European Journal of Control 70 (2023) 100768

Contents lists available at ScienceDirect

European Journal of Control

journal homepage: www.elsevier.com/locate/ejcon

Distributed optimal control of nonlinear systems using a second-order augmented Lagrangian method

Shervin Parvini Ahmadi*, Anders Hansson

Division of Automatic Control, Linköping University, Sweden

ARTICLE INFO

Article history:
Received 1 December 2021
Revised 27 November 2022
Accepted 15 December 2022
Available online 10 January 2023

Recommended by Prof. T Parisini

Keywords:
Distributed optimal control
Distributed model predictive control
Augmented Lagrangian
Distributed optimization

ABSTRACT

In this paper, we propose a distributed second-order augmented Lagrangian method for distributed optimal control problems, which can be exploited for distributed model predictive control. We employ a primal-dual interior-point approach for the inner iteration of the augmented Lagrangian and distribute the corresponding computations using message passing over what is known as the clique tree of the problem. The algorithm converges to its centralized counterpart and it requires fewer communications between sub-systems as compared to algorithms such as the alternating direction method of multipliers. We illustrate the efficiency of the framework when applied to randomly generated interconnected sub-systems as well as to a vehicle platooning problem.

1. Introduction

Consider a discrete-time system composed by the interconnection of m sub-systems. The problem we are interested in solving is the optimal control problem that is solved repeatedly in Model Predictive Control (MPC). It is of the form

$$\min_{x,u} \sum_{i=1}^{m} \left(\sum_{k=1}^{N} J_i(x_i(k), u_i(k)) \right) + J_i^f(x_i(N+1)) \tag{1a}$$

$$\begin{aligned} \text{s.t. } & x_i(k+1) = f_i\big(x_i(k), u_i(k)\big) + \sum_{j \in \mathcal{N}(i)} f_{ij}\big(x_j(k), u_j(k)\big), \\ & g_{ij}\big(x_i(k), u_i(k), x_j(k), u_j(k)\big) = 0, \quad j \in \mathcal{N}(i), \\ & g_i\big(x_i(k), u_i(k)\big) = 0, \\ & h_{ij}\big(x_i(k), u_i(k), x_j(k), u_j(k)\big) \leq 0, \quad j \in \mathcal{N}(i), \\ & h_i\big(x_i(k), u_i(k)\big) \leq 0, \quad k = 1, \dots, N, \quad i = 1, \dots, m, \end{aligned} \tag{1b}$$

where $x_i(k) \in \mathbb{R}^{n_{x_i}}$ and $u_i(k) \in \mathbb{R}^{n_{u_i}}$ are the states and the inputs of sub-system i at time instant k, $J_i : \mathbf{R}^{n_{x_i}+n_{u_i}} \to \mathbf{R}$, $J_i^f : \mathbf{R}^{n_{x_i}} \to \mathbf{R}$, $f_i : \mathbf{R}^{n_{x_i}+n_{u_i}} \to \mathbf{R}^{n_{x_i}}$, $f_{ij} : \mathbf{R}^{n_{x_j}+n_{u_j}} \to \mathbf{R}^{n_{x_i}}$, $g_{ij} : \mathbf{R}^{n_{x_i}+n_{u_i}+n_{x_j}+n_{u_j}} \to \mathbf{R}^{n_{g_{ij}}}$, $g_i : \mathbf{R}^{n_{x_i}+n_{u_i}} \to \mathbf{R}^{n_{g_i}}$, $h_{ij} : \mathbf{R}^{n_{x_i}+n_{u_i}+n_{x_j}+n_{u_j}} \to \mathbf{R}^{n_{h_{ij}}}$, $h_i : \mathbf{R}^{n_{x_i}+n_{u_i}} \to \mathbf{R}^{n_{h_i}}$ are twice differentiable functions, N is the horizon length and $\mathcal{N}(i)$ is the set of all indices of sub-systems which interact with sub-system i.

MPC is a popular advanced control method due to its applicability to both linear and non-linear systems, its capability in handling constraints on both inputs and states, as well as its optimal performance with respect to a cost function, [12,22]. It is based on solving a finite horizon optimal control problem at each sampling instant, where the current state of the plant is used as initial condition for the state prediction. The first part of the resulting control input sequence is then applied to the actual plant, the time horizon is shifted and the same procedure is repeated. Traditionally, the problem is solved using what is known as centralized MPC, where all measurements and information available about the system are collected in a central unit to calculate all control actions. For large-scale systems or systems with considerable number of sub-systems, this approach becomes impractical due to the high computational effort or due to the fact that collecting all information in a central unit is not feasible, [10,18,21,26,40,45]. Examples of such systems are power distribution systems, water distribution systems, transportation systems, biological systems and cooperative payload transport in robotic applications etc, [21]. Decentralized MPC and distributed MPC are two popular approaches to address this issue. In both approaches the computations are calculated distributedly in local units assigned to each sub-system. The main difference, however, is that in decentralized MPC the coupling between sub-systems is ignored and the control decisions for the sub-systems are taken independently, whereas in distributed MPC, sub-systems communicate with each other to compute the optimal control signal, [14,34]. Here we focus on distributed MPC

* Corresponding author.
E-mail address: shervin.parvini.ahmadi@liu.se (S. Parvini Ahmadi).

https://doi.org/10.1016/j.ejcon.2022.100768

which has been studied extensively in the control community. Next we present a short overview of different approaches.

In [47], a non-linear distributed MPC is presented, which is based on gradient projection. The presented algorithm does not require a central coordination level and therefore, it is truly distributed. In [46], a sensitivity-driven distributed MPC for linear time-invariant systems is presented. The coordination between sub-systems is achieved by using a linear approximation of the objective functions of the neighboring controllers within the objective function of each local controller. This, in turn, leads to overall optimality. In [50], a distributed MPC is presented for large-scale networked systems such as power systems. The overall system is assumed to be linear and time-invariant. Each sub-system has its own MPC controller, and the sub-systems work iteratively and cooperatively towards satisfying system-wide control objectives. The framework is based on a terminal penalty, and it achieves performance equivalent to centralized MPC. In general, these frameworks are based on a primal decomposition method. A disadvantage of this class of methods is that each sub-system requires knowledge of the overall system, which in turn limits the scalability of the method [6].

Another class of distributed MPC methods is based on dual decomposition. In [25], a distributed optimization algorithm for strongly convex problems is presented. The algorithm is based on accelerated gradient methods using dual decomposition. It is applied to randomly generated problems arising in distributed model predictive control. In [18], a distributed non-linear optimal controller is presented in which the cost function is assumed to be separable and convex. The algorithm is based on two ingredients. First, the convex problem structure is exploited using a sequential convex programming framework that linearizes the nonlinear dynamics in each iteration. Second, a distributed dual decomposition method is used to solve the resulting problem. Also in [18], a dual decomposition method for distributed model predictive control over networks is presented. The authors assume separable non-convex optimization problems. A sequential convex programming scheme based on a penalty function is used to handle the non-convexity. See [6], for a more complete list of papers which are based on dual decomposition methods. A drawback of this class of methods is that their convergence rate might be low, which leads to several communications among sub-systems. This issue is addressed in what is known as the Alternating Direction Method of Multipliers (ADMM) method. It has received a lot of attention due to its capability in retaining the decomposability of the dual formulation while ensuring faster convergence, [8]. In [6], a scheme for continuous-time non-linear systems based on ADMM is proposed and stability results are presented under two different ADMM convergence assumptions. Similar approaches are used in [10,11,28], where the performance of the framework is evaluated on different applications. Their approach is based on the assumption that the systems are non-linear neighbor-affine. In [21], two frameworks for distributed MPC are proposed based on dual decomposition and ADMM, respectively. In [45], the authors investigate the application of ADMM on the distributed MPC problem, both in primal and dual domains. In [30], a Proximal Jacobian ADMM algorithm is presented for distributed MPC, in particular for building control applications. In [34], a linear, time-invariant, discrete-time plant with coupled subsystems is considered and two approaches based on ADMM are proposed. In [48], the authors apply the ADMM based distributed MPC method to a flocking problem in a network of double integrators. They show that a near-centralized performance can be achieved with only a few tens of iterations. In [17], the authors investigate performance of ADMM and a dual decomposition method based on fast gradient updates, by a systematic computational study. In [31], a distributed algorithm, known as ALADIN (augmented Lagrangian alternating direction inexact Newton), is presented for non-convex smooth optimization problems with coupled affine constraints. The algorithm is a further development of ADMM, which contains ideas from augmented Lagrangian and sequential quadratic programming. The algorithm has a faster convergence rate compared to ADMM based algorithms. This however comes at the price that it requires more communications, [37]. In [20], a distributed version of ALADIN is presented which is based on bi-level distribution, meaning that the outer ALADIN structure is combined with an inner distribution level solving a condensed variant of ALADINs convex coordination quadratic program by decentralized algorithms. See [31], for a complete list of distributed second-order methods. Common for the distributed approaches based on second order information is that they do not address general nonlinear constraints.

As mentioned ADMM improves the speed of convergence as compared to dual decomposition methods. However, as we will see in this paper, this is only the case when moderate accuracy is desired. For high accuracy, it might still be very slow which results in many iterations for convergence, which in turn leads to excessive communications between sub-systems. Similar to our previous work, [2], we take in this paper a different approach based on the Augmented Lagrangian (AL) method proposed in [16]. In [2], we only considered equality constraints and not inequality constraints. In this paper we extend the algorithm to optimization problems with both equality and inequality constraints. The idea is to handle the inequality constraints by incorporating a primal-dual interior-point approach, [41, Ch. 19], in the augmented Lagrangian framework. We distribute the algorithm using message-passing over what is known as the clique tree of the problem, as in [33]. We will see that the proposed algorithm requires much fewer iterations for convergence and, in turn fewer communications, as compared to the most competitive method proposed in [6], which is based on ADMM. However, this comes at the price that we cannot distribute the computations freely. We will come back to this later. However, the fact that we distribute the computations of the AL method using message-passing over a clique tree, does not in any way effect the convergence behaviour of the method. In other words, the distributed version of the AL method behaves exactly as the centralized version of it. See [16], for the convergence properties of the AL method in which it is shown that AL is at least R-linearly convergent regardless of what algorithm is used for the inner iteration.

The main contributions of this paper are

- Development of a distributed second-order optimization method for general non-convex problems.
- Application of the method to general nonlinear distributed predictive control.
- Application of the method to a vehicle platooning problem.

The rest of the paper is organized as follows. In Section 2, we present the augmented Lagrangian algorithm. In Section 3, we discuss how we can distribute the algorithm over the clique tree of the problem. Numerical experiments are presented in Section 4 and we conclude the paper in Section 5.

