Distributed Model Predictive Control based on Dual Decomposition with Neural-Network-based Warm Start

P. Chanfreut, A. Sánchez-Amores, J. M. Maestre, and E. F. Camacho

***Abstract*— This work deals with the application of neural networks to speed up the convergence of a distributed model predictive control (DMPC) algorithm based on dual decomposition. While dual decomposition methods are known to converge to the centralized MPC solution, numerous iterations may be required before convergence is attained, thus increasing computation and communication burden. In this paper, a database containing system states and optimal Lagrange multipliers is created offline to train a neural network, which is incorporated into the online operation of the distributed system. Numerical results on an input-coupled 16 tanks benchmark are provided.**

I. Introduction

Over the past years, the resolution of optimization problems using distributed model predictive control (MPC) has drawn the attention of the research community [1], [2]. In this context, the global system is divided into several coupled subsystems that communicate to optimize an MPC *subproblem*. The distributed approach can preserve centralized performance, presenting significant benefits regarding flexibility and scalability [3], [4]. In this regard, distributed MPC results are particularly suitable when working with large-scale applications, as shown in [5]–[8]. However, non centralized architectures must limit to a minimum the amount of data shared [9], but a lack of coordination may notably decrease global performance in case of strong subsystems interactions. At the same time, distributed strategies can be broadly classified into non-cooperative and cooperative algorithms [2]. In non-cooperative approaches, such as communication-based MPC [10], each agent optimizes a local objective function that does not consider the control goals of the rest of the plant. Conversely, cooperative algorithms like *feasible cooperation-based MPC* in [11], or dual decomposition methods [12], reach the centralized optimum at convergence.

By using dual decomposition, the centralized MPC solution is computed in a distributed fashion following an iterative procedure where a set of auxiliary variables, the so-called Lagrange multipliers, are introduced to enforced constraints satisfaction [12], [13]. Nowadays, the application of dual decomposition techniques into network systems with numerous agents has become very popular [3], [14]. However, a major drawback of this approach is its slow convergence rate, which can involve high computation and communication burden. See for example [15], where the authors provide a quadratic approximation of the dual function, aiming at improving this rate.

With the recent rise of machine learning and artificial intelligence, the implementation of neural networks in complex engineering optimization problems has become widely used. Artificial neural networks present many possibilities to deal with large amounts of data, with multiple successful implementations in engineering optimization problems. In this regard, works such as [16], [17] introduce a neural network model predictive controller (NNMPC), where the neural networks are used to predict the dynamic model of the systems. A similar goal is pursued in [18], where instead of modeling the process, neural networks are employed for predicting the state vector over the prediction horizon. Furthermore, in [19], [20], the authors propose to use directly neural networks to minimize a quadratic cost function and obtain the control law. In this paper, neural networks will be trained to predict optimal Lagrange multipliers for a distributed MPC scheme based on dual decomposition. In particular, each agent optimizes a function where Lagrange multipliers enforce an agreement on coupled actions that affect local dynamics. The neural network models are incorporated into the distributed control scheme so as to provide an initialization of the Lagrange prices at each time instant. As will be seen, this start results in solutions close to the centralized outcome. Consequently, online computational burden will be reduced, speeding up the convergence of a distributed MPC based on dual decomposition.

The rest of the paper is organized as follows. In Section II, the model of the system is presented and the global control problem is formulated. Section III describes the dual-decomposition algorithm and how Lagrange multipliers are learnt. In Section IV, the idea is tested using a simulated 16 tanks benchmark. Finally, concluding remarks are given in Section V.

II. Problem formulation

Consider a global system partitioned into a set $\mathcal{N} = \{1, 2, ..., N\}$ of input-coupled subsystems with LTI dynamics:

$$x_i^+ = A_{ii}x_i + B_{ii}u_i + w_i, \text{ with } w_i = \sum_{j \in \mathcal{N}_i} B_{ij}u_j, \quad (1)$$

where $x_i \in \mathbb{R}^{n_{x_i}}$ and $u_i \in \mathbb{R}^{n_{u_i}}$ are respectively the state and input vectors of subsystem $i \in \mathcal{N}$ for each time instant k, and $w_i \in \mathbb{R}^{n_{x_i}}$ captures the input coupling effect among

This work is supported by the Spanish Training Program for Academic Staff (FPU17/02653), the European Research Council Advanced Grant OCONTSOLAR (SI-1838/24/2018), and the Spanish MINECO Project C3PO (DPI2017-86918-R).

P. Chanfreut, A. Sánchez-Amores, J. M. Maestre and E. F. Camacho are with the Department of Systems and Automation Engineering, University of Seville, Spain, e-mails: {pchanfreut,asamores,pepemaestre,efcamacho}@us.es.

subsystem i and its set of neighbors, i.e., $\mathcal{N}_i = \{j \in \mathcal{N} \setminus \{i\} : B_{ij} \neq \mathbf{0}\}$, where $\mathbf{0}$ is the null matrix of corresponding size. Also, matrices $A_{ii} \in \mathbb{R}^{n_{x_i} \times n_{x_i}}$ and $B_{ij} \in \mathbb{R}^{n_{x_i} \times n_{u_j}}$ are the state transition and the input-to-state matrices for all $i, j \in \mathcal{N}$. Hereafter, we consider that vectors u_i concatenate a subset of the overall inputs, i.e., $u_i = [v_m]_{m \in \mathcal{V}_i}$, with v_m being the input m of the global system and $\mathcal{V}_i$ the subset of inputs associated with subsystem i. For example, for a system of two agents with four inputs and $\mathcal{V}_1 = \{1, 2\}$ and $\mathcal{V}_2 = \{3, 4\}$, one would have $u_1 = [v_1,\ v_2]^{\mathrm{T}}$ and $u_2 = [v_3,\ v_4]^{\mathrm{T}}$. Also, it is assumed that the N subsystems are individually governed by a set of local MPC controllers that can exchange information to determine their control actions.

