ADMM-based Cooperative Control for Platooning of Connected and Autonomous Vehicles

Evangelos Vlachos, Aris S. Lalos,
Industrial Systems Institute, ATHENA Research Center, Platani Riou, 26504, Patras, Greece,
E-mails: {evlachos, lalos}@isi.gr

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9839099

***Abstract*—Distributed model-predictive controllers provide a robust way to adjust the acceleration of each platoon vehicle and avoid collisions. This is achieved by transforming the control problem into an iterative, finite-horizon optimization with local constraints. However, the derivation of the global optimal solution is not straightforward. In this paper, first, the consensus cost function is formulated, constrained by minimum distance requirements between the vehicles. Then, the solution is derived via the alternating direction method of multipliers (ADMM), an iterative and robust solver with minimal communication demands. A low-complexity solution is proposed by casting the problem as stochastic control optimization. The developed techniques are evaluated via simulations, where the trajectory of the leading vehicle is generated by an open-source software for autonomous driving (CARLA).**

I. Introduction

Cooperative platooning of connected and autonomous vehicles (CAVs) provides an efficient traffic management system, that increases the fuel economy, and enhances the road utility and safety in smart cities and highways [1, 2]. To achieve these goals, novel decentralized and intelligent motion controllers are required to automatically accelerate and decelerate the CAVs, to keep a desired distance from the preceding vehicle. Model Predictive Control (MPC) is a state-of-the-art technique which can be used for intelligent real-time optimal control [3, 4]. At each time instance, a constrained optimization problem is formulated, based on the plant specifications, to minimize a defined cost function in a predictive time horizon. The optimization problem is solved online, and this predictive control procedure is repeated with updated states. Distributed model predictive control (DMPC) is the networked version of the MPC strategy. The global cost function is optimized through the contribution of local controllers at the agents, which are physically coupled, and the communication between the local controllers is required for stability and performance.

The inherent difficulty of the MPC optimization problems, namely a high computation demand, arises in DMPC as the need for a large amount of information exchange among the local controllers. Efficient DMPC techniques aim to solve the DMPC problem with the lowest possible communication load on the network while still satisfying the optimality of the solution. In [5], the dual of the DMPC problem is formulated using Lagrange multipliers and evaluated on randomly generated optimization problems. A distributed version of the fast gradient method is then employed to solve the dual problem, which converges faster than the gradient method. However, the cost function was assumed separable, which does not hold for the DMPC problem. In [6] an algorithm is proposed that solves the convex quadratic programming problem based on the conjugate gradient (CG) algorithm. However, it requires several communication steps, to calculate the derivatives in a distributed manner. In [7], the DMPC problem is first recast as a global consensus problem and then the alternating direction method of multipliers (ADMM) is employed to find an optimal solution to the primal and dual of the DMPC problem.

In this work, we consider the problem of cooperative control of a CAVs platoon. Inspired by [7], we employ the ADMM to solve the control problem that occurs from the distributed model predictive controller. We propose the employment of a penalty function to substitute the constraints of the DMPC problem. This simplifies the problem in terms of complexity cost, and two different ADMM-based algorithms are derived with different tracking performance and computational cost trade-offs. The evaluation of the proposed techniques is obtained via tracking the distance and speed errors between the vehicles, as well as the cumulative distribution function (CDF) of the distance between the successive vehicles, based on Monte-Carlo simulations. The trajectories for the leading vehicle has been obtained using the Carla simulator [8]. The simulation results verify that the proposed ADMM solution is able to perform similarly to interior-point methods, with lower computational and communication costs.

Notation: $\mathbf{A}$, $\mathbf{a}$, a, $|\cdot|$, $(\cdot)^T$, $(\cdot)^H$ and $\|\cdot\|_F$ denote matrix, vector, scalar, determinant, transpose, complex conjugate transpose and Frobenius norm, respectively; $[\mathbf{A}]_{ij}$ and $[\mathbf{A}]_k$ is (i,j)-th element and the k-th row of matrix $\mathbf{A}$, respectively; $\mathbb{C}$, $\mathbb{R}$ and $\mathcal{E}$ denote sets of complex and real numbers, and expectation operator, respectively; $\text{diag}(\mathbf{x})$ denotes the $N \times N$ diagonal matrix with the $N \times 1$ vector $\mathbf{x}$ on its diagonal.

II. Cooperative Control Preliminaries

Cooperative control aims to design appropriate distributed algorithms such that the group of vehicles can reach consensus on the shared information in the presence of limited and unreliable information exchange and dynamically changing interaction topologies [9]. We use an information graph $\mathcal{G} = (V, E)$, to model the interaction topology between the vehicles. The set V denotes the nodes, i.e., the vehicles, and the set $E \subseteq V \times V$ the edges, which specifies the information flow between neighboring vehicles. An edge (i, j) exists if the i-th vehicle has information exchange with the j-th

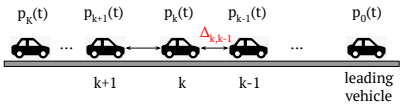

Fig. 1. Illustration of the 1D platooning with a leading and following vehicles.

vehicle. The set of neighbors of the i-th vehicle is defined as $\mathcal{N}_i \triangleq \{j \in V : (i,j) \in E\}$.

