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

Limiting Computation Levels in Prioritized Trajectory Planning with Safety Guarantees

Patrick Scheffe, Jianye Xu, and Bassam Alrifaee

Abstract— **In prioritized planning for vehicles, vehicles plan trajectories in parallel or in sequence. In parallel prioritized planning, the computation time remains approximately constant with an increasing number of vehicles, but it is difficult to guarantee collision-free trajectories. Although sequential prioritized planning can guarantee collision-free trajectories, the computation time increases with the number of sequentially computing vehicles, which we call computation levels. This number is determined by the directed coupling graph which results from the coupling and prioritization of vehicles. This work's contribution is twofold. First, we guarantee safe trajectories in parallel planning through reachability analysis. Although these trajectories are collision-free, they tend to be conservative. Second, we address this conservativeness by planning with a subset of vehicles in sequence. We formulate the problem of selecting this subset as a graph partitioning problem, in which we limit the size of the resulting subgraphs. Consequently, we can choose the number of computation levels independently from the directed coupling graph, and thus are able to limit the computation time in prioritized planning. In our simulations, we reduce the number of computation levels to approximately 64% compared to sequential prioritized planning while maintaining the solution quality.**

Video youtu.be/di6X6XTGt88
Code github.com/embedded-software-laboratory/p-dmpc

I. Introduction

A. Motivation

Networked control systems (NCSs) are spatially distributed systems within which controllers communicate with each other [1]. When each controller uses model predictive control (MPC), in which an optimal control problem (OCP) is solved, we speak of networked MPC. Networked MPC strategies include centralized model predictive control (CMPC) and distributed model predictive control (DMPC) [2].

CMPC is characterized by its global system knowledge, thus being able to find the optimum to its OCP, if it exists [3]. However, its time complexity may be exponential in the number of agents [4]. Unlike CMPC, DMPC is characterized by a local controller for each agent, which both solves an OCP with only the decision variables of the agent and communicates with other local controllers. Since the OCPs in DMPC are smaller than the centralized OCP, the computation time of the NCS is shorter. However, due to the local system knowledge, the control quality is typically worse than in CMPC.

This work focuses on trajectory planning with prioritized distributed model predictive control (P-DMPC), in which lower-priority vehicles avoid collisions with neighboring, higher-priority vehicles. In P-DMPC, vehicles plan sequentially or in parallel. When planning sequentially, the number of computation levels, i.e., the number of sequentially computing vehicles, increases approximately linearly in the number of vehicles. In large-scale NCS, this can result in long computation times. When planning in parallel, the computation time remains approximately constant with an increasing number of vehicles, as the number of computation levels always equals one. However, this typically comes at the cost of worse solution quality or even unsafe solutions. The goal of this work is to guarantee a limit of the number of computation levels during trajectory planning with sequential P-DMPC, while also guaranteeing the safety of the planned trajectories. This achieves a scalability of the approach in the number of vehicles.

B. Related Work

1) Sequential Planning: In sequential planning, a higher-priority vehicle plans before its neighboring, lower-priority vehicles and then communicates its prediction to those vehicles. Using this prediction, the lower-priority vehicles can guarantee collision avoidance.

To reduce the computation time of sequential planning, extensive research has been conducted in the literature. One strategy is to reduce the number of computation levels. In our previous work [5], we proposed an algorithm that finds parallelizable computations. However, the number of parallelizable computations depends on the coupling and prioritization of vehicles. In [6], the number of parallelizable computations is maximized by determining the prioritization through a centralized graph coloring problem. In our previous work [7], we proposed a decentralized coloring algorithm which maximizes the number of parallelizable computations and additionally reduces the number of constraints in the OCP, thus further reducing computation time. In [8], the authors split agents into fixed-sized groups, each comprising three agents. These groups are prioritized and formulate their own centralized OCPs, which are solved sequentially. This effectively cuts the number of computation levels in third. However, the computations are more involving. In

This research was funded by (1) the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within the Priority Program SPP 1835 "Cooperative Interacting Automobiles" (grant number: KO 1430/17-1) and (2) the Bundesministerium für Digitales und Verkehr (German Federal Ministry for Digital and Transport) within the project "Harmonizing Mobility" (grant number 19FS2035A).

P. Scheffe and J. Xu are with the Chair of Embedded Software, RWTH Aachen University, Germany, {scheffe, xu}@embedded.rwth-aachen.de

B. Alrifaee is with the Department of Aerospace Engineering, University of the Bundeswehr Munich, Germany, bassam.alrifaee@unibw.de

[9], all vehicles compute in parallel. Lower-priority vehicles recompute in the event of a conflict, i.e., when their trajectory collides with a trajectory of a higher-priority vehicle. At best, all vehicles can compute in parallel; yet at worst, they compute sequentially. In the majority of the above works, the coupling graph and prioritization determine a computation order which ensures all constraints are considered.

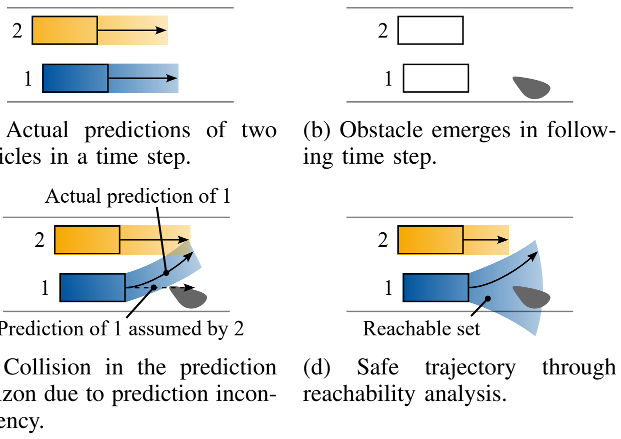
The computation time of sequential planning can also be decreased by decreasing the computation time for each individual agent. This is often achieved through the use of linearized dynamics [10]–[13] or sophisticated numerical solvers [14]–[16].

