2024 European Control Conference (ECC)
June 25-28, 2024. Stockholm, Sweden

Non-linear Distributed MPC Coordination of Autonomous Vehicles using Optimality Condition Decomposition

Marc Facerías[1], Vicenç Puig[2] and Alexandru Stancu[1,3]

Abstract— **In this paper, a novel non-linear distributed MPC coordination scheme for autonomous vehicles based on the Optimality Condition Decomposition (OCD) algorithm is proposed. As result, a local planner for each vehicle is obtained via the OCD application to the MPC coordination scheme is obtained for each vehicle, providing realistic paths to be executed in a collaborative manner. The proposed approach is able to determine paths that consider both individual goals and the environment so that agents can collaborate to safely navigate the studied environment. A case study in a ROS simulation environment is used to assess the validity of the proposed approach for real-time implementation.**

I. Introduction

In the last few years, the automotive sector has been advancing more and more towards automatising transport systems, with autonomous cars being few years away from a general public adoption. In the new context of highly connected autonomous devices it is expected to see a paradigm shift, from the roads being streams of independent agents competing with each other to reach their goal to a flow of interconnected entities that seek to accomplish their objectives through peer to peer collaboration. Autonomous driving, and thus collaborative driving, can not be considered an unique problem, as there exists a wide range of scenarios that involve different behaviours. Consequently, current research is focused on solving individual sub-problems, such as intersection management, assistive driving , or lane changes, among others.

This work focuses on treating the platooning problem in the domain of autonomous driving. We can differentiate two trends when studying collaborative navigation, depending on whether an optimisation problem is solved in a distributed or decentralised manner. On the one hand, decentralised techniques separate the systems via either estimations or relaxations of the coupling behaviours. On the other hand, distributed techniques rely on mathematical relaxations of the problem to solve it in a parallel iterative manner.

Regarding distributed approaches, several research techniques are being developed to deal with collaborative navigation. For instance in [1], the alternating direction method of multipliers (ADMM) algorithm is applied, being the main innovation with respect to other approaches the usage of a spline based characterisation of the environment. In similar lines of work, [2] presents a platooning ADMM algorithm with a leader-follower structure and evaluates its performance in Carla, a high-fidelity simulator for autonomous driving. A distributed collision avoidance approach is presented in [3], where simple agents modelled through a double integrator are coordinated using ADMM and a linear approximation of the collision avoidance constraints that allow solving the problem in a simpler manner. Similarly, [4] presented an ADMM coordination strategy that relies on separating lines for obstacle avoidance and spline-based trajectory characterisation, showing how the distributed algorithm improved the results of its centralised counterpart. Moreover, distributed coordination strategies allow system reconfiguration in an ad-hoc manner, preventing local failures to compromise the whole system. An example of this behaviour can be found in [5], where a novel online platoon reconfiguration technique is presented and compared to the current European standards, with an special focus on the physical limitations that need to be overcome to implement this family of algorithms. Following a different path, [6] proposes an ADMM optimal consensus controller applied to systems of particles. Interestingly, some lines of work do not rely on optimising a common performance metric to force a collaborative nature. For instance, in [7], a game theoretic approach based on Gauss-Seidel iterative method is proposed, allowing agents to decide whether to compete or collaborate, achieving a more human-like behaviour while improving their performance compared with a classic MPC. In a similar line of work [8] proposes a coordinated graphs and Max-Plus algorithm to decide high level actions that agents should perform to rearrange themselves in a lane-free environment.

Alternatively, decentralised algorithms switch the paradigm and decouple each subsystem by ignoring their interactions or estimating optimisation variables of the neighbouring agents. In other words, they effectively assume that global performance is subject to maximising the performance of each individual in an isolated manner, leading to a competitive behaviour which opposes the collaborative nature of other architectures. For instance, in [9] presents a decentralised solution for the navigation in road intersections where each agent acts individually and a second coordination layer gives priority to each of them in a centralised manner to minimise the cost related to its intersection. In another line of work, [10] presents a platooning algorithm with an special focus on the physical implementation of the network with a real fleet of

[1] M. Facerías and A. Stancu belong to the Faculty of Electrical Engineering, University of Manchester, The United Kingdom marc.faceriaspelegri@manchester.ac.uk

[2]V. Puig belongs to the Institute of Robotics, CSIC-UPC, Barcelona Spain vicenc.puig@upc.edu

[3]A. Stancu belongs to Manchester Robotics ltd.