Let us consider a network of vehicles moving in line, as shown in Fig 1. For ease of exposition, we only consider one dimension (1D) of the translation motion. Note that, the following analysis is also applicable to two and three dimensions, as long as the dynamics of the vehicle in each coordinate of the Euclidean space is decoupled. The position of the k-th vehicle is denoted by p_k, its velocity by $\dot{p}_k$, and its acceleration $\ddot{p}_k(t)$. Double-integrator dynamic model captures the dynamics of physical agents such as CAVs in a platooning system. The information states with double-integrator dynamics for the k-th vehicle is given by:

$$\ddot{p}_k = u_k + n_k, \tag{1}$$

where u_k is the control input, n_k is the external disturbance, and $k = 1, \ldots, K$. The control objective is to make the CAVs maintain a rigid formation geometry by following a desired trajectory.

We consider the following distributed linear control law, where the control action u_k only depends on the relative position and relative velocity information from its neighbors [9], i.e.,

$$u_k = -\sum_{j \in \mathcal{N}_k} \left(c_1(p_k - p_j + \Delta p_{j,k}) + c_2(\dot{p}_k - \dot{p}_j)\right), \tag{2}$$

where $\Delta p_{j,k}$ is the desired inter-vehicle gap between vehicles k and j, while c_1 and c_2 are positive constants.

III. Controller Design for Cooperative Platooning

For the network of CAVs, the control objective is to make the CAVs maintain a rigid formation geometry by following a desired trajectory. The desired geometry of the formation is specified by the desired gap, as shown in Fig. 1. For each vehicle k, with $k = 1, 2, \ldots, K$, we assume that the underlying physical modeling (plant) at each time instance t, is described by the state and input/output equations, i.e.,

$$\boldsymbol{\xi}_k(t+1) = \hat{\mathbf{A}}_k \boldsymbol{\xi}_k(t) + \hat{\mathbf{b}}_k u_k(t), \tag{3}$$

$$y_k(t) = \hat{\mathbf{c}}_k^T \boldsymbol{\xi}_k(t), \tag{4}$$

where $\boldsymbol{\xi}_k(t) \in \mathbb{R}^{2\times 1}$ is the state vector of the k-th vehicle, $u_k(t) \in \mathbb{R}$ is the control input, $y_k(t) \in \mathbb{R}$ is the measured output, and the system matrices/vectors are expressed as:

$$\hat{\mathbf{A}}_k = \begin{bmatrix} 0 & 1 \\ -c_{1,k} & -c_{2,k} \end{bmatrix}, \hat{\mathbf{b}}_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \hat{\mathbf{c}}_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

For simplicity, we assume identical vehicles, thus $c_{1,1} = \ldots, c_{1,K} \triangleq c_1$, and $c_{2,1} = \ldots = c_{2,K} \triangleq c_2$. Moreover, we adopt a common sub-index to denote the per-vehicle quantities, i.e.,

$$\hat{\mathbf{A}}_s \triangleq \hat{\mathbf{A}}_k,\ \hat{\mathbf{b}}_s \triangleq \hat{\mathbf{b}}_k,\ \hat{\mathbf{c}}_s \triangleq \hat{\mathbf{c}}_k \text{ for } k = 1, \ldots, K. \tag{5}$$

A. Unconstrained MPC

For the single-directional architecture of the platoon (Fig. 1), the generalized Laplacian $\mathbf{L} \in \mathbb{R}^{K\times K}$ is a non-symmetric matrix with the main diagonal and an upper diagonal, namely:

$$[\mathbf{L}]_{ij} = \begin{cases} 1 & \text{for } i = j, i = 1, \ldots, K, \\ -1/K & \text{for } j = i+1, \\ 0 & \text{elsewhere.} \end{cases} \tag{6}$$

The Laplacian matrix is used to extend the plant model of (3)-(4) for the entire fleet of vehicles, where the network-wide plant is expressed as [9]:

$$\boldsymbol{\xi}(t+1) = \hat{\mathbf{A}}\boldsymbol{\xi}(t) + \hat{\mathbf{B}}\mathbf{u}(t), \tag{7}$$

$$\mathbf{y}(t) = \hat{\mathbf{C}}^T \boldsymbol{\xi}(t), \tag{8}$$

where $\boldsymbol{\xi}(t) \triangleq \begin{bmatrix} \boldsymbol{\xi}_1^T(t) \cdots \boldsymbol{\xi}_K^T(t) \end{bmatrix}^T \in \mathbb{R}^{2K\times 1}$ is the state vector for all vehicles, $\mathbf{u}(t) \triangleq \begin{bmatrix} u_1(t) \cdots u_K(t) \end{bmatrix}^T \in \mathbb{R}^{K\times 1}$ is the control input vector for the K vehicles, $\mathbf{y}(t) \in \mathbb{R}^{K\times 1}$ is the measured output vector of all the K vehicles, while

