Chapter 12
Distributed Optimization for MPC of Linear Dynamic Networks

E. Camponogara

Abstract This chapter presents existing models and distributed optimization algorithms for model predictive control (MPC) of linear dynamic networks (LDNs). The models consist of networks of subsystems with deterministic and uncertain dynamics subject to local and coupling constraints on the control and output signals. The distributed optimization algorithms are based on gradient-projection, subgradient, interior-point, and dual strategies that depend on the nature of the couplings and constraints of the underlying networks. The focus will be on a class of LDNs in which the dynamics of the subsystems are influenced by the control signals of the upstream subsystems with constraints on state and control variables. A distributed gradient-based algorithm is presented for implementing an interior-point method distributively with a network of agents, one for each subsystem.

12.1 Introduction

Dynamic networks are systems of subsystems that can model large, geographically distributed systems such as urban traffic networks, sewage systems, and petrochemical plants. A dynamic network (DN) consists of a graph whose nodes represent dynamic subsystems, each characterized by a local state that evolves dynamically depending on its control signals and the signals from other subsystems. When the differential equations governing the dynamics of the subsystems are linear the network is called linear dynamic network (LDN).

In [3] a distributed model predictive control (DMPC) framework was developed for operating LDNs in which the dynamics of the subsystems depend on their local control signals and state, but only on the control inputs of the upstream subsystems

E. Camponogara(✉)
Department of Automation and Systems Engineering, Federal University of Santa Catarina, Florianópolis, Brazil
e-mail: camponog@das.ufsc.br

J. M. Maestre and R. R. Negenborn (eds.), *Distributed Model Predictive Control Made Easy*, Intelligent Systems, Control and Automation: Science and Engineering 69,
DOI: 10.1007/978-94-007-7006-5_12,

while being subject to constraints on local controls. This work was extended by accounting for uncertainties in the dynamic models and by proposing a distributed subgradient optimization algorithm for implementing DMPC [4]. In [5] constraints on the state variables are admitted which induce algebraic couplings between subsystems in addition to the dynamic interconnections. Distributed algorithms based on the dual strategy [10] and the interior-point method [5] have been proposed in the literature for handling such algebraic couplings. The focus of this chapter is on the latter strategy.

A number of other distributed optimization strategies have also appeared in the literature. In [11] the solution of optimization problems within a distributed model predictive control framework is addressed, whereby Lagrangian duality is applied to handle coupling variables among neighboring agents. In [14] a cooperation-based iterative process was developed for optimization and control of linear systems with constrains on local control signals. In [15] distributed primal-dual subgradient algorithms are developed for general convex optimization with multi-agent systems, considering global objectives and constraints.

This chapter is organized as follows. Section 12.2 introduces three classes of linear dynamic networks with focus on LDNs with coupling state constraints. Section 12.3 presents a distributed optimization algorithm for solving quadratic programs that arise from the model predictive control (MPC) of LDNs with state constraints. Section 12.4 briefly discusses some applications of distributed MPC on LDNs.

12.2 Linear Dynamic Networks

A dynamic network consists of a directed graph with nodes modeling dynamic subsystems and arcs representing the direct influence between the subsystems. Dynamic networks can serve as models for geographically distributed systems such as urban traffic networks, electric power systems, and water pipelines. The structure of a DN is represented by a coupling graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ with $\mathcal{N} = \{1, \ldots, n\}$ being the set of nodes and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ being the set of arcs.

Each node $i \in \mathcal{N}$ is characterized by a local state vector $\mathbf{x}_i$ of dimension $n_{\mathbf{x}_i}$ and a local control-input vector $\mathbf{u}_i$ of dimension $n_{\mathbf{u}_i}$. The state $\mathbf{x}_i$ of subsystem i evolves dynamically depending on the local state and controls and also on the state and controls of its *input neighborhood* $\mathcal{I}(i) = \{j : (j, i) \in \mathcal{E}\} \cup \{i\}$.

To illustrate these concepts consider the small urban traffic network and the corresponding coupling graph depicted in Fig. 12.1. In this dynamic system the subsystems correspond to intersections, the state of an intersection consists of the number of vehicles in the roads leading to it, and the control signals are the green times assigned to each stage which is assumed to be one for each state for simplicity. For instance, the state of subsystem 3 is $\mathbf{x}_3 = [x_{3,1}\ x_{3,4}]^{\mathrm{T}}$ and its control vector is $\mathbf{u}_3 = [u_{3,1}\ u_{3,4}]^{\mathrm{T}}$. The input neighborhood of this subsystem is $\mathcal{I}(3) = \{1, 3, 4\}$. In such traffic systems, the discharge of queues at the upstream subsystems becomes the arrival of vehicles in the roads that correspond to the state of subsystem 3.

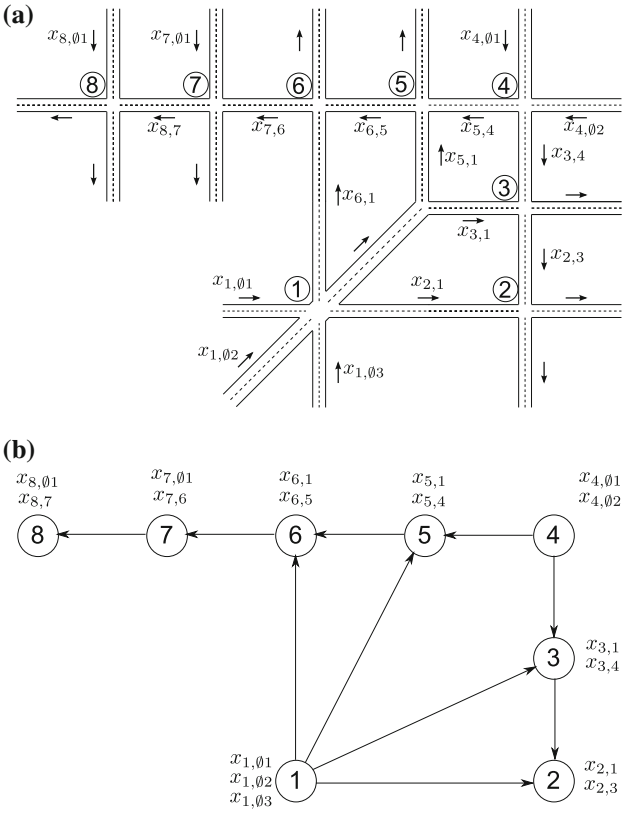

Fig. 12.1 Dynamic network of an urban traffic system. **a** Urban traffic system with 8 intersections. **b** Graph of the linear dynamic network

The potential to represent actual systems and the complexity of the control strategy will depend greatly on the nature of the differential equations governing the dynamics of the nodes, which in general are given by:

$$\mathbf{x}_i(k+1) = \mathbf{F}_i(\mathbf{x}_i^I(k), \mathbf{u}_i^I(k)). \tag{12.1}$$

where $\mathbf{x}_i^I = [\mathbf{x}_j^{\mathrm{T}} : j \in \mathcal{I}(i)]^{\mathrm{T}}$ and $\mathbf{u}_i^I = [\mathbf{u}_j^{\mathrm{T}} : j \in \mathcal{I}(i)]^{\mathrm{T}}$ are vectors collecting all of the states and control inputs influencing the state of subsystem i, respectively. Previous works have developed optimization and control frameworks assuming specialized structures for the dynamic equation (12.1) and constraints. Frameworks for three general classes of dynamic networks are presented below.