We conclude that in state-of-the-art sequential planning approaches, the number of computation levels increases approximately linearly with the number of vehicles. There exists no work that guarantees a limited number of computation levels in sequential planning.

2) Parallel Planning: In parallel planning, all vehicles plan at the same time. Consequently, a lower-priority vehicle cannot wait for a communicated trajectory prediction of a higher-priority vehicle, but still must avoid a collision. Therefore, it needs to predict the trajectory of a higher-priority vehicle.

Predicting other vehicles' trajectories is one of the major challenges in autonomous driving [17]. If vehicles communicate, a previously communicated trajectory prediction can be shifted to the current time and further predicted to the required duration [6], [18]. If vehicles do not communicate, the trajectory can be predicted with physics-based [19], [20] or learning-based methods [21], [22]. However, it is impossible to guarantee that a trajectory prediction is correct. If the prediction is incorrect, i.e., it does not coincide with the actual prediction, the NCS is not prediction consistent. The loss of prediction consistency can lead to undesired behavior. This is illustrated for two vehicles in Fig. 1, in which vehicle 2 has lower priority than vehicle 1. Figure 1a shows the actual predictions. In the following time step, vehicle 2 needs to estimate the trajectory of vehicle 1, since they plan in parallel. A reasonable assumption is that vehicle 1 stays on its predicted trajectory [6], [18]. However, due to an obstacle emerging (Fig. 1b), the prediction of vehicle 1 changes (Fig. 1c), which consequently leads to a collision in the prediction horizon.

There are mainly two strategies in the literature dealing with the prediction inconsistency problem in parallel planning. One strategy is to prevent it from occurrence. This can be achieved by consensus-based approaches, which synchronize the predictions among vehicles [23], [24]. However, consensus-based approaches rely on iterations of synchronization among vehicles. The number of iterations required to reach a consensus depends on many factors and may be unbounded. Because of the resulting unbounded computation time, these approaches may be impractical for real-time applications. Although centralized controllers are always prediction consistent [25]–[28], their computation time may increase exponentially with the number of vehicles, making them unsuitable for NCSs with many vehicles.



(a) Actual predictions of two vehicles in a time step.
(b) Obstacle emerges in following time step.
(c) Collision in the prediction horizon due to prediction inconsistency.
(d) Safe trajectory through reachability analysis.

Fig. 1: Example of prioritized trajectory planning with prediction inconsistency in a dynamic environment. Vehicle 2 has lower priority. Arrows indicate predictions, gradient fills indicate predicted occupied areas.

Another strategy is to find solutions that are feasible even with prediction inconsistency. In [29]–[32], the change of system states or control inputs between consecutive time steps is constrained. Lower-priority vehicles can incorporate the bounded uncertainty when considering the prediction of the previous time step of higher-priority vehicles. Although lower-priority vehicles benefit, the maneuverability of higher-priority vehicles is impaired and thus their responsiveness to dynamic environments is reduced. In general, reducing the maneuverability will decrease the solution quality. In dynamic environments, it might even lead to unsafe behavior.

C. Contribution of this Paper

This paper's main contribution is a method to limit the number of sequentially computing vehicles in prioritized planning. We first propose a method for planning guaranteed safe trajectories through reachability analysis despite inconsistent predictions among vehicles. As Fig. 1d illustrates, avoiding the reachable set of a vehicle guarantees safe trajectories. Consequently, all vehicles can compute solutions to their OCP in parallel. Parallel computation reduces the computation time, but also reduces the solution quality. To address this shortcoming, we propose an approach that involves sequentializing a subset of the computations. This way, we are able to limit the computation time of the NCS while enhancing the solution quality.

D. Notation

In this paper, we speak of agents whenever concepts are generally applicable to P-DMPC. A variable x is marked with a superscript $x^{(i)}$ if it belongs to agent i. The actual value of a variable x at time k is written as x_k, while values predicted for time $k+l$ at time k are written as $x_{k+l|k}$. A trajectory is denoted by replacing the time argument with $(\cdot)$ as in $x_{\cdot|k}$. For any set $\mathcal{S}$, the cardinality of the set is denoted by $|\mathcal{S}|$.

E. Further Structure of this Paper

In Section II, we describe how we guarantee safe trajectories despite inconsistent predictions through reachability analysis. In Section III, we present an algorithm that improves solution quality by sequential computation of a subset of vehicles through graph partitioning. In Section IV, we evaluate the proposed framework in simulation.

II. Safe Plans Despite Prediction Inconsistency

This paper presents a distributed framework for prioritized planning as shown in Fig. 2 for a vehicle $i \in \{1, \ldots, N_A\}$, where N_A denotes the number of vehicles. The communication with other vehicles is indicated by arrows to the communication network.

This section elaborates on how we generate guaranteed safe trajectories through reachability analysis in parallel P-DMPC. We delineate how to compute reachable sets in Section II-A, how to couple vehicles in Section II-B, and how to plan trajectories with MPC in Section II-C. Vehicles can determine priorities with any prioritization algorithm, as long as each vehicle obtains a unique priority. The corresponding block is therefore not the subject of the present work but can be designed according to [7], [33]. Furthermore, Section III details how to group vehicles.

A. Computing Reachable Sets

We compute the reachable sets of the states of vehicles to determine the possibly occupied areas over a time horizon $N_p \in \mathbb{N}$. These reachable sets provide a basis for determining the coupling between vehicles (Section II-B) and planning safe trajectories (Section II-C).

To define a vehicle's reachable set, we introduce the number of system states $n \in \mathbb{N}$, the number of control inputs $m \in \mathbb{N}$, the set of admissible system states $\mathcal{X} \subseteq \mathbb{R}^n$, the set of admissible control inputs $\mathcal{U} \subseteq \mathbb{R}^m$, the system states $\boldsymbol{x}(t) \in \mathcal{X}$, the control inputs $\boldsymbol{u}(t) \in \mathcal{U}$, and the system model $\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t))$. For brevity, we omit the time argument if it is clear from the context.