$$\hat{\mathbf{A}} \triangleq \mathbf{L}(t) \otimes \hat{\mathbf{A}}_s \in \mathbb{R}^{2K\times 2K}, \hat{\mathbf{B}} \triangleq \mathbf{L}(t) \otimes \hat{\mathbf{b}}_s \in \mathbb{R}^{2K\times K} \tag{9}$$

and

$$\hat{\mathbf{C}} \triangleq \mathbf{L}(t) \otimes \hat{\mathbf{c}}_s^T \in \mathbb{R}^{K\times 2K},$$

where $\mathbf{L}(t)$ is the Laplacian matrix for the t-th time instance. Following the standard MPC problem formulation [10], we take the difference on both sides of (7), i.e.,

$$\boldsymbol{\xi}(t+1) - \boldsymbol{\xi}(t) = \hat{\mathbf{A}}(\boldsymbol{\xi}(t) - \boldsymbol{\xi}(t-1)) + \hat{\mathbf{B}}(\mathbf{u}(t) - \mathbf{u}(t-1)) \tag{10}$$

or equivalently, the new state equation is expressed as:

$$\boldsymbol{\Delta\xi}(t+1) = \hat{\mathbf{A}}\boldsymbol{\Delta\xi}(t) + \hat{\mathbf{B}}\boldsymbol{\Delta}\mathbf{u}(t), \tag{11}$$

where $\boldsymbol{\Delta\xi}(t+1) \triangleq \boldsymbol{\xi}(t+1) - \boldsymbol{\xi}(t)$, and

$$\boldsymbol{\Delta}\mathbf{u}(t) \triangleq \mathbf{u}(t) - \mathbf{u}(t-1).$$

Furthermore, the new input/output model is expressed as:

$$\begin{aligned} \mathbf{y}(t+1) &= \mathbf{y}(t) + \hat{\mathbf{C}}^T(\boldsymbol{\xi}(t+1) - \boldsymbol{\xi}(t)) = \mathbf{y}(t) + \hat{\mathbf{C}}^T\boldsymbol{\Delta\xi}(t+1) \\ &= \mathbf{y}(t) + \hat{\mathbf{C}}^T\hat{\mathbf{A}}\boldsymbol{\Delta\xi}(t) + \hat{\mathbf{C}}^T\hat{\mathbf{B}}\boldsymbol{\Delta}\mathbf{u}(t). \end{aligned} \tag{12}$$

The next step is to define a new state variable vector which connects the state $\boldsymbol{\Delta\xi}(t)$ with the output $\mathbf{y}(t)$. Thus, putting together (11) and (12) leads to the following augmented state-space model:

$$\underbrace{\begin{bmatrix} \boldsymbol{\Delta\xi}(t+1) \\ \mathbf{y}(t+1) \end{bmatrix}}_{\mathbf{x}(t+1)} = \underbrace{\begin{bmatrix} \hat{\mathbf{A}} & \mathbf{O}_{K,K} \\ \hat{\mathbf{C}}^T\hat{\mathbf{A}} & \mathbf{E}_{K,K} \end{bmatrix}}_{\triangleq \mathbf{A}} \underbrace{\begin{bmatrix} \boldsymbol{\Delta\xi}(t) \\ \mathbf{y}(t) \end{bmatrix}}_{\mathbf{x}(t)} + \underbrace{\begin{bmatrix} \hat{\mathbf{B}} \\ \hat{\mathbf{C}}^T\hat{\mathbf{B}} \end{bmatrix}}_{\triangleq \mathbf{B}} \boldsymbol{\Delta}\mathbf{u}(t) \tag{13}$$

$$\mathbf{y}(t+1) = \underbrace{\begin{bmatrix} \mathbf{O}_{K,K} & \mathbf{E}_{K,K} \end{bmatrix}}_{\triangleq \mathbf{C}} \begin{bmatrix} \boldsymbol{\Delta\xi}(t) \\ \mathbf{y}(t) \end{bmatrix}, \tag{14}$$

where $\mathbf{O}_{K,K} \in \{0\}^{K\times K}$ is a matrix filled with zeros, and $\mathbf{E}_{K,K} \in \{1\}^{K\times K}$ is a matrix filled with ones.

The MPC for prediction horizon of N_p and control horizon of N_c is expressed compactly as:

$$\mathbf{z}(t+1) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{v}(t), \tag{15}$$

where

$$\mathbf{z}(t+1) \triangleq \begin{bmatrix} \mathbf{x}^T(t+1) & \cdots & \mathbf{x}^T(t+N_p) \end{bmatrix}^T \in \mathbb{R}^{2N_pK\times 1},$$

$$\mathbf{v}(t) \triangleq \begin{bmatrix} \mathbf{\Delta u}^T(t) & \cdots & \mathbf{\Delta u}^T(t+N_c-1) \end{bmatrix}^T \in \mathbb{R}^{N_cK\times 1},$$

$$\mathbf{F} = \begin{bmatrix} (\mathbf{CA})^T & \dots & (\mathbf{CA}^{N_p})^T \end{bmatrix}^T \in \mathbb{R}^{2N_pK\times 2K}, \tag{16}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{CB} & & \\ \vdots & \ddots & \\ \mathbf{CA}^{N_p-1}\mathbf{B} & \cdots & \mathbf{CA}^{N_c-1}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{2N_pK\times N_cK}. \tag{17}$$

