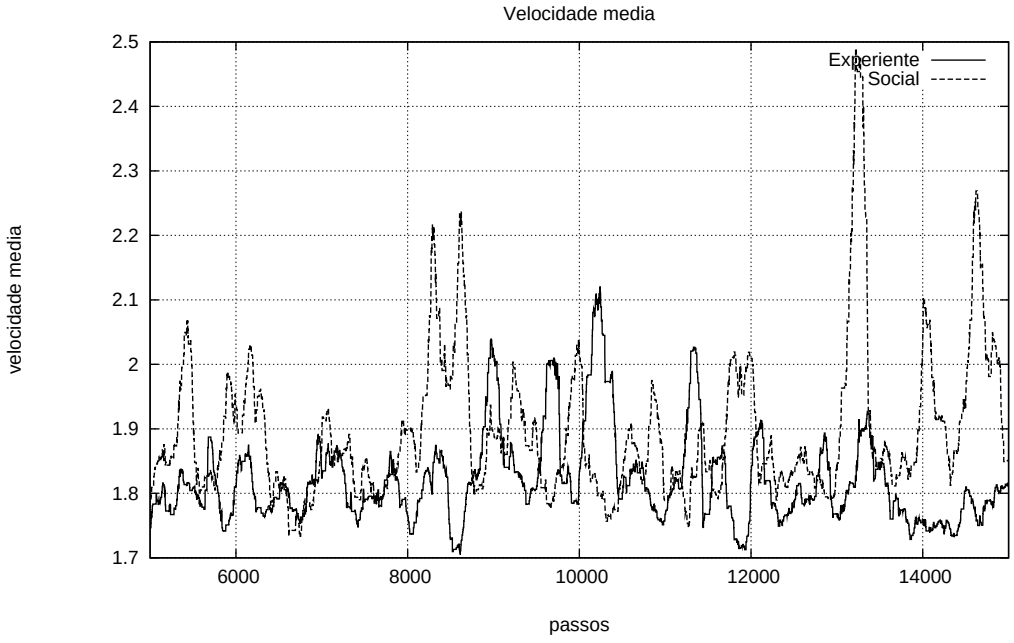

Figura 6.8: Velocidade média obtida pelos motoristas Experiente e Social no experimento II.

intenções geradas se traduzam em interações válidas com a plataforma para implementação de motoristas *DRIVER-DFW*.

De certa forma, o agente *BDI* será restrito pois deverá ter algum plano relacionado à escolha de rota, obedecendo um formato bem definido, a fim de gerar uma intenção que será traduzida em decisão de rota. Outra restrição é utilizar um conjunto de crenças pré-definidas (neste caso as possibilidades de rotas) de forma que a plataforma *DRIVER-DFW* possa atuar diretamente sobre ela.

Apesar dos desafios envolvendo a criação das estruturas mínimas para possibilitar o uso da lógica *BDI* para definição de motoristas inteligentes e impor restrições ao modelo *BDI* a ser utilizado, é possível utilizar motoristas baseados na lógica *BDI*.

6.7 Conclusão

Foi apresentado um cenário de teste assim como sua concepção que tenta reproduzir o cenário apresentado na seção 4.1.3. Assim como os considerações apresentadas na seção 4.1.3 aqui também se observou o mesmo fenômeno (a informação de velocidade média apresenta o pior desempenho, em termos de ganhos, em relação aos demais).

Fez-se ainda uma extensão do cenário original acrescentando dois novos tipos de motoristas e uma distribuição de motoristas ignorantes diferente do cenário original. Neste novo experimento foram feitas as comparações de eficiência e desempenho dos motoristas, tanto em termos de velocidade média quanto de ganho, para se verificar qual dos modelos de motoristas apresenta o melhor desempenho.

Além destes testes foram apresentadas as diretrizes para se permitir o uso da lógica *BDI* na modelagem de motoristas. Com isso buscou-se mostrar a versatilidade da metodologia e plataforma de implementação de motoristas *DRIVER-DFW*.

