Ricardo Santos da Silva

A DISTRIBUTED DUAL ALGORITHM FOR DISTRIBUTED MPC OF LINEAR DYNAMIC NETWORKS

Dissertation presented to the Graduate Program in Automation and Systems Engineering in partial fulfillment of the requirements for the degree of Master in Automation and Systems Engineering.

Advisor: Prof. Eduardo Camponogara

Florianópolis

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Silva, Ricardo da
A Distributed Dual Algorithm for Distributed MPC of Linear Dynamic Networks / Ricardo da Silva ; orientador, Eduardo Camponogara, 2017.
84 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2017.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Otimização distribuída. 3. Controle de tráfego urbano. 4. Redes dinâmicas lineares. 5. Lagrangeano aumentado. I. Camponogara, Eduardo. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

Ricardo Santos da Silva

A DISTRIBUTED DUAL ALGORITHM FOR DISTRIBUTED MPC OF LINEAR DYNAMIC NETWORKS

This Dissertation is recommended in partial fulfillment of the requirements for the degree of "Master in Automation and Systems Engineering", which has been approved in its present form by the Graduate Program in Automation and Systems Engineering.

Florianópolis, July 5th 2017.

Graduate Program Coordinator

Dissertation Committee:

Prof. Eduardo Camponogara
Advisor
Federal University of Santa Catarina

Dr. Andres Codas
IBM Research

Prof. Daniel Ferreira Coutinho
Federal University of Santa Catarina

Prof. Rodrigo Castelan Carlson
Federal University of Santa Catarina

This work is dedicated to my family and friends.

ACKNOWLEDGEMENTS

I would like to thank my family, Ronaldo, Edna and Gabriela for all the support in this ardue journey, without them it would not be possible to complete this dissertation, my friends who listen to all my problems and helped me endure the bad times, especially Marco and Maurício my best friends, who helped me vent out my frustrations, my friends in the laboratory - GOS - for all the insight and for providing a friendly and nice work enviroment, and last but not least, my advisor Professor Camponogara who took me like a son and helped me grow to my potential with patience with all my shortcomings.

RESUMO

Uma rede dinâmica linear (LDN) é um sistema de subsistemas interligados que são acoplados através de dinâmicas e restrições, que podem modelar sistemas geograficamente distribuídos, como redes de tráfego urbano. Para o controle preditivo baseado em modelo (MPC) de LDNs, um algoritmo dual distribuído é proposto para otimizar os sinais de controle em um horizonte de predição. Enquanto um problema mestre atualiza multiplicadores Lagrangeanos e fatores de penalidade para as restrições, um problema escravo é decomposto em um conjunto de subproblemas distribuídos cujas variáveis são restritas apenas em sinal. Sob condições de convexidade, foi demonstrado que o algoritmo distribuído produz uma sequência de iterandos que converge para o ótimo global. Numa aplicação ao controle de tempo de verde em redes de tráfego veicular urbano, o algoritmo distribuído produziu soluções ótimas o que confirma a convergência estabelecida pela análise teórica.

Palavras-chave: Otimização distribuída; Lagrangeano aumentado; Controle distribuído; MPC; Redes dinâmicas lineares; Controle de tráfego urbano.

ABSTRACT

A linear dynamic network (LDN) is a system of interconnected subsystems that are coupled through dynamics and constraints, which can model geographically distributed systems such as urban traffic networks. For model predictive control (MPC) of LDNs, a distributed dual algorithm is proposed for optimizing control signals over a prediction horizon. While a master problem updates Lagrangian multipliers and penalty factors for the constraints, a slave problem is decomposed into a set of distributed subproblems whose variables are constrained only in sign. Under convexity assumptions, the distributed algorithm was shown to produce a sequence of iterates converging to the globally optimal solution. In an application to green-time control of an urban traffic network, the distributed algorithm yielded optimal solutions that corroborate the theoretical analysis.

Keywords: Distributed optimization; Augmented Lagrangian; Distributed control; MPC; Linear dynamic networks; Urban traffic control.

LIST OF FIGURES

LIST OF TABLES

SYMBOLS

x	Variable
$\mathbf{x}$	Vector of variable
$f(x)$	Function of x
$c(x)$	Matrix of inequality constraints of x
$d(x)$	Matrix of equality constraints of x
$\nabla f(x)$	Gradient of $f(x)$
$\mathcal{L}(x)$	Lagrangian of x
λ	Lagrangian multiplier
μ	Penalizer multiplier
$Q(x, \mu)$	Quadratic penalty function
$P(x, \mu)$	Logarithmic barrier function
u	Control
$\mathbf{u}$	Vector of control
J	Cost function
$\mathbf{A}$	Matrix that infers the variable's past dynamics
$\mathbf{B}$	Matrix that infers the control's past dynamics
$\widehat{\mathbf{x}}$	Vector of variables in the observed period of time
$\widehat{\mathbf{u}}$	Vector of controls in the observed period of time
$\widehat{\mathbf{A}}$	Matrix that infers the variable's past dynamics for all variable in the observed period of time
$\widehat{\mathbf{B}}$	Matrix that infers the control's past dynamics for all control in the observed period of time
$\mathbf{Q}$	Matrix with variables weigths for the cost function J
$\mathbf{R}$	Matrix with controls weigths for the cost function J
$\mathbf{X}$	Matrix with variables constraints
$\mathbf{U}$	Matrix with controls constraints
$\mathbf{x}^{\max}$	Vector with variables bounds
$\mathbf{u}^{\max}$	Vector with controls bounds
$\widehat{\mathbf{H}}$	Matrix of minimizing problem
$\widehat{\mathbf{g}}$	Vector of minimizing problem
$\widehat{\mathbf{s}}$	Vector of slack variables

$\widehat{\mathbf{Z}}$	Matrix of control and variables constraints
$\widehat{\mathbf{z}}^{bd}$	Vector of control and variables bounds
$\widehat{\mathbf{y}}$	Vector of neighborhood control variables
$\widehat{\mathbf{v}}$	Vector of control and slack variables
C	Cycle time
q	Inflow of a link
p	Outflow of a link
d	Demand of a link
s	Outflow within a link
t	Turning rate towards a link
S	Saturation flow
$\widetilde{\mathbf{u}}$	Vector of nominal control
T_h	Prediction horizon

CONTENTS

1 INTRODUCTION

1.1 MOTIVATION

Geographically distributed systems arise from the interconnection of dozens, and even hundreds, of dynamic subsystems that influence and impose constraints on one another. In urban traffic networks, for instance, each intersection is a subsystem whose control variables correspond to green time signals, and whose state is characterized by vehicle queues that evolve in time under the influence of the neighboring subsystems. Physical and technological constraints are translated into road capacity, speed limits, and the splitting of green time signals at the intersections [1].

The management and control of large dynamic systems remains a challenge, despite the scientific and technological advancement witnessed in the last decades. Two extremes of control strategies are the *centralized* and *decentralized* control. Decentralized control advocates the independent design of control laws for each subsystem, while considers the influence from other subsystems as perturbations [2]. On the other hand, centralized control provides system wide operation by explicitly considering the interactions and constraints between subsystems. Despite being scalable, decentralized strategies can be very restrictive with respect to stability and often do not offer guarantees with respect to a performance criterion, since the distributed controllers (agents) can work at cross purposes. Instead, centralized control strategies, such as model-based predictive control (MPC) [3], optimize a performance criterion but are not always scalable for geographically distributed systems, and may also lack robustness due to the concentration of communications and computation at the control center.

Aiming to combine the desirable features of centralized and decentralized control, research in the past decades led to the development of hierarchical and distributed control for large-scale systems [4]. Hierarchical control employs models that are structured in layers which operate at different time scales, delegating the decisions to the various layers. The top layer relies on simplified models to make system wide prediction and reach decisions that are informed to the lower layers which, in turn, rely on detailed models to make local decisions that are consistent with the strategies received from the upper layers. While hierarchical control operates with layers that vary from simplified and far reaching to detailed and local, distributed control tackles complexity by dividing the control effort among the distributed systems.

Motivated by the need of distributed control strategies to operate geographically distributed systems, this dissertation proposes a new distributed algorithm for a class of linear dynamic networks. The Linear Dynamic Network (LDN) of concern consists of a directed graph whose nodes represent dynamic subsystems and whose arcs model the influence of the upstreams on the downstreams subsystems. As such, the dynamics of subsystem state is characterized by a discrete-time linear dynamic equation that depends on the local state and controls, and also on the control signals of the upstream subsystems. Constraints can be imposed on the local controls and state, there by inducing constraint couplings among the subsystems which render distributed optimization a challenge. According to the classification from [5], the problem of operating the considered LDN is a kind of Decoupled Cost and Coupled Constraint (DCCC) problem, which becomes Coupled Cost and Coupled Constraint (CCCC) if the state variables are eliminated by replacing them with the dynamic equations.

Linear dynamic networks are convenient models for a wide range of systems, particularly urban traffic networks. Intersections are modeled as nodes of the LDN, while roads are represented by arcs. Despite the long-lasting research and developments worldwide [6], urban signal control is still an area susceptible of further improvements, particularly under saturated traffic conditions. The usually limited availability of space in the urban centers prevents the extension of the existing infrastructure, and, along with the continuously increasing mobility requirements, urge for solutions that will release the serious congestion problems through the best possible utilization of the available infrastructure. From the control point of view, this may be translated into the employment of actuated systems that respond automatically to the prevailing traffic conditions. In view of the relevance of traffic control, this dissertation will consider and application of the distributed algorithm to the green-time control of an urban traffic network.

1.2 OBJECTIVE

The main objective of this dissertation rests on the design, convergence analysis and computational test of a distributed dual algorithm based on the augmented Lagrangian for the distributed control of linear dynamic networks. The choice of such algorithm comes from one its strong points, which is the ability to handle cou-

pling inequalities and equalities, whereas other algorithms such as the distributed interior-point method cannot tackle equalities [7].

Augmented Lagrangian methods are a certain class of algorithms for solving constrained optimization problems. They have similarities to penalty methods in that they replace a constrained optimization problem by a series of unconstrained problems and add a penalty term to the objective; the difference is that the augmented Lagrangian method adds yet another term, designed to mimic a Lagrange multiplier. By dualizing the coupling constraints with an augmented Lagrangian, the optimization problem for the entire LDN is decomposed into a set of distributed problems, coupled only in the objective, thereby enabling the design of a distributed algorithm. To this end, a distributed gradient projection method will be developed for optimizing the augmented Lagrangian.

1.3 DISSERTATION ORGANIZATION

This dissertation is organized as follows.

Chapter 2 presents the key concepts and mathematical concepts necessary for the understanding of this dissertation.

Chapter 3 presents linear dynamic networks (LDN), their advantages and applications, such as control and problem decomposition.

Chapter 4 presents the algorithm core to the dissertation, the dual augmented Lagrangian algorithm which has an outer and inner loop. The inner loop consists of a distributed gradient-projection algorithm that optimizes the augmented Langrangian. The outer loop uses the approximated solution produced by the inner loop to update the Lagrangian multipliers and penalty factor.

Chapter 5 presents the application of distributed dual algorithm to the green-time control of an urban-traffic network, contrasting its performance against an off-the-shelf algorithm and a standard version of the augmented Lagrangian method in which the inner loop consists of a centralized gradient-projection method.

Finally, Chapter 6 concludes the dissertation with some final remarks and directions for future work.

2 MATHEMATICAL FUNDAMENTS

This chapter contains some mathematical fundaments important to the understanding of this dissertation. In Section 2.1 we review the basis of optimization, presenting a number of different configurations and paths to solution of said problems. In Section 2.2 basics concepts of Model Predictive Control are presented, to serve as a refresher to the reader.

2.1 BRIEF REVIEW OF OPTIMIZATION

In continuous optimization, the variables in the model are nominally allowed to take on a value from a continuous range, usually real numbers. This feature distinguishes continuous optimization from discrete or combinatorial optimization, in which the variables may be binary (restricted to the values 0 and 1), integer (for which only integer values are allowed), or more abstract objects drawn from sets with finitely many elements. Continuous optimization problems are typically solved using algorithms that generate a sequence of values of the variables, known as iterates, that converge to a solution of the problem. A continuous optimization problem is to find a solution vector x^*, for function $f(x)$ such that

$$f(x^*) \leq f(x), \quad \forall x \in \mathbb{R}^n. \tag{2.1}$$

Note that there is no loss in generality in concentrating here on minimization, since

$$\max f(x) = -\min f(x). \tag{2.2}$$

In deciding how to step from one iterate to the next, the algorithm makes use of knowledge gained at previous iterates, and information about the model at the current iterate, possibly including information about its sensitivity to perturbations in the variables. The continuous nature of the problem allows sensitivities to be defined in terms of first and second derivatives of the functions that define the model.

There are a number of subclasses of optimization problems. The simplest being the unconstrained minimization:

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2.3}$$

A more complex subclass is equality constrained minimization,

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.4a}$$

$$\text{subject to} \quad d_i(x) = 0, \quad i = 1, \ldots, q, \tag{2.4b}$$

and inequality contrained minimization,

$$\min_{x \in \mathbb{R}^n} f(x) \tag{2.5a}$$

$$\text{subject to} \quad c_i(x) \geq 0, \quad i = 1, \ldots, p, \tag{2.5b}$$

For the purpose of generalization, we can assume that the general form of optimization problems is as below:

$$\min f(x) \tag{2.6a}$$

$$\text{subject to} \quad c_i(x) \geq 0, \quad i = 1, \ldots, p \tag{2.6b}$$

$$d_i(x) = 0, \quad i = 1, \ldots, q, \tag{2.6c}$$

in which $c(x) : \mathbb{R}^n \to \mathbb{R}^p$ and $d(x) : \mathbb{R}^n \to \mathbb{R}^q$.

The most fundamental issues are: How to define a solution to the problem, and how to recognize such a point? These issues become more complex as we expand the classes of functions allowed in the formulation. The type of solution most amenable to analysis is a local solution. A point x^* is a local solution if x^* is feasible, and there is an open neighborhood $\mathcal{N}$ around x^* such that $f(x^*) \leq f(x)$ for all feasible points $x \in \mathcal{N}$. Further, x^* is a strict local solution if $f(x^*) \leq f(x)$ for all $x \in \mathcal{N}$, with $x \neq x^*$. A global solution is a point x^* such that $f(x^*) < f(x)$ for all feasible x. It is dificult to verify global optimality, even when the objective and constraints are smooth, because of the dificulty of gaining a global perspective on these functions. However, in convex optimization, where the objective f is a convex function and the set of feasible points is also convex, all local solutions are global solutions.

For unconstrained optimization of a smooth function f, we have the following necessary condition.

$$\text{If } x^* \text{is a local solution of } \min_x f(x), \text{then } \nabla f(x^*) = 0$$

where $\nabla f(x)$ denotes the gradient of f. Note that this is only a necessary condition; it is possible to have $\nabla f(x) = 0$ without x being a minimizer. To complement this result, we have the following

suficient condition.

If we have a point x^*such that $\nabla f(x^*) = 0$ with $\nabla^2 f(x^*)$ positive definite, then $x^*$is a strict local solution of $\min_x f(x)$.

Turning to constrained optimization, for general form (2.6) with smooth functions, identification of local solutions becomes somewhat more complex. We can obtain a necessary condition based on the gradients of ∇f, ∇c and ∇d, but this depends on an additional condition called a constraint qualification.

A central role in characterizing solutions of constrained optimization problems is played by the Lagrangian function, defined as follows:

$$\mathcal{L}(x, \lambda, \mu) = f(x) - \sum_{i=1}^{p} \lambda_i c_i(x) - \sum_{i=1}^{q} \mu_i d_i(x). \tag{2.8}$$

This is a linear combination of objective and constraints, where the weights λ_i and μ_i are called Lagrange multipliers. At a local solution x^* for (2.6), the following conditions will hold for some values of λ_i and μ_i

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0 \tag{2.9a}$$
$$c_i(x^*) \geq 0, \quad i = 1, \ldots, p \tag{2.9b}$$
$$d_i(x^*) = 0, \quad i = 1, \ldots, q \tag{2.9c}$$
$$\lambda_i^* \geq 0, \quad i = 1, \ldots, p \tag{2.9d}$$
$$\lambda_i^* c_i(x^*) = 0, \quad i = 1, \ldots, p \tag{2.9e}$$

Condition (2.9e) is a complementarity condition that indicates complementarity between each inequality constraint value $c_i(x^*)$ and its Lagrange multiplier λ_i^*: For each i, at least one of these two quantities must be zero. Roughly speaking, the Lagrange multipliers measure the sensitivity of the optimal objective value $f(x^*)$ to perturbations in the constraints c_i. The conditions (2.9) are often known as the Karush-Kuhn-Tucker conditions, or KKT conditions for short, named after those who discovered the conditions.

2.1.1 Penalty Function Optimization

One fundamental approach to constrained optimization is to replace the original problem by a penalty function that consists of:

- The objective of the original optimization problem.
- One additional term for each constraint, positive when the current point x violates that constraint and zero otherwise.

Most approaches define a sequence of such penalty functions, in which the penalty terms for the constraint violations are multiplied by some positive coefficient. By making this coefficient larger and larger, we penalize constraint violations more and more severely, thereby forcing the minimizer of the penalty function closer and closer to the feasible region for the constrained problem. The simplest penalty function of this type is the quadratic penalty function, in which the penalty terms are the squares of the constraint violations. For the equation (2.6) we can define a penalty function Q such as:

$$Q(x, \mu) = f(x) + \frac{1}{2\mu}([c(x)]^{-})^2 + \frac{1}{2\mu}(d(x))^2 \tag{2.10}$$

where $[y]^{-}$ denotates $\max(-y, 0)$. A general framework for algorithms based on the penalty function (2.10) can be specified as follows.