Let us define the centralized cost function

$$\mathcal{J}(\mathbf{v}(t)) = \|\boldsymbol{\beta}(t) - \mathbf{F}\mathbf{x}(t) - \mathbf{G}\mathbf{v}(t)\|^2, \tag{18}$$

where

$$\boldsymbol{\beta}(t) \triangleq \mathbf{E}_{N_p,1} \otimes \begin{bmatrix} \Delta p_1^*(t) & \cdots & \Delta p_K^*(t) \end{bmatrix}^T \in \mathbb{R}^{N_pK\times 1},$$

is the set-point information, defined as:

$$\Delta p_k^*(t) = p_k^*(t) - p_k^*(t-1), \tag{19}$$

and $p_k^*(t)$ is the provided waypoint for the k-th vehicle at the t-th time instance, and $\mathbf{E}_{N_p,1} \in \{1\}^{N_p\times 1}$. Note that, the waypoints for the leading vehicle (e.g., $k=1$) are provided externally. For $k>1$, we use $p_k^*(t) \triangleq p_{k-1}(t-1)$, which expressed that the k-th vehicle has to follow the preceding vehicle $k-1$ and its trajectory.

Then, the unconstrained optimization problem that provides the MPC solution is expressed as:

$$\min_{\mathbf{v}(t)} \mathcal{J}(\mathbf{v}(t)), \tag{20}$$

which has a closed-form solution given by solving of the following system of equations, $\mathbf{G}\mathbf{v}(t) = \boldsymbol{\beta}(t) - \mathbf{F}\mathbf{x}(t)$.

B. Constrained MPC

The unconstrained MPC design provides a simple solution to the problem, with minimal computational cost, that exhibits desirable performance in the case where the leading vehicle moves with constant speed. In realistic scenarios, where the speed of the leading vehicle is variable, it fails to guarantee a safe distance between the vehicles. To overcome this problem, we formulate a constrained MPC problem, which ensures that the $k-1$-th vehicle will be in front of the k vehicle. Let the position of the leading vehicle is p_0 on the x-axe, as shown in Fig. 1, while for the K following vehicles we assume that:

$$p_0 < p_1 < \ldots < p_{k-1} < p_k < \ldots < p_K.$$

Then, the constrained MPC problem formulation for all K vehicles is expressed as:

$$\min_{\mathbf{v}(t)} \mathcal{J}(\mathbf{v}(t)) \text{ s.t. } p_k(t) \geq p_{k-1}(t) \text{ for } k=1,\ldots,K, \tag{21}$$

where, recall that, the vector $\mathbf{v}(t) \in \mathbb{R}^{N_cK\times 1}$ is the control input of the MPC.

To solve (21), first we have to treat the inequality constraints. Thus, we introduce K auxiliary variables y_k, with $k=1,\ldots,K$, and the penalty function $I(y_k)$ which is defined as:

$$I(y_k) = \begin{cases} \infty & y_k < 0 \\ 0 & y_k \geq 0. \end{cases} \tag{22}$$

Then, the problem (21) becomes [11], for $k=1,\ldots,K$:

$$\min_{\substack{\mathbf{v}(t) \\ \{y_k\}_{k=1}^K}} \mathcal{J}(\mathbf{v}(t)) + \sum_{k=1}^{K} I(y_k(t)) \text{ s.t. } p_{k-1}(t) - p_k(t) - y_k(t) = 0. \tag{23}$$

As it will be seen next, the problem (23) can be solved using the ADMM. To proceed, let us first express the constraints with respect to the unknown variable $\mathbf{v}(t)$. Specifically, the estimated position of the k-th vehicle can be expressed by using the state vector as $p_k(t) = [\boldsymbol{\xi}(t)]_k$, where $[\mathbf{q}]_k$ denotes the k-th element of the vector $\mathbf{q}$. Thus,

$$[\boldsymbol{\xi}(t)]_k - [\boldsymbol{\xi}(t)]_{k-1} = \tag{24}$$

$$[\hat{\mathbf{A}}\boldsymbol{\xi}(t-1) + \hat{\mathbf{B}}\mathbf{u}(t-1)]_k - [\hat{\mathbf{A}}\boldsymbol{\xi}(t-1) + \hat{\mathbf{B}}\mathbf{u}(t-1)]_{k-1}$$
$$= ([\hat{\mathbf{A}}]_k - [\hat{\mathbf{A}}]_{k-1})\boldsymbol{\xi}(t-1) + ([\hat{\mathbf{B}}]_k - [\hat{\mathbf{B}}]_{k-1})\mathbf{u}(t-1)$$
$$= \gamma_k(t) + [\mathbf{R}]_k\mathbf{u}(t), \tag{25}$$

where

$$\gamma_k(t) \triangleq ([\hat{\mathbf{A}}]_k - [\hat{\mathbf{A}}]_{k-1})\boldsymbol{\xi}(t-1),$$

and $[\mathbf{R}]_k \triangleq [\hat{\mathbf{B}}]_k - [\hat{\mathbf{B}}]_{k-1}$.

