ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Cloud-assisted Distributed Nonlinear Optimal Control for Dynamics over Graph

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

*Published Version:*

*Availability:*
This version is available at: https://hdl.handle.net/11585/678707 since: 2020-02-28

*Published:*

DOI: http://doi.org/10.1016/j.ifacol.2018.12.062

*Terms of use:*

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

19 April 2024

This is the post peer-review accepted manuscript of:

S. Spedicato and G. Notarstefano, “Cloud-assisted Distributed Nonlinear Optimal Control for Dynamics over Graph”, IFAC-PapersOnLine, Volume 51, Issue 23, 2018, pp 361-366.

The published version is available online at:

https://doi.org/10.1016/j.ifacol.2018.12.062

# Cloud-assisted Distributed Nonlinear Optimal Control for Dynamics over Graph ⋆

**Sara Spedicato** * **Giuseppe Notarstefano** *

* *Department of Engineering, Università del Salento, Via per Monteroni, 73100 Lecce, Italy (e-mail: name.lastname@unisalento.it).*

**Abstract:** Dynamics over graph are large-scale systems in which the dynamic coupling among subsystems is modeled by a graph. Examples arise in spatially distributed systems (as discretized PDEs), multi-agent control systems or social dynamics. In this paper, we propose a cloud-assisted distributed algorithm to solve optimal control problems for *nonlinear* dynamics over graph. Inspired by the centralized Hauser's projection operator approach for optimal control, our main contribution is the design of a descent method in which at each step agents of a network compute a local descent direction, and then obtain a new system trajectory through a distributed feedback controller. Such a controller, iteratively designed by a cloud, allows agents of the network to use only information from neighboring agents, thus resulting into a *distributed projection operator over graph*. The main advantages of our globally convergent algorithm are dynamic feasibility at each iteration and numerical robustness (thanks to the closed-loop updates) even for unstable dynamics. In order to show the effectiveness of our strategy, we present numerical computations on a discretized model of the Burgers' nonlinear partial differential equation.

*Keywords:* distributed optimal control, distributed optimization, distributed MPC, structured optimal control

## 1. INTRODUCTION

Several modern optimal control problems involve large-scale systems in which the state of each subsystem depends on the states of few other subsystems only. This feature arises in several, different contexts as, e.g., in social dynamics, e.g., Ravazzi et al. (2017); Gray et al. (2018), in cooperative control, e.g., Ahmed et al. (2016), or when discretizing partial differential equations, e.g., Ferrari-Trecate et al. (2006). Depending on the dynamic coupling among subsystems, an interaction graph can be associated to this "multi-agent" dynamics, thus obtaining a *dynamics over graph* (Ferrari-Trecate et al. (2006)). Optimal control problems for dynamics over graph are typically large-scale problems for which a centralized solution is impractical. Because of the large scale nature and the particular structure of the dynamics, which is a system of locally interconnected dynamical systems (one for each agent), the design of distributed algorithms is often desirable. While several distributed algorithms have been developed for optimal control of linear systems, since they give rise to convex problems, the design of distributed algorithms for nonconvex optimal control, involving nonlinear systems, is particularly challenging.

As for distributed optimal control algorithms, the works in Doan et al. (2011); Giselsson et al. (2013); Conte et al. (2016); Groß and Stursberg (2016) consider problems arising in Model Predictive Control schemes, where the cost is convex and the coupled dynamics linear, and thus deal with convex problems. The presence of nonlinear dynamics poses instead several challenges in the design of distributed algorithms. In Necoara et al. (2009); Dinh et al. (2013), sequential convex programming and dual decomposition methods are proposed to deal with (coupled) nonlinear dynamics over graph. In the proposed algorithms only a part of the computation is performed locally by each agent and the dynamic constraint is not satisfied at each iteration. Finally, in our algorithm we need to design a sparse feedback controller. The works in Massioni and Verhaegen (2009); Lin et al. (2011); Wu et al. (2016); Mårtensson and Rantzer (2012) address the design of a static state-feedback (satisfying prescribed optimality conditions) with a given sparsity for time-invariant linear systems.

The main contribution of this paper is the design of a cloud-assisted distributed algorithm for nonlinear optimal control of dynamics over graph. We design a globally convergent algorithm, where the descent direction and the update are executed in a distributed fashion. A central unit is only used to design a stabilizing feedback and update the step-size. The key features of our algorithm are: (i) dynamic feasibility at each iteration, i.e., at each iteration a state-input trajectory is available, and (ii) numerical robustness, i.e., trajectory instability for long time horizons is prevented, thanks to a closed loop update of system trajectories. Note that, typically, constrained optimization algorithms do not enjoy dynamic feasibility at each iteration. For example, standard Sequential quadratic programming does not ensure that all iterates satisfy the dynamics. That is the constraints due to the dynamics are satisfied only asymptotically. Our cloud-assisted distributed algorithm

⋆ This result is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638992 - OPT4SMART).

takes inspiration from and combines the advantages of two centralized algorithms with dynamic feasibility at each iteration: the open loop sequential algorithm in Bertsekas (1999) and the Projection Operator Newton Method for Trajectory Optimization (PRONTO) algorithm, Hauser (2002) (see also the discrete-time version developed in Bayer et al. (2013)). The open loop sequential algorithm is suited for distributed computation when applied to dynamics over graph, but it is not of practical use since it involves an open loop integration of the dynamics at each iteration. On the contrary, PRONTO algorithm is numerically robust but it is not suitable for distributed computation since both the computation of the descent direction (based on the solution of a Linear Quadratic optimal control problem) and the closed loop integration (nonlinear projection) require a centralized computation. In our algorithm the descent direction is a generic state-input curve, rather than an input only as in Bertsekas (1999), but is not required to satisfy the linearized dynamics as in Hauser (2002), so that it can be computed locally by the agents. Moreover, we design a strategy to implement the projection (i.e., the closed-loop integration of the dynamics) in a distributed way. We point out that the design strategy for the sparse controller extends to (linear) time-varying systems the one proposed in Lin et al. (2011) for time-invariant ones, and is thus an additional contribution. Theoretical guarantees for the controller design strategy still need further investigation.