Algorithm 1: Quadratic Penalty

input: starting parameters $\mu_0 > 0$, tolerance $\tau > 0$, starting point x_0^s

for $k := 0, 1, 2, \ldots$ **do**

- **find** an approximate minimizer x_k of $Q(:, \mu_k)$ starting at x_k^s
- **if** $||Q(x, \mu_k)|| \leq \tau$ **then**
  - **stop** with an approximate solution x_k;
- **choose** the next penalty factor: $\mu_{k+1} \in (0, \mu_k)$;
- **update** the starting point for the next iteration to $x_{k+1}^s = x_k$

output: x_k^s

The parameter sequence μ_k can be chosen adaptively, based on the difficulty of minimizing the penalty function at each iteration. When the minimization of $Q(x; \mu_k)$ proves to be expensive for some k, we choose μ_{k+1} to be only modestly smaller than μ_k; for instance $\mu_{k+1} = 0.7\mu_k$. If we find the approximate minimizer

of $Q(x;\mu_k)$ cheaply, we could try a more ambitious reduction, for instance $\mu_{k+1} = 0.1\mu_k$.

A more complex approach is given by the barrier method. We start by describing the concept of barrier functions. Given the inequality-constrained problem described by (2.5), the strictly feasible region is defined by

$$\mathcal{F}^0 = \{x \in \mathbb{R}^n | c(x) > 0\} \tag{2.11}$$

Barrier functions for this problem have the properties that

- they are infinite everywhere except in $\mathcal{F}^0$.
- they are smooth inside $\mathcal{F}^0$.
- their values approach $+\infty$ as x approaches the boundary of $\mathcal{F}^0$.

The most important barrier function is the logarithmic barrier funtion, which has the form

$$-\sum_{i=1}^{p} \log c_i(x) \tag{2.12}$$

where $\log(\cdot)$ denotes the natural logarithm. For the problem in (2.5), the objective function of the barrier subproblem is given by

$$P(x,\mu) = f(x) - \mu \sum_{i=1}^{p} \log c_i(x). \tag{2.13}$$

Since the minimizer $x(\mu)$ of $P(x;\mu)$ lies in the strictly feasible set $\mathcal{F}^0$ (where no constraints are active), we can in principle search for it by using unconstrained minimization algorithms. Unfortunately, the minimizer $x(\mu)$ becomes more and more difficult to find as $\mu \to 0$. The scaling of the function $P(x;\mu)$ becomes poorer and poorer, and the quadratic Taylor series approximation (on which Newton-like methods are based) does not adequately capture the behavior of the true function $P(x;\mu)$. The generic algorithm to solve log-barrier function is very similar to the quadratic penalty function shown in Algorithm 1. Algorithm 2 gives the steps of the logarithmic barrier method.

We now discuss an algorithm known as the method of multipliers or the augmented Lagrangian method. This algorithm is related to the quadratic penalty algorithm, but it reduces the possibility of ill conditioning of the subproblems that are generated in this

Algorithm 2: Logarithmic Barrier

input: starting parameters $\mu_0 > 0$, tolerance $\tau > 0$, starting point x_0^s

for $k := 0, 1, 2, \ldots$ **do**

  find an approximate minimizer x_k of $P(:, u_k)$ starting at x_k^s

  if $||P(x, u_k)|| \leq \tau$ **then**

    stop with an approximate solution x_k;

  choose the next penalty factor: $\mu_{k+1} \in (0, \mu_k)$;

  update the starting point for the next iteration to $x_{k+1}^s = x_k$

output: x_k^s

approach by introducing explicit Lagrange multiplier estimates at each step into the function to be minimized. It also tends to yield less ill conditioned subproblems than does the log barrier approach, and it dispenses with the need for iterates to stay strictly feasible with respect to the inequality constraints. The quadratic penalty function $Q(x; \mu)$ penalizes constraint violations by squaring the infeasibilities and scaling them by $1/(2\mu)$, however, the approximate minimizers x_k of $Q(x; \mu_k)$ do not quite satisfy the feasibility conditions $d(x) = 0$. Instead, they are perturbed slightly to approximately satisfy

$$d(x_k) = -\mu_k \lambda^*. \tag{2.14}$$

To be sure, this perturbation vanishes as $\mu_k \to 0$. We can alter the function $Q(x; \mu_k)$ to avoid this systematic perturbation, that is, to make the approximate minimizers more nearly satisfy the equality constraints $d(x) = 0$. By doing so, we may avoid the need to decrease μ to zero, and thereby avoid the ill conditioning and numerical problems associated with $Q(x; \mu)$ for small values of this penalty parameter.

The augmented Lagrangian function $\mathcal{L}_A(x, \lambda, \mu)$ for the equality constrained problem achieves these goals by including an explicit estimate of the Lagrange multipliers λ in the objective.

$$\mathcal{L}_A(x, \lambda, \mu) = f(x) - \sum_{i=1}^{q} \lambda_i d_i(x) + \frac{1}{2\mu} \sum_{i=1}^{q} d_i^2(x) \tag{2.15}$$

We see that the augmented Lagrangian differs from the (standard) Lagrangian, in Eq (2.8), by the presence of the squared terms,

while it differs from the quadratic penalty function, in Eq (2.10), in the presence of the summation term involving λ. In this sense, it is a combination of the Lagrangian and quadratic penalty functions. We also have to update λ, so that they approximate λ^*, which is the λ that minimizes $\mathcal{L}_A(x, \lambda, \mu)$. For that we can use the following equation:

$$\lambda_i^{k+1} = \lambda_i^k - \frac{d_i(x_k)}{\mu_k}, \quad i = 1, \ldots, q \tag{2.16}$$

To understand that result, consider the problem given by (2.4). Its KKT conditions are given by:

$$\nabla f(x^*) + \sum_{i=1}^{q} \lambda_i^* \nabla d_i(x^*) = 0 \tag{2.17}$$

Now if we take the KKT condition of the problem defined by (2.15) we get:

$$\nabla f(\bar{x}) + \sum_{i=1}^{q} \nabla d_i(\bar{x})(\lambda_i - \frac{d_i(\bar{x})}{\mu}) = 0 \tag{2.18}$$

The problems (2.17) and (2.18) are equivalent, we can replace the term $\lambda_i - \frac{d_i(\bar{x})}{\mu}$ by $\bar{\lambda_i}$ and get the same condition as the previous problem, so

$$\bar{x} \to x^* \tag{2.19a}$$

$$\bar{\lambda} \to \lambda^* \tag{2.19b}$$

$$\lambda_i^{k+1} = \lambda_i^k - \frac{d_i(\bar{x})}{\mu_k} \tag{2.19c}$$

which is exactly Equation (2.16) with $\bar{x} = x_k$.

When the problem formulation contains general inequality constraints, as in formulation (2.5), we can convert it to a problem with equality constraints and bound constraints by introducing slack variables s and replacing the inequalities $c(x) \geq 0$, by:

$$c(x) - s = 0, \quad s \geq 0 \tag{2.20}$$

By defining the augmented Lagrangian in terms of the constraints $c(x) - s = 0$ and applying the bound constraints $s \geq 0$ explicitly, we obtain the following subproblem to be solved at iteration k of

Algorithm 3: Augmented Lagrangian

input: starting parameters $\mu_0 > 0$, tolerance $\tau > 0$, starting point x_0^s and λ^0

for $k := 0, 1, 2, \ldots$ **do**

- **find** an approximate minimizer x_k of $\mathcal{L}_A(:; \lambda^k, \mu_k)$ starting at x_k^s
- **if** $||\nabla_x \mathcal{L}_A(x; \lambda^k; \mu_k)|| \leq \tau$ **then**
  - **stop** with an approximate solution x_k;
- **update** Lagrange multipliers using (2.16) to obtain λ^{k+1}
- **choose** the next penalty factor: $\mu_{k+1} \in (0, \mu_k)$;
- **update** the starting point for the next iteration to $x_{k+1}^s = x_k$

output: x_k^s

Algorithm 3.

$$\min_{x,s} f(x) - \sum_{i=1}^{p} \lambda_i^k (c_i(x) - s_i) + \frac{1}{2\mu} \sum_{i=1}^{p} (c_i(x) - s_i)^2 \tag{2.21a}$$

subject to

$$s \geq 0 \tag{2.21b}$$

2.1.2 Gradient Projection

Gradient projection methods use a feasible direction obtained by solving a subproblem with quadratic cost. While the subproblem may be more complex, the resulting convergence rate is typically better [8]. Gradient projection methods for minimizing a continuously differentiable mapping $f : \mathbb{R}^n \to \mathbb{R}$ on a nonempty closed convex set $\Omega \in \mathbb{R}^n$ were originally proposed in [9, 10]. It is helpful to study the general problem

$$\min\{f(x) : x \in \Omega\}, \tag{2.22}$$

because it clarifies the underlying structure of the algorithms. Most of the current interest in projected gradient has been concerned with the case where Ω is defined by the bound constraints:

$$\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\} \tag{2.23}$$

Given an inner product norm $\|\cdot\|$ and a nonempty closed convex set Ω, the projection into Ω is the mapping $P : \mathbb{R}^n \to \Omega$ defined by $P(x)$.

$$P(x) = \arg\min\{\|z - x\| : z \in \Omega\}. \tag{2.24}$$

Given the projection P into Ω, the gradient projection algorithm is defined by

$$x_{k+1} = P(x_k - \alpha_k \nabla f(x_k)) \tag{2.25}$$

where $\alpha_k > 0$ is the step, and ∇f is the gradient of f with respect to the inner product associated with the norm $\|\cdot\|$.

The simplest gradient projection method is a feasible direction method of the form

$$x^{k+1} = x^k + \alpha^k(\bar{x}^k - x^k) \tag{2.26}$$

where

$$\bar{x}^k = [x^k - s^k \nabla f(x^k)]^+. \tag{2.27}$$

In this equation, $[\cdot]^+$ denotes projection on the set Ω, $\alpha^k \in (0, 1]$ is a setpsize, and s^k is a positive scalar. So, to obtain the vector $\bar{x}^k$ we take a step $-s^k \nabla f(x^k)$ along the negative gradient, then project the result $x^k - s^k \nabla f(x^k)$ on Ω. Finally, we take a step along the feasible direction $(\bar{x}^k - x^k)$ using the stepsize α^k.

The scalar s^k can also be view as a stepsize. When $\alpha^k = 1$ for all k, then $x^{k+1} = \bar{x}^k$ and the method becomes

$$x^{k+1} = [x^k - s^k \nabla f(x^k)]^+. \tag{2.28}$$

If $x^k - s^k \nabla f(x^k)$ is feasible, the gradient projection iteration becomes an unconstrained steepest descent iteration. Note that we have $x^* = [x^* - s\nabla f(x^*)]^+$ for all $s > 0$ if and only if x^* is stationary. Thus the method only stops if and only if it encounters a stationary point.

In order for the method to make practical sense, the projection operation need to be simple. This happens if Ω has a simple structure. For example

$$\Omega = \{x | \alpha_i \leq x_i \leq \beta_i, i = 1, \ldots, n\} \tag{2.29}$$

the i-th coordinate of the projection of a vector x is given by

$$[x]^+ = \begin{cases} \alpha_i & \text{if } x_i \leq \alpha_i \\ \beta_i & \text{if } x_i \geq \beta_i \\ x_i & \text{otherwise} \end{cases} \tag{2.30}$$

There are several stepsize selection procedures in the gradient projection method. The *Limited Minimization Rule* sets s^k constant; that is

$$s^k = s : \text{constant}, \quad k = 0, 1, \ldots \tag{2.31}$$

and α^k is chosen by minimization over $[0, 1]$; that is

$$f(x^k + \alpha^k(\bar{x}^k - x^k)) = \min_{\alpha \in [0,1]} f(x^k + \alpha(\bar{x}^k - x^k)). \tag{2.32}$$

The *Armijo Rule Along the Feasible Direction* uses a constant s^k as depicted in Equation (2.31), but α^k is chosen by the Armijo rule over the interval $[0, 1]$. In particular, for fixed scalars β and $\sigma \in (0, 1)$, we set $\alpha^k = \beta^{m_k}$, where m_k is the smallest nonnegative integer m for which

$$f(x^k) - f(x^k + \beta^m(\bar{x}^k - x^k)) \geq -\sigma\beta^m \nabla f(x^k)'(\bar{x}^k - x^k). \tag{2.33}$$

A variation of the Armijo rule is the *Armijo Rule Along the Projection Arc*. Here the stepsize α^k is fixed at 1.

$$\alpha^k = 1, \quad k = 0, 1, \ldots \tag{2.34}$$

and the stepsize s^k is determined by successive reduction until an Armijo-like inequality is satisfied. This means that x^{k+1} is determined by ther Armijo-like search on the projection arc

$$\{x^k(s) | s > 0\} \tag{2.35}$$

where, for all $s > 0$, $x^k(s)$ is defined by

$$x^k(s) = [x^k - s\nabla f(x^k)]^+. \tag{2.36}$$

In particular, for fixed scalars with $\bar{s} > 0$, β and $\sigma \in (0, 1)$, we set $s^k = \beta^{m_k}\bar{s}$, where m_k is the smallest nonnegative integer m for which

$$f(x^k) - f(x^k(\beta^m\bar{s})) \geq -\sigma\beta^m \nabla f(x^k)'(x^k - x^k(\beta^m\bar{s})) \tag{2.37}$$

We can also use both parameters constant for all k, with

$$s^k = s \quad \text{constant} \tag{2.38a}$$

$$\alpha^k = 1 \tag{2.38b}$$

To check the convergence rate of the method we assume f to be a quadratic function

$$f(x) = \frac{1}{2}x'Qx - b'x \tag{2.39}$$

where Q is positive definite, and let x^* denote the unique minimum of f over Ω. We consider the case of constant setpsize ($\alpha^k = 1$ and $s^k = s$ for all k), then

$$\begin{aligned} ||x^{k+1} - x^*|| &= ||[x^k - s\nabla f(x^k)]^+ - [x^* - s\nabla f(x^*)]^+|| \\ &\leq ||x^k - s\nabla f(x^k) - (x^* - s\nabla f(x^*))|| \\ &= ||(I - sQ)(x^k - x^*)|| \\ &= \max\{|1 - sm|, |1 - sM|\}||x^k - x^*|| \end{aligned} \tag{2.40}$$

where m and M are the minimum and maximum Eigenvalues of Q. We conclude that the gradient projection method suffers from a slow convergence rate, since Equation (2.40) is precisely the rate of convergence estimate obtained for the unconstrained steepest descent method with constant setpsize [8].

2.2 MODEL PREDICTIVE CONTROL

Model predictive control (MPC) is an advanced method of process control that has been in use in the process industries, chemical plants and oil refineries since the 1980s. The main reasons for its popularity are the constraint-handling capabilities and the easy extension to multivariable processes. From the academic side the interest in MPC mainly came from the field of self-tuning control. Model predictive controllers rely on dynamic models of the process, most often linear empirical models obtained by system identification. The main advantage of MPC is the fact that it allows the current timeslot to be optimized, while keeping future timeslots in account. This is achieved by optimizing over a finite time-horizon, but only implementing the current timeslot. MPC has the ability to anticipate future events and can take control actions accordingly. PID and LQR controllers do not have this predictive ability. MPC is based on iterative, finite-horizon optimization of a plant model.

At time t the current plant state is sampled and a cost minimizing control strategy is computed (via a numerical minimization algorithm) for a relatively short time horizon in the future: $[t, t+T]$.

An online calculation is used to find a cost-minimizing control strategy until time $t + T$. Only the first step of the control strategy is implemented, then the plant state is sampled again and the calculations are repeated starting from the new current state, yielding a new control and new predicted state path. As all controller design methodologies, MPC also has its drawbacks:

- A detailed process model is required. This means that either one must have a good insight in the physical behavior of the plant or system identification methods have to be applied to obtain a good model.
- The methodology is open, and many variations have led to a large number of MPC methods.
- Although, in practice, stability and robustness are easily obtained by accurate tuning, theoretical analysis of stability and robustness properties are difficult to derive.

Model predictive control techniques give flexibility in the operation of unit processes by adjustment of the control structure on the basis of given controller objectives, specified operating constraints and actual operating conditions. Model predictive control techniques allow for adjustment of controlled process characteristics in accordance with actual demands.

The model predictive control techniques together with in-line model-based optimization techniques enable the operation of unit processes so that undesired dynamic behavior are compensated for and that the process outputs approximate the desired behavior. The compensation of non-desired dynamic behavior is restricted by the internal mechanisms of the process and by limitations stemming from the controller. MPC is rather a methodology than a single technique. The difference in the various methods is mainly the way the problem is translated into a mathematical formulation, so that the problem becomes solvable in the limited time interval available for calculation of adequate process manipulations in response to external influences on the process behavior (disturbances). However, in all methods, five important items are part of the design procedure:

- Process model and disturbance model.
- Performance index.
- Constraints.

- Optimization.
- Receding horizon principle.

Linear models are mostly chosen to represent processes. On the basis of the model a prediction of the process signals over a specified horizon is made. The state space description is the most general system description, it is well suited for multivariable systems, while still providing a compact model description. The computations are usually well conditioned and the algorithms easy to implement.

An important difference between Model Predictive control (MPC) and PID-kind design-methods is the explicit use of a model. This aspect is both the advantage and the disadvantage of MPC. The advantage is that the behavior of the controller can be studied in detail, simulations can be made and possible failures in plant or controller can be well-detected. The disadvantage is that a detailed study of the plant behavior has to be done before the actual MPC-design can be started. Most of work for MPC is in modelling and identification of the plant. The models applied in MPC serve two purposes:

- Prediction of expected future process output behavior on the basis of inputs and known disturbances applied to the process in the past.
- Calculation of the next process input signal that minimizes the controller objective function.

The models required for these tasks do not necessarily have to be the same. The model applied for prediction may differ from the model applied for calculation of the next control action. In practice though both models are almost always chosen to be the same [11].

An example of a non-linear performance-index or cost function for optimization is given by:

$$J = \sum_{i=1}^{N} w_{x_i}(r_i - x_i)^2 + \sum_{i=1}^{N} w_{u_i} u_i^2 \tag{2.41}$$

where x_i is the i-th controlled variable, r_i is the i-th reference variable, u_i is the i-th manipulated variable, w_{x_i} is the weighting coefficient reflecting the relative importance of x_i and w_{u_i} is the weighting coefficient penalizing relative big changes in u_i. In practice industrial processes are subject to constraints.