Definition 1 (Reachable set of a time interval, adapted from Def. 4 in [34])**.** The reachable set $\mathcal{R}^{(i)}_{[t_0,t_1]|t_0}$ of the states of vehicle i from time t_0 to time t_1 is

$$\mathcal{R}^{(i)}_{[t_0,t_1]|t_0} = \left\{ \bigcup_{t' \in [t_0,t_1]} \int_{t_0}^{t'} f(\boldsymbol{x}, \boldsymbol{u}) dt \;\middle|\; \boldsymbol{x}(t_0) \in \mathcal{X}(t_0), \boldsymbol{u} \in \mathcal{U} \right\}. \quad (1)$$

In discrete time, we denote a reachable set of a time step $k+h$ with a duration of a sample time T_s as $\mathcal{R}^{(i)}_{[k+h,k+h+1]|k}$, termed a one-step reachable set. The time is given as $t = k \cdot T_s$. In the remainder of this work, $\mathcal{R}$ denotes the occupancy of a vehicle in its reachable set of states, referring to the area in x- and y-coordinates that is occupied by the vehicle.

Given the complexity of computing reachable sets, we compute one-step reachable sets offline using a motion primitive automaton (MPA) from our previous work [35]. The MPA forms the system model with a set of states and a set of motion primitives. For each state of the MPA, we compute and unite the occupancies of all available motion primitives at each time step within the horizon N_p, yielding N_p one-step reachable sets $\mathcal{R}^{(i)}_{[0,1]|\text{off.}}, \ldots, \mathcal{R}^{(i)}_{[N_p-1,N_p]|\text{off.}}$. For offline computation, we assume the vehicle is at the origin with a yaw angle of zero. For online trajectory planning, we shift the corresponding precomputed reachable sets to the vehicle's current position and rotate them counterclockwise by the vehicle's current yaw angle. Figure 3 exemplifies this concept for a horizon of $N_p = 3$.

B. Coupling Vehicles

We use the concept of couplings to determine which vehicles should interact with each other and represent them with a coupling graph.

Definition 2 (Coupling graph)**.** A coupling graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a pair of two sets, the set of vertices $\mathcal{V} = \{1, \ldots, N_A\}$ which represents agents and the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ that represents the interaction between them.

Although coupling all vehicles would guarantee that coupling constraints for all vehicles will be considered, it would result in N_A computation levels and thus a long computation time. Therefore, we only couple vehicles that can potentially collide within the horizon N_p. This results in a time-variant coupling graph $\mathcal{G}(k) = (\mathcal{V}, \mathcal{E}(k))$. Formally, we couple two vehicles at a time step k if at least one of their one-step reachable sets intersect within the horizon N_p, resulting in the set of edges

$$\mathcal{E}(k) = \left\{ (i \to j) \in \mathcal{V} \times \mathcal{V} \;\middle|\; \exists\, h \in \{0, \ldots, N_p - 1\} : \mathcal{R}^{(i)}_{[k+h,k+h+1]|k} \cap \mathcal{R}^{(j)}_{[k+h,k+h+1]|k} \neq \emptyset \right\}, \quad (2)$$

with $(i \to j)$ denoting the edge from vehicle i to vehicle j.

In the case of P-DMPC, the coupling graph is directed, and prioritization determines the directions of edges. We denote by $\vec{\mathcal{G}} = (\mathcal{V}, \vec{\mathcal{E}})$ a directed coupling graph, where $\vec{\mathcal{E}} \in \mathcal{E}$ is the set of directed edges. Each directed edge denotes a coupling objective or constraint in the OCP associated with and only with the ending vertex. If an edge between two vertices exists, it points from the higher-priority vertex to the lower-priority vertex. The uniqueness of priorities ensures that the orientation of each edge is unambiguous. We denote the set of all parallelly planning, higher-priority neighbors of vehicle i with

$$\mathcal{V}^{(i\leftarrow)}_{\text{par.}}(k) = \left\{ j \in \mathcal{V} \;\middle|\; \exists\, (j \to i) \in \vec{\mathcal{E}}_{\text{par.}}(k) \right\}, \quad (3)$$

with $\vec{\mathcal{E}}_{\text{par.}} \subseteq \vec{\mathcal{E}}$ denoting the set of edges between parallelly computing neighbors. Note that in this section $\vec{\mathcal{E}}_{\text{par.}} = \vec{\mathcal{E}}$, since we let all vehicles compute parallelly.

C. Planning Trajectories

Our objective is to enable parallel trajectory planning with guaranteed collision avoidance. We achieve this by having lower-priority vehicles avoid the reachable set of higher-priority vehicles, thus eliminating the possibility of

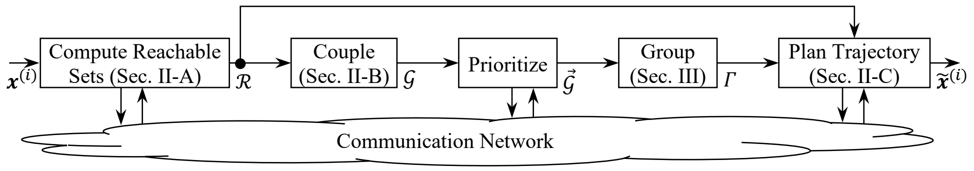
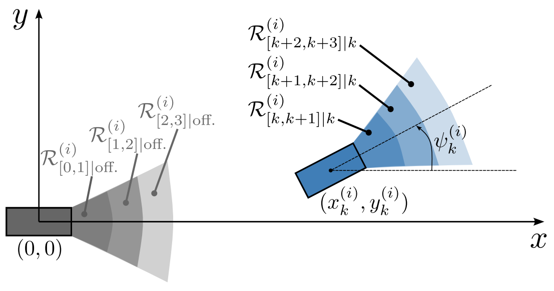


