基于对偶共识交替方向乘子法的网联自动驾驶汽车协同轨迹规划分布式 iLQR

Zhenmin Huang、Shaojie Shen 和 Jun Ma

arXiv:2301.04386v1 [cs.RO] 11 Jan 2023

摘要—**网联自动驾驶汽车（CAVs）的协同轨迹规划发展势头强劲，并日益受到研究者关注。一般而言，此类问题具有显著的非线性和非凸性，使得求得最优解颇具挑战。现有方法通常计算效率较低，这限制了它们在车辆数量不断增加的大规模场景中的适用性。为解决这一问题，我们利用对偶共识交替方向乘子法（ADMM），提出一种新颖的分布式迭代线性二次调节器（iLQR）算法。首先，通过迭代邻域近似，将原始非凸优化问题重构为一系列凸优化问题。随后，证明每个凸优化问题的对偶问题具有共识结构，从而能够在完全分布式并行架构中使用共识交替方向乘子法求解对偶解。最后，通过迭代求解线性二次调节器（LQR）问题，恢复与每辆车轨迹相对应的原始解，并提出一种新颖的轨迹更新策略，以确保车辆的动态可行性。通过上述方法，计算负担显著减轻，从而实现实时性能。本文给出两个交通场景以验证所提算法，并将所提方法与基线方法（包括集中式 iLQR、IPOPT 和序列二次规划（SQP））进行全面比较，以证明所提方法的可扩展性。**

索引词—**自动驾驶、多智能体系统、迭代线性二次调节器（iLQR）、微分动态规划（DDP）、交替方向乘子法（ADMM）、网联自动驾驶汽车、协同轨迹规划、非凸优化。**

I. 引言

道路上车辆数量持续增加，给交通系统带来了沉重负担，导致严重拥堵和安全问题。与此同时，人类驾驶员普遍缺乏协作性，导致他们激烈争夺有限的交通资源，甚至使情况进一步恶化。现存问题，加上自动驾驶技术和信息技术的快速发展，推动了网联自动驾驶汽车（CAVs）的出现，它们提供了一种协作驾驶解决方案，有望缓解这一问题 [1]。CAVs 通常配备一个或多个通信设备，以便与其他道路车辆（车对车）、路侧基础设施（车对基础设施）和云端设备（车对云）交换信息 [2]。此类通信能够在 CAVs 之间传递驾驶意图，并构成协作驾驶的物理基础。然而，从算法开发的角度来看，CAVs 之间的协同轨迹规划问题仍基本未得到解决。协同轨迹规划的主要目标是为所有参与的 CAVs 制定联合决策并生成协同轨迹，使其满足车辆动力学、交通规则、安全条件及其他相关约束，同时最大限度地提高整体效率 [3]–[5]。尽管已有多种方法被提出，但交通场景的复杂性以及协同轨迹规划固有的耦合特性，使这类问题难以求解。此外，随着参与的 CAVs 数量增加，协同轨迹规划问题的规模也随之扩大，导致时间性能下降。这些问题使得实时求得最优解面临巨大困难，尤其是在计算资源有限且场景规模较大的情况下，因此协同轨迹规划长期以来一直是自动驾驶领域的一项挑战。

目前，为解决协同轨迹规划问题，研究人员正重点研究两类方法。基于学习的方法为利用真实世界驾驶数据并捕捉复杂驾驶行为，以应对困难交通场景提供了可行途径 [6]–[8]。然而，对大量真实世界数据的需求，以及神经网络缺乏可解释性，限制了其广泛应用。另一方面，基于优化的方法更加成熟且发展完善，具有可预测的结果和良好的可解释性。这类方法将协同轨迹规划问题 for mulates 为约束优化问题，其中纳入了有关安全条件和控制约束的预设约束，并试图最小化相关目标函数，例如控制成本以及与参考轨迹的偏差等。在基于优化的方法中，集中式方法和分布式方法都已得到充分研究。集中式方法通常采用中央协调器，由其感知完整信息并为所有网联自动驾驶汽车作出决策 [9], [10]。文献 [11], 提出了一种分层集中式协调方案，由中央交通协调器执行，负责处理

Zhenmin Huang、Shaojie Shen 和 Jun Ma 任职于中国香港特别行政区香港科技大学电子及计算机工程系（电子邮件：zhuangdf@connect.ust.hk；eeshaojie@ust.hk；jun.ma@ust.hk）。
本文已提交 IEEE，拟申请发表。版权可能在未另行通知的情况下转让；转让后，此版本可能不再 доступ。

交叉路口的交通流。尽管集中式方法是解决协同轨迹规划问题的自然方案，但其缺点之一是，随着网联自动驾驶汽车数量增加以及中央协调器计算能力有限，计算效率会降低。这种相对于车辆数量的较差可扩展性阻碍了其在大规模场景中的应用。相反，分散式方法要求每辆网联自动驾驶汽车基于局部信息作出自身决策 [12]–[14]。因此，计算负载自然分布于所有车辆之间，从而带来更好的可扩展性。然而，每辆网联自动驾驶汽车感知到的信息不完整，实质上使优化问题更加复杂；显然，这也需要在算法开发方面投入更多努力。

一般而言，基于优化的方法所构建的优化问题具有很强的非线性。因此，通常需要采用非线性规划；在这种情况下，求解效率很大程度上取决于求解器。序列二次规划（SQP）和内点优化器（IPOPT）是广泛用于求解此类优化问题的非线性规划求解器，但在大多数场景下效率较低。与 SQP 和 IPOPT 相比，差分动态规划（DDP）具有内存消耗低、计算效率显著提升等优点，因此已广泛应用于各种最优控制问题 [15]–[17]。与此同时，迭代线性二次调节器（iLQR）通过消去系统动力学的二阶项，提供了 DDP 的简化版本，从而进一步提高效率。阻碍原始 DDP/iLQR 算法应用于车辆轨迹规划的主要问题在于，它无法处理系统动力学以外的约束。为解决这一问题，控制受限差分动态规划 [18] 将控制约束纳入 DDP 框架，而 CiLQR [19] 则将 iLQR 与对数障碍函数相结合，以进一步纳入一般形式的约束。这些工作使 DDP/iLQR 得以应用于轨迹规划。近期将 iLQR 应用于协同轨迹规划的研究受到 [20], 的启发，该研究将交互式轨迹规划问题建模为势博弈，并通过 iLQR 算法求解单个最优控制问题，得到相应的纳什均衡。尽管在最优控制和轨迹规划领域中，DDP/iLQR 属于效率和可扩展性最高的方法之一，但将其直接以集中式方式应用于高维密集系统，仍会导致次优性能。因此，开发分布式版本的 DDP/iLQR，有望进一步提高计算效率和可扩展性。

交替方向乘子法（ADMM） [21] 具有分布式特性，因此被广泛应用于各个领域的优化问题求解 [22]–[25]。ADMM 的基本思想是将原始问题分解为规模较小、易于处理的子问题，以便并行、分布式地求解。文献 [26], 提出了一种基于 ADMM 的轨迹规划算法，将车辆动力学约束与其他约束分离，以便分别处理；并证明该算法在计算效率方面优于序列二次规划（SQP）和 IPOPT。同时，研究者提出了共识交替方向乘子法，用于求解连通无向图上的共识优化问题 [27] 。鉴于具有通信网络结构的系统广泛存在，该方法已应用于资源分配 [28] 和模型预测控制 [29]等多个领域。随后提出的对偶共识交替方向乘子法 [30], [31], , 将共识交替方向乘子法扩展至其对偶问题为共识优化问题的优化问题。文献 [32]–[35]展示了 ADMM 在协同轨迹规划中的近期应用。特别是，文献 [32]–[34], 采用 ADMM 将多智能体系统的独立约束与耦合约束分离，从而能够并行处理前者，得到部分分布式的解。与此同时，文献 [35] 提出了一种完全分布式优化框架，用于求解大规模多机器人协同轨迹规划问题。然而，即使智能体数量相对较少，该方法的性能仍远未达到实时要求。

本研究结合 iLQR 处理非线性的高效性与对偶共识交替方向乘子法的分布式特性，提出了一种全新的完全去中心化、可并行化优化框架，用于网联自动驾驶汽车的协同轨迹规划，并采用凸重构方法处理相应优化问题中固有的非凸性。通过该优化框架，网联自动驾驶汽车之间关于安全距离的耦合约束得以有效解耦，并由所有车辆协同处理。我们还提出了一种全新的动态可行轨迹更新策略，以确保满足非线性车辆动力学和控制约束。本文的主要贡献如下:

- 本文提出了一种凸重构方法，用于处理网联自动驾驶汽车协同轨迹规划原始约束优化问题中的非凸性。
- 本文结合 iLQR 和对偶共识交替方向乘子法的优点，提出了一种全新的完全去中心化、可并行化优化框架，将原始优化问题拆分为多个子问题，从而使所有网联自动驾驶汽车的计算负载得以均衡分配，并显著提高计算效率。
- 本文提出一种基于 iLQR 的创新性动态可行轨迹更新策略，以保证网联自动驾驶汽车满足系统动力学和预设的控制约束。
- 本文还提供了一种基于多进程的所提方法完全并行实现，以实现实时性能。由于本文提出的方法是面向多智能体系统轨迹规划的统一框架，因此相同的实现可以轻松推广到广泛的应用中。

本文其余部分组织如下。第二节介绍网联自动驾驶汽车协同轨迹规划问题的建模。第三节

介绍基于对偶共识交替方向乘子法的凸重构和分布式 iLQR 算法，以解决该问题。第四节给出两个交通场景，以展示所提算法的有效性，并进行深入讨论。最后，第五节总结本文工作，并展望若干可能的未来研究方向。

符号说明： $\mathbb{R}^{a\times b}$ 表示包含 a 行、 b 列的实矩阵空间， $\mathbb{R}^a$ 表示 a维实列向量空间。类似地， $\mathbb{Z}^{a\times b}$表示包含 a 行、 b 列的整数矩阵空间， $\mathbb{Z}^a$ 和 a分别表示整数列向量空间。 $\mathbf{0}^{a\times b}$ 表示一个包含 a行、 b 列的全零矩阵。 I^n 表示一个 $n\times n$ 单位矩阵。 $A^\top$和 $x^\top$ 分别表示矩阵和列向量的转置。算子 $|\cdot||$ 表示向量的欧几里得范数。我们还用 $\text{blocdiag}\{A_1,A_2,...,A_n\}$ 表示一个分块对角矩阵，其分块对角线上的元素为A_1、 A_2、 ...、 A_n。 $(x_1$、 x_2、 ...、 $x_n)$表示以下列向量集合 $\{x_1,x_2,...,x_n\}$拼接而成的向量，即$(x_1$、 x_2、 ...、 $x_n)=[x_1^\top,x_2^\top,...,x_n^\top]^\top$。近端算子定义为 $\text{Prox}_f^\rho(x):=\arg\min_y \{f(y)+\frac{\rho}{2}||x-y||^2\}$。