This work has been partially funded by Manchester Robotics ltd. and by the Spanish State Research Agency (AEI) through the project SaCoAV (ref. MINECO PID2020-114244RB-I00).

trucks following a leader. Similarly, [11] presents an urban navigation network where agents communicate with devices installed in the streets that assign plans seeking to optimise the routes in a collaborative manner. Other approaches rely on estimating the results of coupled sub-problems, as presented in [12], where an MPC structure is used along with compatibility constraints to decouple vehicles in shared environments. In general, this family of strategies tend to add external devices to the system to avoid degraded solutions which may appear due to the simplification of coupled dynamics, as presented in [13].

In this paper, a novel non-linear distributed MPC coordination scheme for autonomous vehicles based on the Optimality Condition Decomposition (OCD) algorithm is proposed. As result, a local planner for each vehicle is obtained via the OCD application to the MPC coordination scheme is obtained for each vehicle, providing realistic paths to be executed in a collaborative manner. The proposed approach is able to determine paths that consider both individual goals and the environment so that agents can collaborate to safely navigate the studied environment. A case study in a ROS simulation environment is used to assess the validity of the proposed approach for real-time implementation.

The structure of the paper is the following: Section II presents the problem statement and the OCD approach. Section III shows the application of the proposed approach in the collaborative coordination of autonomous vehicles. Section IV presents the simulation results in the considered case study scenario. Finally, Section V draws the main conclusions and points several future research paths.

II. Problem statement and proposed approach

A. Problem statement

The collaborative planning problem can be defined as the search of suitable paths for a group of autonomous vehicles (agents) that allow navigating an environment while satisfying a set of constraints (as e.g. collisions avoidance or road margins). These constraints vary depending on the nature of the system, being in this specific scenario the physical limits of the agents and the safety margins between them. The main complication associated with this type of problems is that in order to find a suitable path for each vehicle we need to embed all the agents of the system in a single optimisation problem, which may lead to idle times too big for any real time application. Furthermore, solving a single optimisation problem implies that the system needs a central computing unit and a communication network powerful enough to ensure the propagation of the solutions and retrieving of all the individual agent data.

The inherent fragility and inefficiency of the centralised approach motivated the reformulation of the problem in a distributed manner. Even though there is still a performance bottleneck related with the latency of the network, we can solve N optimisation sub-problems locally while achieving an optimality degree close to the centralised case through the propagation of local solutions and solving the sub-problems iteratively until convergence. This approaches tend to present algorithm with a high computational load. Thus, an alternative can be found in decentralised optimization approaches, which are a relaxed version of the original problem that either ignore the coupling between subsystems or rely on an approximation of their local variables, leading to a set of N independent problems. Such structure is specially interesting for multi-agent systems, as they are physically distributed systems where using a centralised optimisation algorithm would imply a huge computational burden.

B. Lagrangian relaxation methods

Lagrangian relaxation methods are techniques applied to optimisation problems that aim to solve a simplified relaxed version of the problem through the application of the Lagrange multipliers. In order to illustrate this concept, we will introduce an arbitrary optimisation problem in

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & \\ & h(x) \leq 0 \end{aligned} \tag{1}$$

We aim to relax Eq.(1) by adding the constraint $h(x)$ to the cost function through a term λ, known as a Lagrange multiplier, leading to $\min_x f(x) + \lambda h(x)$.