2. Augmented Lagrangian and primal-dual interior-point methods

Consider the optimization problem

$$\min_x \ f(x) \tag{2a}$$

$$\text{s.t. } g(x) = 0, \tag{2b}$$

$$h(x) \leq 0, \tag{2c}$$

where $f : \mathbf{R}^n \to \mathbf{R}$ and $h : \mathbf{R}^n \to \mathbf{R}^q$ are not necessarily convex, nor is $g : \mathbf{R}^n \to \mathbf{R}^p$ necessarily affine. In order to define the augmented Lagrangian, one approach is to convert the inequality constraints to equality constraints and then include all the constraints in the augmented Lagrangian function. The other approach, is to include part of the constraints in the augmented Lagrangian function while the rest of the constraints are dealt with directly. The latter approach is called partial augmented Lagrangian or partial elimination of constraints, [5, Ch. 2], [4, Ch. 4]. Here we only include the equality constraints in the augmented Lagrangian function and we deal with the inequality constraints by imposing the complementary slackness condition, [9, Ch. 11], [41, Ch. 19]. The partial augmented Lagrangian of this problem is given by the function $L_\mu : \mathbf{R}^n \times \mathbf{R}^q \times \mathbf{R}^p \to \mathbf{R}$, where

$$L_\mu(x, \lambda, \nu) = f(x) + \lambda^T h(x) + \nu^T g(x) + \frac{1}{2\mu}||g(x)||_2^2, \tag{3}$$

where λ and ν are Lagrangian multipliers for the inequality and equality constraints, respectively and μ is a penalty parameter. See [24], for different versions of augmented Lagrangian functions and their relations to primal-dual methods. It is shown in [29] and [44] that an augmented Lagrangian method converges to a local minima if the penalty parameter is sufficiently small, and if the augmented Lagrangian is approximately minimized at each iteration. In this paper we use the augmented Lagrangian algorithm presented in [16], which is the basis of the successful software package LANCELOT, [15,41]. The algorithm consists of inner iteration and outer iteration parts. In the inner iteration part the augmented Lagrangian is approximately minimized and in the outer iteration part the Lagrangian multipliers and the penalty parameter are updated. We will use an interior-point method for the inner part. This has previously been considered in e.g. [13], where the authors use a preconditioned conjugate gradient method on a Graphics processing unit (GPU) in order to compute the search directions in parallel. Our approach is instead related to multifrontal factorization techniques for computing the search directions. In [38], the authors propose an efficient implementation of an interior-point algorithm for non-convex problems that uses directions of negative curvature. The method uses the augmented Lagrangian as a merit function. In [7], a hybrid algorithm is proposed in which the interior-point method is replaced by the Newtonian resolution of a KKT system identified by the augmented Lagrangian algorithm. In [3], an interior-point Newton algorithm for nonlinear programming problems is proposed, where they use a generalization of the augmented Lagrangian function as a merit function in order to obtain global convergence. Next we discuss how the inner iteration can be carried out using a primal-dual interior-point method. At the end of the section, we will present the overall augmented Lagrangian method.

Necessary conditions for an optimal x of ((2a)–(2c)) is that there exist Lagrange multipliers λ and μ such that

$$\frac{\partial L_\mu(x, \lambda, \mu)}{\partial x} = 0, \tag{4a}$$

$$g(x) = 0, \tag{4b}$$

$$h(x) \le 0, \tag{4c}$$

$$-\mathbf{diag}(\lambda)h(x) = 0, \tag{4d}$$

$$\lambda \ge 0, \tag{4e}$$

where $\frac{\partial L_\mu(x,\lambda,\mu)}{\partial x}$ can be written as

$$\frac{\partial L_\mu(x, \lambda, \mu)}{\partial x} = \frac{\partial f(x)}{\partial x} + \frac{\partial h(x)^T}{\partial x}\lambda + \frac{\partial g(x)^T}{\partial x}\left(\nu + \frac{1}{\mu}g(x)\right), \tag{5}$$

where $\frac{\partial h(x)^T}{\partial x} = (\frac{\partial h(x)}{\partial x^T})^T$ and $\frac{\partial g(x)^T}{\partial x} = (\frac{\partial g(x)}{\partial x^T})^T$. $\frac{\partial h(x)}{\partial x^T}$ and $\frac{\partial g(x)}{\partial x^T}$ are the Jacobian of $h(x)$ and $g(x)$, respectively. The matrix $\mathbf{diag}(\lambda)$ is a diagonal matrix with the elements of λ on the diagonal. Equation in (4d) is known as the complementary slackness condition, [9, Ch. 11], [41, Ch. 19].

A primal-dual interior-point method computes a solution for ((4a)–(4e)) by applying Newton's method in order to solve them in an iterative fashion, where Eq. (4d) is modified as $-\mathbf{diag}(\lambda)h(x) = (\frac{1}{t})\mathbf{1}$, where $t \ge 0$ and $\mathbf{1}$ is a vector of all ones. Specifically, at each iteration l given $x^{(l)}$, $\lambda^{(l)}$ and $\nu^{(l)}$ in such a way that $h(x^{(l)}) \le 0$ and $\lambda^{(l)} \ge 0$, the search directions Δx, $\Delta\lambda$ and $\Delta\nu$ are computed by solving the following linear system of equations which are obtained by linearizing (4a), (4b) and the modified version of (4d)

$$\begin{aligned}
&\left\{\frac{\partial^2 f(x^{(l)})}{\partial x \partial x^T} + \sum_{i=1}^{q}\left[\frac{\partial^2 h(x^{(l)})}{\partial x \partial x^T}\lambda_i^{(l)}\right]\right. \\
&\quad + \sum_{i=1}^{p}\left[\left(\nu_i^{(l)} + \frac{1}{\mu} \times g_i(x^{(l)})\right)\frac{\partial^2 g_i(x^{(l)})}{\partial x \partial x^T}\right. \\
&\quad \left.\left. + \frac{1}{\mu}\frac{\partial g_i(x^{(l)})}{\partial x}\frac{\partial g_i(x^{(l)})}{\partial x^T}\right]\right\}\Delta x + \frac{\partial h(x^{(l)})^T}{\partial x}\Delta\lambda \\
&\quad + \frac{\partial g(x^{(l)})^T}{\partial x}\Delta\nu = -r_{\text{dual}}^{(l)},
\end{aligned} \tag{6a}$$

$$\frac{\partial g(x^{(l)})^T}{\partial x}\Delta x = -r_{\text{primal}}^{(l)}, \tag{6b}$$

$$-\mathbf{diag}(\lambda^{(l)})\frac{\partial h(x^{(l)})}{\partial x^T}\Delta x - \mathbf{diag}\left(h(x^{(l)})\right)\Delta\lambda = -r_{\text{cent}}^{(l)}, \tag{6c}$$

where $g_i(x)$ and $h_i(x)$ are the ith component of $g(x)$ and $h(x)$, respectively and

$$\begin{aligned}
r_{\text{dual}}^{(l)} &= -\frac{\partial L_\mu(x^{(l)}, \lambda^{(l)}, \mu^{(l)})}{\partial x}, \\
r_{\text{primal}}^{(l)} &= g(x^{(l)}), \\
r_{\text{cent}}^{(l)} &= -\mathbf{diag}(\lambda^{(l)})h(x^{(l)}) - (\frac{1}{t})\mathbf{1}.
\end{aligned}$$

We can reduce the system by eliminating $\Delta\lambda$ as

$$\Delta\lambda = \mathbf{diag}\left(h(x^{(l)})\right)^{-1}\left(-\mathbf{diag}(\lambda^{(l)})\frac{\partial h(x^{(l)})}{\partial x^T}\Delta x + r_{\text{cent}}^{(l)}\right), \tag{7}$$

which will result in the following linear system of equations

$$\begin{bmatrix} \frac{\partial^2 L_\mu(x^{(l)}, \lambda^{(l)}, \nu^{(l)})}{\partial x \partial x^T} & \frac{\partial g(x^{(l)})^T}{\partial x} \\ \frac{\partial g(x^{(l)})}{\partial x^T} & 0 \end{bmatrix}\begin{bmatrix}\Delta x \\ \Delta\lambda\end{bmatrix} = -\begin{bmatrix} r^{(l)} \\ r_{\text{primal}}^{(l)}\end{bmatrix}, \tag{8}$$

where

$$\begin{aligned}
\frac{\partial^2 L_\mu\left(x^{(l)}, \lambda^{(l)}, \nu^{(l)}\right)}{\partial x \partial x^T} &= \frac{\partial^2 f(x^{(l)})}{\partial x \partial x^T} + \sum_{i=1}^{q}\left[\frac{\partial^2 h(x^{(l)})}{\partial x \partial x^T}\lambda_i^{(l)}\right] \\
&\quad + \sum_{i=1}^{p}\left[\left(\nu_i^{(l)} + \frac{1}{\mu} \times g_i(x^{(l)})\right)\frac{\partial^2 g_i(x^{(l)})}{\partial x \partial x^T}\right. \\
&\quad \left. + \frac{1}{\mu}\frac{\partial g_i(x^{(l)})}{\partial x}\frac{\partial g_i(x^{(l)})}{\partial x^T}\right] \\
&\quad - \frac{\partial h(x^{(l)})^T}{\partial x}\mathbf{diag}(x^{(l)})^{-1}\mathbf{diag}(\lambda^{(l)})\frac{\partial h(x^{(l)})}{\partial x^T}r^{(l)} \\
&= r_{\text{dual}}^{(l)} + \frac{\partial h(x^{(l)})^T}{\partial x}\mathbf{diag}\left(h(x^{(l)})\right)^{-1}r_{\text{cent}}^{(l)}.
\end{aligned} \tag{9}$$

Note that the linearized optimality conditions are the optimality conditions for the quadratic optimization problem

$$\min_{\Delta x} \ \frac{1}{2}\Delta x^T \frac{\partial^2 L_\mu\left(x^{(l)}, \lambda^{(l)}, \nu^{(l)}\right)}{\partial x \partial x^T}\Delta x + r^{(l)^T}\Delta x \tag{10a}$$

$$\text{s.t.} \ \frac{\partial g(x^{(l)})}{\partial x^T}\Delta x = -r^{(l)}_{\text{primal}}. \tag{10b}$$

In order to assure that the obtained search direction is a decent direction, the augmented Lagrangian Hessian in (9) should be positive definite on the null space of $\frac{\partial g(x)}{\partial x^T}$. In our previous work, [2], we satisfy this condition by making the augmented Lagrangian Hessian in (9) positive definite at each iteration by adding a multiple of the identity matrix when it is negative definite. In particular, when the Hessian is negative definite, we add $(|\sigma| + \epsilon)I$ to the Hessian, where σ is the most negative eigenvalue and $\epsilon > 0$. However, this approach is conservative. One can instead satisfy the augmented Lagrangian Hessian being positive definite on the null space of $\frac{\partial g(x)}{\partial x^T}$ by checking the *inertia* of the coefficient matrix in (8), which is the number of positive, negative, and zero eigenvalues. The obtained search direction is a decent direction if the matrix has exactly n positive, and p negative and zero eigenvalues, [41]. Therefore, if this condition is not fulfilled, one can add a multiple of the identity ($\alpha_\sigma I$) to the Hessian, i.e. to the (1,1)-block of the coefficient matrix in (8), in such a way that this is satisfied. For this purpose, a simple strategy is suggested in [41, Appendix B], which is presented in Algorithm 1. With this modification, the optimization problem for the search directions in ((10a)–(10b)) becomes

