UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Felipe Augusto de Souza

UMA PLATAFORMA COMPUTACIONAL PARA IMPLEMENTAÇÃO DE CONTROLE PREDITIVO DISTRIBUÍDO COM APLICAÇÕES AO CONTROLE DE TRÁFEGO VEICULAR URBANO EM MACAÉ

Florianópolis (SC)

2012

Felipe Augusto de Souza

UMA PLATAFORMA COMPUTACIONAL PARA IMPLEMENTAÇÃO DE CONTROLE PREDITIVO DISTRIBUÍDO COM APLICAÇÕES AO CONTROLE DE TRÁFEGO VEICULAR URBANO EM MACAÉ

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas para a obtenção do Grau de Mestre.
Orientador: Prof. Eduardo Camponogara, Ph.D.
Coorientador: Prof. Werner Kraus Jr, Ph.D.

Florianópolis (SC)

2012

Catalogação na fonte elaborada pela Biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

Felipe Augusto de Souza

UMA PLATAFORMA COMPUTACIONAL PARA IMPLEMENTAÇÃO DE CONTROLE PREDITIVO DISTRIBUÍDO COM APLICAÇÕES AO CONTROLE DE TRÁFEGO VEICULAR URBANO EM MACAÉ

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre", e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis (SC), 21 de junho 2012.

Prof. Jomi Fred Hübner, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Antonio Augusto Rodrigues Coelho, Dr.
Presidente

Prof. Eduardo Camponogara, Ph.D.
Orientador

Prof. Werner Kraus Jr, Ph.D.
Coorientador

Rodrigo Castelan Carlson, Dr.

Prof. Rômulo Silva de Oliveira, Dr.

Dedico à Vó Maria e ao meu primo Rafael que continuarão sempre na minha memória.

AGRADECIMENTOS

Há muitas pessoas que merecem agradecimentos, como em todos os trabalhos, no entanto houveram fatos no último ano que tornaram muito maior a minha vontade de agradecer pessoas que muito me ajudaram.

Primeiramente gostaria de agradecer amigos que mesmo não participando diretamente do trabalho sempre me deram força para continuar. Nesse grupo eu coloco o pessoal da turma 0.41, a galera do Campeche e aos amigos próximos de Balneário Camboriú. É sempre difícil colocar nomes porque sempre alguns podem ficar injustamente de fora, mas posso citar o Shrek, Lucas, Diego, BB, Crici, Gabriel Paim (Demon), Jonatas, André, Klauss e Thiago.

O fato de eu ter me matriculado no mestrado em setembro de 2008 e estar defendendo em junho de 2012 implica que meu orientador e coorientador tiveram que me aguentar por quase quatro anos e nesses período foram sempre muito receptivos. O Werner, coorientador, acredito que fizemos por uns seis meses uma boa "dupla de dois" e quando o Vinícius entrou no projeto um bom "trio de três" em Macaé-RJ e não é que a Central entrou em Tempo Real ? O Camponogara além de ter que me aguentar duas vezes na mesma disciplina sempre tentou me ajudar de todas as formas. Sempre serei muito grato aos dois.

O projeto em Macaé-RJ foi um grande aprendizado e muito gratificante. Sempre imagino eu daqui a trinta anos lembrando com carinho deste projeto. Das pessoas que trabalharam diretamente comigo existem duas pessoas que trabalharam lá que merecem ser citadas: o Fabiano e o Pedro.

A ATTA já tem 4 anos e esse pessoal sempre me ajudou **muito**. Todos que estão ou já estiveram na ATTA merecem ser citados: Marana, Vini, Rodrigo, Luciano, Daniel, Elisângela, Musa, Marco, Rauh e Ricardo. Entretanto nunca vou me esquecer da atitude dos que estavam perto quando tive um problema de saúde. O Marana, Vini e o Daniel foram simplesmente sensacionais. Gostaria de dar especial agradecimento ao Marana, não foram poucas às vezes em que me ajudou e continua sempre sendo muito prestativo.

Se citei o apoio do pessoal da ATTA, imensurável também é o apoio da família. A minha mãe sempre louca (fiquem tranquilos, não é pejorativo ela sabe que ela é), mas preocupada comigo mais que eu mesmo. Meu pai mais tranquilo, mas não que ajude ou apoie menos, são apenas características

diferentes. A Mariana sempre é a mais chata. Como diz o pai fala abobrinha e come "pôlo" (bolo), mas se eu chamo-a de chata é óbvio que na verdade adoro ela. O Arthur sempre foi e continua sendo meu melhor amigo. O início do mestrado bate quase com a mesma data do maior presente que ele e a Jéssica nos deram que é a menininha Ingrid. A cada encontro é sempre uma grande alegria, tomara que continue sempre assim.

Um agradecimento especial também à minha vó que continua cuidando do meu vô e que nos recebe com muito carinho sempre. Há também outros parentes que gostaria de citar, mas em especial o Daniel.

Como é tradicional nos meus relatórios, sempre tem um agradecimento musical. Desta vez vai para a MPB, Música Pesada Brasileira. No momento mais crítico de Macaé quando batia um pequeno desânimo sempre dava para escutar "*Carry on*" (e muitas outras) do Angra. No final, escrevendo a dissertação, teve bastante Shadowside para me manter ligado. E viva a MPB!

Se conhecimento pode criar problemas, não é através da ignorância que podemos solucioná-los.

Isaac Asimov

RESUMO

Técnicas de Controle Preditivo tem sido cada vez mais utilizadas para o controle de sistemas dinâmicos pelo seu bom desempenho em aplicações. Entretanto, em sistemas geograficamente distribuídos, como redes de tráfego urbano e a rede elétrica, a centralização das informações para computação da ação de controle pode ser um obstáculo em função da necessidade de confiabilidade da comunicação e do elemento central ser um ponto único de falha. Controle Preditivo Distribuído baseado em Modelo (DMPC) distribui o cálculo da ação de controle sem a necessidade de centralização de informações, apenas compartilhando-as com seus vizinhos enquanto preserva características do Controle Preditivo baseado em Modelo (MPC), especialmente o tratamento explícito de restrições. Este trabalho apresenta uma plataforma desenvolvida capaz de modelar sistemas dinâmicos distribuídos, configurar e executar algoritmos distribuídos. Experimentos em simulação foram realizados em redes de tráfego urbano e os resultados foram avaliados sobre dois aspectos. Do ponto de vista do desempenho computacional atingiu-se resultados que indicam a aplicabilidade dos algoritmos; do ponto de vista do tráfego mostrou-se perspectivas de ganho de técnicas de controle preditivo (MPC e DMPC) sobre LQR no controle de percentuais de verde da estrat[egia de controle de tráfego urbano TUC.

Palavras-chave: Otimização de Distribuída, Controle Preditivo, Controle de Tráfego Urbano

ABSTRACT

Predictive Control Techniques have been increasingly used to control dynamical systems due to good performance in applications. However, in geographically distributed systems, such as urban traffic networks and the power grid, centralization of information for computing the control action can be an obstacle because of necessity of communication reliability and the central node be a single point of failure. Distributed Model Based Predictive Control (DMPC) distributes the calculation of the control action without the need to centralize information. This is done by sharing them only with neighbors while preserving Model Predictive Control (MPC) characteristics, especially the explicit treatment of constraints. This work has developed a platform capable of modeling distributed dynamical systems, besides allowing the configuration and execution of distributed algorithms. Simulation experiments were conducted in urban traffic networks and the results were evaluated based on two aspects. From the point of view of computational efficiency, results indicate the applicability of the algorithms in real-life problems; from the point of view of traffic performance, there are gain prospects in predictive control techniques (MPC and DMPC) as compared to the LQR control strategy used in the green split calculations of the well-known TUC urban traffic control strategy.

Keywords: Distributed Optimization, Model Predictive Control, Traffic Urban Control

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS E ABREVIATURAS

Símbolos

ε	Precisão de aproximação na centralização do método de barreira
$\mathbf{u}$	Entrada do sistema ou agente m
$\mathbf{y}$	Saída do sistema ou do agente
f	Termo independente para restrições de desigualdade em MPC.
F_u	Termo dependente das entradas de controle para restrições de desigualdade em MPC.
F_x	Termo dependente dos estados para restrições de desigualdade em MPC.
I	Matriz identidade
μ	Taxa de decrescimento do método de barreira
$\pi(m,t)$	Conjunto de subsistemas que afetam o subsistema m em até t períodos de amostragem à frente
$\sigma(k)$	Índice de carregamento da região, por sua vez baseado nos índices de carregamento de uma região.
σ_{ref}	Índice de carregamento de referência do controlador de ciclo do TUC.
$\sigma_i(k)$	Índice de carregamento de uma via. A proporção entre carros na via e o que cabe dentro dela.
τ	Tolerância da solução
A	Define como o estado de um agente influencia o estado $\mathbf{x}$ de outro agente
$A(\omega_t)$	Matriz definindo como o estado de um subsistema no instante k influencia outro subsistema no instante $k+t$
B	Define como a ação de controle influencia o estado $\mathbf{x}$ de um agente
$B(\omega_t)$	Matriz definindo como a entrada de controle de um subsistema no instante k influencia o estado de outro no instante $k+t$

C	Tempo de ciclo
C_o	Índice de carregamento de referência do controlador de ciclo do TUC.
E_j	Conjunto de estágios da interseção j.
e_z	Conjunto de estágios em que o link z tem direito de passagem.
gs_i	Grau de saturação da via i
J	Função Custo
j_i	Interseção ou junção i.
l_z	Comprimento da via z.
$M(m)$	Vizinhança de entrada do subsistema m
n_i	Fluxo atual ou projetado da via i
P_m	Problema de otimização do agente m
Q	Matriz de ponderação de estados nas técnicas LQR e MPC
R	Matriz de ponderação das ações de controle nas técnicas LQR e MPC
r	Ponderação de controle da técnica TUC
s_i	Fluxo de saturação da aproximação i
T	Horizonte de Predição.
T	Horizonte de Predição
T_s	Período de amostragem
$t_{i,j}$	Proporção de carros que saem da pista i e entram na pista j.
v_z	Velocidade de percurso na via z.
W	Define como o estado de um agente influencia a saída y de outro agente
Z	Define como a ação de controle de um agente agente influencia a saída y de outro agente
$N(m)$	Conjunto dos subsistemas que estão acoplados ao subsistema m

$\tilde{N}(m)$ Conjunto dos subsistemas i que não influenciam e não são influenciados pelo subsistema m, mas ambos afetam algum outro subsistema m

$\widehat{N}(m)$ Conjunto dos subsistemas que são afetados pelas decisões do subsistema m

Abreviaturas

AGDD algoritmo de gradiente descendente distribuído

API Interface de Programação de Aplicativos, do inglês *Application Programming Interface* - API

DMPC Controle Preditivo Distribuído baseado em Modelo, do inglês *Distributed Model Predictive Control* - DMPC

LQR Regulador Linear Quadrático, do inglês *Linear Quadratic Regulator* - LQR

MPC Controle Preditivo baseado em Modelo, do inglês *Model Predictive Control*

RDL Redes Dinâmicas Lineares

SFM *Store and Forward Modelling*.

TUC Técnica de controle de tráfego Trafic Urban Control

SUMÁRIO

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

Controle Preditivo baseado em Modelo (*Model Predictive Control* - MPC) tem sido cada vez mais utilizado para o controle de sistemas dinâmicos (CAMACHO; BORDONS, 2004). Suas principais vantagens sobre técnicas de controle clássicas, como o controlador PID, são o controle explícito de restrições e lidar naturalmente com sistemas com atraso de transporte (RICO; CAMACHO, 2006). As primeiras aplicações de MPC foram em plantas de dinâmica lenta, como a indústria química (GARCIA; MORARI, 1989), em função da necessidade de capacidade de processamento. Entretanto com o aumento do *clock* dos processadores, tem-se utilizado MPC até em controle de robôs móveis (GU; HU, 2005).

A técnica consiste em transformar o problema de controle em uma série de problemas de otimização. A cada instante de amostragem as variáveis são medidas e um problema de otimização é instanciado e resolvido sobre um horizonte de tempo finito, obtendo uma sequência de ações de controle ótima para este horizonte. O primeiro elemento da sequência é aplicado no sistema e no período de amostragem seguinte o processo é repetido e assim sucessivamente. A predição sobre um horizonte induz as ações de controle recentes levar em consideração suas consequências em instantes futuros.

Suas principais vantagens são: o tratamento explícito de restrições; bom seguimento de referência; e garantia de estabilidade para sistemas lineares (sob certas condições para sistemas não lineares também). Possui uma inerente desvantagem que é a necessidade de capacidade computacional para a solução do problema de otimização em tempos compatíveis com o período de amostragem do controle.

A grande maioria dos sistemas estão confinados a uma determinada área nos quais estas considerações são válidas. Entretanto existem sistemas a serem controlados que são geograficamente distribuídos. Considere a rede de tráfego urbano da Figura 1. Existem 3 vias[1] que chegam na primeira interseção (J1), L1 abre no primeiro estágio e L2 e L3 abrem no segundo. Na segunda interseção (J2) L4 abre no primeiro estágio e L5 no segundo. O objetivo do controle é definir os tempos de cada estágio a cada ciclo de tráfego (aproximadamente 80 segundos).

Cada interseção tem seu funcionamento totalmente independente das demais: cada interseção possui um equipamento responsável por acionar as

[1] A terminologia relativa à redes de tráfego é apresentada no Apêndice A.

lâmpadas e de ler a quantidade de carros em cada via. Entretanto a dinâmica é dependente das interseções vizinhas e, portanto, as ações de controle de cada uma destas devem ser coordenadas para melhorar o desempenho de uma determinada região.

A solução de controle realimentado mais adotada por sistemas (em geral, não apenas de tráfego) comerciais e experimentais é usar um elemento central que concentra dados de todos os sensores, calcula a ação de controle e envia-as aos atuadores a cada período de amostragem. Dependendo do sistema, o período de amostragem pode ser de poucos segundos. A concentração de dados em um único ponto é um obstáculo por uma série de fatores. Primeiramente o elementro central é um ponto único de falha, caso ele falhe o sistema inteiro falha. Embora existam soluções como redundância, estas podem ser onerosas ou difíceis de serem implementadas. Como todo o cálculo das ações de controle é feito no elemento central, este deve ter enorme capacidade computacional até para redes não muito grandes (CAMPONOGARA; SCHERER, 2011). Por último, a remoção e inclusão de novos componentes no sistema necessita de uma nova configuração gerando custos de manutenção.

Para esse tipo de sistema, técnicas tem sido propostas como o Controle Hierárquico e o Controle Distribuído por serem escaláveis e de mais fácil reconfiguração (SCATTOLINI, 2009). Controle Distribuído trabalha em subsistemas e algoritmos de coordenação que fazem com que estes subsistemas cooperem até alcançar desempenho ótimo para o modelo proposto. É desejável, no entanto, que o sistema de controle preserve características de desempenho do controle preditivo centralizado. Uma das técnicas propostas neste caso é o Controle Preditivo Distribuído baseado em Modelo (*Distributed Model Predictive Control* - DMPC) (CAMPONOGARA et al., 2002)

DMPC é uma técnica que busca preservar as características do MPC, entretanto medição e controle são realizados por agentes distribuídos que têm uma visão local do sistema. Cada agente lida com poucas variáveis e comunicam-as à outros agentes em sua vizinhança. Por exemplo, no caso de redes de tráfego urbano, cada interseção pode ser vista como um agente que decide e mede poucas variáveis (decide tempos de verde e mede filas) comunicando-as à seus vizinhos de forma adequada até alcançar a solução ótima.

Na Figura 1 o agente 1 é o controlador da interseção 1, quem decide quanto tempo cada estágio deve ter. A visão local dele são suas duas variáveis de decisão (tempo de cada estágio) e a medição da fila de cada uma das 3 vias que chegam na interseção. Este agente está acoplado com o agente da interseção 2, isto é, suas decisões interferem diretamente no agente 2, por sua vez o agente 2 pode estar diretamente acoplado com mais de um agente e assim

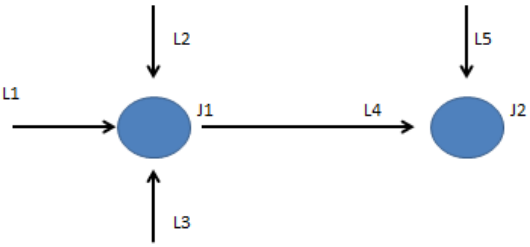

Figura 1: Exemplo de uma malha veicular.

sucessivamente. Tais encadeamentos de acoplamentos entre agentes levam à necessidade de estratégias de coordenação dos agentes para que restrições sejam respeitadas e objetivos globais sejam alcançados.

A estratégia de controle distribuído permite que os agentes tratem apenas modelos e informações locais do sistema podendo atingir a mesma solução que um agente centralizado alcançaria caso tivesse conhecimento e controle global do sistema. Isto demonstra a capacidade de expansão de forma que caso um novo agente (junção) tivesse que ser adicionado, seriam necessários apenas ajustes em seus vizinhos. Em caso de uma grande rede ser dividida em duas subredes por um problema qualquer, os agentes naturalmente iriam operar para alcançar o ótimo nas duas redes, demonstrando grande potencial de reconfigurabilidade.

1.2 OBJETIVOS DO TRABALHO

Os objetivos do trabalho são avaliar a aplicação da técnica MPC em controle de tráfego urbano, aplicando a técnica MPC proposta em (ABOUDOLAS et al., 2007a) sobre uma técnica que originalmente baseava-se na técnica LQR (Regulador Linear Quadrático, do inglês *Linear Quadratic Regulator*) (DIAKAKI et al., 2002), e, ao mesmo tempo, realizar experimentos utilizando a técnica de Controle Preditivo Distribuído.

Este trabalho apresenta uma plataforma computacional que permite modelar sistemas dinâmicos distribuídos, avaliar o desempenho de um algoritmo de DMPC, assim como comparar com o algoritmo centralizado correspondente. Apresenta também experimentos em redes de tráfego urbano (controle de tráfego urbano em tempo real), apresentando uma aplicação prática

para a técnica de DMPC, ainda que em simulação.

1.3 ORGANIZAÇÃO DO DOCUMENTO

O documento está organizado como segue. O Capítulo 2 apresenta as principais técnicas de controle de tráfego urbano. O Capítulo 3 apresenta as técnicas de MPC e DMPC, primeiramente apresentando genericamente a técnica de MPC, e na sequência a modelagem por Redes Dinâmicas Lineares e os algoritmos distribuídos. O Capítulo 4 apresenta a plataforma desenvolvida para modelagem e experimentação de DMPC. Resultados de experimentos para avaliação do algoritmo distribuído implementado e no controle de tráfego urbano são apresentados no Capítulo 5. O Capitulo 6 apresenta as conclusões do trabalho.

2 MODELAGEM E CONTROLE DE TRÁFEGO

Neste capítulo serão apresentadas técnicas de controle de tráfego, principalmente as técnicas cíclicas, com especial enfoque na técnica TUC que serviu de base para a proposta de controle preditivo distribuído.

2.1 FORMAS DE CONTROLE DE TRÁFEGO

Nesta seção são apresentados brevemente os sistemas de controle de tráfego existentes, contendo uma breve descrição de funcionamento, locais propícios a sua utilização, vantagens e desvantagens. Neste texto o termo plano se refere a ciclo, defasagem e tempos de estágio de uma interseção. No contexto de uma rede, plano denota um plano para cada interseção da região.

2.1.1 Controle em Tempo Fixo

Presente na grande maioria das cidades brasileiras, consiste em uma "agenda"(tabela) de planos semafóricos a serem executados, que são aplicados pelo controlador conforme a agenda à medida que o tempo transcorre (GORDON; TIGHE, 2005). Tempo fixo é indicado para interseções isoladas ou em rede (conjunto de cruzamentos próximos) desde que as demandas sejam previsíveis (FHWA, 2008). É a técnica de mais baixo custo, entretanto não responde às variações no tráfego e necessita de uma constante atualização dos planos semafóricos. Isto é feito através de estudos de contagens e recalculado *off-line*.