II. 问题表述

对于涉及 N 辆网联自动驾驶汽车的交通场景，我们用 $\mathcal{N}=\{1,2,...,N\}$ 表示包含每辆网联自动驾驶汽车索引的集合。在离散时间设定下，我们用 $\mathcal{T}=\{0,1,...,T-1\}$ 表示时间戳。每辆车辆 $i, i\in\mathcal{N}$ 的系统动力学由以下离散时间非线性方程给出

$$x_{\tau+1}^i=f(x_\tau^i,u_\tau^i), \tag{1}$$

其中， $x_\tau^i\in\mathbb{R}^n$ 和 $u_\tau^i\in\mathbb{R}^m$ 分别是车辆 i 在时间 τ、对于 $\tau\in\mathcal{T}$ 的状态向量和控制输入向量， x_0^i 已知且固定。

我们将 $x_\tau=(x_\tau^1,x_\tau^2,...,x_\tau^N)$ 和 $u_\tau=(u_\tau^1,u_\tau^2,...,u_\tau^N)$ 分别表示为时间 τ 所有车辆状态变量和控制输入的串接向量；此外， $U=(u_0,u_1...,u_{T-1})$ 表示所有车辆在所有时间的输入串接向量。根据 [20], ，总体代价可表示为

$$J(U)=\sum_{\tau=0}^{T-1}P_\tau(x_\tau,u_\tau)+P_T(x_T) \tag{2}$$

其中

$$P_\tau(x_\tau,u_\tau)=\sum_{i=1}^N C_\tau^i(x_\tau^i,u_\tau^i)+\sum_{1\le i<j\le N}C_\tau^{ij}(x_\tau^i,x_\tau^j), \tag{3}$$

$$P_T(x_T)=\sum_{i=1}^N C_T^i(x_T^i)+\sum_{1\le i<j\le N}C_T^{ij}(x_T^i,x_T^j). \tag{4}$$

稍微滥用一下记号，我们分别用 C_τ^i 和 C_T^i 表示车辆 i 在时间 τ 的运行代价和终端代价。需要注意的是，终端代价不是控制输入的函数。 C_τ^i运行代价衡量的是车辆 i 偏离其参考位置 $x_{\tau,ref}^i$ 以及时间 τ的控制代价，定义为

$$C_\tau^i(x_\tau^i,u_\tau^i)=(x_\tau^i-x_{\tau,ref}^i)^\top Q(x_\tau^i-x_{\tau,ref}^i)+u_\tau^{i\top}Ru_\tau^i. \tag{5}$$

类似地，终端代价 C_T^i 定义为

$$C_T^i=(x_T^i-x_{T,ref}^i)^\top Q(x_T^i-x_{T,ref}^i) \tag{6}$$

其中 $x_{T,\ ref}^i$ 为终端参考位置。 C_τ^{ij}是车辆 i 与车辆 j 在时间 τ的成对碰撞规避代价，其定义为

$$C_\tau^{ij}(x_\tau^i,x_\tau^j)=\begin{cases}\beta(d_\tau^{ij}-d_{\text{safe}})^2 & \text{if } d_\tau^{ij}<d_{\text{safe}}\\ 0 & \text{else}\end{cases} \tag{7}$$

其中， d_τ^{ij}是时间 τ车辆i 与 j 中心之间的欧氏距离， d_{safe}是给定的安全距离，当两辆车彼此靠近时，便会在总体代价中增加惩罚项。 β 是决定两种代价比例的系数。理论上，可以通过选取极大的 β 来保证安全性，使软惩罚近似发挥硬约束的作用。

同时，由于各种物理限制，车辆的控制输入受到约束。这些限制包括转向角限制，以及因发动机驱动力和制动力有限而导致的加速度限制。因此，施加以下盒约束。

$$\underline{u}_\tau^i\preceq u_\tau^i\preceq\overline{u}_\tau^i,\forall i\in\mathcal{N},\forall\tau\in\mathcal{T} \tag{8}$$

考虑车辆动力学、目标函数和约束后，可得出以下优化问题。

$$\begin{aligned}\min_U\quad & J(U)\\ s.t.\quad & x_{\tau+1}^i=f(x_\tau^i,u_\tau^i),\\ & \underline{u}_\tau^i\preceq u_\tau^i\preceq\overline{u}_\tau^i,\\ & \forall i\in\mathcal{N},\forall\tau\in\mathcal{T}\end{aligned} \tag{9}$$

注释 1. 此类优化问题可视为最优控制问题（OCP），并可通过约束 iLQR [19]求解。然而，由于存在成对的碰撞规避代价，以集中式方式求解此类问题需要对一个稠密正定矩阵求逆，该矩阵在每个时间戳的元素数量与 N^2 成正比，因此计算复杂度为 $O(N^3)$。这种方法在车辆数量方面的可扩展性较差，因此很难应用于大规模交通场景。

III. 基于对偶共识交替方向乘子法的协同轨迹规划分布式 iLQR

本节对上述最优控制问题（OCP）（9）进行凸重构，并结合 iLQR 与对偶共识交替方向乘子法各自的优势，提出一种优化框架。该框架以完全分布式且可并行化的方式求解该 OCP。因此，它在车辆数量方面表现出显著的可扩展性。我们假设通信网络结构为完全图，使每对网联自动驾驶汽车（CAVs）都能彼此直接交换信息。

A. 凸重构

需要指出的是，由于存在非线性车辆动力学，以及碰撞规避代价带来的非凸性，问题（9）是强非凸的。这种非凸性是大多数协同轨迹规划问题固有的。然而，应用双重共识交替方向乘子法算法需要问题满足强对偶性，而这通常要求问题具有凸性。因此，我们首先需要将问题（9）重构为凸优化问题。凸重构包括对系统动力学进行线性化，并围绕名义车辆轨迹的邻域对所有代价进行凸近似。

给定车辆 i的一条可行名义轨迹 $\{x_\tau^i, u_\tau^i\} \in \mathbb{R}^M$，其中 $M=(T+1)n+Tm$为轨迹的总维度，可将车辆动力学线性近似为

$$\begin{aligned}\delta x_{\tau+1}^i &= A_\tau^i \delta x_\tau^i + B_\tau^i \delta u_\tau^i,\\ A_\tau^i &= \frac{\partial f(x_\tau^i, u_\tau^i)}{\partial x_\tau^i}, B_\tau^i = \frac{\partial f(x_\tau^i, u_\tau^i)}{\partial u_\tau^i}, \forall i \in \mathcal{N}.\end{aligned} \tag{10}$$

对于所有主体代价 C_τ^i 和终端代价 C_T^i，将扰动后的代价记为 $\bar{C}_\tau^i(\delta x_\tau^i, \delta u_\tau^i) = C_\tau^i(x_\tau^i+\delta x_\tau^i, u_\tau^i+\delta u_\tau^i)$ 和$\bar{C}_T^i(\delta x_T^i) = C_T^i(x_T^i + \delta x_T^i)$。通过进行二阶泰勒展开，可得到此类扰动代价的近似形式：

$$\begin{aligned}\bar{C}_\tau^i(\delta x_\tau^i, \delta u_\tau^i) \approx & C_\tau^i(x_\tau^i, u_\tau^i) + \delta u_\tau^{i\top} C_{\tau,u}^i + \delta x_\tau^{i\top} C_{\tau,x}^i \\ & + \frac{1}{2}\delta u_\tau^{i\top} C_{\tau,uu}^i \delta u_\tau^i + \frac{1}{2}\delta x_\tau^{i\top} C_{\tau,xx}^i \delta x_\tau^i \\ \bar{C}_T^i(\delta x_T^i) \approx & C_T^i(x_T^i) + \delta x_T^{i\top} C_{T,x}^i + \frac{1}{2}\delta x_T^{i\top} C_{T,xx}^i \delta x_T^i\end{aligned} \tag{11}$$

其中，$C_{\tau,u}^i$和$C_{\tau,x}^i$是 C_τ^i关于 u_τ^i和 x_τ^i的偏导数，$C_{\tau,uu}^i$和$C_{\tau,xx}^i$是 C_τ^i关于 u_τ^i和 x_τ^i的二阶偏导数。类似地，$C_{T,x}^i$和$C_{T,xx}^i$分别是 C_T^i关于 x_T^i的一阶和二阶导数。

接下来，我们将扰动后的主体总成本定义为

$$\begin{aligned}\bar{F}^i(\delta X^i) = & \sum_{\tau=0}^{T-1} \bar{C}_\tau^i(\delta x_\tau^i, \delta u_\tau^i) + \bar{C}_T^i(\delta x_T^i) \\ & - \sum_{\tau=0}^{T-1} C_\tau^i(x_\tau^i, u_\tau^i) - C_T^i(x_T^i)\end{aligned} \tag{12}$$

它可以改写为紧凑矩阵形式。我们定义$\delta X^i = (\delta x_0^i,\ \delta u_0^i,\ \ldots\ \ldots,\ \delta x_T^i) \in \mathbb{R}^M$为与车辆 i对应的所有状态向量和输入向量变化量组成的串接向量，定义 $L_1^i = (C_{0,\ x}^i,\ C_{0,\ u}^i,\ \cdots,\ C_{T,\ x}^i) \in \mathbb{R}^M$为包含所有一阶雅可比矩阵的列向量，并定义$L_2^i = \text{blocdiag}\{C_{0,xx}^i, C_{0,uu}^i, \cdots, C_{T,xx}^i\} \in \mathbb{R}^{M\times M}$为包含所有二阶海森矩阵的分块对角矩阵。(12)可写为

$$\bar{F}^i(\delta X^i) = \delta X^{i\top} L_1^i + \frac{1}{2}\delta X^{i\top} L_2^i \delta X^i. \tag{13}$$

线性化动态约束也可以用矩阵形式表示为

$$(L_3^i - L_4^i)\delta X^i = 0 \tag{14}$$

其中 $L_3^i = \text{blocdiag}\{[A_0^i\ \ B_0^i], [A_1^i\ \ B_1^i], ..., [A_{T-2}^i\ \ B_{T-2}^i], [A_{T-1}^i\ \ B_{T-1}^i\ \ \mathbf{0}^{n\times n}]\}$，且 $L_4^i = \text{blocdiag}\{[\mathbf{0}^{n\times n}\ \ \mathbf{0}^{n\times m}\ \ I^n], \underbrace{[\mathbf{0}^{n\times m}\ \ I^n],\ .....,\ [\mathbf{0}^{n\times m}\ \ I^n]}\}$。此类线性等式约束$T-1$可通过示性函数处理，并将其加入总体代价中。