Algorithm 1 *Inertia* correction

Given $\alpha_{\sigma_{\text{old}}}$ used in the previous interior-point iteration
Find the *inertia* of the coefficient matrix in (8)
if the *inertia* condition is satisfied, i.e. it has exactly n positive, p negative and zero eigenvalues **then**
  Set $\alpha_\sigma \leftarrow 0$, $\alpha_{\sigma_{\text{old}}} \leftarrow 0$
  break
else if $\alpha_{\sigma_{\text{old}}} = 0$ **then**
  Set $\alpha_\sigma \leftarrow 10^{-4}$
else
  Set $\alpha_\sigma \leftarrow \frac{\alpha_{\sigma_{\text{old}}}}{2}$
end if
repeat
  Find the *inertia* of the coefficient matrix in (8) after adding $\alpha_\sigma I$ to the Hessian, i.e. (1,1) block
  if the *inertia* condition is satisfied, i.e. it has exactly n positive, p negative and zero eigenvalues **then**
    Set $\alpha_{\sigma_{\text{old}}} \leftarrow \alpha_\sigma$
    break
  else
    Set $\alpha_\sigma \leftarrow 10\alpha_\sigma$
  end if
until maximum number of iterations reached

$$\min_{\Delta x} \ \frac{1}{2}\Delta x^T \left(\frac{\partial^2 L_\mu(x^{(l)}, \lambda^{(l)}, \nu^{(l)})}{\partial x \partial x^T} + \alpha_\sigma I\right)\Delta x + r^{(l)^T}\Delta x \tag{11a}$$

$$\text{s.t.} \ \frac{\partial g(x^{(l)})}{\partial x^T}\Delta x = -r^{(l)}_{\text{primal}}. \tag{11b}$$

For computation of the step size at each iteration of the primal-dual interior-point method, we employ the well known backtracking line search method, [9,41]. The general idea in the method is as follows. First we compute the largest positive step, not exceeding one, for which $h(x^{(l+1)}) \leq 0$ and $\lambda^{(l+1)} \geq 0$. Then, we start the backtracking by multiplying the obtained step with $\beta_{\text{step}} \in (0, 1)$, until we have sufficient decrease in the residual norms, i.e. $r^{(l+1)}_{\text{primal}}$ and $r^{(l+1)}_{\text{dual}}$.

For the calculation of the t in the modified version of (4d), first we define what is known as the *surrogate duality gap* as in [33],

$$\eta_{sur}(x, \lambda) = -\lambda^T h(x). \tag{12}$$

The t parameter then can be computed as $t = \frac{\mu_t q}{\eta_{sur}(x,\lambda)}$ for $\mu_t \geq 0$, where a typical value for μ_t is 10, [9], and q is the number of inequality constraints. Note that the presented primal-dual interior-point method is used in the inner iteration of the algorithm (Lines 7–12). Note also that primal-dual interior-point method has a superlinear convergence rate for nonlinear programming problems, which is discussed in [41, Ch. 19].

Now we can summarize the overall augmented Lagrangian algorithm in Algorithm 2, [16, Section 3, Alg. 2]. Note that the presented primal-dual interior-point method is used in the inner iteration of the algorithm (Lines 7–12). Note also that the superscript (k) in Algorithm 2 corresponds to the outer iteration, and it should not be confused with the superscript (l) used in equations ((6a)–(11b)) which corresponds to inner iteration. The algorithm comprises four main steps. The first step (Line 1-2) is the initialization phase. The second step (Lines 4–12) is the inner iteration. Here, first we compute the parameter t associated with the complementary slackness condition. Then, we compute an approximate minimization of the problem in ((2a)–(2c)) by satisfying the optimality conditions in ((4a)–(4e)), using the presented primal-dual interior-point method. Note that one can move the computations of the

Algorithm 2 Augmented Lagrangian algorithm

1: Given the nonnegative constant α_η and the positive constants η_0, μ_0, $\tau < 1$, ω_0, $\gamma < 1$, $\gamma_1 < 1$, $\omega_* \ll 1$, $\eta_* \ll 1$, $\eta^{sur}_* \ll 1$, ζ, α_ω, β_ω, β_η, the optimization variable $x^{(1)}$ and the Lagrangian multipliers $\lambda^{(1)}$ and $\nu^{(1)}$
2: Set $\mu^{(1)} = \mu_0$, $\alpha^{(1)} = \min(\mu^{(1)}, \gamma_1)$, $\omega^{(1)} = \omega_0(\alpha^{(1)})^{\alpha_\omega}$ and $\eta^{(1)} = \eta_0(\alpha^{(1)})^{\alpha_\eta}$
3: **for** k= $1, \ldots, k_{\max}$ **do**
4: $\eta^{(k)}_{sur}(x^{(k)}, \lambda^{(k)}) = -\lambda^{(k)^T} h(x^{(k)})$
5: $t^{(k)} = \frac{\mu_t q}{\eta^{(k)}_{sur}(x^{(k)}, \lambda^{(k)})}$
6: $\hat{x} = x^{(k)}$, $\hat{\lambda} = \lambda^{(k)}$ and $\hat{\nu} = \nu^{(k)}$
7: **while** $||\frac{\partial L_{\mu^{(k)}}(\hat{x}, \hat{\lambda}, \hat{\nu})}{\partial x}|| > \omega^{(k)}$ **do**
8: Find search direction using (11a-11b)
9: Compute $\Delta\lambda$ using (7)
10: Find step size using backtracking line search
11: update $\hat{x}$, $\hat{\lambda}$ and $\hat{\nu}$
12: **end while**
13: $x^{(k+1)} = \hat{x}$, $\lambda^{(k+1)} = \hat{\lambda}$
14: **if** $||g(\hat{x})|| \leq \eta^{(k)}$ **then**
15: **if** $||\frac{\partial L_{\mu^{(k)}}(\hat{x}, \hat{\lambda}, \hat{\nu})}{\partial x}|| \leq \omega_*$ and $||g(\hat{x})|| \leq \eta_*$ and $\eta_{sur}(\hat{x}, \hat{\lambda}) \leq \eta^{sur}_*$ **then**
16: break
17: **end if**
18: $\mu^{(k+1)} = \mu^{(k)}$
19: **if** $||\hat{\nu}|| \leq \zeta(\mu^{(k+1)})^{-\gamma}$ **then**
20: $\nu^{(k+1)} = \hat{\nu}$
21: **else**
22: $\nu^{(k+1)} = \nu^{(k)}$
23: **end if**
24: $\alpha^{(k+1)} = \min(\mu^{(k+1)}, \gamma_1)$
25: $\omega^{(k+1)} = \omega^{(k)}(\alpha^{(k+1)})^{\beta_\omega}$
26: $\eta^{(k+1)} = \eta^{(k)}(\alpha^{(k+1)})^{\beta_\eta}$
27: **else**
28: $\mu^{(k+1)} = \tau\mu^{(k)}$
29: **if** $||\hat{\nu}|| \leq \zeta(\mu^{(k+1)})^{-\gamma}$ **then**
30: $\nu^{(k+1)} = \hat{\nu}$
31: **else**
32: $\nu^{(k+1)} = \nu^{(k)}$
33: **end if**
34: $\alpha^{(k+1)} = \min(\mu^{(k+1)}, \gamma_1)$
35: $\omega^{(k+1)} = \omega_0(\alpha^{(k+1)})^{\alpha_\omega}$
36: $\eta^{(k+1)} = \eta_0(\alpha^{(k+1)})^{\alpha_\eta}$
37: **end if**
38: **end for**

t parameter (Line 5) into the **while** loop (Lines 7–12) which will result in more frequent update of t. We store the resulting x and λ in Line 13 in order to warm start the second step in the next outer iteration $k+1$ (Line 6). Now, if the norm of the equality constraint residual is less than a certain threshold, we proceed with the third step (Lines 15-26). Otherwise, we proceed with the fourth step (Lines 28–36). In the third step, we first we carry out a termination test. If the termination condition is not satisfied, we keep the penalty parameter value the same, update the Lagrangian multipliers for the equality constraints under a certain condition, and then we update the free parameters. We then proceed with the second step. In the fourth step, we update the Lagrangian multiplier for the equality constraints under a certain condition, then we increase the penalty parameter and we update the free parameters. We then proceed with the second step. As suggested in [16], for a well-scaled problem, the typical values for the free parameters in the algorithm are $\alpha_\omega = \beta_\omega = \gamma = \zeta = \eta_0 = \omega_0 = 1$, $\alpha_\eta = \mu_0 = \gamma_1 = 0.1$, $\beta_\eta = 0.9$ and $\tau = 0.01$.

3. Distributed computation for the inner iteration

In this section we will see how we can distribute the computations of the inner iteration in Algorithm 2. Similar to our previous work, [2], we base our computations on what is known as a clique tree. See [33], for details about clique tree. We start by defining a coupling graph for the problem in ((1a)–(1b)). A coupling graph is an undirected graph with the node or vertex set $V_c = \{1, \dots, m\}$ and the edge set $\mathcal{E}_c$ with $(i, j) \in \mathcal{E}_c$ if and only if sub-systems i and j interact with each other, i.e. affect each others dynamics through states and inputs. In graph theory, a clique is a subset of vertices whose induced subgraph is complete, and a clique tree is a tree of cliques. In order to derive a clique tree, one might need to first carry out what is known as chordal embedding (also known as triangulation process). For the sake of brevity, we omit the details regarding that and we refer the reader to the introduction in [43]. In the context of this paper, a clique translates to a grouping of sub-systems and a clique tree will be used as a computational graph. Note that for the numerical experiment of our previous work, [2], we consider problems for which the clique tree is the same as the coupling graph. In this work, we consider more general problems where the clique tree of the problem is not trivial. Let us now consider a problem with 30 sub-systems where the coupling graph is illustrated in Fig. 1. There are various algorithms in the literature to generate the clique tree of the problem. One such method is presented in [32], which is used in the MATLAB library [51]. We use the same code to generate the clique tree. The resulting clique tree for the problem in Fig. 1 is illustrated in Fig. 2, where each clique contains the following sub-systems

$C_1 = \{\underline{1}, \underline{14}, \underline{15}, \underline{22}\}$,
$C_2 = \{14, 15, \underline{16}, 22, \underline{27}\}$,
$C_3 = \{\underline{5}, \underline{21}, 22\}$,
$C_4 = \{15, 16, \underline{19}, 22, 27, \underline{28}, \underline{29}, \underline{30}\}$,
$C_5 = \{\underline{6}, 14, 16\}$,
$C_6 = \{\underline{2}, 28\}$,
$C_7 = \{\underline{3}, 27, 29\}$,
$C_8 = \{\underline{7}, 15, 29\}$,
$C_9 = \{\underline{4}, \underline{8}, \underline{10}, 15, 16, 28\}$,
$C_{10} = \{\underline{11}, 15, 19, 29, 30\}$,
$C_{11} = \{\underline{12}, 28\}$,
$C_{12} = \{\underline{9}, \underline{13}, \underline{17}, \underline{18}, 19, \underline{20}, 22, \underline{23}, \underline{24}, \underline{25}, \underline{26}, 27, 28, 29, 30\}$.