2.1.2 Controle Atuado

No controle atuado, o controlador semafórico realiza a troca de estágios em função de dados vindos de detectores veiculares colocados nas aproximações da interseção.

Em linhas gerais, um estágio persiste enquanto os carros passam com espaçamento temporal menor que um tempo pré-determinado, denominado brecha máxima, sobre o detector veicular ou até que um tempo máximo seja alcançado (FHWA, 2008).

O controle atuado é indicado para interseções isoladas e lida muito bem com variações na chegada dos veículos. A principal vantagem é a ca-

pacidade de responder ao tráfego no ciclo corrente em contrapartida às técnicas cíclicas que, embora utilizem informações recentes, o plano corrente não se altera após seu início (GORDON; TIGHE, 2005).

2.1.3 Seleção de Planos

A seleção de planos é considerada uma evolução das técnicas de tempo fixo. Detectores veiculares são posicionados na área a ser controlada e os dados são enviados a uma central. Um procedimento é utilizado para escolher o plano mais adequado para cada interseção que toma como base medidas de contagem e ocupação. A troca de planos é realizada em tempos que podem variar de 5 a 30 minutos (GORDON; TIGHE, 2005).

É considerada uma opção melhor em relação a tempo fixo quando as variações no tráfego ocorrem em horários diferentes diariamente. Na variação do padrão do tráfego um plano é escolhido, adequado às condições vigentes.

Uma desvantagem é que frequentemente o sistema encontra-se no limiar entre duas condições diferentes de tráfego (*i.e.*, entre dois planos possíveis de serem aplicados) e poderá ficar durante um tempo considerável com um plano inadequado se a escolha for errônea, degradando o desempenho à patamares inferiores ao tempo fixo.

Além disso, da mesma maneira que em tempo fixo, os planos devem ser calculados *off-line*, portanto necessita também de frequentes atualizações dos planos semafóricos. Apesar da disponibilidade de dados históricos para servir de entrada para softwares de cálculo de planos de tempo fixo, como Transyt (CRABTREE et al., 1996), geralmente não há informações de todas as aproximações (em geral existe um detector por interseção apenas) e a sintonia em campo deve ser realizada ao atualizar a tabela de planos.

2.1.4 Controle em Tempo Real (Realimentado)

Controle em tempo real são sistemas computadorizados, em geral centralizados, que realizam de modo geral as seguintes tarefas (GORDON; TIGHE, 2005):

1. coleta as condições vigentes de tráfego através de detectores veículares;
2. realiza cálculos para determinar planos adequados; e
3. o controlador semafórico executa estes planos que podem alterar a cada ciclo ou a cada estágio.

Diferentemente de sistemas de seleção de planos, o controle em tempo real gera automaticamente os planos de tráfego a serem executados no ciclo de tráfego seguinte. Para isto existem detectores veiculares em praticamente todas as pistas, diferentemente de seleção de planos que emprega em geral um por interseção (GORDON; TIGHE, 2005).

Como vantagem consegue responder a variações inesperadas no tráfego, como por exemplo a um veículo quebrado, situações que não são captadas por sistemas de seleção de planos. Além disso, sistemas de tempo real não sofrem com a obsolescência dos planos com passar do tempo, uma vez que calculam planos para as condições vigentes, diferentemente de seleção de planos e tempo fixo, segundo os quais os planos foram construídos para as condições existentes no momento de implantação do sistema.

Apesar de possível, o controle em tempo real não é utilizado para interseções isoladas, onde as técnicas de controle atuado e outras técnicas de controle local podem ser utilizadas.

2.2 TÉCNICAS DE CONTROLE DE TRÁFEGO EM TEMPO REAL

Nesta seção faz-se uma breve apresentação dos fundamentos do controle de tráfego em tempo real e algumas técnicas em particular, com enfoque às técnicas de controle cíclicas.

O controle de tráfego em tempo real precisa de três blocos básicos:

- **sistema de medição:** para captar as condições vigentes de tráfego;
- **módulo de controle:** responsável por calcular os planos mais adequados às condições vigentes; e
- **controlador semafórico:** responsável por executar os planos calculados pelo módulo de controle.

Estes três blocos funcionais estão representados na Figura 2. Os detectores captam o estado do tráfego. A central, com seu algoritmo de controle, calcula novos planos e os enviam para o controlador semafórico, representados na figura pelos focos do semáforo.

SISTEMAS DE MEDIÇÃO

Existem diversas grandezas de tráfego que são utilizadas como entrada para as técnicas de controle. Contagem e ocupação são medidas diretamente e

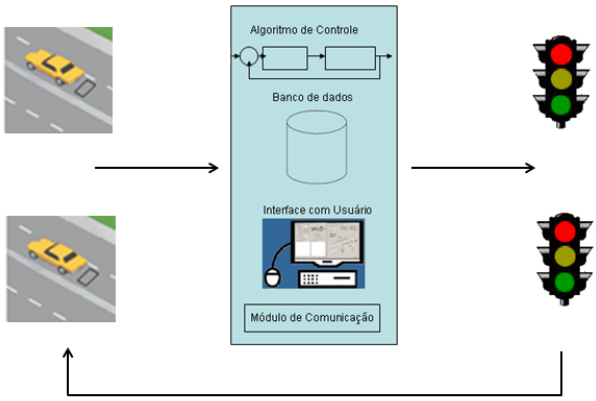

Figura 2: Esquemático de funcionamento em tempo real.

largamente utilizadas. Em termos de controle uma medida indireta importante é o grau de saturação, definido pela seguinte equação (HCM, 2000):

$$gs_i = \frac{n_i C}{s_i G_i} \tag{2.1}$$

onde:

- n_i é o fluxo atual ou projetado para a aproximação i;
- s_i é o fluxo de saturação da aproximação i;
- G_i é o tempo de verde alocado para aproximação i; e
- C é o tempo de ciclo.

Grau de saturação baixo, muito menor que 1, indica que a aproximação está com sobra de capacidade. Por outro lado, um grau de saturação próximo 1 significa que a aproximação está operando perto da saturação, isto é, atende a demanda quase sem folga. Valores maiores que 1 significam que a aproximação está operando acima da capacidade e que as filas tenderão a aumentar.

Durante o projeto de planos de tempo fixo é comum ajustar os tempos de verde e ciclo para que todas as aproximações tenham um grau de saturação em torno de 0,9, tomando como base históricos de fluxo veicular. Na verdade procura-se baixar o ciclo o quanto possível - desta forma diminuindo atrasos, uma vez que os veículos esperam menos tempo para o semáforo abrir - e ainda operando dentro da capacidade com alguma folga. Alguns sistemas de

tempo real utilizam o mesmo princípio, entretanto os fluxos são medidos em tempo real.

Para as medidas citadas laço indutivo é a tecnlogia mais utilizada para medição das grandezas diretas. Outras grandezas como velocidade e tamanho de fila podem ser medidas ou estimadas com a composição adequada de laços indutivos ou outras tecnologias de detecção, como câmeras (KLEIN et al., 2006).

MÓDULO DE CONTROLE

O módulo de controle é responsável por calcular novos planos (ciclos, defasagens e tempos de verde) de tráfego utilizando informações providas pelo sistema de medição (contagens, ocupações, filas, *etc.*), com o objetivo de minimizar ou maximizar critérios de desempenho, como minimização dos tempos de espera e número de paradas, através de uma resposta adequada às flutuações de tráfego.

As técnicas se subdividem em cíclicas e acíclicas As cíclicas são baseadas na geração dos planos do ciclo seguinte (ciclo, defasagem e tempos de estágio) a cada ciclo, utilizando informações do ciclo corrente (CARLSON, 2006). Sistemas acíclicos não possuem o conceito de plano e suas decisões se restringem a manter a indicação semafórica atual ou trocar a qualquer momento. Informações resumidas de técnicas de controle podem ser encontradas em (CARLSON, 2006; GORDON; TIGHE, 2005). Aqui apresenta-se apenas as técnicas cíclicas que são de interesse à pesquisa realizada.

2.2.1 SCOOT – Split, Cycle, Offset Optimization

SCOOT é uma técnica que se baseia no modelo do Transyt (ferramenta de auxílio para geração de planos de tempo fixo) (COHEN, 1982). Esta técnica utiliza histogramas da entrada e saída de cada aproximação, chamados de perfil de entrada e perfil de saída, que são construídos através de dados de detectores localizados em todas as aproximações (ROBERTSON; BRETHERTON, 1991), podendo estar tão distante quanto possível, na linha de retenção ou até mesmo após a saída da pista. SCOOT provavelmente é o sistema de controle de tráfego em tempo real com maior número de instalações ao redor do mundo (GORDON; TIGHE, 2005). Possui três módulos de otimização:

- **ciclo:** busca manter capacidade suficiente para a interseção crítica (grau de saturação menor que 0,9).

- **defasagem (*offset*):** busca gerar coordenação (ondas verdes) entre as fases. Antes de cada alteração de fase decide se antecipa, mantém ou estende em 4 segundos o verde da fase.
- **percentuais de verde (*split*):** define os tempos dos estágios buscando maximizar um critério de desempenho baseado em número de paradas e espera total.

2.2.2 SCATS – Sydney Co-ordinated Traffic Control System

SCATS é uma técnica de controle que ajusta os tempos semafóricos de acordo com a demanda e a capacidade (LOWRIE, 1982), sendo subdividida em duas camadas, uma estratégica e a outra tática.

A camada estratégica é implementada na central de controle sendo responsável por ajustar o ciclo da região, incrementando ou decrementando até 6 segundos a cada ciclo, defasagem e percentuais de verde. Ela tem o objetivo de equilibrar o grau de saturação entre as fases críticas que podem variar em até 4% a cada ciclo.

Trabalhando abaixo do nível estratégico, a camada tática é implementada pelo controlador local que toma algumas ações baseadas em informações e demandas locais, como estender ou retrair tempos de estágios, ou até mesmo omitir estágios, como uma botoeira de pedestre não acionada. Na omissão ou retração de um estágio o controlador decide o que executar, isto é, que fases são habilitadas no intervalo de tempo do estágio omitido.

Os detectores são posicionados na linha de parada de cada aproximação. O desempenho do SCATS se deteriora com tráfego acima da saturação durante períodos longos (ABOUDOLAS et al., 2009). Uma possível explicação é que informações úteis de ocupação não são obtidas uma vez que a detecção de carros é realizada na linha de parada, apenas informações de contagem são obtidas—um detector posicionado próximo à linha de retenção estará praticamente o tempo todo ocupado, bastando existir um único veículo na fila, tornando a estimação de filas e lentidão uma tarefa difícil.

2.3 TUC

TUC (*Traffic-Responsive Urban Control*) (DIAKAKI et al., 2002) é uma técnica desenvolvida para o controle de tráfego veicular em tempo real de redes de grande porte, inclusive em condições saturadas. Inicialmente desenvolvida apenas para o controle de percentuais de verde de estágios, foi estendida possibilitando também o controle de ciclo e defasagem (DIAKAKI

et al., 2003). No que se segue é apresentado brevemente o controle de ciclo, defasagem e percentuais de verde.

2.3.1 Controle de Ciclo

O controle de ciclo está diretamente relacionado à capacidade de escoar veículos das interseções. Ciclos maiores aumentam a capacidade, pois a proporção de tempo perdido (tempo de amarelo, vermelho e de reação) se torna menor, aumentando a proporção de tempo utilizado para o escoamento de veículos (DIAKAKI et al., 2003). Por outro lado ciclos menores são desejáveis na medida do possível. Ciclos menores com interseções operando abaixo de suas capacidades proporcionam atrasos menores, pois veículos que chegam na interseção com sinal vermelho esperam menos tempo para que este seja aberto.

Em tempo fixo, o ciclo de todas as interseções são ajustados para o mesmo valor, denominado ciclo da região, para possibilitar que a diferença de tempo de abertura entre dois semáforos seja constante e, desta forma, pode-se ajustar esta diferença de tempo de abertura, chamado de defasagem, para valores adequados. A estratégia TUC, assim como outras técnicas cíclicas conhecidas, utiliza o mesmo princípio, isto é, a cada instante o ciclo de todas as interseções da região é o mesmo, no entretanto o ciclo da região pode variar com tempo. A figura 3 apresenta a evolução do ciclo em uma interseção de Macaé-RJ durante a realização deste trabalho. Observe o aumento do ciclo nos horários de pico.

No TUC $\mathbf{x}_i(k)$ denota a fila da via i no tempo k, isto é, o número de carros na via no instante k. O número máximo de carros que cabem na via, representado por $\mathbf{X}_i^{\max}$.

O ciclo é calculado de acordo com os seguintes passos:

- O índice de carregamento de cada via i, dado por $\sigma_i(k) = \frac{\mathbf{x}_i(k)}{\mathbf{x_i^{\max}}}$ de cada via é calculado e ordenado;
- O índice de carregamento da região $\sigma(k)$ é feito pela média dos N, sendo este um parâmetro ajustável para este fim, maiores índices de carregamento $\sigma_i(k)$;
- Um controle proporcional é aplicado

$$C(k) = C_o + K(\rho(k) - \sigma_{\text{ref}}) \tag{2.2}$$

onde C_o é o ciclo mínimo, σ_{ref} é o máximo valor de $\sigma(k)$ em que ainda se aplica o ciclo mínimo, K é o ganho do controlador de ciclo (quanto

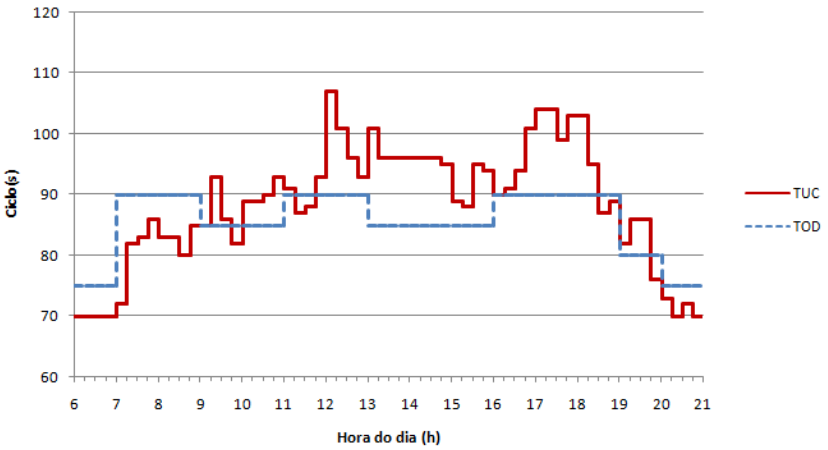

Figura 3: Evolução do ciclo operando em tempo fixo (linha tracejada) e TUC (linha contínua).

maior mais rápida será a ascenção do ciclo no aumento de ρ) e $C(k)$ é o tempo de ciclo gerado pelo controlador.

- O ciclo $C(k)$ é submetido à saturação. Se $C(k)$ é menor que o ciclo mínimo este é aplicado, analogamente quando $C(k)$ for maior que o ciclo máximo, o máximo é aplicado.

- Se o tempo de ciclo for muito alto para algumas interseções de volumes menores, estas podem ter o ciclo dobrado, isto é, executam dois ciclos no período de um ciclo da região.

2.3.2 Controle de Defasagem

O objetivo do controle de defasagem é diminuir os tempos de espera de veículos ajustando o instante de abertura de cada fase, sem que mudanças ocorram no tempo de ciclo e nos percentuais de verde. A situação ideal, impossível de alcançar, seria todos veículos encontrarem os semáforos sempre abertos. Na prática, procura-se um ajuste priorizando os caminhos de maior fluxo, frequentemente vias arteriais, realizando o que se chama ondas verdes. TUC funciona da seguinte forma:

- Configura-se arteriais, que não necessariamente correspondem fisica-

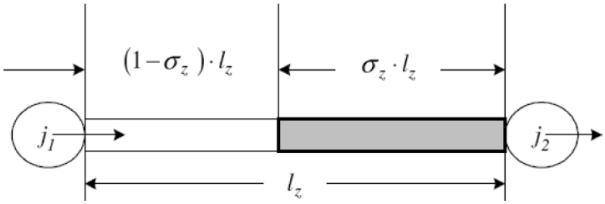

Figura 4: Pista conectando duas junções (retirado de (DIAKAKI et al., 2002)).

mente à arteriais na malha, por uma sequência arbitrária de junções ligadas por uma ou duas (mão dupla) pistas.

- São calculadas defasagens relativas entre cada junção baseado na distância l_z, na velocidade de percurso na via z v_z e na fila $X_z(k)$ de cada pista (Figura 4. Procedimento que será descrito posteriormente.

- A partir da junção de referência (primeira da lista de junções) e as defasagens relativas obtém-se a defasagem de cada junção

Para a defasagem entre duas junções sucessivas, caso não houvesse fila, levaria-se em conta a razão $t = l_z/v_z$ que é o tempo com que o primeiro veículo da fila à montante alcança o cruzamento à jusante, isto é, tempo necessário para percorrer a pista z. Entretanto, na existência de fila é necessário que se tenha tempo suficiente para que a fila à jusante escoe, portanto, no caso geral o objetivo do controle da defasagem é que o primeiro veículo à montante alcance o último veículo da fila à jusante no momento que este entra em movimento

Como mencionado, TUC utiliza a distância da pista l_z, a velocidade v_z e a fila $X_z(k)$ representadas. A Figura 4 representa o link z que sai de j_1 e chega em j_2.

O comprimento linear da fila é estimado como $l_z\sigma_z(k)$. A defasagem ideal entre duas junções é calculada como segue:

$$\frac{[1-\sigma_z(k)]l_z}{v_z} = t_{j1j2} + \frac{\sigma_z(k)l_z}{v_z} \tag{2.3}$$

No caso de vias de mão dupla são obtidos t_{j1j2}, a defasagem ideal em um sentido, e t_{j2j1}, defasagem ideal do sentido contrário. Ao final utiliza-se a média ponderada entre as duas ou alternativamente utiliza-se a defasagem (t_{j1j2} e t_{j2j1}) da pista mais congestionada.

Detalhes e outra proposta para o controle de defasagem podem ser vistos em (CARLSON, 2006).

2.3.3 Controle de Percentuais de Verde

Os percentuais de verde são utilizados para atender de forma harmoniosa todas as pistas da região, isto é, como se distribui os tempos de estágio de cada junção de modo a atender a demanda de cada pista. No caso de uma malha viária interligada esta tarefa ganha complexidade em função da influência que pistas impõem uma sobre as outras, necessitando então de adequadas técnicas de controle.

Sendo o módulo principal da técnica TUC, o objetivo básico é evitar o risco de sobre-saturação das vias de modo a evitar o bloqueio das vias à montante. O controle é multivariável, sendo os estados o número de veículos em cada pista da área controlada e ação de controle o tempo de verde de cada estágio de cada interseção.

A dinâmica do sistema é obtida através do modelo *store-and-forward* (DIAKAKI et al., 2002), obtendo um modelo linear do sistema. O controle é feito através de uma uma matriz de realimentação de estados, sendo estados a fila de cada via, obtida pela metologia do Regulador Quadrático Linear (LQR) (DIAKAKI et al., 2002). Na sequência a construção do modelo *store-and-forward* e a síntese do controlador são apresentados.

MODELAGEM *STORE AND FORWARD*

A malha viária é representada por um grafo direcionado com as pistas $z \in Z$ e as interseções semaforizadas $j \in J$. Cada junção j possui um conjunto de pistas de entrada e pistas de saída, respectivamente I_j e O_j e um tempo de ciclo C_j frequentemente, mas não necessariamente, o mesmo para as interseções da malha. A configuração da interseção consiste em um conjunto de números de estágios, E_j, onde $e_z \subseteq E_j$ denota os estágios que a pista $z \in I_j$ possui direito de passagem. Finalmente, os parâmetros de tráfego são os fluxos de saturação s_z da pista z e as taxas de conversão t_{ij}, a proporção de carros que saem da pista i e entram na pista j.