Specific signals must not violate specified bounds due to safety limitations, environmental regulations, consumer specifications and physical restrictions such as minimum and/or maximum temperature, pressure, level limits in reactor tanks and flows in pipes. Careful tuning of the controller parameters may keep these values away from the bounds. However, because of economical motives, the control system should drive the process towards the constraints as close as possible, without violating them: Closer to limits in general often means closer to maximum profit. Therefore, predictive control employs a more direct approach by modifying the optimal unconstrained solution in such a way that constraints are not violated. This can be done using optimization techniques such as linear programming (LP) or quadratic programming (QP) techniques.

In most cases the constraints can be translated in bounds on control, state or output signals. An optimization algorithm will be applied to compute a sequence of future control signals that minimizes the performance index subject to the given constraints. For linear models with linear constraints and a quadratic performance index the solution can be found using quadratic programming algorithms. In some cases, the optimization problem will have an empty solution set, so the problem is not feasible. In that case we will have to relax one or more of the constraints to find a solution leading to an acceptable control signal.

Predictive control uses the receding horizon principle. This means that after computation of the optimal control sequence, only the first control sample will be implemented, subsequently the horizon is shifted one sample and the optimization is restarted with new measurements. At time k the future control sequence given by $\{u(k|k), \ldots, u(k+N-1|k)\}$ is optimized such that the performance-index J is minimized subject to constraints. At time k the first element of the optimal sequence $(u(k) = u(k|k))$ is applied to the real process. At the next time instant the horizon is shifted and a new optimization at time $k+1$ is solved.

Because of the computational complexity of the centralized MPC, the application area of this type of control is restricted to only relatively small-scale MIMO systems. A distributed approach (DMPC) seems to be the only solution for large-scale dynamically coupled systems. The DMPC is structured as a decentralized law, with a local controller for each subsystem. To achieve better closed-loop control performance, some level of communication may be established between the different controllers, which leads to distributed model predictive control. With respect to the DMPC algo-

rithms available in the literature, a classification can be made according to the topology of the communication network, the different communication protocols used by the local controllers, and the cost function considered in the local controller optimization problem [4].

Within the wide set of distributed MPC algorithms proposed in the literature, a classification can be made depending on the topology of the communication network. Specifically, the following cases can be considered:

- information is transmitted (and received) from any local regulator to all the others (fully connected algorithms);
- information is transmitted (and received) from any local regulator to a given subset of the others (partially connected algorithms).

A partially connected information structure can be convenient in the case of large scale systems made by a great number of loosely connected subsystems. In these cases, restricting the information exchange among directly interacting subsystems produces a negligible performance deterioration. The exchange of information among local regulators can be made according to different protocols:

- information is transmitted (and received) by the local regulators only once within each sampling time (noniterative algorithms);
- information can be transmitted (and received) by the local regulators many times within the sampling time (iterative algorithms).

It is apparent that the amount of information available to the local regulators with iterative algorithms is higher, so that an overall iterative procedure can be set-up to reach a global consensus on the actions to be taken within the sampling interval. To this regard however, a further classification has to be considered:

- distributed algorithms where each local regulator minimizes a local performance index (independent algorithms);
- distributed algorithms where each local regulator minimizes a global cost function (cooperating algorithms).

2.3 SUMMARY

In this chapter we reviewed some concepts of optimization. We showed three classes of optimization problems, the unconstrained optimization, equality contrained optimization and inequality constrained optimization. Futhermore we expanded those cases to a generalized optimization problem and explored optimality conditions known as the KKT-conditions.

With this review we can advance and showcase algorithm to solve problems of concern. We demonstrated three differents algorithms and explained their own particularity, similarities and distinctions between each other, especially the Augmented Lagragian, which is the focus of this dissertation. Shortly thereafter the gradient projection method was introduced, a powerful algorithm to solve optimization problems.

Finally, a review of Model Predictive Control was presented, developing its fundamentals which are essential to the development of this dissertation.

3 PROBLEM DEFINITION

In this chapter we show how to formulate a problem into a Linear Dynamic Network, which has many advantages, such as the simplicity in its decomposition and as such the simplicity in creating a distributed problem. This chapter also presents how to define a MPC problem as a LDN.

3.1 LINEAR DYNAMIC NETWORK (LDN)

A linear dynamic network is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, \ldots, n\}$ is the vertices and $\mathcal{E} \subseteq V \times V$ is the arcs, whose nodes model subsystems and whose arcs represent the direct influence between subsystems. The set $I(i) = \{j : (j, i) \in E\} \cup \{i\}$ is the *input neighborhood* of subsystem i which contains the subsystems that affect its dynamics. The LDN depicted in Figure 3.1 has $n = 6$ nodes where $I(1) = \{1\}$ and $I(3) = \{2, 3, 6\}$. The state of subsystem i are $\mathbf{x}_i \in \mathbb{R}^{n_i}$ while its controls are $\mathbf{u}_i \in \mathbb{R}^{p_i}$. The state of subsystem i is governed by discrete-time linear dynamics:

$$\mathbf{x}_i(t+1) = \mathbf{A}_i \mathbf{x}_i(t) + \sum_{j \in I(i)} \mathbf{B}_{i,j} \mathbf{u}_j(t). \tag{3.1}$$

This is a simplified form of the dynamics treated in [12, 5, 7] which allow for the state of the upstream subsystems to affect downstream subsystems. Because such an extension will require the distributed subsystems to consider more distant subsystems depending on the length of the prediction horizon, this work assumes the simplified dynamics to keep the presentation simple. The LDN with dynamic equation (3.1) can model urban traffic networks [1, 13] with the state of a subsystem (intersection) being the incoming vehicles queues, which evolve depending on green-time signals at the downstream and upstream subsystems.

Given the state of subsystem i at time t and the future control signals in the input neighborhood, the state of subsystem i over the time horizon is given by:

$$\widehat{\mathbf{x}}_i = \widehat{\mathbf{A}}_i \mathbf{x}_i(t) + \sum_{j \in I(i)} \widehat{\mathbf{B}}_{i,j} \widehat{\mathbf{u}}_j \tag{3.2}$$

where $\widehat{\mathbf{x}}_i = (\mathbf{x}_i(t+1), \ldots, \mathbf{x}_i(t+T))$, $\widehat{\mathbf{u}}_i = (\mathbf{u}_i(t), \ldots, \mathbf{u}_i(t+T-1))$ are vectors, and $\widehat{\mathbf{A}}_i$ and $\widehat{\mathbf{B}}_{i,j}$ are matrices obtained from the

dynamics (3.1) in a straightforward manner [14]. Observe that the state of subsystem i at instant k is a function of control and state signals from before k.

$$x_i(k) = A_i^k x_i(0) + \sum_{l=1}^{k} \sum_{i \in I\{i\}} A_i^{l-1} B_{i,j} u_i(k-l). \tag{3.3}$$

Using the previous equation, $\widehat{\mathbf{A}}_i$ and $\widehat{\mathbf{B}}_{i,j}$ are given by:

$$\widehat{\mathbf{A}}_i = \begin{bmatrix} A_i \\ A_i^2 \\ \vdots \\ A_i^T \end{bmatrix}$$

,

$$\widehat{\mathbf{B}}_{i,j} = \begin{bmatrix} B_{i,j} & 0 & 0 & \vdots & 0 \\ A_i B_{i,j} & B_{i,j} & 0 & \vdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_i^{T-1} B_{i,j} & A_i^{T-2} B_{i,j} & A_i^{T-3} B_{i,j} & \dots & B_{i,j} \end{bmatrix}$$

.

3.2 MODEL PREDICTIVE CONTROL OF LDN

Given the network state $\mathbf{x}(t) = (\mathbf{x}_1, \dots, \mathbf{x}_n)(t)$ at time t, we consider the MPC regulation of an LDN which solves the following quadratic program at each sample time:

$$P: \min_u \sum_{i=1}^{n} \sum_{k=t}^{t+T-1} \frac{1}{2}(\mathbf{x}_i(k+1)' \mathbf{Q}_i \mathbf{x}_i(k+1)) + \frac{1}{2}(\mathbf{u}_i(k)' \mathbf{R}_i \mathbf{u}_i(k)) \tag{3.4a}$$

and subject to:

for all $i \in V$, $k = t, \dots, t+T-1$:

$$\mathbf{x}_i(k+1) = \mathbf{A}_i \mathbf{x}_i(k) + \sum_{j \in I(i)} \mathbf{B}_{i,j} \mathbf{u}_j(k) \tag{3.4b}$$

$$\mathbf{X}_i \mathbf{x}_i(k+1) \leq \mathbf{x}_i^{\max} \tag{3.4c}$$

$$\sum_{j \in I(i)} \mathbf{U}_{i,j} \mathbf{u}_j(k) \leq \mathbf{u}_i^{\max} \tag{3.4d}$$

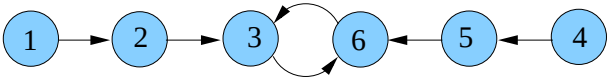

Figure 3.1: Linear dynamic network..

where T is the length of the prediction horizon, matrices $\mathbf{Q}_i = \mathbf{Q}_i' \succeq 0$, $\mathbf{R}_i = \mathbf{R}_i' \succ 0$, matrix $\mathbf{X}_i$ and vector $\mathbf{x}_i^{\max}$ define state constraints, and matrix $\mathbf{U}_{i,j}$ and vector $\mathbf{u}_i^{\max}$ define input constraints. P is solved at each sample time t, but only $\mathbf{u}_i(t)$ is implemented for the time interval $[t, t+1)\tau$, with τ being the sample interval. At the next sample time, P is solved from time $t+1$ to $t+T+1$ and the process is repeated. Problem P can be recast as:

$$P: \ \min \quad f(\widehat{\mathbf{u}}) = \frac{1}{2}\sum_{i\in V}\sum_{j\in I(i)}\sum_{l\in I(i)} \widehat{\mathbf{u}}_j'\widehat{\mathbf{H}}_{i,j,l}\widehat{\mathbf{u}}_l + \sum_{i\in V}\sum_{j\in I(i)} \widehat{\mathbf{g}}_{i,j}'\widehat{\mathbf{u}}_j + \sum_{i\in V}\widehat{c}_i \tag{3.5a}$$

while being subject to:

$$\sum_{j\in I(i)} \widehat{\mathbf{X}}_{i,j}\widehat{\mathbf{x}}_j \leq \widehat{\mathbf{x}}_i^{\text{bd}}, \ \forall i \in V \tag{3.5b}$$

$$\sum_{j\in I(i)} \widehat{\mathbf{U}}_{i,j}\widehat{\mathbf{u}}_j \leq \widehat{\mathbf{u}}_i^{\text{bd}}, \ \forall i \in V \tag{3.5c}$$

where: $\widehat{\mathbf{u}} = (\widehat{\mathbf{u}}_i : i \in V)$ collects the predictions of all control signals; $\widehat{\mathbf{H}}_{i,j,l}$ is a matrix given by $\widehat{B}_{i,j}'\widehat{Q}_i\widehat{B}_{i,j} + \widehat{R}_i$, $\widehat{\mathbf{g}}_{i,j}$ is a vector given by $\widehat{B}_{i,j}'\widehat{Q}_i\widehat{A}_i x_i(0)$, and $\widehat{c}_i$ is a constant, all obtained from the structure of P as detailed in [14, 7]; $\widehat{\mathbf{X}}_{i,j} = \widehat{\mathbf{X}}_i\widehat{\mathbf{B}}_{i,j}$ and $\widehat{\mathbf{X}}_i$ is block-diagonal with T blocks of matrix $\mathbf{X}_i$; $\widehat{\mathbf{x}}_i^{\text{bd}} = \widehat{\mathbf{x}}_i^{\max} - \widehat{\mathbf{X}}_i\widehat{\mathbf{A}}_i\mathbf{x}_i(t)$ and $\widehat{\mathbf{x}}_i^{\max}$ is a vector with T stacked copies of $\mathbf{x}_i^{\max}$; $\widehat{\mathbf{U}}_{i,j}$ is block-diagonal containing T blocks of matrix $\mathbf{U}_{i,j}$; and $\widehat{\mathbf{u}}_i^{\text{bd}}$ is a vector with T stacked copies of $\mathbf{u}_i^{\max}$.

Problem P is expressed in a compact and equivalent form:

$$P:\ \min\ \ f(\widehat{\mathbf{u}}) = \frac{1}{2}\sum_{i\in V}\sum_{j\in I(i)}\sum_{l\in I(i)}\widehat{\mathbf{u}}_j'\widehat{\mathbf{H}}_{i,j,l}\widehat{\mathbf{u}}_l + \sum_{i\in V}\sum_{j\in I(i)}\widehat{\mathbf{g}}_{i,j}'\widehat{\mathbf{u}}_j + \sum_{i\in V}\widehat{c}_i \quad (3.6a)$$

while being subject to:

$$\sum_{j\in I(i)}\widehat{\mathbf{Z}}_{i,j}\widehat{\mathbf{u}}_j + \widehat{\mathbf{s}}_i = \widehat{\mathbf{z}}_i^{\text{bd}},\ \forall i\in V \quad (3.6b)$$

$$\widehat{\mathbf{s}}_i \geq \mathbf{0},\ \forall i\in V \quad (3.6c)$$

where $\widehat{\mathbf{Z}}_{i,j} = \left[\widehat{\mathbf{X}}_{i,j}'\ \widehat{\mathbf{U}}_{i,j}'\right]'$, $\widehat{\mathbf{z}}_i^{\text{bd}} = \left[\widehat{\mathbf{x}}_i^{\text{bd}\prime}\ \widehat{\mathbf{u}}_i^{\text{bd}\prime}\right]'$, and $\widehat{\mathbf{s}}_i$ is the vector of slack variables. Notice that P given by (3.4) is DCCC, whereas the forms given by (3.5) and (3.6) are CCCC [5].

Remark. *$f(\widehat{\mathbf{u}})$ is strictly convex.*

Remark. *P is convex.*

3.3 PROBLEM DECOMPOSITION

This work is about decomposing P into a set $\{P_i : i\in V\} = \{P_i\}$ of subproblems which are then solved by a network of distributed agents, one for each subsystem. For any agent i, let:

- $O(i) = \{j : i\in I(j)\}$ be the *output neighborhood*, the subsystems affected by subsystem i;
- $C(i) = \{j : \exists\, l\neq i,j$ such that $i,j\in I(l)\}\setminus(I(i)\cup O(i))$ be the *indirect neighborhood*, the subsystems j not coupled with subsystem i which affect a subsystem l affected by i;
- $N(i) = (I(i)\cup O(i)\cup C(i))\setminus\{i\}$ be the *neighborhood*;
- $N^+(i) = N(i)\cup\{i\}$ be the *extended neighborhood*.

For the LDN of Figure 3.1, subsystem 2 has $I(2)=\{1,2\}$, $O(2)=\{2,3\}$, $C(2)=\{6\}$, $N(2)=\{1,3,6\}$, and $N^+(2)=\{1,2,3,6\}$. From the view of agent i, the network variables are split in:

- *local variables:* the vector $\widehat{\mathbf{u}}_i$ of variables exclusively defined by agent i;
- *neighborhood variables:* the vector $\widehat{\mathbf{y}}_i = (\widehat{\mathbf{u}}_j : j \in N(i))$ of variables set exclusively by the agents in the neighborhood;
- *shared variables:* the vector $\widehat{\mathbf{z}}_i = (\widehat{\mathbf{s}}_j : j \in O(i))$ with the slack variables of the constraints in the output neighborhood that are affected by agent i's decisions;
- *remote variables:* $\widehat{\mathbf{r}}_i = (\widehat{\mathbf{u}}_j : j \notin N^+(i), \widehat{\mathbf{s}}_j : j \notin O(i))$ with all of the other variables.

Let $\widehat{\mathbf{v}} = (\widehat{\mathbf{u}}, \widehat{\mathbf{s}})$ collect all of the decision variables over the prediction horizon. This agent view induces a *perfect problem decomposition*, whereby $\widehat{\mathbf{v}} = (\widehat{\mathbf{u}}_i, \widehat{\mathbf{y}}_i, \widehat{\mathbf{z}}_i, \widehat{\mathbf{r}}_i)$ and P_i is:

$$P_i(\widehat{\mathbf{y}}_i): \min_{\widehat{\mathbf{u}}_i, \widehat{\mathbf{z}}_i} f_i(\widehat{\mathbf{v}}_i, \widehat{\mathbf{z}}_i; \widehat{\mathbf{y}}_i) = \frac{1}{2}\widehat{\mathbf{u}}_i'\widehat{\mathbf{H}}_i\widehat{\mathbf{u}}_i + \widehat{\mathbf{g}}_i'\widehat{\mathbf{u}}_i + \widehat{c}_i \tag{3.7a}$$

$$\text{s.t.}: \sum_{l \in I(j)} \widehat{\mathbf{Z}}_{j,l}\widehat{\mathbf{u}}_l - \widehat{\mathbf{z}}_j^{\text{bd}} + \widehat{\mathbf{s}}_j = \mathbf{0},\ j \in O(i) \tag{3.7b}$$

$$\widehat{\mathbf{s}}_j \geq \mathbf{0},\ j \in O(i) \tag{3.7c}$$

where $\widehat{\mathbf{g}}_i = \sum_{j \in O(i)} \widehat{\mathbf{g}}_{j,i} + \frac{1}{2}\sum_{j \in O(i)} \sum_{l \in I(j) \setminus \{i\}} (\widehat{\mathbf{H}}'_{j,l,i} + \widehat{\mathbf{H}}_{j,i,l})\widehat{\mathbf{u}}_l$ and $\widehat{\mathbf{H}}_i = \sum_{j \in O(i)} \widehat{\mathbf{H}}_{j,i,i}$. The direct optimization of the problem set $\{P_i\}$ will not be effective: any agent i attempting to solve $P_i(\widehat{\mathbf{y}}_i)$ may not be able to reduce the objective value due to the coupling constraints (3.7b)-(3.7c). Instead, a series of sign-constrained problems approximating $\{P_i\}$ will be solved with the Lagrangian dual method [8, 15]. The decomposition is convenient to establish the relationship between the centralized problem P and a consistent set of subproblems $\{P_i\}$. However, the problems P_i can be defined directly whereby P is the result of the composition of $\{P_i\}$.