1. *Dynamic dependency on upstream control signals, subject to constraints on local control signals*

A first work on dynamic networks was inspired by the store-and-forward modeling of urban traffic flow [9]. Traffic flow can be modeled with linear dynamic equations depending on the control signals from the input neighborhood and local state, being expressed for a subsystem $i \in \mathcal{N}$ as:

$$\mathbf{x}_i(k+1) = \mathbf{A}_i \mathbf{x}_i(k) + \sum_{j \in \mathcal{I}(i)} \mathbf{B}_{i,j} \mathbf{u}_j(k) \tag{12.2}$$

where $\mathbf{A}_i \in \mathbb{R}^{n_{\mathbf{x}_i} \times n_{\mathbf{x}_i}}$ and $\mathbf{B}_{i,j} \in \mathbb{R}^{n_{\mathbf{x}_i} \times n_{\mathbf{u}_j}}$ are matrices of suitable dimensions. For the particular case of traffic flow modeling with store-and-forward, the matrix $\mathbf{A}_i$ is the identity while the $\mathbf{B}_{i,j}$ matrices model the discharges from and arrivals in the vehicle queues defining the state of intersection i. Considering intersection 3 the store-and-forward approach defines these matrices in terms of physical characteristics of the traffic network as:

$$\mathbf{B}_{3,1} = T_3 \begin{bmatrix} \rho_{3,1,\emptyset 1} \cdot \frac{S_{1,\emptyset 1}}{C_1} & \rho_{3,1,\emptyset 2} \cdot \frac{S_{1,\emptyset 2}}{C_1} & \rho_{3,1,\emptyset 3} \cdot \frac{S_{1,\emptyset 3}}{C_1} \\ 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{B}_{3,3} = T_3 \begin{bmatrix} -\frac{S_{3,1}}{C_3} & 0 \\ 0 & -\frac{S_{3,4}}{C_3} \end{bmatrix},$$

$$\mathbf{B}_{3,4} = T_3 \begin{bmatrix} 0 & 0 \\ \rho_{3,4,\emptyset 1} \cdot \frac{S_{4,\emptyset 1}}{C_4} & \rho_{3,4,\emptyset 2} \cdot \frac{S_{4,\emptyset 2}}{C_4} \end{bmatrix}$$

where T_3 is the sample time (in seconds), $S_{i,j}$ is the saturation flow on the link/road from intersection j to i (in vehicles per second), $\rho_{m,i,j}$ is the rate at which vehicles from link j to i enter link i to m, and C_i (in seconds) is the cycle time of junction i with a cycle being composed by a cyclic sequence of stages which are specific traffic light configurations.

This class of systems is referred to as linear dynamic networks (LDNs) in view of the nature of linear dynamics. The controls of any subsystem i can be subject to linear constraints in the form:

$$\mathbf{D}_i \mathbf{u}_i(k) \leq \mathbf{d}_i, \tag{12.3a}$$

$$\mathbf{E}_i \mathbf{u}_i(k) = \mathbf{e}_i. \tag{12.3b}$$

In the case of the traffic system these constraints are used to force the green times to add up to cycle time and impose bounds on these control signals.

A distributed MPC framework based on gradient projection [1] was developed for controlling the LDN with a network of agents, one for each subsystem, which minimizes a quadratic cost function on states and control inputs:

$$J = \min \sum_{i \in \mathcal{N}} \sum_{k=1}^{N_p} \frac{1}{2} \mathbf{x}_i(k)^T \mathbf{Q}_i \mathbf{x}_i(k) + \sum_{i \in \mathcal{N}} \sum_{k=0}^{N_c-1} \frac{1}{2} \mathbf{u}_i(k)^T \mathbf{R}_i \mathbf{u}_i(k). \quad (12.4)$$

Following a synchronization protocol that allows only uncoupled agents to iterate simultaneously, the gradient-projection algorithm was shown to converge to an optimal solution to the quadratic program of centralized MPC [3].

2 *Dynamic dependency on upstream control signals with uncertainty, subject to constraints on local control signals*
The previous class of LDNs was recently generalized to consider uncertainty on the dynamic equations:

$$\mathbf{x}_i(k+1) = \mathbf{A}_i(k)\mathbf{x}_i(k) + \sum_{j \in \mathcal{I}(i)} \mathbf{B}_{i,j}(k)\mathbf{u}_j(k) \quad (12.5)$$

where $(\mathbf{A}_i(k), \mathbf{B}_{i,j}(k) : j \in \mathcal{I}(i)) \in \Omega_i$ with Ω_i being a set of possible dynamic realizations of subsystem i. For an urban traffic network, Ω_i can model uncertainties related to the conversion rates and traffic patterns. The LDN is also subject to linear constraints of the form of Eq. (12.3) imposed on the control signals.
A robust MPC strategy would hedge against the worst-case scenario by optimizing the objective:

$$J = \min \sum_{i \in \mathcal{N}} \max \left\{ \sum_{k=1}^{N_p} \frac{1}{2} \mathbf{x}_i(k|\boldsymbol{\omega})^T \mathbf{Q}_i \mathbf{x}_i(k|\boldsymbol{\omega}) + \sum_{k=0}^{N_c-1} \frac{1}{2} \mathbf{u}_i(k)^T \mathbf{R}_i \mathbf{u}_i(k) : \boldsymbol{\omega} \in \Omega_i(N_p) \right\} \quad (12.6)$$

where $\Omega_i(N_p)$ is the set of all possible dynamic trajectories for subsystem i during a horizon of length N_p, $\boldsymbol{\omega}$ is a particular trajectory, and $\mathbf{x}_i(k|\boldsymbol{\omega})$ is the predicted state at time t_k assuming a dynamic trajectory $\boldsymbol{\omega}$.
Because the objective function to be minimized in (12.6) is nondifferentiable, derivative-based approaches such as gradient projection could not be directly applied. Thus a distributed subgradient algorithm was developed for solving the optimization problem with a network of distributed agents [4]. The algorithm and synchronization protocol are similar to the ones developed for the differentiable case, differing in the iterative process followed by the agents which implement projections onto the feasible space of iterates produced by taking steps along the subgradient. This distributed algorithm was also shown to produce a sequence of iterates converging to the optimum.