O modelo leva em consideração o balanço de fluxo em cada pista. Uma saída em uma pista à montante causa uma entrada na pista à jusante de acordo com a respectiva taxa de conversão. Considere a pista z conectando as interseções M e N na rede da Figura 5, tal que $z \in O_M$ e $z \in I_N$. A dinâmica

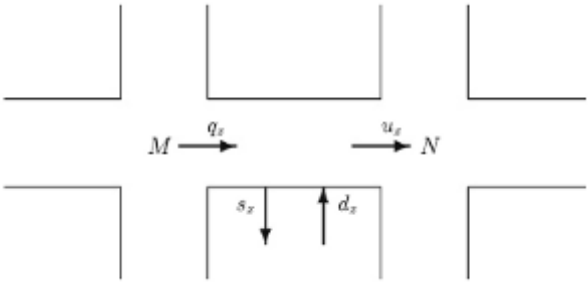

Figura 5: Balanço de fluxo em uma pista (retirado de (DIAKAKI et al., 2002)).

da pista z é dada pela equação da conservação:

$$x_z(k+1) = x_z(k) + T_s(q_z(k) - s_z(k) + d_z(k) - u_z(k)) \tag{2.4}$$

onde $x_z(k)$ é a fila (número de veículos) na pista z no instante kT, $q_z(k)$ e $u_z(k)$ são respectivamente os fluxos de entrada e saída da pista, enquanto $d_z(k)$ e $s_z(k)$ são os fluxos de entrada e saída da própria pista (saída e entrada de estacionamento por exemplo). T é o período de amostragem, sendo igual ao tempo de ciclo nesta abordagem. O fluxo de entrada para a pista z é dado por:

$$q_z(k) = \sum_{w \in I_M} t_{w,z} u_w \tag{2.5}$$

e o fluxo de saída por:

$$u_z = \frac{G_z(k) s_z}{C} \tag{2.6}$$

onde $G_z(k)$ é o verde dado à pista z, calculado conforme segue:

$$G_z(k) = (ou \leq) \sum_{i \in p_z} g_{j,i}(k) \tag{2.7}$$

sendo $g_{j,i}$ o tempo do estágio i da junção j.

Observe que o tempo de verde $G_z(k)$ gera o escoamento de veículos da pista. Em outras palavras, o modelo pressupõe que enquanto o verde está habilitado, os veículos escoam no fluxo de saturação, existindo ou não carros na pista. Na modelagem por controle preditivo isto é tratado inserindo explicitamente restrições para evitar que o número de veículos na fila se torne negativo.

O modelo descrito nas equações (2.4)-(2.7) correlaciona o sinal de

controle, u_z, e os estados, x_z. Manipulando-se as variáveis das equações acima e ignorando as perturbações, chega-se a seguinte representação compacta do modelo *store-and-forward*:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{g}(k) \tag{2.8}$$

sendo $\mathbf{x}(k)$ um vetor com o número de veículos em todas as pistas e $\mathbf{g}(k)$ um vetor com tempo de todos os estágios, organizados da seguinte forma:

$$\mathbf{g}(k) = \begin{bmatrix} g_{1,1} & g_{1,2} & g_{2,1} & \cdots & g_{j,i} \end{bmatrix}$$

e B uma matriz construída a partir da modelagem apresentada. Segundo a equação (2.4), $A = I$, portanto chega-se ao seguinte modelo dinâmico:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + B\mathbf{g}(k) \tag{2.9}$$

SÍNTESE DE CONTROLE DE PERCENTUAIS DE VERDE POR LQR

TUC utiliza o modelo *store and forward* para síntese do controle. A partir da equação (2.9) uma lei de controle multivariável é sintetizada através da metodologia LQR (*Linear Quadratic Regulator*). Esta técnica consiste em projetar um controle linear (isto é, obter L) em malha fechada caracterizado pela equação:

$$\mathbf{u}(k) = -L\mathbf{x}(k) \tag{2.10}$$

de forma a minimizar uma função custo, ponderando os desvios do estado de referência e a ação de controle em um horizonte infinito. Mais detalhes sobre a técnica podem ser obtidos em (DORATO et al., 1995).

No controle dos tempos de estágio, a função custo é definida segundo a equação:

$$J = \sum_{k=0}^{\infty} \left(\mathbf{x}(k)' Q \mathbf{x}(k) + \mathbf{u}(k)' R \mathbf{u}(k) \right) \tag{2.11}$$

onde Q é uma matriz positiva semidefinida com a ponderação associada aos estados, enquanto R é uma matriz positiva definida com a ponderação associada ao esforço de controle.

Embora a soma dos tempos de estágio deva ser exatamente igual ao tempo de ciclo, o controle é tratado como irrestrito na lei de controle proposta. Esta restrição e os limites nos tempos de estágio são tratados a posteriori.

Uma vez que a estratégia TUC busca evitar a sobressaturação das pis-

tas, a matriz Q é diagonal com elementos dados por:

$$Q_{ii} = \frac{1}{x_i^{\max}} \tag{2.12}$$

onde $x_i^{\max}$ é a quantidade de carros que podem ser armazenados na pista i, isto é, a máxima fila permitida. Desta forma o controle procura esvaziar as pistas e balancear a razão $x_i/x_i^{\max}$, evitando assim a sobressaturção.

A matriz R é escolhida como rI, com r definido por tentativa e erro (DIAKAKI et al., 2002). Quanto maior o valor de r, menos agressivo será o controle. O valor r é considerado adequado quando se obtém uma solução de compromisso, com resposta rápida à variação de tráfego e resposta adequada às perturbações. Valores de r excessivamente baixos podem tornar a resposta do controle exagerada às perturbações, pois o controle é irrestrito e permite valores de tempos de estágio que podem não fazer sentido como, por exemplo, valores maiores que o tempo de ciclo.

Os tempos de verde $\mathbf{g}(k)$ dependem também de valores nominais $\mathbf{g}_{\text{nom}}$ conforme a equação:

$$\mathbf{g}(k) = \mathbf{g}_{\text{nom}} - L\mathbf{x} \tag{2.13}$$

Após a realimentação dos estados e o cáclulo dos valores g_{ji} segundo a equação (2.13), as restrições são tratadas em cada interseção j resolvendo o seguinte problema de otimização:

$$\underset{G_{j,i}}{\text{minimize}} \sum_{i \in F_j} (g_{j,i} - G_{j,i})^2 \tag{2.14a}$$

sujeito a:

$$\sum_{i \in F_j} G_{j,i} + L_j = C \tag{2.14b}$$

$$G_{j,i} \in [g_{j,i}^{\min}, g_{j,i}^{\max}],\ \forall i \in F_j. \tag{2.14c}$$

onde $G_{j,i}$ é o tempo do estágio i da interseção j a ser efetivamente implementado pelo controlador, L_j é o tempo perdido por ciclo (*i.e.*, a soma dos períodos de entreverdes de todos os estágios), e $g_{j,i}^{\min}$ e $g_{j,i}^{\max}$ definem o mínimo e o máximo do tempo do estágio i. Este problema pode ser resolvido de maneira eficiente por métodos iterativos.

SÍNTESE DE CONTROLE DE PERCENTUAIS DE VERDE POR MPC

Controle Preditivo baseado em modelo (MPC) é uma técnica de controle recente, cuja difusão na indústria ocorreu na década de 80 (RICO; CA-

MACHO, 2006). Em vez de utilizar uma matriz de realimentação de estados $u(k) = -L\mathbf{x}(k)$, a técnica MPC obtém o sinal $\mathbf{u}(k)$ através da resolução de um problema de otimização que busca minimizar uma função de custo, de maneira semelhante à técnica LQR, porém incluindo restrições ao problema de otimização, em oposição a um tratamento de forma *ad hoc* de adequação de restrições, comumente visto em controle clássico, como no caso do controle de tempos de verde utilizando LQR, onde a adequação às restrições é feita resolvendo o problema (2.14).

O controle preditivo busca não apenas tratar os efeitos imediatos da ação de controle, mas também contabilizar seu efeito de longo prazo. Levando em consideração o estado inicial (presente) do sistema, a partir do controle aplicado nos T seguintes períodos de amostragem $\mathbf{U} = (\mathbf{u}(k), \mathbf{u}(k+1), \ldots, \mathbf{u}(k+T-1))$ e do modelo dinâmico do sistema ($\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k)$) é possível obter $\mathbf{X} = (\mathbf{x}(k+1), \mathbf{x}(k+2), \ldots, \mathbf{x}(k+T))$ (predição). Um problema de otimização que embute o modelo, as restrições e a função objetivo, tendo como variáveis de decisão a sequência de ações de controle $\mathbf{U}$ é instanciado e resolvido, isto é, obtém-se a sequência $\mathbf{U}^\star$ ótima. Apenas $\mathbf{u}(k)^\star$ é aplicado no sistema no instante k. No período de amostragem seguinte, em $k+1$, $\mathbf{x}(k+1)$ é medido e o mesmo procedimento é realizado com referência ao ponto de amostragem $k+1$.

Esta técnica também é conhecida por *horizonte deslizante* ou *horizonte móvel*, pois o procedimento leva em consideração n períodos de amostragem, do instante k até o instante $k+n$, isto é, seu *horizonte de predição*. No instante seguinte, o horizonte é *deslizado* ou *rolado* para a frente de forma a se estender do instante $k+1$ até o instante $k+n+1$. Mais detalhes sobre a técnica serão apresentados no capítulo seguinte.

2.4 SUMÁRIO

Este capítulo apresentou as principais formas de controle de tráfego veicular, compreendendo as técnicas de controle a tempo fixo, atuado, seleção de planos e tempo real. Após uma descrição sucinta das técnicas de controle tempo real SCOOT e SCATS, a modelagem *store-and-forward* e a estratégia de controle TUC foram apresentadas em detalhes. Enquanto a abordagem *store-and-forward* será utilizada na proposta de controle preditivo distribuído, a estratégia TUC servirá de base para comparação nos próximos capítulos.

3 OTIMIZAÇÃO E CONTROLE PREDITIVO DISTRIBUÍDO

Este capítulo desenvolve uma abordagem de controle preditivo baseado em modelo voltada ao controle dos percentuais de verde da técnica TUC. Primeiramente, os elementos básicos e o princípio de funcionamento da técnica de controle preditivo são apresentados. Na sequência, o problema de ajuste dos percentuais de verde é colocado como um problema de controle preditivo. Na sequência a modelagem por redes dinâmicas lineares é apresentada em conjunto com os algoritmos centralizado e distribuído para esta modelagem.

3.1 CONTROLE PREDITIVO BASEADO EM MODELO – MPC

O termo controle preditivo baseado em modelo não denota uma técnica de controle específica. É uma gama de métodos de controle que utilizam explicitamente o modelo do sistema para obter o sinal de controle, minimizando uma função custo (RICO; CAMACHO, 2006). O objetivo de controle, incluído de alguma forma na função custo, é, em geral, minimizar desvios em relação às referências das variáveis controladas com o mínimo de esforço de controle. As principais características são (CAMACHO; BORDONS, 2004):

- uso explícito de um modelo do sistema para predição de saídas em instantes futuros (horizonte);
- cálculo de uma sequência de controle $\mathbf{U} = (\mathbf{u}(k), \mathbf{u}(k+1), \ldots, \mathbf{u}(k+n-1))$ minimizando uma função objetivo (custo);
- a cada instante de controle, uma sequência de controle é calculada minizando uma função objetivo. O primeiro elemento da sequência é aplicado ao sistema. No instante de amostragem seguinte o processo é efetuado novamente, porém "deslizado" de um período de amostragem;
- o cálculo da sequência de controle é realizado por um algoritmo de otimização adequado.

MPC apresenta uma série de vantagens em relação às técnicas clássicas. Uma das principais é o tratamento explícito de restrições, tanto físicas quanto operacionais. Desta forma restrições tratadas de forma *ad hoc* em controle clássico são tratadas explicitamente em controle preditivo. Portanto, é possível especificar restrições como valores máximos e mínimos para as entradas de

controle e de saída, saídas sem sobre-sinal, *etc*. Outras vantagens são (CAMACHO; BORDONS, 2004):

- trata naturalmente sistemas com atraso de transporte;
- estabilidade é sempre garantida para sistemas lineares. Sob certas condições, há garantias de estabilidade para sistemas não-lineares;
- induz à bom desempenho no seguimento de referência em função do uso de valores de referência futuros no cálculo do controle;
- consegue responder a variações paramétricas do sistema (alterações das matrizes A e B) através de técnicas de identificação.

Sua principal desvantagem é o longo tempo de computação do controle, impossibilitando seu uso, ao menos atualmente, em plantas com dinâmicas rápidas. Pesquisas caminham para algoritmos cada vez mais eficientes para o controle preditivo. Para sistemas lineares (ou linearizados) com restrições lineares, o cálculo da ação de controle requer a solução de um problema de otimização de programação quadrática. Para esta classe de problemas existem algoritmos eficientes para resolução do problema, alguns dos quais são apresentados no presente trabalho.

3.1.1 Estratégia MPC para Controle Multivariável

Para caracterização de controle preditivo são necessários três elementos (CAMACHO; BORDONS, 2004):

- modelo de predição;
- função objetivo;
- lei de controle.

MODELO DE PREDIÇÃO

Um modelo de predição é utilizado para obter $\mathbf{x}(t+k|t)$, isto é, o valor estimado para $\mathbf{x}$ no instante $t+k$ a partir da **medida x** dos estados no instante t e as **ações de controle u** do instante t até $t+k-1$. Dentre as diversas modelagens possíveis, utiliza-se aqui a modelagem multivariável por espaço de estados discreta:

$$\mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) + \mathbf{d}(k) \tag{3.1}$$

onde A e B são matrizes e $\mathbf{d}$ é um vetor com as perturbações, todos com dimensões apropriadas.

Considera-se que todos os estados podem ser medidos ou ao menos estimados. A aplicação recursiva da equação (3.1) pode ser utilizada para o cálculo do valor estimado dos estados em instantes futuros.

FUNÇÃO OBJETIVO

A função objetivo norteia o comportamento esperado em malha fechada do sistema. Aplicações de controle apresentam, com diferenças de importância dependendo do processo, os seguintes objetivos:

- **Seguimento de referência**: tornar os valores de $\mathbf{x}$ mais próximos possíveis de um valor de referêcia desejado $\mathbf{w}$;
- **Resposta a perturbações**: fazer com que os estados voltem rapidamente à referência, após a ocorrência de uma perturbação no sistema.
- **Minimização das ações de controle**: busca-se controlar o sistema com o mínimo de esforço de controle possível. Em geral, valores absolutos altos de $\mathbf{u}(k)$ e variações $\mathbf{u}(k)-\mathbf{u}(k-1)$ são minimizados ou limitados através das restrições. Em controle de tráfego, por exemplo, a variação dos tempos de verde entre ciclos é indesejada.

A função objetivo deve expressar matematicamente os objetivos de desempenho. Aqui define-se a função objetivo para o problema de regulação (referência nula), ponderando os estados e a ação de controle:

$$J = \sum_{k=0}^{T} \left(\mathbf{x}(k)' Q \mathbf{x}(k) + \mathbf{u}(k)' R \mathbf{u}(k) \right) \tag{3.2}$$

onde T é o horizonte de predição, isto é, quantos instantes à frente o controle está levando em consideração.

A sintonia do controlador pode ser realizada através de alterações na função objetivo:

- um horizonte de predição, T, baixo pode fazer com que o controle negligencie efeitos que podem ocorrer no futuro, enquanto um horizonte longo pode progagar erros de modelagem;
- A ponderação de matrizes Q e R mexe diretamente na dinâmica dos estados. Valores da matriz R baixos permitem controles agressivos, assim como valores baixos na matriz Q tornam a dinâmica dos estados suave.

Existe também a possibilidade de se priorizar um estado em detrimento do outro, de acordo com valores relativos de suas ponderações.

Existem outras propostas para definição da função objetivo não abordadas aqui. Um exemplo é o emprego de matrizes Q e R variantes ao longo do horizonte. Outro exemplo ocorre em controle de processos, onde é comum horizontes de controle e predição diferentes (RICO; CAMACHO, 2006).

MPC permite a inclusão explícita de restrições, como valores máximos e mínimos para ações de controle. Por modelagem pode-se incluir restrições nos estados, embora isto se reflita na ação de controle, pois esta é a variável de decisão. Consideramos aqui restrições como uma série de inequações lineares como segue:

$$F_x\mathbf{x}(k) + F_u\mathbf{u}(k) \leq \mathbf{f} \tag{3.3}$$

Sendo cada linha da equação 3.3 uma restrição de desigualdade do problema. Sendo F_x, F_u as matrizes que definem, respectivamente, a dependência com relação aos estados e entrada de controle e f o termo independente.

Na existência de restrições de igualdade, estas podem ser suprimidas eliminando-se variáveis tornando o problema apenas com restrições de desigualdade. Portanto não há perda generalidade em não considerá-las.

SÍNTESE DA LEI DE CONTROLE

No MPC clássico, a obtenção do sinal de controle ótimo pode ser dispendiosa dependendo do modelo de predição, da função objetivo e das restrições impostas ao modelo. Para encontrar a (uma) solução é necessário utilizar algoritmos de otimização para buscar $\mathbf{u}$ ótimo. Em muitos casos é possível encontrar a ação de controle ótima, porém em tempo computacional não compatível com o sistema a ser controlado.

Para obter o valor de $\mathbf{u}(k)$ é necessário calcular o valor de $\mathbf{u}$ que minimize (3.2) sujeito às restrições (3.1) e (3.3).

3.2 MPC PARA CONTROLE DE PERCENTUAIS DE VERDE DO TUC

Na aplicação ao problema alvo, utiliza-se MPC para obter $\mathbf{g}(k)$. A filosofia é parecida com a metodologia LQR apresentada no capítulo anterior, com modelagem também realizada por *store and forward*, porém difere nos seguintes itens:

- em vez do horizonte ser infinito, o horizonte é finito;

- as restrições são tratadas explicitamente no cálculo da ação de controle, não necessitando de tratamento posterior para adequar os sinais de controle às restrições;

- em vez da dinâmica estar relacionada com os tempos de estágio, na abordagem por MPC os tempos de verde das pistas são variáveis independentes que não podem exceder a soma dos tempos dos estágios durante os quais as pistas têm direito de passagem.

A função objetivo se torna:

$$J = \sum_{k=0}^{T} \left(\mathbf{x}(k)' Q \mathbf{x}(k) + \mathbf{g}(k)' R \mathbf{g}(k) \right) \tag{3.4}$$

onde T é o horizonte de predição e $\mathbf{g}$ é o sinal de controle, contendo não apenas os tempos dos estágios, mas também os verdes de cada pista:

$$\mathbf{g}(k) = \begin{bmatrix} g_{1,1} & g_{1,2} & g_{2,1} & \dots & g_{j,i} & G_1 & \dots & G_z \end{bmatrix}'$$

onde $g_{j,i}$ é o tempo do estágio i da junção j e G_z é o tempo de verde da pista z.