The rest of the paper is organized as follows. In Section 2 we describe the problem set-up and provide a brief description of the open loop sequential method in Bertsekas (1999). In Section 3 we present our algorithm with the convergence result. The proof is omitted due to space limitations. In Section 4 we present numerical computations to show the effectiveness of our strategy.

## 2. PROBLEM SET-UP AND PRELIMINARIES

### *2.1 Problem set-up*

We consider a nonlinear discrete-time system of $N$ agents for which the dynamic coupling can be modeled by a fixed, connected and undirected graph $\mathcal{G} = \{\{1,\dots,N\},\mathcal{E}\}$. That is, $(i,j) \in \mathcal{E}$ if the dynamics of agent $i$ depends on the state of agent $j$. Note that the topology of the graph is not a design parameter, but it is imposed by the dynamics. Formally, let $\mathcal{N}_i$ be the set of neighbors of node $i$, i.e., $\mathcal{N}_i := \{j \in \{1,\dots,N\} | (i,j) \in \mathcal{E}\}$. Also, let $a_{ij}$ denote the element $i,j$ of the adjacency matrix associated to $\mathcal{G}$. We recall that $a_{ij} = 1$ if $(i,j) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise. Let us consider the nonlinear *dynamics over graph*

$$x_{i,t+1} = f_i(x_{\mathcal{N}_i,t}, u_{i,t}), \quad t \in \mathbb{N}, i \in \{1,\dots,N\} \tag{1}$$

where $x_{i,t} \in \mathbb{R}^n$ is the state of agent $i$ at time $t$, $x_{i,0}$ its (given) initial condition, $x_{\mathcal{N}_i,t} \in \mathbb{R}^{n|\mathcal{N}_i|}$, with $|\mathcal{N}_i|$ the cardinality of $\mathcal{N}_i$, is a stack vector of all $x_{j,t}$, $j \in \mathcal{N}_i$, $u_{i,t} \in \mathbb{R}^m$ is the input of agent $i$ at time $t$, and $f_i : \mathbb{R}^{n|\mathcal{N}_i|} \times \mathbb{R}^m \to \mathbb{R}^n$ is the local state function of agent $i$.

Let us define the sets $\mathbb{T}_{[0,M]} := \{0,\dots,M\}, \mathbb{T}_{[0,M-1]} := \{0,\dots,M-1\}, \mathbb{T}_{[1,M]} := \{1,\dots,M\}$, where $M \in \mathbb{N}$. A state-input *trajectory* of (1) on the horizon $M$ is a vector $\eta \in \mathbb{R}^{nN(M+1)+mNM}$ defined as $\eta := [x^\top u^\top]^\top$, where $x \in \mathbb{R}^{nN(M+1)}$ and $u \in \mathbb{R}^{mNM}$ are respectively the stacks of vectors $x_{i,t}$, $i \in \{1,\dots,N\}$, $t \in \mathbb{T}_{[0,M]}$, and $u_{i,t}$, $i \in \{1,\dots,N\}$, $t \in \mathbb{T}_{[0,M-1]}$, that satisfy equations (1). We instead denote a generic state-input *curve* on the horizon $M$ by $\xi \in \mathbb{R}^{nN(M+1)+mNM}$, $\xi := [\alpha^\top \mu^\top]^\top$, where $\alpha \in \mathbb{R}^{nN(M+1)}$ is the stack of vectors $\alpha_{i,t} \in \mathbb{R}^n$, $i \in \{1,\dots,N\}$, $t \in \mathbb{T}_{[0,M]}$, and $\mu \in \mathbb{R}^{mNM}$ is the stack of vectors $\mu_{i,t} \in \mathbb{R}^m$, $i \in \{1,\dots,N\}$, $t \in \mathbb{T}_{[0,M-1]}$.

We deal with the nonlinear optimal control problem

$$\begin{aligned} \min_{\substack{x_{i,0},\dots,x_{i,M} \\ u_{i,0},\dots,u_{i,M-1} \\ i\in\{1,\dots,N\}}} & \sum_{i=1}^{N} \Big( \sum_{t=0}^{M-1} \Big(\ell_i(x_{i,t},u_{i,t})\Big) + m_i(x_{i,M})\Big), \\ \text{subj. to } & x_{i,t+1} = f_i(x_{\mathcal{N}_i,t}, u_{i,t}), \quad t \in \mathbb{T}_{[0,M-1]}, \\ & i \in \{1,...,N\}, \end{aligned} \tag{2}$$

where $\ell_i : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, $m_i : \mathbb{R}^n \to \mathbb{R}$ and $f_i : \mathbb{R}^{n|\mathcal{N}_i|} \times \mathbb{R}^m \to \mathbb{R}^n$, for all $i \in \{1,\dots,N\}$, are continuously differentiable functions. In our scenario, each agent has computation and communication capabilities and only a partial knowledge of the problem. In particular, agent $i$ only knows its own $\ell_i(\cdot,\cdot), m_i(\cdot)$ and $f_i(\cdot,\cdot)$ and can communicate only with its neighbors $j \in \mathcal{N}_i$. When needed, it can be also supported by a cloud to perform additional operations. We aim to design a cloud-assisted distributed algorithm to solve problem (2). Since the problem is nonconvex we seek for points $(x^*,u^*,p^*)$ that satisfy the first-order necessary conditions for optimality of problem (2). In particular, each agent $i$ aims to locally compute its own $x^*_{i,t}$, $t \in \mathbb{T}_{[0,M]}$, $u^*_{i,t}$, $t \in \mathbb{T}_{[0,M-1]}$, $p^*_{i,t}$, $t \in \mathbb{T}_{[1,M]}$, of the vectors $x^*,u^*,p^*$, respectively, via local communication with the neighbors and the cloud.