It can be observed that the problem is enforced to minimise the term $\lambda h(x)$, which for an appropriate positive value of λ is equivalent to treating $h(x)$ as a constraint. The same principle is exploited by several decomposition techniques, which take advantage of embedding certain coupling constraints as part of the cost function to solve the complete problem in a distributed manner. Even though there is a vast literature regarding the solution of this distributed problems, they can be classified as either Lagrangian decomposition or decentralised solution of the KKT conditions for local optimality [14]. Two of the most popular techniques are the ADMM and OCD, which belong to the first and second approaches, respectively. It can be seen that among the literature presented in the introduction, ADMM is the most extended technique and, even though OCD is similar in essence, the latter presents some advantages that allow a system topology more convenient for the problem treated in this paper. As presented in [15], the main advantage of OCD over ADMM is that the latter does require a central coordination unit to perform the dual variable update, while OCD only requires communication with peers that present coupling with each other. This key difference in the system structure not only avoids relying on a specific agent but opens the door to the implementation of security features and dynamic agent reallocation in a simpler manner. In particular, ADMM requires a centralised update method, while OCD updates λ in a peer-to-peer basis. In this way, communication is only strictly necessary among vehicles sharing a local constraint.

As mentioned before, OCD is a technique based on the Lagrangian decomposition method, proposed in [16], that aims to divide a complex problem into simpler sub-problems

by separating the coupled constraints. In this context, consider an optimisation problem with two groups of variables $x = [x_1, x_2]$, as follows

$$\begin{aligned} \min_{x_1,x_2} \quad & f(x_1, x_2) \\ \text{s.t.} \quad & \\ & g_1(x_1) \leq 0 \\ & g_2(x_2) \leq 0 \\ & h_1(x_1, x_2) \leq 0 \\ & h_2(x_1, x_2) \leq 0 \end{aligned} \tag{2}$$

In this example, we consider g_i separable constraints, which involve only one subset of states, while h_i are constraints that do not allow a direct decomposition. This can be solved by fixing a certain subset of variables, leading to two problems of the following shape

$$\begin{aligned} \min_{x_1} \quad & f(x_1, \hat{x}_2)) + \lambda_2 h_2(x_1, \hat{x}_2) \\ \text{s.t.} \quad & \\ & g_1(x_1) \leq 0 \\ & h_1(x_1, \hat{x}_2) \leq 0 \end{aligned} \tag{3}$$

For clarity purposes we will not write the second half of the distributed problem, as it can be obtained by switching indexes 1 and 2. By iterating through the update procedure in Algorithm 1, an approximation of the optimal solution is computed. It is worth to note that any update policy of α could be applied to improve the convergence rate of λ, but for simplicity purposes, a constant positive update rate has been used in this formulation.

Algorithm 1 Optimal Condition Decomposition

while not convergence **do**
  Collect current x_i and set $\hat{x}_i = x_i$
  Update $\lambda_n = \lambda_n + \alpha h(x_i^k, x_n^k)$
  Compute a step of i_{th} optimisation problem
  Forward x_n thorough the system
end while

C. Non Linear DMPC formulation

In order to propose a distributed control problem, its centralised counterpart is going to be presented, which in this case is equivalent to consider all the vehicles in the road and their interactions as a single system. The MPC structure is specially suitable as an optimisation problem as it provides reasonable estimates for successive problem while allowing a precise agent representation. This is specially interesting in its NL case, as by providing a good initial search point we can limit the number of search steps and safely assume that the solution will be located close to a reasonable system state.

$$\begin{aligned} \min_{\Delta u_n^k} \quad & \mathrm{J} = \sum_{k=0}^{H} \sum_{n=0}^{N} (J_n^k(x_n^k)) \\ \text{s.t.} \quad & \forall k \in [0, H], \forall n \in [1, N] \\ & x_n^{k+1} = f(x_n^k, u_n^k) \\ & u_n^k = u_n^{k-1} + \Delta u_n^k \\ & u_n^k \in [\underline{u_n^k}, \overline{u_n^k}] \\ & e_{y_n}^k \in [\underline{e_{y_n}^k}, \overline{e_{y_n}^k}] \\ & \forall k \in [0, H], \forall n \in [1, N], \forall j \in [1, N] \ j \neq n \\ & g(p_n^k, p_j^k) \equiv \|p_n^k, p_j^k\|_2 \leq D_{sf} \end{aligned} \tag{4}$$