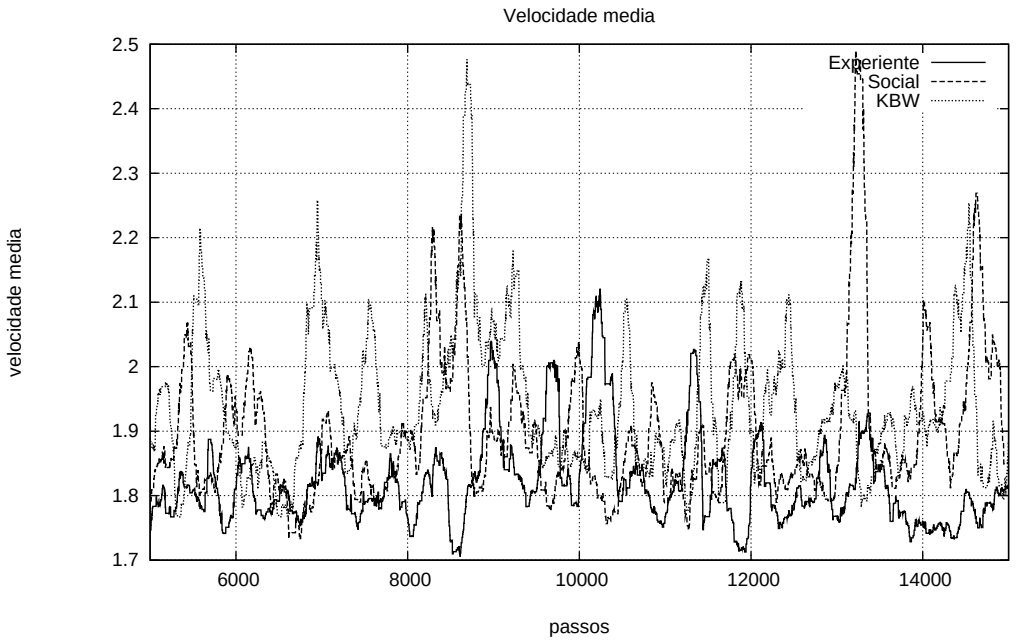

Figura 6.9: Velocidade média referente aos diferentes motoristas no experimento II.

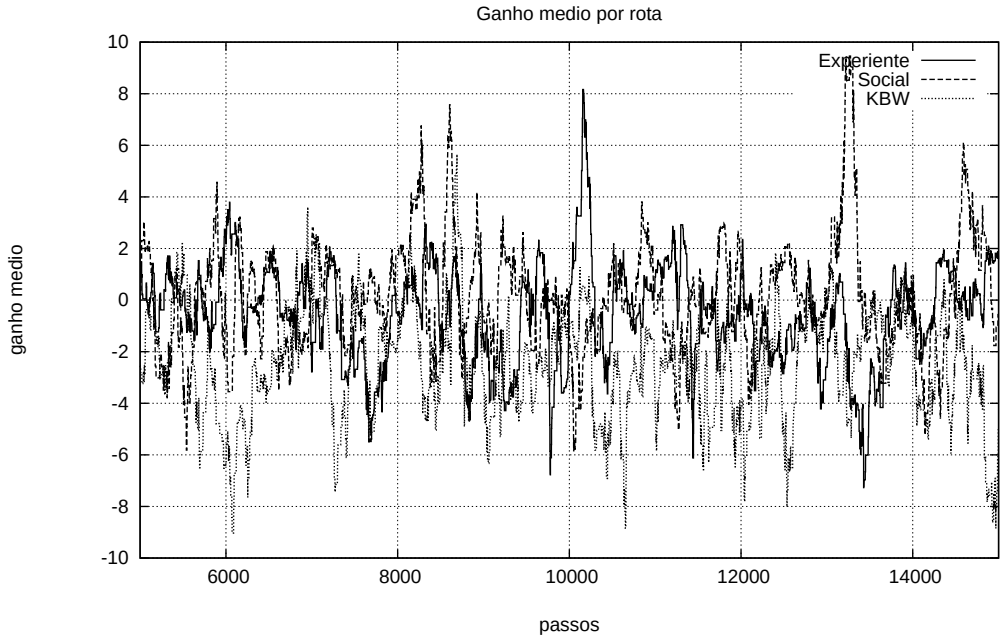

Figura 6.10: Ganho médio referente aos diferentes motoristas no experimento II.

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foram estudados o modelo de autômato celular Nagel–Schreckenberg bem com algumas de seus aprimoramentos. Este modelo de autômato celular é a base do simulador do projeto *ITSUMO*, que foi apresentado no capítulo 3. Em seguida foram apresentados algumas heurísticas para a escolha de rota e delas uma foi escolhida o caso de testes.