Recall that, $\mathbf{u}(t) = \mathbf{u}(t-1) + \mathbf{\Delta u}(t)$, while $\mathbf{\Delta u}(t)$ represents the first K entries of the vector $\mathbf{v}(t)$. In order express (25) with respect to $\mathbf{v}(t)$, let us employ the following mathematical notation,

$$\mathbf{u}(t) = \text{diag}(\mathbf{e}_K)\mathbf{v}(t), \tag{26}$$

where $\mathbf{e}_K \in \{0,1\}^{KN_c\times 1}$ with $[\mathbf{e}_K]_i = 1$ for $i=1,\ldots,K$ and $[\mathbf{e}_K]_i = 0$ for $i=K+1,\ldots,KN_c$. Then,

$$[\boldsymbol{\xi}(t)]_k - [\boldsymbol{\xi}(t)]_{k-1} = \gamma_k(t) + [\mathbf{R}]_k\text{diag}(\mathbf{e}_K)\mathbf{v}(t), \tag{27}$$

Therefore, by dropping the time index for simplicity, the problem (23) becomes, for $k=1,\ldots,K$:

$$\min_{\mathbf{v},\mathbf{y}} \mathcal{J}(\mathbf{v}) + \sum_{k=1}^{K} I(y_k) \text{ s.t. } \gamma_k + [\mathbf{R}]_k\text{diag}(\mathbf{e}_K)\mathbf{v} - y_k = 0 \tag{28}$$

Let q_k, for $k=1,\ldots,K$, denote the K dual variables, then the augmented Lagrangian function is expressed as:

$$\mathcal{C}(\mathbf{v},\{y_k\},\{q_k\}) = \mathcal{J}(\mathbf{v}) + \sum_{k=1}^{K} I(y_k)$$
$$+ \frac{\rho}{2}(\|\gamma_k + [\mathbf{R}]_k\text{diag}(\mathbf{e}_K)\mathbf{v} - y_k + q_k\|^2 + \|q_k\|^2). \tag{29}$$

Next, we employ the ADMM steps to solve (29). These steps are executed for a number of iterations, $i = 1, \ldots, I_{\max}$. The first step is to minimize $\mathcal{C}(\mathbf{v}, \{y_k\}, \{q_k\})$ over $\mathbf{v}$, e.g.,

$$\mathbf{v}^{(i+1)} = arg\min_{\mathbf{v}} \|\boldsymbol{\beta} - \mathbf{F}\mathbf{x} - \mathbf{G}\mathbf{v}\|^2 + \frac{\rho}{2}(\|\gamma_k + [\mathbf{R}]_k \text{diag}(\mathbf{e}_K)\mathbf{v} - y_k^{(i)} + q_k^{(i)}\|^2). \tag{30}$$

The minimization problem of (30) is equivalent to the solution of the following system of equations [11]

$$(\mathbf{G}^T\mathbf{G} + \rho \text{diag}(\mathbf{e}_K)^T \sum_{k=1}^{K} [\mathbf{R}]_k^T [\mathbf{R}]_k \text{diag}(\mathbf{e}_K))\ \mathbf{v}^{(i+1)} = \mathbf{G}^T\boldsymbol{\beta} + \rho \sum_{k=1}^{K} (\gamma_k - y_k^{(i)} + q_k^{(i)}). \tag{31}$$

Note that, the quantities $\mathbf{G}, \mathbf{R}, \boldsymbol{\beta}$ are known to all vehicles and assumed static over the interval of interest. Hence, the solution of (31) requires only the scalar quantities y_k and u_k from the other vehicles for consensus.

At the next step, we use the estimated value $\mathbf{v}^{(i+1)}$, and solve over y_k, for $k = 1, \ldots, K$ [10]:

$$y_k^{(i+1)} = \max(0, \gamma_k + [\mathbf{B}]_k \text{diag}(\mathbf{e}_K)\mathbf{v}^{(i+1)} + q_k^{(i)}) \tag{32}$$

At the third step, we update the dual variable for $k = 1, \ldots, K$:

$$q_k^{(i+1)} = q_k^{(i)} + \gamma_k + [\mathbf{B}]_k \text{diag}(\mathbf{e}_K)\mathbf{v}^{(i+1)} - y_k^{(i+1)} \tag{33}$$

The described iterative steps of the ADMM method that solve (23) are provided in Algorithm 1.