Once the clique tree is found, we choose one of the cliques as the root of the tree. Let C_1 be the root. Next, we assign each sub-system to a clique. The assignment is not unique and can be done in different ways. Our strategy is as follows. A sub-system is assigned to a clique if it is not present in the parent clique. For example, sub-system 12 is assigned to C_{11} as it is not present in the parent clique of C_{11} which is C_4. For this problem, the assigned sub-systems for each clique are specified with underscore. Note that, to be precise, each clique comprises the variables associated with the contained sub-systems (not only the assigned ones), together with the variables of the sub-systems that they are interacting with.

We should note that, as mentioned earlier, in order to compute a clique tree, we employ a general purpose algorithm which takes the coupling graph as the input and generates what is known as chordal embedding [43], and the corresponding clique tree. Therefore, we cannot know a priori how the structure of the clique tree is. Obviously, if the coupling graph is very dense, we may end up with a clique tree which has a few number of cliques, and the worst case scenario is when all sub-systems are connected to each other, and in that case the clique tree will be a single clique. If the coupling graph for the sub-systems is loose or sparse, it is likely that the generated clique tree has many cliques, which is desirable. Note that if the generated clique tree is a chain of cliques, one can pick the middle clique as the root of the tree, and therefore the resulting clique tree will have two parallel branches. Last but not least, we should point out that we distribute the computations over sub-systems. One can further distribute the computations in each clique over time instances. In [27, Section 3.4], the au-

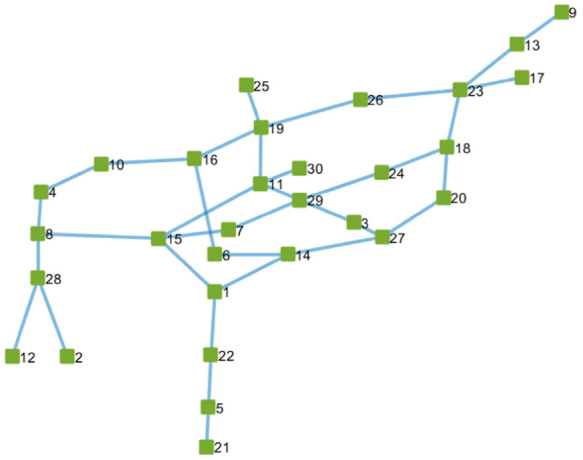

Fig. 1. Coupling graph for a problem with 30 sub-systems.

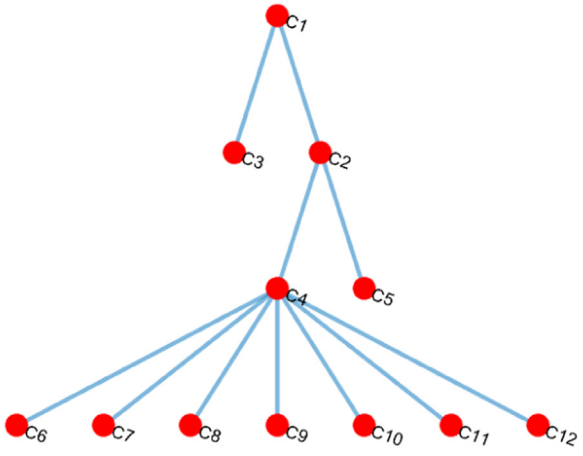

Fig. 2. The corresponding clique tree for the problem in Fig. 1.

thors show how parallel computations can be exploited over time instances for MPC problems.

Now let us write the problem in ((1a)–(1b)) in a more compact form as

$$\min_{x} \; F(x) = \sum_{i=1}^{q} F_i(x_{C_i}) \tag{13a}$$

$$\text{s.t. } G_i(x_{C_i}) = 0, \quad i = 1, \ldots, q, \tag{13b}$$

$$H_i(x_{C_i}) \leq 0, \quad i = 1, \ldots, q, \tag{13c}$$

where $F_i(x_{C_i})$ are the terms in $(1a)$ which are assigned to clique i. The vector $G_i(x_{C_i})$ is a vector obtained by stacking all the equality constraints in $(1b)$ which are assigned to clique i, on top of each other. Similarly, $H_i(x_{C_i})$ is a vector obtained by stacking all the inequality constraints in $(1b)$, which are assigned to clique i, on top of each other. The vector x_{C_i} contains all the variables in clique i, x is a vector including all the variables in the optimization problem and q is the total number of cliques.

The optimization problem for the search direction in ((11a)–(11b)) at iteration l, can be written distributedly over the cliques as

$$\min_{\Delta x} \; \sum_{i=1}^{q} \frac{1}{2} \Delta x_{C_i}^T \mathbf{H}_i^{(l)} \Delta x_{C_i} + \mathbf{r}_i^{(l)^T} \Delta x_{C_i} \tag{14a}$$

$$\text{s.t. } \mathbf{A}_i^{(l)} \Delta x_{C_i} = \mathbf{b}_i^{(l)}, \quad i = 1, \ldots, q. \tag{14b}$$

See Appendix B for the definitions of $\mathbf{H}_i^{(l)}$, $\mathbf{r}_i^{(l)^T}$, $\mathbf{A}_i^{(l)}$ and $\mathbf{b}_i^{(l)}$. Once Δx_{C_i} is computed, $\Delta \lambda_{C_i}$ can be obtained as

$$\Delta \lambda_{C_i} = -\mathbf{diag}\left(H_i(x_{C_i}^{(l)})\right)^{-1} \Bigg\{ \mathbf{diag}\left(\lambda_{C_i}^{(l)}\right) \frac{\partial H_i\left(x_{C_i}^{(l)}\right)}{\partial x_{C_i}^T} \Delta x_{C_i} + \mathbf{diag}\left(\lambda_{C_i}^{(l)}\right) H_i\left(x_{C_i}^{(l)}\right) + \left(\frac{1}{t}\right)\mathbf{1} \Bigg\}. \tag{15}$$

The optimization problem in ((14a)–(14b)) can be solved using message passing over the clique tree as outlined in [33]. Their approach is summarized briefly in the following. Consider the clique tree in Fig. 2. For each and every leaf optimize the term of the objective function that has been assigned to the leaf with respect to the variables that it does not share with its parent. The optimization should be done parametrically. The resulting optimal value, which is a function of the variables shared with the parent, is passed to the parent as a message. Once the parents receive the messages from all their children, they add them to their objective function terms and the same procedure is repeated assuming that the children cliques have been pruned away. Finally, we reach the root of the tree where the remaining variables are optimized. Then we can go down the tree and recover all optimal variables. This is based on the fact that we have stored the parametric optimal solutions in the nodes of the clique tree. See the introduction in [43], for an example of optimization over clique tree using message passing.

As it is discussed in Section 2, in order to see if the Hessian in (9) needs to be modified to get a decent direction, we need to know the *inertia* of the coefficient matrix in (8) at each iteration. Fortunately, we can calculate it distributedly, thanks to Sylvester's law of inertia, [23]. Let us consider an LDL^T factorization of a matrix $\mathbb{X}$, i.e. $\mathbb{X} = \mathbb{L}\mathbb{D}\mathbb{L}^T$, where $\mathbb{L}$ is a lower triangular square matrix with unity diagonal elements, $\mathbb{D}$ is a block diagonal matrix. It follows from Sylvester's law that $inertia(\mathbb{X}) = inertia(\mathbb{D})$. Now because of the fact that conducting message passing over a clique tree is the same as calculating an LDL^T factorization, [33, Section 6], we can calculate the *inertia* distributedly by summing up the *inertia* of the block diagonal elements of $\mathbb{D}$ which we get during the upward pass from leaves of the clique tree to the root. Therefore, Algorithm 1 can be run in the root of the clique tree to obtain the modification $\alpha_\sigma I$ of the overall Hessian. The overall Hessian is a sum of terms, see (14a). Moreover, Δx_{C_i} and Δx_{C_j}, for $i \neq j$, may have common variables. Hence, the overall Hessian depends on $\mathbf{H}_i^{(l)}$ in a complicated way, and care has to be taken when modifying each $\mathbf{H}_i^{(l)}$ so that the overall Hessian is modified by $\alpha_\sigma I$. The details of this are given in Appendix B; specifically notice the term $\alpha_\sigma \bar{E}_{C_i} \bar{E}_{C_i}^T$ in the computation of $\mathbf{H}_i^{(l)}$. Since the overall modifications can be carried out as modifications of $\mathbf{H}_i^{(l)}$, they distribute over the clique tree.

In order to be able to use the message-passing technique, there is a rank condition for the equality constraints in (14b) that should be imposed. This can be imposed in a similar way as in *Lemma 6.2* in [33]. However we need to impose them in every iteration since our original problem is non-convex. For the computations of step size and termination criteria, we again refer to [33, Section 6], where it is thoroughly explained how they can be distributedly computed over the clique tree.

Last but not least, it should be pointed out that all other computations in the outer iteration of Algorithm 2 can be carried out in the root of the clique tree. Note that what clique is chosen as root does not affect the number of communications between the cliques required for converging to a solution. However, it affects how computations can be carried out in parallel. See our previous work [1], for parallel exploitation of the clique tree.

4. Numerical experiments

4.1. Randomly generated interconnected sub-systems

For the first experiment, we consider model predictive control of a discrete-time system composed by the interconnection of m non-linear sub-systems. Let us assume that the following optimization problem needs to be solved at each iteration of the MPC

$$\min_{x,u} \; \sum_{i=1}^{m} \left(\sum_{k=1}^{N} r_x x_i(k)^4 + r_u u_i(k)^4 \right) + r_x x_i(N+1)^4 \tag{16a}$$

$$\begin{aligned} \text{s.t. } x_i(k+1) &= \alpha_i x_i(k)^3 + \beta_i u_i(k) + \sum_{j \in \mathcal{N}(i)} x_j(k), \\ & x_i(k)^2 + u_i(k)^2 \leq \bar{\gamma}_i, \\ & x_{i,\min} \leq x_i(k) \leq x_{i,\max}, \\ & u_{i,\min} \leq u_i(k) \leq u_{i,\max}, \\ & x_i(1) = \tilde{x}_i, \quad k = 1, \ldots, N, \quad i = 1, \ldots, m, \end{aligned} \tag{16b}$$

where $x_i(k) \in \mathbb{R}^{n_{x_i}}$ and $u_i(k) \in \mathbb{R}^{n_{u_i}}$ are the states and the inputs of sub-system i at time instant k, N is the horizon length and $\mathcal{N}(i)$ is the set of all indices of sub-systems which interact with sub-system i. $r_x \geq 0$, $r_u \geq 0$, α_i, β_i, $\bar{\gamma}_i \geq 0$, $x_{i,\min}$, $x_{i,\max}$, $u_{i,\min}$, $u_{i,\max}$ and $\tilde{x}_i$ are generated randomly. The last equation is the initial condition. We also generate all the coupling graphs in a random fashion and we make sure that they are connected. For all the randomly generated systems in the paper, the average number of sub-systems that each sub-subsystem interacts with, is 2.23.