### *2.2 Open loop sequential method*

The open loop sequential approach in Bertsekas (1999), Section 1.9, consists in re-writing optimal control problem (2) as an unconstrained problem with $u$ being the (only) optimization variable. Since equations (1) hold, the states $x_{i,t}$, for all $i$ and $t$, are written as a function of the input $u$, i.e., $x_{i,t} = \varphi_{i,t}(u)$, where $\varphi_{i,t}(\cdot)$, for all $i$ and $t$, are suitably defined functions. By using the latter equations, the dynamically constrained problem (2) becomes the unconstrained problem

$$\min_u \ \sum_{i=1}^{N} \Big( \sum_{t=0}^{M-1} \Big(\ell_i(\varphi_{i,t}(u),u_{i,t})\Big) + m_i(\varphi_{i,M}(u))\Big). \tag{3}$$

Defining

$$F(x,u) := \sum_{i=1}^{N} \Big( \sum_{t=0}^{M-1} \Big(\ell_i(x_{i,t},u_{i,t})\Big) + m_i(x_{i,M})\Big), \tag{4}$$

problem (3) can be equivalently written as

$$\min_u F(\varphi(u),u), \tag{5}$$

where $\varphi(u) \in \mathbb{R}^{nN(M+1)}$ is the stack of vectors $\varphi_{i,t}(u) \in \mathbb{R}^n$, $i \in \{1,\dots,N\}$, $t \in \mathbb{T}_{[0,M]}$.

Numerical solutions to the unconstrained problem (5) are computed via the steepest descent method in Algorithm 1, where, at each iteration $k$, a trajectory $x^{k+1}_{i,t}$, $u^{k+1}_{i,t}$, for all $i$ and $t$, is avaliable. The descent direction

$v^k_{i,t}$, for all $i$ and $t$, is computed via (6), where $a^k_{i,t} \in \mathbb{R}^n, b^k_{i,t} \in \mathbb{R}^m, A^k_{ji,t} \in \mathbb{R}^{n\times n}, B^k_{ii,t} \in \mathbb{R}^{n\times m}$ are defined as $a^k_{i,t} := \nabla_{x_{i,t}}\ell_i(x^k_{i,t}, u^k_{i,t}), b^k_{i,t} := \nabla_{u_{i,t}}\ell_i(x^k_{i,t}, u^k_{i,t}), A^k_{ji,t} := \nabla_{x_{i,t}} f_j(x^k_{\mathcal{N}_j,t}, u^k_{j,t})^\top, B^k_{ii,t} := \nabla_{u_{i,t}} f_i(x^k_{\mathcal{N}_i,t}, u^k_{i,t})^\top$, and, for a generic scalar function $g(\cdot,\cdot)$, $\nabla_y g(y,z)$ denotes the gradient with respect to $y$, while, for a vector function $g(\cdot,\cdot)$ with values in $\mathbb{R}^r$, we denote by $\nabla_y g(y,z) = [\nabla_y g_1(y,z) \dots \nabla_y g_r(y,z)]$ the gradient matrix with respect to $y$. We also use $\nabla_z g(y,z)$, similarly. The update is performed in (7) via the the open loop dynamics (1).

**Algorithm 1** Open loop sequential algorithm

**Require:** $x^0_{i,t}, u^0_{i,t}$, for all $i$ and $t$, trajectory of (1)
  **for** $k = 0, 1, 2 \dots$ **do**
    set $p^k_{i,M} = \nabla m_i(x^k_{i,M})$, for all $i \in \{1, \dots, N\}$
    **for** $t = M-1, \dots, 0$ **do**
      compute, for all $i \in \{1, \dots, N\}$,

$$\begin{aligned} v^k_{i,t} &= -\Big(b^k_{i,t} + B^{k\top}_{ii,t}\, p^k_{i,t+1}\Big) \\ p^k_{i,t} &= \sum_{j\in\mathcal{N}_i}\Big(A^{k\top}_{ji,t}\, p^k_{j,t+1}\Big) + a^k_{i,t} \end{aligned} \tag{6}$$

    compute step-size $\beta^k$ via the Armijo rule
    **for** $t = 0, \dots, M-1$ **do**
      compute, for all $i \in \{1, \dots, N\}$,

$$\begin{aligned} u^{k+1}_{i,t} &= u^k_{i,t} + \beta^k v^k_{i,t}, \\ x^{k+1}_{i,t+1} &= f_i(x^{k+1}_{\mathcal{N}_i,t}, u^{k+1}_{i,t}). \end{aligned} \tag{7}$$

Algorithm 1 enjoys dynamic feasibility at each iteration, (i.e., a trajectory satisfying the dynamics is available at each iteration). Also, the descent direction computation and the update can be performed in a distributed fashion. Nevertheless, the algorithm suffers of numerical instability, because of the open loop update. This is a well-known drawback in the literature that we aim to overcome with our algorithm presented in Section 3.

## 3. CLOUD-ASSISTED DISTRIBUTED OPTIMAL CONTROL ALGORITHM

In this section we present our cloud-assisted distributed algorithm to solve the nonlinear optimal control problem (2). We start by recalling the notion of Hauser's projection operator for dynamical systems, Hauser (2002), applied to our discrete-time set-up, and then introduce our distributed projection operator over graph.

### *3.1 Distributed projection operator over graph*

Let $\xi = [\alpha^\top \mu^\top]^\top$ be a curve lying in a neighborhood of a trajectory $\tilde\eta = [\tilde x^\top \ \tilde u^\top]^\top$ of the nonlinear system (1). Hauser's projection operator $\mathcal{P} : \xi \to \eta$, mapping the curve $\xi$ into a trajectory $\eta = [x^\top \ u^\top]^\top$ of (1), is defined as the feedback system

$$\begin{aligned} x_{i,t+1} &= f_i(x_{\mathcal{N}_i,t}, u_{i,t}), \\ u_{i,t} &= \mu_{i,t} + \sum_{j=1}^N K_{ij,t}\Big(\alpha_{j,t} - x_{j,t}\Big), \\ \alpha_{i,0} &= x_{i,0}, \end{aligned} \tag{8}$$

for all $i \in \{1, ..., N\}$ and $t \in \mathbb{T}_{[0,M-1]}$, where $K_{ij,t} \in \mathbb{R}^{m\times n}$ is the block $i,j$ of a controller matrix $K_t \in \mathbb{R}^{mN\times nN}$. The main idea is that, in order to make the optimization algorithm numerically robust, the controller matrix needs to have some stabilizability-like property, namely it has to exponentially stabilize the trajectory $\tilde x_{i,t},\ t \in \mathbb{T}_{[0,M]},\ \tilde u_{i,t},\ t \in \mathbb{T}_{[0,M-1]},\ i \in \{1, \dots, N\}$, as $M \to \infty$. While a trajectory of the open loop dynamics (1) can be computed in a distributed way, such a distributed computation is not possible, in general, for trajectories of the closed loop dynamics (8) unless a particular structure is imposed on $K_t,\ t \in \mathbb{T}_{[0,M-1]}$.