Fig. 2: Distributed planning framework overview, illustrated for vehicle i. $\boldsymbol{x}^{(i)}$: measured states; $\mathcal{R}$: reachable sets; $\mathcal{G}$: undirected coupling graph; $\vec{\mathcal{G}}$: directed coupling graph; $\varGamma$: graph partition; $\tilde{\boldsymbol{x}}^{(i)}$: predicted trajectory. Time argument omitted.



y
x
$\mathcal{R}^{(i)}_{[k+2,k+3]|k}$
$\mathcal{R}^{(i)}_{[k+1,k+2]|k}$
$\mathcal{R}^{(i)}_{[k,k+1]|k}$
$\mathcal{R}^{(i)}_{[2,3]|\text{off.}}$
$\mathcal{R}^{(i)}_{[1,2]|\text{off.}}$
$\mathcal{R}^{(i)}_{[0,1]|\text{off.}}$
$\psi^{(i)}_k$
$(x^{(i)}_k, y^{(i)}_k)$
$(0,0)$

Fig. 3: Example of transforming precomputed offline one-step reachable sets (depicted in grey). Actual reachable sets in blue. $x^{(i)}_k, y^{(i)}_k$ and $\psi^{(i)}_k$: current x-, y-coordinate, and yaw angle.

a collision. Figure 1d illustrates our approach, which is inspired from [36]. While [36] simplifies the prediction of other vehicles' reachable sets using a point mass model for computational efficiency, our method achieves computational efficiency by computing reachable sets offline. We use the nonlinear kinematic single-track model [37, section 2.2] in this work, which assumes no slip on the front and rear wheels, and no forces acting on the vehicle. The model equations are

$$f(\boldsymbol{x},\boldsymbol{u}) = \begin{cases} \dot{x}(t) = \mathrm{v}(t)\cdot\cos(\psi(t)+\beta(t)), \\ \dot{y}(t) = \mathrm{v}(t)\cdot\sin(\psi(t)+\beta(t)), \\ \dot{\psi}(t) = \mathrm{v}(t)\cdot\dfrac{1}{L}\cdot\tan(\delta(t))\cos(\beta(t)), \\ \dot{\mathrm{v}}(t) = u_{\mathrm{v}}(t), \\ \dot{\delta}(t) = u_{\delta}(t), \end{cases} \tag{4}$$

with

$$\beta(t) = \tan^{-1}\left(\frac{\ell_r}{L}\tan(\delta(t))\right), \tag{5}$$

where the vector field $f\colon \mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^n$ is the continuous-time nonlinear system model, $x\in\mathbb{R}$ and $y\in\mathbb{R}$ describe the position of the center of gravity (CG), $\psi\in[0,2\pi)$ is the orientation, $\beta\in[-\pi,\pi)$ is the side slip angle, $\delta\in[-\pi,\pi)$ and $u_\delta\in\mathbb{R}$ are the steering angle and its derivative respectively, $\mathrm{v}\in\mathbb{R}$ and $u_{\mathrm{v}}\in\mathbb{R}$ are the speed and acceleration of the CG respectively, L is the wheelbase length and ℓ_r is the length from the rear axle to the CG.

Formally, we guarantee collision avoidance with the following OCP, which is solved inside the P-DMPC of the planner of each vehicle $i\in\{1,\dots,N_A\}$ at each time step k.

$$\underset{\boldsymbol{u}^{(i)}_{\cdot|k}}{\text{minimize}} \quad \sum_{h=1}^{N_p} l^{(i)}_x\left(\boldsymbol{x}^{(i)}_{k+h|k}, \boldsymbol{r}^{(i)}_{k+h|k}\right) \tag{6a}$$

subject to

$$\boldsymbol{x}^{(i)}_{k+h+1|k} = f^{(i)}_d\left(\boldsymbol{x}^{(i)}_{k+h|k}, \boldsymbol{u}^{(i)}_{k+h|k}\right), \quad h=0,\dots,N_p-1, \tag{6b}$$

$$\boldsymbol{x}^{(i)}_{k+h|k}\in\mathcal{X}^{(i)}, \quad h=1,\dots,N_p-1, \tag{6c}$$

$$\boldsymbol{x}^{(i)}_{k+N_p}\in\mathcal{X}^{(i)}_{N_p}, \tag{6d}$$

$$\boldsymbol{u}^{(i)}_{k+h|k}\in\mathcal{U}^{(i)}, \quad h=0,\dots,N_p-1, \tag{6e}$$

$$\mathcal{O}\left(\tilde{\boldsymbol{x}}^{(i)}_{[k+h,k+h+1]|k}\right)\cap\mathcal{R}^{(j)}_{[k+h,k+h+1]|k}=\emptyset, \quad \forall\, j\in\mathcal{V}^{(i\leftarrow)}_{\text{par.}}(k), h=0,\dots,N_p-1. \tag{6f}$$

The function $l^{(i)}_x\colon\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}$ penalizes a deviation to the reference trajectory $\boldsymbol{r}^{(i)}_{\cdot|k}$ of vehicle i and composes the objective function (6a). The vector field $f^{(i)}_d\colon\mathbb{R}^n\times\mathbb{R}^m\to\mathbb{R}^n$ in (6b) resembles the discrete-time nonlinear system model obtained from (4). $\mathcal{O}(\tilde{\boldsymbol{x}}^{(i)}_{[k_1,k_2]})$ denotes vehicle i's occupancy between the time steps k_1 and k_2. We guarantee safe trajectories despite prediction inconsistency by constraining the occupancy $\mathcal{O}\left(\tilde{\boldsymbol{x}}^{(i)}_{[k+h,k+h+1]|k}\right)$ of each one-step trajectory $\tilde{\boldsymbol{x}}^{(i)}_{[k+h,k+h+1]|k}$ of vehicle i within the prediction horizon to the area outside the corresponding one-step reachable set $\mathcal{R}^{(j)}_{[k+h,k+h+1]|k}$ of all parallelly planning, higher-priority neighbors $j\in\mathcal{V}^{(i\leftarrow)}_{\text{par.}}(k)$ in (6f). It is computationally hard to find the global optimum to OCP (6) due to its nonlinearity and nonconvexity. We approximate (6) with a receding horizon graph search which can be solved online from our previous work [35].