3. *Dynamic dependency on upstream states and control signals, subject to constraints on local states and control signals*
The above LDN models were extended to allow constraints on state variables. These constraints introduce hard couplings among the subsystems and the control

agents, thereby rendering distributed MPC more challenging. The gradient and subgradient projection strategies will fail to converge to the optimal solution in the presence of such constraints. In this generalized class of LDNs the state of a subsystem i depends on the state of its upstream subsystem, leading to the following dynamic equation:

$$\mathbf{x}_i(k+1) = \sum_{j \in \mathcal{I}(i)} \left(\mathbf{A}_{i,j} \mathbf{x}_j(k) + \mathbf{B}_{i,j} \mathbf{u}_j(k) \right). \tag{12.7}$$

Each subsystem i is also subject to constraints on state and control variables:

$$\mathbf{C}_i \mathbf{x}_i(k) \leq \mathbf{c}_i, \tag{12.8a}$$
$$\mathbf{D}_i \mathbf{u}_i(k) \leq \mathbf{d}_i, \tag{12.8b}$$
$$\mathbf{E}_i \mathbf{u}_i(k) = \mathbf{e}_i. \tag{12.8c}$$

In the application to urban traffic networks the constraint (12.8a) can be used to limit the number of vehicles in each road link, while the remaining ones enforce bounds on the green-time signals and cycle time.

Because $\mathbf{x}_i$ is now a function of $\mathbf{x}_i^I$, the state constraint (12.8a) is directly affected by all the agents that belong to the input neighborhood of subsystem i. The objective function for this LDN is identical to objective (12.4). In [5], a distributed MPC framework was proposed to control such LDNs whereby a network of agents implements an interior-point method, solving a sequence of unconstrained approximation problems with a distributed gradient-descent algorithm.

From now on the focus of this chapter is on this class of LDNs for being a generalization of the others. Actually, the nondifferentiable objective function (12.6) can be represented by a system of inequalities and thereby handled by a straightforward extension of the inequalities (12.8). The following section presents a distributed optimization framework for solving the MPC optimization problem with a network of agents, thereby implementing a distributed MPC strategy for this class of LDNs.

12.3 Distributed Optimization Framework

This section presents a distributed optimization algorithm for the class of LDNs in which constraints are imposed on the state of the subsystems. To keep the presentation simple but without losing generality, the states are assumed to be independent of the state of the upstream subsystems, namely:

$$\mathbf{x}_i(k+1) = \mathbf{A}_i \mathbf{x}_i(k) + \sum_{j \in \mathcal{I}(i)} \mathbf{B}_{i,j} \mathbf{u}_j(k). \tag{12.9}$$

Given the state of subsystem i at time t_k and the future control signals from the input neighborhood, the state at time t_{k+l} can be anticipated as

$$\mathbf{x}_i(k+l) = \mathbf{A}_i^l \mathbf{x}_i(k) + \sum_{t=1}^{l} \sum_{j \in \mathcal{I}(i)} \mathbf{A}_i^{l-1} \mathbf{B}_{i,j} \mathbf{u}_j(k+l-t). \tag{12.10}$$

This means that the future states of subsystem i depend on the input neighborhood regardless of the length of the prediction horizon. However, the predictions for the state of subsystem i will depend on subsystems that are farther away from the input neighborhood, extending outwards as the prediction horizon increases, when the subsystem state evolves according to Eq. (12.7) rather (12.9). For further details the reader can refer to [5].

The concern here is on the distributed solution at time t_k of the following MPC optimization problem for linear dynamic networks:

$$P(k): \ \min \ \sum_{i \in \mathcal{N}} \sum_{l=k+1}^{k+N_\mathrm{p}} \frac{1}{2} \mathbf{x}_i(l)^\mathrm{T} \mathbf{Q}_i \mathbf{x}_i(l) + \sum_{i \in \mathcal{N}} \sum_{l=k}^{k+N_\mathrm{c}-1} \mathbf{u}_i(l)^\mathrm{T} \mathbf{R}_i \mathbf{u}_i(l) \tag{12.11a}$$

$$\text{s.t.}: \ \text{for } i \in \mathcal{N},\ l = k, \ldots, k+N_\mathrm{p}-1:$$

$$\mathbf{x}_i(l+1) = \mathbf{A}_i \mathbf{x}_i(l) + \sum_{j \in \mathcal{I}(i)} \mathbf{B}_{i,j} \mathbf{u}_j(l), \tag{12.11b}$$

$$\mathbf{C}_i \mathbf{x}_i(l+1) \leq \mathbf{c}_i, \tag{12.11c}$$

$$\mathbf{D}_i \mathbf{u}_i(l) \leq \mathbf{d}_i, \tag{12.11d}$$

$$\mathbf{E}_i \mathbf{u}_i(l) = \mathbf{e}_i \tag{12.11e}$$

where the prediction and control horizons have the same length for convenience of mathematical development, i.e. $N_\mathrm{p} = N_\mathrm{c}$. The MPC strategy solves $P(k)$ at each sample time t_k, implementing only the control signals $\mathbf{u}_i(k)$ for the time interval $[t_k, t_{k+1}]$. Then the horizon is rolled forward at the next sample time, $P(k+1)$ is instantiated from time t_{k+1} until $t_{k+N_\mathrm{p}+1}$, and the process is repeated.

Let $\widetilde{\mathbf{u}}_i(k) = [\mathbf{u}_i(k+l)^\mathrm{T} : l = 0, \ldots, N_\mathrm{c}-1]^\mathrm{T}$ be the vector with the control predictions for subsystem i. Using Eq. (12.10) to express subsystem states as a function of control signals, the MPC optimization problem can be recast in terms of the control signals and the current state as follows:

$$P(k): \ \min \ \frac{1}{2} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{I}(i)} \sum_{l \in \mathcal{I}(i)} \widetilde{\mathbf{u}}_j(k)^\mathrm{T} \widetilde{\mathbf{H}}_{i,j,l} \widetilde{\mathbf{u}}_l(k) + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{I}(i)} \widetilde{\mathbf{g}}_{i,j}(k)^\mathrm{T} \widetilde{\mathbf{u}}_j(k) + \sum_{i \in \mathcal{N}} \widetilde{c}_i(k) \tag{12.12a}$$

$$\text{s.t.}: \ \text{for all } i \in \mathcal{N}:$$

$$\widetilde{\mathbf{C}}_i \left(\widetilde{\mathbf{A}}_i \mathbf{x}_i(k) + \sum_{j \in \mathcal{I}(i)} \widetilde{\mathbf{B}}_{i,j} \widetilde{\mathbf{u}}_j(k) \right) \leq \widetilde{\mathbf{c}}_i, \tag{12.12b}$$

$$\widetilde{\mathbf{D}}_i \widetilde{\mathbf{u}}_i(k) \leq \widetilde{\mathbf{d}}_i, \tag{12.12c}$$

$$\widetilde{\mathbf{E}}_i \widetilde{\mathbf{u}}_i(k) = \widetilde{\mathbf{e}}_i. \tag{12.12d}$$

where $\widetilde{\mathbf{H}}_{i,j,l}$, $\widetilde{\mathbf{A}}_i$, and $\widetilde{\mathbf{B}}_{i,j}$ are constant matrices, $\widetilde{\mathbf{g}}_{i,j}(k)$ is a constant vector, and $\widetilde{c}_i(k)$ is a constant obtained from the structure of the LDN and problem $P(k)$ defined in Eq. (12.11). The terms $\widetilde{\mathbf{g}}_{i,j}(k)$ and $\widetilde{c}_i(k)$ are functions of k because they depend on the initial state $\mathbf{x}_i(k)$ of the subsystems. Detailed procedures for computing these parameters are found in [3, 5]. Further, $\widetilde{\mathbf{C}}_i$, $\widetilde{\mathbf{D}}_i$, and $\widetilde{\mathbf{E}}_i$ are block diagonal matrices whose blocks are $\mathbf{C}_i$, $\mathbf{D}_i$, and $\mathbf{E}_i$ respectively, and $\widetilde{\mathbf{c}}_i$, $\widetilde{\mathbf{d}}_i$, and $\widetilde{\mathbf{e}}_i$ are vectors with stacked copies of $\mathbf{c}_i$, $\mathbf{d}_i$, and $\mathbf{e}_i$ respectively.