By defining the overall state and input vectors as the succession of every x_i and u_i, i.e., $x_{\mathcal{N}} = [x_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n_x}$ and $u_{\mathcal{N}} = [u_i]_{i \in \mathcal{N}} \in \mathbb{R}^{n_u}$, the overall system dynamics can be modeled as

$$x_{\mathcal{N}}^{+} = A_{\mathcal{N}} x_{\mathcal{N}} + B_{\mathcal{N}} u_{\mathcal{N}}, \tag{2}$$

where $A_{\mathcal{N}}$ and $B_{\mathcal{N}}$ aggregate A_{ii} and B_{ij} for all $i, j \in \mathcal{N}$ into single matrices. Note that all subsystem interactions defined by w_i are implicitly considered by matrix $B_{\mathcal{N}}$ in (2).

A. Centralized control problem

At each time instant k, the N local agents seek to optimize in a distributed manner the following global MPC control problem:

$$\begin{aligned}
\min_{[U_i]_{i \in \mathcal{N}}} \quad & \sum_{n=0}^{N_{\mathrm{p}}-1} \sum_{i \in \mathcal{N}} \ell_i(n) \\
\text{s.t.} \quad & x_i(0) = x_i(k), && \text{(3a)} \\
& x_i(n+1) = A_{ii} x_i(n) + B_{ii} u_i(n) + w_i(n), && \text{(3b)} \\
& G_{x_i} x_i(n+1) \leq g_{x_i}, && \text{(3c)} \\
& G_{v_m} v_m(n) \leq g_{v_m},\ \forall m \in [1, n_u], && \text{(3d)} \\
& u_i(n) = [v_m(n)]_{m \in \mathcal{V}_i}, && \text{(3e)} \\
& \forall i \in \mathcal{N}, && \text{(3f)} \\
& n = 0, ..., N_{\mathrm{p}} - 1, && \text{(3g)}
\end{aligned}$$

where the time index n between brackets indicates the step time of the prediction horizon, e.g., $x_i(n)$ is the prediction on subsystem i state for instant $k + n$. The stage cost of subsystem $\ell_i(\cdot)$ is defined as

$$\ell_i(n) = x_i^{\mathrm{T}}(n+1) Q_i x_i(n+1) + u_i^{\mathrm{T}}(n) R_i u_i(n). \tag{4}$$

for all $i \in \mathcal{N}$, with $Q_i \geq 0$ and $R_i > 0$ being weighting matrices. Also, G_{x_i}, g_{x_i}, G_{v_m} and g_{v_m} are matrices and vectors defining the constraints on subsystem i states and inputs, and U_i is its sequence of actions for a time horizon of N_{p} steps, i.e., $U_i = [u_i(0)^{\mathrm{T}}, u_i(1)^{\mathrm{T}}, ..., u_i(N_{\mathrm{p}} - 1)^{\mathrm{T}}]^{\mathrm{T}}$.

III. Dual Decomposition with Neural-Network based Warm Start

A. Dual decomposition

In what follows, we consider the dual decomposition algorithm described in [12](Procedure 1), which allows to compute the solution of global problem (3) in a distributed way. In particular, we assume that each agent $i \in \mathcal{N}$ optimizes not only its vector u_i, but also neighboring system inputs with impact on its dynamics, leading to a *shared* optimization of variables connecting coupled subsystems. Convergence to the centralized solution is achieved using an iterative negotiation procedure in which Lagrange multipliers are used to coordinate shared variables. Hereon, let $\mathcal{V}_i^{\mathrm{a}}$ denote the *augmented* set of inputs that affect the dynamics of agents i, i.e., $\mathcal{V}_i^{\mathrm{a}} = \mathcal{V}_i \cup \{m \in [1, n_u] \ :\ B_{ij}^m \neq 0,\ j \in \mathcal{N}_i\}$ where B_{ij}^m is the subblock of matrix B_{ij} mapping input v_m into state x_i. Notice that although agent $j \in \mathcal{N}_i$, some inputs in vector u_j may not influence subsystem i dynamics, hence $\mathcal{V}_i^{\mathrm{a}} \subseteq \cup_{j \in \{i\} \cup \mathcal{N}_i} \mathcal{V}_j$. Also, let us define $\mathbf{u}_i = [v_m]_{m \in \mathcal{V}_i^{\mathrm{a}}}$ and

$$\boldsymbol{\ell}_i(n) = x_i^{\mathrm{T}}(n+1) Q_i x_i(n+1) + \mathbf{u}_i^{\mathrm{T}}(n) \mathbf{R}_i \mathbf{u}_i(n), \tag{5}$$

where matrix $\mathbf{R}_i$ is built to guarantee that the sum of all local indexes leads to the centralized objective in (3), i.e., $\sum_{i \in \mathcal{N}} \boldsymbol{\ell}_i(n) = \sum_{i \in \mathcal{N}} \ell_i(n)$. Considering this, the local objective function of agent i becomes $\sum_{n=0}^{N_{\mathrm{p}}-1} \boldsymbol{\ell}_i(n)$.

