Flexible MPC-based Conflict Resolution Using Online Adaptive ADMM

Jerry An*,†, Giulia Giordano‡ and Changliu Liu*

Abstract— **Decentralized conflict resolution for autonomous vehicles is needed in many places where a centralized method is not feasible, e.g., parking lots, rural roads, merge lanes, etc. However, existing methods generally do not fully utilize optimization in decentralized conflict resolution. We propose a decentralized conflict resolution method for autonomous vehicles based on a novel extension to the Alternating Directions Method of Multipliers (ADMM), called Online Adaptive ADMM (OA-ADMM), and on Model Predictive Control (MPC). OA-ADMM is tailored to online systems, where fast and adaptive real-time optimization is crucial, and allows the use of safety information about the physical system to improve safety in real-time control. We prove convergence in the static case and give requirements for online convergence. Combining OA-ADMM and MPC allows for robust decentralized motion planning and control that seamlessly integrates decentralized conflict resolution. The effectiveness of our proposed method is shown through simulations in CARLA, an open-source vehicle simulator, resulting in a reduction of 47.93% in mean added delay compared with the next best method.**

I. Introduction

When designing fully autonomous vehicles, reducing traffic congestion is a crucial goal. Intersections are a major contributor to traffic delays and accidents, hence autonomous vehicles need to be equipped to deal with them efficiently [1]. Since intersections often lack the infrastructure required to centrally resolve the conflicts [2], autonomous vehicles must be able to resolve conflicts without any external infrastructure. Navigating unmanaged intersections using decentralized policies is challenging due to the risk of deadlocks or accidents: communication and conflict resolution protocols among autonomous vehicles are needed.

Traditional approaches for conflict resolution include heuristic intersection protocols using various priority policies [3], [4], which allow real-time adjustments of the priorities and can utilize constrained optimal control to improve their performance [5]. Online distributed motion planning for the formation control of multi-agent systems, based on a single-iteration receding horizon approach, is proposed in [6], while [7] also considers inter-vehicle collision avoidance through the use of separating hyperplanes. The ADMM (Alternating Directions Method of Multipliers) variant, introduced in [8] for autonomous vessels, utilizes a central coordinator and claims to improve the convergence rate by iteratively adding approximated collision avoidance constraint. A method similar to that of [6], [7] is introduced in [9], utilizing a linearized

* Robotics Institute, Carnegie Mellon University, Pittsburgh, USA, jerryan.x@gmail.com, cliu6@andrew.cmu.edu.

† Delft Center for Systems and Control, Delft University of Technology, Delft, the Netherlands, jerryan.x@gmail.com.

‡ Department of Industrial Engineering, University of Trento, Italy, giulia.giordano@unitn.it.

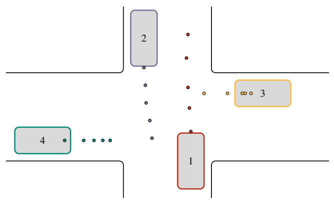


Fig. 1: Example of conflict resolution using OA-ADMM and MPC. The colored dots represent the MPC trajectories of each vehicle.

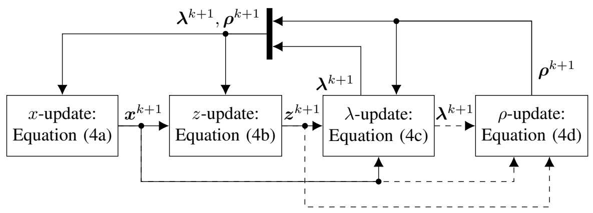
collision avoidance constraint and incorporating deadlock-protection. The nonlinear MPC-based approach in [10] reduces the need for linearized constraints. The distributed MPC approach in [11] incorporates the residual balancing method from [12], while [13] utilizes distributed trajectory optimization based on a decomposition technique that avoids communication between agents until convergence.

The decentralized protocols in [3], [4], [5] require prior knowledge of the environment and cannot adjust the vehicles' trajectories. The online distributed motion planning techniques proposed in [6], [7], [9], [10] do not allow deadlock resolution or adaptive penalty parameters, making safety during real-time implementation questionable. The distributed MPC approach from [11] does not utilize the adaptive penalty parameter to improve the system safety. The distributed trajectory optimization technique from [13] requires performing the optimization steps until convergence and the decomposition method has no convergence proofs, making the method unsuitable for autonomous vehicles.

We propose a novel MPC-based method for decentralized conflict resolution that relies on our proposed Online Adaptive ADMM (OA-ADMM) algorithm, which improves efficiency because it allows trajectory deviation. When ADMM cannot be performed until convergence due to the required control frequency, feasibility results can be poor to the extent that safety is an issue, in addition convergence can also be poor when results from the previous control step are not utilized. OA-ADMM solves this using two user-designed functions: the similarity function, which is a forgetting factor between two time steps of the online system, and the adaptation function, which adjusts the penalty parameters between updates. In our application we design the similarity function and adaptation function based on the physical safety of the system, increasing both values when distance between planned trajectories decrease. Our main contributions are:

- Unification of online application of ADMM under one framework (OA-ADMM)[1].

[1]The extended version of this paper, including the appendix, can be found at https://arxiv.org/abs/2103.14118.