Após a apresentação do sistema base para este trabalho, foi apresentada uma metodologia para modelagem de motoristas bem como uma plataforma para implementação de motoristas, baseado na metodologia apresentada. Além disso, foram realizados alguns experimentos utilizando o novo ferramental criado.

Estes experimentos foram apresentados como ilustração para mostrar as funcionalidades da plataforma *DRIVER-DFW* e como a metodologia pode ser utilizada para facilitar a concepção de modelos de motoristas. Além disso, estes casos de testes foram utilizados para validar o trabalho realizado, onde se obteve resultados semelhantes aos obtidos através de outra ferramenta.

Diversos pontos podem ser explorados no futuro. Entre eles está a integração com sistemas do tipo SIG (Sistemas de Informações Geográficas) que permitiriam utilizar qualquer malha viária armazenada desta maneira. Este tipo de sistema de representação está em fase de padronização e conta um banco de dados em constante crescimento[1].

Além disso, uma série de melhorias na ferramenta precisam ser efetuadas, uma vez que foi manifestado interesse de terceiros em utilizá-la. Dentre estas melhorias está aprimorar sua eficiência computacional e melhor documentar o código fonte.

É preciso também disponibilizar um conjunto maior de ferramentas de modelagem. Entre elas estaria um interpretador de *AgentSpeak(L)*, para possibilitar o uso facilitado da lógica *BDI* na modelagem de motoristas. Outros paradigmas também necessitam ser facilitados, como o uso de redes neurais para a escolha de rotas.

Uma das barreiras mais difíceis a ser vencida é desvincular a modelagem da programação em baixo nível. É importante disponibilizar uma linguagem de mais alto nível para modelagem dos motoristas. Com isso seria possível aumentar significativamente o número de pessoas aptas a usufruir da metodologia aliada à ferramenta *DRIVER-DFW*.

[1]Para maiores informações consulte: *http://www.opengis.org/*.

REFERÊNCIAS

ANDRIOTTI, G. K. **Implementação de veículo tracejador para o simulador do projeto SISCOT**. 2001. 41p. Projeto de Diplomação (Bacharelado em Ciência de Computação) — Instituto de Informática — Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, Brasil.

ANDRIOTTI, G. K. **Revisão bibliográfica do modelo Nagel–Schreckenberg e modelos de motoristas**. 2003. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática — Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, Brasil.

ANDRIOTTI, G. K. ITSUMODFW: a framework to implement drivers for discrete traffic simulations. In: OPTIMISATION FOR URBAN TRANSPORT SYSTEMS, 2004, Mexico. **Proceedings...** [S.l.: s.n.], 2004.

ANDRIOTTI, G. K.; BAZZAN, A. L. C. An Object-Oriented Microscopic Traffic Simulator. In: CONFERÊCIA LATINOAMERICANA DE INFORMÁTICA, CLEI, 28., 2002, Montevideo. **[Articulos]**. Montevideo: Universidad de la Republica, 2002. p.306–310.

BARLOVIC, R.; SANTEN, L.; SCHADSCHNEIDER, A.; SCHRECKENBERG, M. Metastable States in Cellular Automata for Traffic Flow. **Euro. Phys. Journal B**, [S.l.], v.5, p.793, 1998.

BAZZAN, A. L. C.; KLÜGL, F. Route Decision Behaviour in a Commuting Scenario: simple heuristics adaptation and effect of traffic forecast. In: EUROWORKSHOP ON BEHAVIOURAL RESPONSES TO ITS, 2003, Eindhoven. **Proceedings...** [S.l.: s.n.], 2003.

BENJAMIN, S. C.; JOHNSON, N. F.; HUI, P. M. Cellular Automaton Models of Traffic Flow Along a Highway Containing a Junction. **Journal of Phys. A**, [S.l.], n.29, p.3119, 1996.

DIJKSTRA, E. W. A Note on Two Problems in Connexion with Graphs. In: NUMERISCHE MATHEMATIK, 1955. **Proceedings...** [S.l.: s.n.], 1955. v.I, p.269–271.

KLÜGL, F.; BAZZAN, A. L. C. Simulation of Adaptive Agents: learning heuristics for route choice in a commuter scenario. In: AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, AAMAS, 2002, Bologna, Italy. **Proceedings...** New York: ACM Press, 2002. v.1, p.217–218. Extended Abstract.

KLÜGL, F.; BAZZAN, A. L. C.; WAHLE, J. Selection of Information Types Based on Personal Utility - a Testbed for Traffic Information Markets. In: AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, AAMAS, 2003, Melbourne, Australia. **Proceedings...** New York: ACM Press, 2003.