where $J_n^k(x_n^k)$ has the following quadratic shape:

$$J_n^k(x_n^k) = xQx + \Delta u Q_u \Delta u + q_x x + q_u u + q_\sigma \sigma \tag{5}$$

being x is the vehicle state vector, u the input vector, $e_{y_i}^k$ the lateral error and σ a set of slack variables. Q_x, Q_u, q_x, q_u and q_σ are weighting factors applied to each variable, while H and N refer to the planning horizon and number of agents in the fleet, respectively. Obstacle avoidance constraints are represented by the term g and formulated using the Euclidean distance, where D_{sf} is the minimum allowed security distance. Finally, the vehicle model is an arbitrary function $f(x_n^k, u_n^k)$, which will be particularised in further sections of this paper. Applying OCD to Eq. (4) leads to N subsystems of varying shape depending on their communication topology. In order to exemplify their structure we are going to present a simplified case with two agents a_1 and a_2. Eq. (6) refers to a_1 while in Eq.(7) refers to a_2.

$$\begin{aligned} \min_{u_k} \quad & \mathrm{J} = \sum_{k=0}^{H} (J_1^k(x_1^k) + J_2^k(\hat{x}_2^k)) \\ \text{s.t.} \quad & \forall k \in [0, H] \\ & x_1^{k+1} = f(x_1^k, u_1^k) \\ & u_1^k = u_1^{k-1} + \Delta u_1 \\ & u_1^k \in [\underline{u_1^k}, \overline{u_1^k}] \\ & e_{y_1}^k \in [\underline{e_{y_1}^k}, \overline{e_{y_1}^k}] \\ & g(p_1^k, \hat{p}_2^k) \end{aligned} \tag{6}$$

$$\begin{aligned} \min_{\Delta u_n^k} \quad & \mathrm{J} = \sum_{k=0}^{H} (J_2^k(x_2^k) + J_1^k(\hat{x}_1^k)) + \lambda_{1,2}^k g(p_2^k, \hat{p}_1^k) \\ \text{s.t.} \quad & \forall k \in [0, H] \\ & x_2^{k+1} = f(x_2^k, u_2^k) \\ & u_2^k = u_2^{k-1} + \Delta u_2^k \\ & u_2^k \in [\underline{u_2^k}, \overline{u_2^k}] \\ & e_{y_2}^k \in [\underline{e_{y_2}^k}, \overline{e_{y_2}^k}] \end{aligned} \tag{7}$$

In this minimal example two agents coupled through a common obstacle avoidance constraint are decoupled through the addition of a Langrange multiplier to one of the subproblems. In this way, the optimal solution can be found

via an iterative approach without the need of performing a centralised optimisation. Note that terms denoted as $\hat{\cdot}$ represent information about the fixed subset, e.g the previous solution of coupled sub-problems, of the term involved. Such fixed value is updated online in the intermediate steps of the optimisation and thus its value is shared through the subsystems accessing it.

As it can be seen, applying this formulation as presented in Eq. (4) leads to duplicated constraints. Specifically, when dealing with an arbitrary pair i and j, two equivalent constraints are generated, as $g^{k,i,j} \equiv g^{k,j,i}$. In order to prevent it, we can apply the following heuristics:

- if $i < j$, we will consider i as the ego vehicle and add the constraint $g(p_i^k, \hat{p}_j^k)$ to its optimisation problem. Accordingly, $\lambda_{1,2}^h g(\hat{p}_i^k, p_j^k)$ is added to the neighbouring agent j.
- if $i > j$, we will consider j as the ego, treating the constraints in the same manner already presented, but inverting the roles of i and j.

Additionally, slack variables σ_u, σ_{e_y}, σ_g, σ_h were added to the system and introduced in the cost function with appropriate weighting factors. These variables allow the constraints associated to them to be violated, dramatically increasing the cost function value. Thus, this will only happen in situations that otherwise would lead to system deadlocks, e.g a bottleneck where the only way to advance is by being closer than D_{sf}.