III. Increasing Solution Quality Through Grouping

In P-DMPC, reachability analysis-based parallel planning guarantees safety but may sacrifice solution quality. Contrarily, sequential planning achieves consistent predictions and leads to less conservative trajectories because each vehicle i must avoid only the predicted occupancy $\mathcal{O}(\tilde{\boldsymbol{x}}^{(j)})\subseteq\mathcal{R}^{(j)}$ of all vehicles $j\in\mathcal{V}^{(i\leftarrow)}_{\text{seq.}}$, where $\mathcal{V}^{(i\leftarrow)}_{\text{seq.}}$ denotes the set of all sequentially planning, higher-priority neighbors of vehicle i:

$$\mathcal{V}^{(i\leftarrow)}_{\text{seq.}}(k) = \left\{ j\in\mathcal{V} \;\middle|\; \exists\,(j\to i)\in\vec{\mathcal{E}}_{\text{seq.}}(k) \right\}. \tag{7}$$

Note that we split the set of couplings in sequential couplings $\vec{\mathcal{E}}_{\text{seq.}}$ and parallel couplings $\vec{\mathcal{E}}_{\text{par.}}$ with

$$\vec{\mathcal{E}} = \vec{\mathcal{E}}_{\text{seq.}} \cup \vec{\mathcal{E}}_{\text{par.}}, \quad \vec{\mathcal{E}}_{\text{seq.}} \cap \vec{\mathcal{E}}_{\text{par.}} = \emptyset, \tag{8}$$

where $\vec{\mathcal{E}}_{\text{seq.}}$ denotes edges that indicate sequential computation of the neighbors.

A sequentially planning, lower-priority vehicle i avoids collisions by avoiding an intersection of the occupancies as

$$\mathcal{O}\left(\tilde{\boldsymbol{x}}^{(i)}_{[k+h,k+h+1]|k}\right) \cap \mathcal{O}\left(\tilde{\boldsymbol{x}}^{(j)}_{[k+h,k+h+1]|k}\right) = \emptyset,$$
$$\forall\, j \in \mathcal{V}^{(i\leftarrow)}_{\text{seq.}}(k), h = 0, \ldots, N_p - 1. \tag{9}$$

Given the existing constraint (6f) in the OCP (6) and by adding (9) as an additional constraint, we guarantee collision-freeness between both parallelly and sequentially planning vehicles.

Prioritized planning is computationally efficient, since the OCPs have few decision variables. We can therefore increase the solution quality of parallel planning by sequentializing a subset of computations. We call the number of sequential computations the number of computation levels N_{CL}. Let $T_{\text{sol.,max}}$ denote the maximum time that any vehicle needs to solve its OCP.

Assumption 1. The computation time T for any vehicle to solve its OCP is bounded by $T_{\text{sol.,max}}$, i.e., $T \leq T_{\text{sol.,max}}$.

Remark 1. Assumption 1 is mild if an anytime trajectory planner such as [38] is employed.

Definition 3 (Anytime trajectory planner)**.** An anytime trajectory planner is a trajectory planner that quickly identifies a feasible trajectory and incrementally improves it over time.

Given a sample time T_s, we introduce the maximum allowed number of computation levels

$$N_{\text{CL,al.}} = \left\lfloor \frac{T_s}{T_{\text{sol.,max}}} \right\rfloor. \tag{10}$$

Remark 2. We guarantee real-time planning if the actual number of computation levels is less than or equal to the maximum allowed number of computation levels.

Assumption 2. The sample time T_s is chosen such that is larger or equal to the maximum computation time $T_{\text{sol.,max}}$, i.e., $T_{\text{sol.,max}} \leq T_s$.

Remark 3. Assumption 2 guarantees the maximum allowed number of computation levels in (10) being at least 1.

Given a coupling graph, a prioritization, and the maximum allowed number of computation levels, we sequentialize computations by weighing edges in the coupling graph and grouping vehicles that should plan sequentially.

A. Weighing Couplings

The effect on the solution quality of a sequential or parallel coupling depends on the traffic situation. Quantifying this effect exactly would require us to solve the networked OCP for all combinations of couplings, i.e., $2^{|\vec{\mathcal{E}}|}$ times.

Since this is computationally intractable, we propose to heuristically weigh the edges in the coupling graph to determine how much the NCS might benefit from a sequential coupling. A high expected benefit results in a high weight. When two vehicles are far from each other, the intersection of their reachable sets is small. Even though the lower-priority vehicle must avoid the other vehicle's reachable set, it maintains a high maneuverability. We express this relation with a heuristic based on the shortest time to a collision (STC) $t_{\text{STC}}\colon \vec{\mathcal{E}} \to \mathbb{R}$. We compute this time based on the states and acceleration capabilities of the vehicles. The concept of STC has been used in the literature, for example, to measure the criticality of traffic situations [39]. We propose an edge-weighing function $w\colon \vec{\mathcal{E}} \to \mathbb{R}$ to weigh a coupling higher if vehicles have a lower STC:

$$w\left((i \to j)\right) = e^{-t_{\text{STC}}((i \to j))}. \tag{11}$$

The resulting graph is called an edge-weighted directed coupling graph

$$\vec{\mathcal{G}}_w = \left(\mathcal{V}, \vec{\mathcal{E}}, w\right). \tag{12}$$

B. Grouping Vehicles

Identifying which computations to sequentialize corresponds to partitioning the coupling graph into subgraphs of sequentially computing vehicles. To achieve the highest benefit, we must partition the graph with a minimal sum of cut weight, i.e., the sum of the weights of the edges that connect subgraphs, and a number of computation levels $N^{(p)}_{\text{CL}}$ of each subgraph p which is less than the allowed number of computation levels $N_{\text{CL,al.}}$. Without constraining each subgraph's depth[1] , this corresponds to the well-known min-cut clustering problem in graph theory, as formulated in Problem 1.