$\lambda^{k+1}, \rho^{k+1}$
λ^{k+1}
ρ^{k+1}
x-update: Equation (4a)
x^{k+1}
z-update: Equation (4b)
z^{k+1}
λ-update: Equation (4c)
λ^{k+1}
ρ-update: Equation (4d)

Fig. 2: Diagram of OA-ADMM steps, dashed arrows resemble potential, but not necessary, inputs.

- Proposal of a physical safety based adaptation function to improve online robustness and safety.
- Application of OA-ADMM and MPC, achieving trajectory deviation in decentralized conflict resolution, with fewer requirement in terms of prior knowledge.

II. Problem Formulation

When approaching conflict resolution as a centralized optimization problem, it takes the form:

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \sum_{i=1}^{M} J_i(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x}_i \in \mathcal{X}_i^f, \\ & d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \forall j \in \mathcal{N}_i, \quad \forall i \in \{1, ..., N\}, \end{aligned} \tag{1}$$

where $\boldsymbol{x}_i$ is the state vector for agent i (e.g. for a kinematic bicycle model $\boldsymbol{x}_i = [\mathrm{x}, \mathrm{y}, v, a, \beta]^\top$, with x,y the coordinates in the local coordinate frame, a the input acceleration, and β the steering angle); $\mathcal{X}_i^f$ is the feasible set for $\boldsymbol{x}_i$ including all trajectories that adhere to the system dynamics, input constraint, and environmental collision avoidance constraints; $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the distance function between two capsules, and $\mathcal{N}_i$ is the set of vehicles neighboring i. A capsule is a line segment of length l inflated with a radius r.

The optimization problem (1) is difficult to solve, being nonlinear and nonconvex, and the constraints couple the states of more agents. For real time applications, a fast (<100ms) solution is required, making a centralized approach unpractical as the problem does not scale well. Due to the coupling in the constraints and objective function, primal decomposition cannot be applied to problem (1), which is shown to be NP-hard in [14]. For real-time optimization-based conflict resolution, problem (1) can be reformulated into the MPC-based finite horizon form, with state vector $\boldsymbol{x} \in \mathbb{R}^{N_x}$, where N_x is the length of the finite horizon time the amount of states per time step. Also, to be able to apply our proposed Online Adaptive ADMM approach to problem (1), we need to reformulate it as a general ADMM problem:

$$\min_{\boldsymbol{x},\boldsymbol{z}} \quad f(\boldsymbol{x}) + g(\boldsymbol{z}) \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c}, \tag{2}$$

where $f(\boldsymbol{x})$ and $g(\boldsymbol{z})$ are convex functions, $\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{z} \in \mathbb{R}^m, \boldsymbol{A} \in \mathbb{R}^{p \times n}, \boldsymbol{B} \in \mathbb{R}^{p \times m}$, and $\boldsymbol{c} \in \mathbb{R}^p$.

III. Online Adaptive ADMM (OA-ADMM)

We propose a novel ADMM-based method, OA-ADMM, tailored to problems that require the ability to adapt constraint feasibility in real-time and for which conventional optimization methods cannot achieve convergence at the desired control frequency. To be suitable for online optimization, OA-ADMM applies an ADMM-based strategy that yields admissible online results. Also, an adaptive penalty parameter ρ is designed to allow prioritization of the constraint violations in online results. As we will see, OA-ADMM guarantees improved robustness through the user-designed adaptation function ϕ and similarity function μ, along with the vectorization of the penalty parameter ρ.

Similar to ADMM, the coupled constraints are integrated into an augmented Lagrangian to separate the problem, i.e.

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + g(\boldsymbol{z}) + \boldsymbol{\lambda}^\top(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) \\ + \frac{1}{2}\|\boldsymbol{R}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c})\|_2^2. \end{aligned} \tag{3}$$

where $\boldsymbol{R} \in \mathbb{R}^{p \times p}$ is a diagonal matrix with $\mathrm{diag}(\boldsymbol{R}) = \boldsymbol{\rho}^{\circ \frac{1}{2}}$. The operator denoted by $(\cdot)^\circ$ is the *Hadamard power* (or element-wise power). Using a penalty vector $\boldsymbol{\rho}$ we can adjust the penalty for each element of the primal residual $\boldsymbol{r}$, where the primal residual is $\boldsymbol{r}^{k+1} = \boldsymbol{A}\boldsymbol{x}^{k+1} + \boldsymbol{B}\boldsymbol{z}^{k+1} - \boldsymbol{c}$. The dual residual is $\boldsymbol{s}^{k+1} = \boldsymbol{A}^\top \boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)$.

OA-ADMM requires two optimization steps, a Lagrangian multiplier update, and a ρ update:

$$\boldsymbol{x}^{k+1} := \arg\min_{\boldsymbol{x}} \mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k), \tag{4a}$$

$$\boldsymbol{z}^{k+1} := \arg\min_{\boldsymbol{z}} \mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{x}^{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}^k, \boldsymbol{\rho}^k), \tag{4b}$$

$$\boldsymbol{\lambda}^{k+1} := \mu(\cdot)\boldsymbol{\lambda}^k + \boldsymbol{\rho}^k \circ \boldsymbol{r}^{k+1}, \tag{4c}$$

$$\boldsymbol{\rho}^{k+1} := \phi(\cdot), \tag{4d}$$

where ϕ is the adaptation function (see Section IV-A) and μ is the similarity function (see Section IV-B). The structure of OA-ADMM is visualized in Figure 2.