We thus define a *distributed projection operator over graph* as the feedback system (8), for all $i \in \{1, ..., N\}$ and $t \in \mathbb{T}_{[0,M-1]}$, where $K_t,\ t \in \mathbb{T}_{[0,M-1]}$, satisfies both a stabilizability-like property and the sparsity condition $K_{ij,t} = 0$ if $j \notin \mathcal{N}_i$, for all $i \in \{1, \dots, N\}$ and $t \in \mathbb{N}$. This will result into an input depending only on neighboring agents, see equations (13) in our algorithm. The design of a controller $K_t,\ t \in \mathbb{T}_{[0,M-1]}$, satisfying the above conditions is nontrivial. In Section 3.3 we propose a methodology.

### *3.2 Distributed Hauser-Projection Descent for Trajectory Optimization*

In this subsection, we present our cloud-assisted distributed algorithm to solve problem (2). The algorithm can be interpreted as a closed-loop version of the open loop sequential method presented in Section 2. In the open loop sequential method, the state trajectories are expressed as a function of the input $u$ via the open loop dynamics (1) thus obtaining the unconstrained problem (3). Therefore, in Algorithm 1 the descent direction consists of variations $v_{i,t}$, for all $i$ and $t$, of the input $u_{i,t}$, and the trajectory update is performed on the open loop dynamics. Inspired by Hauser (2002), we use the distributed projection operator over graph to express both states and inputs of trajectories as functions of state-input curves. A state-input trajectory $x_{i,t}, u_{i,t}$, for all $i$ and $t$, satisfying equations (8), can be written, in fact, as a function of the curve $\xi$ as $x_{i,t} = \phi_{i,t}(\xi), u_{i,t} = \gamma_{i,t}(\xi)$, with suitably defined $\phi_{i,t}(\cdot)$ and $\gamma_{i,t}(\cdot)$, for all $i$ and $t$. Thus, we can write the constrained problem (2) as the unconstrained problem

$$\min_\xi \ \sum_{i=1}^N \Big( \sum_{t=0}^{M-1} \Big(\ell_i(\phi_{i,t}(\xi), \gamma_{i,t}(\xi))\Big) + m_i(\phi_{i,M}(\xi))\Big), \tag{9}$$

where the only optimization variable is the state-input curve $\xi$. Problem (9) can be equivalently written as

$$\min_\xi \ \ F(\phi(\xi), \gamma(\xi)), \tag{10}$$

where $\phi(\xi) \in \mathbb{R}^{nN(M+1)}$ and $\gamma(\xi) \in \mathbb{R}^{mNM}$ are, respectively, the stacks of $\phi_{i,t}(\xi) \in \mathbb{R}^n,\ i \in \{1, \dots, N\},\ t \in \mathbb{T}_{[0,M]},\ \gamma_{i,t}(\xi) \in \mathbb{R}^m,\ i \in \{1, \dots, N\},\ t \in \mathbb{T}_{[0,M-1]}$ and $F(\cdot,\cdot)$ is defined in (4). In our algorithm, the descent direction consists of variations $z_{i,t}$ and $v_{i,t}$ of both states and inputs of the curve. Moreover, due to the presence of the feedback, $z_{i,t}$ and $v_{i,t}$ depend on $K_{ji,t},\ j \in \mathcal{N}_i$, and the trajectory update is performed in closed loop.

We now describe our Distributed Hauser-Projection Descent for Trajectory Optimization (Algorithm 2) from the perspective of agent $i$. At each iteration $k$, agent $i$ performs both cloud-assisted and distributed steps in order

to compute its $x_{i,t}^{k+1}, u_{i,t}^{k+1}$, for all $t$, of a system trajectory. *Send2Cloud*$(\cdot)$ and *ReceiveFromCloud*$(\cdot)$ are used to indicate messages sent to the cloud and received from the cloud, respectively. Agent $i$ performs the following steps: (i) communicates with the cloud to get $K_{ij,t}^k, K_{ji,t}^k, j \in \mathcal{N}_i$, for all $t$ (defining a distributed projection operator in the neighborhood of the current trajectory iterate), (ii) computes in a distributed way a descent direction $z_{i,t}^k, v_{i,t}^k$, for all $t$, via (11) and (12), where $A_{ji,t}^{cl,k} := A_{ji,t}^k - B_{jj,t}^k K_{ji,t}^k$, (iii) communicates with the cloud to get the step-size $\beta^k$ (computed via the Armijo rule), (iv) performs a distributed projection to get $x_{i,t}^{k+1}, u_{i,t}^{k+1}$, for all $t$.

*Remark 1.* The cloud could be used intermittently if the same matrix $K_t$, $t \in \mathbb{T}_{[0,M-1]}$, stabilizes different trajectories and a constant step-size is used. Moreover, differently from Bertsekas (1999) and Hauser (2002), the descent direction of our algorithm is a generic state-input curve that is not required to satisfy any linearized dynamics. □

As a comparison with Algorithm 1, note that, by setting $K_{ij,t}^k = 0$ and $K_{ji,t}^k = 0$, for all $j$ and $t$, equations (11) and (12) become equations (6), while equations (13), where $\mu_{i,t}^{k+1} = u_{i,t}^k + \beta^k v_{i,t}^k$, become equations (7). Finally, note that dynamic feasibility is guaranteed at each iteration. Thus, the algorithm can be stopped at any iteration with a twofold guarantee for the agents, namely they will have: a trajectory of (1) and a distributed controller to track it.