Denoting by $\mathcal{V}_p$ the set of vertices belonging to subgraph $\vec{\mathcal{G}}_p$, $\vec{\mathcal{E}}_{pq}$ the set of edges connecting subgraphs p and q

$$\vec{\mathcal{E}}_{pq} = \left\{(i \to j) \in \vec{\mathcal{E}} \;\middle|\; i \in \mathcal{V}_p, j \in \mathcal{V}_q, p \neq q\right\}, \tag{13}$$

$d\colon \mathcal{G} \to \mathbb{N}$ a function returning the depth[1] of a graph, and N_g the number of subgraphs, we formulate an adapted min-cut clustering problem as follows:

Problem 1 (Min-cut clustering with limited depth)**.** Given an edge-weighted directed coupling graph $\vec{\mathcal{G}}_w$, find a partition $\mathit{\Gamma} = \{\vec{\mathcal{G}}_1, \ldots, \vec{\mathcal{G}}_{N_g}\}$ such that

$$\mathit{\Gamma} = \arg\min_{\mathit{\Gamma}} \sum_{p=1}^{N_g - 1} \sum_{q=p+1}^{N_g} \sum_{(i \to j) \in \mathcal{E}_{pq}} w((i \to j)) \tag{14a}$$

subject to

$$\bigcup_{p=1}^{N_g} \mathcal{V}_p = \mathcal{V}, \tag{14b}$$

$$\mathcal{V}_p \cap \mathcal{V}_q = \emptyset, \qquad \forall\, p, q \in \{1, \ldots, N_g\}, p \neq q, \tag{14c}$$

$$d(\mathcal{G}_p) < N_{\text{CL,al.}}, \qquad \forall\, p \in \{1, \ldots, N_g\}, \tag{14d}$$

[1]The depth of a graph is the length of the longest path between any two of its vertices.

The classical min-cut clustering problem consists of (14a) to (14c) and is NP-hard [40]. We additionally limit each subgraph's depth with (14d).

We propose to solve (14) with our greedy Algorithm 1. We iterate until the depth of all subgraphs in the partition is lower than the desired allowed number of computation levels (Line 3). In each iteration, we min-cut [41] the deepest subgraph (Line 4), which splits an edge-weighted graph into two subgraphs such that the sum of cut weight is minimal. Our algorithm returns a partition $\mathit{\Gamma}$ composed of subgraphs with a guaranteed maximum depth, resulting in a limit on the computation levels $N_{\text{CL,al.}}$.

Proposition 1. Algorithm 1 terminates at worst in $|\mathcal{V}|-1$ iterations.

Proof. Since Algorithm 1 separates two vertices in each iteration, all vertices are in separate subgraphs after $|\mathcal{V}|-1$ iterations. □

Proposition 2. The computational complexity of Algorithm 1 is $O(|\mathcal{V}|^2|\mathcal{E}|+|\mathcal{V}|^3\log|\mathcal{V}|)$.

Proof. Given Proposition 1, the complexity follows directly from the analysis in [41]. □

Algorithm 1 Greedy min-cut for partition with limited depth

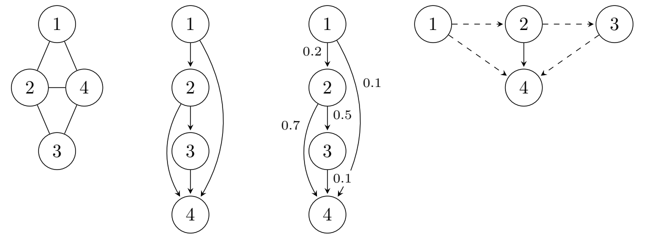
```
Input: edge-weighted graph $\vec{\mathcal{G}}_w$, maximum levels $N_{\text{CL,al.}}$
Output: Partition $\mathit{\Gamma}$ (depth of subgraphs limited to $N_{\text{CL,al.}}$)
1: $\vec{\mathcal{G}}_{w,\max} \leftarrow \vec{\mathcal{G}}_w$                ▷ initialize deepest subgraph
2: $\mathit{\Gamma} \leftarrow \{\vec{\mathcal{G}}_{w,\max}\}$                ▷ initialize partition
3: while $d(\vec{\mathcal{G}}_{w,\max}) \geq N_{\text{CL,al.}}$ do
4:     $\{\vec{\mathcal{G}}_{w,a}, \vec{\mathcal{G}}_{w,b}\} \leftarrow$ min-cut $\vec{\mathcal{G}}_{w,\max}$
5:     $\mathit{\Gamma} \leftarrow (\mathit{\Gamma} \setminus \vec{\mathcal{G}}_{w,\max}) \cup \{\vec{\mathcal{G}}_{w,a}, \vec{\mathcal{G}}_{w,b}\}$ ▷ replace subgraph
6:     $\vec{\mathcal{G}}_{w,\max} \leftarrow$ deepest subgraph in $\mathit{\Gamma}$
7: return $\mathit{\Gamma}$
```

C. Example of Limiting Computation Time by Grouping

Figure 4 exemplifies the steps in our framework for four vehicles and a computation level limit of two. First, we couple vehicles if their reachable sets intersect (Fig. 4a). Second, we prioritize vehicles. In this example, priorities are equal to the vertex number (Fig. 4b). Note that sequential P-DMPC would result in four computation levels in this case. Third, we weigh couplings using the STC (Fig. 4c). Fourth, we apply Algorithm 1 to partition the graph (Fig. 4d). Two cuts are required in this example. The first cut separates vehicle 1 with a cut weight of 0.3, the second cut separates vehicle 3 with a cut weight of 0.6. This results in the desired computation level limit of two.