定义 1. 关于集合 $\mathcal{X}$的示性函数定义为

$$\mathcal{I}_\mathcal{X} = \begin{cases} 0 & \text{if } x \in \mathcal{X} \\ \infty & \text{otherwise} \end{cases} \tag{15}$$

我们定义 $\mathcal{X}^i = \{\delta X^i | (L_3^i - L_4^i)\delta X^i = 0\}$，它表示线性化系统动力学的解集。将关于 $\mathcal{X}^i$ 的示性函数与主车代价相加，得到

$$F^i(\delta X^i) := \bar{F}^i(\delta X^i) + \mathcal{I}_{\mathcal{X}^i}(\delta X^i). \tag{16}$$

显然，当且仅当 L_2^i 中的所有海森矩阵均为半正定矩阵时，函数 $F^i(\delta X^i)$才是凸函数。

对于成对碰撞避免项，直接进行二阶泰勒展开并计算海森矩阵需要大量计算，导致运行速度缓慢。此外，这种近似通常无法得到凸函数。相反，我们采用高斯-牛顿近似，这不仅降低了计算难度，还能保证凸性。我们定义 $\delta x^i = (\delta x_0^i,\ \delta x_1^i,\ .....,\ \delta x_N^i) \in \mathbb{R}^{(T+1)n}$，它通过从 δX^i中消去所有控制输入得到，并且

$$l_\tau = [l_\tau^{ij}]_{1\le i<j\le N} \in \mathbb{R}^{\frac{N(N-1)}{2}}, l_\tau^{ij} = \sqrt{\beta}\min\{d^{ij} - d_{\text{safe}}, 0\} \tag{17}$$

使得 l_τ 是所有满足 $1 \le i < j \le N$的可能$(i,\ j)$对所对应的 C_τ^{ij}平方根构成的串接列向量。根据此定义，时间 τ 的所有碰撞规避代价之和可简单地改写为 l_τ的平方 L_2-范数，即

$$\sum_{1\le i<j\le N} C_\tau^{ij}(x_\tau^i, x_\tau^j) = ||l_\tau||^2. \tag{18}$$

此外，l_τ关于 δx_τ^i的雅可比矩阵定义为 $J_\tau^i = \partial l_\tau / \partial x_\tau^i \in \mathbb{R}^{\frac{N(N-1)}{2}\times n}$。

注 2. 雅可比矩阵 $J_\tau^i \in \mathbb{R}^{\frac{N(N-1)}{2}\times n}$ 至多包含 $N-1$ 个非零行。这是因为，在总共 $N(N-1)/2$个碰撞规避项中，只有 $N-1$个与车辆 i相关。因此，J_τ^i 是稀疏的。

此外，我们定义 $l = (l_0,\ l_1,\ .....,\ l_T) \in \mathbb{R}^{\frac{N(N-1)}{2}(T+1)}$。根据上述定义，我们可以围绕当前名义轨迹对 l_τ 进行线性近似，从而得到碰撞规避代价的高斯-牛顿近似。我们将扰动后的碰撞规避代价定义为 $\bar{C}_\tau^{ij}(\delta x_\tau^i, \delta x_\tau^j) = C_\tau^{ij}(x_\tau^i + \delta x_\tau^i,\ x_\tau^j + \delta x_\tau^j)$，使得

$$\sum_{1\le i<j\le N} \bar{C}_\tau^{ij}(\delta x_\tau^i, \delta x_\tau^j) \approx \left\| \sum_{i=0}^{N} J_\tau^i \delta x_\tau^i + l_\tau \right\|^2 \tag{19}$$

将式 (19) 对所有时间戳求和可得

$$\begin{aligned}
&\sum_{\tau=0}^{T}\sum_{1\le i<j\le N}\bar{C}_\tau^{ij}(\delta x_\tau^i,\delta x_\tau^j)\approx\sum_{\tau=0}^{T}\left\|\sum_{i=0}^{N}J_\tau^i\delta x_\tau^i+l_\tau\right\|^2\\
&=\left\|(\sum_{i=0}^{N}J_0^i\delta x_0^i,\sum_{i=0}^{N}J_1^i\delta x_1^i,...,\sum_{i=0}^{N}J_T^i\delta x_T^i)+l\right\|^2\\
&=\left\|\sum_{i=0}^{N}(J_0^i\delta x_0^i,J_1^i\delta x_1^i,...,J_T^i\delta x_T^i)+l\right\|^2\\
&=\left\|\sum_{i=1}^{N}\text{blocdiag}\{J_0^i,J_1^i,...,J_T^i\}\delta x^i+l\right\|^2.
\end{aligned}\tag{20}$$

为了在表达式中也纳入控制输入，我们定义 $\tilde{J}^i=\text{blocdiag}\{[J_0^i\ \underbrace{\mathbf{0}\ \cdots\ \mathbf{0}}_{N(N-1)^2\times m}],\ .....,\ [J_{T-1}^i\ \underbrace{\mathbf{0}\ \cdots\ \mathbf{0}}_{N(N-1)^2\times m}],\ J_T^i\}$，由此可得

$$\sum_{\tau=0}^{T}\sum_{1\le i<j\le N}\bar{C}_\tau^{ij}(\delta x_\tau^i,\delta x_\tau^j)\approx G_1\left(\sum_{i=1}^{N}\tilde{J}^i\delta X^i\right)\tag{21}$$

其中

$$G_1(x):=||x+l||^2\tag{22}$$

是一个凸函数。

此外，为纳入对控制输入施加的盒约束，我们定义凸集 $\mathcal{X}^u=\{u_b|u-u\preceq u_b\preceq u-u\}$，其中 $u=((u_0^1,\ .....,\ u_{T-1}^1).....,\ (u_0^N,\ .....,\ u_{T-1}^N))$ 和 $\underline{u}=((u_0^1,\ .....,\ u_{T-1}^1)...\ ...,\ (u_0^N,\ .....,\ u_{T-1}^N))$ 分别表示所有控制输入的下界和上界，而 $u=((u_0^1,\ .....,\ u_{T-1}^1).....,\ (u_0^N,\ .....,\ u_{T-1}^N))$ 表示车辆当前名义轨迹中的所有控制输入。显然，$\delta u=((\delta u_0^1,\ .....,\ \delta u_{T-1}^1).....,\ (\delta u_0^N,\ .....,\ \delta u_{T-1}^N))$ 必须满足 $\delta u\in\mathcal{X}^u$，以使更新后的控制输入 $u+\delta u$ 满足盒约束。

定义矩阵 $O^i=\text{blocdiag}\{[\underbrace{\mathbf{0}^{m\times n}I^m],...,[\mathbf{0}^{m\times n}I^m]}_{T},\mathbf{0}^{m\times n}\}\in\mathbb{Z}^{Tm\times M}$，使得 $\delta u^i=O^i\delta X^i$；此外，定义 $\tilde{O}^i=[\underbrace{\mathbf{0}^{M\times Tm},...,\mathbf{0}^{M\times Tm}}_{i-1},O^{i\top},\underbrace{\mathbf{0}^{M\times Tm},...,\mathbf{0}^{M\times Tm}}_{N-i}]^\top\in\mathbb{R}^{NTm\times M}$，使得 $\delta u=\sum_{i=1}^{N}\tilde{O}^i\delta X^i$。因此，所有盒约束都可以通过以下凸指示函数处理

$$G_2(\delta u):=\mathcal{I}_{\mathcal{X}^u}(\delta u)=\mathcal{I}_{\mathcal{X}^u}\left(\sum_{i=1}^{N}\tilde{O}^i\delta X^i\right)\tag{23}$$

使得当 $u+\delta u$ 违反盒约束时，会在总体代价中加入无穷大的代价。

接下来，我们将说明，可以将盒约束代价添加到碰撞规避代价中，同时仍保持与式 (21) 相同的形式。定义一个新的矩阵 $J^i=[\tilde{J}^{i\top}\tilde{O}^{i\top}]^\top$ 和一个新的凸函数

$$G(x)=G\left(\begin{bmatrix}x_{[1]}\\x_{[2]}\end{bmatrix}\right):=G_1(x_{[1]})+G_2(x_{[2]})\tag{24}$$

其中 $x\in\mathbb{R}^{\frac{1}{2}N(N-1)(T+1)+NTm}$，$x_{[1]}$、$x_{[2]}$ 分别表示 x 的前 $\frac{1}{2}N(N-1)(T+1)$ 个元素和最后 NTm 个元素。然后

$$\begin{aligned}
G\left(\sum_{i=1}^{N}J^i\delta X^i\right)&=G\left(\begin{bmatrix}\sum_{i=1}^{N}\tilde{J}^i\delta X^i\\\sum_{i=1}^{N}\tilde{O}^i\delta X^i\end{bmatrix}\right)\\
&=G_1\left(\sum_{i=1}^{N}\tilde{J}^i\delta X^i\right)+G_2\left(\sum_{i=1}^{N}\tilde{O}^i\delta X^i\right)
\end{aligned}\tag{25}$$

表示所有碰撞规避代价以及用于强制满足盒约束的示性函数之和。将其加入主车代价后，可得以下无约束优化问题

$$\min_{\delta X^1,...,\delta X^N}\sum_{i=1}^{N}F^i(\delta X^i)+G\left(\sum_{i=1}^{N}J^i\delta X^i\right)\tag{26}$$

这是式（9）的凸重构。

B. 去中心化与可并行化优化框架

本节提出一种基于对偶共识交替方向乘子法的新型去中心化且可并行化的优化框架，用于求解 (9)。为完整起见，下面重述对偶共识交替方向乘子法的相关预备知识（详见 [30], [31] ）。

凸优化问题 (26) 可重写为

$$\begin{aligned}
\min_{w,\delta X^1,...,\delta X^N}\ &\sum_{i=1}^{N}F^i(\delta X^i)+G(w)\\
s.t.\ &\sum_{i=1}^{N}J^i\delta X^i=w.
\end{aligned}\tag{27}$$

对于该问题，由于约束为仿射约束，因此满足 Slater 条件，从而强对偶性成立。

(27) 的拉格朗日函数为

$$\mathcal{L}(\delta X,w,y)=\sum_{i=1}^{N}F^i(\delta X^i)+G(w)+y^\top\left(\sum_{i=1}^{N}J^i\delta X^i-w\right)\tag{28}$$

其中 $y\in\mathbb{R}^{\frac{1}{2}N\ (N-1)\ \ (T+1)\ +NTm}$ 为对偶变量。对偶函数可表示为

$$h(y)=\inf_{\delta X,w}\mathcal{L}(\delta X,w,y)=-\sum_{i=1}^{N}F^{i*}(-J^{i\top}y)-G^*(y)\tag{29}$$