III. Coordination of autonomous vehicles

The proposed approach presented in previous section will be used to deal with an environment where a group of N agents enter a highway and rearrange themselves into a platooning formation. The system is rewarded to traverse the environment as close as possible to the maximum allowed velocity while keeping a safety distance between agents and respecting both the physical limits of the vehicle and the highway. Each vehicle is modelled after a scaled car and we consider that all members of the platoon have the same global path G_p, defined by a set of arc segments of length l and curvature k, being $G_p = [l_0, k_0, l_1, k_1, ..., l_N, k_N]$. Additionally, we consider no lanes present in the track, thus agents can position themselves in any formation as long as they remain within the road limits and keep inter-vehicular safety distance. Finally, each agent is assumed to follow local plans perfectly. While this assumption is not needed to assess the viability of the algorithm, it simplifies the experimental process and decouples possible control errors from the plans generated, which by design will be suitable for the agents. It is worth to mention that in strict theory, we could use the control actions generated by the MPC directly as the input of each individual in the platoon. However, due to the latency of the algorithm this would lead to poor performance.

A. Non-linear vehicle model

In this work, we opted to use a dual model, involving both Frenet frame coordinates and Cartesian coordinates.

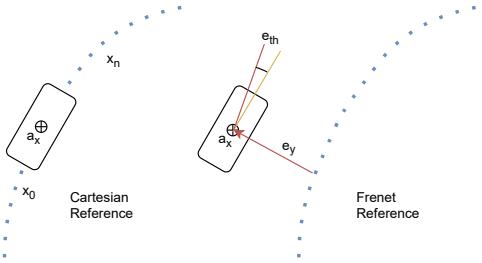

Fig. 1. Two equivalent paths represented in both Cartesian and Frenet frames

Both formulations have particular strengths. However, models purely based on Frenet coordinates lack representing the vehicle physical insights necessary in autonomous driving, which motives proposing a mixed model. In this application, the variables associated to a Frenet frame are used to embed of the reference signal into the model via following a curvilinear reference defined by a curvature k and an arc length s at which the agent is located. On the other hand, we will add the terms related with the Cartesian positions and velocities, as they are adequate for collision checking, which is strongly related to the geometric disposition of the system. Furthermore, having access to v_x, v_y and w is critical from a performance point of view, as it allows adding comfort constraints. The Cartesian variables of the car model have been derived from a bicycle model. A visual representation of both Frenet and Cartesian frames can be seen in Figure 2. The continuous-time non-linear model is described by the following set of equations in the Cartesian coordinates

$$\begin{aligned}
\dot{v}_x &= a + \frac{-F_{yf}\sin\delta - \mu g}{m} + \omega v_y \\
\dot{v}_y &= \frac{F_{yf}\cos\delta + F_{yr}}{m} - \omega v_x \\
\dot{\omega} &= \frac{F_{yf} l_f \cos\delta - F_{yr} l_r}{I} \\
\dot{x} &= v_x\cos\theta - v_y\sin\theta \\
\dot{y} &= v_x\sin\theta + v_y\cos\theta \\
\dot{\theta} &= \omega
\end{aligned} \tag{8}$$

where $F_{yf} = C_f\alpha_f$, $F_{yr} = C_r\alpha_r$, $\alpha_f = \delta - \tan^{-1}\left(\frac{v_y}{v_x} - \frac{l_f\omega}{v_x}\right)$ and $\alpha_r = -\tan^{-1}\left(\frac{v_y}{v_x} + \frac{l_r\omega}{v_x}\right)$. Cartesian state variables v_x, v_y and ω represent the body frame velocities, i.e. linear in x, linear in y and angular velocities, respectively. Accordingly, the position is defined by x, y, θ coordinates with respect to a global reference frame. The variables δ and a are the steering angle at the front wheels and the longitudinal acceleration vector on the rear wheels, respectively, and represent the inputs of the system. F_{yf} and F_{yr} are the lateral forces produced in front and rear tires, respectively. Front and rear slip angles are represented as α_f and α_r, respectively, and C_f and C_r are the front and rear tire stiffness coefficients. m and I represent the vehicle mass and inertia and l_f and l_r are the distances from the vehicle

TABLE I

DYNAMIC MODEL PARAMETERS OF THE VEHICLE

Parameter	Value	Parameter	Value
l_f	0.125 m	l_r	0.125 m
m	1.98 kg	I	0.06 $kg\ m^2$
C_f	60	C_r	60
μ	0.05	g	9.81 $\frac{m}{s^2}$

center of mass to the front and rear wheel axes, respectively.