IV. Evaluation

We evaluate our framework numerically in MATLAB. The simulation setup replicates the Cyber-Physical Mobility Lab [42], an open-source, small-scale testbed for connected and automated vehicles (CAVs). Its road network consists of an



(a) Couple (b) Prioritize (c) Weigh (d) Group

Fig. 4: Example of four vehicles in our distributed planning framework illustrated in Fig. 2: (a) couple vehicles ($\mathcal{G}$), (b) prioritize vehicles ($\vec{\mathcal{G}}$), (c) weigh couplings ($\vec{\mathcal{G}}_w$), (d) group vehicles ($\mathit{\Gamma}$). Solid lines indicate sequential couplings, dashed lines indicate parallel couplings.

urban intersection, a highway, and highway on- and off-ramps. We use a horizon of $N_p = 7$ and a sample time for the NCS of $T_s = 0.2\,\text{s}$. The code[2] to reproduce the results and a video[3] are available online.

A. Evaluation of Safe Planning Despite Prediction Inconsistency

Figure 5 visualizes a simulation to compare our method, which considers reachable sets (right), with state-of-the-art methods that consider trajectories of the previous time step (left) [6], [18]. In this simulation, three vehicles cross an intersection, all vehicles compute in parallel, all vehicles are coupled, and priorities are equal to the vehicle numbers.

Figure 5a shows the footprints of all vehicles in the simulation. On the left side, two vehicles collide at time step $k = 8$. The occupancies of the vehicles at this time step are depicted in red. On the right side, all trajectories are safe due to our trajectory planning method using reachability analysis.

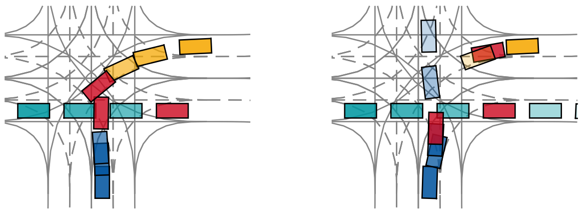
Figure 5b visualizes vehicle 3's parallel coupling constraints at a critical time step $k = 5$ over the horizon and its resulting trajectory prediction. On the left, the constraints are the time-shifted occupancies from the previous time step. On the right, the constraints are the reachable sets. The vehicle enters the intersection further on the left than on the right. In both cases, there is no collision in vehicle 3's OCP.

Figure 5c depicts the actual trajectory predictions of all vehicles. On the left, the trajectory predictions of vehicles 2 and 3 actually intersect due to the prediction inconsistency. After executing the first step of its trajectory prediction, vehicle 3 will fail to find feasible solutions to its OCP in the following time steps, leading to a collision. On the right, the trajectory predictions do not intersect. Besides showing the safety of our approach, this example also illustrates its conservativeness, as vehicle 3 completely stops in front of the intersection to avoid vehicle 1's reachable set.

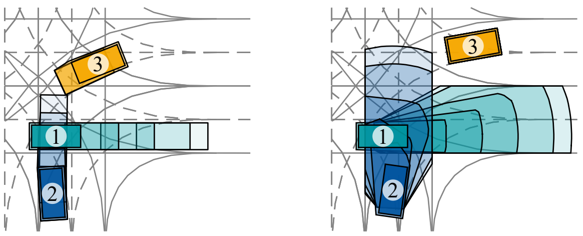
B. Evaluation of Grouping Effect on Solution Quality

In order to evaluate the effect of grouping vehicles on the solution quality, we simulated 20 vehicles on the road

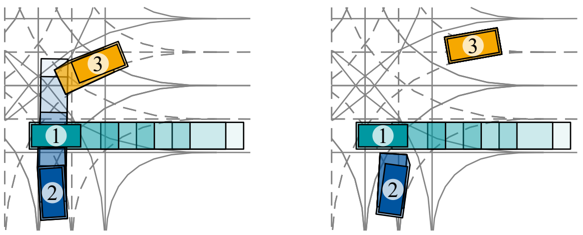
[2]github.com/embedded-software-laboratory/p-dmpc
[3]youtu.be/di6X6XTGt88



(a) Footprint of even time steps. Timestep $k = 8$ in red, left colliding, right safe.



(b) Parallel coupling constraints and trajectory prediction of vehicle 3 at a critical time step $k = 5$. Left: assumed predictions, right: reachable sets.



(c) Actual predictions of the scene in Fig. 5b.

Fig. 5: Parallel trajectory planning with P-DMPC. Parallel coupling constraints are on the left time-shifted previous trajectories [6], [18], on the right reachable sets (our approach). Occupancies are inflated to account for uncertainty.

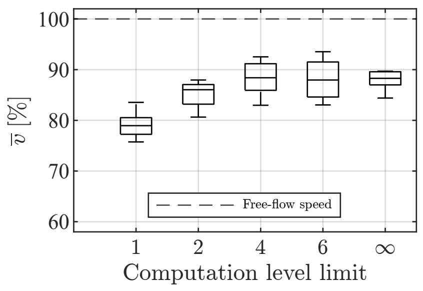


Fig. 6: Effect of computation level limit on solution quality. $\bar{v}$: normalized average speed. Computation level limit of 1: purely parallel computation, computation level limit of ∞: purely sequential computation [5].

network. In each of the ten simulations, we change the reference paths of the vehicles. Their goals are to stay close to their reference paths with desired speeds.

We define the solution quality in these simulations as the normalized average speed $\bar{v}$, which is the relation of the average speed over all vehicles and time steps to the free-flow speed. We compute the free-flow speed as the average speed over all vehicles and time steps by ignoring collisions. A lower normalized average speed indicates a lower solution quality.