We now provide convergence results and proofs. In the static case, with the problem assumed to be time-independent, using a similar approach to [15, Appendix A] we prove that the OA-ADMM scheme converges as $k \to \infty$.

To prove convergence, we assume that the original Lagrangian has a saddle point at $(\boldsymbol{x}^\star, \boldsymbol{z}^\star, \boldsymbol{\lambda}^\star)$ and we propose a candidate Lyapunov function V, such that $V \geq 0$ and $V = 0$ only at the saddle point:

$$V^k = \|\boldsymbol{R}^{\circ -1}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^\star)\|_2^2 + \|\boldsymbol{R}\boldsymbol{B}(\boldsymbol{z}^k - \boldsymbol{z}^\star)\|_2^2, \tag{5}$$

We also state some preliminary lemmas.

Lemma 1 (Converging penalty parameter)**.** *For ADMM to convergence with an adaptive penalty vector $\boldsymbol{\rho}$, it is necessary that $\boldsymbol{\rho}^k \to \boldsymbol{\rho}^\star$ as $k \to \infty$, and that all elements of $\boldsymbol{\rho}$ are positive.*

Convergence for an adaptive penalty parameter is proven in [12] when ρ converges to a certain $\rho^\star$. The requirement for all elements of $\boldsymbol{\rho}$ to be positive is given in [15, Section 3.4.2]. The following lemmas have been proven for ADMM with a conventional penalty parameter in [15, Appendix A] and can be easily extended to the case when the parameter is replaced by a vector (which leads to small changes and the need of using the Hadamard operator).

Lemma 2 (Objective suboptimality bounds)**.** *The suboptimality of the objective function p at step $k+1$, i.e. the difference between p^{k+1} and the saddle point $p^\star$, is bounded as:* $-\boldsymbol{\lambda}^{\star\top}\boldsymbol{r}^{k+1} \leq p^{k+1} - p^\star \leq -\boldsymbol{\lambda}^{k+1\top}\boldsymbol{r}^{k+1} - \left(\boldsymbol{\rho}^k \circ \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\right)^\top \left(-\boldsymbol{r}^{k+1} + \boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^\star)\right)$

Lemma 3 (Lyapunov decrease)**.** *The Lyapunov function V in* (5) *decreases at each iteration as:*

$$V^{k+1} \leq V^k - \|\boldsymbol{R}^k\boldsymbol{r}^{k+1}\|_2^2 - \|\boldsymbol{R}^k\boldsymbol{B}(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k)\|_2^2. \quad (6)$$

The convergence results for the static case can be summarized in the following theorem.

Theorem 1. *When applying the OA-ADMM algorithm in* (4)*, given closed, proper, and convex (See [15, Section 3.2]) functions f and g, a saddle point $p^\star$ in the Lagrangian $\mathcal{L}$, and a converging $\boldsymbol{\rho}^k$, the primal and dual residuals converge to zero, $\boldsymbol{r}^k \to 0$ and $\boldsymbol{s}^k \to 0$, and the objective function value converges to its saddle point, $p^k \to p^\star$.*

For proof, see Appendix I[1]. To prove online convergence, we analyze the change of the system compared with the convergence of OA-ADMM. We distinguish between the OA-ADMM iteration parameter k and the real time time step t: $\boldsymbol{x}^\star(t)$ is $\boldsymbol{x}$ at the saddle point for time t, whereas $\boldsymbol{x}^k(t)$ is $\boldsymbol{x}$ at iteration k at time step t.

Whilst the static convergence given in Theorem 1 proves residual convergence and objective convergence as $k \to \infty$, this result does not directly extend to the online case: for online systems, $\boldsymbol{x}^\star(t) = \boldsymbol{x}^\star(t+\delta t)$ cannot be assumed unless $\delta \to 0$, i.e. the control time step is very small. The change in optimum may be larger than the rate of convergence of OA-ADMM, which would result in each iteration converging towards its optimum, whilst never reaching the optimum at the next time step. We therefore define online convergence to be that OA-ADMM can always converge towards the online optimum as $k, t \to \infty$. This convergence requirement can be written as $\boldsymbol{x}^\star(t) = \boldsymbol{x}^{k+1}(t)$, and $\boldsymbol{z}^\star(t) = \boldsymbol{z}^{k+1}(t)$, for $k, t \to \infty$. Since proving convergence depends on the rate of change of the optimum over time, the convergence rate of OA-ADMM needs to always dominate the change of optimum as $k, t \to \infty$, i.e. $\|\boldsymbol{x}^{k_N}(t) - \boldsymbol{x}^\star(t)\|_2 - \|\boldsymbol{x}^{k_0}(t) - \boldsymbol{x}^\star(t)\|_2 > \|\boldsymbol{x}^\star(t+\delta t) - \boldsymbol{x}^\star(t)\|_2$, $\|\boldsymbol{z}^{k_N}(t) - \boldsymbol{z}^\star(t)\|_2 - \|\boldsymbol{z}^{k_0}(t) - \boldsymbol{z}^\star(t)\|_2 > \|\boldsymbol{z}^\star(t+\delta t) - \boldsymbol{z}^\star(t)\|_2$, $\forall t$, where k_0 (resp. k_N) is the first (resp. last) iteration per time step. However, we are not aware of methods that can guarantee this in general for ADMM based methods; online convergence analysis should be performed on a system specific basis. In practice, to improve online convergence, it is advised to increase the amount of OA-ADMM iterations per control step, or the control frequency.