Algorithm 1 ADMM algorithm

Input: ρ, $\boldsymbol{\beta}(t)$, $\mathbf{G}$, $\gamma(t)$
Output: $\mathbf{v}(t)$
1: **for** $i = 1, \ldots, I_{\max}$ **do**
2: Each vehicle solves its own version of (31)
3: $y_k^{(i+1)} = \max(0, \gamma_k + [\mathbf{B}]_k \text{diag}(\mathbf{e}_K)\mathbf{v} + q_k^{(i)})$, for all k
4: Each vehicle broadcasts $y_k^{(i+1)}$
5: $q_k^{(i+1)} = q_k^{(i)} + \gamma_k + [\mathbf{B}]_k \text{diag}(\mathbf{e}_K)\mathbf{v} - y_k^{(i+1)}$, for all k
6: Each vehicle broadcasts $q_k^{(i+1)}$
7: **end for**

IV. Efficient Implementation

The computational complexity of the Algorithm 1 depends on the parameters of the problem, such as the platoon size, the prediction horizon of N_p, the control horizon of N_c, and the number of iterations, $I_{\max}$, that are required for the convergence of the ADMM algorithm. At each iteration i, the most costly step is at line 3, which requires the solution of (31). Moreover, at each iteration i, the updated quantities have to broadcasted to the other vehicles, increasing the communication load. Thus, to reduce the overall complexity of the Algorithm 1, either we have to reduce the cost of step 3, or we have to minimize the number of iterations $I_{\max}$. To reduce the cost of step 3, the direct inversion method could be replaced with an iterative solver, e.g., in [6].

In this work, we focus on the case where the number of iterations is reduced to $I_{\max} = 1$. This will result into a special case of the algorithm, listed in Algorithm 2. Indeed, its convergence is achieved over the time instances, thus, the problem cost function $\mathcal{J}(\mathbf{v}(t))$ will be different at each time iteration. Given that the cost function is not varying significantly at each time update, the cost function can be seen as *stochastic*, i.e., it also depends on a random variation. In this case, it can be seen that the proposed algorithm converges to the minimum value of the constrained stochastic optimization,

$$\min_{\mathbf{v}(t), \{y_k\}_{k=1}^K} \mathcal{E}\{\mathcal{J}(\mathbf{v}(t), \xi)\} + \sum_{k=1}^{K} I(y_k(t)) \quad \text{subject to } p_{k-1}(t) - p_k(t) - y_k(t) = 0 \text{ for } k = 1, \ldots, K, \tag{34}$$

where ξ is an unknown random variable with zero mean and variance σ_ξ^2, which depends on the variability of the cost function. Next, due to high sampling frequency, we assume that σ_ξ is negligible. Thus, to formulate a solution for (34), first we drop the iteration indexing (i) and replace with the time instances index, i.e., $\mathbf{v}^{(i+1)}(t) = \mathbf{v}(t+1)$, and for $k = 1, \ldots, K$: $q_k^{(i+1)}(t) = q_k(t+1)$, and $y_k^{(i+1)}(t) = y_k(t+1)$. Then, the steps can be similarly obtained as in (30) - (33), as shown in Algorithm 2.

Although Algorithm 2 reduces significantly the computational complexity, it may also introduce an error due to the unknown random variable. However, ADMM is able to converge under relaxed conditions for the introduced error at each step, while a common measure to ensure the convergence, is to adopt a time-varying step-length $\rho(t)$ [11], such as:

$$\rho(t) = \epsilon\rho(t-1),$$

where $\epsilon \in [0, 1]$. Then, it can be shown that: $\sum_t \rho(t) < \infty$.

Algorithm 2 ADMM-L algorithm

Input: $\rho(t)$, $\boldsymbol{\beta}(t)$, $\mathbf{G}$, $\gamma(t)$, $\mathbf{v}(t)$, $\{y_k(t)\}_{k=1}^K$, $\{u_k(t)\}_{k=1}^K$
Output: $\mathbf{v}(t+1)$, $\{y_k(t+1)\}_{k=1}^K$, $\{u_k(t+1)\}_{k=1}^K$
1: Each vehicle solves its own version of

$$(\mathbf{G}^T\mathbf{G} + \rho(t) \text{diag}(\mathbf{e}_K)^T \sum_{k=1}^{K} [\mathbf{R}]_k^T [\mathbf{R}]_k \text{diag}(\mathbf{e}_K))\ \mathbf{v}(t+1) = \mathbf{G}^T\boldsymbol{\beta} + \rho(t) \sum_{k=1}^{K} (\gamma_k - y_k(t) + q_k(t)).$$

2: $y_k(t+1) = \max(0, \gamma_k + [\mathbf{B}]_k \text{diag}(\mathbf{e}_K)\mathbf{v} + q_k(t), \forall k$
3: Each vehicle broadcasts $y_k(t+1)$
4: $q_k(t+1) = q_k(t) + \gamma_k + [\mathbf{B}]_k \text{diag}(\mathbf{e}_K)\mathbf{v} - y_k(t+1), \forall k$
5: Each vehicle broadcasts $q_k(t+1)$
6: $\rho(t) = \epsilon\rho(t-1)$

In Table I, we compare the computational and communication cost for each of the considered techniques using Big-O notation. The computation cost measures the number of

TABLE I
ORDER OF COMPUTATIONAL AND COMMUNICATION COSTS

	MPC	ADMM	ADMM-L
Computation	$\mathcal{O}(N_p^2 N_c K^3)$	$I_{\max}\mathcal{O}(N_p^2 N_c K^3)$	$\mathcal{O}(N_p^2 N_c K^3)$
Communication	$\mathcal{O}(N_c K)$	$I_{\max}\mathcal{O}(N_c K)$	$\mathcal{O}(N_c K)$