We compare performance of the proposed algorithm, referred to as AL-PDIP (AL refers to Augmented Lagrangian and PDIP refers to Primal-Dual Interior-Point), with an ADMM based distributed algorithm proposed in [6], which is explained in the following. We refer to this algorithm as ADMM-original. In the algorithm, first local copies of coupling variables are defined for each sub-system. After that, consistency constraints with coordination variables are

introduced so that the local copies and the original variables coincide at optimality. The outline of the algorithm for each sub-system is given in Algorithm 3. Note that in forming the augmented Lagrangian in Step 2, only the corresponding cost function and the consistency constraints are used. The augmented Lagrangian is then minimized subject to the local dynamics by fixing the coordination variables. Note also that for the convergence test in Step 4, a central coordinator determines whether the criterion is satisfied for all sub-systems which requires a global communication at the end of each iteration. The first part of Step 2 in which an optimization problem needed be solved is the most computationally heavy part of the algorithm.

Algorithm 3 The ADMM algorithm outline in [6]

1: **Local initialization**
2: **Local minimization**

- minimize the corresponding augmented Lagrangian subject to the local dynamics by fixing the coordination variables
- receive local copy of coupling variables values from neighbors
- Compute local coordination variables
- receive coordination variables values from neighbors

3: **Local multiplier update**

- compute local Lagrangian multipliers
- compute local copies of Lagrangian multipliers for neighbors

4: **Stopping criterion**

- quit if termination criterion is satisfied
- otherwise return to Step 2

It should be noted that the way the computations are distributed in AL-PDIP algorithm and the ADMM based algorithm in [6], are different. In the AL-PDIP algorithm, in order to carry out the computations, it is enough to assign one computational agent per clique. Therefore, there will be as many computational agents as the number of the cliques. This is related to how we distribute the computations which is based on the clique tree. In the ADMM based algorithm, however, one computational agent per sub-system is assigned. Hence, there will be as many computational agents as the number of the sub-systems. One can, therefore, argue that in a sense the ADMM based algorithm is further distributed than AL-PDIP algorithm. However, the advantage of having one computational agent per clique is that in general the number of communications are lower compared to the case where there is one computational agent per sub-system. This property can be useful when the communications are more costly than carrying out the computations. It should also be mentioned that one might expect that the computations carried out in a clique (AL-PDIP algorithm) may be heavier than the computations carried out in a sub-system (ADMM based algorithm). This is not necessarily the case. The reason is that in the former case, at each iteration the dominant computation is matrix factorization, whereas in the latter case at each iteration, a local optimization problem should be solved which can be costly. It should also be stressed that our proposed AL-PDIP algorithm is limited to using the clique tree, and the form of this tree depends on the type of coupling we have. There are actually examples for which we would not be able to distribute the computations at all. This is the case when each sub-system is connected to all other sub-systems in the coupling graph, or when the generated clique tree is a single clique. However, there are many cases for which our method works and as we will see in the rest of the section, our method outperforms ADMM based methods by far.

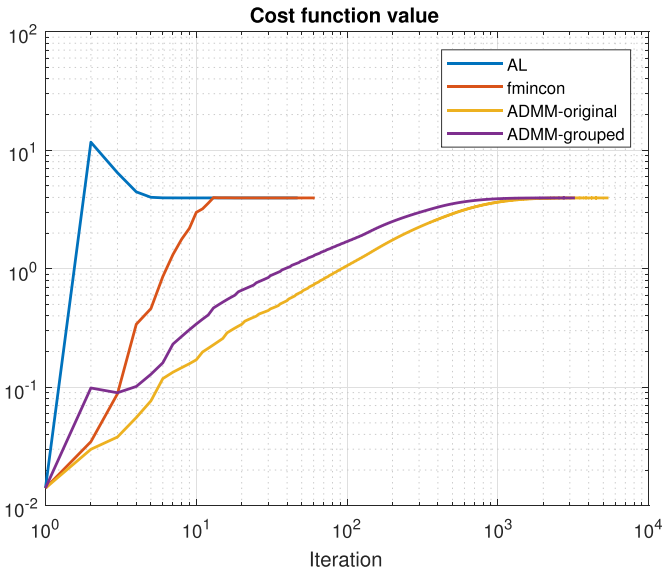

Fig. 3. The objective function value in (16a) for a system with 10 interconnected sub-systems.

In order to have a fair comparison with the ADMM based algorithm, we also solve the problem by applying the ADMM algorithm on the clique tree of the problem. In other words, each clique which is a grouping of sub-systems, is considered as a new sub-system with the clique tree being the new coupling graph of the problem, and then we apply ADMM to this new structure of the problem. We refer to this as ADMM-grouped.

Additionally, we solve the problem using `fmincon` (Interior-point method) in MATLAB as a benchmark. Termination criteria are set to default values which is 10^{-6} for the derivative of objective function and the equality constraint residual norm. As for the AL-PDIP algorithm, we terminate when the derivative of the augmented Lagrangian, the equality constraints residuals norm and the surrogate duality gap are below 10^{-6}. The termination criterion for the ADMM based algorithm is rather different. It is based on the norm of difference of current and previous iterates of the Lagrangian multipliers and the coordination variable values. In order to ensure a fair comparison, we define the solution obtained from `fmincon` as the optimal cost. The algorithm is then terminated when it converges to the optimal cost with tolerance of 10^{-6} and the equality constraint residual norm is below 10^{-4}. Notice that the latter threshold is looser than the corresponding threshold chosen for the AL-PDIP algorithm, which is 10^{-6}. The reason will become apparent later, and it is related to the fact that the ADMM based algorithm requires many iterations to converge. The time horizon N is set to 4 and all algorithms are initialized with the same initial value of the optimization variables, which is generated randomly from a normal distribution.

In the first setup, we consider a system with 10 interconnected sub-systems. The objective function value in (16a) for all algorithms is shown in Fig. 3. As can be seen, both the ADMM-original and ADMM-grouped approaches require many more iterations to converge than the AL-PDIP algorithm. The ADMM-grouped approach converges in less number of iterations than the ADMM-original approach. This, however, comes at the price that the local optimization problems in the ADMM-grouped approach are larger and hence computationally more expensive to solve than the ADMM-original approach.

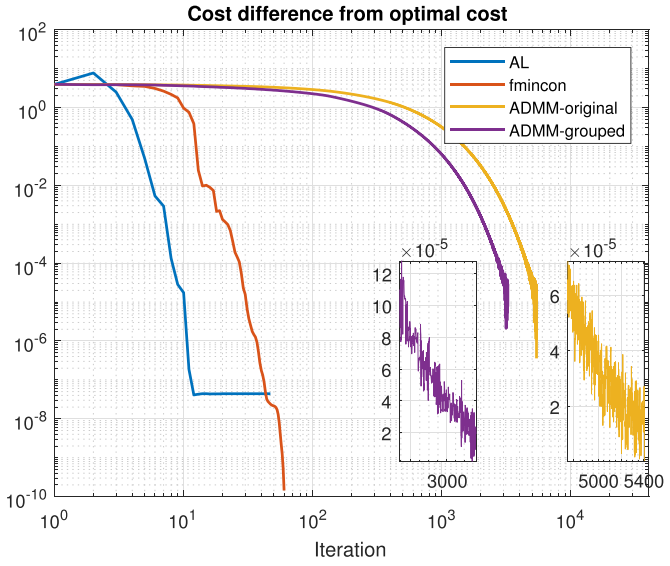

Fig. 4. Objective value difference from the optimal cost for a system with 10 interconnected sub-systems.

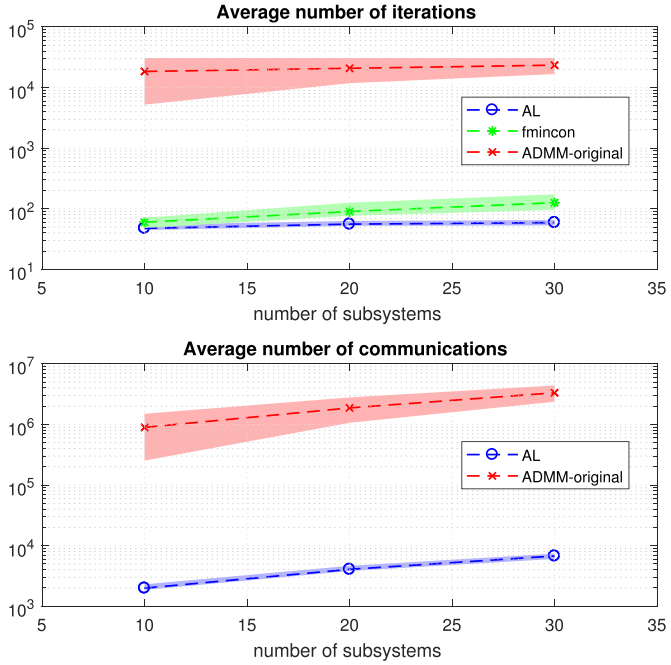

Fig. 5. The minimum, maximum and average number of iterations (top figure) and communications (bottom figure) over 5 randomly generated problems for AL-PDIP, fmincon and ADMM-original approaches.

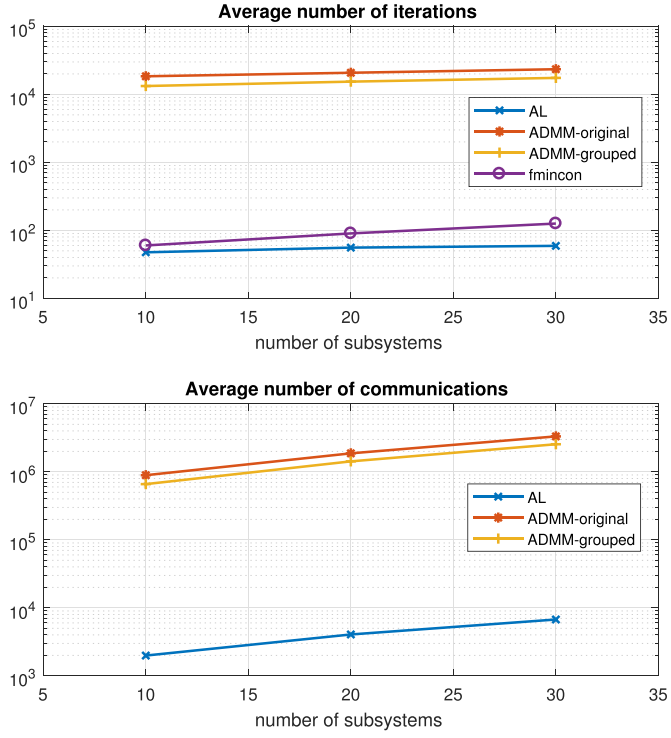

Fig. 6. The average number of iterations (top figure) and communications (bottom figure) over 5 randomly generated problems for AL-PDIP, fmincon, ADMM-original and ADMM-grouped approaches.