IV. OA-ADMM MPC

OA-ADMM can be applied to the MPC problem formulation (1) provided that a finite horizon is used. Similarly to [6], this is done by introducing a slack variable $\boldsymbol{z}$ and an equality constraint $\boldsymbol{x} = \boldsymbol{z}$:

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{z_{ij}}} \quad & \sum_{i=1}^{N} J_i(x) \\ s.t. \quad & \boldsymbol{x}_i \in \mathcal{X}_i^f, \\ & d(\boldsymbol{z}_i, \boldsymbol{z}_j) \geq 0, \forall j \in \mathcal{N}_i, \\ & \boldsymbol{x}_i = \boldsymbol{z}_{ii}, \boldsymbol{x}_j = \boldsymbol{z}_{ij}, \forall j \in \mathcal{N}_i, \quad \forall i \in \{1, ..., N\}. \end{aligned} \quad (7)$$

The OA-ADMM augmented Lagrangian $\mathcal{L}_{\boldsymbol{\rho}}$ for this problem is then

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\rho}} = \sum_{i=1}^{N} \Big[& J_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}_{ii}^\top(\boldsymbol{x}_i - \boldsymbol{z}_i) + \|\boldsymbol{R}_{ii}(\boldsymbol{x}_i - \boldsymbol{z}_{ii})\|_2^2 \\ & + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^\top(\boldsymbol{x}_j - \boldsymbol{z}_{ij}) + \|\boldsymbol{R}_{ij}(\boldsymbol{x}_j - \boldsymbol{z}_{ij})\|_2^2 \right) \Big], \end{aligned} \quad (8)$$

where $\boldsymbol{\lambda}_{ii}$ (resp. $\boldsymbol{\lambda}_{ij}$) is the Lagrange multiplier in vector form for agent i (resp. agent i to agent j), and $\boldsymbol{R}_{ii}$ (resp. $\boldsymbol{R}_{ij}$) is the diagonal matrix form of $\boldsymbol{\rho}$ for agent i (resp. agent i to agent j).

We separate the overall optimization problem into smaller steps to enable the distribution of the computational load. The first step is the trajectory optimization (or x-update):

$$\boldsymbol{x}_i^{k+1} := \arg\min_{\boldsymbol{x}_i \in \mathcal{X}_i^f} \quad \mathcal{L}_{\boldsymbol{\rho},x,i}(\boldsymbol{x}_i, \boldsymbol{z}_{JI}^k, \boldsymbol{\lambda}_{JI}^k, \boldsymbol{\rho}_{JI}^k), \quad (9)$$

where $\boldsymbol{z}_{JI}$ includes $\boldsymbol{z}_{ii}$ and $\boldsymbol{z}_{ji}, \forall j \in \mathcal{N}_i$, $\boldsymbol{\lambda}_{JI}$ is the combination of $\boldsymbol{\lambda}_{ii}$ and $\boldsymbol{\lambda}_{ji}, \forall j \in \mathcal{N}_i$, and $\boldsymbol{\rho}_{JI}$ includes $\boldsymbol{\rho}_{ii}$ and $\boldsymbol{\rho}_{ji}, \forall j \in \mathcal{N}_i$. Since the x-update only adjusts $\boldsymbol{x}_i$, we can use a lighter version of the full augmented Lagrangian in (8):

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\rho},x,i} = & J_i(\boldsymbol{x}_i) + \boldsymbol{\lambda}_{ii}^{k\top}(\boldsymbol{x}_i - \boldsymbol{z}_{ii}^k) + \|\boldsymbol{R}_{ii}^k(\boldsymbol{x}_i - \boldsymbol{z}_{ii}^k)\|_2^2 \\ & + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ji}^{k\top}(\boldsymbol{x}_i - \boldsymbol{z}_{ji}^k) + \|\boldsymbol{R}_{ji}^k(\boldsymbol{x}_i - \boldsymbol{z}_{ji}^k)\|_2^2 \right), \end{aligned} \quad (10)$$

where $\boldsymbol{\lambda}_{ji}^k$ (resp. $\boldsymbol{R}_{ji}^k$) is the Lagrange multiplier (resp. penalty matrix) for agent j w.r.t agent i. The trajectory optimization step is fully parallelizable given that all the values in the augmented Lagrangian (10) are either known or independent of other agents. Then, the resulting $\boldsymbol{x}_i^{k+1}$ has to be communicated to all nearby agents. After sending $\boldsymbol{x}_i^{k+1}$ to, and receiving $\boldsymbol{x}_j^{k+1}$ from, all $j \in \mathcal{N}_i$, the copy optimization step (or z-update) can be performed, which can be seen as the collision avoidance update because of the $d(\cdot) \geq 0$ constraint:

$$\begin{aligned} \boldsymbol{z}_{IJ}^{k+1} := \arg\min_{\boldsymbol{z}_{IJ}} \quad & \mathcal{L}_{\boldsymbol{\rho},z,i}(\boldsymbol{x}_I^{k+1}, \boldsymbol{z}_{IJ}^k, \boldsymbol{\lambda}_{IJ}^k, \boldsymbol{\rho}_{IJ}^k) \\ s.t. \quad & d(\boldsymbol{z}_i^{k+1}, \boldsymbol{z}_j^{k+1}) \geq 0, \ \forall j \in \mathcal{N}_i, \end{aligned} \quad (11)$$