3.4 SUMMARY

This chapter presented the development of a Linear Dynamic Network, how it can be used to represent linear systems with a network structure, so that it can better show how the system works and how the subsystems interact with eachother. With the knowledge previously acquired it is shown how to apply the same control methods for linear systems, the MPC seen in the previous chapter

in specific, which can be accomplished with the developments and tools presentend in Section 3.2. To end the chapter it is presented one of the great advantages of LDN, the decomposition of the problem, in Section 3.3 that feature is explored, introducing key concepts, such as neighborhood, and the tools needed for it.

4 DISTRIBUTED DUAL OPTIMIZATION ALGORITHM

The distributed algorithm for solving $\{P_i\}$ follows the augmented Lagrangian method. Lagrange multipliers and penalty factors for the constraints define a signed-constrained approximation which is solved by the subsystem agents with a distributed gradient-projection algorithm.

4.1 AUGMENTED LAGRANGIAN

The augmented Lagrangian is not the same as the method of Lagrange multipliers. Viewed differently, the unconstrained objective is the Lagrangian of the constrained problem, with an additional penalty term (the augmentation). These methods were used in structural optimization. The method was also studied by Dimitri Bertsekas, notably in his book [16], together with extensions involving nonquadratic regularization functions, such as entropic regularization, which gives rise to the "exponential method of multipliers," a method that handles inequality constraints with a twice differentiable augmented Lagrangian function. Since the 1970s, sequential quadratic programming (SQP) and interior point methods (IPM) have had increasing attention, in part because they more easily use sparse matrix subroutines from numerical software libraries, and in part because IPMs have proven complexity results via the theory of self-concordant functions. The augmented Lagrangian method was rejuvenated by the optimization systems LANCELOT and AMPL, which allowed sparse matrix techniques to be used on seemingly dense but "partially separable" problems. The augmented Lagrangian method is generally preferred to the quadratic penalty method since there is little extra computational cost and the parameter μ need not go to infinity, thus avoiding ill-conditioning.

The Lagrange multiplier vector $\widehat{\boldsymbol{\lambda}}_i$ and penalty factor $\mu > 0$ are associated with the constraint (3.6b), for each $i \in V$, to obtain

the augmented Lagrangian of P:

$$\mathcal{L}(\widehat{\mathbf{v}};\widehat{\boldsymbol{\lambda}},\mu) = \frac{1}{2}\sum_{i\in V}\sum_{j\in I(i)}\sum_{l\in I(i)}\widehat{\mathbf{u}}_j'\widehat{\mathbf{H}}_{i,j,l}\widehat{\mathbf{u}}_l + \sum_{i\in V}\sum_{j\in I(i)}\widehat{\mathbf{g}}_{i,j}'\widehat{\mathbf{u}}_j + \sum_{i\in V}\widehat{\boldsymbol{\lambda}}_i'\left(\sum_{j\in I(i)}\widehat{\mathbf{Z}}_{i,j}\widehat{\mathbf{u}}_j - \widehat{\mathbf{z}}_i^{\text{bd}} + \widehat{\mathbf{s}}_i\right) + \frac{1}{2\mu}\sum_{i\in V}\|\sum_{j\in I(i)}\widehat{\mathbf{Z}}_{i,j}\widehat{\mathbf{u}}_j - \widehat{\mathbf{z}}_i^{\text{bd}} + \widehat{\mathbf{s}}_i\|^2 + \sum_{i\in V}\widehat{c}_i \quad (4.1)$$

where $\widehat{\boldsymbol{\lambda}} = (\widehat{\boldsymbol{\lambda}}_i : i \in V)$.

Remark. *$\mathcal{L}(\widehat{\mathbf{v}};\widehat{\boldsymbol{\lambda}},\mu)$ is a convex function on $\widehat{\mathbf{v}}$.*

The algorithm yields a series $(\widehat{\mathbf{v}},\widehat{\boldsymbol{\lambda}},\mu)^{(k)}$ converging to the optimal solution $\widehat{\mathbf{v}}^\star$ of P and the corresponding optimal Lagrange vector $\widehat{\boldsymbol{\lambda}}^\star$. Let $\widehat{\boldsymbol{\lambda}}^{(k)}$ be the Lagrange vector, $\mu^{(k)}$ be the penalty factor, and $\widehat{\mathbf{v}}^{(k)}$ be the minimizer of $\mathcal{L}$ at iteration k. From [15, pp. 513-515], an analysis of the equation $\nabla_{\widehat{\mathbf{v}}}\mathcal{L} = \mathbf{0}$ leads to an approximation of the optimal Lagrange multipliers:

$$\widehat{\boldsymbol{\lambda}}_i^\star \approx \widehat{\boldsymbol{\lambda}}_i^{(k)} + \left(\sum_{j\in I(i)}\widehat{\mathbf{Z}}_{i,j}\widehat{\mathbf{u}}_j - \widehat{\mathbf{z}}_i^{\text{bd}}\right)/\mu^{(k)}.$$

By rearranging this equation, an approximation is obtained for the penalty factor:

$$\mu^{(k)}(\widehat{\boldsymbol{\lambda}}_i^\star - \widehat{\boldsymbol{\lambda}}_i^{(k)}) \approx \left(\sum_{j\in I(i)}\widehat{\mathbf{Z}}_{i,j}\widehat{\mathbf{u}}_j - \widehat{\mathbf{z}}_i^{\text{bd}}\right).$$

When $\widehat{\boldsymbol{\lambda}}^{(k)}$ is close to $\widehat{\boldsymbol{\lambda}}^\star$, this equation shows that the infeasibility of the current iterate $\widehat{\mathbf{v}}^{(k)}$ is much smaller than $\mu^{(k)}$, not just proportional to $\mu^{(k)}$ as in penalty algorithms.

These ideas lead to a dual method for solving problem P: given $(\widehat{\boldsymbol{\lambda}}^{(k)},\mu^{(k)})$, find the minimizer $\widehat{\mathbf{v}}^{(k)}$ of $\mathcal{L}$; update the Lagrange multipliers using the approximations above and reduce the penalty factor to obtain $(\widehat{\boldsymbol{\lambda}}^{(k+1)},\mu^{(k+1)})$; repeat these steps until conver-

gence. Algorithm 4 formalizes the dual method, whose most demanding step is the Lagrangian subproblem:

$$LM: \min_{\widehat{\mathbf{v}}} \ \mathcal{L}(\widehat{\mathbf{v}}; \widehat{\boldsymbol{\lambda}}, \mu) \tag{4.2a}$$

$$\text{s.t.}: \widehat{\mathbf{s}} \geq \mathbf{0}. \tag{4.2b}$$

Algorithm 4: Augmented Lagrangian Algorithm

input: starting parameters $\mu^{(0)} > 0$, $\widehat{\boldsymbol{\lambda}}^{(0)}$, and $\widehat{\mathbf{v}}^{\mathrm{s}}$;
for $k := 0, 1, 2, \ldots$ **do**
  find an approximate solution $\widehat{\mathbf{v}}^{(k)} = (\widehat{\mathbf{u}}, \widehat{\mathbf{s}})^{(k)}$ to the problem starting at $\widehat{\mathbf{v}}^{\mathrm{s}}$:

$$\min_{\widehat{\mathbf{v}}\,:\,\widehat{\mathbf{s}} \geq \mathbf{0}} \ \mathcal{L}(\widehat{\mathbf{v}}; \widehat{\boldsymbol{\lambda}}^{(k)}, \mu^{(k)}).$$

  if *convergence is attained* **then**
    stop with an approximate solution $\widehat{\mathbf{v}}^{(k)}$;
  update the Lagrange multipliers for all $i \in V$:

$$\widehat{\boldsymbol{\lambda}}_i^{(k+1)} := \widehat{\boldsymbol{\lambda}}_i^{(k)} + \left(\sum_{j \in I(i)} \widehat{\mathbf{Z}}_{i,j} \widehat{\mathbf{u}}_j - \widehat{\mathbf{z}}_i^{\mathrm{bd}} \right) / \mu^{(k)};$$

  choose the next penalty factor: $\mu^{(k+1)} \in (0, \mu^{(k)})$;
  update the starting point: $\widehat{\mathbf{v}}^{\mathrm{s}} := \widehat{\mathbf{v}}^{(k)}$;
output: $\widehat{\mathbf{v}}^{(k)}$

4.2 GRADIENT PROJECTION METHOD (GPM)

The gradient projection method (GPM) can solve LM as follows. Given a feasible $\widehat{\mathbf{v}}^{(r)}$ for LM at iteration r, GPM generates the next point with the iteration:

$$\widehat{\mathbf{v}}^{(r+1)} = \widehat{\mathbf{v}}^{(r)} + \alpha^{(r)} \widehat{\mathbf{d}}^{(r)} \tag{4.3}$$

where $\alpha^{(r)} \in (0, 1]$ is the step length in the direction $\widehat{\mathbf{d}}^{(r)}$ defined as:

$$\widehat{\mathbf{d}}^{(r)} = \left[\widehat{\mathbf{v}}^{(r)} - s \nabla \mathcal{L}(\widehat{\mathbf{v}}^{(r)}) \right]^{+} - \widehat{\mathbf{v}}^{(r)} \tag{4.4}$$

where $s > 0$ and $[\widehat{\mathbf{v}}]^+$ denotes the projection of $\widehat{\mathbf{v}}$ onto the feasible space $\mathcal{V} = \{\widehat{\mathbf{v}} : \widehat{\mathbf{s}} \geq \mathbf{0}\}$. Actually, the iteration counter for GPM should be (k, r) since a sequence $\{\widehat{\mathbf{v}}^{(k,r)}\}_{r=0}^{\infty}$ is produced for each iteration k, which is omitted to keep notation simple. Thus, k is the *outer* whereas r is the *inner* iteration counter. Although the parameter s can change from one inner iteration to the next, it is assumed constant and typically equal to 1 to simplify the convergence proof. The projection of a vector $\widehat{\mathbf{v}}$ onto a convex set $\mathcal{V}$ is defined as:

$$[\widehat{\mathbf{v}}]^+ = \arg\min_{\widetilde{\mathbf{v}} \in \mathcal{V}} \|\widetilde{\mathbf{v}} - \widehat{\mathbf{v}}\|^2. \tag{4.5}$$

The direction $\widehat{\mathbf{d}}^{(r)}$ defined in Eq. (4.4) is a feasible descent direction for a nonstationary[1] point $\widehat{\mathbf{v}}^{(r)}$ because $\nabla\mathcal{L}(\widehat{\mathbf{v}}^{(r)})'\widehat{\mathbf{d}}^{(r)} < 0$ and $(\widehat{\mathbf{v}}^{(r)} + \alpha\widehat{\mathbf{d}}^{(r)}) \in \mathcal{V}$ for all sufficiently small $\alpha > 0$. Notice that $\widehat{\mathbf{v}}^\star$ is optimal for LM if and only if $\widehat{\mathbf{v}}^\star = [\widehat{\mathbf{v}}^\star - s\nabla\mathcal{L}(\widehat{\mathbf{v}}^\star)]^+$ ([8, pp. 203-204]).

At each iteration r, GPM projects the gradient onto the feasible space to define $\widehat{\mathbf{d}}^{(r)}$ and then finds a suitable step $\alpha^{(r)}$. Notice that GPM is solved very efficiently for the problem at hand: $[\widehat{\mathbf{v}}]^+$ is easily computed by setting to 0 all of the negative entries of $\widehat{\mathbf{v}}$ associated with the slack variables $\widehat{\mathbf{s}}$. A popular strategy for choosing the step length is the Armijo rule which defines $\alpha^{(r)} = \gamma^{l(r)}$ where $\gamma \in (0, 1)$, $\sigma \in (0, 1)$, and $l(r) \in \{0, 1, \ldots\}$ is the least nonnegative integer for which

$$\mathcal{L}(\widehat{\mathbf{v}}^{(r)} + \gamma^{l(r)}\widehat{\mathbf{d}}^{(r)}) \leq \mathcal{L}(\widehat{\mathbf{v}}^{(r)}) + \sigma\gamma^{l(r)}\nabla\mathcal{L}(\widehat{\mathbf{v}}^{(r)})'\widehat{\mathbf{d}}^{(r)}.$$

The iterative GPM of Eq. (4.3) yields a series $\{\widehat{\mathbf{v}}^{(r)}\}$ converging to the optimum $\widehat{\mathbf{v}}^\star$, if $\widehat{\mathbf{d}}^{(r)}$ is given by gradient projection and $\alpha^{(r)}$ is defined by the Armijo rule [8, Prop. 2.3.1].

4.3 DISTRIBUTED MODEL FOR LAGRANGIAN MINIMIZATION

Here, LM is decomposed into a set $\{LM_i\}$ of subproblems, one for each subsystem, following P_i in the form (3.7a)-(3.7c). For some i and $j \in O(i)$, constraint (3.7b) is recast as:

$$\widehat{\mathbf{Z}}_{j,i}\widehat{\mathbf{u}}_i + \sum_{l \in I(j)\setminus\{i\}} \widehat{\mathbf{Z}}_{j,l}\widehat{\mathbf{u}}_l - \widehat{\mathbf{z}}_j^{\text{bd}} + \widehat{\mathbf{s}}_j = \widehat{\mathbf{Z}}_{j,i}\widehat{\mathbf{u}}_i + \widehat{\mathbf{z}}_{j,i}^{\text{bd}} + \widehat{\mathbf{s}}_j = \mathbf{0}$$

[1] Point $\widehat{\mathbf{v}}$ is stationary if it satisfies first-order optimality conditions for LM.

where $\widehat{\mathbf{z}}_{j,i}^{\text{bd}} = \sum_{l \in I(j) \setminus \{i\}} \widehat{\mathbf{Z}}_{j,l} \widehat{\mathbf{u}}_l - \widehat{\mathbf{z}}_j^{\text{bd}}$. From this equation and the structure of P_i, LM_i becomes:

$$LM_i(\widehat{\mathbf{y}}_i, \widehat{\mathbf{\Lambda}}_i, \mu): \min_{\widehat{\mathbf{u}}_i, \widehat{\mathbf{z}}_i} \mathcal{L}_i = \frac{1}{2}\widehat{\mathbf{u}}_i' \widehat{\mathbf{H}}_i \widehat{\mathbf{u}}_i + \widehat{\mathbf{g}}_i' \widehat{\mathbf{u}}_i + \widehat{c}_i + \sum_{j \in O(i)} \widehat{\boldsymbol{\lambda}}_j' (\widehat{\mathbf{Z}}_{j,i} \widehat{\mathbf{u}}_i + \widehat{\mathbf{z}}_{j,i}^{\text{bd}} + \widehat{\mathbf{s}}_j) + \frac{1}{2\mu} \sum_{j \in O(i)} \|\widehat{\mathbf{Z}}_{j,i} \widehat{\mathbf{u}}_i + \widehat{\mathbf{z}}_{j,i}^{\text{bd}} + \widehat{\mathbf{s}}_j\|^2 \quad (4.6a)$$

$$\text{s.t.}: \ \widehat{\mathbf{s}}_j \geq \mathbf{0}, \ j \in O(i) \quad (4.6b)$$

where $\widehat{\mathbf{\Lambda}}_i = (\widehat{\boldsymbol{\lambda}}_j : j \in O(i))$ has the Lagrange multipliers of the constraints of all subsystems with which subsystem i is coupled. For a subsystem i, define the following:

$$\widetilde{\mathbf{H}}_i = \widehat{\mathbf{H}}_i + \frac{1}{\mu} \sum_{j \in O(i)} \widehat{\mathbf{Z}}_{j,i}' \widehat{\mathbf{Z}}_{j,i},$$

$$\widetilde{c}_i = \widehat{c}_i + \sum_{j \in O(i)} \left(\widehat{\boldsymbol{\lambda}}_j' \widehat{\mathbf{z}}_{j,i}^{\text{bd}} + \frac{1}{2\mu} \|\widehat{\mathbf{z}}_{j,i}^{\text{bd}}\|^2 \right),$$

$$\widetilde{\mathbf{g}}_i = \widehat{\mathbf{g}}_i + \sum_{j \in O(i)} \left(\frac{1}{\mu} \widehat{\mathbf{Z}}_{j,i}' \mathbf{z}_{j,i}^{\text{bd}} + \widehat{\mathbf{Z}}_{j,i}' \widehat{\boldsymbol{\lambda}}_j \right),$$

$$\widetilde{\mathbf{b}}_{i,j} = \widehat{\boldsymbol{\lambda}}_j + \frac{1}{\mu} \widehat{\mathbf{z}}_{i,j}^{\text{bd}}.$$

Then, LM_i is put in a compact form in terms of the local control signals and shared variables:

$$\min_{\widehat{\boldsymbol{\theta}}_i} \mathcal{L}_i = \frac{1}{2}\widehat{\mathbf{u}}_i' \widetilde{\mathbf{H}}_i \widehat{\mathbf{u}}_i + \widetilde{\mathbf{g}}_i' \widehat{\mathbf{u}}_i + \frac{1}{\mu} \sum_{j \in O(i)} \widehat{\mathbf{u}}_i' Z_{j,i}' \widehat{\mathbf{s}}_j + \sum_{j \in O(i)} \left(\widetilde{\mathbf{b}}_{i,j}' \widehat{\mathbf{s}}_j + \frac{1}{2\mu} \|\widehat{\mathbf{s}}_j\|^2 \right) + \widetilde{c}_i \quad (4.7)$$

while being subject to $\widehat{\mathbf{z}}_i = (\widehat{\mathbf{s}}_j : j \in O(i)) \geq \mathbf{0}$ and where $\widehat{\boldsymbol{\theta}}_i = (\widehat{\mathbf{u}}_i, \widehat{\mathbf{z}}_i)$. Notice that LM_i is obtained from LM by dropping from the objective all of the terms that do not depend on $\widehat{\boldsymbol{\theta}}_i$ and removing all of the constraints not affected by $\widehat{\boldsymbol{\theta}}_i$.

