Delft University of Technology

Delft Center for Systems and Control

Technical report 09-034

A distributed version of Han's method for DMPC of dynamically coupled systems with coupled constraints*

D. Doan, T. Keviczky, I. Necoara, M. Diehl, and B. De Schutter

If you want to cite this report, please use the following reference instead:

D. Doan, T. Keviczky, I. Necoara, M. Diehl, and B. De Schutter, "A distributed version of Han's method for DMPC of dynamically coupled systems with coupled constraints," *Proceedings of the 1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys 2009)*, Venice, Italy, pp. 240–245, Sept. 2009. doi:10.3182/20090924-3-IT-4005.00041

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/09_034.html

A Distributed Version of Han's Method for DMPC of Dynamically Coupled Systems with Coupled Constraints ⋆

Dang Doan * **Tamás Keviczky** * **Ion Necoara** **
Moritz Diehl *** **Bart De Schutter** *

* *Delft University of Technology, Delft, The Netherlands (e-mail: {m.d.doan,t.keviczky}@tudelft.nl, b@deschutter.info)*
** *Politehnica University of Bucharest, Bucharest, Romania (e-mail: i.necoara@ics.pub.ro)*
*** *K.U.Leuven, Heverlee, Belgium (e-mail: moritz.diehl@esat.kuleuven.be)*

Abstract: Most of the literature on Distributed Model Predictive Control (DMPC) for dynamically coupled linear systems typically focuses on situations where coupling constraints between subsystems are absent. In order to address the presence of convex coupling constraints, we present a distributed version of Han's parallel algorithm for a class of convex programs. The algorithm we propose relies on local iterative updates only, instead of using system-wide information exchange as in Han's original algorithm. The new algorithm is then used to develop a new distributed MPC method that is applicable to sparsely coupled linear dynamical systems with coupled linear constraints. Convergence to the global optimum, recursive feasibility, and stability can be established using only local communications between the subsystems.

Keywords: distributed optimization and control, model predictive control, large-scale systems

1. INTRODUCTION

Model predictive control (MPC) (Maciejowski, 2002; Mayne et al., 2000) is a very popular controller design method in the process industry. A key advantage of MPC is that it can accommodate hard constraints on the inputs, states, and outputs of the controlled system. In essence, MPC is an on-line receding-horizon control approach in which a model is used to predict the future behavior of the system and in which a cost criterion is optimized subject to constraints on the inputs, states, and outputs.

For large-scale systems *centralized* MPC is considered to be impractical, inflexible, and unsuitable due to its computational and information exchange requirements. In order to deal with these limitations, *distributed model predictive control* (DMPC) has been proposed, by decomposing the overall system into small subsystems (Jia and Krogh, 2001; Camponogara et al., 2002; Rawlings and Stewart, 2008). The subsystems employ distinct MPC controllers that solve local optimization problems, use local information from neighboring subsystems only, and collaborate to achieve globally attractive solutions.

Several classes of DMPC approaches can be distinguished. Dunbar and Murray (2006) proposed a DMPC scheme for systems with decoupled dynamics, focusing on multiple vehicles with coupled cost functions, and utilizing predicted trajectories of the neighbors in each subsystem's optimization. A DMPC scheme with a sufficient stability test for dynamically decoupled systems was proposed by Keviczky et al. (2006), in which each subsystem optimizes also over the behaviors of its neighbors. Richards and How (2007) proposed a robust DMPC method for decoupled systems with coupled constraints, based on constraint tightening and a serial solution approach. For systems with coupled dynamics and decoupled constraints Venkat et al. (2008) proposed a distributed MPC scheme, based on a Jacobi algorithm that deals with the primal problem, using a convex combination of new and old solutions.

We propose a DMPC algorithm that is able to handle linear time-invariant dynamics with linear dynamical couplings and coupled linear constraints. Each local controller will only need to communicate with its direct neighbors to exchange predictions, which are iteratively updated by the local controllers. The algorithm can be implemented using only local communications, while guaranteeing global feasibility and stability.

This paper is organized as follows. Section 2 describes the setup of the problem. In Section 3, the centralized MPC problem is formulated. The resulting optimization problem can be solved using a parallel computing scheme based on Han's method, which is summarized in Section 4. The main contribution of this paper is then presented in the form of a distributed algorithm exploiting the structure of the optimization problem for local communications, followed by the proof of its equivalence to Han's algorithm in Section 5. As a consequence of this equivalence, the proposed DMPC scheme using this distributed optimization

⋆ Research supported by the European 7th framework STREP project "Hierarchical and distributed model predictive control (HD-MPC)", contract number INFSO-ICT-223854.

procedure achieves the global optimum upon convergence and thus inherits feasibility and stability properties from its centralized MPC counterpart.