where $\boldsymbol{z}_{IJ}$ (resp. $\boldsymbol{x}_I$) contains both $\boldsymbol{z}_{ii}$ and $\boldsymbol{z}_{ij}, \forall j \in \mathcal{N}_i$ (resp. $\boldsymbol{x}_i$ and $\boldsymbol{x}_j, \forall j \in \mathcal{N}_i$). The reduced augmented Lagrangian $\mathcal{L}_{\boldsymbol{\rho},z,i}$ for the z-update is defined as

$$\begin{aligned} \mathcal{L}_{\boldsymbol{\rho},z,i} = & \boldsymbol{\lambda}_{ii}^{k\top}(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_i) + \|\boldsymbol{R}_{ii}^k(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_{ii})\|_2^2 \\ & + \sum_{j \in \mathcal{N}_i} \left(\boldsymbol{\lambda}_{ij}^{k\top}(\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij}) + \|\boldsymbol{R}_{ij}^k(\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij})\|_2^2 \right). \end{aligned} \quad (12)$$

Following the x and z updates, the λ-update is performed:

$$\begin{aligned} \boldsymbol{\lambda}_{ii}^{k+1} & := \mu_i(\cdot)\boldsymbol{\lambda}_{ii}^k + \boldsymbol{\rho}_{ii}^k \circ (\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_{ii}^{k+1}), \\ \boldsymbol{\lambda}_{ij}^{k+1} & := \mu_{ij}(\cdot)\boldsymbol{\lambda}_{ij}^k + \boldsymbol{\rho}_{ij}^k \circ (\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij}^{k+1}), \forall j \in \mathcal{N}_i, \end{aligned} \quad (13)$$

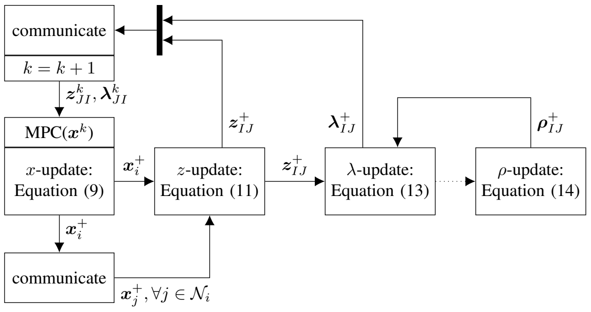

communicate
$k = k+1$
z^k_{JI}, λ^k_{JI}
MPC(x^k)
x-update: Equation (9)
x_i^+
z-update: Equation (11)
z^+_{IJ}
λ^+_{IJ}
λ-update: Equation (13)
ρ^+_{IJ}
ρ-update: Equation (14)
x_i^+
communicate
$x_j^+, \forall j \in \mathcal{N}_i$

Fig. 3: OA-ADMM based conflict resolution algorithm for agent i. We use the notation $x^+ := x^{k+1}$. The dotted arrow indicates code execution order; variables are automatically perpetuated along the arrows, e.g. the λ-update block receives x_j^+ via the z-update block.

where $\rho_{ii}^{k+1}, \rho_{ij}^{k+1}, z_{ii}^{k+1}$, and z_{ij}^{k+1} are all available locally, and μ is a forgetting factor representing the similarity between the system at the current and the previous time step. When multiple iterations of OA-ADMM are performed per control step, the value of μ can be assumed to be 1 for all iterations in the same real-time time step.

The final step of a single OA-ADMM iteration involves updating the penalty vector ρ:

$$\begin{aligned}\rho_{ij}^{k+1} &:= \phi_{ij}(x_i^{k+1}, x_j^{k+1}), \forall j \in \mathcal{N}_i \\ \rho_{ii}^{k+1} &:= \phi_{ii}(x_i^{k+1}),\end{aligned} \tag{14}$$

where $\phi(\cdot)$ is a function of the states, whose expression can be chosen depending on the desired behavior, provided that the resulting R is always a positive definite diagonal matrix. The updated values of z^{k+1}, ρ^{k+1}, and λ^{k+1} are then communicated to complete one OA-ADMM iteration.

A. Designing the adaptation function $\phi(\cdot)$

Possible approaches to design adaptive penalty parameters for faster convergence have been explored in [12] and [16]. With OA-ADMM, we may want to fit other needs. For example, in the case of real-time optimization for decentralized conflict resolution, we prioritize safety over convergence speed. Furthermore, instead of using rule-based adaptation schemes as in [12] and [16], we use a more general adaptation function ϕ.

The requirements on $\phi(\cdot)$ for online convergence can be summarized as $\dot{\phi}(\cdot) = 0, \text{for } t \to \infty$. In addition of the basic requirement on ϕ, we also have to take into account the purpose of ϕ in the real-time control case. Since we are applying OA-ADMM and MPC to a motion planning problem involving autonomous vehicles, we wish to use ϕ to improve the online collision avoidance behavior. The value of the penalty parameter ρ has a large influence on the convergence rate of OA-ADMM; additionally, ρ tunes the importance of the primal and dual residuals during optimization, with a large ρ prioritizing the primal residual r^k, and a small ρ prioritizing the dual residual s^k.