Remark. *LM_i is a convex problem on $\widehat{\boldsymbol{\theta}}_i$.*

4.4 DISTRIBUTED GPM

Here, a distributed gradient projection algorithm is demonstrated to solve $\{LM_i : i \in V\}$ with a network of agents, whereby only the nonneighboring agents revise their decisions in each iteration r. The algorithm behaves like centralized GPM with the projected gradient being split into subvectors that are not coupled in LM, each one for the local and shared variables of a particular agent.

Definition 1. *Given an iterate $\widehat{\mathbf{v}}^{(r)}$ and tolerance $\tau \geq 0$, $V(r) \subseteq V$ is a set of nonconflicting and nonoptimal agents if:*

- *for all $i, j \in V(r)$, $i \notin N(j)$ and $j \notin N(i)$, meaning that the agents belonging to $V(r)$ are decoupled with respect to the objective and constraints of LM;*
- *$\|[\widehat{\boldsymbol{\theta}}_i^{(r)} - s\nabla_{\widehat{\boldsymbol{\theta}}_i}\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})]^+ - \widehat{\boldsymbol{\theta}}_i^{(r)}\|^2 > \tau/|V|$, $\forall i \in V(r)$, i.e., the current iterate does not solve LM_i approximately.*

Given a feasible point $\widehat{\boldsymbol{\theta}}_i^{(r)}$ for LM_i at iteration r, agent i generates its next point as:

$$\widehat{\boldsymbol{\theta}}_i^{(r+1)} = \widehat{\boldsymbol{\theta}}_i^{(r)} + \alpha_i^{(r)}\widehat{\mathbf{d}}_i^{(r)} \tag{4.8}$$

where $\alpha_i^{(r)} \in (0, 1]$ is the step length in the direction $\widehat{\mathbf{d}}_i^{(r)}$ defined by gradient projection as:

$$\widehat{\mathbf{d}}_i^{(r)} = \left[\widehat{\boldsymbol{\theta}}_i^{(r)} - s\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})\right]^+ - \widehat{\boldsymbol{\theta}}_i^{(r)} \tag{4.9}$$

where $s > 0$ and $[\widehat{\boldsymbol{\theta}}_i]^+$ is $\widehat{\boldsymbol{\theta}}_i$'s projection onto the feasible space $\mathcal{V}_i := \{\widehat{\boldsymbol{\theta}}_i : \widehat{\mathbf{z}}_i \geq \mathbf{0}\}$ of agent i.

The distributed gradient-projection method for solving problems $\{LM_i\}$ appears in Algorithm 5. In each iteration r, the algorithm selects a group of nonconflicting and nonoptimal agents which apply the gradient-projection method to their subproblems. Algorithm 5 can be shown to converge to the optimal solution $\widehat{\mathbf{v}}^\star$ of LM with tolerance $\tau = 0$. The proof is based on an analysis of the feasible directions $\{\widehat{\mathbf{d}}^{(r)}\}$ obtained by stacking the feasible directions $\{\widehat{\mathbf{d}}_i^{(r)} : i \in V(r)\}$ of the agents that iterate, while the entries of the remaining variables are all zero, *i.e.*, the variables not appearing in $\widehat{\boldsymbol{\theta}}_i$, $\forall i \in V(r)$.

Algorithm 5: Distributed Gradient Projection Method

input: Lagrange vector $\widehat{\boldsymbol{\lambda}}$, penalty μ, initial solution $\widehat{\mathbf{v}}^{\mathrm{s}}$, tolerance τ, and line search parameters s, σ, and γ
set $\widehat{\mathbf{v}}^{(0)} := \widehat{\mathbf{v}}^{\mathrm{s}}$;
for $r := 0, 1, 2, \ldots$ **do**
  if $\|[\widehat{\mathbf{v}}^{(r)} - s\nabla\mathcal{L}(\widehat{\mathbf{v}}^{(r)}, \widehat{\boldsymbol{\lambda}}, \mu)]^{+} - \widehat{\mathbf{v}}^{(r)}\|^2 \leq \tau$ **then**
    return $\widehat{\mathbf{v}}^{(r)}$
  initialize the next iterate: $\widehat{\mathbf{v}}^{(r+1)} := \widehat{\mathbf{v}}^{(r)}$;
  let $V(r) \subseteq V$ be a subset of nonconflicting and nonoptimal agents at $\widehat{\mathbf{v}}^{(r)}$
  for *each agent* $i \in V(r)$ *in parallel* **do**
    obtain $\widehat{\mathbf{y}}_i^{(r)}$ and $\widehat{\mathbf{z}}_i^{(r)}$ from neighbors N_i;
    compute feasible direction $\widehat{\mathbf{d}}_i^{(r)}$ for LM_i at $\widehat{\boldsymbol{\theta}}_i^{(r)}$;
    set $l_i(r) := 0$;
    while $[\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)} + \gamma^{l_i(r)}\widehat{\mathbf{d}}_i^{(r)}) >$
     $\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)}) + \sigma\gamma^{l_i(r)}\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}^{(r)})'\widehat{\mathbf{d}}_i^{(r)}]$ **do**
      set $l_i(r) := l_i(r) + 1$;
    update $\widehat{\mathbf{v}}^{(r+1)}$ by setting $\widehat{\boldsymbol{\theta}}_i^{(r+1)} := \widehat{\boldsymbol{\theta}}_i^{(r)} + \gamma^{l_i(r)}\widehat{\mathbf{d}}_i^{(r)}$;

Suppose that $\{\widehat{\mathbf{v}}^{(r)}\}_{r\in\mathcal{R}}$ converges to a nonstationary point $\widetilde{\mathbf{v}}$ where $\mathcal{R}$ is a subsequence of $\{r\}_{r=0}^{\infty}$. Since $\{\widehat{\mathbf{v}}^{(r)}\}_{r\in\mathcal{R}}$ converges to $\widetilde{\mathbf{v}}$, so does $\{\widehat{\boldsymbol{\theta}}_i^{(r)}\}_{r\in\mathcal{R}}$ to $\widetilde{\boldsymbol{\theta}}_i$ for any agent i. By continuity of the projection operator $[\cdot]^{+}$, for any i we have:

$$\begin{aligned}\lim_{r\to\infty, r\in\mathcal{R}} \widehat{\mathbf{d}}_i^{(r)} &= \lim_{r\to\infty, r\in\mathcal{R}} \left([\widehat{\boldsymbol{\theta}}_i^{(r)} - s\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})]^{+} - \widehat{\boldsymbol{\theta}}_i^{(r)}\right) \\ &= [\widetilde{\boldsymbol{\theta}}_i - s\nabla\mathcal{L}_i(\widetilde{\boldsymbol{\theta}}_i)]^{+} - \widetilde{\boldsymbol{\theta}}_i = [\widetilde{\boldsymbol{\theta}}_i - s\nabla_{\widehat{\boldsymbol{\theta}}_i}\mathcal{L}(\widetilde{\mathbf{v}})]^{+} - \widetilde{\boldsymbol{\theta}}_i\end{aligned}$$

with the last equality from perfect decomposition. Thus,

$$\lim_{r\to\infty, r\in\mathcal{R}} (\widehat{\mathbf{d}}_i^{(r)} : i \in V) = \left([\widetilde{\boldsymbol{\theta}}_i - s\nabla_{\widehat{\boldsymbol{\theta}}_i}\mathcal{L}(\widetilde{\mathbf{v}})]^{+} - \widetilde{\boldsymbol{\theta}}_i : i \in V\right)$$

which implies the first condition for convergence (see condition (2.10) of [8, pp.194]):

$$\limsup_{r\to\infty, r\in\mathcal{R}} \|\widehat{\mathbf{d}}^{(r)}\| = \|[\widetilde{\mathbf{v}} - s\nabla\mathcal{L}(\widetilde{\mathbf{v}})]^{+} - \widetilde{\mathbf{v}}\| = \|\widetilde{\mathbf{d}}\| < \infty \qquad (4.10)$$

because i) the decomposition is perfect, ii) $\widehat{\boldsymbol{\theta}}_i = (\widehat{\mathbf{u}}_i, \widehat{\mathbf{z}}_i)$, and iii) the projection with respect to the shared variables $\widehat{\mathbf{z}}_i$ is identical on the nonnegative orthant, regardless of which agent i calculates the projection. From the characteristics of the projection operator established by Proposition 2.1.3 from [8, pp. 183], for any iterate $\widehat{\mathbf{v}}^{(r)}$ and agent i we have that:

$$(\widehat{\boldsymbol{\theta}}_i^{(r)} - s\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)}) - \widehat{\boldsymbol{\phi}}_i^{(r)})'(\widehat{\boldsymbol{\theta}}_i - \widehat{\boldsymbol{\phi}}_i^{(r)}) \leq 0,\ \forall \widehat{\boldsymbol{\theta}}_i \in \mathcal{V}_i$$

where $\widehat{\boldsymbol{\phi}}_i^{(r)} = [\widehat{\boldsymbol{\theta}}_i^{(r)} - s\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})]^+$ denotes the projection on $\mathcal{V}_i$. By manipulating this inequality with $\widehat{\boldsymbol{\theta}}_i = \widehat{\boldsymbol{\theta}}_i^{(r)}$, we obtain:

$$\begin{aligned}\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})'(\widehat{\boldsymbol{\phi}}_i^{(r)} - \widehat{\boldsymbol{\theta}}_i^{(r)}) = \nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})'\widehat{\mathbf{d}}_i^{(r)} &\leq -\frac{1}{s}\|\widehat{\boldsymbol{\theta}}_i^{(r)} - \widehat{\boldsymbol{\phi}}_i^{(r)}\|^2 \\ &= -\frac{1}{s}\|\widehat{\mathbf{d}}_i^{(r)}\|^2. \quad (4.11)\end{aligned}$$

At any iteration r, the global direction $\widehat{\mathbf{d}}^{(r)}$ is given by stacking the vectors $\widehat{\mathbf{d}}_i^{(r)}$ and associating them with $\widehat{\boldsymbol{\theta}}_i$ for all $i \in V(r)$, while the remaining entries are set to zero, which are neither local nor shared variables of the agents $i \in V(r)$. Therefore,

$$\begin{aligned}\nabla\mathcal{L}(\widehat{\mathbf{v}}^{(r)})'\widehat{\mathbf{d}}^{(r)} = \sum_{i\in V(r)} \nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})'\widehat{\mathbf{d}}_i^{(r)} &\leq -\frac{1}{s}\sum_{i\in V(r)} \|\widehat{\mathbf{d}}_i^{(r)}\|^2 \\ &= -\frac{1}{s}\|\widehat{\mathbf{d}}^{(r)}\|^2.\end{aligned}$$

Taking the limit of this relation, the second condition for convergence is obtained:

$$\limsup_{r\to\infty, r\in\mathcal{R}} \nabla\mathcal{L}(\widehat{\mathbf{v}}^{(r)})'\widehat{\mathbf{d}}^{(r)} \leq -\frac{1}{s}\|\widetilde{\mathbf{d}}\|^2 = -\frac{1}{s}\sum_{i\in\widetilde{V}} \|\widetilde{\mathbf{d}}_i\|^2 < 0 \quad (4.12)$$

where $\widetilde{\mathbf{d}}_i = [\widetilde{\boldsymbol{\theta}}_i - s\nabla\mathcal{L}_i(\widetilde{\boldsymbol{\theta}}_i)]^+ - \widetilde{\boldsymbol{\theta}}_i$ and $\widetilde{V}$ is any nonoptimal and nonconflicting subset of agents that iterate at the nonstationary point $\widetilde{\mathbf{v}}$, to which the subseries $\{\widehat{\mathbf{v}}^{(r)}\}_{r\in\mathcal{R}}$ converges.

Lemma 1. *The sequence of feasible directions $\{\widehat{\mathbf{d}}^{(r)}\}$ produced by the gradient projection of the agents that work in each iteration r is gradient related.*

Proof. Because the conditions (2.10) of [8, pp. 194] are ensured from inequalities (4.10) and (4.12), the sequence of feasible directions $\{\widehat{\mathbf{d}}^{(r)}\}$ is gradient related. □

The step length $\alpha_i^{(r)}$ implemented by each agent $i \in V(r)$ satisfies the Armijo rule:

$$\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)} + \alpha_i^{(r)}\widehat{\mathbf{d}}_i^{(r)}) \leq \mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)}) + \sigma\alpha_i^{(r)}\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})'\widehat{\mathbf{d}}_i^{(r)} \tag{4.13}$$

where $\alpha_i^{(r)} = \gamma^{l_i(r)}$ for some $l_i(r) \in \{0, 1, 2, \ldots\}$ computed by Algorithm 5. Let $\alpha^{(r)} = \min\{\alpha_i^{(r)} : i \in V(r)\}$. Because $\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)} + \alpha_i\widehat{\mathbf{d}}_i^{(r)})$ is convex on α_i and $\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)})'\widehat{\mathbf{d}}_i^{(r)} < 0$, the Armijo rule (4.13) is satisfied for any $\alpha_i \in (0, \alpha_i^{(r)}]$. Adding up inequalities (4.13) over all $i \in V(r)$ results in

$$\begin{aligned}
&\sum_{i\in V(r)} \mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)} + \alpha_i^{(r)}\widehat{\mathbf{d}}_i^{(r)}) \\
&\qquad \leq \sum_{i\in V(r)} \left(\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)}) + \sigma\alpha_i^{(r)}\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}^{(r)})'\widehat{\mathbf{d}}_i^{(r)}\right) \\
&\qquad \leq \sum_{i\in V(r)} \left(\mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)}) + \sigma\alpha^{(r)}\nabla\mathcal{L}_i(\widehat{\boldsymbol{\theta}}^{(r)})'\widehat{\mathbf{d}}_i^{(r)}\right) \\
&\qquad\qquad = \sum_{i\in V(r)} \mathcal{L}_i(\widehat{\boldsymbol{\theta}}_i^{(r)}) + \sigma\alpha^{(r)}\nabla\mathcal{L}(\widehat{\mathbf{v}}^{(r)})'\widehat{\mathbf{d}}^{(r)}.
\end{aligned}$$

Let $c^{(r)}$ be the constant corresponding to the part of $\mathcal{L}(\widehat{\mathbf{v}})$ that does not depend on $(\widehat{\boldsymbol{\theta}}_i : i \in V(r))$ and evaluated at $\widehat{\mathbf{v}}^{(r)}$. By adding $c^{(r)}$ to both sides of this inequality, we obtain:

$$\mathcal{L}(\widehat{\mathbf{v}}^{(r)} + \alpha^{(r)}\widehat{\mathbf{d}}^{(r)}) \leq \mathcal{L}(\widehat{\mathbf{v}}^{(r)}) + \sigma\alpha^{(r)}\nabla\mathcal{L}(\widehat{\mathbf{v}}^{(r)})'\widehat{\mathbf{d}}^{(r)}. \tag{4.14}$$

Lemma 2. *The step* $\alpha^{(r)} = \min\{\alpha_i^{(r)} : i \in V(r)\}$ *satisfies the Armijo rule* (4.14) *at* $\widehat{\mathbf{v}}^{(r)}$ *for the feasible direction* $\widehat{\mathbf{d}}^{(r)}$.

Theorem 1. *The sequence* $\{\widehat{\mathbf{v}}^{(r)}\}_{r=0}^{\infty}$ *generated by distributed GPM converges to an optimal solution* $\widehat{\mathbf{v}}^{\star}$ *of* LM.

Proof. At each iteration r, distributed GPM yields no less decrease on the augmented Lagrangian $\mathcal{L}$ than the centralized descent approach induced by the aggregated direction $\widehat{\mathbf{d}}^{(r)}$ with step length

$\alpha^{(r)}$. Since $\{\widehat{\mathbf{d}}^{(r)}\}_{r=0}^{\infty}$ is gradient related by Lemma 1 and $\alpha^{(r)}$ satisfies the Armijo rule by Lemma 2, Prop. 2.2.1 of [8] ensures convergence of $\{\widehat{\mathbf{v}}^{(r)}\}_{r=0}^{\infty}$ to $\widehat{\mathbf{v}}^{\star}$. □

The strategy of inequality (4.13) for finding an acceptable step length $\alpha_i^{(r)}$, meaning one that satisfies the Armijo rule, is also known as backtracking line search. Other line search strategies can be used such as the optimal step length $\alpha_i^{(r)\star}$ which minimizes $\mathcal{L}_i(\widehat{\mathbf{v}}^{(r)} + \alpha_i \widehat{\mathbf{d}}_i^{(r)})$. The optimal step length $\alpha_i^{(r)\star}$ obviously satisfies the Armijo rule since it induces the maximum decrease along the search direction. Since $\mathcal{L}_i$ is convex, the optimal step length can be found using exact line search methods with and without derivatives (*e.g.*, Dichotomous search and Golden section) [17].

4.5 STRUCTURE OF DISTRIBUTED GPM

The distributed dual algorithm has an *outer loop*, which updates the Lagrange multipliers and penalty factor, and an *inner loop*, which solves the Lagrangian subproblem:

- the *outer loop* consists of Algorithm 4, which yields a sequence $\{\widehat{\mathbf{v}}^{(k)}\}_{k=1}^{K}$ converging to a solution to P. The sequence $\{\widehat{\mathbf{v}}^{(k)}\}$ is obtained by solving the Lagrange subproblem LM_k, which is equivalent to $LM_k(\boldsymbol{\lambda}^{(k)}, \mu^{(k)})$ for the current multipliers and penalty factor. In this process, the algorithm produces a decreasing sequence $\{\mu^{(k)}\}$ of penalty factors and a sequence $\{\boldsymbol{\lambda}^{(k)}\}$ of Lagrange multipliers, whereby $\boldsymbol{\lambda}^{(k+1)}$ is calculated from $\boldsymbol{\lambda}^{(k)}$ and $\widehat{\mathbf{v}}^{(k)}$.