Quanto às restrições presentes no modelo, os estados não podem assumir valores negativos:

$$\mathbf{x}(k) \geq 0,\ 0 \leq k \leq T \tag{3.5}$$

O verde da pista deve ser não negativo e não pode exceder o tempo dos estágios que os habilitam:

$$0 \leq G_z(k) \leq \sum_{i \in e_z} g_{j,i}(k) \tag{3.6}$$

e a soma dos tempos de estágio deve ser igual ao tempo de ciclo:

$$\sum_{i \in E_j} g_{j,i}(k) + L_j = C_j \tag{3.7}$$

O tempo de estágio deve ser maior ou igual a um valor mínimo para evitar valores baixos de tempo de verde, seja para evitar que tempo insuficiente para travessia de pedestres; ou que o tempo de verde de qualquer estágio seja menor ou igual ao tempo de reação:

$$g_{j,i}(k) \geq g_{j,i}^{\min} \tag{3.8}$$

A função custo (3.4) associada às restrições (3.5), (3.6), (3.7) e (3.8) constitui um problema de otimização de programação quadrática. Como resultado da resolução do problema, obtém-se os valores estimados de $\mathbf{x}(k)$ em todos os instantes do horizonte, bem como os valores dos tempos de verde dos estágios, $g_{j,i}(k)$, e das pistas, $G_z(k)$. Para resolução do problema de otimização, *solvers* (software específico para resolução de problemas de otimização) comerciais podem ser utilizados e no caso do presente trabalho, uma plataforma foi desenvolvida para permitir a resolução do problema de forma centralizada e distribuída.

3.3 REDES DINÂMICAS LINEARES

Uma rede dinâmica linear (RDL) é um grafo direcionado $G = (\mathscr{V}, \mathscr{E})$, $\mathscr{V} = \{1, \ldots, M\}$ e $E \subseteq \mathscr{V} \times \mathscr{V}$, cujos nós representam subsistemas e cujos arcos modelam a influência direta entre subsistemas. O conjunto $M(m) = \{i : (i,m) \in E\} \cup \{m\}$ é chamado *vizinhança de entrada* do subsistema m contendo os subsistemas que afetam sua dinâmica. O estado do subsistema m é $\mathbf{x}_m \in \mathbb{R}^{n_m}$ enquanto o vetor de controle é $\mathbf{u}_m \in \mathbb{R}^{p_m}$, onde n_m e p_m são o número de entradas e o número de saídas do subsistema.

A dinâmica do subsistema m é dada pela seguinte equação em tempo discreto:

$$\mathbf{x}_m(k+1) = \sum_{i \in M(m)} \left(A_{m,i}\mathbf{x}_i(k) + B_{m,i}\mathbf{u}_i(k) \right) \tag{3.9}$$

Sabendo o estado $\mathbf{x}(k) = (\mathbf{x}_1, \ldots, \mathbf{x}_M)(k)$ no tempo k, o estado do subsistema m no tempo t é dado por:

$$\begin{aligned} \mathbf{x}_m(k+t) = & \sum_{\omega_t \in \pi(m,t)} A(\omega_t)\mathbf{x}_{i_t}(k) + \sum_{\omega_t \in \pi(m,t)} B(\omega_t)\mathbf{u}_{i_t}(k) + \\ & \sum_{\omega_{t-1} \in \pi(m,t-1)} B(\omega_{t-1})\mathbf{u}_{i_{t-1}}(k+1) + \cdots \\ & + \sum_{\omega_1 \in \pi(m,1)} B(\omega_1)\mathbf{u}_{i_1}(k+t-1) \end{aligned} \tag{3.10}$$

onde:

- $\pi(m,t) = \{\langle i_0, i_1, \ldots, i_t \rangle : i_0 = m, i_1 \in M(i_0), i_2 \in M(i_1), \ldots, i_t \in M(i_{t-1})\}$ define quais subsistemas i_t afetam, t períodos de amostragem à frente, o estado do subsistema m;

- $A(\omega_t) = A_{i_0,i_1}A_{i_1,i_2}\ldots A_{i_{t-1},i_t}$ é uma matriz que define como o estado do subsistema i_t no instante k influencia o estado do subsistema i_0 no tempo $k+t$, onde $\omega_t = \langle i_0, i_1, \ldots, i_t\rangle$;

- $B(\omega_t) = A_{i_0,i_1}A_{i_1,i_2}\ldots A_{i_{t-2},i_{t-1}}B_{i_{t-1},i_t}$ é uma matriz que modela como a entrada de controle do subsistema i_t no instante k influencia o estado do subsistema i_0 no instante $k+t$.

Para a rede com $M = 6$ subsistemas, dado na Figura 6, $M(1) = \{1\}$, $M(2) = \{1,2\}, M(3) = \{2,3,6\}$ e $M(4) = \{4\}$. Já $\pi(3,1) = \{\langle 3,2\rangle, \langle 3,3\rangle, \langle 3,6\rangle\}$ e $\pi(3,2) = \{\langle 3,2,1\rangle, \langle 3,2,2\rangle, \langle 3,3,2\rangle, \langle 3,3,3\rangle, \langle 3,3,6\rangle, \langle 3,6,3\rangle, \langle 3,6,5\rangle, \langle 3,6,6\rangle\}$. As cadeias $\pi(3,2)$ induzem as matrizes dinâmicas $A(\langle 3,2,1\rangle) = A_{3,2}A_{2,1}$, ..., $A(\langle 3,6,6\rangle) = A_{3,6}A_{6,6}$ e as matrizes de controle $B(\langle 3,2,1\rangle) = A_{3,2}B_{2,1}$,..., $B(\langle 3,6,6\rangle) = A_{3,6}B_{6,6}$.

A saída $\mathbf{y}_m \in \mathbb{R}^{q_m}$ do subsistema m, onde q_m é o número de saídas deste, depende dos estado e dos sinais de controle próprios e dos subsistemas em sua vizinhança como segue:

$$\mathbf{y}_m(k+1) = \sum_{i\in M(m)} (W_{m,i}\mathbf{x}_i(k) + Z_{m,i}\mathbf{u}_i(k)) \tag{3.11}$$

onde $W_{m,i}$ e $Z_{m,i}$ são matrizes que definem as saídas do sistema. A saída do subsistema m no instante $k+t+1$ é uma função do estado da rede no instante k e dos sinais do controle de k até $k+t$ dado por:

$$\begin{aligned}\mathbf{y}_m(k+t+1) = \sum_{i\in M(m)} W_{m,i}\Bigg(&\sum_{\omega_t\in\pi(m,t)} A(\omega_t)\mathbf{x}_{i_t}(k) \\ &+ \sum_{\omega_t\in\pi(m,t)} B(\omega_t)\mathbf{u}_{i_t}(k) + \cdots + \sum_{\omega_1\in\pi(m,1)} B(\omega_1)\mathbf{u}_{i_1}(k+t-1)\Bigg) \\ &+ \sum_{i\in M(m)} Z_{m,i}\mathbf{u}_i(k+t)\end{aligned} \tag{3.12}$$

onde $\pi(m,t)$ tem apenas a matriz identidade se $t = 0$, *i.e.*, $\pi(m,0) = \{\omega_0\}$ e $A(\omega_0) = I$. Portanto a Eq. (3.12) se torna (3.11) quando $t = 0$.

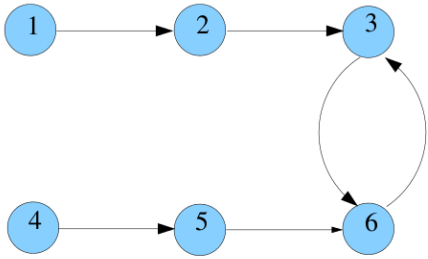

Figura 6: Rede Dinâmica Linear com 6 agentes (retirado de (CAMPONOGARA; OLIVEIRA, 2009)).

3.3.1 Modelagem Centralizada

O foco deste trabalho está na solução distribuída do seguinte problema de otimização voltado ao controle preditivo de RDLs:

$$P: \quad \min \sum_{m=1}^{M} f_m = \sum_{m=1}^{M} \sum_{t=k}^{k+T-1} \frac{1}{2} \left[\mathbf{y}_m(t+1)' Q_m \mathbf{y}_m(t+1) + \mathbf{u}_m(t)' R_m \mathbf{u}_m(t) \right] \quad (3.13a)$$

$$\text{s.a}: \text{ para todo } m \in \mathscr{M},\ t = k, \ldots, k+T-1:$$

$$\mathbf{x}_m(t+1) = \sum_{i \in M(m)} \left(A_{m,i} \mathbf{x}_i(t) + B_{m,i} \mathbf{u}_i(t) \right) \quad (3.13b)$$

$$\mathbf{y}_m(t+1) = \sum_{i \in M(m)} \left(W_{m,i} \mathbf{x}_i(t) + Z_{m,i} \mathbf{u}_i(t) \right) \quad (3.13c)$$

$$\mathbf{y}_m^{\min} \leq \mathbf{y}_m(t+1) \leq \mathbf{y}_m^{\max} \quad (3.13d)$$

$$\mathbf{u}_m^{\min} \leq \mathbf{u}_m(t) \leq \mathbf{u}_m^{\max} \quad (3.13e)$$

com restrições nos sinais de controle ($u_m^{\min}$ e $u_m^{\max}$ valores mínimos e máximos máximos) e na saída ($y_m^{\min}$ e $y_m^{\max}$ valores mínimos e máximos máximos) e $\mathscr{M} = \{1, \ldots, M\}$, Q_m são simétricas (denotada por $Q_m = Q_m'$) e positivas semi-definidas (denotada por $Q_m \succeq 0$) e $R_m = R_m'$ são positivas definidas (denotada por $R_m \succ 0$). P é resolvido em cada instante de tempo k. Os sinais de controle referentes ao intervalo $[t_k, t_{k+1}]$ (instante k) são aplicados. No

período de amostragem seguinte, o horizonte é rolado para frente, P é instanciado para o instante $k+1$ até $k+T+1$ e o processo é repetido. Detalhes em como obter esta formulação a partir da estrutura da Rede Dinâmica Linear, assim como as seguintes, são encontradas em (CAMPONOGARA; SCHERER, 2011).

Seja $\widehat{\mathbf{u}}_m = [\mathbf{u}_m(k)' \cdots \mathbf{u}_m(k+T-1)']'$ o vetor com as predições de controle para horizonte de comprimento T. Definições análogas são utilizadas para definir os vetores $\widehat{\mathbf{x}}_m$, $\widehat{\mathbf{u}}_m^{\max}$, $\widehat{\mathbf{u}}_m^{\min}$, $\widehat{\mathbf{y}}_m^{\min}$ e $\widehat{\mathbf{y}}_m^{\min}$ e também para as matrizes $\widehat{A}_{mi}$, $\widehat{B}_{mi}$, $\widehat{W}_{mi}$ e $\widehat{Z}_{mi}$ representando dinâmicas e saídas sobre todo o horizonte. Como obter as matrizes pode ser visto em (CAMPONOGARA; SCHERER, 2011). Usando as equações (3.10) e (3.12) para escrever estado e saída como função dos sinais de controle, o problema de MPC pode ser colocado na seguinte forma:

$$P: \min \quad f(\widehat{\mathbf{u}}) = \frac{1}{2} \sum_{m \in \mathscr{M}} \sum_{i \in \widehat{M}(m)} \sum_{j \in \widehat{M}(m)} \widehat{\mathbf{u}}_i' H_{m,i,j} \widehat{\mathbf{u}}_j + \sum_{m \in \mathscr{M}} \sum_{i \in \widehat{M}(m)} \mathbf{g}_{m,i}' \widehat{\mathbf{u}}_i + \sum_{m \in \mathscr{M}} c_m \tag{3.14a}$$

$$\text{s.a : para todo } m \in \mathscr{M} :$$

$$\widehat{\mathbf{y}}_m^{\min} \leq \sum_{i \in \widehat{M}(m)} (\widehat{Z}_{m,i} \widehat{\mathbf{u}}_i + \widehat{W}_{m,i} \mathbf{x}_i(k)) \leq \widehat{\mathbf{y}}_m^{\max} \tag{3.14b}$$

$$\widehat{\mathbf{u}}_m^{\min} \leq \widehat{\mathbf{u}}_m \leq \widehat{\mathbf{u}}_m^{\max} \tag{3.14c}$$

onde $H_{m,i,j}$ são matrizes, $\mathbf{g}_{m,i}$ são vetores e c_m são constantes obtidas da estrutura de P. O conjunto $\widehat{M}(m) = \{j : \langle i_0, i_1, \ldots, i_T \rangle \in \pi(m,T), i_T = j\}$ define a vizinhança de entrada do subsistema m para o horizonte de comprimento T. Para a rede dinâmica da Figura 6, $\widehat{M}(3) = M(3)$ para $T = 1$ enquanto $\widehat{M}(2) = \{1,2,3,4,5,6\}$ para $T = 2$. Para simplificar o projeto do algoritmo, P pode ser colocado de forma compacta:

$$P: \quad \min \quad f(\widehat{\mathbf{u}}) \tag{3.15a}$$

$$\text{s.a : } h_i(\widehat{\mathbf{u}}) \leq 0,\ i = 1, \ldots, p \tag{3.15b}$$

onde $\widehat{\mathbf{u}} = [\widehat{\mathbf{u}}_1' \cdots \widehat{\mathbf{u}}_M']'$, $f : \mathbb{R}^n \to \mathbb{R}$ define a função objetivo e $h_1, \ldots, h_p : \mathbb{R}^n \to \mathbb{R}$ definem as restrições dadas por (3.16c) e (3.14c), com $n = T \sum_{m \in \mathscr{M}} p_m$ sendo a dimensão de $\widehat{\mathbf{u}}$ e p sendo o número de desigualdades. Note que $H \succ 0$. O conjunto factível é $\Omega = \{\widehat{\mathbf{u}} : h_i(\widehat{\mathbf{u}}) \leq 0,\ i = 1, \ldots, p\}$, enquanto seu interior é $\overline{\Omega} = \{\widehat{\mathbf{u}} : h_i(\widehat{\mathbf{u}}) < 0,\ i = 1, \ldots, p\}$.

Proposição 1. *$f(\widehat{\mathbf{u}})$ é fortemente convexa (CAMPONOGARA; SCHERER,*

2011).

3.3.2 Modelagem Distribuída

Uma decomposição do problema MPC P em um conjunto $\{P_m : m \in V\}$ (ou simplesmente $\{P_m\}$) de subproblemas é desenvolvida a seguir. Para qualquer agente i e comprimento T do horizonte de predição, seja:

- $\widehat{N}(m) = \{j : m \in M(j)\}$ a *vizinhança de saída* que consiste dos subsistemas que são afetados pelas decisões do subsistema m;

- $\tilde{N}(m) = \{i : \exists\ j \neq i \mid i,m \in \widehat{N}(j) - (\widehat{M}(m) \cup \widehat{N}(m))$ a *vizinhança indireta* que compreende os subsistemas i que não influenciam e não são influenciados pelo subsistema m, mas ambos afetam algum outro subsistema;

- $N(m) = (\widehat{M}(m) \cup \widehat{N}(m) \cup \tilde{N}(m)) \setminus \{m\}$ a *vizinhança* que engloba os subsistemas que estão acoplados ao subsistema m.

Na rede dinâmica da Figura 6, o subsistema 1 tem $\widehat{M}(1) = \{1\}$, $\widehat{N}(1) = \{1,2,3\}$, $\tilde{N}(1) = \{5,6\}$ e $N(1) = \{2,3,5,6\}$ enquanto o subsistema tem $\widehat{M}(3) = \{1,2,3,5,6\}$, $\widehat{N}(3) = \{4,5\}$, $\tilde{N}(3) = \{1,2,3,4,5,6\}$ com horizonte $T = 2$.

A decomposição $\{P_m\}$ é dita *perfeita* se cada P_m é obtido a partir de P removendo da função objetivo todos os termos, e descartando todas as restrições, que não dependam de $\widehat{\mathbf{u}}_m$ (CAMPONOGARA; TALUKDAR, 2005). Para uma decomposição perfeita, a visão que cada agente m possui da rede é dividida em:

- *variáveis locais:* as variáveis $\widehat{\mathbf{u}}_m$ definidas pelo agente m;

- *variáveis vizinhas:* o vetor $\widehat{\mathbf{w}}_m = (\widehat{\mathbf{u}}_j : j \in N(m))$ com as variáveis definidas pelos agentes vizinhos;

- *variáveis remotas:* o vetor $\widehat{\mathbf{r}}_m = (\widehat{\mathbf{u}}_j : j \notin N(m) \cup \{m\})$ com todas as outras variáveis.

Com uma decomposição perfeita, $\widehat{\mathbf{u}} = (\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m, \widehat{\mathbf{r}}_m)$, o subproblema P_m é definido como:

$$P_m(\widehat{\mathbf{w}}_m): \ \min \ f_m(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m) = \frac{1}{2}\widehat{\mathbf{u}}_m' \widehat{H}_m \widehat{\mathbf{u}}_m + \widehat{\mathbf{g}}_m' \widehat{\mathbf{u}}_m + \hat{c}_m \tag{3.16a}$$

$$\text{s.a}: \tag{3.16b}$$

$$\widehat{\mathbf{y}}_m^{\min} \leq \sum_{j \in \widehat{M}(m)} (\widehat{Z}_{m,j}\widehat{\mathbf{u}}_j + \widehat{W}_{m,j}\mathbf{x}_j(k)) \leq \widehat{\mathbf{y}}_m^{\max} \tag{3.16c}$$

$$\widehat{\mathbf{u}}_m^{\min} \leq \widehat{\mathbf{u}}_m \leq \widehat{\mathbf{u}}_m^{\max} \tag{3.16d}$$

onde

$$\widehat{H}_m = \sum_{j \in \widehat{N}(m)} \widehat{H}_{j,m,i} \tag{3.17}$$

$$\widehat{\mathbf{g}}_m = \sum_{j \in \widehat{N}(m)} \widehat{\mathbf{g}}_{j,m} + \frac{1}{2} \sum_{j \in \widehat{N}(m)} \sum_{l \in I(j) - \{m\}} (\widehat{H}'_{j,l,m} + \widehat{H}_{j,m,l})\widehat{\mathbf{u}}_l \tag{3.18}$$

Para facilitar o projeto de algoritmos, P_m é colocado na seguinte forma:

$$P_m(\widehat{\mathbf{w}}_i): \quad \min \quad f_m(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m) \tag{3.19a}$$

$$\text{s.a}: \ h_j(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m) \leq 0, \ j \in \mathscr{H}_m \tag{3.19b}$$

onde $\mathscr{H}_m = \{m : h_m \text{ é uma função de } \widehat{\mathbf{u}}_m\}$. Para todo $j \in \mathscr{H}_m$, h_j não depende de $\widehat{\mathbf{r}}_m$ quando a decomposição é perfeita. Além disso, h_j não é uma função de $\widehat{\mathbf{u}}_m$, para todo $j \notin \mathscr{H}_m$.

Corolário 1. *$f_m(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m)$ é fortemente convexa em $(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m)$.*

3.4 ALGORITMO DISTRIBUÍDO

Uma questão relevante é como se resolve o conjunto $\{P_m\}$ de subproblemas com uma rede de agentes. Estratégias iterativas em que um agente m reaje às decisões dos seus vizinhos na iteração k, $\widehat{\mathbf{w}}_m^{(k)}$, com uma solução $\widehat{\mathbf{u}}_m^{(k+1)}$ para $P_m(\widehat{\mathbf{w}}_m^{(k)})$ pode levar a um comportamento indesejável devido às restrições de acomplamento segundo as desigualdades (3.16c). Estratégias que permitem que grupos de agentes iterem em paralelo quando eles são não vizinhos podem convergir para pontos fixos não ótimos. Por outro lado, estratégias que permitem paralelismo completo (VENKAT et al., 2008) podem atingir soluções infactíveis.

A solução algorítmica apresentada neste trabalho consite do uso de um método de ponto interior para aproximar o problema (3.15a)–(3.15b) como um problema irrestrito, para o qual técnicas baseadas em gradiente podem ser aplicadas (BOYD; VANDENBERGHE, 2004). Esta aproximação leva em consideração as restrições por meio de uma função *barreira logarítmica*[1]:

$$\phi(\widehat{\mathbf{u}}) = -\sum_{m=1}^{p} \log(-h_m(\widehat{\mathbf{u}})) \tag{3.20}$$

O problema de aproximação é dito *problema de centralização* sendo definido por:

$$P(\varepsilon): \quad \min_{\widehat{\mathbf{u}} \in \operatorname{dom}\theta} \ \theta(\widehat{\mathbf{u}}) = f(\widehat{\mathbf{u}}) + \varepsilon\phi(\widehat{\mathbf{u}}) \tag{3.21}$$

onde $\operatorname{dom}\theta = \overline{\Omega}$ e o parâmetro $\varepsilon > 0$ especifica a precisão da aproximação. O método de barreira resolve $P(\varepsilon^{(l)})$ para $\varepsilon^{(l)}$ decrescente. A solução $\widehat{\mathbf{u}}(\varepsilon^{(l)})$ para o problema dos centros $P(\varepsilon^{(l)})$ tende à solução $\widehat{\mathbf{u}}^{\star}$ de P à medida que $\varepsilon^{(l)}$ se aproxima de 0. O Algoritmo 1 descreve o método de barreira logarítmica.