Given that the primal residual for (7) is $x - z$, increasing ρ effectively increases the penalty for the actual trajectory deviating from the copies. However, since (7) is a multi-agent problem, there are also agent specific values of ρ. In essence, each agent i has three relevant types of ρ, namely ρ_{ii}, ρ_{ij} and ρ_{ji}. The first type, ρ_{ii}, directly affects λ_{ii}^{k+1} by scaling part of the primal residual ($x_{ii}^{k+1} - z_{ii}^{k+1}$), it is also present in the augmented Lagrangian, acting as a weight on the residual inside the squared L2 norm. The value of ρ_{ii} can therefore be summarized as the weight for the cost of the deviation between x_{ii}^{k+1} (resp. z_{ii}^{k+1}) and z_{ii}^{k} (resp. x_{ii}^{k+1}). The second type of ρ is ρ_{ij}, which is present in the calculation of λ_{ij}^{k+1} as a weight for the residual $x_k^{k+1} - z_{ij}^{k+1}$ and in the augmented Lagrangian $\mathcal{L}_{\rho,z,i}$ in which it scales the same residual inside the norm. As a result, the value of ρ_{ij} directly affects the z-update for agent i, with larger values allowing less deviation of z_{ij}^{k+1} from x_j^{k+1}. The final ρ is ρ_{ji}, which is in essence the reverse of ρ_{ij}, i.e. $\rho_{1,2} = \rho_{2,1}$. From the perspective of agent i, ρ_{ji} acts purely in the x-update, penalizing deviation of x_i^{k+1} from z_{ji}^{k+1}.

The adaptation function used in this paper, to enhance online safety and robustness, adapts the penalty parameter based on the physical states:

$$\phi(\cdot)_{ij} = \begin{cases} w_i \phi^{min}, \text{ if } \left(\frac{D}{d(x_i^k, x_j^k)}\right)^a < \phi^{min} \\ w_i \phi^{max}, \text{ if } \left(\frac{D}{d(x_i^k, x_j^k)}\right)^a > \phi^{max} \\ w_i \left(\frac{D}{d(x_i^k, x_j^k)}\right)^a, \text{ otherwise} \end{cases} \tag{15}$$

and

$$\phi(\cdot)_{ii} = w_i \frac{1}{N_i} \sum_{j \in \mathcal{N}_i} \left(\phi(\cdot)_{ij}\right)^a, \tag{16}$$

where D is the minimum distance, w_i is a weight that can modify the importance of agent i, a is a variable that determines the shape of ϕ, and N_i is the amount of agents in $\mathcal{N}_i$. The minimum bound ϕ^{min} ensures that $\rho > 0$, avoiding the problem becoming ill-conditioned, the maximum bound ϕ^{max} limits the possibility for extreme values of ϕ, which can destabilize the system. For (15), this can occur when $d(x_i, x_j) \approx 0$, which can occur when planned trajectories overlap. The ϕ given in (15)-(16) therefore adjusts the value of ρ_{ij} and ρ_{ii} when the online MPC is planning trajectories with high probability of collisions, increasing the primal feasibility whilst sacrificing individual optimality. The advantage of this method is that this distance-based approach is simple to design, yet achieves results similar to more advanced techniques. The function given in (15)-(16) can be interpreted as a control barrier or potential field function for the multi-agent motion planning problem.

B. Designing the similarity function $\mu(\cdot)$

When $\mu(\cdot) = 1$, the change from t to $t + \delta t$ has no effect on the previous OA-ADMM iterations, i.e. $\lambda^{k+1} = \lambda^k + \rho^k \circ r^{k+1}$. Conversely a $\mu(\cdot) = 0$ implies that there is no useful relation between the previous time step t and the current time step $t + \delta t$, i.e. $\lambda^{k+1} = \rho^k \circ r^{k+1}$. The difficulty is however designing a function μ that, using the information available, results in effective online performance. If the system is fully known, it might be possible to analytically find the optimal $\mu^\star$, this is however a time intensive procedure and very system dependent. Instead we attempt an intuitive approach to find a μ which approximates the behavior of $\mu^\star$.

We know that if $\mu(\cdot)^\star = 1$ it should hold that $\boldsymbol{x}^\star(t) = \boldsymbol{x}^\star(t+\delta t)$: the optimum should not change from t to $t + \delta t$ if $\mu = 1$. Additionally, if OA-ADMM has reached the optimum, then $\boldsymbol{x}^k(t) = \boldsymbol{x}^{k+1}(t)$. Ergo, if OA-ADMM has converged and $\mu^\star = 1$, then $\boldsymbol{x}^k(t) = \boldsymbol{x}^{k+1}(t+\delta t)$. Given that this can be calculated in run-time, we can utilize this to construct a μ which approximates $\mu^\star$ in the optimum. For example, the following formula satisfies the requirements:

$$\mu(\cdot) = w_x \left(1 - \frac{\|\boldsymbol{x}^{k+1}(t+\delta t) - \boldsymbol{x}^k(t)\|_2}{\|\boldsymbol{x}^k(t)\|_2} \right), \qquad (17)$$

where $w_x + w_z + w_\lambda + w_\rho = 1$. Note that this also hold for $\boldsymbol{z}, \boldsymbol{\lambda}$, and $\boldsymbol{\rho}$. This approach, however, requires that OA-ADMM is performed until convergence, as only then $\boldsymbol{x}^\star(t) = \boldsymbol{x}^\star(t+\delta t)$ is guaranteed.