**Algorithm 2** Distributed Hauser-Projection Descent for Trajectory Optimization

**Require:** $x_{j,t}^0, u_{j,t}^0$, for all $t$, for $j \in \mathcal{N}_i$, such that $\eta^0 = [x^{0\top} u^{0\top}]^\top$ is a trajectory of (1)
**for** $k = 0, 1, 2 \dots$ **do**
*Send2Cloud*$(x_{i,t}^k, t \in \mathbb{T}_{[0,M]}, u_{i,t}^k, t \in \mathbb{T}_{[0,M-1]})$
*ReceiveFromCloud*$(K_{ij,t}^k, K_{ji,t}^k, j \in \mathcal{N}_i, t \in \mathbb{T}_{[0,M-1]})$
set $p_{i,M}^k = \nabla m_i(x_{i,M}^k)$
**for** $t = M-1, \dots, 0$ **do**

$$v_{i,t}^k = -\Big(b_{i,t}^k + B_{ii,t}^{k\top} p_{i,t+1}^k\Big) \tag{11}$$

receive $A_{ji,t}^{cl,k}, v_{j,t}^k, b_{j,t}^k, p_{j,t+1}^k$, $j \in \mathcal{N}_i \setminus \{i\}$

$$z_{i,t}^k = -\sum_{j \in \mathcal{N}_i} \Big(K_{ji,t}^{k\top} v_{j,t}^k\Big)$$

$$p_{i,t}^k = \sum_{j \in \mathcal{N}_i} \Big(A_{ji,t}^{cl,k\ \top} p_{j,t+1}^k - K_{ji,t}^{k\top}\, b_{j,t}^k\Big) + a_{i,t}^k \tag{12}$$

*Send2Cloud*$(z_{i,t}^k, t \in \mathbb{T}_{[0,M]}, v_{i,t}^k, t \in \mathbb{T}_{[0,M-1]})$
*ReceiveFromCloud*$(\beta^k)$
**for** $t = 0, 1, \dots, M-1$ **do**
receive $z_{j,t}^k$ and $x_{j,t}^{k+1}$, $j \in \mathcal{N}_i \setminus \{i\}$
set $\alpha_{j,t}^{k+1} = x_{j,t}^k + \beta^k z_{j,t}^k$, $j \in \mathcal{N}_i$
set $\mu_{i,t}^{k+1} = u_{i,t}^k + \beta^k v_{i,t}^k$

$$u_{i,t}^{k+1} = \mu_{i,t}^{k+1} + \sum_{j \in \mathcal{N}_i} K_{ij,t}^k \Big(\alpha_{j,t}^{k+1} - x_{j,t}^{k+1}\Big) \tag{13}$$

$$x_{i,t+1}^{k+1} = f_i(x_{\mathcal{N}_i,t}^{k+1}, u_{i,t}^{k+1})$$

Let us denote by $x^k, u^k, p^k$ the stacks, for all $i$ and $t$, of $x_{i,t}^k, u_{i,t}^k, p_{i,t}^k$ in Algorithm 2, respectively. We can state the following theorem.

*Theorem 1.* Let $\{x^k, u^k, p^k\}$ be the sequence generated by Algorithm 2. Every limit point of $\{x^k, u^k\}$ is a stationary point of $F(\cdot,\cdot)$. If a limit point $(x^*, u^*)$ of $\{x^k, u^k\}$ exists, there exists also a limit point $p^*$ of $\{p^k\}$ and $(x^*, u^*, p^*)$ satisfies the first-order necessary conditions of optimality for problem (2), i.e., $\nabla \mathcal{L}(x^*, u^*, p^*) = 0$, where $\mathcal{L}(\cdot,\cdot,\cdot)$ denotes the Lagrangian of problem (2). □

Due to space limitations, we provide only an informal idea of the proof of Theorem 1. The proof is divided in two parts. First, we prove that every limit point $\xi^*$ of the sequence of curves $\{\xi^k\}$, where $\xi^k$ is the stack of $\alpha_{i,t}^k, \mu_{i,t}^k$, for all $i$ and $t$, generated via Algorithm 2, is a stationary point of $J(\cdot) := F(\phi(\cdot), \gamma(\cdot))$. To prove this statement, we show that the agents are actually implementing a steepest descent method for problem (10), where, at iteration $k$, the descent direction is the stack of vectors $z_{i,t}^k = -\nabla_{\alpha_{i,t}} J(\xi^k)$, $v_{i,t}^k = -\nabla_{\mu_{i,t}} J(\xi^k)$, for all $i$ and $t$, and the update is $\alpha_{i,t}^{k+1} = \alpha_{i,t}^k + \beta^k z_{i,t}^k$, $\mu_{i,t}^{k+1} = \mu_{i,t}^k + \beta^k v_{i,t}^k$, for all $i$ and $t$, where $\beta^k$ is a step-size. Moreover, if a limit point $\xi^*$ of $\{\xi^k\}$ exists, there exists also $(x^*, u^*)$ such that $x^* = \phi(\xi^*)$ and $u^* = \gamma(\xi^*)$. Second, we prove the existence of $p^*$ and, by using the expression of $\nabla J(\xi)$ obtained by noting that $J(\xi) = \mathcal{L}(\phi(\xi), \gamma(\xi), p)$, $\forall p$, we prove that $(x^*, u^*, p^*)$ satisfies $\nabla \mathcal{L}(x^*, u^*, p^*) = 0$. Note that, since we deal with a finite horizon problem, the convergence of the algorithm does not depend on the stability properties of $K_t$, $t \in \mathbb{N}$. The controller matrix needs to have stabilizability-like properties only to guarantee the numerical robustness of the algorithm.