It is well known that dual decomposition enforces constraint satisfaction by incorporating Lagrange multipliers into local objective functions. In this regard, let $v_m^i = M_i^m \mathbf{u}_i$ contain the components of $\mathbf{u}_i$ associated with input v_m, where M_i^m is a mapping matrix defined accordingly. Then, we consider equality constraints

$$v_m^i - v_m^j = 0,\ \text{with } m \in \mathcal{V}_i^{\mathrm{a}}, m \in \mathcal{V}_j^{\mathrm{a}}. \tag{6}$$

Notice that (6) implies that the solution of agent i for a certain input v_m must be equal to the solution of other agent j for the same input, with v_m being a shared variable for agents i and j. Additionally, let λ_m be the Lagrange multiplier associated with constraint (6). Then, problem (3) can be rewritten as

$$\max_{[\Lambda_m]_{m \in \mathcal{S}}} \min_{[\mathbf{U}_i]_{i \in \mathcal{N}}} \mathbf{J}(x_{\mathcal{N}}(k), [\Lambda_m]_{t=1}^{n_u}, [\mathbf{U}_i]_{i \in \mathcal{N}}) \tag{7}$$

with

$$\begin{aligned}
\mathbf{J}(x_{\mathcal{N}}(k), [\Lambda_m]_{t=1}^{n_u}, [\mathbf{U}_i]_{i \in \mathcal{N}}) = & \sum_{n=0}^{N_{\mathrm{p}}-1} \sum_{i \in \mathcal{N}} \boldsymbol{\ell}_i(n) + \\
& \sum_{n=0}^{N_{\mathrm{p}}-1} \sum_{i,j \in \mathcal{N}} \sum_{\substack{m \in \mathcal{V}_i^{\mathrm{a}}, \\ m \in \mathcal{V}_j^{\mathrm{a}}}} \lambda_m(n) (M_i^m \mathbf{u}_i(n) - M_j^m \mathbf{u}_j(n)).
\end{aligned} \tag{8}$$

subject to (3a) to (3g). The sequences of variables $\mathbf{u}_i$ and Lagrange multipliers λ_m along the time horizon are respectively represented by $\mathbf{U}_i = [\mathbf{u}_i(0)^{\mathrm{T}}\ \mathbf{u}_i(1)^{\mathrm{T}}\ \dots\ \mathbf{u}_i(N_{\mathrm{p}-1} - 1)^{\mathrm{T}}]^{\mathrm{T}}$ and $\Lambda_m = [\lambda_m(0)\ \lambda_m(1)\ \dots\ \lambda_m(N_{\mathrm{p}-1} - 1)]$. The objective function in (7) can be *broken up* into N functions depending just on variables x_i, $\mathbf{U}_i$ and the corresponding Lagrange multipliers, which allows us to distribute the global MPC problem between the N local controllers. In particular, each iteration p, agents $i \in \mathcal{N}$ first solve the following MPC optimization problem to find the optimal sequence $\mathbf{U}_i^p$ for some fixed prices $\Lambda_m^p = [\lambda_m^p(n)]_{n=0}^{N_{\mathrm{p}}-1}$:

$$\mathbf{U}_i^p = \arg\min_{\mathbf{U}_i} \quad \mathbf{J}_i(x_i(k), \mathbf{U}_i) = \sum_{n=0}^{N_\mathrm{p}-1} \boldsymbol{\ell}_i(n) \pm \sum_{n=0}^{N_\mathrm{p}-1} \sum_{\substack{m\in\mathcal{S},\\ m\in\mathcal{V}_i^\mathrm{a}}} \lambda_m^p(n) M_i^m \mathbf{u}_i(n)$$

$$\text{s.t. } x_i(0) = x_i(k), \tag{9a}$$

$$x_i(n+1) = A_{ii}x_i(n) + B_{ii}u_i(n) + \sum_{j\in\mathcal{N}_i} B_{ij}u_j(n), \tag{9b}$$

$$G_{x_i}x_i(n+1) \le g_{x_i}, \tag{9c}$$

$$G_{v_m}v_m(n) \le g_{v_m},\ \forall m \in \mathcal{V}_i^\mathrm{a}, \tag{9d}$$

$$u_i(n) = [v_m(n)]_{m\in\mathcal{V}_i}, \tag{9e}$$

$$\mathbf{u}_i(n) = [v_m(n)]_{m\in\mathcal{V}_i^\mathrm{a}}, \tag{9f}$$

$$n = 0, ..., N_\mathrm{p} - 1, \tag{9g}$$

where the sign $\pm$ is set according to (7), i.e., $\sum_{i\in\mathcal{N}} \mathbf{J}_i(\cdot) = \mathbf{J}(\cdot)$, and set $\mathcal{S}$ groups all shared inputs, i.e.,

$$\mathcal{S} = \{m \in [1, n_u]\ : m \in \mathcal{V}_i^\mathrm{a}, m \in \mathcal{V}_j^\mathrm{a}, \text{ with } i, j \in \mathcal{N},\ i \neq j\}.$$

Subsequently, the agents update the values of Λ_m^{p+1} according to the solutions obtained for $\mathbf{U}_i$, i.e.,

$$\Lambda_m^{p+1} = \Lambda_m^p + \gamma(\mathbf{U}_i^p - \mathbf{U}_j^p), \text{ with } m \in \mathcal{V}_i^\mathrm{a}, m \in \mathcal{V}_j^\mathrm{a}, \tag{10}$$

and $\gamma > 0$ being the step size. The process is repeated iteratively until convergence to the optimal solution is attained. Note that the nature of the algorithm is cooperative since all agents coordinate their decisions to optimize the plant-wide objective function in (3).