KLÜGL, F.; PUPPE, F. The Multi-Agent Simulation Environment SeSAm. In: WORKSHOPS "SIMULATION IN KNOWLEDGE-BASED SYSTEMS", 1998. **Proceedings...** [S.l.]: Reihe Informatik, Universität Paderborn, 1998. p.194.

KNOSPE, W.; SANTEN, L.; SCHADSCHNEIDER, A.; SCHRECKENBERG, M. A realistic two-lane traffic model for highway traffic. **J. Phys. A**, [S.l.], n.35, p.3369–3388, 2003.

MACHADO, R.; BORDINI, R. H. Running AgentSpeak(L) agents on SIM_AGENT. In: UKMAS, 2001. **The Fourth UK Workshop on Multi-Agent Systems: working notes...** [S.l.: s.n.], 2001.

NAGEL, K.; SCHRECKENBERG, M. A cellular automaton model for freeway traffic. **J. Phys. I France**, [S.l.], v.2, p.2221, 1992.

RAO, A. S. AgentSpeak(L): bdi agents speak out in a logical computable language. In: WORKSHOP ON MODELLING AUTONOMOUS AGENTS IN A MULTI-AGENT WORLD, 7., 1996, Eindhoven, The Netherlands. **Proceedings...** [S.l.: s.n.], 1996.

RICKERT, M.; NAGEL, K.; SCHRECKENBERG, M.; LATOUR, A. Two Lane Traffic Simulation on Cellular Automata. **Physica A**, [S.l.], v.231, p.534, 1996.

ROSSETTI, R. J. F.; BORDINI, R. H.; BAZZAN, A. L.; BAMPI, S.; LIU, R.; VAN VLIET, D. Using BDI Agents to Improve Driver Modelling in a Commuter Scenario. **Transportation Research Part C: Emerging Technologies**, [S.l.], v.10, n.5–6, p.47–72, 2002.

ROSSETTI, R.; LIU, R.; VAN VLIET, D.; BAMPI, S.; CYBIS, H. Applying an agent–based approach to traffic modeling. In: WORKSHOP ON AGENTS IN TRAFFIC AND TRANSPORTATION, 2000, Barcelona. **Proceedings...** [S.l.: s.n.], 2000. p.45–59.

ROSSETTI, R.; LIU, R.; VAN VLIET, D.; CYBIS, H. An agent–based framework for the assessment of drivers' decision–making. In: INTELLIGENT TRANSPORTATION SYSTEMS, 2000, Dearborn, MI, USA. **Proceedings...** Piscat away: IEEE, 2000. p.387–392.

SILVA, B. C.; ANDRIOTTI, G. K.; BAZZAN, A. L. C.; FERREIRA, P. R.; LOPES, F.; OLIVEIRA, D.; PAPAGEORGIOU, T. M. ITSUMO: an intelligent transportation system for urban mobility. In: OPTIMISATION FOR URBAN TRANSPORT SYSTEMS, 2004. **Proceedings...** [S.l.: s.n.], 2004.

WAGNER, P.; NAGEL, K.; WOLF, D. E. Realistic multi-lane traffic rules for cellular automata. **J. Phys. A**, [S.l.], v.234, p.687–698, 1997.

WAHLE, J.; BAZZAN, A. L. C.; KLÜGL, F.; SCHRECKENBERG, M. Decision Dynamics in a Traffic Scenario. **J. Phys. A**, [S.l.], v.287, p.669–681, 2000.

WARDROP, J. G. Some theoretical aspects of road traffic research. In: INSTITUTION OF CIVIL ENGINEERS, PART II, 1952, London. **Proceedings...** [S.l.: s.n.], 1952. v.1, p.325–378.

ANEXO A DETALHAMENTO DA IMPLEMENTAÇÃO DE UM MOTORISTAS

Para implementar um motoristas deve-se entender como o simulador procede com a requisição de uma decisão junto aos motoristas. Os passos para a requisição de uma decisão de movimentação segue abaixo (a requisição de planejamento é análoga).

- O objeto da classe *Car* solicita ao seu objeto da classe *Driver* (que é a classe abstrata de *DriverInterface*) uma decisão (diagrama UML na figura 3.1), através do método `takeDecision()`;
- O objeto da classe *DriverInterface* trata esta chamada e a transforma em uma chamada de `takeLocalDecision()` do seu objeto da classe *AbstractDriverModule*.