### *3.3 Design of the distributed projection operator over graph*

In this subsection, we present our strategy to design $K_t$, $t \in \mathbb{T}_{[0,M-1]}$, suitable for the definition of a distributed projection operator over graph in a neighborhood of a trajectory $\tilde{\eta}$. Specifically, $K_t$, $t \in \mathbb{T}_{[0,M-1]}$, has to satisfy a stabilizability-like property on $\tilde{\eta}$ and the sparsity condition such that $K_{ij,t} = 0$ if $j \notin \mathcal{N}_i$, for all $i \in \{1, \dots, N\}$ and $t \in \mathbb{T}_{[0,M-1]}$. For the sake of space, we are not giving all the details for a comprehensive understanding of our strategy, which is an additional contribution of our work. The proposed strategy is, in fact, an extension to (linear) time-varying systems of the one in Lin et al. (2011).

In order to find the controller matrices $K_t$, $t \in \mathbb{T}_{[0,M-1]}$, we consider an optimal control problem on a linear time-varying system (the linearization about a trajectory) with quadratic cost and additional constraints imposing the required structure on each $K_t$. These constraints render the problem quite challenging (while it would be a standard LQR without them). We then re-formulate the optimal control problem by using all matrix variables. Let us define $A_{\mathcal{G}}^c := \mathbf{1} - A_{\mathcal{G}}$, where $\mathbf{1}$ is the matrix with all entries equal to one, and $A_{\mathcal{G}} \in \mathbb{R}^{mN \times nN}$ is the matrix with $i,j$ block $A_{ij,\mathcal{G}} \in \mathbb{R}^{m \times n}$ such that, $A_{ij,\mathcal{G}} = \mathbf{1}$ if $a_{ij} = 1$ and $A_{ij,\mathcal{G}} = 0$ if $a_{ij} = 0$, for all $i$ and $j$. The reformulated problem is

$$\begin{aligned} \min_{\substack{L_0,\dots,L_M \\ K_0,\dots,K_{M-1}}} \ & \frac{1}{2} \sum_{t=0}^{M-1} \operatorname{trace}\Big(\Big(Q + K_t^\top R K_t\Big) L_t\Big) \\ \text{subj. to } & L_{t+1} = (A_t - B_t K_t) L_t (A_t - B_t K_t)^\top, \\ & K_t \circ A_{\mathcal{G}}^c = 0, \quad t \in \mathbb{T}_{[0,M-1]}, \end{aligned} \tag{14}$$

where $Q \in \mathbb{R}^{nN\times nN}$, $R \in \mathbb{R}^{mN\times mN}$ are weight matrices satisfying $Q \geq 0$, $R > 0$, the matrices $L_t \in \mathbb{R}^{nN\times nN}$, $t \in \mathbb{T}_{[0,M-1]}$, are optimization variables together with $K_t \in \mathbb{R}^{mN\times nN}$, $t \in \mathbb{T}_{[0,M-1]}$, the matrices $A_t \in \mathbb{R}^{nN\times nN}$, $B_t \in \mathbb{R}^{nN\times mN}$ have, respectively, non zero blocks $A_{ij,t} := \nabla_{x_{j,t}} f_i(\tilde{x}_{\mathcal{N}_i,t}, \tilde{u}_{i,t})^\top$, $B_{ii,t} := \nabla_{u_{i,t}} f_i(\tilde{x}_{\mathcal{N}_i,t}, \tilde{u}_{i,t})^\top$, $i \in \{1,\dots,N\}$, $j \in \mathcal{N}_i$, and $\circ$ indicates element-wise multiplication. We solve problem (14) by means of the Augmented Lagrangian approach in Algorithm 3, where the augmented Lagrangian of problem (14) is defined as

$$\mathcal{L}_c(K_0,\dots,K_{M-1},\Lambda_0,\dots,\Lambda_{M-1}) := \sum_{t=0}^{M-1}\Big(\frac{1}{2}\text{trace}\Big(\Big(Q+K_t^\top RK_t\Big)L_t\Big) + \text{trace}(\Lambda_t^\top(K_t\circ A_{\mathcal{G}}^c)) + \frac{c}{2}\|K_t\circ A_{\mathcal{G}}^c\|_F\Big),$$

where $L_t$ satisfy, $\forall t \in \mathbb{T}_{[0,M-1]}$, $L_{t+1} = (A_t - B_tK_t)L_t(A_t - B_tK_t)^\top$, $\Lambda_t \in \mathbb{R}^{mN\times nN}$, $t \in \mathbb{T}_{[0,M-1]}$, $c$ is a positive parameter and $\|\cdot\|_F$ denotes the Frobenius norm.

**Algorithm 3** Augmented Lagrangian method for (14)

**Require:** $\Lambda_0^0,\dots,\Lambda_{M-1}^0 = 0$, $c^0 > 0$, $\epsilon > 0$, $\gamma > 1$
**for** $k = 0,1,2\dots$ **do**
compute $K_{t,opt}$, $t \in \mathbb{T}_{[0,M-1]}$, by solving

$$\min_{K_0,\dots,K_{M-1}} \mathcal{L}_c(K_0,\dots,K_{M-1},\Lambda_0^k,\dots,\Lambda_{M-1}^k) \quad (15)$$

**if** $\|K_{t,opt}\circ A_{\mathcal{G}}^c\|_F < \epsilon$, for all $t$, **then**
return
update $\Lambda_t^{k+1} = \Lambda_t^k + c^k(K_{t,opt}\circ A_{\mathcal{G}}^c)$, $t \in \mathbb{T}_{[0,M-1]}$
update $c^{k+1} = \gamma c^k$

Problem (15) in Algorithm 3 is solved, at each iteration $k$, via Algorithm 4, where we set $\bar{\Lambda}_t = \Lambda_t^k$, $t \in \mathbb{T}_{[0,M-1]}$, and $\bar{c} = c^k$. In Algorithm 4, which iterations are also indexed by $k$, the necessary conditions for optimality of problem (15) are used in order to compute $\bar{K}_t^k$, $t \in \mathbb{T}_{[0,M-1]}$. The update is then performed using the step-size $\beta^k$ and the descent direction $\bar{K}_t^k - K_t^k, t \in \mathbb{T}_{[0,M-1]}$.