Algoritmo 1: Método de Barreira

entrada: uma solução estritamente factível $\widehat{\mathbf{u}}$, parâmetro inicial ε, taxa de decrescimento $0 < \mu < 1$ e tolerância τ
inicialização: $l := -1$ e $\varepsilon^{(0)} := \varepsilon$
1 **repeat**
2 $l := l + 1$
  centralização: obtenha $\widehat{\mathbf{u}}^{(l)}$ resolvendo $P(\varepsilon^{(l)})$ com uma solução inicial $\widehat{\mathbf{u}}$
3 **if** $\varepsilon^{(l)} > \tau/p$ **then**
4 $\widehat{\mathbf{u}} := \widehat{\mathbf{u}}^{(l)}$ e $\varepsilon^{(l+1)} := \mu\varepsilon^{(l)}$
5 **until** $\varepsilon^{(l)} \leq \tau/p$;
saída: $\widehat{\mathbf{u}}^{(l)}$

A influência de $\varepsilon^{(0)}$ e da taxa de decrescimento μ no desempenho do método de barreira é discutida em (BOYD; VANDENBERGHE, 2004). Um ponto factível inicial é obtido resolvendo um problema auxiliar, tendo como objetivo a minimização de uma variável de folga s e restrições $h_m(\widehat{\mathbf{u}}) \leq s$, $m = 1, \ldots, p$ e $s \geq 0$. Uma solução ótima para o problema auxiliar é factível para o problema original se seu objetivo tem valor $s = 0$.

[1] Uma função com valor real no interior do conjunto factível que tende para infinito à medida que a solução se aproxima da fronteira definida por qualquer restrição.

O algoritmo distribuído faz com que os agentes produzam uma série de iterandos $\widehat{\mathbf{u}}^{(l)}$ convergente à solução $\widehat{\mathbf{u}}(\varepsilon)$. Dadas as variáveis vizinhas $\widehat{\mathbf{w}}_m$, o problema de centralização do agente m é dado por:

$$P_m(\varepsilon, \widehat{\mathbf{w}}_m) : \min_{\widehat{\mathbf{u}}_m \in \operatorname{dom} \theta_m} \theta_m(\widehat{\mathbf{u}}_m) = f_m(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m) + \varepsilon \phi_m(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m)$$

onde

$$\phi_m(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m) = -\sum_{j \in \mathscr{H}_m} \log(-h_j(\widehat{\mathbf{u}}_m, \widehat{\mathbf{w}}_m))$$

é a barreira logarítmica para as restrições que dependem de $\widehat{\mathbf{u}}_m$. Dado $(\widehat{\mathbf{u}}_m^{(l)}, \widehat{\mathbf{w}}_m^{(l)})$, o agente m gera o próximo iterando $\widehat{\mathbf{u}}_m^{(l+1)}$ dando um passo $s_m^{(l)} > 0$ na direção de descenso $-\nabla\theta_m(\widehat{\mathbf{u}}_m^{(l)})$. Mais formalmente,

$$\widehat{\mathbf{u}}_m^{(l+1)} = \widehat{\mathbf{u}}_m^{(l)} - s_m^{(l)} \nabla \theta_m(\widehat{\mathbf{u}}_m^{(l)}) \tag{3.22}$$

A rede de agentes resolve $\{P_m(\varepsilon)\}$ com o Algoritmo 2. A série de iterandos $\widehat{\mathbf{u}}^{(l)}$ produzida pelo algoritmo distribuído converge para uma solução $\widehat{\mathbf{u}}(\varepsilon)$ de $P(\varepsilon)$. As seguintes hipóteses são necessárias para garantir convergência.

Hipótese 1. *A função $\theta(\widehat{\mathbf{u}})$ é fortemente convexa.*

Hipótese 2. (Trabalho Síncrono) *Se o agente m atualiza as variáveis na iteração l, então:*

1. *o agente m usa $\widehat{\mathbf{w}}_m = \widehat{\mathbf{w}}_m^{(l)}$ e utiliza o método de gradiente descendente para obter uma solução aproximada $\widehat{\mathbf{u}}_m^{(l+1)}$ para $P_m(\varepsilon, \widehat{\mathbf{w}}_m)$;*
2. *$\widehat{\mathbf{u}}_m^{(l)}$ não é uma solução ótima para $P_m(\varepsilon, \widehat{\mathbf{w}}_m)$; e*
3. *cada vizinho do agente m não itera, o que significa que $\widehat{\mathbf{u}}_j^{(l+1)} = \widehat{\mathbf{u}}_j^{(l)}$ para todo $j \in N(m)$.*

Hipótese 3. (Trabalho Máximo e Contínuo) *Seja $J(l) = \{m \in (M) : \|\nabla\theta_m(\widehat{\mathbf{u}}_m^{(l)})\| = \max_{j \in V} \|\nabla\theta_j(\widehat{\mathbf{u}}_j^{(l)})\|\}$. Se $\widehat{\mathbf{u}}^{(l)}$ não é ótimo para $P(\varepsilon)$ na iteração l, então qualquer agente $m(l) \in J(l)$ realiza uma busca linear com retrocesso, iniciando em $\widehat{\mathbf{u}}_{m(l)}^{(l)}$, para gerar a próxima solução $\widehat{\mathbf{u}}_{m(l)}^{(l+1)}$.*

Teorema 1. *Sob as hipóteses 1, 2 e 3, o algoritmo de gradiente descendente distribuído gera uma sequência de iterandos $\widehat{\mathbf{u}}^{(l)}$ que converge à solução*

Algoritmo 2: Algoritmo de Gradiente Descendente Distribuído para Solução de $P(\varepsilon)$

entrada: uma solução estritamente factível $\widehat{\mathbf{u}}$, parâmetro de barreira ε, parâmetros para busca linear com retrocesso $\alpha \in (0, 1/2)$ e $\beta \in (0,1)$, e tolerância τ

inicialização: $l := 0$ e $\widehat{\mathbf{u}}^{(0)} := \widehat{\mathbf{u}}$

while $\|\nabla\theta(\widehat{\mathbf{u}}^{(l)})\| > \tau$ **do**
  seja $S(l)$ um subconjunto de agentes não vizinhos tal que $m(k) \in S(l)$
  for *cada* $m \in S(l)$ *em paralelo* **do**
    obtenha $\widehat{\mathbf{w}}_m^{(l)}$ dos vizinhos $N(m)$
    $s_m^{(l)} := 1$
    while $\theta_m(\widehat{\mathbf{u}}_m^{(l)} - s_m^{(l)}\nabla\theta_m(\widehat{\mathbf{u}}_m^{(l)})) > \theta_m(\widehat{\mathbf{u}}_m^{(l)}) - \alpha s_m^{(l)}\|\nabla\theta_m(\widehat{\mathbf{u}}_m^{(l)})\|^2$ **do**
      $s_m^{(l)} := \beta s_m^{(l)}$
    $\widehat{\mathbf{u}}_m^{(l+1)} := \widehat{\mathbf{u}}_m^{(l)} - s_m^{(l)}\nabla\theta_m(\widehat{\mathbf{u}}_m^{(l)})$
  for *cada* $m \in \mathscr{M} - S(l)$ *em paralelo* **do**
    $\widehat{\mathbf{u}}_m^{(l+1)} := \widehat{\mathbf{u}}_m^{(l)}$
  $l := l + 1$
  saída: $\widehat{\mathbf{u}}^{(l)}$

ótima $\widehat{\mathbf{u}}^{\star}(\varepsilon)$ *do problema de centralização* $P(\varepsilon)$ *(CAMPONOGARA; SCHERER, 2011).*

O algoritmo de gradiente descendente distribuído (AGDD) pode ser modificado para utilizar a direção de Newton $-\nabla^2\theta_m(\widehat{\mathbf{u}}_m^{(l)})^{-1}\nabla\theta_m(\widehat{\mathbf{u}}_m^{(l)})$ em vez da direção de descenso $-\nabla\theta_m(\widehat{\mathbf{u}}_m^{(l)})$, executar várias buscas lineares com retrocesso, e ainda utilizar algoritmos disponíveis para resolver $P_m(\widehat{\mathbf{w}}_m)$ até sua otimalidade. O impacto dessas estratégias no desempenho de AGDD foi investigado em (CAMPONOGARA; SCHERER, 2011).

3.4.1 Modelo *Store and Forward* como Uma Rede Dinâmica Linear

O modelo *stored and forward* para uma rede de tráfego veicular proposto em (DIAKAKI et al., 2002) pode ser representado como uma rede

dinâmica linear de uma forma direta. Os subsistemas irão representar as interseções de tráfego com o conjunto de vértices V correspondendo às interseções. Dado o conjunto de pistas Z, juntamente com o conjunto $Z_{\mathrm{i}}(m)$ de pistas de entrada e o conjunto $Z_{\mathrm{o}}(m)$ de pistas de saída da interseção $m \in V$, o vetor de estados do subsistema m é dado por $\mathbf{x}_m = (x_z : z \in Z_{\mathrm{i}}(m))$ sendo x_z o número de veículos na pista z.

Os sinais de controle do subsistema m são modelados pelo vetor $\mathbf{u}_m = (G_z : z \in Z_{\mathrm{i}}(m)) \cup (g_{s,m} : s \in E_m)$ onde g_z é o tempo de verde total atribuído à pista z, F_m é o conjunto de estágios da interseção m, e $g_{s,m}$ é o tempo de verde alocado ao estágio s desta interseção.

Um subsistema m influencia o estado do subsistema j se existe uma pista z tal que $z \in Z_{\mathrm{o}}(m) \cap z \in Z_{\mathrm{i}}(j)$, o que significa que $(m, j) \in E$. Portanto, $M(m) = \{j : Z_{\mathrm{o}}(j) \cap Z_{\mathrm{i}}(m) \neq \emptyset\}$ e $\widehat{N}(m) = \{j : Z_{\mathrm{o}}(m) \cap Z_{\mathrm{i}}(j) \neq \emptyset\}$. Seguindo a modelagem de fluxo de tráfego proposta em (DIAKAKI et al., 2002), a qual define o fluxo de saída em função do fluxo de entrada e demanda nula para as pistas, as matrizes $B_{m,j}$ são obtidas a partir das taxas de conversão, o fluxo de saturação e os estágios por meio do modelo *store and forward* para todo $m \in V$, $j \in M(m)$. A matriz de estados $A_{m,m}$ é a identidade.

A restrição a seguir elimina a única restrição de igualdade e garante que a soma dos tempos de verde e tempo perdido equivale ao tempo de ciclo:

$$\sum_{s \in E_m} g_{s,m} + L_m = C_m, m \in V \tag{3.23a}$$

O modelo desagregado para o tempos de verde das pistas e dos estágios (ABOUDOLAS et al., 2007b, 2009) é reproduzido aqui por meio da introdução de uma restrição limitando o tempos de verde das pistas conforme os tempos dos estágios:

$$0 \leq G_z \leq \sum_{s \in e_z} g_{s,m}, m \in \mathscr{M}, z \in Z_{\mathrm{i}}(m) \tag{3.23b}$$

onde $e_z \subseteq E_m$ é o conjunto de estágios durante os quais a pista z tem direito de passagem.

O tempo de verde mínimo para os estágios e a capacidade das pistas são expressos, respectivamente, por:

$$g_{s,m} \geq g_{s,m}^{\min}, m \in V, m \in F_m \tag{3.23c}$$

$$0 \leq x_z \leq x_z^{\max}, z \in Z \tag{3.23d}$$

Claramente, as restrições (3.23a)-(3.23d) são expressas por meio das desigualdades (3.16c)-(3.16d) utilizando-se matrizes $W_{m,i}$, $Z_{m,i}$ e vetores $\mathbf{y}_m^{\max}$,

$\mathbf{y}_m^{\min}$, $\mathbf{u}_m^{\min}$ e $\mathbf{u}_m^{\max}$ apropriados.

A matriz Q_m de custo para os estados da interseção m é uma matriz diagonal com elementos dados por $1/(x_z^{\max})^2$ para $z \in Z_{\mathrm{i}}(m)$, pela proposta de (ABOUDOLAS et al., 2007b, 2009). Na proposta de (ABOUDOLAS et al., 2007b, 2009), maior peso se dá para as vias com menor capacidade de armazenar carros (vias menores). A ação de controle segundo a equação (3.13a) possui termos da forma $(\mathbf{u}_m(t) - \mathbf{u}_m^{\mathrm{nom}})' R_m (\mathbf{u}_m(t) - \mathbf{u}_m^{\mathrm{nom}})$ onde $\mathbf{u}_m^{\mathrm{nom}}$ é o tempo de verde nominal, $R_m = rI$ e r é um parâmetro de sintonia. Pode-se utilizar a variação no sinal de controle $\Delta\mathbf{u}_m(t) = \mathbf{u}_m(t) - \mathbf{u}_m^{\mathrm{nom}}$ em vez do sinal de controle $\mathbf{u}_m(t)$ no modelo de rede dinâmica linear apresentado. Entretanto, para simplificar a apresentação e porque aplicações MPC ao controle veicular definem $r = 0$, a matriz R_m é definida como a matriz nula e as restrições a seguir são impostas para limitar a variação no tempo de verde:

$$-\Delta g_{s,m}^{\max} \leq \mathbf{g}_{s,m} - \mathbf{g}_{s,m}^{\mathrm{nom}} \leq \Delta g_{s,m}^{\max},\ m \in V,\ s \in F_m \tag{3.24}$$

Em (SOUZA et al., 2010) uma RDL é instanciada sobre uma pequena rede de tráfego, onde pode-se ver em detalhes como obter as matrizes para se adequar a formulação apresentada.

3.5 SUMÁRIO

Este capítulo iniciou com uma breve revisão da estratégia de controle preditivo, contemplando o modelo de predição, a função objetivo, as restrições e o algoritmo de otimização. Redes dinâmicas lineares foram definidas como grafos direcionados cujos nós representam subsistemas dinâmicos e cujos arcos indicam a influência direta entre os subsistemas. Tais redes representam naturalmente a dinâmica de fluxo veicular baseda no modelo *store and forward* e utilizada pela estratégia de controle TUC. Uma abordagem de controle preditivo distribuído foi desenvolvida para redes dinâmicas, contemplando um algoritmo de otimização distribuído que viabiliza a implementação distribuída da estratégia de controle preditivo com uma rede de agentes.

4 PLATAFORMA COMPUTACIONAL

Neste capítulo será apresentado o pacote de software desenvolvido no presente trabalho para a realização de experimentos dos algoritmos apresentados no capítulo 3. Inicia-se com uma breve contextualização à implementação prática de Controle Digital Discreto, na sequência o desenvolvimento realizado é justificado e apresentado. O capítulo finaliza com um detalhamento da implementação.

4.1 CONTEXTUALIZAÇÃO EM CONTROLE DISCRETO

Apresenta-se uma breve introdução de controle discreto com relação a sua aplicação prática, com o objetivo de elucidar exatamente o papel da plataforma desenvolvida.

Qualquer sistema de controle pode ser representado genericamente pela Figura 7. Um sistema de medição alimenta o controlador com dados sobre o estado da planta, o sistema a ser controlado. O controlador decide a ação que deve ser tomada, para finalmente o atuador a executar. Estes passos são realizados ciclicamente. Nos restringindo ao controle discreto, a ação é calculada através de um algoritmo embutido no controlador.

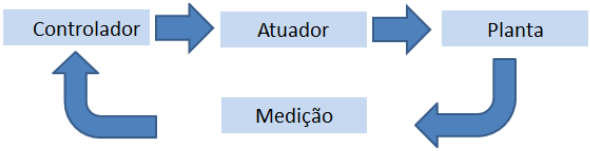

Figura 7: Diagrama de um sistema de controle

Fisicamente um controlador é um dispositivo microprocessado, frequentemente um computador, e o algoritmo de controle é um software contido nele. O controlador deve ter uma forma de receber dados da planta, como ler um valor analógico de um sensor, e uma forma de enviar a próxima ação para o atuador, como por exemplo o envio dos tempos aos controladores semafóricos. Uma visão esquemática pode ser vista na Figura 8.

Ao se referir ao algoritmo de controle, este é um módulo contido em um software que possui outros dois módulos. Implementar MPC se refere a implementar o algoritmo de controle utilizando tal técnica e implementar significa ter um software que recebe a medição da planta, gera o problema de

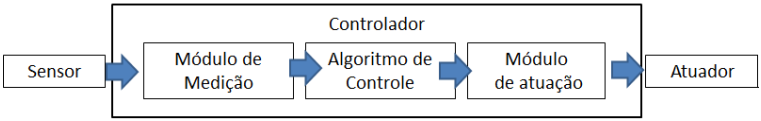

Figura 8: Componentes estruturais de um controlador.

otimização e o resolve para então o controlador enviar o resultado ao atuador.

Um controlador digital discreto pode estar em diversos de microprocessadores, desde microntroladores de pouca capacidade computacional até em um servidor com grande capacidade computacional. Neste texto nos referimos à ambientes com capacidade computacional compatível com aplicação de MPC.

A plataforma desenvolvida no presente de trabalho é uma ferramenta para ser utilizada por módulos de controle e constituir o algoritmo de controle do controlador.

4.2 JUSTIFICATIVA

A implementação de um MPC distribuído para um dado problema é uma tarefa dispendiosa. A **realização** de uma aplicação consiste em três atividades fundamentais:

- **modelagem do problema**: construir um modelo matemático para o sistema a ser controlado, esta atividade está ligada ao seu domínio de aplicação;
- **implementação do algoritmo**: obter a estrutura de dados (de acordo com a Seção 3.3 e o código de cada agente, bem como o ajuste de parâmetros do algoritmo; e
- **sintonia de controle**: realizar pequenas alterações nas matrizes de ponderação ou mudança no horizonte de predição.

Estas atividades fecham um ciclo iterativo que pode ser visto na Figura 9.

No contexto de Redes Dinâmicas Lineares, a modelagem do sistema é feita de acordo com o conjunto de equações (3.13a-3.13e) da seção 3.3. A implementação do algoritmo consiste em, a partir das entradas do problema ($A_{i,j}$, $B_{i,j}$, $W_{i,j}$, $Z_{i,j}$, $\mathbf{u}_m^{\max}$, $\mathbf{u}_m^{\min}$, $\mathbf{y}_m^{\max}$, $\mathbf{y}_m^{\min}$, Q_i e R_i), gerar o código que a

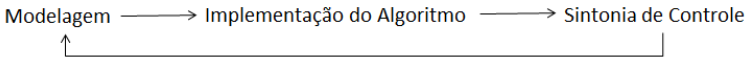

Figura 9: Etapas de implementação.

cada perído de amostragem consiga instanciar o problema P (3.13a-3.13e) e resolver, sendo que no caso distribuído deve gerar cada $P_i(\widehat{\mathbf{w}}_i)$ (equações 3.16a-3.16d).