B. Warm start based on Neural Networks

The objective of this paper is to exploit the predictive ability of neural networks to provide an optimal warm start for Lagrange multipliers, say $\Lambda^* = [\Lambda_m^*]_{m\in\mathcal{S}}$, for any system state $x_\mathcal{N}$. In case of perfect prediction, local controllers would just need to solve problem (9) once per time step, leading to a notable reduction of cooperation efforts. Moreover, only local states would need to be exchanged to compute the corresponding Λ^*, thus reducing also the overall communication load.

To create the neural network model, a data base $\mathcal{DB}$ gathering a set of D global state points and their associated optimal multipliers Λ_m^* is generated offline, i.e.,

$$\mathcal{DB} = \{x_\mathcal{N}^d,\ [\Lambda_m^{*d}]_{m\in\mathcal{S}}\}_{d\in[1,D]}. \tag{11}$$

where superscript d is used to index each of the data base entries. To this end, the dual decomposition algorithm in [12](Procedure 1) is repeatedly simulated until convergence is attained for the D states in $\mathcal{DB}$, and the optimal multipliers are correspondingly stored. From (11), it is possible to derive a model

$$\Lambda_m^* = f_m(x_\mathcal{N}) \tag{12}$$

for any $m \in \mathcal{S}$, by training a neural network with features $\mathcal{DB}_x = \{x_\mathcal{N}^d\}_{d\in[1,D]}$ and target $\mathcal{DB}_{\Lambda_m} = \{\Lambda_m^{*d}\}_{d\in[1,D]}$. Likewise, by simply aggregating (12) for all m, i.e.,

$$\underbrace{\begin{bmatrix} \Lambda_1^* \\ \Lambda_2^* \\ \vdots \end{bmatrix}}_{\Lambda^*} = \underbrace{\begin{bmatrix} f_1(x_\mathcal{N}) \\ f_2(x_\mathcal{N}) \\ \vdots \end{bmatrix}}_{f(x_\mathcal{N})} \tag{13}$$

one can derive a model $\Lambda^* = f(x_\mathcal{N})$ that directly *predicts* the optimal values of all Lagrangian prices. Alternatively, it is possible to train directly a single neural network considering Λ^* as target, with the drawback of dealing with $N_\mathrm{p}|\mathcal{S}|$ target variables, i.e., the number of shared variables $|\mathcal{S}|$ multiplied by the prediction horizon N_p.

Considering the above, once the training process is completed, models (12) are introduced in the distributed control scheme for its online use. In particular, Algorithm 1 shows the pseudo-code with the steps followed by the local agents at each time step k.

Algorithm 1 Control Scheme

At each sample time k, starting with $p = 0$, the system proceed as follows:

1: **if** $\|\Delta x_\mathcal{N}\| \le \epsilon$ **then**
2: The agents use as initial guess the final value of the multipliers obtained at instant $k-1$.
3: **else**
4: The agents share their state and compute initial guess $\Lambda_m^0 = f_m(x_\mathcal{N})$ for all shared inputs $m \in \mathcal{S}$.
5: **end if**
6: All agents $i \in \mathcal{N}$ solve problem (9) and find optimal sequences $\mathbf{U}_i^p$.
7: Update Lagrange prices, i.e., $\Lambda_m^{p+1} = \Lambda_m^p + \gamma(\mathbf{U}_i^p - \mathbf{U}_j^p)$ for all $m \in \mathcal{S}$.
8: Set $p \leftarrow p+1$ and go to Step 6 until convergence is attained or a maximum number of iterations $\bar{p}$ is reached.
9: Each agent $i \in \mathcal{N}$ implements the first component of the agreed input sequences and updates x_i according to (1).

Note that if the change in $x_\mathcal{N}$ between consecutive time instants is not significant, e.g., when the system reaches the steady state, then a warm start based on the previous step may also provide a suitable initial guess for the Lagrange prices. That is, if at instant k, $\|x_\mathcal{N} - x_\mathcal{N}^-\| \le \epsilon$, then, instead of using model (13), one may choose $\Lambda^0 = \Lambda^{*-}$, where $x_\mathcal{N}^-$ and Λ^{*-} are respectively the system state and the optimal Lagrangian prices of the previous instant $k-1$, and ϵ is the threshold.