Another approach is to construct a $\mu(\cdot)$ by evaluating the role of μ. The similarity function implemented is based on the idea that the relevance of the previous $\boldsymbol{\lambda}$ is positively correlated to the value of $\boldsymbol{\rho}$. An intuitive explanation for the conflict resolution case is to view $\boldsymbol{\lambda}$ as a penalty by OA-ADMM aiming to enforce the collision avoidance constraint. When a collision is likely, $\boldsymbol{\rho}$ will increase due to the design of the ϕ. In this case, it is desirable to increase the penalty $\boldsymbol{\lambda}$ to enforce the constraint. However, when collision are unlikely, continuing with the previous $\boldsymbol{\lambda}$ can result in suboptimality. This concept is implemented as follows:

$$\mu(\cdot, k) = \eta\mu(\cdot, k-1) + (1-\eta)\min(\rho_{JI}\frac{1}{w_i}, \mathbf{1}), \qquad (18)$$

where the elements of μ are bounded to be less or equal to one, along with a weighted average (scaled with $0 \leq \eta \leq 1$ acting as a simple filter to reduce the effects of disturbances) between the current and the previous value of μ.

V. Numerical Simulations

In this section, we evaluate the robustness of OA-ADMM MPC for an autonomous vehicle simulated in CARLA, additionally we compare the conflict resolution efficiency against the decentralized conflict resolution methods from [4] (AMP-IP) and [5] (TDCR[2]). Both methods are limited in terms of control input, both only able to adjust their velocities along the planned trajectory. Whilst AMP-IP is a reactive strategy, TDCR uses prediction in their method, which allows vehicles to plan their velocities ahead accordingly.

A. Simulation Setup

The simulations are carried out using the benchmarking tool described in Appendix II[1]. The metrics measured are the total travel time per vehicle for their respective cases, these are compared against the no conflict case for each respective protocol to get the added delay caused by each protocol. The no conflict case for each protocol simulates the same amount of vehicles with the same exact reference velocities, ensured by the identical random seeds. A major difference between OA-ADMM MPC and the traditional methods of AMP-IP and TDCR lies in that OA-ADMM MPC does not require the map of the environment beforehand. To attempt to show the effects of this prior knowledge, the traditional approaches are simulated for the three different fidelity cases: a 1x1 grid (low fidelity), a 4x4 grid (medium fidelity), and a 8x8 grid (high fidelity), with all grids having a dimensions of 18x18m centered at the intersection. All the protocols are tested by spawning vehicles at equal distance to the intersection center with a reference velocity of 4 m/s, with uniformly distributed variations between -0.15 m/s and 0.15 m/s. All the possible cases depicted in Figure 5[1], except for the vehicle follower case, are simulated in threefold and averaged out. The simulator is ran at a frequency of 160 Hz, with the vehicles running the protocols at 20 Hz; vehicles perform the low level control at 40 Hz to reduce instability.

[2]The conflict resolution method proposed in [5] is unnamed, for convenience sake we will refer to it as the Timeslot-based Decentralized Conflict Resolution method (TDCR).

B. Simulation Results

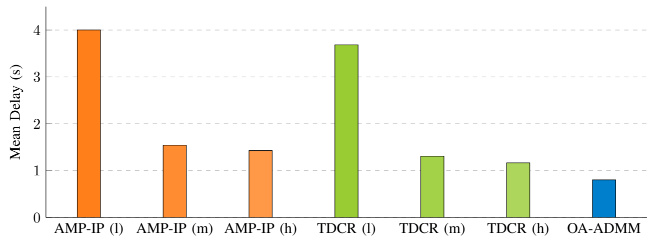


Fig. 4: Mean delay values for the decentralized protocols.

The average delay for the AMP-IP, TDCR, and OA-ADMM is shown in Figure 4, where AMP-IP and TDCR are shown separately for each of their grid fidelity cases.

Protocol (case)	Mean Time (s)	Mean Delay (s)	Mean Add. Delay (s)
AMP-IP (n)	17.804	(–)	(–)
AMP-IP (l)	21.806	4.002	3.593
AMP-IP (m)	19.346	1.542	1.133
AMP-IP (h)	19.231	1.427	1.018
TDCR (n)	17.809	(–)	(–)
TDCR (l)	21.493	3.684	3.275
TDCR (m)	19.117	1.307	0.898
TDCR (h)	18.974	1.165	0.756
OA-ADMM (n)	18.158	(–)	(–)
OA-ADMM	18.960	0.802	0.394

TABLE I: Summary of AMP-IP, TDCR, and OA-ADMM results. The mean delay is measured relative to the no conflict case, the mean added delay is measured against the mean estimated delay (0.4089 s) from the benchmark.

The mean time and mean delay for all the protocols are given in Table I. Compared with AMP-IP, OA-ADMM MPC is found to have a 79.95%, 47.95%, and 43.74% decrease in mean delay for the low, medium, and high fidelity cases respectively. The percentage decrease in average added delay regarding the low, medium, and high fidelity cases for AMP-IP are 89.04%, 65.25%, and 61.32% respectively. Compared with TDCR, OA-ADMM MPC is found to have a 78.21%, 38.61%, and 31.11% decrease in mean delay for the low, medium, and high fidelity cases respectively. The percentage decrease in mean added delay regarding the low, medium, and high fidelity cases for TDCR are 87.98%, 56.18%, and 47.93% respectively.