**Algorithm 4** Alternating method to solve problem (15)

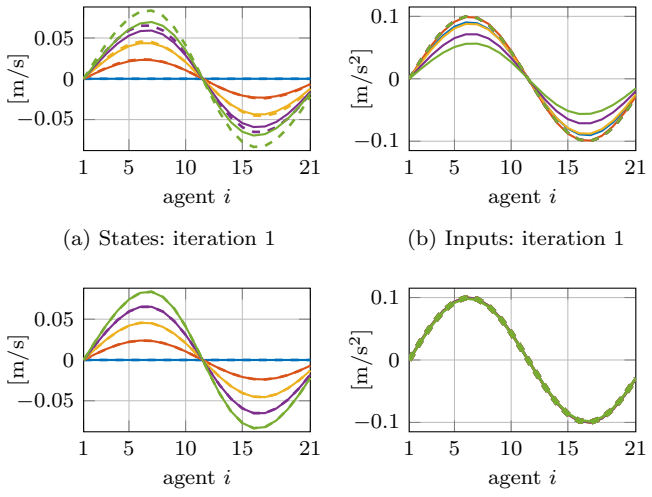
**Require:** $K_t^0$, $\bar{\Lambda}_t$, $t \in \mathbb{T}_{[0,M-1]}$, $\bar{c}$
**for** $k = 0,1,2\dots$ **do**
compute $L_t^k$ and $P_t^k$, $t \in \mathbb{T}_{[0,M]}$, satisfying
$L_{t+1}^k = (A_t - B_tK_t^k)L_t^k(A_t - B_tK_t^k)^\top$,
$P_t^k = (A_t - B_tK_t^k)^\top P_{t+1}^k(A_t - B_tK_t^k) + (Q + K_t^{k\top}RK_t^k)$,
$t \in \mathbb{T}_{[0,M-1]}$,
compute $\bar{K}_t^k$ satisfying, for all $t \in \mathbb{T}_{[0,M-1]}$,
$2\Big(R\bar{K}_t^k - B_t^\top P_{t+1}^k(A_t - B_t\bar{K}_t^k)\Big)L_t^k + \bar{\Lambda}_t + c(\bar{K}_t^k\circ A_{\mathcal{G}}^c) = 0$
compute $\beta^k$ via the Armijo rule
update $K_t^{k+1} = K_t^k + \beta^k(\bar{K}_t^k - K_t^k)$, $t \in \mathbb{T}_{[0,M-1]}$
**if** termination condition satisfied **then**
$K_{t,opt} = K_t^{k+1}$, $t \in \mathbb{T}_{[0,M-1]}$
return

## 4. NUMERICAL COMPUTATIONS

In this section we provide a numerical example showing the effectiveness of the proposed algorithm. We consider an optimal control problem (2) where the dynamics is the finite difference discretization of the Burgers' equation (Hashemi and Werner (2011)) with one-dimensional space variable. We associate an agent to each point of the discretization. In particular, we consider the dynamics in (2) where, for all $i = 1,\dots,N$,

$$f_i(x_{\mathcal{N}_i,t}, u_{i,t}) = x_{i,t} - \frac{\Delta t}{2\Delta s}x_{i,t}(x_{i+1,t} - x_{i-1,t}) + \frac{\nu\Delta t}{\Delta s^2}(x_{i+1,t} - 2x_{i,t} + x_{i-1,t}) + \Delta t\, u_{i,t}, \quad (16)$$

$x_{i,t} \in \mathbb{R}$ represents the velocity of the $i$th point at time $t$, $u_{i,t} \in \mathbb{R}$ represents the acceleration of the $i$th point at time $t$, $x_{0,t} = x_{N,t}$, $x_{N+1,t} = x_{1,t}$, for all $t$, $\nu > 0$ is the viscosity parameter, and $\Delta t$ and $\Delta s$ are discretization steps in time and space respectively. Moreover, we consider the cost in (2) with $\ell_i(x_{i,t}, u_{i,t}) = Q_{cost,i}(x_{i,t} - x_{des,i,t})^2 + R_{cost,i}\,(u_{i,t} - u_{des,i,t})^2$, and $m_i(x_{i,M}) = P_{cost,i,M}\,(x_{i,M} - x_{des,i,M})^2$, for all $i \in \{1,\dots,N\}$, where $x_{des,i,t} \in \mathbb{R}$, $u_{des,i,t} \in \mathbb{R}$ are desired states and inputs of agent $i$ at time $t$, $Q_{cost,i} \in \mathbb{R}$, $R_{cost,i} \in \mathbb{R}$, $P_{cost,i,M} \in \mathbb{R}$.




(a) States: iteration 1 (b) Inputs: iteration 1

(c) States: iteration 8 (d) Inputs: iteration 8

Fig. 1. States $x_{i,t}^k$ (left column) and inputs $u_{i,t}^k$ (right column), for all $i$, at iterations $k = 1$ and $k = 8$ and time instants $t = 0$ (blue), $t = 250$ (red) $t = 500$ (yellow), $t = 750$ (violet), $t = 999$ (green). Dotted lines indicate desired states and inputs. Solid lines indicate actual states and inputs.

For our numerical tests, we set $N = 21$, $M = 1000$, $\Delta t = 0.001$ $s$, $\Delta s = 0.05$ $m$, $\nu = 0.01$ $m^2/s$. We consider the desired input $u_{des,i,t} = 0.1\sin((i-1)\Delta s)$, for all $i$ and $t$, and we generate the desired state $x_{des,i,t}$, for all $i$ and $t$, via an open loop simulation of the dynamics with $x_{i,0} = 0$, for all $i$. Moreover, for the iteration $k = 0$, we use $x_{i,t}^0 = 0$ and $u_{i,t}^0 = 0$, for all $i$ and $t$, as initial trajectory and we run our cloud-assisted distributed algorithm. The results are depicted in Figure 1, where states $x_{i,t}^k$ and inputs $u_{i,t}^k$ are depicted for all $i \in \{1,\dots,N\}$, at selected time instants and at iterations 1 and 8. Note that, at iteration 8, the

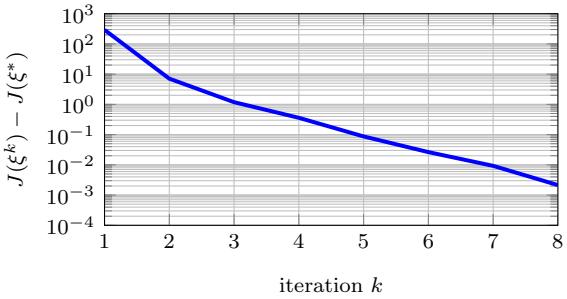



Fig. 2. Cost error $J(\xi^k) - J(\xi^*)$ at algorithm iterations.