A geração da estrutura de dados para $P_m(\widehat{\mathbf{w}}_m)$ sem o auxílio de uma ferramenta para tal é uma tarefa árdua, por dois motivos principais:

- o alto número de entradas nas matrizes, até mesmo para problemas pequenos. Por exemplo uma aplicação ao controle de tráfego com uma rede com 3 junções (agentes), 2 pistas (estados) em cada junção e 2 estágios com horizonte de predição 2, o vetor $\widehat{\mathbf{u}}_m$ contém 24 entradas e a matriz H_m possui 576 entradas, sem contar as matrizes de restrições. Ainda que vários blocos se repetem ou são nulos, o usuário montar as matrizes é uma tarefa susceptível a erros;

- a frequente necessidade de alteração no modelo em função da sintonia de controle, ocasionando alteração em parâmetros do modelo e uma consequente mudança na estrutura de dados. Em casos como mudança no período de amostragem, toda a estrutura de dados deve ser refeita.

Em vista destes problemas, o desenvolvimento de uma plataforma de experimentação foi proposto, considerando três objetivos:

1. realizar a tarefa de montar a estrutura de dados do problema a partir da modelagem, isto é, a partir das dinâmicas e restrições de cada agente e seus acoplamentos de primeira ordem, gerar toda estrutura de dados subsequente;

2. gerar o problema distribuído e o centralizado recíproco para fins de comparação de resultados;

3. executar o algoritmo centralizado e distribuído.

Na seção a seguir a plataforma é apresentada em maiores detalhes.

4.3 A PLATAFORMA

A plataforma é uma biblioteca de software desenvolvida para auxiliar a implementação e experimentação de MPC distribuído, facilitando sua implementação em todas as etapas, por receber de entrada uma modelagem em

alto nível de abstração, natural para o engenheiro realizando a tarefa e por conseguir executar todos os passos intermediários até a solução final.

Possui implementação do Algoritmo de Gradiente Descendente Distribuído (Algoritmo 2 apresentado na Seção 3.3.2), com agentes executando em série e usando compartilhamento de memória como meio de comunicação. Para uso dos *solvers*, a plataforma também possui funções para construção do modelo bem como rotinas para, a partir do modelo gerar a estrutura de dados para chamada dos algoritmos. A seguir um exemplo de uso é apresentado:

```
modelo = modelagem.modeloRDL()
modelo.adicionaAcoplamento(agente_de_entrada = 1,
    agente_de_saida = 1, A = [0.3], B = [0.2])

estrutura_de_dados = geracao.
    geraProblemaDistribuido(modelo)

solver = solvers.BarreiraDistribuido(
    estrutura_de_dados)
resultado = solver.executa()
```

Nas duas primeiras linhas, o modelo é instanciando e então o modelo começa ser construído. Neste caso, é adicionado um acoplamento do agente 1 para si próprio (dinâmica própria), passando o valor de A_1 e $B_{1,1}$. Outras chamadas devem ser feitas de maneira similiar para completar o modelo.

Na terceira linha, é feita a chamada para gerar o problema distribuído. Com este mesmo modelo, seria possível gerar o problema centralizado também.

No último bloco, de posse da estrutura de dados adequada, o solver é instanciado e executado.

A plataforma é sub-divididas em três módulos, como ilustra a Figura 10, sendo suas responsabilidades:

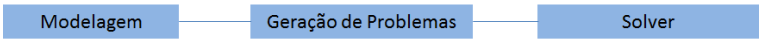

Figura 10: Módulos da plataforma.

- **Módulo de Modelagem**: contém classes para representar uma Rede Dinâmica Linear em uma estrutura de dados apropriada, estas disponibilizam métodos na semântica de Redes Dinâmicas Lineares, para construção do modelo;

- **Módulo de Geração de Problemas**: recebendo como entranda modelos, possui classes para gerar tanto o problema distribuído, quanto o recíproco centralizado. Gerar problema neste contexto se refere a obter todas as estruturas de dados que serão necessárias para chamada do solver;

- **Solver**: Contém algoritmos para Barreira Logarítmica Distribuído e Centralizado especializados para o problema definido nas classes de modelagem. Retorna solução ótima caso existe ou informa infactibilidade.

Todos os módulos podem ser usados independentemente.

O desenvolvimento foi realizado em linguagem de programação Python, escolhida por possuir uma série de bibliotecas disponíveis para computação científica (SCIPY, 2010), algebra linear (NUMPY, 2010) e otimização. Além de ser uma uma linguagem de uso geral, contendo uma série de ferramentas disponíveis para aplicação prática (banco de dados, interface gráfica, acesso a recursos do sistema operacional, etc).

4.4 IMPLEMENTAÇÃO

Nesta seção se apresenta detalhes de implementação da plataforma desenvolvida, primeiramente as ferramentas utilizadas, na sequência detalhes relacionados à estrutura do software.

FERRAMENTAS UTILIZADAS

O desenvolvimento da plataforma foi realizado em linguagem de programação Python, sendo as seguintes ferramentas utilizadas:

- **Eclipse**: ferramenta para desenvolvimento de software em qualquer linguagem de programação, com possibilidade de integrar outros *plug-ins* (ECLIPSE, 2010);

- **PyDev**: *plugin* para programação da linguagem em Python no Eclipse (PYDEV, 2010).

- **CVXOPT**: biblioteca para otimização convexa em linguagem Python (CVXOPT, 2010), esta por sua vez utiliza outras ferramentas, sendo relevantes NumPy (NUMPY, 2010) para matrizes e Lapack (LAPACK, 2010) para álgebra linear.

ESTRUTURA

Cada macro funcionalidade foi implementada por um módulo, sendo cada um codificado independentemente de outros. O módulo de modelagem contém uma classe para a construção completa de um modelo; o módulo de geração gera a estrutura de dados para chamada dos solvers; e os solvers contém as implementações especializadas dos algoritmos apresentados na Seção 3.3.2.

O diagrama de classes é apresentado nas Figuras 11, 12 e 13 e a seguir cada módulo e classe são apresentados.

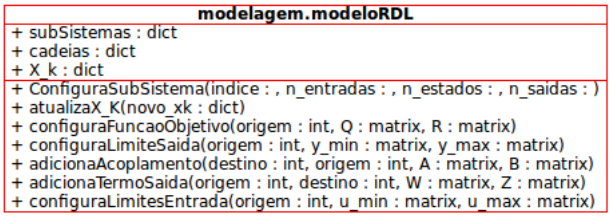

Figura 11: Módulo de Modelagem.

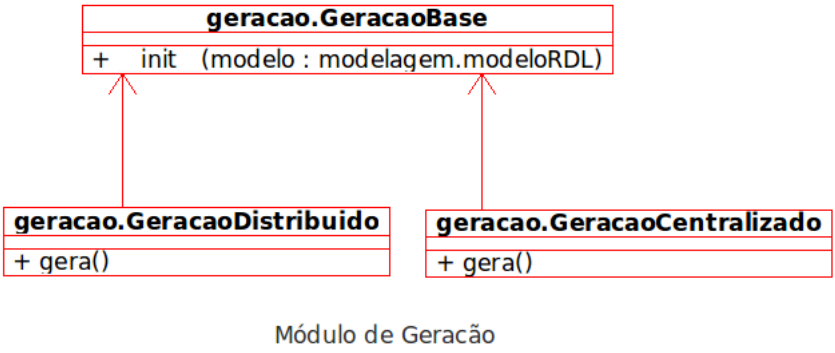

Figura 12: Módulo de Geração.

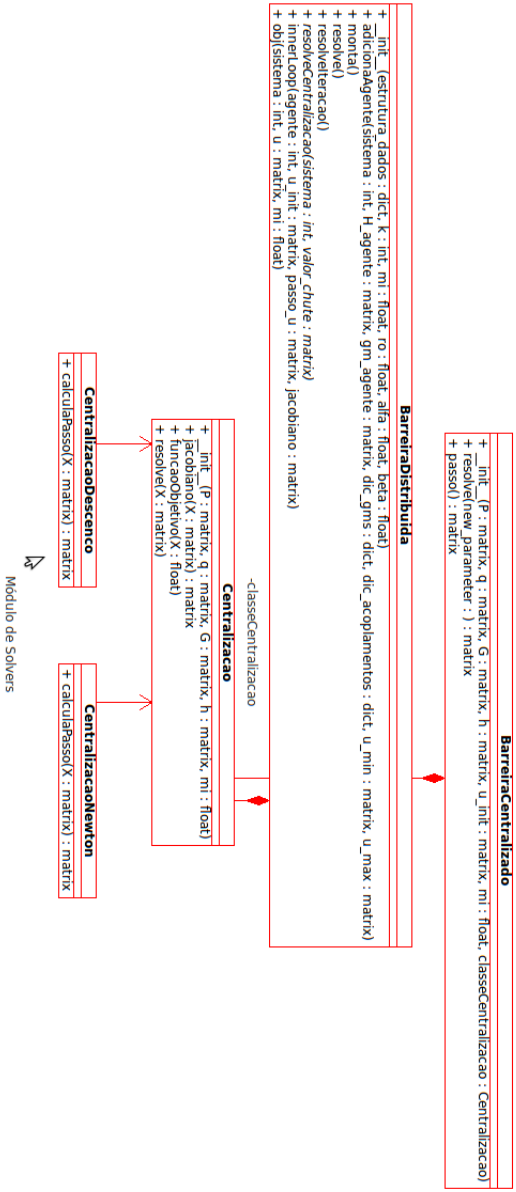

Figura 13: Módulo de *Solvers*.

MÓDULO DE MODELAGEM

O módulo de modelagem contém atualmente apenas uma classe. Apenas a modelagem apresentada em (SOUZA et al., 2010) foi implementada. Entretanto futuras implementações residirão neste módulo, como a abordagem apresentada em (CAMPONOGARA; OLIVEIRA, 2009). A estratégia do módulo é cada classe manter uma estrutura de dados interna para representar uma rede dinâmica linear.

A classe **ModelagemRDL** contém 4 métodos básicos e a estrutura de dados interna é armazenada em uma série de dicionários (par chave valor) encadeados.

Fundamentalmente a estrutura de dados interna possui dois dicionários:

- **subSistemas**: possui uma chave para cada agente, seu respectivo valor é um dicionário contendo uma série de chaves (entradas, saídas, estados, restrições, Q, R, $\mathbf{y}_m^{max}$, $\mathbf{y}_m^{min}$, $\mathbf{u}_m^{max}$,$\mathbf{u}_m^{min}$.
- **dinâmica**: possui uma série de chaves (destino, origem) para um dicionario que contém A e B de cada acoplamento.

A utilização destes dicionários pode ser exemplificado com os seguintes trechos de código, primeiramente o dicionário de subSistemas:

```
def adicionaTermoSaida(self, sistema, origem, W, Z)
    :

  self.subSistemas[sistema]['saida'][origem] = {'W'
      :W, 'Z':Z}
```

A primeira linha define o método da classe e na linha abaixo seu código. Para o caso da cadeia dinâmica o tratamento é análogo:

```
def adicionaAcoplamento(self, destino, origem,
        A_dinamica, B_dinamica):

    self.dinamica[destino][origem] = {}
    self.dinamica[destino][origem]['A'] =
        A_dinamica
    self.dinamica[destino][origem]['B'] =
        B_dinamica
```

Seguindo a mesma metodologia toda a estrutura de dados é preenchida com chamada de outros métodos, como por exemplo **configuraFuncaoObjetivo** que irá alterar

```
subSistemas[sistema]['Q']
subSistemas[sistemas]['R']
```

Como se pode observar, a responsabilidade dessa classe é apenas armanezar a estrutura de dados. Ela não implementa nada relacionado ao que foi discutido no Capítulo 3.

MÓDULO DE GERAÇÃO DE PROBLEMAS

Este módulo é o responsável por, a partir de modelos (como ModeloRDL), gerar as estruturas de dados necessárias à chamada do solver. Sua função dentro da arquitetura é de extrema importância por fazer a ponte entre dois domínios de conhecimento diferentes:

1. modelagem, onde se lida com elementos relacionados ao domínio de aplicação, como tráfego, tais como equações dinâmicas, restrições, acoplamentos e etc; e

2. solvers, que são genéricos para qualquer aplicação, com parâmetros relacionados aos algoritmos.

Em outras palavras é classe que implementa o conhecimento da seção da 3.3.

A implementação foi seguindo o que foi descrito nas Seções 3.3.1 e 3.3.2, com os procedimentos descritos em (CAMPONOGARA; OLIVEIRA, 2009; CAMPONOGARA; SCHERER, 2011). Em linhas gerais para cada matriz apresentada (como $H_{m,j,l}$, *etc*) foi mapeado por um método na classe de geração e sua implementação seguiu o que foi descrito na bibliografia.

Como um exemplo:

$$H_{m,i,j} = \widehat{Z}_{m,i}\widehat{Q}_m\widehat{Z}_{m,j} \; \forall i,j \in \widehat{M(m)}, \; i \neq m \; ou \; j \neq m$$

$$H_{m,m,m} = \widehat{Z}_{m,i}\widehat{Q}_m\widehat{Z}_{m,m} + \widehat{R}_m \; \forall \; i,j \; \in \widehat{M(m)}, i \neq m \; ou \; j \neq m$$

foi mapeado para o seguinte método:

```
def calculaH(self, m, i, j):

    H = self.calculaZchapeu(m, i).trans()*self.
        calculaQchapeu(m)*self.calculaZchapeu(m, j)

    if i==m and j == m:
            H = H + self.calculaRchapeu(m)
```

```
return H
```

Percebe-se a clara relação entre a implementação e a definição. Outro aspecto a ser notado é que pelo trecho apresentado, uma segunda chamada com os mesmos parâmetros iria ocorrer na execução do código novamente para algo que já foi calculado. Primeiramente, não foi o objetivo do trabalho refinar o processo de geração, por em geral não demandar tempo considerável; e caso isso seja problema, Python (e grande parte das linguagens orientadas a objeto) oferece ferramentas de *memoization* (CORMEN, 2002) em que poderiam solucionar o problema com modificações mínimas.

O núcleo do módulo está na classe **GeracaoBase** e esta possui duas classes herdadas que são as que devem ser instanciadas para geração do problemas:

- **GeracaoCentralizado**: Seu método **gera()** retorna a estrutura de dados para chamada do solver centralizado; e
- **GeracaoDistribuido**: o qual o método **gera()** retorna a estrutura de dados para chamada do solver distribuído.

SOLVERS

O módulo de solvers contêm as implementações do algoritmo distribuído apresentado na Seção 3.4, e seu análogo centralizado, método de barreira logarítmica centralizado. Ambos devem utilizar um algoritmo para resolver o problema de centralização, foram implementados tanto descenso quanto Newton. A seguir linhas gerais de implementação de cada classe:

A classe **Centralização** realiza a implementação como descrito em (CAMPONOGARA; OLIVEIRA, 2009; CAMPONOGARA; SCHERER, 2011), possuindo duas classes herdadas que sobrecarregam a forma de calcular o passo (mínimo descenso ou passo de Newton).

A classe **BarreiraCentralizado** tem a implementação do algoritmo de barreira. O laço principal pode ser visto no trecho abaixo:

```
def resolve(self):
  while self.mi > TOLERANCIA/len(self.P.size[0]):
      self.U, inner_iters = self.passo()
      self.mi = self.ro*self.mi
```

Sendo o método passo definido como:

```
def passo(self):
```

```
self.centralizacao.mi = self.mi
res = self.centralizacao.resolve(self.U)
```

Onde centralização é um objeto da classe CentralizacaoNewton ou CentralizacaoDescenso.

BARREIRA LOGARÍTMICA DISTRIBUÍDA

A classe **BarreiraDistribuida** implementa o Algoritmo 2, mais complexa do que as outras apresentadas, especialmente pela necessidade de comunicação e sincronização entre os agentes, ainda que nesta implementação utilizou-se memória compartilhada para simular a comunicação e o código de cada agente é executado serialmente.

A estrutura de dados de entrada contêm informações de acoplamentos que é passada na inicialização:

```
def __init__(self, ..., estrutura_de_dados)
...
  for indice in estrutura_dados['agentes']:
      dados_agente = estrutura_dados['agentes'][
          indice]

      self.adicionaAgente(indice, dados_agente['H'
          ],
                  dados_agente['gim'], dados_agente[
                      'dic_gms'],
                  dados_agente['dic_acoplamento'],
                  dados_agente['u_min'],
                      dados_agente['u_max'])
```

Já adicionaAgente coloca as informações em uma estrutura adequada ao uso no algoritmo. O laço principal executa as seguintes tarefas:

1. gerar cada $P_m(\widehat{\mathbf{w}}_m)$ a partir dos dados atualizados da iteração anterior, isto é feito pelo método monta() que itera sobre todos os agentes e atualiza a estrutura de dados de cada agente;

2. buscar o agente de menor norma, realizado no método resolveIteracao(); e

3. resolver o problema de centralização para o agente de maior norma, implementado no método resolveCentralizacao().

Abaixo segue o trecho de código correspondente ao descrito acima:

```
def resolve(self):
    trajetoria = []
    while self.mi > TOLERANCIA:
        self.monta()
        self.resultados = copy.deepcopy(self.U)

        self.resolveIteracao()
        self.mi = self.ro*self.mi
        trajetoria.append(self.U)

    return self.U, self.iteracoes
```

4.5 SUMÁRIO

Este capítulo iniciou com a uma apresentação da plataforma desenvolvida. Nela é possível implementar Controle Preditivo Distribuído pelo método de Barreira Distribuído, bem como seu recíproco centralizado. A plataforma foi codificada em linguagem Python e possui três módulos: modelagem, geração e solvers.

5 RESULTADOS EXPERIMENTAIS

Neste capítulo são apresentados resultados de experimentos utilizando a plataforma apresentada no capítulo 4. A primeira seção contém uma análise genérica sobre o desempenho do algoritmo de barreira distribuída em diferentes cenários. A segunda seção apresenta um experimento de simulação aplicado em controle de tráfego urbano utilizando também a plataforma desenvolvida, objetivo principal do trabalho.

5.1 EXPERIMENTOS NUMÉRICOS EM REDES ARBITRÁRIAS

A primeira série de experimentos foi realizada com o intuito de validar a plataforma. O primeiro objetivo foi reproduzir os resultados apresentados em (CAMPONOGARA; OLIVEIRA, 2009). No entanto, uma vez com a plataforma à disposição foi possível automatizar os experimentos e também avaliar a influência da topologia da rede e de parâmetros do algoritmo no desempenho, medido em tempo total de execução e número de iterações.

A influência dos seguintes fatores foram observadas:

- **Horizonte de Predição:** qual dos algoritmos, centralizado ou distribuído, sofre maior influência no seu desempenho com o aumento do horizonte de predição;
- **Taxa de Decrescimento:** qual taxa μ cada um dos algoritmos apresenta melhor desempenho;
- **Épsilon Inicial:** como varia o desempenho para diferentes $\varepsilon^{(0)}$;
- **Topologia da rede:** espera-se perda de desempenho em redes mais acopladas, por isso buscou-se observar em que medida.

Foram construídas duas topologias. A mais acoplada pode ser vista na Figura 14 e a menos acoplada na Figura 15. Para cada topologia foram gerados diferentes modelos, sendo que em todos os modelos os agentes possuem um estado, uma entrada e uma saída. Os elementos das matrizes $A_{m,i}$, $B_{m,i}$, $W_{m,i}$, $Z_{m,i}$, Q_m e R_m são aleatórios, com distribuição uniforme entre -0,5 e 0,5, exceto Q_m e R_m que podiam variar entre 0 e 1. Os valores máximos de saída ($\mathbf{y_m}^{max}$ e $\mathbf{y_m}^{min}$) em módulo eram fixos em 2 e em entrada ($\mathbf{u_m}^{max}$ e $\mathbf{u_m}^{min}$) em 5. A condição inicial, $\mathbf{x}_k$ também aleatória com distribuição normal entre -0,5 e 0,5.