Setting	Value
Number of vehicles	5
Update time T_c	1s
Inter-vehicle gap	1m
Max. speed	10 m/s
Control horizon N_c	5
Samples number N_p	10

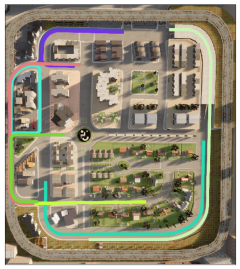

Fig. 2. Simulation Parameters and Road network instance, taken from Carla simulator, showing the trajectories of several vehicles.

multiplications between matrices. The communication cost measures the quantity of scalars that have to be the broadcast of the Laplacian matrix and the $N_c \times 1$ state vector of the $K-1$ vehicles.

V. SIMULATION RESULTS

In this section, we validate the introduced theoretical framework, by performing computer based simulations, using the MATLAB$^{\text{TM}}$ software and the Carla simulator for generating vehicle trajectories.

Initially, all the vehicles are stopped, placed with a Δ meters gap between them. We consider several variable speed scenarios for the leading vehicle, generated using the Carla open-source simulator for autonomous driving [8]. In both cases, the goal is to end up with a platoon formation where the cars follow the leading vehicle, avoiding any collisions between them. To avoid collisions, the vehicles have to keep a safe distance, while the controller has to update its state at each T_c second. More details about the parameters of the simulation results are shown in the Table of Fig. 2.

We consider three open-loop controller designs:

- MPC: The conventional closed-form solution of the uncosntrained optimization problem of (20).
- ADMM: The proposed solution of the constrained optimization problem of (28) provided by Algorithm 1.
- ADMM-L: The proposed low-complexity solution of (28) provided by Algorithm 2.

Concerning the computation and communication overhead, it is important to note that, in our simulations, we have also employed the standard interior-point method, using the CVX [12] public library. However, in all cases, the output of CVX was identical to the ADMM results, and for that reason are not displayed in the figures. As a figure of merit, the proposed ADMM solution was 35x faster than the CVX interior-point technique, while the proposed ADMM-L was 35x faster than the ADMM. As for the communcation cost, the proposed ADMM requires only the transmission of scalar quantities between the vehicles, for each iteration i in Algorithm 1, avoiding the transmission of whole matrices as for the

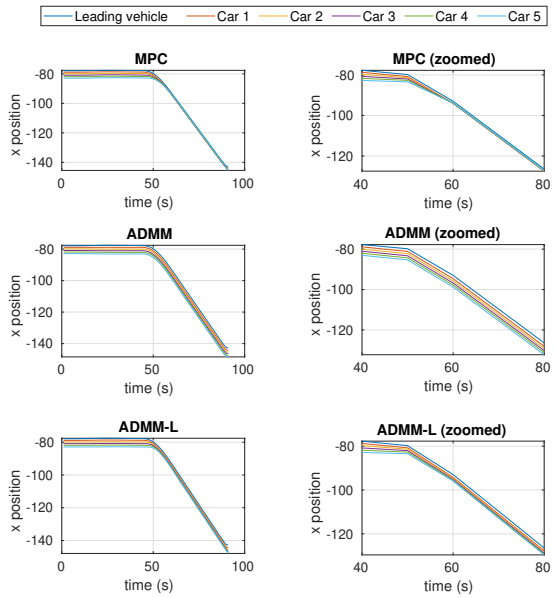

Fig. 3. Vehicles trajectory on the x-axe over time for the case where the leading vehicle follows the trajectory of ID 86 for the Carla generated scenario.

standard interior-point methods (e.g., Newton-based methods). Moreover, the ADMM-L futher reduces the communication overhead, by minimizing the number of iterations $I_{\max}$.

In Fig. 3 we plot the vehicle trajectories on the x-axe for the considered controller techniques over time. The leading vehicle is not moving till $t = 45s$, and then it starts to move over x-axe with relative constant speed. The MPC technique results into trajectories that collide after $t = 50s$. The constrained MPC techniques, represented by the ADMM and ADMM-L techniques, are able to prevent the collisions. The ADMM exhibits the best performance by keeping the vehicles at the desired safe distance. The ADMM-L, which represents the low-complexity version of the ADMM, avoids the collisions, however the desired distance is not always achieved.

Let us evaluate the Distance Error for each platoon vehicle k over time in seconds (s). This is defined as, the difference between the estimated position $p_k(t)$ and the desired position $p_k^*(t) = p_0(t) - k\Delta_{k,k-1}$, and it is measured in meters (m), i.e.,

$$\text{Distance Error} \triangleq |p_k(t) - (p_0(t) - k\Delta_{k,k-1})|\,(m), \quad (35)$$

where the technique with the smaller distance error for each vehicle indicates that the platoon follows the desired waypoints more accurately. We have considered the trajectory case for the leading vehicle with identity (ID) 86, from the Carla generated scenario of Fig. 2.