- the *inner loop* consists of Algorithm 5, which solves the Lagrange subproblem $LM_k(\boldsymbol{\lambda}^{(k)}, \mu^{(k)})$. To this end, this paper proposed the distributed gradient-projection algorithm, which produces a sequence $\{\widehat{\mathbf{v}}^{(k,r)}\}_{r=1}^{R}$ of iterates arriving at the solution $\widehat{\mathbf{v}}^{(k)}$ to $LM_k(\boldsymbol{\lambda}^{(k)}, \mu^{(k)})$.

The structure and information flow of the algorithm is depicted in Figure 4.1. Synchronization, information exchange and condition detection, such as convergence, can be carried out in a distributed manner with asynchronous protocols based on massage passing. This work does not address such an implementation, but an overview of how this can be achieved is found in [18, 19].

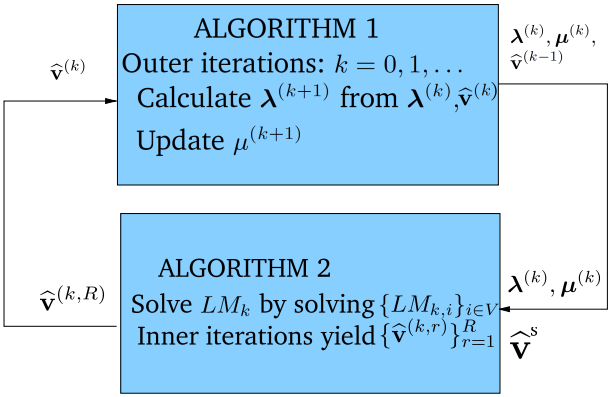

Figure 4.1: Structure of the distributed dual algorithm.

4.6 BRIEF LITERATURE REVIEW

In [20] a model-based generalized predictive control is reported, where a two-level decentralized Kalman filter is used to locally estimate the states of each subprocess, the output predictor is designed using the state estimates from a set of Kalman filters. A decentralized Kalman filter is formulated using maximum a posteriori approach and a quadratic performance index similar to a LQ index is selected for each subprocess. [21] does a review on predictive control and discusses the behavior and the performance of the system under the decentralized receding horizon control for known and unknown initial condition cases as well as a time varying case. [22] present a detailed study of the inclusion concept in dynamic systems, which is a suitable mathematical framework for comparing systems with different dimensions. The framework offers immediate results in reduced-order modeling and the overlapping decentralized control of complex systems. It is limited to linear constant systems, but the formulation of the principle is given in terms of the motions of the dynamical systems rather than in algebraic terms. This way, the principle can be applied to stability and control problems of time-varying and nonlinear systems as well as models involving hereditary elements and stochastic effects.

Two methods are presented in [23] for grouping actuators while minimizing performance degradation in distributed control systems with saturation constraints. The first of these formulates the

problem into an open-loop Max k-Cut optimization problem, while the second, using a closed-loop dynamic performance objective, is a recursive algorithm based on the Linear Quadratic Regulator. [24] proposes a DMPC design approach for open-loop asymptotically stable processes whose dynamics are not necessarily decoupled. A set of partially decoupled approximate prediction models are defined and used by different MPC controllers. The less coupled are the submodels, the lighter the computational burden and the load for transmission of information among the decentralized MPC controllers, but the worse is the performance of the control system and the less likely the proposed stability test succeeds. [25] presents a differential game approach to formation control of mobile robots. The formation control is formulated as a linear-quadratic Nash differential game through the use of graph theory. An open-loop Nash equilibrium solution is investigated by establishing existence and stability conditions of the solutions of coupled Riccati differential equations.

4.7 DISCUSSION AND ILUSTRATIVE PROBLEM

In the literature, one will find distributed optimization methods based on the augmented Lagrangian that are related to this work. Of particular interest are Lagrangian decomposition [17, Sec. 6.4] and the Alternating Direction Method of Multipliers (ADMM) [26], both of which rely on consensus variables and constraints. To illustrate their difference with respect to our method, consider a simple problem:

$$\min_{x_1,x_2} \quad f_1(x_1) + f_{12}(x_1,x_2) + f_2(x_2) \tag{4.15a}$$

$$\text{s.t. : } g_1(x_1) \leq 0, \tag{4.15b}$$

$$g_{12}(x_1,x_2) \leq 0, \tag{4.15c}$$

$$g_2(x_2) \leq 0. \tag{4.15d}$$

associated to a LDN with $V = \{1,2\}$ and $E = \{(1,2)\}$. By introducing a consensus variable for $\widetilde{x}_2$, a reformulation is obtained:

$$\min_{x_1,x_2,\widetilde{x}_2} \quad f_1(x_1) + f_{12}(x_1,\widetilde{x}_2) + f_2(x_2) \tag{4.16a}$$

$$\text{s.t. } g_1(x_1) \leq 0, \tag{4.16b}$$

$$g_{12}(x_1,\widetilde{x}_2) \leq 0, \tag{4.16c}$$

$$g_2(x_2) \leq 0, \tag{4.16d}$$

$$\widetilde{x}_2 = x_2. \tag{4.16e}$$

which, upon dualization of the consensus constraint (4.16e), leads to an augmented Lagrangian subproblem with a structure which is exploited by ADMM:

1. For subsystem 1, the iterative process becomes:

$$(x_1^{k+1}, \widetilde{x}_2^{k+1}) = \underset{x_1, \widetilde{x}_2}{\arg\min}\, f_1(x_1) + f_{12}(x_1, \widetilde{x}_2) + \lambda^k(x_2^k - \widetilde{x}_2) + (x_2^k - \widetilde{x}_2)^2/(2\mu^k)$$

$$\text{s.t.} : g_1(x_1) \leq 0,$$
$$g_{12}(x_1, \widetilde{x}_2) \leq 0.$$

2. For subsystem 2:

$$x_2^{k+1} = \underset{x_2}{\arg\min}\, f_2(x_2) + \lambda^k(x_2 - \widetilde{x}_2^{k+1}) + (x_2 - \widetilde{x}_2^{k+1})^2/(2\mu^k)$$

$$\text{s.t.} : g_2(x_2) \leq 0.$$

3. Completing the iterative process, the Lagrange multiplier and penalty factor are updated as follows:

$$\lambda^{k+1} = \lambda^k + (x_2^{k+1} - \widetilde{x}_2^{k+1})/\mu^k,$$
$$\mu^{k+1} = \gamma\mu^k,$$

with $\gamma \in (0, 1)$ being a decreasing rate.

Notice that ADMM does not have an inner loop, whereby decision variables and multipliers are updated at the same rate.

On the other hand, the proposed distributed algorithm converts inequalities into equalities by introducing slack variables. Subproblem LM_k, at outer iteration k, is projected onto the decision space of each subsystem i, which includes its local variables $\widehat{\mathbf{u}}_i$ and the slack variables $\widehat{\mathbf{z}}_i$ of the coupled constraints that may be shared with other subsystems. Let gpm denote an operator that performs gradient projection followed by a line-search that satisfies the Armijo rule, namely the operator that computes:

$$\widehat{\boldsymbol{\theta}}_i^{(k,r+1)} = \widehat{\boldsymbol{\theta}}_i^{(k,r)} + \gamma^{l_i(k,r)}\widehat{\mathbf{d}}_i^{(k,r)}.$$

Then resulting iterative processes for inner iterations are:

1. For an even numbered iteration r, an improved solution to $LM_{k,1}$ is obtained as follows:

$$\begin{aligned}(x_1^{k,r+1}, s_1^{k,r+1}, s_{12}^{k,r+1}) = \underset{x_1,s_1,s_{12}}{\text{gpm}} \; & f_1(x_1) \\ & + f_{12}(x_1, x_2^{k,r}) + \lambda_1^k(g_1(x_1) + s_1) \\ & + \lambda_{12}^k(g_{12}(x_1, x_2^{k,r}) + s_{12}) \\ & + (g_1(x_1) + s_1)^2/(2\mu^k) + (x_1, x_2^{k,r}) + s_{12})^2/(2\mu^k)\end{aligned}$$

$$\text{s.t.} : s_1, s_{12} \geq 0,$$

while subsystems 2's decision variables are kept fixed:

$$(x_2^{k,r+1}, s_2^{k,r+1}) = (x_2^{k,r}, s_2^{k,r}).$$

2. For an odd numbered iteration r, an improved solution to $LM_{k,2}$ is obtained:

$$\begin{aligned}(x_2^{k,r+1}, s_2^{k,r+1}, s_{12}^{k,r+1}) = \underset{x_2,s_2,s_{12}}{\text{gpm}} \; & f_2(x_2) \\ & + f_{12}(x_1^{k,r+1}, x_2) + \lambda_2^k(g_2(x_2) + s_2) \\ & + \lambda_{12}^k(g_{12}(x_1^{k,r}, x_2) + s_{12}) \\ & + (g_2(x_2) + s_2)^2/(2\mu^k) + (x_1^{k,r}, x_2) + s_{12})^2/(2\mu^k)\end{aligned}$$

$$\text{s.t.} : s_2, s_{12} \geq 0.$$

while subsystem 1's decision variables are not affected:

$$(x_1^{k,r+1}, s_1^{k,r+1}) = (x_1^{k,r}, s_1^{k,r}).$$

Upon convergence of the iterates in steps 1 and 2, an approximate solution $\widehat{\mathbf{v}}^k = (x_1^k, x_2^k, s_1^k, s_2^k, s_{12}^k)$ to LM_k is obtained. Then, the Lagrange multipliers and penalty factor are updated

$$\begin{aligned}\lambda_1^{k+1} &= \lambda_1^k + g_1(x_1^k)/\mu^k, \\ \lambda_2^{k+1} &= \lambda_2^k + g_2(x_2^k)/\mu^k, \\ \lambda_{12}^{k+1} &= \lambda_{12}^k + g_{12}(x_1^k, x_2^k)/\mu^k, \\ \mu^{k+1} &= \gamma\mu^k,\end{aligned}$$

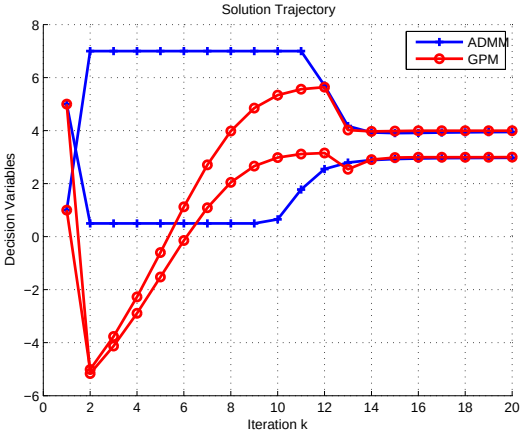

Figure 4.2: Iterations generated by ADMM and distributed GPM for the sample problem..

and the next outer iteration $k+1$ is started.

Aiming to illustrate the behavior of ADMM and distributed GPM, their iterative processes given above were applied to problem (4.15) defined by the functions:

$$\begin{aligned} f_1(x_1) &= x_1^2/2, & f_2(x_2) &= x_2^2 - 4x_2, \\ f_{12}(x_1, x_2) &= -x_1 x_2, & & \\ g_1(x_1) &= x_1 - 7, & g_2(x_2) &= -x_2 + 0.5, \\ g_{12}(x_1, x_2) &= -x_1 + 2x_2 - 2. & & \end{aligned}$$

The starting solution for ADMM was $x_1^0 = 1$, $x_2^0 = 5$, $\lambda^0 = 10$, and $\mu^0 = 30$. For GPM, the starting solution was the same, except that $s_1^0 = s_2^0 = s_{12}^0 = 30$ and $\lambda_1^0 = \lambda_2^0 = \lambda_{12}^0 = 10$. The decreasing rate was $\gamma = 0.7$. The trajectories traced by the algorithms in the (x_1, x_2) space appear in Figure 4.2. It can be noticed that the trajectories of both algorithms converge to the optimum $(x_1^\star, x_2^\star) = (4, 3)$.

4.8 SUMMARY

This chapter presented the augmented lagragian algorithm, picked up from the model seen in the previous chapter, the algorithm is formalized and the formulae for choosing the lagragian

multiplier is expressed. To solve the inner problems of the previous algorithm, the gradient projection method is presented as an option. The augmented lagragian algorithm is then decomposed, to represent a distributed model, one subproblem for each subsystem and thus so is the gradient projection method. The structure of the algorithm is reinforced, to show its two phases, the outter and inner loops. A example is demonstraded to facilitate the reader's understanding of the chapter, to present the advantages and to compare the presented algorithm with others existed in the literature.

5 APPLICATION TO URBAN TRAFFIC CONTROL

The dual augmented Lagragian algorithm for distributed model predictive control is applied to the control of an urban-traffic network. This method is compared with other methods to validate the proposed algorithm.

5.1 TRAFFIC FLOW MODEL

The urban road network is represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with junctions $j \in \mathcal{V}$, where $\mathcal{V} = \{1, \ldots, n\}$, and links $e \in \mathcal{E}$, where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. For each junction j we define a set of upstream $I(j)$ and downstream $O(j)$ junctions, thus $\mathcal{E}(j) = \{(i,j) \in \mathcal{E} : i \in I(j)\}$ is the set of links reaching junction j. We also assume that the cycle time C_j is fixed or calculated in real-time by another algorithm. In addition, to enable network coordination, we can assume that the cycle time is the same for every junction. This model comes from [1].

The constraint

$$\sum_{(i,j)\in\mathcal{E}(j)} u_{i,j} = C_j \tag{5.1}$$

ensures that the green times add up to cycle time, where $u_{i,j}$ is the green time of link (i,j) at junction j, with no lost time.

Considering a link $z = (i,j)$ connecting two junctions i and j, the dynamics of link z is given by the equation:

$$x_z(k+1) = x_z(k) + \tau[q_z(k) - s_z(k) + d_z(k) - p_z(k)] \tag{5.2}$$

where $x_z(k)$ is the number of vehicles in link z at time $k\tau$ (queues), q_z and p_z are respectively the inflow and outflow of link z during the period $[k\tau, (k+1)\tau]$, τ is the discrete-time step and k is the discrete-time index, d_z and s_z are the demand and outflow within the link, respectively [27]. Figure 5.1 shows how this happens in the link z.

The mathematical model based in Equation (5.2), which is chosen to describe the dynamics of the vehicle queues, is known as store-and-forward. There, the evolution of the queues in a link depends on its initial queues, the physical characteristics of the traffic network, and the green time of its traffic lights and of those that feed it.

Queues are subject to the constraints

$$0 \leq x_z(k) \leq x_z^{\max}, \quad \forall z \in \mathcal{E} \tag{5.3}$$

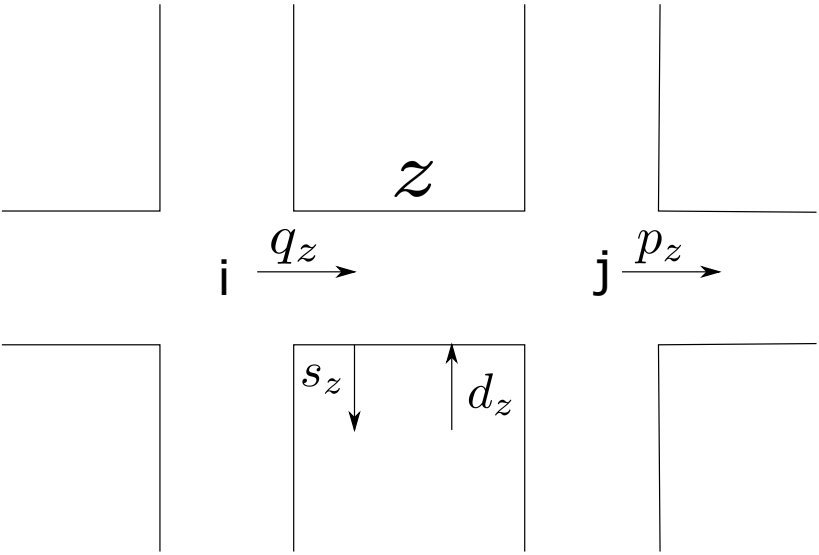

Figure 5.1: Link z in a traffic network.

where $x_z^{\max}$ is the maximum queue length in vehicles of link z.

Green times are subject to the constraints

$$u_z^{\min} \leq u_z(k) \leq u_z^{\max}, \quad \forall z \in \mathcal{E} \tag{5.4}$$

where $u_z^{\max}$ and $u_z^{\min}$ are the maximum and minimum green times giving right of way to link z.

The inflow to link z is given by:

$$q_z(k) = \sum_{w \in \mathcal{E}(j)} t_{w,z} p_w(k) \tag{5.5}$$

where $t_{w,z} \in [0,1]$ with $w \in \mathcal{E}(j)$ is the turning rate towards link z from link w reaching junction j and p_w is the outflow of link w. This means that the arrival of vehicles in link z depends of the rate of conversion of links upstream of z into the link z and the outflow of vehicles of such links. Notice that $\sum_{l \in O(j)} t_{w,(j,l)} = 1$.

Assuming that p_z is equal to the saturation flow S_z if the link $z = (i,j)$ has right-of-way, and equal to zero otherwise, simplifications can be made for the outflow p_z. If the discrete-time step τ

is equal to the cycle time C, an average value for the outflow is obtained

$$p_z(k) = u_z(k)S_z/C \tag{5.6}$$

in which $u_z(k)$ is the green time allocated to link z.

In order to simplify computational analysis and notation, this work assumes that each link z will have a dedicated phase, during which the green time is exclusively assigned to the discharge of its vehicles.

Figure 5.2 represents a traffic network with 8 junctions, where the links are characterized by two junctions involved, the one that discharges vehicles and the other which is affected by this discharging.