Also, from (11), it is possible to assess the impact of each subsystem state on the Lagrange multipliers so as to improve models (12). For example, if a multiplier Λ_m^* is almost independent from some subsystem states, then a reduced model $\hat{\Lambda}_m^* = \hat{f}_m(x_{\mathcal{C}_m})$ can be generated, where $x_{\mathcal{C}_m}$ aggregates the subsystem states with a notable effect on price Λ_m^*, with $\mathcal{C}_m \subseteq \mathcal{N}$. The latter allows for a distributed initialization of the Lagrange multipliers, i.e., if Λ_m appears in agents i and j control problems, then i and j just need to know the states in $x_{\mathcal{C}_m}$ to find Λ_m^0. By using the same

reasoning, if $\mathcal{C}_m \subset \mathcal{N}$ for all $m \in \mathcal{S}$, then the all-to-all data exchange may not be required for the prices initialization.

IV. Simulation results

The proposed algorithm has been simulated on the 16 tanks shown in Figure 1. We assume that each of the tanks represents a subsystem, hence, $\mathcal{N} = \{1, \dots, 16\}$, with dynamics modeled as

$$x_i^+ = x_i + \frac{T_s}{A_i} \sum_{m \in \mathcal{I}_i} v_m - \frac{T_s}{A_i} \sum_{m \in \mathcal{O}_i} v_m, \qquad (14)$$

where x_i is the relative water level to the operating point in tank i, v_m is the flow through pipe m, and $\mathcal{I}_i$ and $\mathcal{O}_i$ contain respectively the set of inlet and outlet pipes of tank i. Also, $A_i = 2\pi r_i^2$ is the cross section of tank i, where r_i denotes the radius, and T_s represents the sample time.

Hereafter, we use Figure 1, e.g., $\mathcal{I}_1 = \{1\}$ and $\mathcal{O}_1 = \{2, 3\}$. Note that model (14) can be rewritten as (1) for a proper definition of vector u_i. In this regard, we consider Table I, assuming that each agent $i \in \mathcal{N}$ can manipulate the flow through its outlet pipes and pump water into its neighbouring tanks. On the other hand, the inflows of each tank i are seen as coupling disturbances for all $i > 2$. Note that agent 1 also manipulates the inflow through pipe 1, which is an external flow source. Vectors $\mathbf{u}_i$, which contain the control actions considered in the local optimization problems, are defined accordingly as $\mathbf{u}_i = [v_m]_{m \in \mathcal{I}_i \cup \mathcal{O}_i}$ for any $i \in \mathcal{N}$. Therefore, if tanks i and j are connected by pipe m, then inputs sequences $\mathbf{U}_i = [\mathbf{u}_i(n)]_{n=0}^{N_\mathrm{p}-1}$ and $\mathbf{U}_j = [\mathbf{u}_j]_{n=0}^{N_\mathrm{p}-1}$ should satisfy $v_m^i = v_m^j$ on the flow through a given pipe m through the entire prediction horizon. This generates a set of 19 equality constraints that can be formulated as $M_i^m \mathbf{u}_i^m(n) - M_j^m \mathbf{u}_j^m(n) = 0$, with $n = 0, ..., N_\mathrm{p} - 1$, and that are distributed using dual decomposition in the corresponding local MPC control problems.

The goal is to regulate the 16 tanks towards the operating point while satisfying the following constraints:

$$|x_i| \leq 1 \quad \forall i \in \mathcal{N}, \qquad (15a)$$

$$0 \leq v_m \leq 0.5 \quad \forall m \in \mathcal{S}, \text{ and} \qquad (15b)$$

$$0 \leq v_1, v_{20} \leq 1. \qquad (15c)$$

To this end, stage cost (4) with weighting matrices $Q_i = 1$ and $R_i = 0.2 \cdot \mathbf{I}_{n_{u_i}}$ for all $i \in \mathcal{N}$ is used, where $\mathbf{I}_{n_{u_i}}$ is the identity matrix of dimensions $n_{u_i} \times n_{u_i}$. Other parameters used in the simulations are $N_\mathrm{p} = 3$, $T_\mathrm{s} = 0.2$ and $r_i = 0.5$.

Neural networks have been generated using Matlab® `Deep Learning Toolbox`, specifically the Levenberg-Marquardt algorithm with 25 hidden layer neurons, in a 1.8 GHz quad-core Intel® Core™ i7/8 GB RAM computer. To this end, a data base $\mathcal{DB}$ with 3500 states and optimal Lagrange multipliers was previously created. The simulation results obtained are summarized below.

Figure 2 illustrates the performance of Algorithm 1 when the initial state is $x_\mathcal{N} = [\ 0.6, -0.7,\ 0.5,\ 0.6, -0.3, -0.5, -0.9,\ 0.8, -0.8,\ 0.7, -0.8, -0.6,\ 0.5,\ 0.7, -0.9,\ 0.8]^\mathrm{T}$, and a maximum number of iterations $\bar{p} = 1$ is considered.

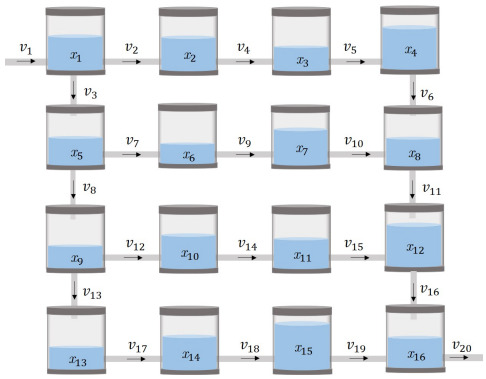


Fig. 1. Scheme of the 16 tanks system.