Os eventos acima ocorrem dentro da plataforma de implementação. A responsabilidade do usuário é criar uma classe concreta baseada na classe *AbstractDriverModule* que use o método `takeLocalDecision()` para informar a decisão de movimentação do seu motoristas.

Para implementar o motorista experiente da seção 6.2 é necessário dividir a tarefa em etapas.

A.1 Motorista baseado no autômato celular Nagel–Schreckenberg

A tomada de decisão de movimentação do modelo Nagel–Schreckenberg foi dividida em duas partes: mudança de faixa e avanço de células. Primeiro é feita a verificação para a troca de faixa, método `isLaneChangeSecure()`, que informa ao motoristas se a troca de faixa é segura, isto é, não haverá colisão ao efetuar a manobra.

Após isto é verificado se existe vantagem em efetuar a troca de faixas, algoritmo apresentado na seção 2.2. Avaliando-se a troca de faixa segue-se com o cálculo da próxima velocidade, pelo método `naschCalculateNewSpeed()` que é apresentado pelas equações 2.1 e 2.2. Com isso, obtém-se a tomada de decisão de movimentação baseado no autômato celular Nagel–Schreckenberg.

Para a tomada de decisão de planejamento, método `chooseNextDestination()`, apenas é feita uma escolha aleatória entre os possíveis destinos, conforme a ponderação apresentada na seção 3.4. A obtenção de tais dados é feita à partir da consulta das probabilidades dos caminhos da *LaneSet* em que o motoristas se encontra. Isto é feito através do *Helper* que é consultado, pelo método `queryLaneSetInfo()`, a respeito das propriedades da *LaneSet*. Dentro destas propriedades encontram-se estas probabilidades.

Este é o motoristas implementado na classe *NaSchDriver*.

A.2 Motorista com restrição de velocidade baseada em tipos

Para incluir uma restrição de velocidade baseada em tipos criou-se a classe *NoiseDriverCT*. Nela estão as definições dos tipos, conforme a tabela 6.1 e para aplicar estas etiquetas de tipo foi necessário utilizar o método `initializeCarExtraData()` que atribui um dado genérico a ser colocado nos veículos. Este método é chamado sempre que um novo veículo é "atrelado" ao motorista (a plataforma se encarrega de colocar o valor retornado dentro do veículo).

Para implementar a restrição de tipo o motoristas *NoiseDriverCT* sobrescreve o método `naschCalculateNewSpeed()` da classe *NaSchDriver*. Neste método é feita a seguinte modificação: limita-se a velocidade máxima àquela permitida conforme o tipo do veículo em questão. Para obter o tipo do veículo basta consultar o argumento `carCurrentStateRef` passado na chamada do método `takeLocalDecision()`. Nenhum outro algoritmo precisa ser alterado.

A.3 Motorista especializado no cenário apresentado no capítulo 6

Como foi feito um planejamento para utilizar esta classe, chamada *NoiseDriverCTFixed*, como base para as demais, a implementação foi dirigida para este fim. Sabia-se, de antemão, quais os cenários a serem simulados então especializou-se o motoristas da classe *NoiseDriverCTFixed* para tratar somente estes cenários.

O algoritmo de movimentação não foi modificado, apenas o de planejamento. Para permitir um melhor aproveitamento sobrescreveu-se o método `chooseNextDestination()` para utilizar uma probabilidade, fornecida pelo método `calculateHeuristic()`, para decidir quais das rotas escolher. Neste motorista básico o método sempre retorna 0.50 ou, caso especificado, um valor específico para o tipo de veículo que solicita a decisão. Este valor específico, se existir, fica armazenado em um arquivo de configuração usado na inicialização das simulações.

A.4 Motoristas experiente

O motoristas experiente, chamado *ExperiencedDriverCT*, decide qual rota escolher através do algoritmo apresentado na seção 6.4. Este algoritmo foi implementado sobrescrevendo o método `calculateHeuristic()` da classe *NoiseDriverCTFixed*.

Foi necessário também modificar o algoritmo `chooseNextDestination()` para contabilizar, ao final do percurso, o tempo consumido na rota adotada.

Aqui não foram expostos todos os detalhes da implementação, como geração dos resultados em arquivos ou mesmo a geração de registros para depuração, pois não são relevantes ao funcionamento do modelo de motorista.