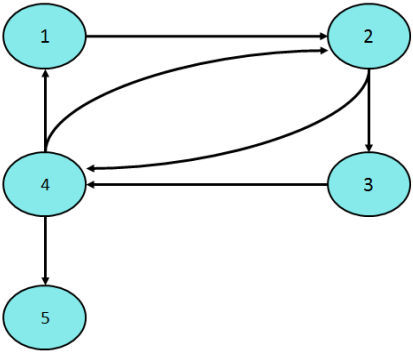

Figura 14: Rede com maior acoplamento

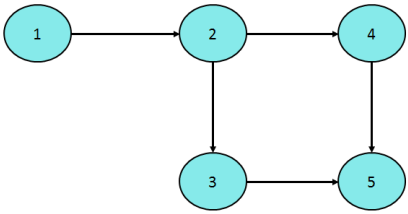

Figura 15: Rede com menor acoplamento

Foram geradas 6 redes para cada topologia. Testou-se três diferentes horizontes de predição (1, 2 e 3), taxas de decrescimento μ (0,7, 0,1 e 0,03) e valores para $\varepsilon^{(0)}$ (0,01, 0,1 e 1). Todas as combinações possíveis foram executadas. Isto significa que para cada rede e topologia foram realizados $3^3 = 27$ experimentos.

Os valores de α e β foram fixados em 0,2 e 0,4 respectivamente. O algoritmo de programação linear presente no CVXOPT (CVXOPT, 2010) foi utilizado para gerar a solução inicial dos problemas. A tolerância de solução τ na centralização (algoritmo 1) foi de 10^{-9} e a tolerância em cada centralização de 10^{-10}.

Os experimentos foram realizados com processador Intel(R) Core(TM) i7-2630QM CPU de 2.00GHz com 2GB de RAM em Sistema Operacional Linux Ubuntu 10.10.

MÉTODO DE CENTRALIZAÇÃO

A plataforma possui os Método de Descenso e o Método de Newton para resolver cada problema de centralização $P(\varepsilon^{(l)})$. Na Tabela 1 estão o tempo total de CPU e o total de iterações realizadas por cada método sobre todos os casos possíveis (cada topologia, cada taxa de decrescimento, *etc*).

Tabela 1: Resultados Computacionais relativo ao problema de centralização

Caso	Tempo Total de CPU (s)	Total de Iterações
Descenso Centralizado	11,42	44.137
Newton Centralizado	2,81	7.050
Descenso Distribuído	123,09	281.978
Newton Distribuído	108,14	102.088

Os resultados apresentados vão ao encontro da literatura anterior (CAMPONOGARA; OLIVEIRA, 2009): os algoritmos distribuídos despendem mais tempo para alcançar a solução ótima. O método de Newton tanto no distribuído quanto no centralizado apresentou melhores resultados, embora a diferença seja proporcionalmente menor no distribuído.

HORIZONTES DE PREDIÇÃO

Para este experimento, fixou-se ε em 1 e μ em 0,1 sobre a rede mais acoplada e variou-se os horizontes de predição em 1, 2 e 3.

Os resultados são apresentados na Tabela 2. As colunas t-1, t-2 e t-3

referem-se ao tempo de CPU despendido, em segundos, para horizontes de comprimento 1, 2 e 3 respectivamente. Da mesma forma I-1, I-2 e I-3 que referem-se ao total de iterações. Na tabela são apresentados os totais para cada método, uma vez que os experimentos foram feitos sobre 6 instâncias da rede mais acoplada (6 redes dinâmicas de mesma topologia), para obter-se a média basta dividir o total sobre o número de redes diferentes (no caso 6).

Tabela 2: Horizontes de predição, com tempo em segundos nas colunas t-*i* e total de iterações nas colunas I-*i*, onde *i* é o horizonte de predição.

Algoritmo	t-1	I-1	t-2	I-2	t-3	I-3
Descenso Cent.	0,06288	591	0,14713	656	0,25864	703
Newton Cent.	0,01797	138	0,03967	142	0,08144	149
Descenso Dist.	0,94987	3.476	2,76037	6.721	5,23893	11.957
Newton Distr.	0,91142	1.902	2,52035	3.116	4,19700	4.516

A Tabela 3 apresenta o percentual de aumento de tempo e iterações de um horizonte para outro. t-1→t2 na tabela apresenta o aumento de tempo de execução entre horizonte 1 e 2. Analogamente para I-1→I3 que apresenta o aumento percentual do número de iterações entre horizontes 1 e 3.

Tabela 3: Aumento percentual no tempo de solução no aumento do horizonte. As colunas t-i → t-j apresentam aumento no tempo de horizonte i para horizonte j. As colunas I-i →I-j apresentam o aumento percentual do total de iterações de horizonte i para horizonte j.

Algoritmo	t-1→t2	I-1→I2	t-2→t3	I-2→I-3	t-1→t-3	I-1→I-3
Descenso Cent.	57,26%	9,90%	43,11%	6,68%	75,68%	15,93%
Newton Cent.	54,70%	2,86%	51,28%	4,69%	77,93%	7,38%
Descenso Dist.	65,58%	48,28%	47,31%	43,79%	81,87%	70,93%
Newton Distr.	63,84%	38,96%	39,94%	31,00%	78,28%	57,88%

Os resultados mostram que o aumento do horizonte de predição aumenta também o tempo total despendido para alcançar a solução. Nesta tabela não se observa comportamento diferente na ordem de crescimento em nenhum dos métodos. É visível na Tabela 3 que o número de iterações varia pouco nos métodos centralizados, diferentemente dos métodos distribuídos que variam na mesma ordem de grandeza dos tempos despendidos.

TAXA DE DECRESCIMENTO

Sobre todos os cenários possíveis (topologia, horizonte, *epsilon* e algoritmo) colheu-se o tempo total de CPU e o total de iterações para cada taxa de decrescimento μ. A Tabela 4 apresenta os resultados.

Tabela 4: Tempos computacionais e número de iterações para cada taxa de decrescimento μ.

Caso - μ	Tempo Total de CPU (s)	Total de Iterações
Descenso Centralizado - 0,03	1,72	6.551
Descenso Centralizado - 0,1	2,24	8.696
Descenso Centralizado - 0,7	7,45	28.890
Newton Centralizado - 0,03	0,40	1.135
Newton Centralizado - 0,1	0,57	1.375
Newton Centralizado - 0,7	1,83	4.540
Descenso Distribuído - 0,03	26,01	68.361
Descenso Distribuído - 0,1	29,15	74.932
Descenso Distribuído - 0,7	67,92	138.685
Newton Distribuído - 0,03	22,46	24.771
Newton Distribuído - 0,1	25,36	26.939
Newton Distribuído - 0,7	60,30	50.378

De acordo com a Tabela 4, a taxa de decrescimento de 0,03 apresentou melhor resultado, entretanto muito próximo ao de 0,1. A taxa mais lenta (0,7) apresentou tempos muito maiores que as taxas de decrescimento mais rápidas (0,03 e 0,1).

VALOR INICIAL DE PRECISÃO DE APROXIMAÇÃO ε

Sobre todos os cenários possíveis (topologia, horizonte, μ e algoritmo) colheu-se o tempo total e o total de iterações para cada *epsilon* inicial ε. A Tabela 5 apresenta os resultados.

Para os casos estudados, valores iniciais menores de ε resultaram em valores menores para alcançar a solução. Outros casos devem ser estudados para avaliar esse resultado. Possivelmente estes dados sugerem que as restrições são fracas para os experimentos, uma vez que se não existissem restrições o melhor valor de ε inicial seria zero.

Tabela 5: Resultados computacionais para cada caso variando a Precisão de Aproximação ε inicial.

Caso	Tempo Total de CPU (s)	Total de Iterações
Descenso Centralizado - 0,01	2,78	10.099
Descenso Centralizado - 0,1	3,75	14.490
Descenso Centralizado - 1	4,88	10.548
Newton Centralizado - 0,01	0,61	1.104
Newton Centralizado - 0,1	0,85	1.936
Newton Centralizado - 1	1,34	4.010
Descenso Distribuído - 0,01	25,29	59.244
Descenso Distribuído - 0,1	37,86	86.155
Descenso Distribuído - 1	59,93	136.579
Newton Distribuído - 0,01	21,45	14.263
Newton Distribuído - 0,1	33,48	25.149
Newton Distribuído - 1	53,20	62.676

TOPOLOGIA DA REDE

Como citado, duas redes dinâmicas lineares foram criadas, uma acoplada, ilustrada na Figura 14, e outra menos acoplada, ilustrada na Figura 15. Buscou-se observar em que medida a topologia pode interferir no desempenho do algoritmo. Os resultados estão apresentados na Tabela 6.

Tabela 6: Resultados computacionais relativos à topologias de rede

Caso	CPU (s)	Iterações
Acoplada - Descenso Centr.	6,64	26.174
Acoplada - Newton Centr.	1,49	3.868
Acoplada - Descenso Distr.	85,28	193.332
Acoplada - Newton Distr.	73,04	63.858
Desacoplada - Descenso Centr.	4,78	17.963
Desacoplada - Newton Centr.	1,31	3.182
Desacoplada - Descenso Distr.	37,80	88.646
Desacoplada - Newton Distr.	35,09	38.230

Os algoritmos distribuídos diminuiram o tempo total praticamente pela metade entre a rede acoplada e a rede desacoplada. O resultado é esperado: quando um agente itera ele está sempre alterando as restrições de seus vizinhos. Portanto quanto menos acoplada for a rede, melhor o algoritmo distribuído irá se comportar.

5.2 SIMULAÇÃO EM TRÁFEGO

Para avaliação prática da plataforma e também em vista de futuras aplicações em controle de tráfego urbano, realizou-se experimentos sobre simulador de tráfego urbano, utilizando a técnica de controle de tráfego TUC apresentada na Seção 3.4.1 como uma RDL.

A malha viária escolhida para realização do experimento foi o centro da cidade de Macaé-RJ, cidade localizada a cerca de 200 km ao norte do Rio de Janeiro. Sua escolha se deveu primordialmente pelo DAS-UFSC ter instalado uma central de tráfego (desenvolvida no DAS (KRAUS et al., 2011)).

Os resultados foram avaliados sob dois aspectos:

- **Numérico**: o custo computacional e o tempo despendido para execução do algoritmo; e

- **Tráfego**: o impacto da técnica do controle sobre o tráfego, inferidos através de métricas de tráfego que o simulador fornece como o atraso total dos veículos e velocidade média.

5.2.1 O Software de Simulação

Utilizou-se o simulador microscópico de tráfego Aimsun (BARCELÓ; CASAS, 2002) para realização dos experimentos. Um simulador microscópico de tráfego é um software capaz de representar o comportamento individual de cada veículo utilizando um modelo de seguimento veícular (GIPPS, 1981) (*car-following model*), fornecendo como saída diversas métricas de tráfego como velocidade média, atraso médio, *etc*. Especificamente no Aimsun um modelo de seguimento comportamental é utilizado (OLSTAM; TAPANI, 2004). Em uma comparação realizada coletando-se dados em campo, Aimsun apresentou resultados mais precisos em relação a outros dois simuladores largamente utilizados (PANWAI; DIA, 2005).

Além de ser possível modelar uma rede viária, Aimsun oferece uma API (Application Programming Interface) em que é possível interagir dinamicamente com a simulação, como trocar o tempo de verde de um semáforo ou obter o número de veículos na fila de determinada via. A API está disponível para uma série de linguagens, mais uma vez utilizou-se Python.

Com a malha viária e a API à disposição é possível comparar diferentes cenários, como por exemplo a diferença de desempenho entre tempo fixo e tempo real ou o mesmo cenário com o algoritmo de controle configurado com parâmetros diferentes para auxílio no ajuste de parâmetros de controle.

No presente trabalho comparou-se tempo fixo, LQR e MPC do ponto de vista das métricas de tráfego e realizou-se também a execução centralizada e distribuída do controle MPC para análise computacional.

5.2.2 Modelo de Simulação

O modelo de simulação pode ser visto na Figura 16, composto por:

- 16 junções;
- 34 estágios controlados e 2 não controlados; e
- 41 pistas, mapeadas para 41 estados.

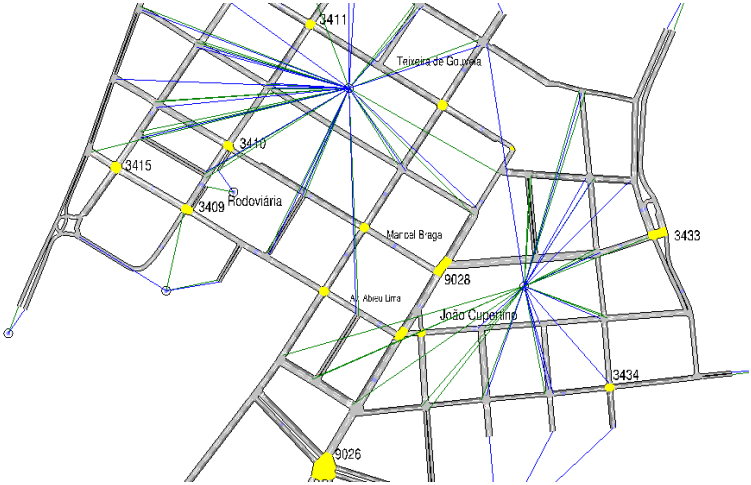

Figura 16: Malha simulada em Macaé-RJ.

No contexto da implantação do sistema em Macaé-RJ foi realizada uma coleta de campo em que foram obtidos os fluxos de saturação e as taxas de conversão de cada via. A quantidade de veículos entrando na malha viária foi obtida através de relatórios de contagem que a central de controle de tráfego em instalação fornece.

5.2.3 Modelagem em Redes Dinâmicas Lineares

Uma Rede Dinâmica Linear foi construída a partir da malha viária apresentada na seção anterior. O grafo que representa esta rede dinâmica é ilustrado na Figura 17, contendo os seguintes componentes:

- 16 junções mapeadas para 16 agentes (subsistemas); e
- 41 vias, mapeadas para 41 estados.

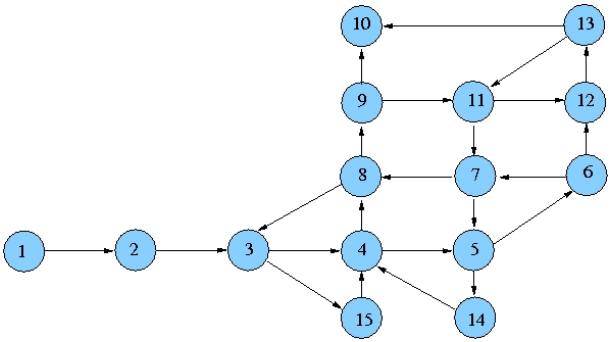

Figura 17: Rede dinâmica linear de Macaé-RJ.

Para o caso deste experimento, tanto a descrição da malha viária quanto os ajustes dos parâmetros de controle estavam disponíveis no banco de dados da central de controle de tráfego desenvolvida no contexto do projeto (KRAUS et al., 2011). Para acesso a estes dados, utilizou-se de módulos da central para acesso à banco de dados.

5.2.4 Análise Numérica

A análise numérica foi realizada com objetivo de observar os resultados estritamente numéricos, tempo despendido e módulo da diferença em relação a solução ótima, sem aplicar a solução no simulador.

Ambos algoritmos, distribuído e centralizado, convergem para solução a mesma solução de acordo com a tolerância estipulada. No entanto é possível interromper a execução do algoritmo e utilizar a solução existente até aquele momento que, embora não ótima, pode ser razoável. Foi realizada a técnica de parada prematura de maneira semelhante encontrada em (WANG; BOYD, 2010).

A parada prematura foi realizada nos algoritmos distribuídos, limitando o número de *inner iterations* (Consulte o Algoritmo 2). O objetivo é avaliar se é possível parar prematuramente em tempos menores com soluções não distantes da solução ótima.

DESCRIÇÃO DO EXPERIMENTO

Para este experimento foram realizados sobre 12 condições iniciais diferentes, sendo quatro de tráfego leve, quatro de tráfego médio e os últimos quatro de tráfego pesado. Neste trabalho adotou-se filas $\mathbf{x}_m(k)$ como um percentual de filas máximas $\mathbf{x}_m^{\max}$. Para tráfego leve as médias são de 20%, médio de 30% e pesado de 50%. Todas podendo ter variação de 10% para mais ou para menos com distribuição uniforme. Os resultados podem ser vistos na Tabela 7.

Tabela 7: Resultados computacionais relativos à diferentes números de iterações, horizontes de predição e *inner iterations*. A linha ∞ refere-se a executar o algoritmo normalmente até alcançar a tolerância estipulada

Inner	Horizonte $T = 1$		Horizonte $T = 2$	
Iter.	CPU (s)	Dist. (%)	CPU (s)	Dist. (%)
50	4.36	0.1589	21.81	0.9788
100	8.09	0.0073	45.73	0.1816
300	12.70	0.0000	124.31	0.0004
1000	12.47	0.0000	258.56	0.0000
∞	14.19		452.53	

Quanto à qualidade da solução, os resultados apresentados na Tabela 7 mostram que o interrompimento prematuro não compromete significativamente a solução, apresentando uma distância máxima da solução de 0,97%. Outro item a ser observado é que em função da vizinhança estendida crescer com o horizonte de predição, o tempo de computação para horizonte dois cresceu significativamente no algoritmo distribuído.

O resultado interessante é que os resultados para parada prematura apresentaram erros percentuais pequenos para um tempo de computação significativamente menor. Considerando por exemplo 0,97% do TUC-DMPC-50. Após a solução ser alcançada, é necessário ainda que todos os tempos de verde sejam inteiros. Em um ciclo de 90 segundos, um arredondamento pode apresentar um erro maior do que o apresentado, portanto essa distância não compromete o desempenho.

A Figura 18 ilustra o comportamento da distância em relação à solução

ótima em função de *inner iterations*. O módulo decresce com maior velocidade nas primeiras iterações, alcançando uma solução próxima em poucas iterações. A medida que o número de iterações passa, a velocidade de decrescimento passa a ser menor.

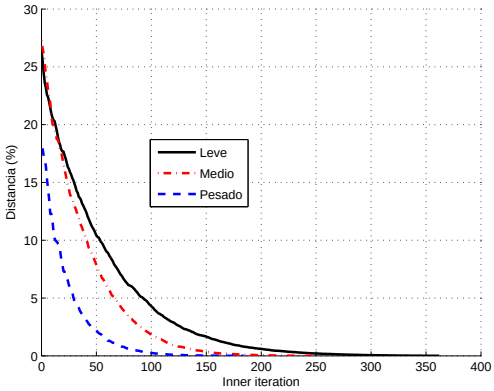

Figura 18: Distância relativa $|\theta(\widehat{\mathbf{u}}^{(l)} - \theta(\widehat{\mathbf{u}}^{\star})|/|\theta(\widehat{\mathbf{u}}^{\star})|$ por *inner iteration* l.

5.2.5 Análise de Tráfego

Realizou-se simulações sobre a malha viária de Macaé-RJ no simulador Aimsun com dois objetivos:

- comparar a execução centralizada e distribuída do TUC-MPC (MPC e DMPC) do ponto de vista de computacional; e
- comparar métricas de tráfego entre controle de tráfego em tempo real com TUC-LQR e em tempo real com TUC-MPC.

Utilizou-se as contagens das vias de entrada e taxas de conversão de um levantamento de campo realizado na época de implantação de uma central de controle de tráfego no local. O ciclo de 90 segundos, o mesmo de tempo fixo, suporta sem folga os fluxos de entrada. A defasagem também foi fixa e a mesma das de tempo fixo.

Não foi feito procedimento quantitativo para calibração do modelo. No entanto foi verificado se as contagens das vias do centro da malha estavam

condizentes com a realidade, uma vez que as entradas foram definidas pela contagens das vias de entrada e das taxas de conversão.

O tempo de simulação foi de uma hora, com uma perturbação nas contagens durante 20 minutos da simulação, do minuto 20 ao 40. Não houve tempo de "*warm up*" (tempo em que a simulação roda sem coletar estatísticas.