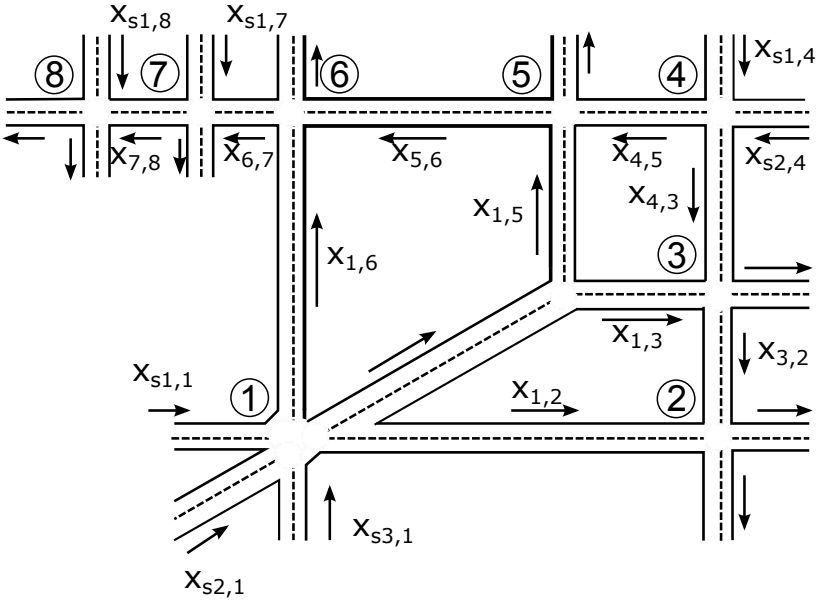

Figure 5.2: Traffic network. Variable $x_{i,j}$ models the queue of vehicles in link $z = (i, j)$ that leaves junction i and reaches junction j, with the exception of the external links (s, j) which represent arrivals at junction j.

Because the network $\mathcal{G}$ only represents internal links of the network, the external links that model the arrival of vehicles must be represented. Let $s(j)$ be the number of external links reaching

junction j. In Figure 5.2, three external links reach junction $j = 1$, with queues given by $x^1_{s,j}$, $x^2_{s,j}$, and $x^3_{s,j}$.

Figure 5.3 represents the traffic network system from Figure 5.2 as a graph. Each node represents a junction whose state $x_z(k)$ is the number of vehicles in the links z that reach the junction, and $u_z(k)$ is the green time assigned to these links during the k-th cycle. For instance, the state of junction 5 is $x_5 = (x_{1,5}, x_{4,5})$ and its control vector is $u_5 = (u_{1,5}, u_{4,5})$.

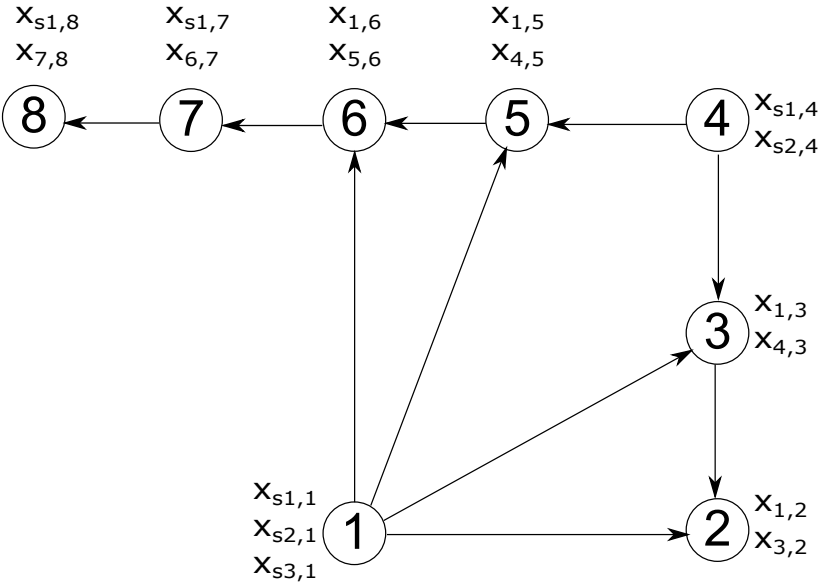

Figure 5.3: Graph model of the proposed traffic network.

Replacing (5.6) and (5.5) in (5.2) and rearranging all resulting equations in one single vector-based equation, leads to a linear state-space model

$$\mathbf{x}(k+1) = \mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) + T\mathbf{d}(k) \tag{5.7}$$

where:

- $\mathbf{x} = (x_{i,j} : (i,j) \in \mathcal{E}) \cup (x^l_{s,j} : l = 1, \ldots, s(j),\ j \in \mathcal{V})$ is the state vector, consisting of the queues of each link,

- $\mathbf{u} = (u_{i,j} : (i,j) \in \mathcal{E}) \cup (u^l_{s,j} : l = 1, \ldots, s(j),\ j \in \mathcal{V})$ is the control vector, consisting of all the green times,
- $\mathbf{d}$ is the disturbance vector, consisting of the d_z and s_z of each link, and
- $\mathbf{B}$ is obtained from the substitution of (5.6) and (5.5) in (5.2).

As a consequence of the simplification (5.6), the model is not aware of short-term oscillations for taking sample time equal to cycle time. Further, offsets and cycle times have no impact in the model, which are assumed fixed or updated in real-time independently.

5.2 URBAN TRAFFIC CONTROL

Traffic-responsive Urban Control (TUC) has been developed to tackle the problem of traffic-responsive network-wide signal control under saturated traffic conditions. In contrast to other proposed methods for urban signal control, the feedback approach that is pursued by TUC involves the application of systematic and powerful control design methods. The basic philosophy and the importance of these methods are related to their general applicability to any process that can be described by certain types of mathematical models, regardless of the physical process nature [28]. Moreover, in contrast to other proposed methodologies, the specific store-and-forward modeling approach employed in the design of TUC, permits the use of highly efficient optimization and control methods with polynomial complexity leading to a straightforward network-wide applicability, easy installation and maintenance, as well as low requirements regarding real-time traffic measurements.

In traffic control terminology, *stage* is a particular set of green indications that give simultaneous right of way to non-conflicting movements of an intersection; *cycle* is the time needed to complete a full round of traffic light stages; and *split* is the portion of the cycle allocated to a certain stage. The control objective is to minimize and balance the number of vehicles within the urban links approaching urban signalized junctions by varying, in a co-ordinated manner, the green split of the signal cycles under some assumptions [29].

The LDN with dynamic equation (3.1) can model urban traffic networks [1, 13] with the state of a subsystem (intersection) being the incoming vehicles queues, which evolve depending on green-time signals at the downstream and upstream subsystems. In order

for the application of the LQ-methodology to lead to a feedback control law without feedforward terms, disturbance is assumed null. In [30] the application is solved using an algorithm embedded in a rolling horizon (model-predictive) scheme. The optimal control problem is solved once per cycle using the current state of the system as well predicted demand flows over the finite horizon Th. The optimization yields an optimal control sequence for Th cycles, but only the first control in this sequence is actually applied to the junctions of the traffic network.

Given the state of subsystem i at time t and the future control signals from the input neighborhood, the state $\widehat{x}_i$ of subsystem i over the time horizon is given by

$$\widehat{\mathbf{x}}_i = \widehat{\mathbf{A}}_i \mathbf{x}_i(t) + \sum_{j \in I(i)} \widehat{\mathbf{B}}_{i,j} \widehat{\mathbf{u}}_j \tag{5.8}$$

where $\widehat{\mathbf{x}}_i = (\mathbf{x}_i(t+1), \ldots, \mathbf{x}_i(t+T))$, $\widehat{\mathbf{u}}_i = (\mathbf{u}_i(t), \ldots, \mathbf{u}_i(t+T-1))$, and $\widehat{\mathbf{A}}_i$ and $\widehat{\mathbf{B}}_{i,j}$ are matrices obtained from the dynamics (3.1) in a straightforward manner [14].

A quadratic criterion that considers this control objective has the general form

$$\mathbf{J} = \frac{1}{2} \sum_{t=0}^{\infty} (\widehat{\mathbf{x}}(t)' \mathbf{Q} \widehat{\mathbf{x}}(t) + \widehat{\mathbf{u}}(t)' \mathbf{R} \widehat{\mathbf{u}}(t)) \tag{5.9}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are non-negative definite, diagonal weighting matrices. The infinite time horizon in (5.9) is taken in order to obtain a time-invariant feedback law according to the LQ optimization theory. The first term in (5.9) is responsible for minimization and balancing of the relative queues of the network links. To this end, the diagonal elements of $\mathbf{Q}$ are set equal to the inverses of the storage capacities of the corresponding links. Furthermore, the magnitude of the control reactions can be influenced by the choice of the weighting matrix $\mathbf{R}$.

In practice Equation (5.9) is not used, because it can vary the green-time abruptly, instead the control variable is a penalization of the nominal control $\widetilde{\mathbf{u}}$. Then the objective would take the form:

$$\mathbf{J} = \frac{1}{2} \sum_{t=0}^{\infty} [\widehat{\mathbf{x}}(t)' \mathbf{Q} \widehat{\mathbf{x}}(t) + (\widehat{\mathbf{u}}(t) - \widetilde{\mathbf{u}})' \mathbf{R} (\widehat{\mathbf{u}}(t) - \widetilde{\mathbf{u}})] \tag{5.10}$$

5.3 DISTRIBUTED MODEL PREDICTIVE CONTROL

As seen in the sections above, it is possible to represent a traffic network with a state-space representation which makes possible to utilize the techniques seen in previous chapters to solve the optimization problem described by Equation (5.10).

For the network modeled by Figure 5.3 there are 8 subsystems, therefore 8 subproblems to solve. The network is constrained by its road storage, a maximum and a minimum green time, and the cycle time of each subsystem. For the model to better represent the reality the constraint that the queues should be non negative is added in each link of the network. To ensure that this constraint will be always feasible, the model was modified by introducing the effective green-time, which changes the cycle restriction from an equality to an inequality.

As seen in Chapter 3, the constraints for subsystem 8, for exemple, become:

$$\widehat{\mathbf{u}}_8^{\min} \leq \widehat{\mathbf{u}}_8 \leq \widehat{\mathbf{u}}_8^{\max} \tag{5.11a}$$

$$\widehat{\mathbf{M}}_8 \widehat{\mathbf{u}}_8 \leq C \tag{5.11b}$$

$$-(\widehat{\mathbf{A}}_8 \widehat{\mathbf{x}}_8 + \widehat{\mathbf{B}}_{87} \widehat{\mathbf{u}}_7) \leq \widehat{\mathbf{B}}_8 \widehat{\mathbf{u}}_8 \leq \widehat{\mathbf{x}}_8^{\max} - \widehat{\mathbf{A}}_8 \widehat{\mathbf{x}}_8 - \widehat{\mathbf{B}}_{87} \widehat{\mathbf{u}}_7 \tag{5.11c}$$

where $\widehat{\mathbf{M}}_8$ is a matrix that considers all the controls signals of subsytem 8 in each period of prediction, C is the cycle time of the system, and matrix $\widehat{\mathbf{B}}_{87}$ models the influence of subsystem 7 on subsystem 8.

With that the subproblem 8 becomes:

$$P_8 : \min \frac{1}{2} \widehat{\mathbf{u}}_8' \widehat{\mathbf{H}}_8 \widehat{\mathbf{u}}_8 + \widehat{\mathbf{g}}_8' \widehat{\mathbf{u}}_8 + \widehat{c}_8 \tag{5.12a}$$

$$\text{s.t. :} \widehat{\mathbf{u}}_8^{\min} \leq \widehat{\mathbf{u}}_8 \leq \widehat{\mathbf{u}}_8^{\max} \tag{5.12b}$$

$$\widehat{\mathbf{M}}_8 \widehat{\mathbf{u}_8} \leq C \tag{5.12c}$$

$$-(\widehat{\mathbf{A}}_8 \widehat{\mathbf{x}}_8 + \widehat{\mathbf{B}}_{87} \widehat{\mathbf{u}}_7) \leq \widehat{\mathbf{B}}_8 \widehat{\mathbf{u}}_8 \leq \widehat{\mathbf{x}}_8^{\max} - \widehat{\mathbf{A}}_8 \widehat{\mathbf{x}}_8 - \widehat{\mathbf{B}}_{87} \widehat{\mathbf{u}}_7 \tag{5.12d}$$

Subproblem 7 has a subproblem downstrem and upstream,

its equations becomes:

$$P_7 : \min \frac{1}{2}\widehat{\mathbf{u}}_7'\widehat{\mathbf{H}}_7\widehat{\mathbf{u}}_7 + \widehat{\mathbf{g}}_7'\widehat{\mathbf{u}}_7 + \widehat{c}_7 \tag{5.13a}$$

$$\text{s.t. :}\widehat{\mathbf{u}}_7^{\min} \leq \widehat{\mathbf{u}}_7 \leq \widehat{\mathbf{u}}_7^{\max} \tag{5.13b}$$

$$\widehat{\mathbf{M}_7\mathbf{u}_7} \leq C \tag{5.13c}$$

$$-(\widehat{\mathbf{A}}_7\widehat{\mathbf{x}}_7 + \widehat{\mathbf{B}}_{76}\widehat{\mathbf{u}}_7) \leq \widehat{\mathbf{B}}_7\widehat{\mathbf{u}}_7 \leq \widehat{\mathbf{x}}_7^{\max} - \widehat{\mathbf{A}}_7\widehat{\mathbf{x}}_7 - \widehat{\mathbf{B}}_{76}\widehat{\mathbf{u}}_7 \tag{5.13d}$$

$$-(\widehat{\mathbf{A}}_8\widehat{\mathbf{x}}_8 + \widehat{\mathbf{B}}_8\widehat{\mathbf{u}}_8) \leq \widehat{\mathbf{B}}_7\widehat{\mathbf{u}}_7 \leq \widehat{\mathbf{x}}_8^{\max} - \widehat{\mathbf{A}}_8\widehat{\mathbf{x}}_8 - \widehat{\mathbf{B}}_8\widehat{\mathbf{u}}_8 \tag{5.13e}$$

For the purpose of simpiflication only subproblem 8 displays the relaxation, since the process is analog for all subproblems. Equation (5.13e) can be relaxed with slack variables, shown in Equations (3.6), making the use of the dual algorithm possible, allowing that the coupled variables, $\widehat{\mathbf{B}}_{87}\mathbf{u}_7$, to be handled.

$$P_8 : \min \frac{1}{2}\widehat{\mathbf{u}}_8'\widehat{\mathbf{H}}_8\widehat{\mathbf{u}}_8 + \widehat{\mathbf{g}}_8'\widehat{\mathbf{u}}_8 + \widehat{c}_8 \tag{5.14a}$$

$$\text{s.t. :}\widehat{\mathbf{u}}_8^{\min} \leq \widehat{\mathbf{u}}_8 \leq \widehat{\mathbf{u}}_8^{\max} \tag{5.14b}$$

$$\widehat{\mathbf{M}_8\mathbf{u}_8} \leq C \tag{5.14c}$$

$$-(\widehat{\mathbf{B}}_8\widehat{\mathbf{u}}_8 + \widehat{\mathbf{A}}_8\widehat{\mathbf{x}}_8 + \widehat{\mathbf{B}}_{87}\widehat{\mathbf{u}}_7) + \widehat{\mathbf{s}}_8 \leq 0 \tag{5.14d}$$

$$\widehat{\mathbf{B}}_8\widehat{\mathbf{u}}_8 - \widehat{\mathbf{x}}_8^{\max} + \widehat{\mathbf{A}}_8\widehat{\mathbf{x}}_8 + \widehat{\mathbf{B}}_{87}\widehat{\mathbf{u}}_7 + \widehat{\mathbf{s}}_8 \leq 0 \tag{5.14e}$$

Equations (5.14d) and (5.14e) can be dualized by the Algorithm 4 seen in Chapter 4.

This process is similar for every other subsystem in the centralized problem.

5.4 COMPUTATIONAL ANALYSIS

The problem presented in the above section is chosen as a representative system for the analysis, its objective is simply empty the already saturated queues of vehicles. The problem is implemented on the Matlab environment due to its tools and the fact that *Matlab* allows matrix manipulations, plotting of functions and data and implementation of algorithms, facilitating not only the model but the algorithm implementations.

The network links have capacity bounds, meaning that the states are subject to bound constraints $0 \leq x_{i,j} \leq x_{i,j}^{\max} = 200$ vehicles for all $(i,j) \in \mathcal{E}$. Control signals are also bounded, whereby

the sum of the green times at a crossing i must not exceed cycle time, $\sum_{j \in I(i) \setminus \{i\}} u_{i,j} \leq C = 120$, with $u_{i,j}$ being the green time assigned to link (j, i) and $\mathbf{u}_i = (u_{i,j} : j \in I(i) \setminus \{i\})$. Thus the control signals are effective green times, meaning that green times will be assigned to a queue provided that the corresponding state variable remains feasible, *i.e.* nonnegative and within road capacity. Upper and lower bounds are also imposed on control signals, so that one of the roads in a crossing cannot be allocated all of the cycle time for itself, $0 \leq u_{i,j} \leq 80$.

The matrices $\widehat{A}_i$ and $\widehat{B}_{i,j}$ result from the application of the Webster's procedure presented in [13]. The cost matrices were defined similarly but ensuring $Q_i \succeq 0$ and $R_i \succ 0$. The prediction horizon was $T_h = 3$.

Three cases were considered for computational analysis:

1. the standard QP algorithm available in Matlab.

2. the centralized Augmented Lagrangian (CAL) given by Algorithm 4 in which the subproblems LM_k are solved with an embedded solver in Matlab called *quadprog*.

3. the distributed Augmented Lagrangian (DAL) given by Algorithm 4, however the subproblems $\{LM_{k,i}\}$ are solved as constrained versions of the centralized problem.

These cases are also considered in [31].

The analyses consisted in solving problem P for 10 different initial conditions, obtained by varying the queue states from 110 to 200 vehicles randomly.