In Fig. 4, for each algorithm the norm of differences of the objective function values from the optimal cost (obtained from `fmincon`) is shown. As can be seen, for this particular problem, the AL-PDIP algorithm converges to an accurate solution in less than 100 iterations (inner iterations). As for the ADMM-grouped and ADMM-original approaches, they converge to a modest accuracy ($\sim 10^{-2}$) in around 2000 and 3000 iterations, respectively. For high accuracy, however, as can be seen, both approaches struggle, where the ADMM-original approach struggles slightly more as there are more agents in ADMM-original to reach to a consensus compared to ADMM-grouped. This issue becomes even more severe when the number of sub-systems increases.

In the second setup, we consider problems with 10, 20 and 30 sub-systems and for each problem, we generate 5 problems randomly. The average number of iterations (total number of inner iterations for the AL-PDIP approach and total number of outer iterations for the ADMM approach) and communications between sub-systems required for converging to a solution over these problems are shown in Figs. 5 and 6. The shaded areas in Fig. 5 depict the maximum and minimum values. We do not include ADMM-grouped in Fig. 5 to avoid cluttering the graph and instead, we include all the algorithms in Fig. 6 where we do not show the shaded areas. It should be pointed out that in the simulations for both ADMM-original and ADMM-grouped approaches, the maximum number of iterations is 30,000 and for some problems the algorithm did not converge within this threshold. To be precise, for the ADMM-original approach, 2, 2 and 1 out of the 5 problems with 10, 20 and 30 sub-systems are converged within this threshold. For the ADMM-grouped approach, 4, 4 and 2 out of the 5 problems with 10, 20 and 30 sub-systems converged within this threshold. In addition, for both ADMM-original and ADMM-grouped approaches, 1, 1 and 2 out of the 5 problems with 10, 20 and 30 sub-systems did not converge to an acceptable solution within this threshold. For both ADMM-original and ADMM-grouped approaches in Figs. 5 and 6, we assigned 30,000 iterations for the problems which did not converge within the considered threshold. We can see that the AL-PDIP algorithm requires much less number of iterations for convergence than both ADMM based approaches. ADMM-grouped approach requires slightly less number of iterations for convergence than the ADMM-original approach.

As for the total number of communications, we proceed as follows. For the proposed AL-PDIP algorithm, it can be calculated as

$$m_{\text{tree}} \times (3 \times \text{iter}_{\text{in}} + 3 \times \text{iter}_{\text{Hessian}} + \text{iter}_{\text{step}} + \text{iter}_{\text{out}}),$$

where m_{tree} is the number of edges in the clique tree, $iter_{in}$ is the total number of iterations through the clique tree for the inner iteration of the algorithm, $iter_{Hessian}$ is the total number of iterations through the clique tree required for modifying the Hessian, $iter_{step}$ is the total number of iterations through the clique tree required computing the step size, $iter_{out}$ is the total number of iterations through the clique tree for the outer iteration of the algorithm.

For both ADMM approaches, the total number of communications is

$$iter_{tot} \times (2 \times m_{system} + n_{system}),$$

where $iter_{tot}$ is the total number of iterations, m_{system} is the total number of couplings between the sub-systems and n_{system} is the total number of the sub-systems. The term $2 \times m_{system}$ corresponds to the exchange of local copy variables and coordination variables among neighboring sub-systems and the term n_{system} corresponds to the global communication of sub-systems with a central unit which evaluates the termination criteria of the algorithm. The communicated data are of the type matrix/vector and vector for the AL-PDIP and ADMM based algorithms, respectively. As can be seen in Figs. 5 and 6, the proposed AL-PDIP algorithm requires two order of magnitude less number of communications compared to both ADMM based approaches.

It should be pointed out that the message-passing scheme can be viewed as a distributed multi-frontal indefinite block LDL^T factorization technique that relies on fixed pivoting, [33]. This reliance is imposed by the structure of the problem which can in turn make the algorithm vulnerable to numerical problems, such as propagation of rounding errors due to ill-posed sub-problems. For the problem in ((1a)–(1b)), as the number of sub-systems increases, it will be more likely that one will get at least one pivot that is ill-conditioned. Note that this issue is not related to the convergence properties of the algorithm, but rather it is related to the conditioning of the problem. We can overcome this issue by merging the cliques. In particular, whenever the step size becomes too small in a couple of consecutive iterations and the algorithm does not make progress, we proceed as follows. When calculating the search direction, we store the condition numbers of the matrices to be inverted in all cliques and then we terminate the algorithm. After that, we merge the clique with the highest condition number with its parent clique and we run the algorithm again. If it fails again, we repeat the same procedure until we get a clique tree for which the algorithm converges successfully. Note that the worst case scenario is to end up with a clique tree with one clique which is equivalent to solving the centralized version of the problem. For numerical experiments, we consider the third setup consisting of a large number of sub-systems with dense coupling graphs. In particular, we generate 5 problems with different coupling graphs with 25, 50 and 75 sub-systems, respectively. The average number of cliques for the original clique tree for which the algorithm failed to converge and the average number of cliques after merging the cliques for which the algorithm converged successfully, is shown in Fig. 7. Note that for a few problems, we did not need to merge the cliques and the algorithm converged successfully for the original clique tree.

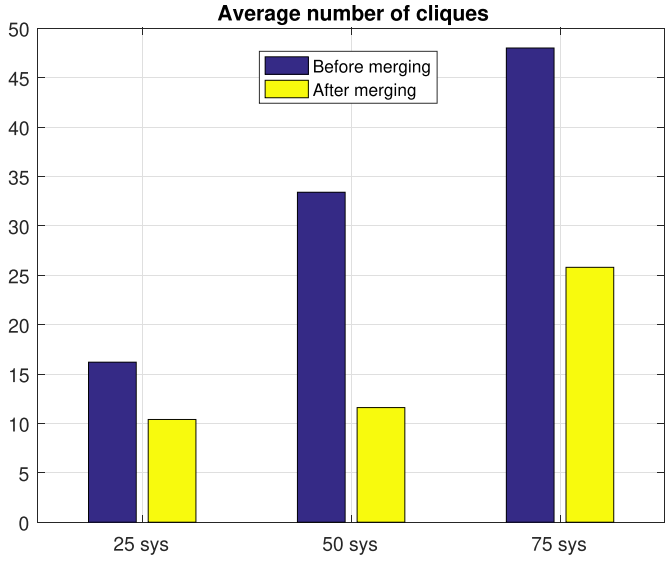

Fig. 7. The average number of cliques before and after merging over 5 randomely generated problems.

4.2. Vehicle platooning problem

For the second experiment, we consider the problem of distributed model predictive control of a platoon of vehicles with non-linear dynamics, [42], [19,35,36,49]. In addition, we consider collision avoidance constraint to assure that each vehicle in the platoon stays at a safe distance away from the neighboring vehicles. To be specific, we consider the optimization problem

$$\min_{x,u} \sum_{i=1}^{m}\sum_{k=1}^{N} \left(x_i(k)-\bar{x}_i^r\right)^2 + \left(y_i(k)-\bar{y}_i^r\right)^2 + v_i(k)^2 + \omega_i(k)^2 \quad (17a)$$

$$\text{s.t. } x_i(k+1) = x_i(k) + T_s v_i(k)\cos(\theta_i(k)), \quad (17b)$$

$$y_i(k+1) = y_i(k) + T_s v_i(k)\sin(\theta_i(k)), \quad (17c)$$

$$\theta_i(k+1) = \theta_i(k) + \frac{1}{L} T_s v_i(k)\tan(\psi_i(k)), \quad (17d)$$

$$\psi_i(k+1) = \psi_i(k) + T_s \omega_i(k), \quad (17e)$$

$$\underline{x}_i \le x_i(k) \le \bar{x}_i, \quad \underline{y}_i \le y_i(k) \le \bar{y}_i, \quad (17f)$$

$$\underline{\theta}_i \le \theta_i(k) \le \bar{\theta}_i, \quad \underline{\psi}_i \le \psi_i(k) \le \bar{\psi}_i, \quad (17g)$$

$$\underline{v}_i \le v_i(k) \le \bar{v}_i, \quad \underline{\omega}_i \le \omega_i(k) \le \bar{\omega}_i, \quad (17h)$$

$$x_i(1) = \tilde{x}_i, \quad y_i(1) = \tilde{y}_i, \quad \theta_i(1) = \tilde{\theta}_i, \quad (17i)$$

$$\psi_i(1) = \tilde{\psi}_i, \quad k = 1,\dots,N, \quad i = 1,\dots,m, \quad (17j)$$

$$\left\| \begin{bmatrix} x_i(k) \\ y_i(k) \end{bmatrix} - \begin{bmatrix} x_{i+1}(k) \\ y_{i+1}(k) \end{bmatrix} \right\|_2 \ge \tilde{d}, \quad k=1,\dots,N, \quad i=1,\dots,m-1, \quad (17k)$$

where $x_i(k)$, $y_i(k)$, $\theta_i(k)$ and $\psi_i(k)$ are the states and $v_i(k)$ and $\omega_i(k)$ are the inputs for the ith vehicle, $i = 1,\dots,m$ where m is the number of vehicles. For the ith vehicle, $(x_i(k), y_i(k))$ is the position of vehicles center, $\theta_i(k)$ is the orientation angle and $\psi_i(k)$ is the steering angle of the front wheels with respect to the vehicles body. The quantities $v_i(k)$ and $\omega_i(k)$ denote the speed and the steering rate, respectively. The vector $(\bar{x}_i^r, \bar{y}_i^r)$ is the fixed reference signal for the ith vehicle, L is the wheelbase length, T_s is the sampling time and N refers to the horizon length. The quantities $\underline{x}_i$, $\underline{y}_i$, $\underline{\theta}_i$, $\underline{\psi}_i$, $\underline{v}_i$, $\underline{\omega}_i$ are the fixed lower bounds on the states and the inputs. Similarly, $\bar{x}_i$, $\bar{y}_i$, $\bar{\theta}_i$, $\bar{\psi}_i$, $\bar{v}_i$, $\bar{\omega}_i$ are the fixed upper bounds on the states and the inputs. Finally, $\tilde{x}_i$, $\tilde{y}_i$, $\tilde{\theta}_i$ and $\tilde{\psi}_i$ are

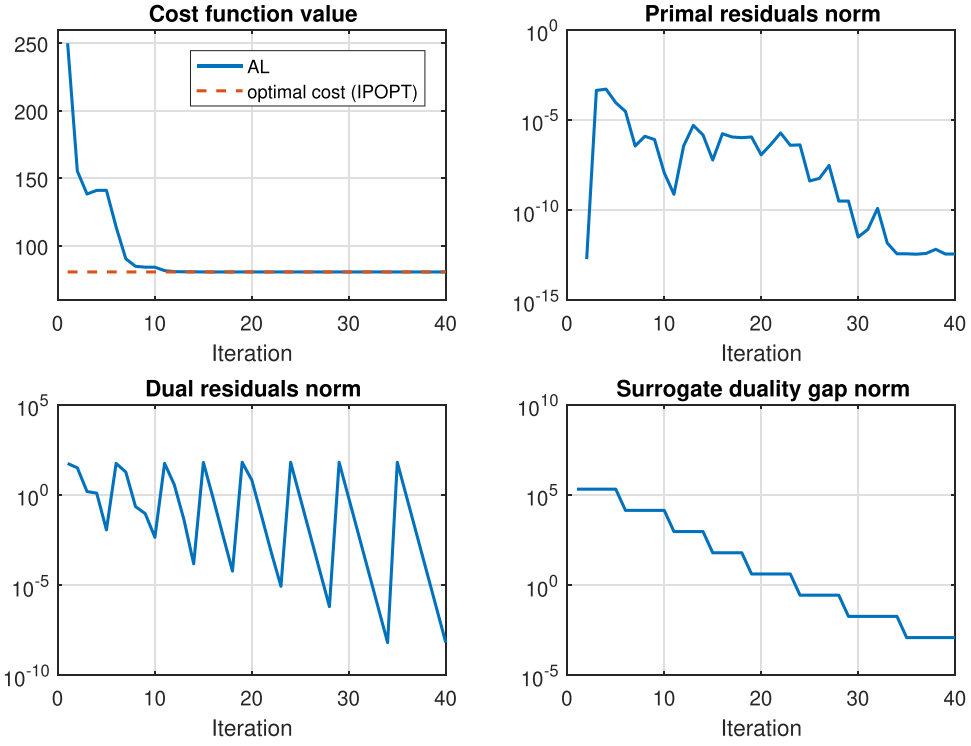

Fig. 8. The cost function values for AL-PDIP and IPOPT (top left figure), the primal residuals norm (top right figure), the dual residuals norm (bottom left figure) and the surrogate duality gap (bottom right figure) for $m = 100$.