其中，F^{i*} 和 G^* 分别是 F^i 和 G 的共轭函数。对偶问题 $\max_y h(y)$ 是一个分解后的共识优化问题，所有智能体共同对优化变量 y 作出决策。因此，可以采用共识交替方向乘子法 [31, 算法 1]，其中 $\theta_i:=F_i^*\circ(-J^{i\top})$，且 $\xi_i:=\frac{1}{N}G^*$。

Algorithm 1 Consensus ADMM [31]

1: **choose** $\sigma, \rho > 0$
2: **initialize** for all $i \in \mathcal{N}$: $p^{i,0} = y^{i,0} = z^{i,0} = s^{i,0} = 0$
3: **repeat**: for all $i \in \mathcal{N}$
4: Broadcast $y^{i,k}$ to all other vehicles
5: $p^{i,k+1} = p^{i,k} + \rho \sum_{j \neq i}(y^{i,k} - y^{j,k})$
6: $s^{i,k+1} = s^{i,k} + \sigma(y^{i,k} - z^{i,k})$
7: $y^{i,k+1} = \arg\min_{y^i}\{\theta^i(y^i) + y^{i\top}(p^{i,k+1} + s^{i,k+1}) + \frac{\sigma}{2}||y^i - z^{i,k}||^2 + \rho \sum_{j \neq i}||y^i - \frac{y^{i,k}+y^{j,k}}{2}||^2\}$
8: $z^{i,k+1} = \arg\min_{z^i}\{\xi^i(z^i) - z^{i\top}s^{i,k+1} + \frac{\sigma}{2}||z^i - y^{i,k+1}||^2\}$
9: **until** termination criterion is satisfied

具体而言，算法1的步骤7通过以下方式执行：

$$y^{i,k+1} = \frac{1}{\sigma + 2\rho d_i}(J^i \delta X^{i,k+1} + r^{i,k+1}), \tag{30}$$

其中，$r^{i,k+1} = \rho \sum_{j \neq i}(y^{i,k} + y^{j,k}) + \sigma z^{i,k} - p^{i,k+1} - s^{i,k+1}$.

$$\delta X^{i,k+1} = \arg\min_{\delta X^i}\{F^i(\delta X^i) + \frac{||J^i \delta X^i + r^{i,k+1}||^2}{2(\sigma + 2\rho d_i)}\} \tag{31}$$

是一个辅助变量，并收敛到式（26）的极小化解 [31]。d_i 是节点 i 的度，在我们的情形中等于 $N-1$。同时，算法1的第8步由

$$z^{i,k+1} = \frac{s^{i,k+1}}{\sigma} + y^{i,k+1} - \frac{1}{N\sigma}\text{Prox}_G^{\frac{1}{N\sigma}}(N(s^{i,k+1} + \sigma y^{i,k+1})). \tag{32}$$

基于对现有对偶共识交替方向乘子法的前述重新表述，我们提出了以下针对我们优化问题的对偶更新和原始解恢复方法。具体而言，式（31）和式（32）需要进一步讨论。

1）对偶更新：在式（32）中，$z^{i,k+1}_{:}$的前 $\frac{1}{2}N(N-1)(T+1)$个元素和最后 NTm 个元素分别对应碰撞规避代价和盒约束，因此应分别处理。最后一项为

$$\begin{aligned}
&\text{Prox}_G^{\frac{1}{N\sigma}}(N(s^{i,k+1} + \sigma y^{i,k+1})) \\
&= \arg\min_z\{G(z) + \frac{1}{2N\sigma}||z - N(s^{i,k+1} + \sigma y^{i,k+1})||^2\} \\
&= (\arg\min_{z_{[1]}}\{||z_{[1]} + l||^2 \\
&+ \frac{1}{2N\sigma}||z_{[1]} - N(s^{i,k+1}_{[1]} + \sigma y^{i,k+1}_{[1]})||^2\}, \\
&\arg\min_{z_{[2]}}\{I_{\mathcal{X}^u}(z_{[2]}) + \frac{1}{2N\sigma}||z_{[2]} - N(s^{i,k+1}_{[2]} + \sigma y^{i,k+1}_{[2]})||^2\}).
\end{aligned} \tag{33}$$

因此，我们将该问题分解为两个并行子问题。第一个子问题是一个简单的无约束二次规划，其解析解为

$$z_{[1]} = \frac{N}{2N\sigma + 1}(s^{i,k+1}_{[1]} + \sigma y^{i,k+1}_{[1]} - 2\sigma l). \tag{34}$$

将其代回（32）式，可得

$$z^{i,k+1}_{[1]} = \frac{2}{2N\sigma + 1}(N s^{i,k+1}_{[1]} + N\sigma y^{i,k+1}_{[1]} + l) \tag{35}$$

它对 s、y和 l进行加权求和。

第二个子问题等价于以下约束优化问题：

$$\begin{aligned}
&\min_{z_{[2]}} ||z_{[2]} - N(s^{i,k+1}_{[2]} + \sigma y^{i,k+1}_{[2]})||^2 \\
&s.t.\ z_{[2]} \in \mathcal{X}^u,
\end{aligned} \tag{36}$$

且 $z_{[2]}$ 是（36）式的极小化解。为求解 $z_{[2]}$，我们首先引入以下投影的定义。

定义2。给定 $\mathcal{D}$中的一个闭凸集 $\mathcal{X}$，对于每个 $v \in \mathcal{D}$，以下问题 $\min_x\{||x-v||^2 | x \in \mathcal{X}\}$均存在唯一的极小化解，该解称为 v在$\mathcal{X}$ 上的投影，记为 $\text{Proj}_{\mathcal{X}}(v)$。

根据该定义，第二个子问题的解为

$$z^{i,k+1,*}_{[2]} = \text{Proj}_{\mathcal{X}^u}(N(s^{i,k+1}_{[2]} + \sigma y^{i,k+1}_{[2]})). \tag{37}$$

由于凸集 $\mathcal{X}^u$ 是通过对向量 u_b的每个元素分别施加盒约束构造的，因此，该投影的结果只需将 N的每个元素（$s^{i,k+1}_{[2],:} + \sigma y^{i,k+1}_{[2],:}$）分别限制在集合 $\mathcal{X}^u$内，这可以通过逐元素最小-最大运算实现。将其代回（32）式，可得

$$z^{i,k+1}_{[2]} = \frac{s^{i,k+1}}{\sigma} + y^{i,k+1} - \frac{1}{N\sigma}z^{i,k+1,*}_{[2]}. \tag{38}$$

最后，我们得到 $z^{i,k+1}_{:} = (z^{i,k+1}_{[1],:}, z^{i,k+1}_{[2],:})$。

这些结果表明，z 通过逐元素加法和最小最大运算进行更新，因此可以高效地执行。方程（35）、（37）、（38）和（30）构成了对偶变量 y 和 z的更新步骤。

2）原始解恢复：展开式（31）中的平方项可得

$$\begin{aligned}
\delta X^{i,k+1} &= \arg\min_{\delta X^i}\{F^i(\delta X^i) \\
&+ \frac{1}{2(\sigma + 2\rho d_i)}(\delta X^{i\top} J^{i\top} J^i \delta X^i + 2r^{i,k+1\top} J^i \delta X^i)\}
\end{aligned} \tag{39}$$

利用雅可比矩阵 $J^i = [\tilde{J}^\top \tilde{O}^\top]^\top$的稀疏结构，我们可以得到 δX^i的二阶项如下

$$\begin{aligned}
\delta X^{i\top} J^{i\top} J^i \delta X^i &= \delta X^{i\top} \tilde{J}^{i\top} \tilde{J}^i \delta X^i + \delta X^{i\top} \tilde{O}^{i\top} \tilde{O}^i \delta X^i \\
&= \delta X^{i\top} \tilde{J}^{i\top} \tilde{J}^i \delta X^i + \delta u^{i\top} \delta u^i \\
&= \sum_{\tau=0}^{T} \delta x_\tau^{i\top} J_\tau^{i\top} J_\tau^i \delta x_\tau^i + \sum_{\tau=0}^{T-1} \delta u_\tau^{i\top} \delta u_\tau^i.
\end{aligned} \tag{40}$$

类似地，对于 δX^i 的一阶项，我们有

$$\begin{aligned}
2r^{i,k+1\top} J^i \delta X^i &= 2r^{i,k+1\top}_{[1]} \tilde{J}^i \delta X + 2r^{i,k+1\top}_{[2]} \tilde{O}^i \delta X \\
&= 2r^{i,k+1\top}_{[1]} \tilde{J}^i \delta X + 2\tilde{r}^{i,k+1\top}_{[2]} \delta u^i \\
&= \sum_{\tau=0}^{T} 2r^{i,k+1\top}_{[1],\tau} J_\tau^i \delta x_\tau^i + \sum_{\tau=0}^{T-1} 2\tilde{r}^{i,k+1\top}_{[2],\tau} \delta u_\tau^i.
\end{aligned} \tag{41}$$

具体来说，$r_{[1],\tau}$ 表示$r_{[1]}$ 中从行号 $N(N-1)\tau/2$到行号 $N(N-1)(\tau+1)/2-1$的部分，$\tilde{r}^i_{[2]}$ 表示 $r^i_{[2]}$ 中从行号 $(i-1)Tm$

到行号 $iTm-1$的部分，并且 $\tilde{r}^i_{[2],\ \tau}$表示 $\tilde{r}^i_{[2]}$中从行号 τm到（$\tau+1$）$m-1$（含）的部分。

将 (16)、(40) 和 (41) 代回 (39)，可以证明 $\delta X^{i,}_{\ k+1}$ 是以下最小化问题的优化解

$$\begin{aligned}\min_{\delta x^i_0,\delta u^i_0,\ldots,\delta x^i_T}\ & \sum_{\tau=0}^{T}\frac{1}{2}\delta x_\tau^{i\top}(C^i_{\tau,xx}+\frac{J_\tau^{i\top}J^i_\tau}{2(\sigma+2\rho d_i)})\delta x^i_\tau\\ &+\delta x_\tau^{i\top}(C^i_{\tau,x}+\frac{J_\tau^{i\top}r^{i,k+1}_{[1],\tau}}{\sigma+2\rho d_i})\\ &+\sum_{\tau=0}^{T-1}\frac{1}{2}\delta u_\tau^{i\top}(C^i_{\tau,uu}+\frac{I}{2(\sigma+2\rho d_i)})\delta u^i_\tau\\ &+\delta u_\tau^{i\top}(C^i_{\tau,u}+\frac{\tilde{r}^{i,k+1}_{[2],\tau}}{\sigma+2\rho d_i})\\ s.t.\ &\delta x^i_{\tau+1}=A^i_\tau\delta x^i_\tau+B^i_\tau\delta u^i_\tau.\end{aligned}\tag{42}$$