On the other hand, the vehicle dynamics can be represented in the Frenet frame coordinate set as follows

$$\begin{aligned} \dot{s} &= \frac{v_x cos\theta_e - v_y sin\theta_e}{1 - e_y k} \\ \dot{e}_y &= v_x \sin\theta_e + v_y \cos\theta_e \\ \dot{\theta}_e &= \omega - \frac{v_x \cos\theta_e - v_y \sin\theta_e}{1 - e_y \kappa}\kappa \end{aligned} \tag{9}$$

where Frenet state variables are denoted as s, e_y and θ_e and represent the position of the vehicle along the curve, the lateral error and the orientation error, respectively. Finally this model is particularised for a small scale vehicle by substituting the constants present in Table I.

B. Environment modelling

The reference to be followed by the platoon is modelled through a set of curvilinear segments of constant curvature k and a given length l. This is embedded into the model by the time-varying constant k and the state variables s, e_y and θ_e. In order to update the parameter k, a greedy search is performed through the set of curves seeking the one that encloses the current position of the vehicle, taking into account the Frenet frame state variables. By considering the previous position and taking into account that each segment has a constant curvature, we can perform this search locally without having a major impact on the algorithm computational time. Note that the correctness of k is subject to the precision of the state variable s which in a real scenario will depend on both the model quality and the onboard localisation module. In this paper, it is assumed that the reference curves are computed offline, resembling what could be a segment of a highway.

IV. RESULTS

The effectiveness of the algorithm proposed in Section II-C is going to be assessed by applying it under the conditions presented in Section III. The results of this section were generated using CASADI and IPOPT as a non-linear solver. The code was deployed in a ROS network where each agent acts as an individual computing thread, resembling a real network of autonomous agents. All the computing times presented in this section do not include communication delays. However, they should be taken into consideration for any practical implementation. Finally, all tests were carried out in a consumer laptop with 16Gb of RAM, an i7-13700H CPU and no specific hardware acceleration. It