Table 1
Parameters for the problem in ((17a)–(17k)) (for $i = 1, \dots, m$).

Parameter	Value	Parameter	Value
$\underline{x}_i$	-100	$\bar{x}_i$	100
$\underline{y}_i$	-100	$\bar{y}_i$	100
$\underline{\theta}_i$	$-\pi$	$\bar{\theta}_i$	π
$\underline{\psi}_i$	$\frac{-\pi}{2}$	$\bar{\psi}_i$	$\frac{\pi}{2}$
$\underline{v}_i$	-0.5	$\bar{v}_i$	0.5
$\underline{\omega}_i$	-0.005	$\bar{\omega}_i$	0.005
$\tilde{\theta}_i$	$\frac{\pi}{4}$	$\tilde{\psi}_i$	0
$(\tilde{x}_i^r, \tilde{y}_i^r)$	(i, i)	$(\tilde{x}_i^r, \tilde{y}_i^r)$	$(i-0.5, i-0.5)$

the initial values for the states and $\tilde{d}$ is the specified safe distance between the neighboring vehicles. The equality constraints ((17b)–(17e)) are the vehicles dynamic, [39]. The inequalities ((17f)–(17h)) are the box constraints on the states and inputs and the equality constraints ((17i)–(17j)) are the initial conditions for the states. Finally, the inequality constraint (17k) is the collision avoidance constraint for the neighboring vehicles. Note that for this problem the interaction between sub-systems is through the collision avoidance constraint. Note also that both the coupling graph and the clique tree for this problem is similar and can be described with a chain. The number of cliques in the clique tree is m and each clique i contains the variables associated with ith vehicle, plus the position variables of the $(i+1)$th vehicle.

Let $T_s = 1$, $L = 0.1$ $N = 5$, $\tilde{d} = 1$ and the other parameters be as in Table 1. Let also $(\tilde{x}_i^r, \tilde{y}_i^r) = (i, i)$ and $(\tilde{x}_i^r, \tilde{y}_i^r) = (i-0.5, i-0.5)$, for $i = 1, \dots, m$.

In the first setup, for the given values we solve the problem with the proposed AL-PDIP algorithm for 100 vehicles, i.e. $m = 100$. The results are shown in Fig. 8. The top left figure is the cost function value. We also solve the problem using IPOPT. It can be seen that both algorithms converge to the same value. It is worth noting that fmincon was not able to solve the problem and therefore we chose IPOPT for the benchmark purpose. The top right figure is the primal residuals norm which is the equality constraints residual norm. The bottom left figure is the dual residuals norm which is the gradient of the augmented Lagrangian and the bottom right figure is the surrogate duality gap norm which concerns the inequality constraints. As can be seen, all the residuals converge to zero eventually.

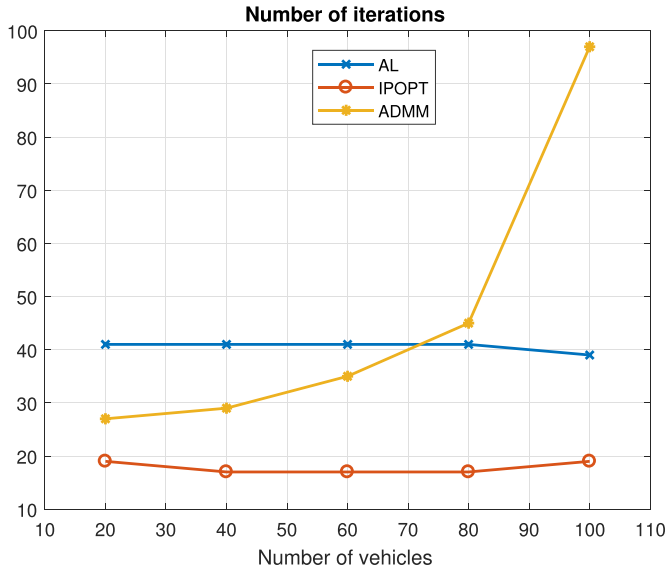

Fig. 9. The number of iterations for AL-PDIP, IPOPT and ADMM based approaches.

In the second setup, we consider problems with 20, 40, 60, 80 and 100 vehicles, respectively. We then solve the problems with AL-PDIP, IPOPT and the ADMM approaches. Note that the ADMM-original and ADMM-grouped approaches are the same for this problem as the coupling graph and the clique tree is identical. We choose the same termination criterion as before. The number of iterations for the mentioned approaches is shown in Fig. 9. As can

be seen, here the ADMM approach preforms well when the number of vehicles is low compared to the experiments in Section 4.1, in which the ADMM based approaches required much more number of iterations for convergence. The main reason is that for this problem the coupling graph which is a chain, is much looser than for the experiments in Section 4.1, where the interaction between sub-systems are more strong. Nevertheless, even for this problem, as can be seen in the figure, as soon as we increase the number of the vehicles, the number of iterations for the ADMM approach grows significantly. For the proposed AL-PDIP approach, however, the number of iteration for convergence does not depend on the number of vehicles. Last but not least, it should be stressed that each iteration of the ADMM approach consists of solving m local optimization problems which is computationally much more heavy than the computations at each iteration of the AL-PDIP algorithm, specially when m is large.

5. Conclusion

In this paper, we presented a distributed algorithm for optimal control of coupled systems, with application to model predictive control. The algorithm is based on an augmented Lagrangian approach in which a primal-dual interior-point method is used for the inner iteration. We distributed the computations for search direction, step size, and termination criteria over a clique tree of the problem and calculated each of them using message passing. We showed the superiority of the algorithm in terms of number of iterations and communications using a set of numerical examples. For future work we intend to investigate the possibility of distributing the preconditioned conjugate gradient computations of [13].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partially founded by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, which is gratefully acknowledged.

Appendix A

Let E_{C_i} be the zero-one matrices such that $E_{C_i}x = x_{C_i}$ for all $i = 1, \dots, q$. The gradients and Hessians of F, G_is and H_is for the problem in ((13a)–(13c)) can be derived as

$$\frac{\partial F(x)}{\partial x} = \sum_{i=1}^{q} E_{C_i}^T \frac{\partial F_i(x_{C_i})}{\partial x_{C_i}},$$

$$\frac{\partial^2 F(x)}{\partial x \partial x^T} = \sum_{i=1}^{q} E_{C_i}^T \frac{\partial^2 F_i(x_{C_i})}{\partial x_{C_i} \partial x_{C_i}^T} E_{C_i},$$

$$\frac{\partial \left(G_i(x_{C_i})\right)_j}{\partial x} = E_{C_i}^T \frac{\partial \left(G_i(x_{C_i})\right)_j}{\partial x_{C_i}}, \quad j = 1, \dots, p_i,$$

$$\frac{\partial^2 \left(G_i(x_{C_i})\right)_j}{\partial x \partial x^T} = E_{C_i}^T \frac{\partial^2 \left(G_i(x_{C_i})\right)_j}{\partial x_{C_i} \partial x_{C_i}^T} E_{C_i}, \quad j = 1, \dots, p_i,$$

$$\frac{\partial \left(H_i(x_{C_i})\right)_j}{\partial x} = E_{C_i}^T \frac{\partial \left(H_i(x_{C_i})\right)_j}{\partial x_{C_i}}, \quad j = 1, \dots, q_i,$$

$$\frac{\partial^2 \left(H_i(x_{C_i})\right)_j}{\partial x \partial x^T} = E_{C_i}^T \frac{\partial^2 \left(H_i(x_{C_i})\right)_j}{\partial x_{C_i} \partial x_{C_i}^T} E_{C_i}, \quad j = 1, \dots, q_i,$$

where p_i and q_i are the number of equality and inequality constraints in clique i, respectively.

Appendix B

First see Appendix A, for the gradients and Hessians of F, G_is and H_is in ((13a)–(13c)). Let us now define the modifications of E_{C_i} that we call $\bar{E}_{C_i}$. They are obtained by identifying the non-zero columns which E_{C_i} have in common with $E_{C_{par(i)}}$, where $par(i)$ is the parent of the ith clique in the clique tree. Then $\bar{E}_{C_i}$ is defined to be equal to E_{C_i}, except for these columns, which are set equal to zero. With this definition, $\mathbf{H}_i^{(l)}$, $\mathbf{r}_i^{(l)^T}$, $\mathbf{A}_i^{(l)}$ and $\mathbf{b}_i^{(l)}$ in problem ((14a)–(14b)) can be written as

$$\begin{aligned}\mathbf{H}_i^{(l)} = {} & \frac{\partial^2 F\left(x_{C_i}^{(l)}\right)}{\partial x_{C_i} \partial x_{C_i}^T} + \sum_{j=1}^{q_i} \lambda_j^{(l)} \frac{\partial^2 \left(H_i(x_{C_i}^{(l)})\right)_j}{\partial x_{C_i} \partial x_{C_i}^T} \\ & \sum_{j=1}^{p_i} \left[\left(\nu_j^{(l)} + \frac{1}{\mu} \left(G_i(x_{C_i}^{(l)})\right)_j \right) \frac{\partial^2 \left(G_i(x_{C_i}^{(l)})\right)_j}{\partial x_{C_i} \partial x_{C_i}^T} \right. \\ & \left. + \frac{1}{\mu} \frac{\partial \left(G_i(x_{C_i}^{(l)})\right)_j}{\partial x_{C_i}} \frac{\partial \left(G_i(x_{C_i}^{(l)})\right)_j}{\partial x_{C_i}^T} \right] + \alpha_\sigma \bar{E}_{C_i} \bar{E}_{C_i}^T \\ & - \frac{\partial H_i(x_{C_i}^{(l)})^T}{\partial x_{C_i}} \mathbf{diag}\left(H_i(x_{C_i}^{(l)})\right)^{-1} \mathbf{diag}\left(\lambda_{C_i}^{(l)}\right) \frac{\partial H_i(x_{C_i}^{(l)})}{\partial x_{C_i}^T},\end{aligned}$$