A detailed overview of the delays for the protocols is given in Appendix III[1], where the delay for each individual conflict case is given separately. Note that only the high fidelity cases are shown, the delays for the lower fidelity cases are deemed less relevant for the detailed comparison as they are always higher than the delays for the high fidelity cases.

The results indicate that OA-ADMM MPC is outperforming both conventional methods for all cases. This can be attributed to the use of a collision avoidance constraint compared with an entry time constraint used by TDCR. An entry time constraint is limited in detail by the size of the cells, thereby effective conflict resolution requires detailed prior knowledge of the environment, which can be costly to obtain/save and is not always available. OA-ADMM MPC, however, only requires the relative positions of the vehicles, which can be obtained in real-time with relative ease.

C. Tuning Complexity

In addition to the simulations conducted in CARLA using the benchmark, the tuning complexity of OA-ADMM compared with ADMM is analyzed using a simpler MATLAB test case. The test case involves four holonomic circular robots approaching an intersection simultaneously. To avoid deadlocks, vehicles on the horizontal lane have their values of w_i from (15) doubled.

Algorithm	Time outs	Violations	Resolved	Mean Delay (s)	Mean MSV (m^2)
ADMM	86	133	1	5.35	$2.33{\cdot}10^{-2}$
OA-ADMM	49	83	88	2.93	$8.83{\cdot}10^{-3}$

TABLE II: Comparison between ADMM and OA-ADMM for 220 hyperparameter combinations. Timeouts imply that the case did not resolve within 30 seconds; violations are cases with constraint violations; resolved implies that no constraint violation or timeout has occurred.

To gain insight on the tuning complexity of OA-ADMM, the simulation is performed for a range of hyperparameter combinations: $D \in \{0, 0.1, ..., 1\}$ and $w_i \in \{0.25, 0.5, ..., 5\}$. These combinations are then simulated and evaluated for delay and mean square constraint violations (MSV), the results of which are given in Table II. The results indicate that OA-ADMM, for the tested cases, resolves significantly more cases, whilst having shorter delays. In the cases where the constraint was violated, OA-ADMM had lower values of MSV, indicating that OA-ADMM is significantly easier to tune and more robust then conventional ADMM when used in combination with MPC.

VI. Conclusion

OA-ADMM is a novel, flexible framework to use ADMM for robust online optimization. In our case study, the chosen adaptation function improves the robustness of decentralized MPC enough to achieve improved conflict resolution efficiency compared with competing decentralized conflict resolution methods like AMP-IP and TDCR.

Given that OA-ADMM is a novel framework, a lot of work can be done to further explore the proposed adaptation function and similarity function, including e.g. the possibility of an optimal similarity function. Better guarantees for online convergence of OA-ADMM could be provided in combination with stricter requirements on the online system and the adaptation and similarity functions.

References

[1] Y. Rahmati and A. Talebpour, "Towards a collaborative connected, automated driving environment: A game theory based decision framework for unprotected left turn maneuvers," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1316–1321.

[2] J. A. Khan, L. Wang, E. Jacobs, A. Talebian, S. Mishra, C. A. Santo, M. Golias, and C. Astorne-Figari, "Smart Cities Connected and Autonomous Vehicles Readiness Index," in *ACM SCC*, Portland, OR, United States, 2019.

[3] R. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Intersection Management using Vehicular Networks," in *SAE 2012 World Congress & Exhibition*, Apr. 2012, pp. 2012–01–0292.

[4] S. Azimi, G. Bhatia, R. Rajkumar, and P. Mudalige, "Reliable intersection protocols using vehicular networks," in *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, Apr. 2013, pp. 1–10.

[5] C. Liu, C. Lin, S. Shiraishi, and M. Tomizuka, "Distributed Conflict Resolution for Connected Autonomous Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 18–29, Mar. 2018.

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

[7] ——, "Distributed model predictive formation control with intervehicle collision avoidance," in *2017 11th Asian Control Conference (ASCC)*, Dec. 2017, pp. 2399–2404.

[8] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Fast ADMM for Distributed Model Predictive Control of Cooperative Waterborne AGVs," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1406–1413, Jul. 2017.

[9] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully Decentralized ADMM for Coordination and Collision Avoidance," in *2018 European Control Conference (ECC)*, Jun. 2018, pp. 825–830.

[10] R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, "A Distributed Multi-Robot Coordination Algorithm for Navigation in Tight Environments," *arXiv:2006.11492 [cs]*, Jun. 2020.

[11] L. Chen, H. Hopman, and R. R. Negenborn, "Distributed model predictive control for vessel train formations of cooperative multi-vessel systems," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 101–118, Jul. 2018.

[12] B. S. He, H. Yang, and S. L. Wang, "Alternating Direction Method with Self-Adaptive Penalty Parameters for Monotone Variational Inequalities," *Journal of Optimization Theory and Applications*, vol. 106, no. 2, pp. 337–356, Aug. 2000.

[13] B. E. Jackson, T. A. Howell, K. Shah, M. Schwager, and Z. Manchester, "Scalable cooperative transport of cable-suspended loads with uavs using distributed trajectory optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3368–3374, 2020.

[14] A. Colombo and D. Del Vecchio, "Efficient Algorithms for Collision Avoidance at Intersections," in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '12. New York, NY, USA: ACM, 2012, pp. 145–154.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jul. 2011.

[16] Z. Xu, M. Figueiredo, and T. Goldstein, "Adaptive admm with spectral penalty parameter selection," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 718–727.