2. PROBLEM DESCRIPTION

Consider a plant consisting of M subsystems. The dynamics of each subsystem are assumed to be influenced directly by only a *small number* of other subsystems. Moreover, each subsystem i is assumed to have local linear coupled constraints involving only variables from a small number of other subsystems. We define the *neighborhood of subsystem* i, $\mathcal{N}^i$, as the set of indices of subsystem i and the subsystems that have either a direct dynamical or linear constraint coupling with subsystem i. In order to benefit from an increased computational speed when using a distributed algorithm, the couplings between subsystems are assumed to be sparse, i.e., the size of each neighborhood $\mathcal{N}^i$ is relatively small in comparison with the total number of subsystems M.

So we assume that each subsystem can be represented by a discrete-time, linear time-invariant model of the form:

$$x_{k+1}^i = \sum_{j\in\mathcal{N}^i} A^{ij}x_k^j + B^{ij}u_k^j, \tag{1}$$

where $x_k^i \in \mathbb{R}^{n^i}$ and $u_k^i \in \mathbb{R}^{m^i}$ are the states and control inputs of the i-th subsystem at time step k, respectively.

Moreover, each subsystem i is assumed to have local linear coupled constraints involving only variables within its neighborhood $\mathcal{N}^i$. Let N be the prediction horizon. All constraints that subsystem i is involved in are written as

$$\sum_{j\in\mathcal{N}^i}\sum_{k=0}^{N-1} D_k^{ij}x_k^j + E_k^{ij}u_k^j = c_{\text{eq}} \tag{2}$$

$$\sum_{j\in\mathcal{N}^i}\sum_{k=0}^{N-1} \bar{D}_k^{ij}x_k^j + \bar{E}_k^{ij}u_k^j \le \bar{c}_{\text{ineq}} \tag{3}$$

3. CENTRALIZED MPC PROBLEM

We will formulate the centralized MPC problem for systems of the form (1) using a *terminal point constraint* approach that imposes that all states are steered to 0 at the end of the prediction horizon. Under the conditions that a feasible solution of the centralized MPC problem exists, and that the point with zero states and inputs is in the relative interior of the constraint set, this MPC scheme ensures feasibility and stability, as shown by Mayne et al. (2000) and Keerthi and Gilbert (1988). We will further assume without loss of generality that the initial time is 0.

3.1 Problem formulation

The optimization variable of the centralized MPC problem is constructed as a stacked vector of predicted subsystem control inputs *and* states over the prediction horizon:

$$\mathbf{x} = \Big[\big(u_0^1\big)^T,\dots,\big(u_0^M\big)^T,\dots,\big(u_{N-1}^1\big)^T,\dots,\big(u_{N-1}^M\big)^T, \big(x_1^1\big)^T,\dots,\big(x_1^M\big)^T,\dots,\big(x_N^1\big)^T,\dots,\big(x_N^M\big)^T\Big]^T \tag{4}$$

Recall that N denotes the prediction horizon and that n^i and m^i denote the numbers of states and inputs of subsystem i. The size of $\mathbf{x}$ is thus equal to $n_{\mathbf{x}} = N\sum_{i=1}^M m^i + N\sum_{i=1}^M n^i$.

The cost function of the centralized MPC problem is assumed to be decoupled and convex quadratic:

$$J = \sum_{i=1}^{M}\sum_{k=0}^{N-1}\Big(\big(u_k^i\big)^T R_i u_k^i + \big(x_{k+1}^i\big)^T Q_i x_{k+1}^i\Big) \tag{5}$$

with positive definite weights R_i, Q_i. It is easy to verify that this cost function can be rewritten as $J = \mathbf{x}^T H\mathbf{x}$ where H is a block-diagonal, positive definite matrix.

The overall centralized MPC problem is then defined as:

$$\min_{\mathbf{x}} \quad \mathbf{x}^T H\mathbf{x} \tag{6}$$

s.t.

$$x_{k+1}^i = \sum_{j\in\mathcal{N}^i} A^{ij}x_k^j + B^{ij}u_k^j, \quad i = 1,\dots,M, \quad k = 0,\dots,N-1 \tag{7}$$

$$x_N^i = 0, \quad i = 1,\dots,M \tag{8}$$

$$\sum_{j\in\mathcal{N}^i}\sum_{k=0}^{N-1} D_k^{ij}x_k^j + E_k^{ij}u_k^j = c_{\text{eq}}, \quad i = 1,\dots,M \tag{9}$$

$$\sum_{j\in\mathcal{N}^i}\sum_{k=0}^{N-1} \bar{D}_k^{ij}x_k^j + \bar{E}_k^{ij}u_k^j \le \bar{c}_{\text{ineq}}, \quad i = 1,\dots,M \tag{10}$$