Problem $P(k)$ is further simplified by explicitly removing the linear dependencies induced by Eq. (12.12d). Let $\widetilde{\mathbf{u}}_i^{\dagger}$ be any solution for the system of linear equations (12.12d) and let $\Phi(\widetilde{\mathbf{E}}_i)$ be a basis for the null space of $\widetilde{\mathbf{E}}_i$. Then, replacing $\widetilde{\mathbf{u}}_i(k)$ with $\widetilde{\mathbf{u}}_i^{\dagger} + \Phi(\widetilde{\mathbf{E}}_i)\widehat{\mathbf{u}}_i(k)$ produces the following equivalent form of the problem:

$$\begin{aligned} P(k): \ \min \ & \frac{1}{2} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{I}(i)} \sum_{l \in \mathcal{I}(i)} \widehat{\mathbf{u}}_j(k)^{\mathrm{T}} \widehat{\mathbf{H}}_{i,j,l} \widehat{\mathbf{u}}_l(k) \\ & + \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{I}(i)} \widehat{\mathbf{g}}_{i,j}(k)^{\mathrm{T}} \widehat{\mathbf{u}}_j(k) + \sum_{i \in \mathcal{N}} \widehat{c}_i(k) \qquad (12.13a) \\ \text{s.t.}: \ & \text{for all } i \in \mathcal{N}: \\ & \sum_{j \in \mathcal{I}(i)} \widehat{\mathbf{B}}_{i,j} \widehat{\mathbf{u}}_j(k) \leq \widehat{\mathbf{c}}_i(k), \qquad (12.13b) \\ & \widehat{\mathbf{D}}_i \widehat{\mathbf{u}}_i(k) \leq \widehat{\mathbf{d}}_i. \qquad (12.13c) \end{aligned}$$

where the matrices $\widehat{\mathbf{H}}_{i,j,l}$, $\widehat{\mathbf{B}}_{i,j}$, and $\widehat{\mathbf{D}}_i$, vectors $\widehat{\mathbf{g}}_{i,j}(k)$, $\widehat{\mathbf{c}}_i(k)$, and $\widehat{\mathbf{d}}_i$, and constants $\widehat{c}_i(k)$ are easily obtained. In particular, $\widehat{\mathbf{H}}_{i,j,l} = \Phi(\widetilde{\mathbf{E}}_i)^{\mathrm{T}} \widetilde{\mathbf{H}}_{i,j,l} \Phi(\widetilde{\mathbf{E}}_i)$. The design of algorithms will be simplified by casting $P(k)$ in the form:

$$\begin{aligned} P(k): \ \min \ & f(\widehat{\mathbf{u}}(k)) \qquad (12.14a) \\ \text{s.t.}: \ & h_i(\widehat{\mathbf{u}}(k)) \leq 0, \ i \in \mathcal{H}, \qquad (12.14b) \end{aligned}$$

where $\widehat{\mathbf{u}}(k) = [\widehat{\mathbf{u}}_i(k)^{\mathrm{T}} : i \in \mathcal{N}]^{\mathrm{T}}$ is the vector with the control variables, $f : \mathbb{R}^{\widehat{n}} \to \mathbb{R}$ defines the objective, and $h_i : \mathbb{R}^{\widehat{n}} \to \mathbb{R}$, $i \in \mathcal{H}$, define the constraints given by (12.13b)–(12.13c) with $\widehat{n}$ being the dimension of $\widehat{\mathbf{u}}$. Actually, $f(\widehat{\mathbf{u}}) = \frac{1}{2}\widehat{\mathbf{u}}(k)^{\mathrm{T}} \widehat{\mathbf{H}} \widehat{\mathbf{u}}(k) + \widehat{\mathbf{g}}^{\mathrm{T}} \widehat{\mathbf{u}}(k) + \widehat{c}$ for suitable $\widehat{\mathbf{H}}$, $\widehat{\mathbf{g}}$, and $\widehat{c}$, and $h_i = \widehat{\mathbf{a}}_i^{\mathrm{T}} \widehat{\mathbf{u}}(k) - \widehat{b}_i$ for suitable $\widehat{\mathbf{a}}_i$ and $\widehat{b}_i$, $i \in \mathcal{H}$. Notice that $\widehat{\mathbf{H}} \succ 0$. The feasible set is $\Omega = \{\widehat{\mathbf{u}}(k) : h_i(\widehat{\mathbf{u}}(k)) \leq 0, \ i \in \mathcal{H}\}$, whereas the interior of the feasible set is $\bar{\Omega} = \{\widehat{\mathbf{u}}(k) : h_i(\widehat{\mathbf{u}}(k)) < 0, \ i \in \mathcal{H}\}$.

12.3.1 Distributed Modeling

The distributed solution of $P(k)$ starts with the decomposition into a set $\{P_i(k) : i \in \mathcal{N}\}$ of subproblems, $\{P_i(k)\}$ for short. The proposed decomposition is carried out based on the following definitions of relationships of agent i with agents controlling other subsystems:

- $\mathcal{O}(i) = \{j : i \in \mathcal{I}(j)\}$ is the *output neighborhood* which consists of the subsystems that are affected by the decisions at subsystem i;
- $\mathcal{C}(i) = \{j : \exists l \neq i, j, \text{ such that } i, j \in \mathcal{I}(l)\} \setminus (\mathcal{I}(i) \cup \mathcal{O}(i))$ is the *indirect neighborhood* which encompasses the subsystems j that do not influence nor are influenced by subsystem i, but both affect some other subsystem l;
- $\mathcal{N}(i) = (\mathcal{I}(i) \cup \mathcal{O}(i) \cup \mathcal{C}(i)) \setminus \{i\}$ is the *neighborhood* which comprises the subsystems that are coupled with subsystem i.

In the linear network of Fig. 12.2, subsystem 4 has $\mathcal{I}(4) = \{4\}$, $\mathcal{O}(4) = \{3, 4, 5\}$, $\mathcal{C}(4) = \{1\}$, and $\mathcal{N}(4) = \{1, 3, 5\}$.