$$\begin{aligned}\mathbf{r}_i = {} & \frac{\partial F_i(x_{C_i}^{(l)})}{\partial x_{C_i}} + \sum_{j=1}^{q_i} \lambda_j^{(l)} \frac{\partial \left(H_i(x_{C_i}^{(l)})\right)_j}{\partial x_{C_i}} \\ & + \sum_{j=1}^{p_i} \left(\nu_j^{(l)} + \frac{1}{\mu} (G_i(x_{C_i}^{(l)}))_j \right) \frac{\partial \left(G_i(x_{C_i}^{(l)})\right)_j}{\partial x_{C_i}} \\ & - \frac{\partial H_i(x_{C_i}^{(l)})^T}{\partial x_{C_i}} \mathbf{diag}\left(H_i(x_{C_i}^{(l)})\right)^{-1} \left(\mathbf{diag}(\lambda_{C_i}^{(l)}) H_i(x_{C_i}^{(l)}) + \left(\frac{1}{t}\right)\mathbf{1} \right),\end{aligned}$$

$$\mathbf{A}_i^{(l)} = \begin{bmatrix} \frac{\partial \left(G_i(x_{C_i}^{(l)})\right)_1}{\partial x_{C_i}^T} \\ \vdots \\ \frac{\partial \left(G_i(x_{C_i}^{(l)})\right)_{p_i}}{\partial x_{C_i}^T} \end{bmatrix}, \quad \mathbf{b}_i^{(l)} = -G_i\left(x_{C_i}^{(l)}\right).$$

Note that if the Hessian does not need to be modified, then $\alpha_\sigma = 0$.

References

[1] S.P. Ahmadi, A. Hansson, Parallel exploitation for tree-structured coupled quadratic programming in Julia, in: Proceedings of the 22nd International Conference on System Theory, Control and Computing (ICSTCC), IEEE, 2018, pp. 597–602.
[2] S.P. Ahmadi, A. Hansson, A distributed second-order augmented lagrangian method for distributed model predictive control, IFAC-PapersOnLine 54 (6) (2021) 192–199.
[3] M. Argáez, R. Tapia, On the global convergence of a modified augmented lagrangian linesearch interior-point newton method for nonlinear programming, J. Optim. Theory Appl. 114 (1) (2002) 1–25.
[4] D.P. Bertsekas, Nonlinear programming, J. Oper. Res. Soc. 48 (3) (1997) 334.
[5] D.P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Academic Press, 2014.
[6] A. Bestler, K. Graichen, Distributed model predictive control for continuous–time nonlinear systems based on suboptimal ADMM, Optim. Control Appl. Methods 40 (1) (2019) 1–23.
[7] E.G. Birgin, J.M. Martínez, Improving ultimate convergence of an augmented lagrangian method, Optim. Methods Softw. 23 (2) (2008) 177–195.

[8] S. Boyd, N. Parikh, E. Chu, Distributed optimization and statistical learning via the alternating direction method of multipliers, Now Publishers Inc, 2011.
[9] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.
[10] D. Burk, A. Völz, K. Graichen, Towards a modular framework for distributed model predictive control of nonlinear neighbor-affine systems, in: Proceedings of the 58th IEEE Conference on Decision and Control (CDC), IEEE, 2019, pp. 5279–5284.
[11] D. Burk, A. Völz, K. Graichen, A modular framework for distributed model predictive control of nonlinear continuous-time systems (GRAMPC-D), Optim. Eng. 23 (2) (2022) 771–795.
[12] E.F. Camacho, C.B. Alba, Springer Science & Business Media, 2013.
[13] Y. Cao, A. Seth, C.D. Laird, An augmented lagrangian interior-point approach for large-scale nlp problems on graphics processing units, Comput. Chem. Eng. 85 (2016) 76–83.
[14] P.D. Christofides, R. Scattolini, D.M. de la Pena, J. Liu, Distributed model predictive control: a tutorial review and future research directions, Computers & Chemical Engineering 51 (2013) 21–41.
[15] A.R. Conn, N. Gould, P.L. Toint, Numerical experiments with the LANCELOT package (release a) for large-scale nonlinear optimization, Math. Program. 73 (1) (1996) 73.
[16] A.R. Conn, N.I. Gould, P. Toint, A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, SIAM J. Numer. Anal. 28 (2) (1991) 545–572.
[17] C. Conte, T. Summers, M.N. Zeilinger, M. Morari, C.N. Jones, Computational aspects of distributed optimization in model predictive control, in: Proceedings of the 51th IEEE Conference on Decision and Control (CDC), IEEE, 2012, pp. 6819–6824.
[18] Q.T. Dinh, I. Necoara, M. Diehl, A dual decomposition algorithm for separable nonconvex optimization using the penalty function framework, in: Proceedings of the 52th IEEE Conference on Decision and Control (CDC), IEEE, 2013, pp. 2372–2377.
[19] W.B. Dunbar, D.S. Caveney, Distributed receding horizon control of vehicle platoons: stability and string stability, IEEE Trans. Autom. Control 57 (3) (2011) 620–633.
[20] A. Engelmann, Y. Jiang, B. Houska, T. Faulwasser, Decomposition of nonconvex optimization via bi-level distributed Aladin, IEEE Trans. Control Netw. Syst. 7 (4) (2020) 1848–1858.
[21] F. Farokhi, I. Shames, K.H. Johansson, Distributed MPC via dual decomposition and alternative direction method of multipliers, in: Distributed Model Predictive Control Made Easy, Springer, 2014, pp. 115–131.
[22] C.E. Garcia, D.M. Prett, M. Morari, Model predictive control: theory and practicea survey, Automatica 25 (3) (1989) 335–348.
[23] P.E. Gill, W. Murray, M.A. Saunders, M.H. Wright, Inertia-controlling methods for general quadratic programming, SIAM Rev. 33 (1) (1991) 1–36.
[24] P.E. Gill, D.P. Robinson, A primal-dual augmented Lagrangian, Comput. Optim. Appl. 51 (1) (2012) 1–25.
[25] P. Giselsson, M.D. Doan, T. Keviczky, B. De Schutter, A. Rantzer, Accelerated gradient methods and dual decomposition in distributed model predictive control, Automatica 49 (3) (2013) 829–833.
[26] A. Grancharova, T.A. Johansen, Distributed MPC of interconnected nonlinear systems by dynamic dual decomposition, in: Distributed Model Predictive Control Made Easy, Springer, 2014, pp. 293–308.
[27] A. Hansson, S.K. Pakazad, Exploiting chordality in optimization algorithms for model predictive control, in: Large-Scale and Distributed Optimization, Springer, 2018, pp. 11–32.
[28] S. Hentzelt, K. Graichen, An augmented Lagrangian method in distributed dynamic optimization based on approximate neighbor dynamics, in: Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2013, pp. 571–576.
[29] M.R. Hestenes, Multiplier and gradient methods, J. Optim. Theory Appl. 4 (5) (1969) 303–320.
[30] X. Hou, Y. Xiao, J. Cai, J. Hu, J.E. Braun, Distributed model predictive control via proximal Jacobian ADMM for building control applications, in: Proceedings of the 2017 American Control Conference (ACC), IEEE, 2017, pp. 37–43.
[31] B. Houska, J. Frasch, M. Diehl, An augmented lagrangian based algorithm for distributed nonconvex optimization, SIAM J. Optim. 26 (2) (2016) 1101–1127.
[32] F.V. Jensen, F. Jensen, Optimal junction trees, in: Uncertainty Proceedings 1994, Elsevier, Morgan Kaufmann, 1994, pp. 360–366.
[33] S. Khoshfetrat Pakazad, A. Hansson, M.S. Andersen, I. Nielsen, Distributed primal–dual interior-point methods for solving tree-structured coupled convex problems using message-passing, Optim. Methods Softw. 32 (3) (2017) 401–435.
[34] M. Kögel, R. Findeisen, Cooperative distributed MPC using the alternating direction multiplier method, IFAC Proc. Vol. 45 (15) (2012) 445–450.
[35] S.E. Li, Y. Zheng, K. Li, Y. Wu, J.K. Hedrick, F. Gao, H. Zhang, Dynamical modeling and distributed control of connected and automated vehicles: challenges and opportunities, IEEE Intell. Transp. Syst. Mag. 9 (3) (2017) 46–58.
[36] P. Liu, A. Kurt, U. Ozguner, Distributed model predictive control for cooperative and flexible vehicle platooning, IEEE Trans. Control Syst. Technol. 27 (3) (2018) 1115–1128.
[37] N. Meyer-Huebner, M. Suriyah, T. Leibfried, Distributed optimal power flow in hybrid AC–DC grids, IEEE Trans. Power Syst. 34 (4) (2019) 2937–2946.
[38] J.M. Moguerza, F.J. Prieto, An augmented lagrangian interior-point method using directions of negative curvature, Math. Program. 95 (3) (2003) 573–616.
[39] F. Mohseni, E. Frisk, L. Nielsen, Distributed cooperative MPC for autonomous driving in different traffic scenarios, IEEE Trans. Intell. Veh. 6 (2) (2020) 299–309.
[40] I. Necoara, C. Savorgnan, D.Q. Tran, J. Suykens, M. Diehl, Distributed nonlinear optimal control using sequential convex programming and smoothing techniques, in: Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, IEEE, 2009, pp. 543–548.
[41] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[42] S.K. Pakazad, Divide and conquer: Distributed optimization and robustness analysis, Department of Electrical Engineering, Linköping University, 2015.
[43] S. Parvini Ahmadi, Distributed Optimization for Control and Estimation, Linköping University Electronic Press, 2022. Ph.D. thesis
[44] M.J. Powell, A method for nonlinear constraints in minimization problems, Optimization (1969) 283–298.
[45] R. Rostami, G. Costantini, D. Görges, ADMM-based distributed model predictive control: primal and dual approaches, in: Proceedings of the 56th IEEE Conference on Decision and Control (CDC), IEEE, 2017, pp. 6598–6603.
[46] H. Scheu, W. Marquardt, Sensitivity-based coordination in distributed model predictive control, J. Process Control 21 (5) (2011) 715–728.
[47] B.T. Stewart, S.J. Wright, J.B. Rawlings, Cooperative distributed model predictive control for nonlinear systems, J. Process Control 21 (5) (2011) 698–704.
[48] T.H. Summers, J. Lygeros, Distributed model predictive consensus via the alternating direction method of multipliers, in: Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2012, pp. 79–84.
[49] R. Van Parys, G. Pipeleers, Distributed MPC for multi-vehicle systems moving in formation, Robot. Auton. Syst. 97 (2017) 144–152.
[50] 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) (2008) 1192–1206.
[51] X. Xie, A recursive method to learn Bayesian network, 2020, https://se.mathworks.com/matlabcentral/fileexchange/20678-a-recursive-method-to-learn-bayesian-networkl.