3.2 Centralized optimization problem

We can rewrite the problem (6)–(10) in a compact form as

$$\begin{aligned}\min_{\mathbf{x}} \quad & \mathbf{x}^T H\mathbf{x} \\ \text{s.t.} \quad & a_l^T\mathbf{x} = b_l, \quad l = 1,\dots,n_{\text{eq}} \\ & a_l^T\mathbf{x} \le b_l, \quad l = n_{\text{eq}}+1,\dots,s\end{aligned} \tag{11}$$

with $s = n_{\text{eq}} + n_{\text{ineq}}$. Note that due to sparse couplings between subsystems, a_l has very few non-zero elements.

4. HAN'S ALGORITHM

First, we summarize the main elements of Han's method (Han and Lou, 1988) for a class of convex programs, followed by a simplified version for the case of *definite quadratic programs (QPs)*.

4.1 Han's algorithm for general convex problems

The class of optimization problems tackled by Han's algorithm is defined as follows:

$$\begin{aligned}\min_{\mathbf{x}} \quad & q(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in C \triangleq C_1 \cap \cdots \cap C_s\end{aligned} \tag{12}$$

where $q(\mathbf{x})$ is uniformly convex and differentiable on $\mathbb{R}^{n_{\mathbf{x}}}$ and where $C_1, \cdots, C_s$ are closed convex sets and $C \neq \varnothing$.

Algorithm 1. **Han's method for convex programs**
The algorithm is an iterative procedure. We use p as iteration counter of the algorithm and a superscript (p) to denote the values of variables computed at iteration p.

Let α be a sufficiently large numberand define $\mathbf{y}^{(0)} = \mathbf{y}_1^{(0)} = \cdots = \mathbf{y}_s^{(0)} = 0$, with $\mathbf{y}^{(0)}, \mathbf{y}_l^{(0)} \in \mathbb{R}^{n_{\mathbf{x}}}, l = 1,\dots,s$, and $\mathbf{x}^{(0)} = \nabla q^*\left(\mathbf{y}^{(0)}\right)$ with q^* being the conjugate

function [1] of q. For $p = 1, 2, \ldots$, we perform the following computations:

1) For $l = 1, \ldots, s$, find $\mathbf{z}_l^{(p)}$ that solves
$$\min_{\mathbf{z}} \quad \frac{1}{2}\|\mathbf{z} + \alpha\mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\|_2^2 \qquad (13)$$
$$\text{s.t.} \quad \mathbf{z} \in C_l$$
2) Assign
$$\mathbf{y}_l^{(p)} = \mathbf{y}_l^{(p-1)} + (1/\alpha)\left(\mathbf{z}_l^{(p)} - \mathbf{x}^{(p-1)}\right) \qquad (14)$$
3) Set $\mathbf{y}^{(p)} = \mathbf{y}_1^{(p)} + \cdots + \mathbf{y}_s^{(p)}$
4) Compute $\mathbf{x}^{(p)} = \nabla q^*\left(\mathbf{y}^{(p)}\right)$.

Han and Lou (1988) showed that $\|\mathbf{y}^{(p)} - \mathbf{y}^{(p-1)}\|_2 \to 0$ and $\|\mathbf{x}^{(p)} - \mathbf{x}^{(p-1)}\|_2 \to 0$ as $p \to \infty$. They also showed that their algorithm converges to the global optimum if $q(\mathbf{x})$ is uniformly convex and differentiable on $R^{n_\mathbf{x}}$.

4.2 Han's algorithm for definite QPs

In case the optimization problem has a positive definite cost function and linear constraints as in (11), the optimization problem (13) and ∇q^* have analytical solutions, and then Han's method becomes simpler. In the following we revise how the analytical solutions of (13) and ∇q^* can be obtained when applying Algorithm 1 to problem (11). Note that the result of simplifying Han's method in this section is slightly different from the original one described in Han and Lou (1988), to correct the minor mistakes we found in that paper.