In Fig. 4 we present the results for the distance error and the vehicles speed. The error of the unconstrained MPC can be as large as 4.2 (m), while the error is much smaller for the constrained MPC cases. The low-complexity implementation, ADMM-L, reduces the error of the MPC to 2.1 (m), while

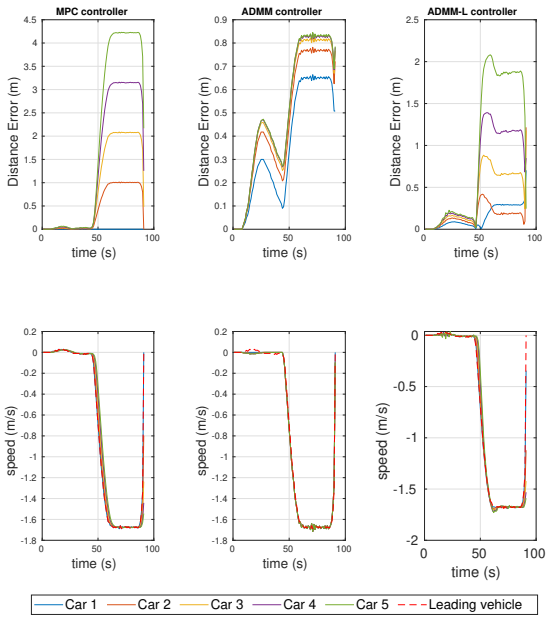

Fig. 4. Controllers performance evaluation for vehicle with ID 86.

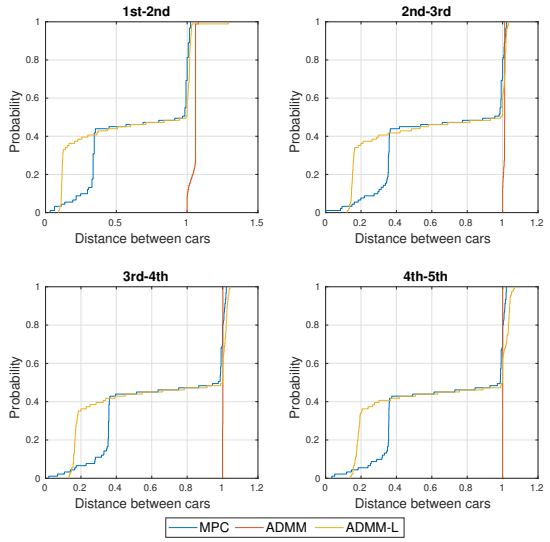

Fig. 5. CDF for the distance between successive vehicles.

the ADMM to 0.85 (m). In Fig. 5, we plot the CDF of the distance between two successive vehicles, when it is computed over the observed time frame. So for the ADMM case, the distance between all successive vehicles is over 1(m) with probability 1. For the unconstrained MPC and the low-complexity implementation ADMM-L, the probability for the distance between the successive vehicles to be smaller than 1 (m) is not zero. Specifically, it is 50% probable to have distance around 0.4 (m).

So far, we have considered that the controller update time is $T_c = 1s$. However, the update interval has significant impact on the performance of the controller. In particular, by decreasing the update interval, the unconstrained MPC can perform similarly to ADMM, while the performance of the ADMM-L also improves. However, this performance improvement is not always possible, due to the network latency and computational complexity issues.

VI. Conclusions

The proposed cooperative control techniques are able to avoid collisions under realistic scenarios, that have been generated via the open-source simulator CARLA. The low-complexity stochastic controller ADMM-L presents a promising approach, with much lower computational and communication overhead, however the distance error is still high. Further investigation of the stochastic MPC with the ADMM approach is left as a future work.

References

[1] P. Wang et al., "Platoon Cooperation in Cellular V2X Networks for 5G and Beyond," *IEEE Transactions on Wireless Communications*, vol. 18, no. 8, 2019.

[2] K. Xiong et al., "Recouping efficient safety distance in iov-enhanced transportation systems," in *IEEE International Conference on Communications (ICC)*, 2019.

[3] Y. Lin et al., "Comparison of deep reinforcement learning and model predictive control for adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, 2021.

[4] M. Rahmani-andebili et al., "Cooperative distributed energy scheduling for smart homes applying stochastic model predictive control," in *IEEE International Conference on Communications (ICC)*, 2017, pp. 1–6.

[5] P. Giselsson et al., "Accelerated gradient methods and dual decomposition in distributed model predictive control," *Automatica*, vol. 49, no. 3, pp. 829–833, 2013.

[6] A. Kozma, J. V. Frasch, and M. Diehl, "A distributed method for convex quadratic programming problems arising in optimal control of distributed systems," in *52nd IEEE Conference on Decision and Control*, 2013.

[7] R. Rostami, G. Costantini, and D. Görges, "ADMM-based distributed model predictive control: Primal and dual approaches," in *IEEE Conference on Decision and Control*, 2017, pp. 6598–6603.

[8] A. Dosovitskiy et al., "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16.

[9] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications*, 1st ed. Springer Publishing Company, 2007.

[10] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*, 1st ed. Springer Publishing Company, 2009.

[11] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," vol. 3, no. 1, p. 1–122, Jan. 2011.

[12] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," Mar. 2014.