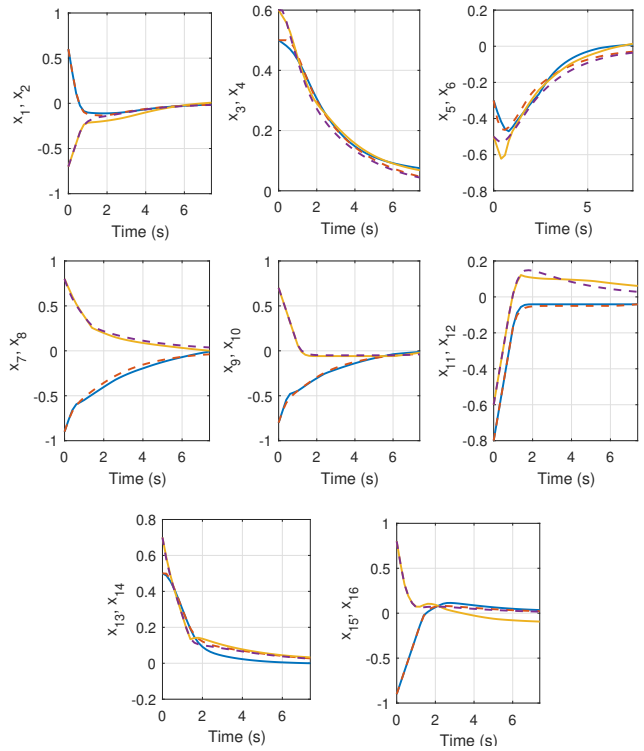
TABLE I

Definition of the subsystems input vectors u_i and optimization variables $\mathbf{u}_i$ for each agent i.

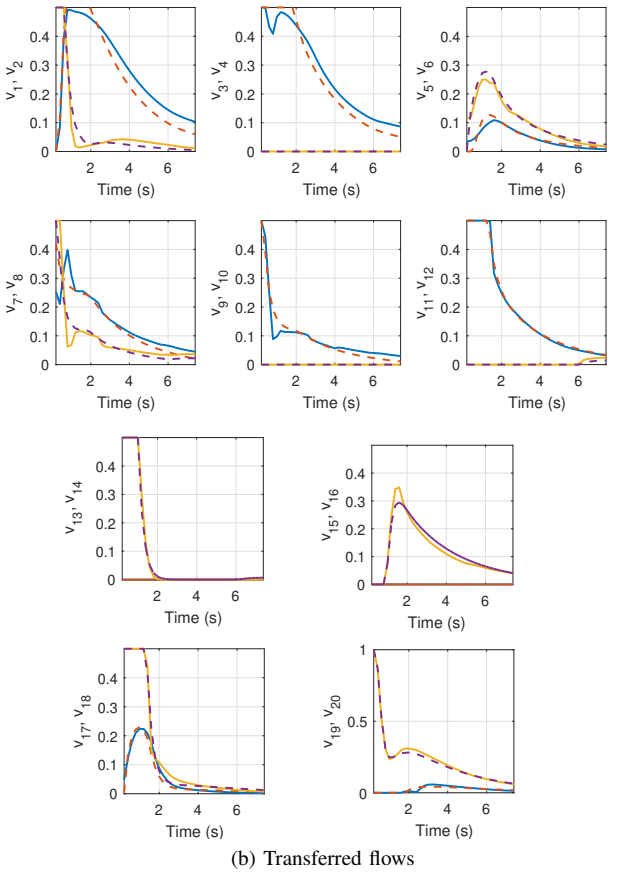
i	u_i^T	$\mathbf{u}_i^\mathrm{T}$	i	u_i^T	$\mathbf{u}_i^\mathrm{T}$
1	$[v_1, v_2, v_3]$	$[v_1, v_2, v_3]$	9	$[v_{12}, v_{13}]$	$[v_8, v_{12}, v_{13}]$
2	$[v_4]$	$[v_2, v_4]$	10	$[v_{14}]$	$[v_{12}, v_{14}]$
3	$[v_5]$	$[v_4, v_5]$	11	$[v_{15}]$	$[v_{14}, v_{15}]$
4	$[v_6]$	$[v_5, v_6]$	12	$[v_{16}]$	$[v_{11}, v_{15}, v_{16}]$
5	$[v_7, v_8]$	$[v_3, v_7, v_8]$	13	$[v_{17}]$	$[v_{13}, v_{17}]$
6	$[v_9]$	$[v_7, v_9]$	14	$[v_{18}]$	$[v_{17}, v_{18}]$
7	$[v_{10}]$	$[v_9, v_{10}]$	15	$[v_{19}]$	$[v_{18}, v_{19}]$
8	$[v_{11}]$	$[v_6, v_{10}, v_{11}]$	16	$[v_{20}]$	$[v_{16}, v_{19}, v_{20}]$

Also, parameter ϵ has been set to 0 in this case, i.e., at each time instant, the agents implement directly the first component of the solutions obtained with $\Lambda^0 = f(x_\mathcal{N})$, according to Step 4 in Algorithm 1. Note that if $\epsilon = 0$, the condition in Step 1, i.e., $\|\Delta x_\mathcal{N}\| \leq 0$, is not satisfied unless there are no state changes between consecutive instants. In particular, solid lines represent subsystems' state and inputs evolution when Algorithm 1 is applied and dashed lines show the results corresponding to centralized MPC. As can be seen, state and inputs trajectories follow closely the centralized solution despite not allowing the agents to iterate. However, if the agents negotiation is stopped before convergence, state constraint (15a) may be violated. To avoid any loss of feasibility, state restrictions have been introduced as soft constraints in each subsystem local problem.