As in (11), each constraint $\mathbf{x} \in C_l$ is implicitly expressed by a scalar linear equality or inequality constraint. So (13) takes one of following two forms:
$$\min \quad \frac{1}{2}\|\mathbf{z} + \alpha\mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\|_2^2 \qquad (15)$$
$$\text{s.t.} \quad a_l^T\mathbf{z} = b_l$$
or
$$\min \quad \frac{1}{2}\|\mathbf{z} + \alpha\mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\|_2^2 \qquad (16)$$
$$\text{s.t.} \quad a_l^T\mathbf{z} \le b_l$$

First consider (16):

- If $a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) \le b_l$, then $\mathbf{z}_l^{(p)} = \mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}$ is the solution of (16). Substituting this $\mathbf{z}_l^{(p)}$ into (14), leads to the following update of $\mathbf{y}_l^{(p)}$:
$$\mathbf{y}_l^{(p)} = \mathbf{y}_l^{(p-1)} + (1/\alpha)\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)} - \mathbf{x}^{(p-1)}\right)$$
$$\Rightarrow \mathbf{y}_l^{(p)} = 0 \qquad (17)$$
- If $a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) > b_l$, then the constraint is active. The optimization problem (16) is to find the point in the half-space $a_l^T\mathbf{z} \le b_l$ that minimizes its distance to the point $\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}$ (which is outside that half-space). The solution is the projection of the point $\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}$ on the hyperplane $a_l^T\mathbf{z} = b_l$, which is given by the following formula:
$$\mathbf{z}_l^{(p)} = \mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)} - \frac{a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l\right) - b_l}{a_l^T a_l}a_l \qquad (18)$$
Substituting this $\mathbf{z}_l^{(p)}$ into (14), leads to:
$$\mathbf{y}_l^{(p)} = \mathbf{y}_l^{(p-1)} + \frac{1}{\alpha}\left(-\alpha\mathbf{y}_l^{(p-1)} - \frac{a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) - b_l}{a_l^T a_l}a_l\right)$$
$$= -\frac{a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}_l^{(p-1)}\right) - b_l}{\alpha a_l^T a_l}a_l \qquad (19)$$
Then defining $\gamma_l^{(p)} = a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}^{(p-1)}\right) - b_l$ yields
$$\mathbf{y}_l^{(p)} = -\frac{\gamma_l^{(p)}}{\alpha a_l^T a_l}a_l \qquad (20)$$
If we define $\gamma_l^{(p)} = \max\{a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}^{(p-1)}\right) - b_l, 0\}$, then we can use the update formula (20) for both cases.

Similarly, for the minimization under equality constraint (15), we define $\gamma_l^{(p)} = a_l^T\left(\mathbf{x}^{(p-1)} - \alpha\mathbf{y}^{(p-1)}\right) - b_l$ and the update formula (20) gives the result of (14).

Now consider step 4) of Algorithm 1. As shown by Boyd and Vandenberghe (2004), the function $q(\mathbf{x}) = \mathbf{x}^T H\mathbf{x}$ with H being a positive definite matrix, is strongly convex and has the conjugate function $q^*(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T H^{-1}\mathbf{y}$. Hence, $\nabla q^*(\mathbf{y}) = H^{-1}\mathbf{y}$. Consequently, in Han's algorithm for the *definite QP* (11), it is not necessary to compute $\mathbf{z}^{(p)}$, and $\mathbf{y}^{(p)}$ can be eliminated using (20), which leads to the following simplified algorithm:

Algorithm 2. **Han's method for definite QPs**
The optimization problem to be considered is (11). As discussed in Han and Lou (1988), we choose $\alpha = s/\rho$, where $s = n_{\text{eq}} + n_{\text{ineq}}$ is the number of constraints and ρ is one half of the smallest eigenvalue of H.

For $l = 1, \ldots, s$, compute $c_l = \dfrac{-1}{\alpha a_l^T a_l}H^{-1}a_l$.

Initialize $\gamma_1^{(0)} = \cdots = \gamma_s^{(0)} = 0$ and $\mathbf{x}^{(0)} = 0$. For $p = 1, 2, \ldots$, we perform the following computations:

1) For each l corresponding to an equality constraint $(1 \le l \le n_{\text{eq}})$, compute $\gamma_l^{(p)} = a_l^T\mathbf{x}^{(p-1)} + \gamma_l^{(p-1)} - b_l$. For each l corresponding to an inequality constraint $(n_{\text{eq}} + 1 \le l \le s)$, compute $\gamma_l^{(p)} = \max\{a_l^T\mathbf{x}^{(p-1)} + \gamma_l^{(p-1)} - b_l, 0\}$;
2) Set
$$\mathbf{x}^{(p)} = \sum_{l=1}^{s}\gamma_l^{(p)}c_l \qquad (21)$$

Note that Han's method splits up the computation into s parallel subproblems, with s the number of constraints. Although Algorithm 2 is simpler than the original form in Algorithm 1, it still requires a *global update scheme* and the parallel problems still operate with the full-sized decision

[1] The conjugate function of a function $q(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}}$ is defined by: $q^*(\mathbf{y}) = \sup_{\mathbf{x}\in\mathbb{R}^{n_\mathbf{x}}}\left(\mathbf{y}^T\mathbf{x} - q(\mathbf{x})\right)$.

vector. Implementing the scheme in a DMPC system, where the goal is to reduce the size of local computations and to rely on local communication between subsystems only, is not straightforward. In the following section, we will exploit the structure of the problem (11), resulting in a distributed algorithm that does not require global communications.