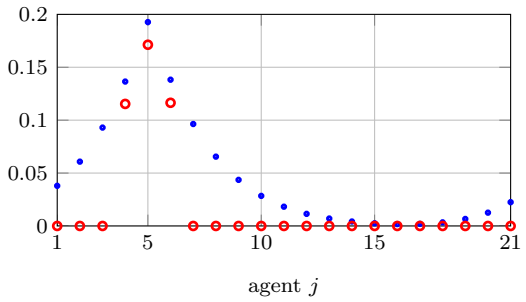



Fig. 3. Sparse vs LQR controller. The values of $K^k_{j5,t}$, for $j = 1, \ldots, 21$, at $t = 0$ and $k = 1$, generated via our strategy, are depicted with red circles. The blue ones are instead obtained by solving an LQR problem.

actual values (solid lines) reach the desired ones (dotted lines). Moreover, Figure 2 shows that the cost decreases at each iteration of our algorithm. Finally, the sparse structure of our controller matrix is depicted in Figure 3, where we compare the values $K^k_{j5,t}$ for $j = 1, \ldots, 21$, at $t = 0$ and $k = 1$, generated via our strategy, with values obtained by solving a standard LQR problem. The red circles are the values of the sparse controller, while the blue circles are the ones of the LQR controller. While for the sparse controller we have $K^1_{j5,0} = 0$, for all $j \notin \mathcal{N}_5$, this condition is not satisfied when we compute the controller matrix via the LQR problem.

## 5. CONCLUSION

In this paper we have proposed a cloud-assisted distributed algorithm to solve optimal control problems on nonlinear dynamics over graph. Our globally convergent algorithm builds on a closed-loop extension of the centralized open loop sequential method. While the latter suffers of numerical instability, our algorithm avoids this drawback via the design of a closed loop dynamics with an ad-hoc sparse controller. This feature enables the agents to compute the descent direction and perform the trajectory update via local communication with neighbors. Numerical computations finally show the effectiveness of the proposed strategy.

## REFERENCES


Ahmed, N., Cortes, J., and Martínez, S. (2016). Distributed control and estimation of robotic vehicle networks: overview of the special issue. *IEEE Control Systems*, 36(2), 36–40.

Bayer, F.A., Notarstefano, G., and Allgöwer, F. (2013). A projected SQP method for nonlinear optimal control with quadratic convergence. In *IEEE Conference on Decision and Control (CDC)*, 6463–6468.

Bertsekas, D.P. (1999). *Nonlinear programming*. Athena scientific Belmont.

Conte, C., Jones, C.N., Morari, M., and Zeilinger, M.N. (2016). Distributed synthesis and stability of cooperative distributed model predictive control for linear systems. *Automatica*, 69, 117–125.

Dinh, Q.T., Necoara, I., and Diehl, M. (2013). A dual decomposition algorithm for separable nonconvex optimization using the penalty function framework. In *IEEE Conference on Decision and Control (CDC)*, 2372–2377.

Doan, M.D., Keviczky, T., and De Schutter, B. (2011). An iterative scheme for distributed model predictive control using Fenchel's duality. *Journal of Process Control*, 21(5), 746–755.

Ferrari-Trecate, G., Buffa, A., and Gati, M. (2006). Analysis of coordination in multi-agent systems through partial difference equations. *IEEE Transactions on Automatic Control*, 51(6), 1058–1063.

Giselsson, P., Doan, M.D., Keviczky, T., De Schutter, B., and Rantzer, A. (2013). Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3), 829–833.

Gray, R., Franci, A., Srivastava, V., and Leonard, N.E. (2018). Multi-agent decision-making dynamics inspired by honeybees. *IEEE Transactions on Control of Network Systems*.

Groß, D. and Stursberg, O. (2016). A cooperative distributed MPC algorithm with event-based communication and parallel optimization. *IEEE Transactions on Control of Network Systems*, 3(3), 275–285.

Hashemi, S.M. and Werner, H. (2011). LPV Modelling and Control of Burgers' Equation. *IFAC Proceedings Volumes*, 44(1), 5430–5435.

Hauser, J. (2002). A projection operator approach to the optimization of trajectory functionals. *IFAC Proceedings Volumes*, 35(1), 377–382.

Lin, F., Fardad, M., and Jovanović, M.R. (2011). Augmented Lagrangian approach to design of structured optimal state feedback gains. *IEEE Transactions on Automatic Control*, 56(12), 2923–2929.

Mårtensson, K. and Rantzer, A. (2012). A scalable method for continuous-time distributed control synthesis. In *American Control Conference (ACC), 2012*, 6308–6313.

Massioni, P. and Verhaegen, M. (2009). Distributed control for identical dynamically coupled systems: A decomposition approach. *IEEE Transactions on Automatic Control*, 54(1), 124–135.

Necoara, I., Savorgnan, C., Tran, D.Q., Suykens, J., and Diehl, M. (2009). Distributed nonlinear optimal control using sequential convex programming and smoothing techniques. In *IEEE Conference on Decision and Control*, 543–548.

Ravazzi, C., Tempo, R., and Dabbene, F. (2017). Learning influence structure in sparse social networks. *IEEE Transactions on Control of Network Systems*.

Wu, X., Dörfler, F., and Jovanović, M.R. (2016). Input-output analysis and decentralized optimal control of inter-area oscillations in power systems. *IEEE Transactions on Power Systems*, 31(3), 2434–2444.