值得注意的是，(42) 是一个标准的 LQR最优控制问题，可以通过动态规划高效求解。

注释 3. 引入问题 (42) 后，原先耦合的优化问题实质上实现了解耦，因为每辆车求解的最优控制问题仅涉及自身的状态变量和输入变量。其他车辆对其自身轨迹的影响则体现在包含 J 和 r 的附加项中。每辆车迭代求解线性二次型调节问题 (42)，直至交替方向乘子法终止，这相当于在集中式 iLQR 求解器中执行一次反向遍历，其中涉及所有车辆的状态变量。基于围绕当前名义轨迹对原始问题进行迭代凸重构，本文提出了一种分布式 iLQR 算法。

基于以上讨论，我们提出算法2，即一种用于求解问题 (9) 的分布式 iLQR 算法。每辆车执行两个循环：外循环对问题 (9) 进行凸重构，这与集中式 iLQR 求解器的外循环类似。内循环通过交替方向乘子法迭代以分布式且可并行化的方式求解引入的凸问题；在此过程中，每辆车求解一个线性二次型调节问题 (42)，并更新对偶变量 y、z、p 和 s。

备注4。需要注意的是，在算法2的步骤7中，只有 p 和 s 会被重置。先前交替方向乘子法迭代得到的 y 和 z的值会保留到下一轮交替方向乘子法迭代中，并用作初始化。其背后的原理是，在连续的凸重构过程中，所诱导的凸优化问题变化不大，只会缓慢漂移。因此，先前凸重构循环得到的对偶变量 y 和 z 可作为下一轮循环中最优 y^* 和 z^* 的良好初始猜测，从而大幅加快交替方向乘子法的收敛。

C. 动态可行轨迹更新

需要注意的是，通过求解问题（42）得到的车辆 i 的 δX^i 仅对线性化系统动力学可行，但对原始非线性

算法2 基于对偶共识交替方向乘子法的分布式 iLQR（针对车辆 i）

1: **初始化** $\{x^i_\tau,u^i_\tau\}^T_{\tau=0}$
2: **初始化** 所有 $i\in\mathcal{N}$:
$p^{i,0}=y^{i,0}=z^{i,0}=s^{i,0}=0$
3: **选择** $\sigma,\ \rho>0$
4: **重复**:
5: 发送 $\{x^i_\tau\}^T_{\tau=1}$，接收 $\{x^j_\tau\}^T_{\tau=1}$ 来自 $j\in\mathcal{N}-\{i\}$
6: 计算 $l,\ J^i,\ \{A^i_\tau\}^{T-1}_{\tau=0},\ \{B^i_\tau\}^{T-1}_{\tau=0}$
7: **重置** $p^{i,0}=s^{i,0}=0$
8: **重复**:
9: 发送 $y^{i,k}$，接收 $y^{j,k}$，来自 $j\in\mathcal{N}-\{i\}$
10: $p^{i,k+1}=p^{i,k}+\rho\sum_{j\neq i}(y^{i,k}-y^{j,k})$
11: $s^{i,k+1}=s^{i,k}+\sigma(y^{i,k}-z^{i,k})$
12: $r^{i,\ k+1}=\rho\sum_{j\neq i}(y^{i,k}+y^{j,k})$
$+\sigma z^{i,\ k}-p^{i,\ k+1}-s^{i,\ k+1}$
13: 通过求解线性二次型调节问题 (42) 计算 $\delta X^{i,\ k+1}$
14: 使用 (30) 更新 $y^{i,\ k+1}$
15: 使用 (35) 和 (37) 更新 $z^{i,\ k+1}$
16: **直至**满足终止准则
17: 更新 $\{x^i_\tau,u^i_\tau\}^T_{\tau=0}$
18: **直至**满足终止准则

系统动力学并不可行。因此，直接将 δX^i 与名义轨迹相加会导致违反非线性动力学约束。为解决这一问题，我们提出了一种动力学可行更新策略，对应算法2的步骤17。

通过动态规划求解问题 (42)，可得到一系列反馈控制矩阵 $\{k^i_\tau,K^i_\tau\}^{T-1}_{\tau=0}$，使得

$$\delta u^i_\tau=k^i_\tau+K^i_\tau\delta x^i_\tau,\forall\tau\in\mathcal{T}.\tag{43}$$

按照典型的 iLQR前向遍历，我们更新名义轨迹如下:

$$\begin{aligned}u^i_\tau&=\hat{u}^i_\tau+\alpha k^i_\tau+K^i_\tau(x^i_\tau-\hat{x}^i_\tau)\\ u^i_\tau&\leftarrow\text{clip}(u^i_\tau)\\ x^i_{\tau+1}&=f(x^i_\tau,u^i_\tau),\end{aligned}\tag{44}$$

其中 α 为线搜索参数。在每个时间戳对输入进行截断，以确保输入严格满足盒约束。

基于前述讨论，我们提出算法 3，为每辆车执行动态可行轨迹更新。为更好地实现收敛，我们采用线搜索方法。在算法 3 中，每辆车依次遍历同一组候选 α 所对应的线搜索参数 α，并通过式 (44) 更新其轨迹，以生成一组候选轨迹。随后，将这些候选轨迹广播给所有其他车辆。接着，每辆车找出与产生最低总体代价的最优 α 相对应的轨迹，并用它更新当前轨迹。需要注意的是，在算法 3 中，同步仅发生在步骤 5，因此对整个算法的影响较小。

D. 复杂度分析

这里简要讨论所提算法算法2的复杂度。在我们的仿真中，最

算法 3 带线搜索的动态可行更新（针对车辆 i）

初始化 t $\underline{list}^i, c$ $\underline{list}^i = empty$
对于 α **属于** $\underline{\alpha}$ $list$:
    计算 $traj$，使用 α （44）
    将 $traj$ 追加到 t $list^i$
发送 t_list^i，从 $j \in \mathcal{N} - \{i\}$接收 t $list^j$
对i 遍历 $len(\underline{\alpha}_ list)$:
    $t_set = empty$
    对 j **遍历** $\mathcal{N}$:
        将 t $list^j_[i]$ 追加到 t_set
    计算 t set的 $cost_$
    将 $cost$追加到 c_list
$index$ = 参数最小值($\underline{c}$ $list$)
$current\ trajectory \leftarrow t_list^i[index]$

计算量最大的部分是算法2的第13步，该步骤需要求解一个LQR最优控制问题。显然，对于每辆车辆，问题（42）仅涉及该车辆的局部状态变量和控制输入，因此其规模不会随着网联自动驾驶汽车数量的增加而扩大。换言之，我们可以得出结论，求解问题（42）的复杂度为 $O(1)$。

作为对比，以集中式方式求解问题（9）需要处理一个LQR最优控制问题，其状态空间随网联自动驾驶汽车数量线性增长，从而导致可扩展性较差。通常， O（N^3）复杂度会随之产生。

需要注意的是，每辆车辆都需要以迭代方式求解问题（42）。假设外循环的迭代次数为 N_1，内循环的迭代次数为 N_2，则可以得出，仅考虑步骤13时，算法2的复杂度为 O（$N_1 \cdot N_2$）。

从理论上讲，步骤10和12等其他步骤具有更高的复杂度，但它们仅涉及逐元素求和等非常简单的操作，因此在我们的仿真中只占用可忽略的时间。

IV. 仿真结果

A. 车辆模型

我们假设仿真中涉及的所有车辆都具有相同的动力学特性。参见 [18],单车动力学由以下方程描述:

$$\begin{aligned} p_{x,\tau+1} &= p_{x,\tau} + f_r(v_\tau, \delta_\tau)\cos(\theta_\tau), \\ p_{y,\tau+1} &= p_{y,\tau} + f_r(v_\tau, \delta_\tau)\sin(\theta_\tau), \\ \theta_{\tau+1} &= \theta_\tau + \arcsin(\frac{\tau_s v_\tau \sin(\delta_\tau)}{b}), \\ v_{\tau+1} &= v_\tau + \tau_s a_\tau. \end{aligned} \tag{45}$$

(45) 描述了一个离散时间车辆模型，其状态向量为 (p_x, p_y, θ, v)，输入向量为 (δ, a)。p_x 和 p_y 分别表示车辆中心在全局笛卡尔坐标系中的 X 和 Y 坐标，θ 表示车辆相对于全局笛卡尔坐标系 X 轴正方向的航向角，v 表示车辆的速度，δ 表示前轮的转向角，而 a 表示加速度。此外，我们使用下标 $[\cdot]_\tau$ 表示时间戳为 τ 的变量，其中 $\tau \in \mathcal{T}$，τ_s 为时间间隔。函数 $f_r(v, \delta)$ 定义为

$$f_r(v,\delta) = b + \tau_s v\cos(\delta) - \sqrt{b^2 - (\tau_s v \sin(\delta))^2} \tag{46}$$

其中，b 表示车辆轴距。

将所有车辆视为一个整体系统，其系统动力学可通过简单地将各单车动力学堆叠起来得到，从而构建出一个包含 $4N$ 个状态变量和 $2N$ 个控制输入的模型。

B. 一般设置

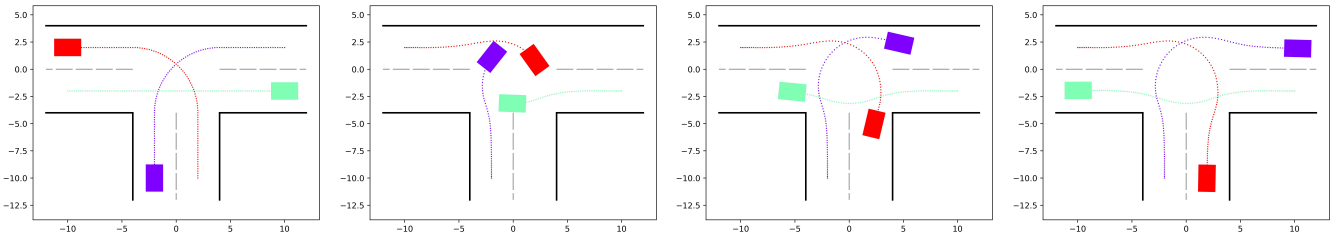
本文考虑与 [33],相同的场景，包括一个包含 3 辆车的丁字路口和一个包含 12 辆车的交叉路口。我们分别实现所提算法的单进程版本和多进程版本，其中进程数与车辆数相等。此外，我们还实现了三种基线方法进行比较，包括带对数障碍函数的集中式 iLQR 求解器 [19], 、IPOPT 求解器和序列二次规划求解器，后两种求解器由 CasADi [36]提供。所有算法均使用 Python 3.7 实现，并运行在配备 $2\times$ Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz 的服务器上。对于多进程实现，每个进程绑定到 CPU 上的一个（逻辑）核心，并通过共享内存进行通信。