5. DISTRIBUTED ALGORITHM FOR THE CENTRALIZED MPC OPTIMIZATION PROBLEM

For our algorithm, we use M local controllers attached to M subsystems. Each controller i then computes $\gamma_l^{(p)}$ with regard to a small set of constraints indexed by l. Subsequently, it performs a local update for its own variables, such that the parallel local update scheme will be equivalent to the global update scheme in Algorithm 2.

5.1 Initialization of the algorithm

We choose α and compute s invariant values c_l as in Algorithm 2:

$$c_l = \frac{-1}{\alpha a_l^T a_l} H^{-1} a_l, \quad l = 1, \ldots, s \tag{22}$$

in which each c_l corresponds to one constraint of (11). Recall that for the centralized MPC problem (6)–(10), H is block-diagonal; so the same holds for H^{-1}. Hence, c_l is as sparse as the corresponding a_l. We can see that c_l can be computed locally by a local controller with *a priori* knowledge of the parameter a_l and the weighting blocks on the diagonal of H that correspond to the non-zero elements of a_l.

We assume that each local controller i knows its local dynamics, and the input and state weights of its neighbors in the cost function. Then each local controller i can compute the c_l associated with its dynamic equality constraints.

5.2 Assign responsibility of each local controller

Each local controller is in charge of updating the variables of its subsystem. Moreover, we also assign to each local controller the responsibility of updating *some* intermediate variables that relate to several equality or inequality constraints in which its subsystem's states or inputs appear. The control designer has to assign each of the s scalar constraints to one of the M local controllers [2] such that the following requirements are satisfied:

- Each constraint is taken care of by one and only one local controller (even for a coupled constraint, there will be only one controller that is responsible).
- A local controller can only be in charge of constraints that involve its own variables.

Note that in general this division is not unique. Let L_i denote the set of indices l that local controller i is in charge of. We also define $L_{\mathcal{N}^i}$ as the set of indices l corresponding to the constraints that are taken care of by subsystem i or by any neighbor of i: $L_{\mathcal{N}^i} = \bigcup_{j \in \mathcal{N}^i} L_j$.

If a local controller is in charge of the constraints indexed by ℓ, then it computes c_ℓ using (22) and exchanges these values with its neighbors. Then each local controller i stores $\{c_\ell\}_{\ell \in L_{\mathcal{N}^i}}$ in its memory throughout the optimization process.

[2] Note that s is often much larger than M.

5.3 Iterative procedure

The distributed algorithm consists of an iterative procedure running within each sampling interval. At each iteration, four steps are executed: two steps are communications between each local controller and its direct neighbors, and two are computation steps that are performed locally by controllers in parallel. Since feasibility is only guaranteed upon convergence of Han's algorithm, we assume that the sampling time used is large enough such that the algorithm can converge within one sampling interval.

Definition 5.1. (Index matrix of subsystems). In order to present the algorithm compactly, we introduce the *index matrix of subsystems*: each subsystem i has a square matrix $\mathfrak{I}^i \in R^{n_\mathbf{x} \times n_\mathbf{x}}$ that is diagonal, with an entry on the diagonal being 1 if it corresponds to the position of a variable of subsystem i in the vector $\mathbf{x}$, and 0 otherwise. In short, $\mathfrak{I}^i$ is a selection matrix such that the multiplication $\mathfrak{I}^i \mathbf{x}$ only retains the variables $u_0^i, \ldots, u_{N-1}^i, x_1^i, \ldots, x_N^i$ of subsystem i in its nonzero entries. We have

$$\sum_{i=1}^{M} \mathfrak{I}^i = I \tag{23}$$

Definition 5.2. (Self image). We denote with $\mathbf{x}^{(p)|i} \in \mathbb{R}^{n_\mathbf{x}}$ the vector that has the same size as $\mathbf{x}$, containing $u_0^{i,(p)}, \ldots, u_{N-1}^{i,(p)}, x_1^{i,(p)}, \ldots, x_N^{i,(p)}$ (i.e. the values of i's variables computed at iteration p) at the right positions, and zeros for the other entries. This vector is called the *self image* of $\mathbf{x}^{(p)}$ made by subsystem i. Using the *index matrix* notation, the relation between $\mathbf{x}^{(p)|i}$ and $\mathbf{x}^{(p)}$ is:

$$\mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}^{(p)} \tag{24}$$

Definition 5.3. (Neighborhood image). Extending the concept of *self image*, we denote with $\mathbf{x}^{(p)|\mathcal{N}^i}$ the *neighborhood image* of subsystem i made from $\mathbf{x}$. At step p of the iteration, subsystem i constructs $\mathbf{x}^{(p)|\mathcal{N}^i}$ by putting the values of its neighbors' variables and its own variables to the right positions, and filling in zeros for the remaining slots of $\mathbf{x}$. The *neighborhood image* $\mathbf{x}^{(p)|\mathcal{N}^i}$ satisfies

$$\mathbf{x}^{(p)|\mathcal{N}^i} = \sum_{j \in \mathcal{N}^i} \mathbf{x}^{(p)|j} \tag{25}$$

$$\mathbf{x}^{(p)|\mathcal{N}^i} = \left(\sum_{j \in \mathcal{N}^i} \mathfrak{I}^j \right) \mathbf{x}^{(p)} \tag{26}$$

By definition, we also have the following relation between the *self image* and the *neighborhood image* made by the same subsystem:

$$\mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}^{(p)|\mathcal{N}^i} \tag{27}$$

Algorithm 3. **Distributed algorithm for the centralized MPC optimization problem**

Initialize with $p = 0$, $x_k^{i,(0)} = 0, u_k^{i,(0)} = 0, \forall i, k \neq 0$ (this means $\mathbf{x}^{(0)|i} = 0, \forall i$, and the centralized variable $\mathbf{x}^{(0)} = 0$), and $\gamma_l^{(0)} = 0, l = 1, \ldots, s$ (recall that s is the number of constraints of the centralized optimization problem).

Next, for $p = 1, 2, \ldots$, the following steps are executed:

1) **Communications to get the updated main variables**
Each controller i communicates with its neighbors $j \in \mathcal{N}^i$ to get updated values of their variables, contained in $\mathbf{x}^{(p-1)|j}$. Vice versa, i also sends its updated variables in $\mathbf{x}^{(p-1)|i}$ to its neighbors as requested.

   After getting information from the neighbors, controller i constructs the *neighborhood image* $\mathbf{x}^{(p-1)|\mathcal{N}^i}$ using formula (25).

2) **Update intermediate variables γ_l in parallel**
In this step, the local controllers update γ_l corresponding to each constraint l under their responsibility. More specifically, each local controller i updates γ_l for each $l \in L_i$ in the following manner:
   - If constraint l is an equality constraint ($1 \le l \le n_{\text{eq}}$), then $\gamma_l^{(p)} = a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l$.
   - If constraint l is an inequality constraint ($n_{\text{eq}} + 1 \le l \le s$), then $\gamma_l^{(p)} = \max\{a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} + \gamma_l^{(p-1)} - b_l, 0\}$.

3) **Communications to get the updated intermediate variables**
Each local controller i communicates with its neighbors to get updated $\gamma_l^{(p)}$ values that the neighbors just computed in step 2).

4) **Update main variables in parallel**
Local controller i uses all $\gamma_l^{(p)}$ values that it has (by communications and those computed by itself) to compute an *assumed neighborhood image* of $\mathbf{x}$:

$$\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i} = \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \tag{28}$$

Note that $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ has the same structure as $\mathbf{x}^{(p-1)|\mathcal{N}^i}$. However, it is not the exact update of the *neighborhood image*, instead $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ is only an *assumed neighborhood image*. An interpretation will be given later (see Remark 5.4 below).

Then controller i selects the values of its variables in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ to construct the new *self image*:

$$\mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i} \tag{29}$$

which contains $u_0^{i,(p)}, \ldots, u_{N-1}^{i,(p)}, x_1^{i,(p)}, \ldots, x_N^{i,(p)}$.

After updating their variables, each local controller checks the local termination criteria. When all local controllers have converged [3], the algorithm stops and the local control actions are implemented, otherwise the controllers proceed to step 1) to start a new iteration.