Foram realizados 12 replicações nos seguintes métodos:

- **Tempo Fixo:** controle em Tempo Fixo;
- **LQR:** controle TUC-LQR;
- **MPC:** controle TUC-MPC centralizado;
- **DMPC-50:** controle TUC-DMPC executando no máximo 50 *inner iterations*;
- **DMPC-100:** controle TUC-DMPC executando no máximo 100 *inner iterations*; e
- **DMPC-300:** controle TUC-DMPC executando no máximo 300 *inner iterations*.

Como há diferenças em parâmetros de filtragem e sintonia entre LQR e MPC, todos foram simulados em condições nominais com a descrição da malha utilizada no sistema em funcionamento em Macaé. A tabela 8 apresenta os resultados com o método utilizado, o atraso em segundos por kilômetro, veículos esperando que é o número de veículos esperando nas vias de entrada para entrar na malha, Fluxo total e o total de CPU por ciclo em segundos.

A técnica MPC apresentou menores tempos de atraso do que a LQR. A técnica DMPC com parada prematura foi a que apresentou, surpreendentemente, os melhores resultados. Para números maiores de iterações (DMPC-100 e DMPC-300) os resultados foram parecidos. Isto ocorre devido a diferenças entre o modelo e o sistema real.

A técnica MPC também apresentou maior fluxo do que LQR, que por sua vez apresentou fluxo maior do que tempo fixo. Portanto as técnicas de malha fechada apresentadas comportam fluxo maior que Tempo Fixo.

Provavelmente o fator que tem maior influência no desempenho do MPC é a manipulação explícitas das restrições, enquanto na abordagem LQR os tempos de verde são calculados por uma matriz de realimentação (matriz L) e as restrições são tratadas posteriormente. O fato que até mesmo em horizonte 1 se apresenta melhores resultados pode ser um indicativo deste fato.

Tabela 8: Resultados de Simulações com os diferentes métodos e horizontes de predição.

Método		Atraso (s/km)	Esperando	Fluxo	CPU(s)
LQR		103.88	115	5221	
Tempo Fixo		96.49	290	5013	
$T = 1$	MPC	96.75	0	5369	10.94
	DMPC-50	91.27	0	5369	4.89
	DMPC-100	95.80	0	5369	8.28
	DMPC-300	96.56	0	5373	15.30
$T = 2$	MPC	97.71	0	5369	62.68
	DMPC-50	96.20	0	5354	25.07
	DMPC-100	99.33	0	5363	40.63
	DMPC-300	96.56	0	5367	97.87

O tempo de CPU apresentado é o total despendido por ciclo. Tanto para o MPC centralizado, 10,93 segundos, quanto para o DMPC-50, 4,89 segundos, apresentam tempos totalmente compatíveis com a aplicação que tem periodo de amostragem de um ciclo, tempos em geral acima de 80 segundos.

Como visto no Capítulo 4, a solução distribuída de MPC é realizada com os agentes executando serialmente e utilizando memória compartilhada como forma de comunicação. A solução por DMPC ainda pode se valer de poder executar em paralelo, no caso da malha de Macaé-RJ é possível formar 4 grupos que executam em paralelo (trocando informações a cada iteração).

5.3 SUMÁRIO

Neste capítulo avaliou-se o comportamento e desempenho dos algoritmos apresentados em uma série de experimentos, observando-se a influência de parâmetros e características da rede, chamando mais atenção o resultado de acoplamento da rede.

Na sequência apresentou-se os resultados de simulação em tráfego urbano. Primeiramente apresentou-se o simulador e o motivo da escolha da malha viária de Macaé-RJ, fundamentalmente por uma central de controle de tráfego ter sido lá implantada.

A análise numérica mostrou que o algoritmo centralizado foi mais rápido e tanto centralizado e distribuído convergiram sempre para a mesma solução. Os resultados de simulação mostraram melhor desempenho da abordagem MPC sobre a LQR.

A técnica de parada prematura também se mostrou promissora, uma

vez que se consegue soluções boas o suficiente para a aplicação em tempos razoáveis.

6 CONCLUSÕES

Um trabalho que culminou com uma aplicação em simulação de Controle Preditivo Distribuído (DMPC) sob o paradigma de Redes Dinâmicas Lineares foi apresentado. A plataforma desenvolvida é genérica e pode ser utilizada para qualquer problema de controle, não se limitando à controle de tráfego urbano. Os resultados práticos mostram que Controle Preditivo baseado em Modelo (MPC) pode ser utilizado com perspectivas de bons resultados em controle de tráfego urbano.

A plataforma pode servir como uma ferramenta para se investigar características de algoritmos distribuídos. Alguns experimentos foram realizados, chegando a resultados esperados segundo a teoria existente (especialmente (CAMPONOGARA; SCHERER, 2011). Destes, podemos destacar o desacoplamento da rede melhorar o desempenho dos algoritmos distribuídos. Experimentos como problemas com região factível pequena e onde vários agentes desacoplados iteram concomitantemente são sugestões.

Mesmo sem a utilização dos algoritmos distribuídos, os três módulos (modelagem, geração e *solvers*) podem ser utilizados separadamente. Modelar e gerar o problema com os dois módulos disponíveis na plataforma e depois utilizar um outro *solver* com algoritmo eventualmente mais adequado ao problema (primal-dual, barreira, *infeasible start*, *etc*) é uma possível aplicação. A plataforma pode também ser estendida inserindo outros algoritmos distribuídos como apresentadas em (CAMPONOGARA; OLIVEIRA, 2009).

Para aplicação no controle de tráfego urbano mais experimentos precisam ser conduzidos. A técnica MPC é fortemente dependente do modelo que se tem do sistema, portanto variações paramétricas interferem diretamente no comportamento do controle. No tráfego urbano taxas de conversão mudam constantemente e até mesmo o fluxo de saturação não é constante, portanto trabalhos futuros podem ser feitos tendo em vista o comportamento do controle por MPC com variações paramétricas.

Deve ser investigado o potencial de reconfigurabilidade do MPC Distribuído. No caso por exemplo de uma junção ficar sem comunicação, naturalmente aquele agente ficará fora do modelo perante aos outros agentes na vizinhança, diferentemente de centralizado onde todo o modelo deve ser reconstruído (função objetivo, restrições) ou, como é feito atualmente, o cálculo leva em consideração todos os agentes, estando eles em comunicação ou não, trazendo erros ao modelo.

Outro fato importante é que apesar do algoritmo ser distribuído, os agentes não necessariamente precisam estar fora do mesmo processador. O algoritmo pode valer-se de processadores com múltiplos núcleos. Nenhum

teste real foi feito com múltiplos núcleos, mas como perspectiva em uma malha como a de Macaé-RJ para horizonte de predição um, poderia-se formar 4 grupos de agentes que nunca estão acoplados e rodar cada grupo em um núcleo.

Todo este trabalho esteve no contexto da implantação da Central de Controle de Tráfego CONTREAL desenvolvida no DAS-UFSC com tecnologia totalmente nacional, primeira do Brasil. Esta central foi implantada na cidade de Macaé-RJ com sucesso (KRAUS et al., 2011). Este projeto conseguiu algo raro no Brasil: a Universidade desenvolver tecnologia, gerar um produto comercial e ganhar com royalties deste para financiar novas pesquisas.

APÊNDICE A – Terminologia

Diversos termos de controle de tráfego foram utilizadas no texto. Apresenta-se neste anexo a definição dos principais deles:

A.1 VIA (*LINK*)

Segmento viário urbano que liga duas interseções, ou que serve de entrada ou saída para uma malha viária urbana.

A.2 APROXIMAÇÃO

Uma via que chega em uma intersecão. A figura 19 representa uma pequena malha viária temos L1, L2 e L3 são aproximacões que chegam na intersecão 1 (J1). Na intersecão 2 (J2) chegam L4 e L5.

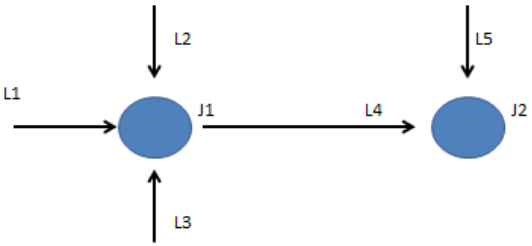

Figura 19: Exemplo de uma malha veicular

A.3 DEFASAGEM

Também conhecida pelo termo em inglês *offset*. É a diferença entre o início de dois estágios pré-determinados de duas interseções consecutivas. Pode ser positiva ou negativa. É positiva se o estágio da interseção a jusante inicia após o estágio da interseção a montante e negativa no caso contrário (GERLOUGH; HUBER, 1975).

A.4 CICLO

Um ciclo compreende a repetição da sequência de indicações semafóricas (fases) de uma interseção, sendo o intervalo de repetição conhecido como tempo de ciclo, usualmente medido em segundos (GERLOUGH; HUBER, 1975). Ver figura 20.

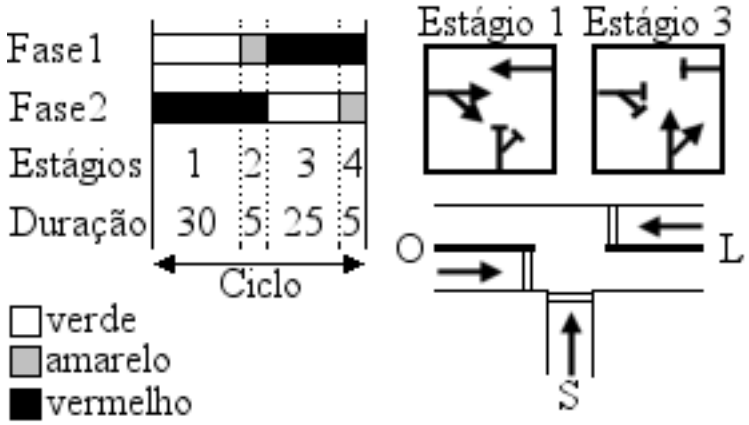

Figura 20: Exemplo de ciclo, estágios e fases (retirado de (CARLSON, 2006)).

A.5 ESTÁGIO

Conjunto de indicações semafóricas (fases) de todas as aproximações de uma interseção em determinado instante de tempo. A Figura 20 mostra uma interseção e seus quatro estágios (GERLOUGH; HUBER, 1975).

A.6 FASE

Indicação semafórica, sequência de verde, amarelo e vermelho para uma aproximação. Em teoria deveria existir uma fase para cada aproximação, entretanto na prática quando duas aproximações apresentam sempre a mesma indicação semafórica as lâmpadas são ligadas em paralelo e duas aproximações são "ligadas"na mesma fase (GERLOUGH; HUBER, 1975). Ver figura 20.

A.7 FLUXO (Q)

Número de veículos (taxa) que passam um ponto durante um período de tempo especificado (GERLOUGH; HUBER, 1975). Normalmente é resultado de uma medida realizada em pouco tempo e transformada para uma unidade de tempo maior.

A.8 FLUXO DE SATURAÇÃO (S)

Taxa máxima de veículos em relação ao tempo, que seriam descarregados de uma via se esta contasse com uma fila de tamanho infinito e indicação semafórica verde durante todo o tempo.

A.9 OCUPAÇÃO

Percentual de tempo em que um ponto da via é ocupada por veículos. Matematicamente definido como (GORDON; TIGHE, 2005):

$$\theta = \frac{100}{T} \sum_{i=1}^{N} (t_i - D) \tag{A.1}$$

onde θ é a ocupação medida sobre o período de tempo T; T é o período de observação em segundos; t_i tempo de pulso do veículo i (isto é, total de tempo em que o sensor ficou detectando o veículo i); D é o tempo de descida do sensor (tempo transcorrido para que o sensor deixe de detectar presença depois de o veículo passar pelo sensor) menos o tempo de subida do sensor, frequentemente desconsiderado.

A.10 GRAU DE SATURAÇÃO

Proporção entre o fluxo q e a capacidade da via (HCM, 2000). É citado suas propriedades na Seção 2.2. Para uma aproximação pode ser calculado como segue:

$$gs_i = \frac{n_i C}{s_i G_i} \tag{A.2}$$

onde n_i é o fluxo atual ou projetado para a aproximação i; s_i é o fluxo de saturação da aproximação i; g_i é o tempo de verde alocado para aproximação

i; e C é o tempo de ciclo.

A.11 PLANO

Plano é o que engloba as variáveis de controle em tráfego. Em (GORDON; TIGHE, 2005) são citados ciclo, defasagem e percentuais de verde para cada estágio como variáveis básicas de um plano. Em geral refere-se a uma interseção, no contexto de uma região denota um plano para cada interseção da região. Os planos podem possuir variáveis diferentes de acordo com o modo de operação. Controle atuado, por exemplo, tem parâmetros de brecha que em tempo fixo não são necessários.

REFERÊNCIAS BIBLIOGRÁFICAS

ABOUDOLAS, K.; PAPAGEORGIOU, M.; KOSMATOPOULOS, E. Control and optimization methods for traffic signal control in large-scale congested urban road networks. *American Control Conference*, p. 3132–3138, 2007.

ABOUDOLAS, K.; PAPAGEORGIOU, M.; KOSMATOPOULOS, E. Control and optimization methods for traffic signal control in large-scale congested urban road networks. In: *Proceedings of the American Control Conference*. New York, USA: American Automatic Control Council, 2007. p. 3132–3138.

ABOUDOLAS, K.; PAPAGEORGIOU, M.; KOSMATOPOULOS, E. Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research, Part C*, v. 17, n. 2, p. 163–174, 2009.

BARCELÓ, J.; CASAS, J. Dynamic network simulation with Aimsun. In: *Proc. of the International Symposium on Transport Simulation*. Barcelona: Transportation Research Board, 2002.

BOYD, S.; VANDENBERGHE, L. *Convex Optimization*. New York, USA: Cambridge University Press, 2004.

CAMACHO, E. F.; BORDONS, C. *Model Predictive Control*. Berlin, Alemanha: Springer, 2004.

CAMPONOGARA, E.; JIA, D.; KROGH, B.; TALUKDAR, S. N. Distributed model predictive control. *IEEE Control Systems Magazine*, v. 22, n. 1, p. 44–52, 2002.

CAMPONOGARA, E.; OLIVEIRA, L. B. de. Distributed optimization for model predictive control of linear dynamic networks. *IEEE Transactions on Systems, Man and Cybernetics – Part A*, v. 39, n. 6, p. 1331–1338, 2009.

CAMPONOGARA, E.; SCHERER, H. F. Distributed optimization for model predictive control of linear dynamic networks with control-input and output constraints. *IEEE Transactions on Automation Science and Engineering*, v. 8, n. 1, 2011.

CAMPONOGARA, E.; TALUKDAR, S. N. Designing communication networks to decompose network control problems. *INFORMS Journal on Computing*, v. 17, n. 2, p. 207–223, 2005.

CARLSON, R. *Aplicação de Maximização de Largura de Banda no Controle de Tráfego Urbano em Tempo Real*. Dissertação (Mestrado) — Departamento de Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, Brasil, 2006.

COHEN, S. L. Concurrent use of MAXBAND and TRANSYT signal timing programs for arterial signal optimization. *Transportation Research Record*, v. 906, p. 81–84, 1982.

CORMEN, e. a. T. H. *Introduction to Algorithms*. San Diego, USA: The MIT Press, 2002. 347-349 p.

CRABTREE, M. R.; VINCENT, R. A.; HARRISON, S. *APPLICATION GUIDE 28 - TRANSYT/10 USER GUIDE*. Crowthorne, UK, 1996.

CVXOPT. *Biblioteca para otimização convexa para Python*. 2010. `http://abel.ee.ucla.edu/cvxopt/`.

DIAKAKI, C. M.; DINOPOULOU, V.; ABOUDOLAS, K.; PAPAGEORGIOU, M.; BEN-SHABAT, E.; SEIDER, E.; LEIBOV, A. Extensions and new applications of the traffic signal control strategy TUC. *Transportation Research Record*, v. 1856, p. 202–216, 2003.

DIAKAKI, C. M.; PAPAGEORGIOU, M.; ABOUDOLAS. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, v. 10, p. 183–195, 2002.

DORATO, P.; ABDALLAH, C.; CERONE, V. *Linear Quadratic Control: A Introduction*. New Jersey, USA: Prentice Hall, 1995.

ECLIPSE. *Ambiente de Desenvolvimento Multi-linguagem livre*. 2010. `http://www.eclipse.org`.

FHWA. *Traffic Signal Timing Manual*. Portland, USA, 2008.

GARCIA, D. M. P. C.; MORARI, M. Model predictive control: Theory and practice-a survey. *Automatica*, v. 25, n. 3, p. 335–348, 1989.

GERLOUGH, D. L.; HUBER, M. J. *Traffic Flow Theory: a monograph*. Washington, D.C., 1975.

GIPPS, P. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, v. 15, n. 2, p. 105–111, abr. 1981. ISSN 0191-2615.

GORDON, P.; TIGHE, W. *Traffic Control Systems Handbook*. Washington D.C., USA: Office of Transportation Management Federal Highway Administration, 2005.

GU, D.; HU, H. A stabilizing receding horizon regulator for nonholonomic mobile robots. *IEEE Transactions on Robotics*, v. 21, n. 5, p. 1022–1028, 2005.

HCM. *Highway Capacity Manual*. Washington D.C., USA: Transportation Research Board, 2000.

KLEIN, L.; MILLS, M.; GIBSON, D. *Traffic Detectors Handbook*. Washington D.C., USA: Office of Transportation Management Federal Highway Administration, 2006.

KRAUS, W.; SOUZA, F.; CARLSON, R.; DANTAS, L.; PAPAGEORGIOU, M.; CAMPONOGARA, E.; KOSMATOPOULOS, E.; ABOUDOLAS, K. A low-cost implementation of the tuc adaptive control strategy. *IEEE ITS Magazine*, v. 2, p. 6–17, 2011.

LAPACK. *Biblioteca para algebra linear*. 2010. `http://www.netlib.org/lapack/`.

LOWRIE, P. R. The Sydney co-ordinated adaptive traffic system - principles, methodology and algorithms. In: *Proceedings of the IEE International Conference on Road Traffic Signalling*. London, UK: IEEE, 1982. p. 67–70.

NUMPY. *Biblioteca para matrizes N-Dimensionais e Álgebra Linear. integrou-se ao SciPy em 2010.* 2010. `http://numpy.scipy.org/`.

OLSTAM, J. J.; TAPANI, A. *Comparison of Car-Following Models*. Linkoping, Sweden: Swedish National Road and Transport Research Institute, 2004. (VTI, 960 A).

PANWAI, S.; DIA, H. Comparative evaluation of microscopic car-following behavior. *IEEE Transactions on Intelligent Transportation Systems*, v. 6, n. 3, p. 314–325, set. 2005.

PYDEV. *Plugin para Desenvolvimento em Python no Eclipse*. 2010. `http://www.pydev.org`.

RICO, J. N.; CAMACHO, E. F. *Control of Dead-Time Process*. Berlin, Alemanha: Springer, 2006.

ROBERTSON, D. I.; BRETHERTON, R. D. Optimizing networks of traffic signals in real time - the SCOOT method. *IEEE Transactions on Vehicular Technology*, v. 40, n. 1, p. 11–15, 1991.

SCATTOLINI, R. Architectures for distributed and hierarchical model predictive control - a review. *Journal of Process Control*, v. 19, p. 723–731, 2009.

SCIPY. *Conjunto de bibliotecas para computação científica em Linguagem Python*. 2010. `http://www.scipy.org/`.

SOUZA, F. A.; PECCIN, V. B.; CAMPONOGARA, E. Distributed model predictive control applied to urban traffic networks: Implementation, experimentation, and analysis. In: *Proceeding of the 6th IEEE Conference on Automation Science and Engineering*. [S.l.]: IEEE, 2010.

VENKAT, A. N.; HISKENS, I. A.; RAWLINGS, J. B.; WRIGHT, S. J. Distributed MPC strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, v. 16, n. 6, p. 1192–1206, 2008.

WANG, Y.; BOYD, S. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems*, v. 18, n. 2, p. 267–278, 2010.