具体而言，我们将每辆车的长和宽分别设为 2.50m 和 1.60m。输入转向角限制在 ± 0.6rad 之间，加速度则在 $[-3.0, 1.5]$ m/s^2 以内。主车代价中的权重矩阵设为 $Q = \text{diag}(1,1,0,0)$ 和 $R = \text{diag}(1,1)$。离散动力学的时间间隔设为 $\tau_s = 0.1$s，时域长度设为 $T = 100$。所有避碰代价的安全距离为 $d_{\text{safe}} = 5.5$m，主车代价与碰撞规避代价之间的缩放因子为 $\beta = 1.44$。所有车辆的轨迹均以零输入初始化。所提方法和集中式 iLQR 求解器的终止条件均根据总体代价的变化设定。当连续两次迭代间总体代价的绝对变化小于 1 时，两种算法均终止。

C. 主要结果

1) 场景 *1*： 我们首先考虑一个有三辆车的 T形路口场景。如图1(a)所示，车辆用不同颜色的矩形表示，每条虚线都是与同色车辆对应的参考轨迹，每个点表示特定时间戳下车辆中心的参考位置。车辆需要尽可能贴近参考轨迹通过丁字路口，并避免发生碰撞。

对于场景 1，我们设定 $\sigma = 0.1$ 和 $\rho = 0.01$。由于采用了备注4中描述的初始化方案，ADMM 的停止准则（算法2第16步）可以设为经过固定的少量迭代后终止。此处，我们将 ADMM 迭代次数设为 2。结果如图1(b)-(d)所示。每个子图对应一个特定的时间戳，矩形表示车辆的位姿，



(a) $\tau = 0$ (b) $\tau = 50$ (c) $\tau = 75$ (d) $\tau = 99$

图 1。 所提方法在场景 1 中的仿真结果，时间戳为 $\tau = 0$、 $\tau = 50$、 $\tau = 75$ 和 $\tau = 99$。

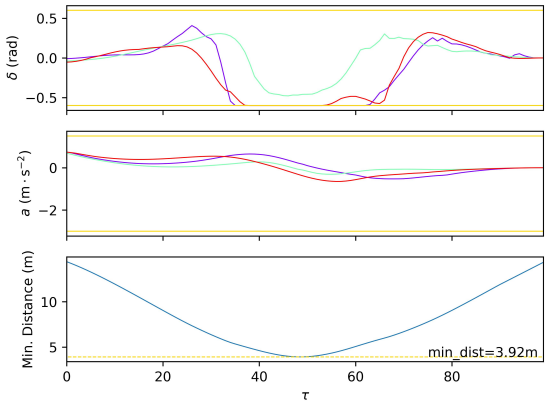


图 2. 场景 1 中 3 辆车辆的转向角和加速度，以及车辆中心之间的最小距离。

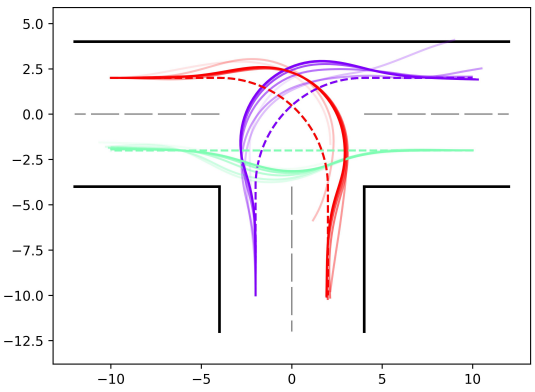


图 3. 场景 1 中 3 辆车辆轨迹的迭代过程。

虚线表示车辆之前的轨迹。三辆车均成功抵达目的地，沿着平滑且可行的轨迹行驶，同时彼此保持安全距离以避免碰撞。图2展示了输入和最小距离。盒约束均得到满足，车辆中心之间的最小距离为 3.92m，这证明这些轨迹不会发生碰撞。单进程和多进程实现得出的数值结果相同。

我们的算法经过 7 次迭代后停止。图 3 展示了车辆轨迹随迭代变化的情况。相同颜色的一组轨迹对应同一辆车辆，其中越实线的轨迹表示迭代次数越多，对应于

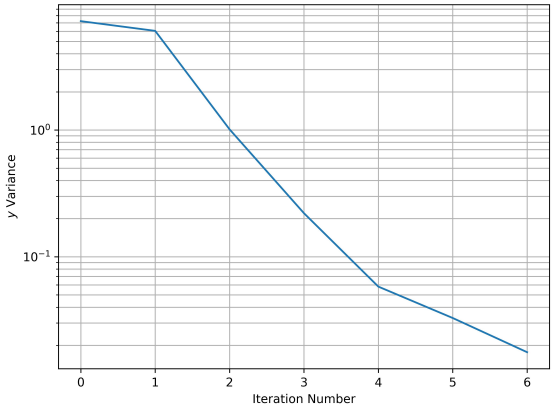
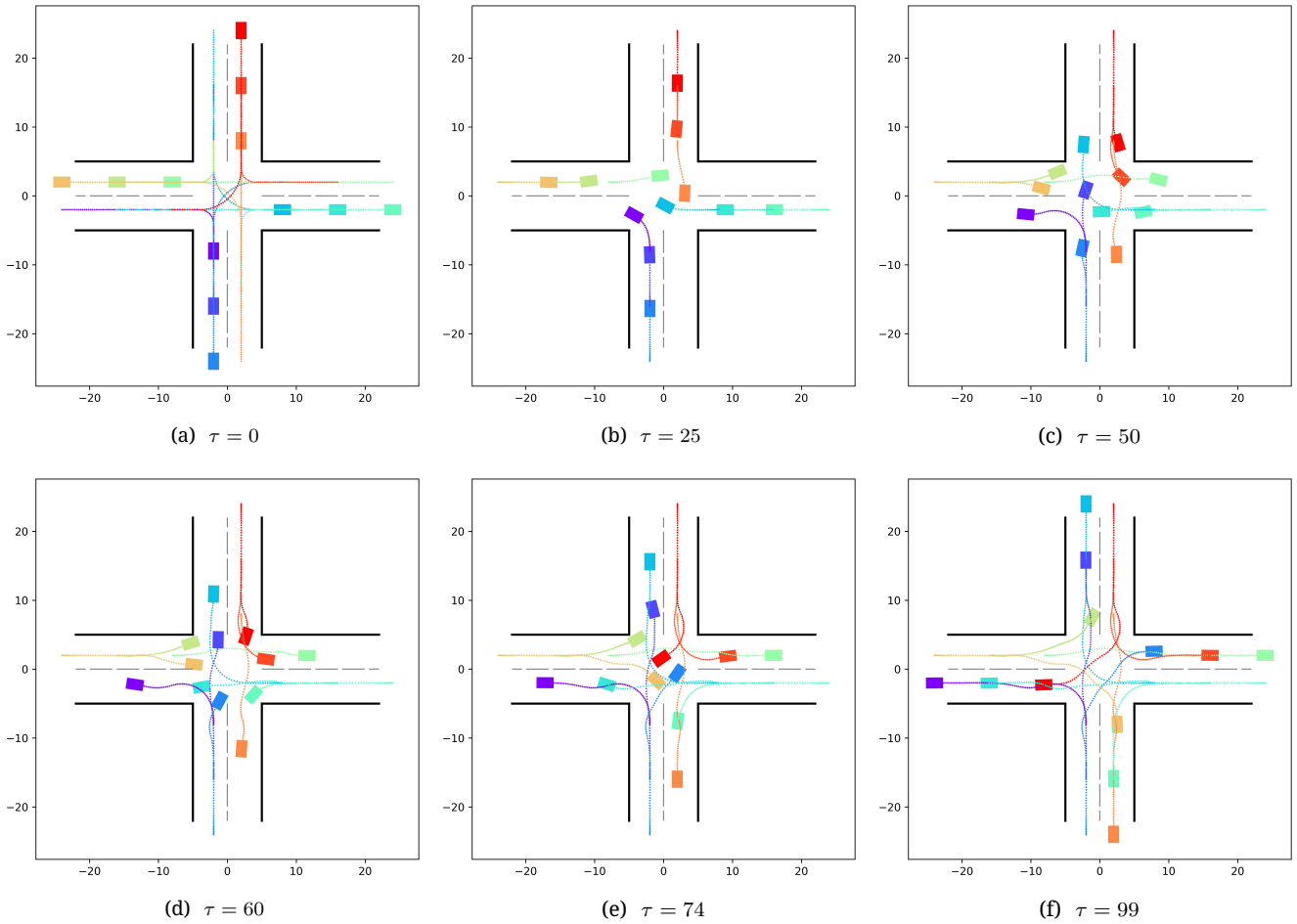


图 4. 场景 1 中 $\{y^i\}$ 随迭代次数的变化方差。

更高的迭代次数。图 3 定性地展示了轨迹如何通过迭代收敛到最优轨迹。为了进一步展示算法的收敛性，我们采用以下结论：随着算法收敛，对偶变量集合 $\{y^i\}$ 应收敛到同一个向量 y^* 。因此，我们计算 $\{y^i\}$ 中每个元素在 i 上的方差，并求这些方差的均值。我们绘制了方差均值随迭代次数的变化情况。结果如图 4 所示，采用对数尺度，显示 $\{y^i\}$ 的方差单调递减，清楚地证明了我们算法的收敛性。

表 I 第一行展示了全部五种实现的计算时间。我们算法的多进程实现仅需 0.035s 即可完成，是所有实现中最快的。与集中式 iLQR 求解器相比，其速度约为 $1.5\times$ 倍；与我们方法的单进程版本相比，速度约为 $2.4\times$ 倍；与 IPOPT 求解器相比，速度约为 $9.3\times$ 倍；与序列二次规划求解器相比，速度约为 $23.4\times$ 倍。需要指出的是，所提方法的单进程实现比集中式 iLQR 求解器更慢，这意味着所提算法并未减少总体计算量，而是提供了一个分布式优化框架，使得可以通过并行计算加速优化过程。

由于终止准则不同，各求解器得到的最终解质量有所差异。表 III 展示了各解的总体代价。所提算法的总体代价与集中式 iLQR 求解器几乎相同，仅略高于 IPOPT 求解器，这表明