Remark 5.4: Interpretation of the assumed neighborhood image
At the end of step 4), each local controller i has an *assumed neighborhood image* $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ of $\mathbf{x}$ that contains information within its interest (i.e., has non-zero values only corresponding to the variables within its neighborhood). However, controller i knows exactly only its own variables, while the variables of i's neighbors contained in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ are the assumption of controller i (since i does not know the interaction between its neighbors and their other neighbors, thus their updates will be different from what i assumes for them). Therefore, i only extracts $u_0^{i,(p)}, \ldots, u_{N-1}^{i,(p)}, x_1^{i,(p)}, \ldots, x_N^{i,(p)}$ from $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ and throws away the other values. The real *neighborhood image* will be made in the next iteration after i receives updated values of its neighbors. In fact, for the actual implementation of the algorithm, we can combine (28) and (29) since we do not need to use $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$.

Remark 5.5 The equivalence between global and local update schemes will be shown later in Section 5.4 by proving that $\mathbf{x}^{(p)} = \sum_{i=1}^M \mathbf{x}^{(p)|i}$ where $\mathbf{x}^{(p)}$ is the centralized variable update resulting from the global update scheme in Algorithm 2, while $\mathbf{x}^{(p)|i}$ are the updates at the end of step 4) in Algorithm 3.

5.4 Proof of equivalence to Han's algorithm using a global update scheme

In Algorithm 2, at step 2), the centralized variable $\mathbf{x}^{(p)}$ is updated via a global update scheme. In Algorithm 3, by the local update scheme we obtain $\mathbf{x}^{(p)|i}$ for $i = 1, \ldots, M$. The equivalence of these two algorithms is stated in the following proposition:

Proposition 1. Applying Algorithms 2 and 3 to the same problem (11) with the same parameter α, at any iteration p, the following properties hold:

a) $\gamma_l^{(p)}$ are the same in Algorithms 2 and 3, for all $l \in \{1, \ldots, s\}$.
b) $\mathbf{x}^{(p)} = \sum_{i=1}^M \mathbf{x}^{(p)|i}$, in which $\mathbf{x}^{(p)}$ is calculated in Algorithm 2 while $\mathbf{x}^{(p)|i}, i = 1, \ldots, M$ are calculated in Algorithm 3.

Hence, Algorithm 2 and Algorithm 3 are equivalent.

Proof: The proposition will be proved by induction.

It is clear that properties a) and b) hold for $p = 0$.

Now consider iteration p, and assume that the properties a) and b) hold for all iterations before iteration p.

First, we prove property a). For any l and i such that $l \in L_i$, we have:

$$\begin{aligned} a_l^T \mathbf{x}^{(p-1)} &= a_l^T \sum_{j=1}^M \mathfrak{I}^j \mathbf{x}^{(p-1)|j} \\ &= a_l^T \left(\sum_{j \in \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} + \sum_{j \notin \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} \right) \end{aligned} \tag{30}$$

Due to the definition of *neighborhood*, a subsystem outside $\mathcal{N}^i$ does not have any coupled constraints with subsystem i. Therefore, $a_l^T \sum_{j \notin \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} = 0$, which leads to:

$$a_l^T \mathbf{x}^{(p-1)} = a_l^T \sum_{j \in \mathcal{N}^i} \mathfrak{I}^j \mathbf{x}^{(p-1)|j} = a_l^T \mathbf{x}^{(p-1)|\mathcal{N}^i} \tag{31}$$

The second equality holds due to (25). Equation (31) guarantees that $\gamma_l^{(p)}$ computed at step 1) of Algorithm 2 and at step 2) of Algorithm 3 are the same.

[3] Checking the termination criteria in a distributed fashion requires a dedicated logic scheme, the description of which is omitted for brevity.

Now consider property b), where the main argument is the following: In order to calculate $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$, subsystem i uses all $\gamma_l^{(p)}$ and c_l that involve any variable of i. The updates of i's variables in $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$ are thus equal to the updates of i's variables made by the centralized scheme in $\mathbf{x}^{(p)}$ (in step 4) of Algorithm 2). The vector $\mathbf{x}^{(p)|i}$ only contains values of i's variables selected from $\mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i}$. Similarly, the updates made by each other subsystem for its variables are guaranteed to be the same as the results of the centralized update scheme. Making the sum of all $\mathbf{x}^{(p)|i}$ is similar to composing them into one vector, which leads to $\mathbf{x}^{(p)}$.

More specifically, we can express the formula of $\mathbf{x}^{(p)|i}$ computed in Algorithm 3 as

$$\begin{aligned}\mathbf{x}^{(p)|i} = \mathfrak{I}^i \mathbf{x}_{\text{assumed}}^{(p)|\mathcal{N}^i} &= \mathfrak{I}^i \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \\ \Rightarrow \sum_{i=1}^{M} \mathbf{x}^{(p)|i} &= \sum_{i=1}^{M} \mathfrak{I}^i \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \end{aligned} \tag{32}$$

Note that in the following equations, $\mathbf{x}^{(p)}$ refers to the update of the decision variable computed by (21) in Algorithm 2, which we can express as

$$\mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathfrak{I}^i \mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathfrak{I}^i \sum_{l=1}^{s} \gamma_l^{(p)} c_l \tag{33}$$

in which the first equality is due to the relation (23), the second equality is from (21).