A decomposition $\{P_i(k)\}$ is said to be *perfect* if each $P_i(k)$ is obtained from $P(k)$ by dropping from the objective all of the terms, and discarding all of the constraints, that do not depend on $\widehat{\mathbf{u}}_i(k)$. Models and algorithms for obtaining approximate decompositions are found in [6]. For a perfect decomposition, agent i's view of the dynamic network is divided in:

- *local variables*: the variables $\widehat{\mathbf{u}}_i(k)$ whose values are set by agent i;
- *neighborhood variables*: the vector $\widehat{\mathbf{w}}_i(k) = (\widehat{\mathbf{u}}_j(k) : j \in \mathcal{N}(i))$ of variables set by the agents in the neighborhood;
- *remote variables*: the vector $\widehat{\mathbf{r}}_i(k) = (\widehat{\mathbf{u}}_j(k) : j \notin \mathcal{N}(i) \cup \{i\})$ with all of the other variables.

Perfect decomposition allows for the decision variables of any agent i to be arranged as $\widehat{\mathbf{u}}(k) = [\widehat{\mathbf{u}}_i(k)^{\mathrm{T}}\, \widehat{\mathbf{w}}_i(k)^{\mathrm{T}}\, \widehat{\mathbf{r}}_i(k)^{\mathrm{T}}]^{\mathrm{T}}$ and for subproblem $P_i(k)$ to be cast in the form:

$$P_i(k, \widehat{\mathbf{w}}_i(k)) : \min_{\widehat{\mathbf{u}}_i(k)} \; f_i(\widehat{\mathbf{u}}_i(k), \widehat{\mathbf{w}}_i(k)) = \frac{1}{2}\widehat{\mathbf{u}}_i(k)^{\mathrm{T}}\widehat{\mathbf{H}}_i\widehat{\mathbf{u}}_i(k) + \widehat{\mathbf{g}}_i(k)^{\mathrm{T}}\widehat{\mathbf{u}}_i(k) + \hat{c}_i(k) \tag{12.15a}$$

$$\text{s.t.} : \sum_{l \in \mathcal{I}(j)} \widehat{\mathbf{B}}_{j,l}\widehat{\mathbf{u}}_l(k) \leq \widehat{\mathbf{c}}_j(k), \;\; j \in \mathcal{O}(i), \tag{12.15b}$$

$$\widehat{\mathbf{D}}_i\widehat{\mathbf{u}}_i(k) \leq \widehat{\mathbf{d}}_i, \tag{12.15c}$$

where:

$$\widehat{\mathbf{g}}_i(k) = \sum_{j \in \mathcal{O}(i)} \widehat{\mathbf{g}}_{j,i}(k) + \frac{1}{2}\sum_{j \in \mathcal{O}(i)} \sum_{l \in \mathcal{I}(j) \setminus \{i\}} (\widehat{\mathbf{H}}_{j,l,i}^{\mathrm{T}} + \widehat{\mathbf{H}}_{j,i,l})\widehat{\mathbf{u}}_l(k), \tag{12.16a}$$

$$\widehat{\mathbf{H}}_i = \sum_{j \in \mathcal{O}(i)} \widehat{\mathbf{H}}_{j,i,i}. \tag{12.16b}$$

The development of a distributed algorithm for solving $\{P_i(k)\}$ is simplified by recasting $P_i(k)$ in the following form:

$$P_i(k, \widehat{\mathbf{w}}_i(k)): \quad \min_{\widehat{\mathbf{u}}_i(k)} \quad f_i(\widehat{\mathbf{u}}_i(k), \widehat{\mathbf{w}}_i(k)) \tag{12.17a}$$

$$\text{s.t.}: \quad h_j(\widehat{\mathbf{u}}_i(k), \widehat{\mathbf{w}}_i(k)) \leq 0, \ j \in \mathcal{H}_i, \tag{12.17b}$$

where $\mathcal{H}_i = \{j \in \mathcal{H} : h_j$ is a function of $\widehat{\mathbf{u}}_i(k)\}$. Notice that h_j does not depend on $\widehat{\mathbf{r}}_i(k)$ under a perfect problem decomposition for all $j \in \mathcal{H}_i$. Further, h_j is not a function of $\widehat{\mathbf{u}}_i(k)$ for all $j \in \mathcal{H} \setminus \mathcal{H}_i$.

12.3.2 Distributed Algorithm

Iterative strategies in which an agent i reacts with a solution $\widehat{\mathbf{u}}_i^{(k+1)}$ for $P_i(k, \widehat{\mathbf{w}}_i^{(k)})$ to the decisions of its neighbors at iteration p, $\widehat{\mathbf{w}}_i^{(k)}$, may lead to undesirable behavior due to the coupling constraints given by Eq. (12.11c), as illustrated below.

Let $\mathbf{h}(k, \cdot) = [h_1 \cdots h_p]^{\mathrm{T}}$ denote the vector function with all of the constraints in $P(k)$. Then, $\Omega = \{\widehat{\mathbf{u}}(k) : \mathbf{h}(k, \widehat{\mathbf{u}}(k)) \leq \mathbf{0}\}$ is the feasible set. A vector $\widehat{\mathbf{u}}(k)^\star \in \Omega$ is a fixed point for the subproblem set $\{P_i(k)\}$ if and only if $\widehat{\mathbf{u}}_i(k)^\star$ is an optimal solution to $P_i(k, \widehat{\mathbf{w}}_i(k)^\star)$ for each $i \in \mathcal{N}$. Without constraints coupling the subsystems, $\widehat{\mathbf{u}}(k)^\star$ is a fixed point for $\{P_i(k)\}$ if and only if $\widehat{\mathbf{u}}(k)^\star$ satisfies first-order optimality conditions for $P(k)$, which in turn imply that $\widehat{\mathbf{u}}(k)^\star$ is globally optimal for $P(k)$ due to convexity [3]. The solving process of $\{P_i(k)\}$ can be thought of as a dynamic game with agents reacting to one another's decisions so as to improve their payoff [7].

However, the equivalence between a fixed point of $\{P_i(k)\}$ and an optimal solution to $P(k)$ does not hold in the presence of coupling constraints. As an illustration, consider the quadratic program:

$$P: \ \min \ f(u_1, u_2) = \frac{1}{2}u_1^2 - u_1u_2 + u_2^2 - 4u_2$$

$$\text{s.t.}: \ -u_1 + 2u_2 \leq 2, \quad 1 \leq u_1 \leq 7, \quad u_2 \geq \frac{1}{2}.$$

The solution of P_i over the decision variable u_i while holding $u_{(i \mod 2)+1}$ fixed is a process defined by the reaction function r_i of the agents as follows:

$$r_1(u_2) = \arg\min_{u_1}\{\frac{1}{2}u_1^2 - u_1u_2 : -u_1 + 2u_2 \leq 2, \ 1 \leq u_1 \leq 7\}$$

$$r_2(u_1) = \arg\min_{u_2}\{u_2^2 - u_1u_2 - 4u_2 : -u_1 + 2u_2 \leq 2, \ u_2 \geq \frac{1}{2}\}$$

Starting from an initial point $\mathbf{u}^{(0)}$ the iterative process following the serial protocol yields a sequence of iterates $\{\mathbf{u}^{(p)}\}$ defined by:

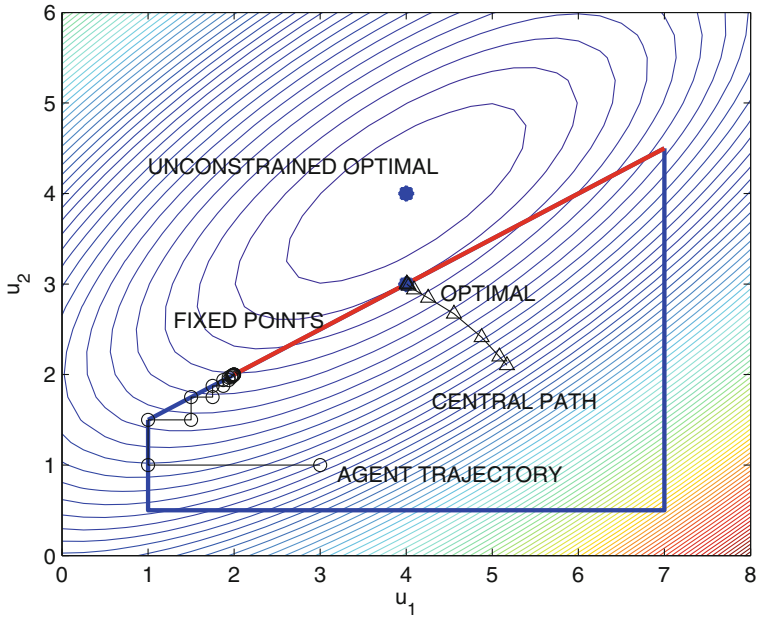

Fig. 12.2 Illustration of iterative processes based on the reactive functions and the interior-point method (extracted from [5]).

$$u_1^{(p)} = \begin{cases} r_1(u_2^{(p-1)}) & \text{if } p \text{ is odd} \\ u_1^{(p-1)} & \text{if } p \text{ is even} \end{cases} \qquad u_2^{(p)} = \begin{cases} u_2^{(p-1)} & \text{if } p \text{ is odd} \\ r_2(u_1^{(p-1)}) & \text{if } p \text{ is even} \end{cases}$$

Figure 12.2 shows the contour curves of f, the feasible region, and the set of fixed points. The trajectory $\mathbf{u}^{(p)} = [u_1^{(p)}\ u_2^{(p)}]^{\mathrm{T}}$ traced by the agents when they start from $[u_1^{(0)}\ u_2^{(0)}]^{\mathrm{T}} = [3\ 1]^{\mathrm{T}}$ is also shown, which converges to the fixed point $\mathbf{u} = [2\ 2]^{\mathrm{T}}$. Actually, all the points in the line segment $\{(u_1, (2+u_1)/2) : 2 \leq u_1 \leq 7\}$ are fixed points for $P_1(u_2)$ and $P_2(u_1)$. The figure also depicts the trajectory followed by agents that implement an interior-point strategy converging to the optimal solution $\mathbf{u}^\star = [4\ 3]^{\mathrm{T}}$. This trajectory is known as *central path*.

The algorithmic solution presented here relies on an interior-point method to approximate problem $P(k)$ given in Eq. (12.14) with an unconstrained problem to which gradient descent is applied [2]. The approximation accounts for the constraints with a *logarithmic barrier* function:

$$\phi(\widehat{\mathbf{u}}(k)) = -\sum_{i \in \mathcal{H}} \log(-h_i(\widehat{\mathbf{u}}(k))) \tag{12.18}$$

which is real-valued within the interior set $\bar{\Omega}$ but tends to infinity as the solution $\widehat{\mathbf{u}}(k)$ is drawn towards the boundary of any constraint.

Algorithm 12.1 Barrier method for solving $P(k)$

1: **Input:** strictly feasible $\widehat{\mathbf{u}}(k)^{\mathrm{s}}$, initial $\epsilon(k)^{(0)}$, decrease rate $\mu < 1$, tolerance τ;
2: **Initialize:** $l := -1$;
3: **Repeat**

a. $l := l + 1$;
b. **Centering step:** obtain $\widehat{\mathbf{u}}(k)^{(l)}$ by solving $P(k, \epsilon(k)^{(l)})$ with initial solution $\widehat{\mathbf{u}}(k)^{\mathrm{s}}$;
c. **If** $\epsilon(k)^{(l)} > \tau/|\mathcal{H}|$ **then**

i. $\widehat{\mathbf{u}}(k)^{\mathrm{s}} := \widehat{\mathbf{u}}(k)^{(l)}$;
ii. $\epsilon(k)^{(l+1)} := \mu\epsilon(k)^{(l)}$;

4: **Until** $\epsilon(k)^{(l)} \leq \tau/|\mathcal{H}|$;
5: **Output:** $\widehat{\mathbf{u}}(k)^{(l)}$;

The approximation problem is called *centering problem* being defined as:

$$P(k, \epsilon): \quad \min_{\widehat{\mathbf{u}}(k) \in \operatorname{dom} \theta} \theta(\widehat{\mathbf{u}}(k)) = f(\widehat{\mathbf{u}}(k)) + \epsilon\phi(\widehat{\mathbf{u}}(k)) \tag{12.19}$$

where $\operatorname{dom} \theta = \bar{\Omega}$ and the parameter $\epsilon > 0$ sets the accuracy of the approximation. The barrier method solves $P(k, \epsilon(k)^{(l)})$ for a decreasing sequence $\{\epsilon(k)^{(l)}\}_{l=0}^{\infty}$. The solution $\widehat{\mathbf{u}}(k, \epsilon(k)^{(l)})$ to the centering problem $P(k, \epsilon(k)^{(l)})$ is drawn towards the optimal solution $\widehat{\mathbf{u}}(k)^{\star}$ to $P(k)$ as $\epsilon(k)^{(l)}$ tends to 0. The barrier method is described in Algorithm 12.1.

A strictly feasible solution $\widehat{\mathbf{u}}(k)^{\mathrm{s}}$ can be obtained by solving the auxiliary convex program [2]:

$$P(k)^{\mathrm{s}}: \quad \min \quad \sum_{i \in \mathcal{H}} s_i \tag{12.20a}$$

$$\text{s.t.}: \quad h_i(\widehat{\mathbf{u}}(k)^{\mathrm{s}}) \leq s_i, \ i \in \mathcal{H}, \tag{12.20b}$$

$$s_i \geq 0, \ i \in \mathcal{H}. \tag{12.20c}$$

The distributed optimization strategy aims at solving the centering problem $P(k, \epsilon)$ with the agent network. The agent of a subsystem i will sense the local state variables $\mathbf{x}_i$ and decide upon the control signals $\mathbf{u}_i$. Communication among agents will ensure a perfect decomposition of the centering problem whereby each agent i solves, given the neighborhood variables $\widehat{\mathbf{w}}_i(k)$, the centering subproblem:

$$P_i(k, \epsilon, \widehat{\mathbf{w}}_i(k)): \quad \min_{\widehat{\mathbf{u}}_i(k) \in \operatorname{dom} \theta_i} \theta_i(\widehat{\mathbf{u}}_i(k)) = f_i(\widehat{\mathbf{u}}_i(k), \widehat{\mathbf{w}}_i(k)) + \epsilon\phi_i(\widehat{\mathbf{u}}_i(k), \widehat{\mathbf{w}}_i(k)) \tag{12.21}$$

where:

$$\phi_i(\widehat{\mathbf{u}}_i(k), \widehat{\mathbf{w}}_i(k)) = -\sum_{j \in \mathcal{H}_i} \log(-h_j(\widehat{\mathbf{u}}_i(k), \widehat{\mathbf{w}}_i(k))) \tag{12.22}$$

is the logarithmic barrier for the constraints depending on $\widehat{\mathbf{u}}_i$. Assuming fixed neighborhood variables any converging algorithm could be applied by agent i to solve $P_i(k, \epsilon, \widehat{\mathbf{w}}_i(k))$. Instead, a simple gradient-descent strategy is suggested to yield the optimal solution to $P(k, \epsilon)$ provided that the agents coordinate their iterations. This descent strategy does not require any agent i to optimally solve $P_i(k, \epsilon, \widehat{\mathbf{w}}_i(k))$, only a sufficient decrease on the objective is needed.

Given $(\widehat{\mathbf{u}}_i(k)^{(l,p)}, \widehat{\mathbf{w}}_i(k)^{(l,p)})$, agent i yields the next iterate $\widehat{\mathbf{u}}_i(k)^{(l,p+1)}$ by taking a step $s_i(k)^{(l,p)} > 0$ in the gradient descent direction $-\nabla\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)})$:

$$\widehat{\mathbf{u}}_i(k)^{(l,p+1)} = \widehat{\mathbf{u}}_i(k)^{(l,p)} - s_i(k)^{(l,p)} \nabla\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)}). \tag{12.23}$$

The iterations of the barrier method given by counter l are called *outer iterations*, whereas the iterations of the distributed gradient-descent method given by counter p are called *inner iterations*. Notice that for each outer iteration l there is a series of inner iterations l being denoted by (l, p). The distributed descent strategy is detailed in Algorithm 12.2 which is followed by the agent network to solve $\{P_i(k, \epsilon(k)^{(l)})\}$ in place of $P(k, \epsilon(k)^{(l)})$. The algorithm requires that at least one agent of maximum descent works in each iteration, with the set of agents of maximum descent being:

$$\mathcal{N}(k)^{(l,p)} = \{i \in \mathcal{N} : \|\nabla\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)})\| = \max_{j \in \mathcal{N}} \|\nabla\theta_j(\widehat{\mathbf{u}}_j(k)^{(l,p)})\|\}.$$

The use of Newton's direction $-\nabla^2\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)})^{-1}\nabla\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)})$, multiple backtracking iterations, and off-the-shelf solvers are discussed in [5]. The distributed identification of a set $\mathcal{V}(k)^{(l,p)}$ containing at least one agent of maximum descent can be performed by defining an agent cycle $\mathcal{A} = \langle \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{K-1} \rangle$ such that $\cup_{k=0}^{K-1} \mathcal{A}_k = \mathcal{N}$, and i and j are nonneighbors for all $i, j \in \mathcal{A}_k$.

Then, a message-exchange protocol can be implemented to follow the sequence $\langle \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{K-1}, \mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{K-1}, \ldots \rangle$, skipping sets until finding $\mathcal{V}(k)^{(l,p)} = \mathcal{A}_t$ such that $\mathcal{A}_t \cap \mathcal{N}(k)^{(l,p)} \neq \emptyset$. In essence, each agent $i \in \mathcal{A}_l$ would send a message with a token to all the agents in $\mathcal{A}_{(l+1) \bmod K}$, which would be blocked waiting for the tokens.

Distributed detection of global convergence can be achieved by keeping track of the number of past iterations that all of the preceding agents have converged with respect to their problems. Besides state and control information, each agent i receives from a neighbor $j \in \mathcal{N}(i)$ a parameter t_j such that: $t_j = 0$ if agent j changed its control decisions in the latest iteration; otherwise $t_j = \min\{t_l : l \in \mathcal{N}(j)\} + 1$. So, if agent j changes its decisions, it will define $t_j = 0$ and pass this value to its neighbors when they request its state and control variables, otherwise $t_j = \min\{t_l : l \in \mathcal{N}(j)\} + 1$. If in any iteration (l, p), an agent $i \in \mathcal{V}(k)^{(l,p)}$ detects

Algorithm 12.2 Distributed gradient descent algorithm for solving $P(k, \epsilon(k)^{(l)})$

1: **Input:** strictly feasible $\widehat{\mathbf{u}}(k)^{(l)}$, barrier parameter $\epsilon(k)^{(l)}$, backtracking parameters $\alpha \in (0, 1/2)$ and $\beta \in (0, 1)$, and tolerance τ;
2: **Initialize:** $p := 0$; $\widehat{\mathbf{u}}(k)^{(l,0)} := \widehat{\mathbf{u}}(k)^{(l)}$;
3: **While** $\|\nabla\theta(\widehat{\mathbf{u}}(k)^{(l,p)})\| > \tau$ **do**

a. Let $\mathcal{V}(k)^{(l,p)} \subseteq \mathcal{N}$ be a subset of non-neighboring agents such that $\mathcal{V}(k)^{(l,p)} \cap \mathcal{N}(k)^{(l,p)} \neq \emptyset$;
b. **For** each $i \in \mathcal{V}(k)^{(l,p)}$ in parallel **do**

i. Obtain $\widehat{\mathbf{w}}_i(k)^{(l,p)}$ from the neighborhood $\mathcal{N}(i)$;
ii. $s_i(k)^{(l,p)} := 1$;
iii. **While** $\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)} - s_i(k)^{(l,p)}\nabla\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)})) > \theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)}) - \alpha s_i(k)^{(l,p)}\|\nabla\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)})\|^2$ **do**

$$s_i(k)^{(l,p)} := \beta s_i(k)^{(l,p)};$$

iv. $\widehat{\mathbf{u}}_i(k)^{(l,p+1)} := \widehat{\mathbf{u}}_i(k)^{(l,p)} - s_i(k)^{(l,p)}\nabla\theta_i(\widehat{\mathbf{u}}_i(k)^{(l,p)})$;

c. **For** each $i \in \mathcal{N} \setminus \mathcal{V}(k)^{(l,p)}$ in parallel **do**

$$\widehat{\mathbf{u}}_i(k)^{(l,p+1)} := \widehat{\mathbf{u}}_i(k)^{(l,p)};$$

d. $p := p + 1$;

4: **Output:** $\widehat{\mathbf{u}}(k)^{(l,p)}$;

that $\min\{t_l : l \in \mathcal{N}(i)\} \geq K$, then global convergence has been achieved assuming that the LDN is a connected graph. This agent i can broadcast a message to all the others or let them detect convergence on their own.