(a) $\tau = 0$ (b) $\tau = 25$ (c) $\tau = 50$

(d) $\tau = 60$ (e) $\tau = 74$ (f) $\tau = 99$

图 5. 场景 2 中所提方法在时间戳 $\tau = 0$、 $\tau = 25$、 $\tau = 50$、 $\tau = 60$、 $\tau = 74$ 和 $\tau = 99$ 下的仿真结果。

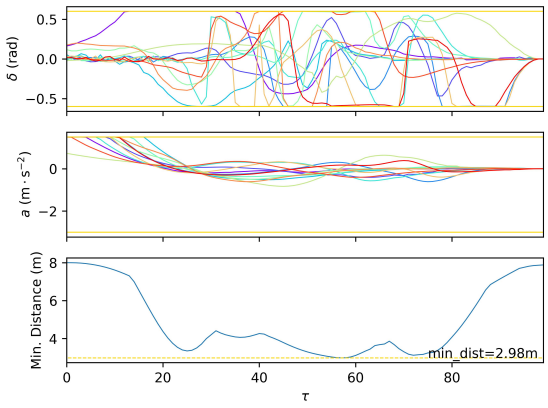


图 6. 场景 2 中 12 辆车辆的转向角和加速度，以及车辆中心之间的最小距离。

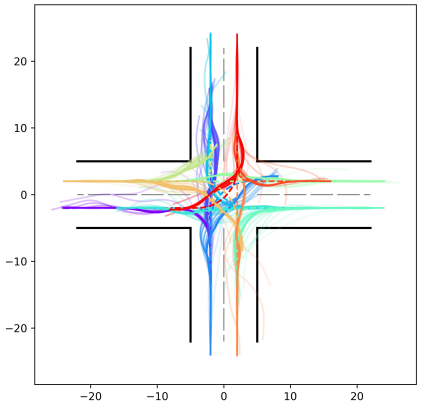


图 7. 场景 2 中 12 辆车辆的轨迹迭代过程。

解的质量相当。与此同时，SQP 求解器未能获得无碰撞解，其总体代价远高于其他求解器，这也反映了这一点。

2) 场景 *2*：为进一步展示所提方法在计算效率方面的优越性，我们考虑一个规模更大的场景。如图 5(a) 所示，场景 2 是一个有 12 辆车辆同时通行的交叉路口。我们发现，随着车辆数量增加，为了加快收敛，参数 σ 和ρ 应当减小。因此，我们将 σ 和 ρ 缩小至 $\sigma = 0.01$ 和 ρ =0。001，并将交替方向乘子法迭代次数设为 3。仿真结果如图 5(b)-(f) 所示。同样得到了平滑的轨迹。图 6 显示，输入满足盒约束，最小距离为 2。98m，这也保证了安全性。所提算法经过 24 次迭代

表 I 场景 1 和 2 中所提方法、集中式 iLQR、IPOPT 和序列二次规划的计算时间比较

	所提方法		集中式 iLQR	IPOPT	SQP
	单进程	多进程			
场景 1	0.084 秒	**0.035 秒**	0.052 秒	0.327 秒	0.818 秒
场景 2	1.691 秒	**0.250 秒**	1.186 秒	11.140 秒	———

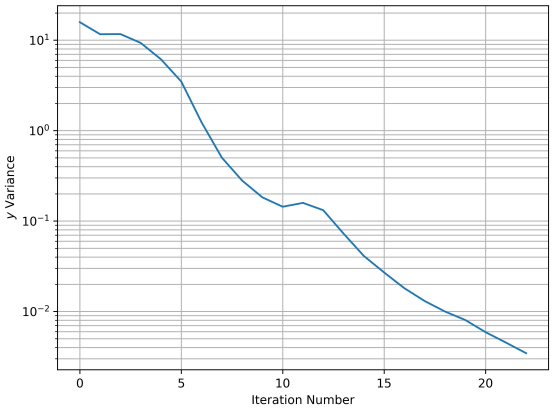


图 8. $\{y^i\}$ 在场景 2 中随迭代次数变化的方差。

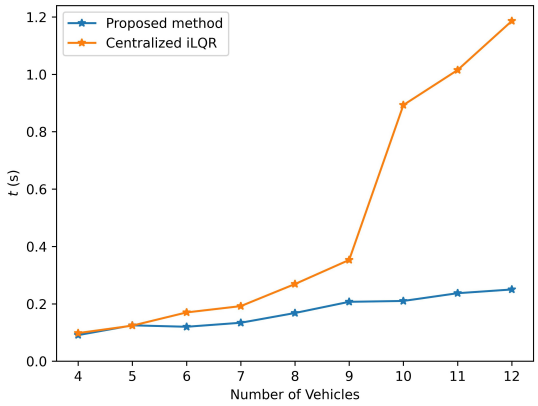


图 9. 集中式 iLQR 求解器和所提方法在不同车辆数量下的计算时间。

后完成。轨迹的迭代过程如图 7 所示。图 8 以对数尺度绘制了 $\{y^i\}$ 相对于迭代次数的方差。同样，$\{y^i\}$的方差近乎单调递减，表明算法收敛。

表 I 的第二行显示了场景 2 中所有实现的计算时间。同样，所提方法的多进程版本速度最快，耗时 $0.250s$ ，约为集中式 iLQR 求解器的 $4.7\times$ 倍、我们方法单进程实现的 $6.8\times$ 倍，以及 IPOPT 求解器的 $44.6\times$ 倍。对于此场景，SQP 求解器未能收敛。与包含 3 辆车辆的场景 1 相比，所提方法实现了高得多的加速倍数，这支持了以下观点：与基线方法相比，该方法能够更好地适应参与车辆数量的增加。

在解的质量方面，表 III 的第二行表明，我们的方法收敛到的总体代价低于集中式 iLQR 求解器，且与 IPOPT 求解器得到的结果相当，不过总体代价略高。我们认为，为了显著加速而在解的质量上做出如此小的妥协是可以接受的。

尽管对于规模更大的场景，降低 σ 和 ρ 的值会更好，但沿用与场景 1 相同的 σ 和 ρ 仍可获得令人满意的结果，次优性仅略有增加（见表 II）。总迭代次数从 24 增至 28，导致计算时间增加约 20%。

D. 讨论

1) 可扩展性：为进一步展示所提算法的可扩展性，我们在场景 2 中进行仿真，车辆数量从 $N = 4$ 变化到 $N = 12$

表 II 使用参数 $\sigma = 0.01$， $\rho = 0.001$ 和 $\sigma = 0.1$， $\rho = 0.01$求解场景 2 的性能比较

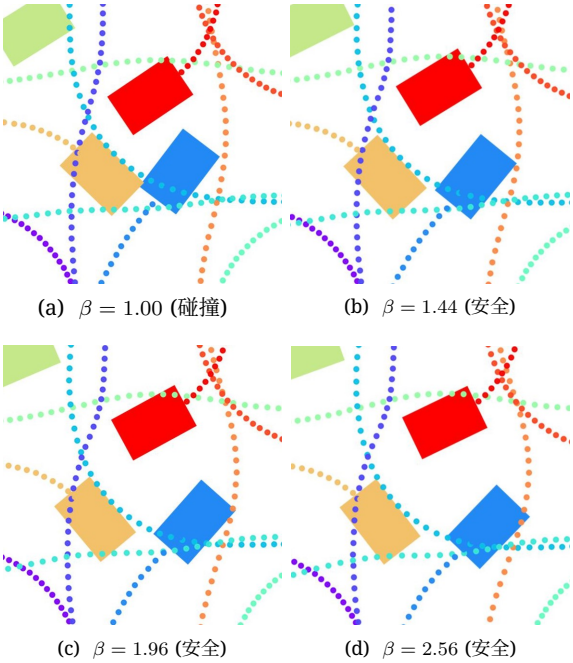
	迭代次数	Cost	Time
$\sigma = 0.01, \rho = 0.001$	24	2303.7	0.250 秒
$\sigma = 0.1, \rho = 0.01$	28	2306.9	0.303 秒

同时保持所有其他参数不变。对于每种情况，我们分别采用集中式 iLQR 求解器和采用多进程实现的所提方法，并测量各自的计算时间。结果如图 9 所示。显然，当车辆数量设为 4 时，我们的方法所需时间与集中式 iLQR 求解器几乎相同。然而，随着车辆数量增加，集中式 iLQR 方法的计算时间急剧增加，远高于我们的方法。具体而言，当网联自动驾驶汽车的数量从 $N = 4$ 增加至 $N = 12$，增至原来的 3 倍时，集中式 iLQR 的计算时间增加了 $12.1\times$，可见其可扩展性较差。相比之下，我们的方法的计算时间仅增加了 $2.74\times$，其增幅低于车辆数量的增幅。

2）安全性：在所提方法中，当两辆车辆之间的距离小于给定的安全距离时，通过在总体代价中添加软惩罚项来实现碰撞规避。通常，这种形式无法保证产生无碰撞解。不过，通过设置足够大的 β 来提高碰撞规避惩罚项的权重，仍可满足安全条件。理论上，当β 趋于无穷大时，只要两辆车辆进入安全距离范围内，就会产生无穷大的代价，因此软惩罚项实际上会变成硬约束；但 β 过大可能导致收敛缓慢。

表 III 场景 1 和 2 中所提方法、集中式 iLQR、IPOPT 与序列二次规划的最终成本比较

	所提方法		集中式 iLQR	IPOPT	SQP
	单进程	多进程			
场景 1	356.02	356.02	356.04	**347.56**	626.40
场景 2	2303.7	2303.7	2394.4	**2297.7**	——



(a) $\beta = 1.00$ (碰撞) (b) $\beta = 1.44$ (安全)

(c) $\beta = 1.96$ (安全) (d) $\beta = 2.56$ (安全)

图10. 在 $\tau = 72$ 时，场景 2 下采用不同 β 值的仿真结果。

为说明这一点，我们在求解场景2时采用不同的 β 值，同时保持所有其他参数不变，并考察 β 对车辆间最小距离和总体计算时间的影响。图10展示了 $\tau = 72$ 时的定性结果，此时两辆车辆的角点之间的距离最近。显然，当 $\beta = 1.00$时，黄色汽车与蓝色汽车会发生碰撞。将 β 设置为大于1.44后，即可避免碰撞；随着 β增大，两辆车辆之间的间距也持续增大。定量结果见表IV。当 β 从1.00增大到2.56时，车辆间的最小距离也随之增大，因此安全性得到提高。然而，较大的 β 需要更多迭代才能使算法收敛，从而导致计算时间更长。

基于前述分析，为确保实际情况下的安全性并保持较高的计算效率，我们可以先用合理的 β 初始化算法，并求解一组候选轨迹。如果检测到候选轨迹发生碰撞，则增大 β，并重新求解轨迹。我们可以继续循环增大 β，直至获得无碰撞轨迹。