Recall that c_l has the same structure as a_l, and if $l \notin L_{\mathcal{N}^i}$ then a_l and c_l do not have any non-zero values at the positions associated with variables of subsystem i. Therefore

$$\begin{aligned}\mathfrak{I}^i \sum_{l=1}^{s} \gamma_l^{(p)} c_l &= \mathfrak{I}^i \left(\sum_{l \notin L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l + \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \right) \\ &= \mathfrak{I}^i \sum_{l \in L_{\mathcal{N}^i}} \gamma_l^{(p)} c_l \end{aligned} \tag{34}$$

This equality shows that (33) and (32) are equivalent, thus proving the equality in property b): $\mathbf{x}^{(p)} = \sum_{i=1}^{M} \mathbf{x}^{(p)|i}$. □

The equivalence of Algorithms 2 and 3 implies that problem (11) can be solved using Algorithm 3. This allows us to implement a DMPC scheme using Algorithm 3 that does not need global communications. In this DMPC scheme, no computation using global variables is required; moreover, each local controller only needs to communicate with its direct neighbors and the only information to exchange is the updates of their predicted variables.

Convergence, feasibility and stability properties of the DMPC scheme using Algorithm 3 are established by the following corollaries (see also (Doan et al., 2009)):

Corollary 5.6. Assume that Q_i and R_i are positive definite for $i = 1, \ldots, M$, and (6)–(10) has a feasible solution. Then Algorithm 3 converges to the centralized solution of (6)–(10) at each sampling step.

Corollary 5.7. Assume that at every sampling step, Algorithm 3 converges. Then the DMPC scheme is recursively feasible and stable.

6. CONCLUSIONS

We have presented a distributed version of Han's method and proposed its use for distributed model predictive control of a class of linear time-invariant system with coupled dynamics and coupled linear constraints. The proposed approach makes use of local communications only between directly connected subsystems, which is especially beneficial in the case of sparse subsystem interconnection topologies. Global optimality is achieved, leading to feasibility and stability.

Future research topics include a detailed convergence analysis, extensive comparison with other DMPC algorithms, finding efficient communication schemes for checking the termination criteria, and relaxing the terminal point constraint requirement.

REFERENCES

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, Cambridge, MA.

Camponogara, E., Jia, D., Krogh, B., and Talukdar, S. (2002). Distributed model predictive control. *IEEE Control Systems Magazine*, 22(1), 44–52.

Doan, D., Keviczky, T., Necoara, I., Diehl, M., and De Schutter, B. (2009). A distributed version of Han's method for DMPC using local communications only. *Control Engineering and Applied Informatics*, 11(3).

Dunbar, W.B. and Murray, R.M. (2006). Distributed receding horizon control for multi-vehicle formation stabilization. *Automatica*, 42, 549–558.

Han, S.P. and Lou, G. (1988). A parallel algorithm for a class of convex programs. *SIAM Journal on Control and Optimization*, 26(2), 345–355.

Jia, D. and Krogh, B. (2001). Distributed model predictive control. In *American Control Conference*, volume 4, 2767–2772. Arlington, VA.

Keerthi, S.S. and Gilbert, E.G. (1988). Optimal infinite-horizon feedback control laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57(2), 265–293.

Keviczky, T., Borrelli, F., and Balas, G.J. (2006). Decentralized receding horizon control for large scale dynamically decoupled systems. *Automatica*, 42, 2105–2115.

Maciejowski, J.M. (2002). *Predictive Control with Constraints*. Prentice Hall, Harlow, England.

Mayne, D.Q., Rawlings, J.B., Rao, C.V., and Scokaert, P.O.M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(7), 789–814.

Rawlings, J.B. and Stewart, B.T. (2008). Coordinating multiple optimization-based controllers: New opportunities and challenges. *Journal of Process Control*, 18(9), 839–845.

Richards, A. and How, J. (2007). Robust distributed model predictive control. *International Journal of Control*, 80(9), 1517–1531.

Venkat, A., Hiskens, I., Rawlings, J., and Wright, S. (2008). Distributed MPC strategies with application to power system automatic generation control. *IEEE Transactions on Control Systems Technology*, 16(6), 1192–1206.