Table II provides the overall cumulative cost for different values of parameter $\bar{p}$ when the step γ is set to $5E-2$ and $\epsilon = 0$. This performance index has been computed as the cumulative sum of state cost (4), i.e., $\sum_{k=1}^{T} \sum_{i \in \mathcal{N}} \ell_i(k)$, being T is the simulation length. Table II shows that even using $\bar{p} = 1$, the loss of performance is just a 3.33% with respect to centralized operation, which indicates that the Lagrange multipliers provided by the neural networks are close to be optimal. Also, for this value of γ, a few iterations allow for a further improvement, e.g., for $\bar{p} = 10$, performance loss decreases to 0.3%. Also, when no limitation in $\bar{p}$ is considered and $\|\mathbf{U}^{p-1} - \mathbf{U}^p\|_2 \leq 5E-4$ is used as convergence condition, being $\mathbf{U}$ the overall input sequence, the system performance practically matches that of the centralized results (29.9232).



(a) Subsystems state



(b) Transferred flows

Fig. 2. Evolution the subsystems state and transferred flows between tanks. The solid lines shows the result when the neural network models are incorporated into the control scheme and a maximum number of $\bar{p} = 1$ iterations is allowed, and the dashed lines represent the centralized MPC solution.

TABLE II

COMPARISON OF THE OVERALL CUMULATIVE PERFORMANCE COST FOR DIFFERENT VALUES OF $\bar{p}$.

	Cumulative cost
$\bar{p} = 1$	30.9123
$\bar{p} = 5$	30.2270
$\bar{p} = 10$	30.0063
Centralized MPC	29.9156

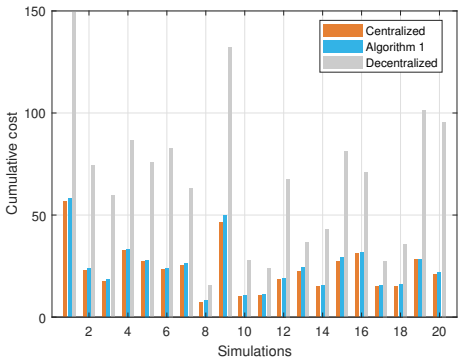


Fig. 3. Cumulative performance costs in 20 simulations starting from different random states. The cost of Algorithm 1 with $\bar{p} = 1$ and $\epsilon = 0$ is compared with the centralized and decentralized MPC approaches.

For a better performance assessment, 20 random initial system states satisfying $|x_i| \leq 1$ have been generated, and the performance of Algorithm 1 (with $\bar{p} = 1$ and $\epsilon = 0$) has been compared with centralized and decentralized MPC (see Figure 3). In the decentralized case, each agent i measures its state x_i and optimizes function $\sum_{n=0}^{N_p-1} \ell_i(n)$ in terms of the variables it can manipulate, i.e., the sequence of inputs u_i. In this respect, the predicted inflows from neighboring controllers are set to zero in the local optimizations, and state constraint (15a) is introduced as a soft constraint. Again, the neural network approach leads to costs that barely differ from the centralized outcome. However, when the decentralized approach is implemented, just those tanks starting with $x_i \geq 0$ regulate their state to the origin, but those with $x_i < 0$ cannot reach the setpoint unless some of the neighboring agents send water to them due to an excess in their own tanks (see Figure 4). This causes a notable increase of the costs, which, in turn, highlights the benefits of coordination.

Finally, Figure 5 compares the number of iterations required each time step for satisfying convergence condition $\|\mathbf{U}^{p-1} - \mathbf{U}^{p}\|_2 \leq 5E - 4$. The blue line shows the result when a warm start based on the previous time instant is always considered, and the red line shows the number of iterations obtained when using the neural network but switching to a previous-step based warm start when $\|\Delta x_{\mathcal{N}}\|_2 \leq 0.04$. Notice that the former approach shows a good performance when the system state reaches the steady state, but requires greater iterations during the first simulation steps, where the neural network allows to speed up the negotiation.

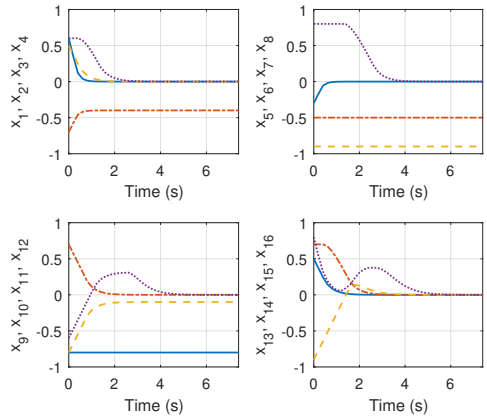


Fig. 4. Subsystem state when the system is controlled in a decentralized manner.