表IV 场景 2 中采用不同 β 时车辆中心之间的最小距离、迭代次数和计算时间。

β	最小距离。	迭代次数	Time
1.00	2.59 米	19	0.206 秒
1.21	2.81 米	21	0.228 秒
1.44	2.98 米	24	0.250 秒
1.69	3.15 米	26	0.282 秒
1.96	3.30 米	29	0.314 秒
2.25	3.37 米	32	0.344 秒
2.56	3.43 米	35	0.374 秒

五、结论

本研究探讨多个网联自动驾驶汽车的协同轨迹规划问题。该问题被表述为一个强非凸优化问题，其中包含非线性车辆动力学及其他相关约束。我们提出一种基于双重共识交替方向乘子法算法的分布式优化框架，以便在所有网联自动驾驶汽车之间均衡分配计算负载，使每辆车都能迭代求解一个规模固定的线性二次型调节问题。我们提供了完全并行的实现，以提升效率并实现实时性能。我们在两个交通场景中，分别采用所提方法、集中式 iLQR 求解器、IPOPT 和序列二次规划进行仿真，以验证所提方法的有效性和计算效率。同时，通过对网联自动驾驶汽车数量逐渐增加的情况进行仿真，证明与集中式 iLQR 求解器相比，所提方法具有更优的可扩展性。未来的一个可能方向是将所提算法部署到图形处理器等高性能并行处理器上，以便将算法扩展至包含数百辆车的超大规模场景。另一项未来工作是开展实地实验，进一步验证所提方法的有效性。

参考文献

[1] X. Sun、F. R. Yu 和 P. Zhang，《网联与自动驾驶车辆（CAVs）网络安全综述》，*IEEE*智能交通系统汇刊，第23卷第7期，第6240–6259页，2022年。[2] A. Chehri、N. Quadar 和 R. Saadane，《智慧城市智能交通系统中车载自组织网络的通信与定位技术：综述》，《智能交通系统*2020*》，第167–177页，2020年。[3] B. Xu、X. J. Ban、Y. Bian、W. Li、J. Wang、S. E. Li 和 K. Li，《孤立交叉口网联车辆交通信号优化与速度控制的协作方法》，*IEEE*智能交通系统汇刊，第20卷第4期，第1390–1403页，2018年。[4] P. Hang、C. Lv、C. Huang、J. Cai、Z. Hu 和 Y. Xing，《考虑社会行为的自动驾驶汽车决策与运动规划一体化框架》，*IEEE*车辆技术汇刊，第69卷第12期，第14 458–14 469页，2020年。[5] P. Hang、C. Lv、C. Huang、Y. Xing 和 Z. Hu，《多车道合流区网联自动驾驶汽车的协作决策：一种联盟博弈方法》，*IEEE*智能交通系统汇刊，第23卷第4期，第3829–3841页，2021年。

[6] Y. Guan、Y. Ren、S. E. Li、Q. Sun、L. Luo 和 K. Li，《通过近端策略优化实现交叉路口网联与自动驾驶车辆的集中式协作》，IEEE车辆技术汇刊，第69卷第11期，第12 597–12 608页，2020年。[7] T. Chu、J. Wang、L. Codec`a 和 Z. Li，《大规模交通信号控制的多智能体深度强化学习》，IEEE智能交通系统汇刊，第21卷第3期，第1086–1095页，2019年。[8] G. Li、Y. Yang、S. Li、X. Qu、N. Lyu 和 S. E. Li，《变道场景下自动驾驶汽车的决策：具有风险感知的深度强化学习方法》，《交通研究C辑：新兴技术》，第134卷，文章编号103452，2022年。[9] J. Wu、F. Perronnet 和 A. Abbas-Turki，《协作式车辆-执行器系统：协作式交叉口管理的基于序列的框架》，IET智能交通系统，第8卷，360，2014 4，352第。页–。[10] G. R. de Campos、P. Falcone 和 J. Sj¨oberg，《自主协作驾驶：一种用于交叉路口通行的基于速度的协商方法》，载于第16届IEEE智能交通系统国际会议（ITSC 2013）。IEEE，2013年，第1456–1461页。[11] X. Pan、B. Chen、S. Timotheou 和 S. A. Evangelou，《自动驾驶汽车交叉路口通行的凸最优控制框架》，arXiv预印本arXiv:2203.16870，2022年。[12] G. R. Campos、P. Falcone、H. Wymeersch、R. Hult 和 J. Sj¨oberg，《交通交叉口的协作式滚动时域冲突消解》，载于第53届IEEE决策与控制会议。IEEE，2014年，第2932–2937页。[13] F. Tedesco、D. M. Raimondo、A. Casavola 和 J. Lygeros，《交互车辆的分布式碰撞避免：一种指令调节器方法》，IFAC会议论文集，第43卷第19期，第293–298页，2010年。[14] K.-D. Kim 和 P. R. Kumar，《一种基于模型预测控制的方法，用于实现自主地面交通可证明的系统级安全性与活性》，IEEE自动控制汇刊，第59卷，3356，2014 12，3341第。页–。[15] Y. Pan、Q. Lin、H. Shah 和 J. M. Dolan，《通过自适应约束迭代线性二次调节器实现自动驾驶安全规划》，载于2020年IEEE/RSJ智能机器人与系统国际会议（IROS）。IEEE，2020年，第2377–2383页。[16] N. J. Kong、C. Li 和 A. M. Johnson，《腿式机器人接触隐式稳定的混合迭代线性二次调节器模型预测控制》，arXiv预印本arXiv:2207.04591，2022年。[17] J. Ma、Z. Cheng、X. Zhang、Z. Lin、F. L. Lewis 和 T. H. Lee，《局部学习驱动的受限轨迹规划迭代线性二次调节器》，IEEE神经网络与学习系统汇刊，2022年。[18] Y. Tassa、N. Mansard 和 E. Todorov，《控制受限微分动态规划》，载于 2014 年 IEEE 机器人与自动化国际会议（ICRA）。IEEE，2014年，第1168–1175页。[19] J. Chen、W. Zhan 和 M. Tomizuka，《采用约束迭代线性二次调节器的自动驾驶运动规划》，IEEE智能车辆汇刊，第4卷第2期，第244–254页，2019年。[20] T. Kavuncu、A. Yaraneri 和 N. Mehr，《势函数迭代线性二次调节器：一种用于规划多智能体交互轨迹的势函数最小化控制器》，arXiv预印本arXiv:2107.04926，2021年。[21] S. Boyd、N. Parikh、E. Chu、B. Peleato、J. Eckstein等，《通过交替方向乘子法实现分布式优化与统计学习》，《机器学习基础与趋势》，第3卷第1期，第1–122页，2011年。[22] G. Raja、S. Anbalagan、G. Vijayaraghavan、S. Theerthagiri、S. V. Surya-narayan 和 X.-W. Wu，《SP-CIDS：面向车载自组织网络的安全隐私协作式入侵检测系统》，IEEE智能交通系统汇刊，第22卷第7期，第4385–4393页，2020年。[23] J. Ma、Z. Cheng、X. Zhang、M. Tomizuka 和 T. H. Lee，《关于采用凸参数化的 $\mathcal{H}_\infty$ 保证成本控制对称高斯–赛德尔交替方向乘子法算法》，IEEE系统、人类与控制论汇刊：系统，2022年。[24] Z. Cheng、J. Ma、X. Zhang 和 T. H. Lee，《用于模型预测控制问题的半近端交替方向乘子法及其在无人机系统中的应用》，载于2020年第20届控制、自动化与系统国际会议（ICCAS）。IEEE，2020年，第82–87页。[25] C. Zhang、M. Ahmad 和 Y. Wang，《基于交替方向乘子法的隐私保护分散式优化》，IEEE信息取证与安全汇刊，第14卷第3期，第565–580页，2018年。[26] J. Ma、Z. Cheng、X. Zhang、M. Tomizuka 和 T. H. Lee，《自动驾驶中约束迭代线性二次调节器的交替方向乘子法》，IEEE智能交通系统汇刊，第23卷第12期，第23 031–23 042页，2022年。[27] G. Mateos、J. A. Bazerque 和 G. B. Giannakis，“分布式稀疏线性回归”，*IEEE* 信号处理汇刊，第 58 卷，第 10 期，第 5262–5276 页，2010 年。[28] M. Doostmohammadian、A. Aghasi、A. I. Rikos、A. Grammenos、E. Ka- lyvianaki、C. N. Hadjicostis、K. H. Johansson 和 T. Charalambous，“具有异构时变延迟的分布式任意时刻可行资源分配”，*IEEE* 开放控制系统期刊，第 1 卷，第 255–267 页，2022 年。[29] A. Aboudonia、G. Banjac、A. Eichler 和 J. Lygeros，“面向参考跟踪的分布式模型预测控制中终端要素的在线计算”，载于*2022*年欧洲控制会议（*ECC*）。IEEE，2022 年，第 847–852 页。[30] G. Banjac、F. Rey、P. Goulart 和 J. Lygeros，“通过对偶共识交替方向乘子法进行去中心化资源分配”，载于*2019*年美国控制会议（*ACC*）。IEEE，2019 年，第 2789–2794 页。[31] P. D. Grontas、M. W. Fisher 和 F. D¨orfler，“通过系统级综合和对偶共识交替方向乘子法进行分布式约束控制设计”，*arXiv* 预印本 *arXiv:2207.06947*，2022 年。[32] Z. Cheng、J. Ma、X. Zhang、C. W. de Silva 和 T. H. Lee，“基于交替方向乘子法的多智能体无碰撞模型预测控制并行优化”，*arXiv* 预印本 *arXiv:2101.09894*，2021 年。[33] X. Zhang、Z. Cheng、J. Ma、S. Huang、F. L. Lewis 和 T. H. Lee，“基于半定松弛的交替方向乘子法：网联自动驾驶汽车的协同规划与控制”，*IEEE*智能交通系统汇刊，2021 年。[34] X. Zhang、Z. Cheng、J. Ma、L. Zhao、C. Xiang 和 T. H. Lee，“利用交替方向乘子法进行并行协同运动规划”，载于*IECON 2021–IEEE* 工业电子学会第*47*届年会。IEEE，2021 年，第 1–6 页。[35] A. D. Saravanos、Y. Aoyama、H. Zhu 和 E. A. Theodorou，“面向大规模多智能体控制的分布式微分动态规划架构”，*arXiv* 预印本 *arXiv:2207.13255*，2022 年。[36] J. A. Andersson、J. Gillis、G. Horn、J. B. Rawlings 和 M. Diehl，“CasADi：用于非线性优化和最优控制的软件框架”，数学规划计算，第 11 卷，第 1 期，第 1–36 页，2019 年。