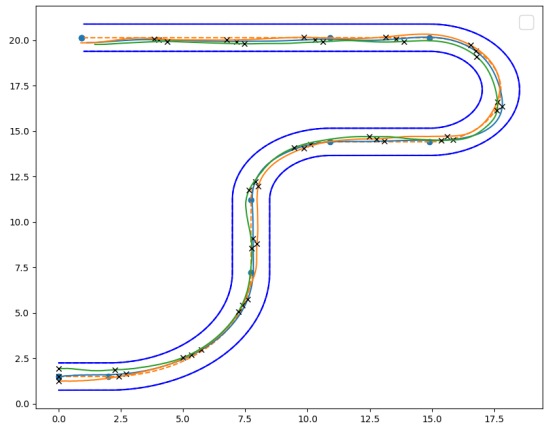

Fig. 2. Path traversed

is worth to note that solving the non-linear formulation is a NP-Hard problem. Thus, results are susceptible to the experimental conditions. However, per definition, OCD will improve inaccurate solutions iteratively which can mitigate the workload by using non-optimal solutions of the sub-problems. The metrics used for the sake of the comparison are the 2D generated path, the safety limits, the look-ahead distance and the computational time per iteration. In order to assess performance, we will use v_x and both inter-vehicular and look-ahead distances. This selection is motivated by the fact that we aim to know how fast are we going to reach our goal and how safe is the trajectory going to be. It is trivial to see that higher linear velocities will lead to shorter trajectories. The selection of the safety metrics aims to represent how well can we react to unforeseen scenarios. Specifically, how far away can a detectable obstacle be to be considered in the plans and what margin do we have in the face of events such as an agent losing control of the vehicle or an animal crossing the road. Firstly, as we can see in Figure 3, agents maintain a velocity around the road limits, which in this case is set to 3 m/s. A strict velocity profile may be enforced through modifying the cost function, but in this case it was treated as a soft objective. With the proposed set up the algorithm look-ahead distance is 5 s, with a prediction horizon of 20 steps, providing a reasonable tradeoff between length and performance. In terms of platoon performance, in Figure 2 it can be seen how vehicles rearrange into a formation that satisfies the minimum distance constraints, and advance along the center of track, reducing e_y. Note that this particular behaviour is enforced through the design of the cost function, as there is no other incentive of minimising the lateral error other than mimicking the expected human behaviour. Consequently, other arrangements could be incentivised through an appropriate cost function. Finally, in terms of computational time we can see in Fig. 4 how the overall algorithm can be executed in under 500ms, which makes it feasible to scale it into a real device. As

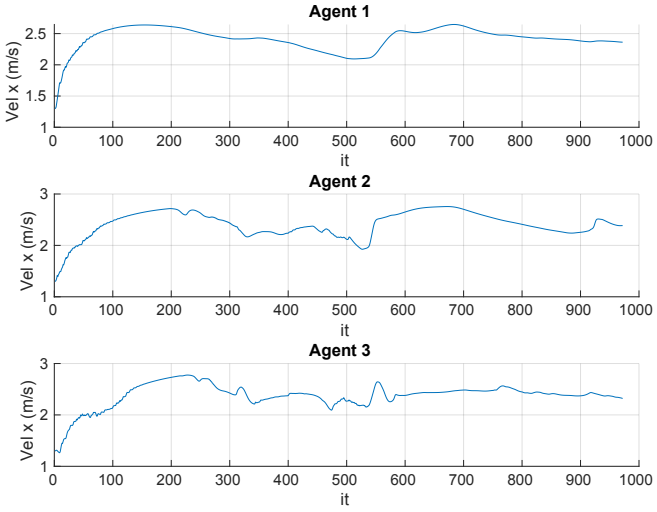

Fig. 3. Velocity profile

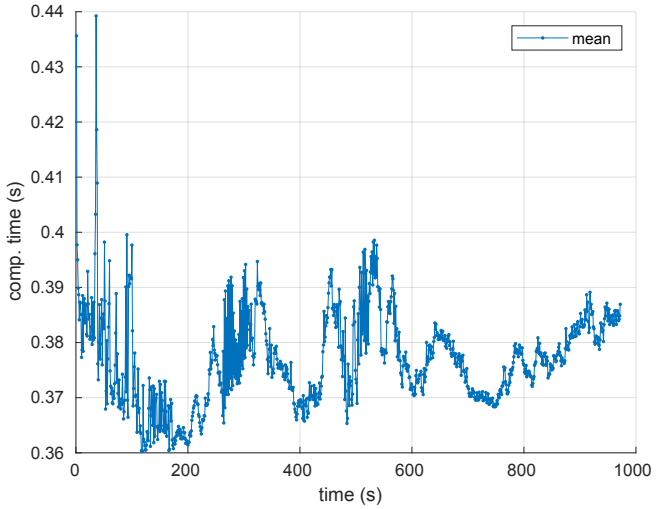

Fig. 4. Average computational time per agent

a clarification, Fig. 4 presents the average optimisation time per agent is presented, in the particular case of 3 agents each of them would locally solve its local problem derived from Eqs. (6) (7). Remarkably, due to the equivalent complexity of each sub-problem individual optimisation times where similar among agents in the fleet. Note that the first iteration is remarkably slower due to the computation of Hessian and Jacobian matrices used by IPOPT with its off-the-shelf configuration to solve the optimisation problem. However, this has no impact on the rest of the iterations as the solver is already warm started with suitable initial conditions, which are reasonably close to the expected optimal values.