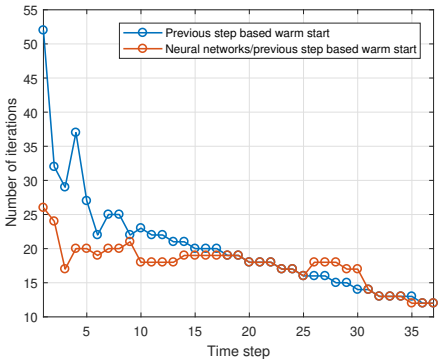


Fig. 5. Number of iterations per time instant obtained with a previous step based warm start and when the latter is combined with the neural network initialization model.

V. Conclusion

In this paper, we deal with a distributed system where dual decomposition is used to coordinate a set of local agents, with a neural network predicting the value of the Lagrange multipliers at each time step as a function of the overall system state. The results on a 16-tanks system show that these initial prices provide performance close to the centralized one. Moreover, at expense of higher offline computation costs derived from the neural network model generation, this approach can reduce the online computation and communication demands associated with the iterative negotiation procedure that the agents must perform to find the control actions.

Further research will study how this idea can be applied to more realistic systems where also disturbances come into play and analyze its scalability. Additionally, we will train the neural network with control actions as features rather than the overall state, that is, the Lagrange multipliers will be modeled as a function on the last optimal inputs, thus avoiding the need of sharing states. In this regard, future work should provide a sensitivity analysis to investigate the effect of different input parameters on the neural network output, and of the type and structure of the network itself. Finally, we plan to provide a suboptimality bound on the loss of performance caused by using directly the predicted Lagrangian prices for computing the implemented inputs.

References

[1] R. R. Negenborn and J. Maestre, "Distributed model predictive control: An overview and roadmap of future research opportunities," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 87–97, 2014.

[2] P. D. Christofides, R. Scattolini, D. M. de la Pena, and J. Liu, "Distributed model predictive control: A tutorial review and future research directions," *Computers & Chemical Engineering*, vol. 51, pp. 21–41, 2013.

[3] M. Razzanelli, E. Crisostomi, L. Pallottino, and G. Pannocchia, "Distributed model predictive control for energy management in a network of microgrids using the dual decomposition method," *Optimal Control Applications and Methods*, vol. 41, no. 1, pp. 25–41, 2019.

[4] A. N. Venkat, I. A. Hiskens, J. B. Rawlings, and S. J. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1192–1206, 2008.

[5] P.-D. Moroşan, R. Bourdais, D. Dumur, and J. Buisson, "Building temperature regulation using a distributed model predictive control," *Energy and Buildings*, vol. 42, no. 9, pp. 1445–1452, 2010.

[6] J. Alejandro, A. Arce, and C. Bordons, "Combined environmental and economic dispatch of smart grids using distributed model predictive control," *International Journal of Electrical Power & Energy Systems*, vol. 54, pp. 65–76, 2014.

[7] F. Garcia-Torres, C. Bordons, and M. A. Ridao, "Optimal economic schedule for a network of microgrids with hybrid energy storage system using distributed model predictive control," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 3, pp. 1919–1929, 2018.

[8] M. Moradzadeh, R. Boel, and L. Vandevelde, "Voltage coordination in multi-area power systems via distributed model predictive control," *IEEE Transactions on Power Systems*, vol. 28, no. 1, pp. 513–521, 2012.

[9] A. Bemporad and D. Barcelli, "Decentralized model predictive control," in *Networked Control Systems*. Springer, 2010, pp. 149–178.

[10] J. B. Rawlings and B. T. Stewart, "Coordinating multiple optimization-based controllers: New opportunities and challenges," *Journal of process control*, vol. 18, no. 9, pp. 839–845, 2008.

[11] A. N. Venkat, J. B. Rawlings, and S. J. Wright, "Stability and optimality of distributed model predictive control," in *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, 2005, pp. 6680–6685.

[12] F. Farokhi, I. Shames, and 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.

[13] P. Giselsson, M. D. Doan, T. Keviczky, B. De Schutter, and A. Rantzer, "Accelerated gradient methods and dual decomposition in distributed model predictive control," *Automatica*, vol. 49, no. 3, pp. 829–833, 2013.

[14] B. Hou, Y. Zheng, and S. Li, "A dual decomposition based dMPC for networked systems with varying topology," in *2019 Chinese Automation Congress (CAC)*. IEEE, 2019, pp. 4541–4546.

[15] P. Giselsson, "Improved dual decomposition for distributed model predictive control," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1203–1209, 2014.

[16] K. O. Temeng, P. D. Schnelle, and T. J. McAvoy, "Model predictive control of an industrial packed bed reactor using neural networks," *Journal of Process Control*, vol. 5, no. 1, pp. 19–27, 1995.

[17] S. Chen, Z. Wu, D. Rincon, and P. Christofides, "Machine learning-based distributed model predictive control of nonlinear processes," *American Institute of Chemical Engineers (AIChE) journal*, vol. 66, no. 11, 06 2020.

[18] P. Kittisupakorn, P. Thitiyasook, M. Hussain, and W. Daosud, "Neural network based model predictive control for a steel pickling process," *Journal of Process Control*, vol. 19, no. 4, pp. 579–590, 2009.

[19] B. M. Åkesson and H. T. Toivonen, "A neural network model predictive controller," *Journal of Process Control*, vol. 16, no. 9, pp. 937–946, 2006.

[20] S. Liu and J. Wang, "A simplified dual neural network for quadratic programming with its kwta application," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1500–1510, 2006.