12.3.3 Theoretical Results

In [5], it was shown that the iterates $\{\widehat{\mathbf{u}}(k)^{(l,p)}\}_{p=0}^{\infty}$ produced by the distributed gradient-descent algorithm converge to the solution $\widehat{\mathbf{u}}(k)^{(l)}$ to the centering problem $P(k, \epsilon(k)^{(l)})$. The convergence is a consequence of the assumptions stated below which are satisfied by Algorithm 12.2.

Assumption 12.1 (Synchronous Work) If agent i updates its variables in iteration p, then:

1. agent i uses $\widehat{\mathbf{w}}_i(k) = \widehat{\mathbf{w}}_i(k)^{(l,p)}$ and follows the descent method to obtain an approximate solution $\widehat{\mathbf{u}}_i(k)^{(l,p+1)}$ to $P_i(k, \epsilon(k)^{(l)}, \widehat{\mathbf{w}}_i(k))$;
2. $\widehat{\mathbf{u}}_i(k)^{(l,p)}$ is not an optimal solution to $P_i(k, \epsilon(k)^{(l)}, \widehat{\mathbf{w}}_i(k))$ since otherwise the agent does not yield any improvement; and
3. each neighbor of agent i does not iterate, meaning that $\widehat{\mathbf{u}}_j(k)^{(l,p+1)} = \widehat{\mathbf{u}}_j(k)^{(l,p)}$ for all $j \in \mathcal{N}(i)$.

Notice that condition 1 of the Assumption 12.1 is met by the algorithm since the agents $i \in \mathcal{V}(k)^{(l,p)}$ request $\widehat{\mathbf{w}}_i(k)^{(l,p)}$ from their neighbors, condition 2 can be

satisfied by agents that are not locally optimal, and condition 3 is satisfied by the agents $i \in \mathcal{N} \setminus \mathcal{V}(k)^{(l,p)}$ which do not iterate.

Assumption 12.2 (Continuous and Maximum Work) If $\widehat{\mathbf{u}}(k)^{(l,p)}$ is not optimal to $P(k, \epsilon(k)^{(l)})$ at outer iteration l, then any agent $i(k, l, p) \in \mathcal{N}(k)^{(l,p)}$ performs a backtracking line search, starting at $\widehat{\mathbf{u}}_{i(k,l,p)}(k)^{(l,p)}$, to produce its next solution $\widehat{\mathbf{u}}_{i(k,l,p)}(k)^{(l,p+1)}$.

This assumption is ensured by Algorithm 12.2 at the step that defines the set $\mathcal{V}(k)^{(l,p)}$ of the agents that iterate.

Assumption 12.3 The objective function $\theta(\widehat{\mathbf{u}}(k))$ of the centering problem $P(k, \epsilon)$ is strongly convex.

With strong convexity of $\theta(\widehat{\mathbf{u}}(k))$ the convergence of Algorithm 12.2 is established as stated below. Actually the rate of convergence can be determined in terms of bounds induced by strong convexity.

Theorem 12.1 *Under Assumptions 12.1, 12.2, and 12.3, the distributed gradient-descent algorithm yields a sequence* $\{\widehat{\mathbf{u}}(k)^{(l,p)}\}_{p=0}^{\infty}$ *of iterates converging to the optimal solution* $\widehat{\mathbf{u}}(k)^{(l)}$ *to the centering problem* $P(k, \epsilon(k)^{(l)})$ *using exact or backtracking line search.*

12.4 Applications

Although distributed MPC has been applied to operate and control several distributed dynamic systems, here a reference is made only to the ones closely related to the control of linear dynamic networks. In [8] the green-time control of urban traffic networks was modeled as the MPC of an LDN with constraints on the control signals. Distributed optimization was carried out using a gradient-projection strategy. In [5] this work was extended to incorporate road capacity by introducing constraints on state variables. A simulation study was performed in the network of the city of Macaé (north of Rio de Janeiro) consisting of 15 junctions and 28 road links [13]. The quadratic programs for MPC were solved using the interior-point strategy and the distributed gradient-descent algorithm. In [12] the dynamic models for LDNs were expressed using transfer functions and the constraints were imposed on the control inputs and output signals. The resulting distributed transfer-function MPC approach was applied to the control of a distillation column.

Acknowledgments This work was funded in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), Brazil.

References

1. D.P. Bertsekas, *Nonlinear Programming* (Athena Scientific, Belmont, MA, 1995)
2. S. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge University Press, Cambridge, 2004)

3. E. Camponogara, L.B. de Oliveira, Distributed optimization for model predictive control of linear dynamic networks. IEEE Trans. Syst. Man Cybern. Part A **39**(6), 1331–1338 (2009)
4. E. Camponogara, M.L. Lima, Distributed optimization for MPC of linear networks with uncertain dynamics. IEEE Trans. Autom. Control **57**(3), 804–809 (2012)
5. E. Camponogara, H.F. Scherer, Distributed optimization for model predictive control of linear dynamic networks with control-input and output constraints. IEEE Trans. Autom. Sci. Eng. **8**(1), 233–242 (2011)
6. E. Camponogara, S.N. Talukdar, Designing communication networks to decompose network control problems. INFORMS J. Comput. **17**(2), 207–223 (2005)
7. E. Camponogara, H. Zhou, S.N. Talukdar, Altruistic agents in uncertain, dynamic games. J. Comput. Syst. Sci. Int. **45**(4), 536–552 (2006)
8. L.B. de Oliveira, E. Camponogara, Multi-agent model predictive control of signaling split in urban traffic networks. Transp. Res. Part C **18**(1), 120–139 (2010)
9. C. Diakaki, M. Papageorgiou, K. Aboudolas, A multivariable regulator approach to traffic-responsive network-wide signal control. Control Eng. Pract. **10**(2), 183–195 (2002)
10. I. Necoara, V. Nedelcu, I. Dumitrache, Parallel and distributed optimization methods for estimation and control in networks. J. Process Control **21**, 756–766 (2011)
11. R.R. Negenborn, B. De Schutter, J. Hellendoorn, Multi-agent model predictive control for transportation networks: serial versus parallel schemes. Eng. Appl. Artif. Intell. **21**(3), 353–366 (2007)
12. H.F. Scherer, E. Camponogara, A. Codas, Transfer function modeling of linear dynamic networks for distributed MPC, in *Proceedings of the 7th IEEE Conference on Automation Science and Engineering*, pp. 613–618, 2011
13. F.A. Souza, V.B. Peccin, E. Camponogara, Distributed model predictive control applied to urban traffic networks: implementation, experimentation, and analysis, in *Proceedings of the 6th IEEE Conference on Automation Science and Engineering*, pp. 399–405, 2010
14. A.N. Venkat, I.A. Hiskens, J.B. Rawlings, S.J. Wright, Distributed MPC strategies with application to power system automatic generation control. IEEE Trans. Control Syst. Technol. **16**(6), 1192–1206 (2008)
15. M. Zhu, S. Martínez, On distributed convex optimization under inequality and equality constraints. IEEE Trans. Autom. Control **57**(1), 151–164 (2012)