V. Conclusions

In this work, a novel non-linear distributed MPC coordination scheme for autonomous vehicles based on the Optimality Condition Decomposition (OCD) algorithm is proposed and tested it is viability in a shared environment. Furthermore, tests were implemented in a distributed manner as a ROS network, mimicking the physical layout of a platoon of autonomous vehicles, with agents reaching some sort of formation and stay in it while traversing the environment under some safety conditions, in what can be understand to be non-competitive behaviour. As future work, we aim to expand this research by apply variations of the planner in a cascade fashion so that collaborative decisions, which tend to have an slower occurrence rate can be taken in a higher level, less constraining both in terms of model accuracy and computational time, while the local plans are computed by a faster distributed algorithm with an special focus on the physical distribution of the agents and its safety.

References

[1] R. V. Parys and G. Pipeleers, "Distributed mpc for multi-vehicle systems moving in formation," *Robotics and Autonomous Systems*, vol. 97, pp. 144–152, 11 2017.

[2] E. Vlachos and A. S. Lalos, "Admm-based cooperative control for platooning of connected and autonomous vehicles," vol. 2022-May. Institute of Electrical and Electronics Engineers Inc., 2022, pp. 4242–4247.

[3] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully decentralized admm for coordination and collision avoidance," in *2018 European Control Conference (ECC)*, 2018, pp. 825–830.

[4] R. Van Parys and G. Pipeleers, "Online distributed motion planning for multi-vehicle systems," in *2016 European Control Conference (ECC)*. IEEE, 2016, pp. 1580–1585.

[5] S. Jeong, Y. Baek, and S. H. Son, "Distributed urban platooning towards high flexibility, adaptability, and stability," *Sensors*, vol. 21, 4 2021.

[6] Q. Wang, Z. Duan, and J. Wang, "Distributed optimal consensus control algorithm for continuous-time multi-agent systems," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 1, pp. 102–106, 2020.

[7] M. Wang, S. P. Hoogendoorn, W. Daamen, B. van Arem, and R. Happee, "Game theoretic approach for predictive lane-changing and car-following control," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 73–92, 9 2015.

[8] D. Troullinos, G. Chalkiadakis, I. Papamichail, and M. Papageorgiou, "Collaborative multiagent decision making for lane-free autonomous driving," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, 2021, pp. 1335–1343.

[9] R. Hult, M. Zanon, S. Gros, and P. Falcone, "Optimal coordination of automated vehicles at intersections: Theory and experiments," *IEEE Transactions on Control Systems Technology*, vol. 27, pp. 2510–2525, 11 2019.

[10] Y. Lee, T. Ahn, C. Lee, S. Kim, and K. Park, "A novel path planning algorithm for truck platooning using v2v communication," *Sensors (Switzerland)*, vol. 20, pp. 1–27, 12 2020.

[11] Y. Regragui and N. Moussa, "A real-time path planning for reducing vehicles traveling time in cooperative-intelligent transportation systems," *Simulation Modelling Practice and Theory*, vol. 123, 2 2023.

[12] F. Mohseni, E. Frisk, and L. Nielsen, "Distributed cooperative mpc for autonomous driving in different traffic scenarios," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 299–309, 2020.

[13] L. Guillaume, "Fast unfolding of communities in large networks," *Journal Statistical Mechanics: Theory and Experiment*, vol. 10, p. P1008, 2008.

[14] A. Kargarian, J. Mohammadi, J. Guo, S. Chakrabarti, M. Barati, G. Hug, S. Kar, and R. Baldick, "Toward distributed/decentralized dc optimal power flow implementation in future electric power systems," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 2574–2594, 2016.

[15] P. Segovia, V. Puig, E. Duviella, and L. Etienne, "Distributed model predictive control using optimality condition decomposition and community detection," *Journal of Process Control*, vol. 99, pp. 54–68, 3 2021.

[16] A. J. Conejo, E. Castillo, R. Minguez, and R. Garcia-Bertrand, *Decomposition techniques in mathematical programming: engineering and science applications*. Springer Science & Business Media, 2006.