The initial Lagrange multipliers were set at 1, the initial penalty factor was $\mu^{(0)} = 1$ and updated with the chosen rule $\mu^{(k+1)} = 0.8\mu^{(k)}$, and the initial solution $\widehat{\mathbf{v}}^{\mathrm{s}}$ was set to 0 with the slack variables calculated accordingly. The distributed Augmented Lagrangian was emulated using a single processor, with agent synchronization and communication performed through shared memory. Although the agents in each of the sets $\{1, 7\}, \{2, 4, 6\}$, and $\{3, 5, 8\}$ can iterate in parallel, our implementation ran one agent at a time for the sake of simplification. The distributed solution of $\{LM_i\}$ was chosen to be limited to 100 inner iterations, counting once every time all agents solved their local problems. The convergence condition was established on the mean squared error of the 2 latest control iterations with $\tau = 10^{-3}$, while the outer iterations use the mean squared error on the latest 2 iterations of $\hat{v}^k$ with $\tau = 10^{-3}$ and also

have a maximum of 100 iterations. The parameters of distributed algorithm were as in the centralized Algorithm 4, with the same initial conditions, and updating rules for Lagrange multipliers and penalty factor. All 3 cases solve the problem for only the first cycle iteration.

Table 5.1: Comparative analysis of the Centralized Augmented Lagragian.

	QP		CAL		
Instance	Time(s)	Obj(10^5)	Time(s)	Obj(10^5)	Dist(%)
1	0.0781	-2.3950	0.3125	-2.3950	0.0
2	0.0625	-2.7507	0.0938	-2.7507	0.0
3	0.1094	-3.1137	0.1094	-3.1137	0.0
4	0.0938	-3.4829	0.1719	-3.4829	0.0
5	0.1094	-3.8563	0.2188	-3.8563	0.0
6	0.0625	-4.2331	0.1250	-4.2331	0.0
7	0.1406	-4.6133	0.0938	-4.6133	0.0
8	0.3281	-4.9968	0.1094	-4.9968	0.0
9	0.1406	-5.3834	0.1094	-5.3834	0.0
10	0.5313	-5.7729	0.1250	-5.7729	0.0
Mean	0.1437	-4.0598	0.1516	-4.0598	0.0

Table 5.2: Comparative analysis of the Distributed Augmente Lagragian.

	QP		DAL		
Instance	Time(s)	Obj(10^5)	Time(s)	Obj(10^5)	Dist(%)
1	0.0781	-2.3950	4.2500	-2.3038	3.8103
2	0.0625	-2.7507	3.7813	-2.7065	1.6095
3	0.1094	-3.1137	4.2813	-3.0881	0.8210
4	0.0938	-3.4829	3.3594	-3.4785	0.1272
5	0.1094	-3.8563	3.0625	-3.8563	0.0
6	0.0625	-4.2331	3.6406	-4.2331	0.0
7	0.1406	-4.6133	3.3906	-4.6133	0.0
8	0.3281	-4.9968	3.2500	-4.9968	0.0
9	0.1406	-5.3834	3.0000	-5.3834	0.0
10	0.5313	-5.7729	3.0625	-5.7729	0.0
Mean	0.1437	-4.0598	3.3969	-3.9988	0.6368

Tables 5.1 and 5.2 present the results where each column gives the objective, CPU time (seconds) and distance of the ob-

jective, in percentage, of each 10 independent runs of LDNs and their mean. The results show that the iterates produced by the distributed Augmented Lagrangian method induced an objective GAP of approximately 0.6%, which can be attributed to the algorithm for $LM_{k,i}$, which is a restricted version of the centralized algorithm obtained by fixing the variables of the neighboring and remote agents. The time of convergence is well inbound the maximum feasible, which is the cycle time. If the time taken to complete 1 the problem for the next period of time is greater than the cycle time, the algorithm becomes impractical, since it is necessary to know the green times in the next cycle before applying it. The gap is inside the expectations, this difference is not significant for the result.

The next analyses consisted in solving problem P for an simulation horizon bigger than the prediction horizon, thus fully implementing the sliding horizon according with MPC: the MPC problem is solved for the prediction horizon in every cycle, but only the controls for the current cycle are input to the network; at the next cycle, the network state is measured, the prediction horizon moved forward, and the process is repeated.

The cost along the simulation horizon is computed for the three cases, calculating the stage cost $\mathbf{x}_i'\mathbf{Q}\mathbf{x}_i + \mathbf{u}_i'\mathbf{R}\mathbf{u}_i$ at each cycle. The simulation horizon has length of 6, 5 instances, varying the initial conditions from 160 to 200 and all the other settings are the same as the previous experiments.

Table 5.3: Comparative analysis over a simulation period.

	QP	CAL		DAL	
Inst.	Cost(10^5)	Cost(10^5)	Dist(%)	Cost(10^5)	Dist(%)
1	4.8717	4.8717	0.0	5.5535	13.99
2	5.9477	5.9477	0.0	6.6148	11.21
3	7.1661	7.1661	0.0	7.7754	8.50
4	8.5427	8.5427	0.0	9.0506	5.94
5	10.0848	10.0848	0.0	10.5062	4.17
Mean	7.3226	7.3226	0.0	7.9001	7.89

Figure 5.4 presents the results of the three simulations for a network that begins with full queues (200 vehicles) and without vehicle arrivals during simulation. The results show that the three algorithms induce nearly the same cost for the network, a more demanding stopping criterion would be needed for the distributed algorithm to yield better solutions.

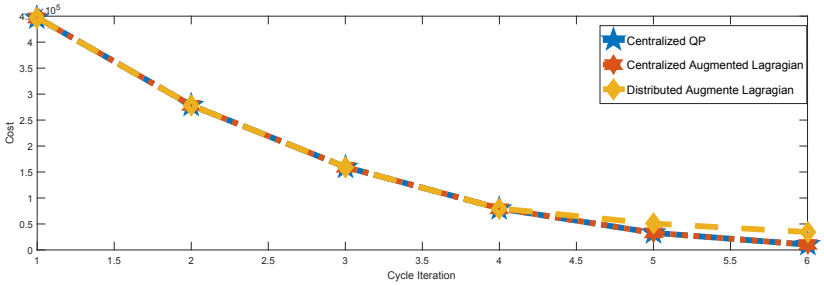

Figure 5.4: Cost along simulation horizon.

5.5 SIMULATION ANALYSIS

Aimsun is an integrated transport modeling software, developed and marketed by *TSS - Transport Simulation Systems* based in Barcelona, Spain.

Aimsun is a traffic modeling software that allows you to model anything from a single bus lane to an entire region with thousands of licensed users in government agencies, consultancies and universities all over the world. *Aimsun* stands out for the exceptionally high speed of its simulations and for fusing travel demand models, static and dynamic traffic assignment with mesoscopic, microscopic and hybrid simulation, all within a single software application. It allows the user to carry out traffic operations assessments of any scale and complexity. It is used to improve road infrastructure, reduce emissions, cut congestion and design urban environments for vehicles and pedestrians.

Figure 5.5 shows the screen of the software. On the left there are tools that aid the modeling of the network while on the right is the input of data, such as traffic demand, control plan, type of vehicles, etc.

The model is the same as the previous section, but it starts with no vehicles in queue, instead the arrival of vehicles are given by the sources outside of the model. Two different controls strategies are applied to the model, a centralized QP MPC and a fixed control strategy. The MPC is programmed in *Python* and is added to the *Aimsun* environment by its own API. The fixed control strategy can be implemmented on *Aimsun*, by fixing the green-time to the desired value, which in this case is proportional to the income of

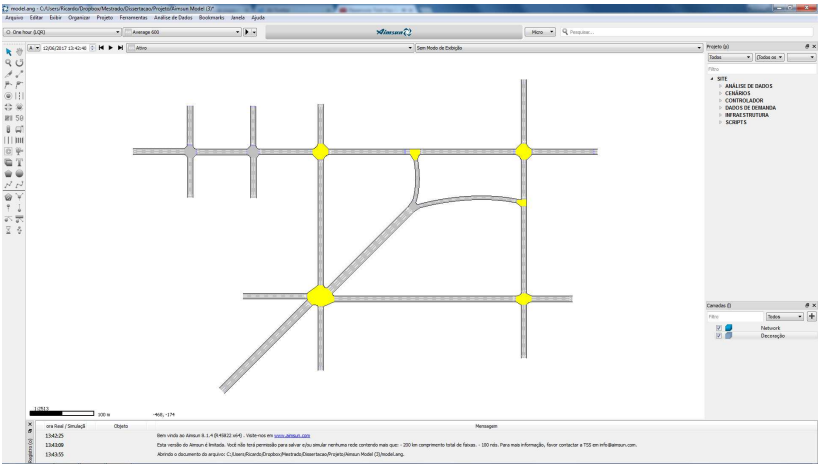

Figure 5.5: Aimsun screenshot.

vehciles in the first hour for the border agents and equally divide for the inner agents.

The model is simulated for a period of two hours, for the first hour of simulation the number of vehicles per hour in each agent is: for Agent 1, [900, 1300, 800], for Agent 4, [900, 700], for Agent 7 and 8, [800]. In the second hour, the arrivals are increased to highlight the difference between the fixed control strategy and the MPC, the arrival of vehicles per hour in each agent is: for Agent 1, [3000, 2500, 2000], for Agent 4, [1000, 2000], for Agent 7, [1600] and for Agent 8, [1500].

Tables 5.4 and 5.5 show the results of the 10 runs. *Flow* shows how many vehicles have passed through the network per hour. *Avg. Queue* denotes the avarege queue of the network. *Delay Time* represents the avarege delayed time a car suffers, in seconds by every kilometer. *Downtime* represents the avarege time, in seconds, a car is stopped by every kilometer. It is notable that the fixed control has an inferior performance, since it can not adapt to the sudden increase of arrivals of vehicles in the system, whereas the MPC control has that ability and achieves the better performance.

Table 5.4: Aimsun traffic statistics induced by the MPC strategy.

	MPC			
Instance	Flow	Avg. Queue	Delay Time	Downtime
1	8532.50	209.15	218.16	194.59
2	8585.50	211.58	220.05	195.99
3	8479	210.86	219.61	195.94
4	8466	213.64	221.04	197.11
5	8500	211.73	220.12	196.19
6	8435.50	207.44	217.83	194.22
7	8430	213.02	221.63	198.02
8	8505.50	220.57	226.61	202.50
9	8497	218.19	224.29	200.11
10	8449	215.48	224.73	200.71
Mean	8488	213.17	221.41	197.54

Table 5.5: Aimsun traffic statistics induced by the Fixed Control strategy.

	Fixed Control			
Instance	Flow	Avg. Queue	Delay Time	Downtime
1	8197.50	254.27	245.60	221.30
2	8157.50	263.20	252.04	227.56
3	8100	254.58	247.03	223.08
4	8110	260.30	251.74	227.34
5	8124	254.99	246.05	221.72
6	8153.50	240.42	234.91	210.86
7	8078	250.39	244.94	220.99
8	8157	260.85	251.89	227.46
9	8149.50	254.75	246.33	222.27
10	8063	260.18	252.32	228.02
Mean	8129	255.39	247.29	223.06

5.6 SUMMARY

This chapter presented the application of this dissertation, the urban traffic control. A traffic flow model is presented before, so it becomes clear to the reader the phisical representation of every variable. The problem is explicitly decomposed to reiterate on the advantages of the decentralized algorithm, the possibility to handle coupled state constraints. The problem is solved, in the *Matlab* environment, by three algorithms, their results are shown and

compared. *Aimsun* is presented and utilized as a form to validate the experiments, by comparing a MPC strategy and a Fixed Control strategy.

6 CONCLUSION

This dissertation has presented a dual algorithm based on the augmented Lagragian, showcasing its equations and mathematical proporties. A gradient projection method is introduced as a means to solve the inner problem of the dual algorithm and a simple example is demonstrated to better the reader's understanding.

A useful tool is presented in this dissertation, linear dynamic networks, which made possible the formulation of the problem in an easy way, and an even easier task to decompose the problem in a number of subproblems, which is applied to this dissertation's application, the urban traffic control. The algorithm is used to solve an urban traffic control problem, with the objective to minimize the queues of the network. The model is explained and three different strategies are utilized to solve it, a centralized quadratic programing algorithm, a centralized augmented Lagragian algorithm and the distributed augmented Lagragian algorithm. All three strategies are compared within one cycle of optimization. The results show that the distributed algorithm has an acceptable time of computation and the objective is sufficiently close to what was obtained with the others algorithms. The model is then simulated along a sliding horizon and compared amongst the strategies. The costs along the simulation horizon are nearly the same.

The model is implemented in the traffic modeling software *Aimsun* where it is simulated through two hours of two different traffic demands. In this environment the model is submitted to two strategies, a fixed control and a MPC. The results are as expected, the MPC has a better performance, since the fixed control is unable to adapt to a new rate of incoming vehicles.

Future work will be geared towards the implementation of the distributed optimization algorithm following the gradient projection method, and using multiagent communication platform [32] for a true distributed implementation. The further understanding of the *Aimsun* enviroment, will allow for more complex cenaries simulations and full implementation of the distributed algorithm.

BIBLIOGRAPHY

1 DIAKAKI, C.; PAPAGEORGIOU, M.; ABOUDOLAS, K. A multivariable regulator approach to traffic-responsive network-wide signal control. *Control Engineering Practice*, v. 10, n. 2, p. 183–195, 2002.

2 SANDELL, N. R. et al. Survey of decentralized control methods for large scale systems. *IEEE Transactions on Automatic Control*, v. 23, p. 108–128, 1978.

3 CAMACHO, E. F.; BORDONS, C. *Model Predictive Control*. [S.l.]: Springer-Verlag, 2004.

4 SCATTOLINI, R. Architectures for distributed and hierarchical model predictive control - a review. *Journal of Process Control*, v. 19, n. 5, p. 723–731, 2009.

5 NECOARA, I.; NEDELCU, V.; DUMITRACHE, I. Parallel and distributed optimization methods for estimation and control in networks. *Journal of Process Control*, v. 21, p. 756–766, 2011.

6 DIAKAKI, C. *Integrated Control of Traffic Flow in Corridor Networks*. Tese (Doutorado), 1999.

7 CAMPONOGARA, E.; SCHERER, H. F. Distributed optimization for MPC of linear dynamic networks with control-input and output constraints. *IEEE Transactions on Automation Science and Engineering*, v. 8, n. 1, p. 233–242, 2011.

8 BERTSEKAS, D. *Nonlinear Programming*. [S.l.]: Athena Scientific, 1995.

9 GOLDSTEIN, A. A. Convex Programming in Hilbert Space. *Bulletin of the American Mathematical Society*, v. 70, n. 5, p. 709–710, 1964.

10 LEVITIN, E.; POLYAK, B. Constrained minimization methods. *USSR Computational Mathematics and Mathematical Physics*, v. 6, n. 5, p. 1–50, 1966.

11 BOOM, T. J. J. van den; BACKX, T. *Model predictive control*. [S.l.: s.n.], 2010. 290 p.

12 DOAN, M. D.; KEVICZKY, T.; SCHUTTER, B. D. An iterative scheme for distributed model predictive control using Fenchel's duality. *Journal of Process Control*, v. 21, p. 746–755, 2011.

13 OLIVEIRA, L. B. de; CAMPONOGARA, E. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, v. 18, n. 1, p. 120–139, 2010.

14 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: Systems and Humans*, v. 39, n. 6, p. 1331–1338, 2009.

15 NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. [S.l.]: Springer, 1999.

16 BERTSEKAS, D. *Constrained optimization and Lagrange multiplier methods*. [S.l.]: Athena Scientific, 1982. 410 p.

17 BAZARAA, M.; SHERALI, H.; SHETTY, C. *Nonlinear Programming: Theory and Algorithms*. [S.l.]: John Wiley & Sons, 2006.

18 CAMPONOGARA, E.; LIMA, M. L. Distributed optimization for MPC of linear networks with uncertain dynamics. *IEEE Transactions on Automatic Control*, v. 57, n. 3, p. 804–809, 2012.

19 SOUZA, F. A. de et al. Distributed MPC for urban traffic networks: A simulation-based performance analysis. *Optimal Control Applications and Methods*, v. 36, p. 353–368, 2015.

20 KATEBI, M. R.; JOHNSON, M. a. Predictive control design for large-scale systems. *Automatica*, v. 33, n. 3, p. 421–425, 1997.

21 ACAR, L. Some examples for the decentralized receding horizon control. In: *Proc. 31st Conf. Decision and Control*. [S.l.: s.n.], 1992. p. 1356–1359.

22 IKEDA, M.; SILJAK, D. D.; WHITE, D. E. An inclusion principle for dynamic systems. *IEEE Transactions on Automatic Control*, v. 29, n. 3, p. 244–249, 1984.

23 JAMOOM, M. B.; FERON, E.; MCCONLEY, M. W. Optimal distributed actuator control grouping schemes. In: IEEE. *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*. [S.l.], 1998. v. 2, p. 1900–1905.

24 ALESSIO, A.; BEMPORAD, A. Decentralized model predictive control of constrained linear systems. *Proceedings European Control Conference*, p. 2813–2818, 2007.

25 GU, D. A differential game approach to formation control. *IEEE Transactions on Control Systems Technology*, v. 16, n. 1, p. 85–93, 2008.

26 BOYD, S. et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, v. 3, n. 1, p. 1–122, 2010.

27 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.

28 PAPAGEORGIOU, M. Automatic control methods in traffic and transportation. In: *Proceedings of the NATO Advanced Study Institute on Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*. [S.l.: s.n.], 1998. p. 46–83.

29 DINOPOULOU, V.; DIAKAKI, C.; PAPAGEORGIOU, M. Simulation investigations of the traffic-responsive urban control strategy TUC. In: *IEEE Proc. Intelligent Transportation Systems Conf.* [S.l.: s.n.], 2000. p. 458–463.

30 ABOUDOLAS, K. et al. A rolling-horizon quadratic-programming approach to the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, v. 18, n. 5, p. 680–694, 2010.

31 CAMPONOGARA, E.; SILVA, R. S. da; AGUIAR, M. A. S. de. A distributed dual algorithm for distributed mpc with application to urban traffic control. In: *To appear in 1st IEEE Conference on Control Technology and Applications*. [S.l.: s.n.], 2017.

32 HÜBNER, J. F.; WOOLDRIDGE, M.; BOLDRINI, R. F. *Programming Multi-Agent Systems in AgentSpeak Using Jason*. [S.l.]: John Wiley & Sons, 2007.