Figure 6 shows the normalized average speeds over different allowed numbers of computation levels $N_{\text{CL,al.}}$. A limit of $N_{\text{CL,al.}} = 1$ corresponds to purely parallel computation, and an infinite limit corresponds to purely sequential computation [5]. The median of the normalized average speed and thus the solution quality increases with the computation level limit in our simulations. From $N_{\text{CL,al.}} = 1$ to $N_{\text{CL,al.}} = 4$ and above, the median speed increases by a total of $10\,\%$. The median speed does not increase after $N_{\text{CL,al.}} = 4$, indicating that in our simulations, a computation level limit of $N_{\text{CL,al.}} = 4$ has no significant negative effect on the solution quality. In purely sequential computation [5], the maximum number of computation levels in our simulations is eleven. Consequently, by setting the allowed number of computation levels to four, our method reduces the actual number of computation levels by about 64% compared to sequential computation, while still guaranteeing collision avoidance among vehicles and maintaining the solution quality.

V. Conclusion

With our distributed planning framework, it is possible to parallelize all computations in P-DMPC through reachability analysis without jeopardizing safety. However, if all computations are parallelized, our approach to guarantee safety leads to conservative solutions. Therefore, we improve the solution quality by sequentializing a subset of computations, while maintaining the ability to limit the number of computation levels. Our approach of guaranteeing a limit on computation levels can be beneficial in P-DMPC for NCSs when the maximum number of computation levels is unknown or very high, such as in trajectory planning for CAVs. Without our method, the sample time must then be chosen such that the worst case of $N_{\text{CL}} = N_A = 20$ computation levels is allowed. In our simulations, we could reduce the maximum number of computation levels compared to sequential planning by about 64%, while guaranteeing safety, and without a significant negative effect on the solution quality.

If the computation time of each agent is bounded, we can determine the allowed number of computation levels for real-time trajectory planning. Therefore, we are working on an anytime trajectory planning algorithm for NCSs, which correlates the overall trajectory planning time with the number of computation levels.

References

[1] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng, "Networked control systems: A survey of trends and techniques," *IEEECAA J. Autom. Sin.*, vol. 7, no. 1, pp. 1–17, Jan. 2020.

[2] J. Lunze, *Control Theory of Digitally Networked Dynamic Systems*. Springer, 2014, vol. 1.

[3] B. Alrifaee, "Networked Model Predictive Control for Vehicle Collision Avoidance," Ph.D. dissertation, RWTH Aachen University, 2017.

[4] M. Kloock, P. Scheffe, S. Marquardt, J. Maczijewski, B. Alrifaee, and S. Kowalewski, "Distributed Model Predictive Intersection Control of Multiple Vehicles," in *IEEE Intell. Transp. Syst. Conf. ITSC*, 2019, pp. 1735–1740.

[5] B. Alrifaee, F.-J. Heßeler, and D. Abel, "Coordinated Non-Cooperative Distributed Model Predictive Control for Decoupled Systems Using Graphs," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 216–221, 2016.

[6] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How, "Distributed Robust Receding Horizon Control for Multivehicle Guidance," *IEEE Trans. Contr. Syst. Technol.*, vol. 15, no. 4, pp. 627–641, Jul. 2007.

[7] P. Scheffe, J. Kahle, and B. Alrifaee, "Reducing Computation Time with Priority Assignment in Distributed Control," 2023, Preprint, 10.36227/techrxiv.20304015.v2.

[8] J. Li, M. Ran, and L. Xie, "Efficient Trajectory Planning for Multiple Non-Holonomic Mobile Robots via Prioritized Trajectory Optimization," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 405–412, Apr. 2021.

[9] M. Čáp, P. Novák, A. Kleiner, and M. Selecký, "Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 3, pp. 835–849, Jul. 2015.

[10] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE Int. Conf. Robot. Autom.* Shanghai, China: IEEE, May 2011, pp. 2520–2525.

[11] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE Int. Conf. Robot. Autom.* St Paul, MN, USA: IEEE, May 2012, pp. 477–483.

[12] P. Scheffe, T. M. Henneken, M. Kloock, and B. Alrifaee, "Sequential Convex Programming Methods for Real-Time Optimal Trajectory Planning in Autonomous Vehicle Racing," *IEEE Trans. Intell. Veh.*, vol. 8, no. 1, pp. 661–672, 2023.

[13] S. A. N. Nouwens, M. M. Paulides, and M. Heemels, "Constraint-adaptive MPC for linear systems: A system-theoretic framework for speeding up MPC through online constraint removal," *Automatica*, vol. 157, p. 111243, Nov. 2023.

[14] A. P. Aguiar, F. A. Bayer, J. Hauser, A. J. Häusler, G. Notarstefano, A. M. Pascoal, A. Rucco, and A. Saccon, "Constrained optimal motion planning for autonomous vehicles using PRONTO," *Sens. Control Auton. Veh. Appl. Land Water Air Veh.*, pp. 207–226, 2017.

[15] M. Chen, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Robust sequential trajectory planning under disturbances and adversarial intruder," *IEEE Trans. Control Syst. Technol.*, vol. 27, no. 4, pp. 1566–1582, 2018.

[16] S. Bansal, M. Chen, K. Tanabe, and C. J. Tomlin, "Provably Safe and Scalable Multivehicle Trajectory Planning," *IEEE Trans. Contr. Syst. Technol.*, vol. 29, no. 6, pp. 2473–2489, Nov. 2021.

[17] M. Gulzar, Y. Muhammad, and N. Muhammad, "A Survey on Motion Prediction of Pedestrians and Vehicles for Autonomous Driving," *IEEE Access*, vol. 9, pp. 137 957–137 969, 2021.

[18] X. Shen and F. Borrelli, "Reinforcement Learning and Distributed Model Predictive Control for Conflict Resolution in Highly Constrained Spaces," in *2023 IEEE Intell. Veh. Symp. IV*, Jun. 2023.

[19] N. Kaempchen, K. Weiss, M. Schaefer, and K. Dietmayer, "IMM object tracking for high dynamic driving maneuvers," in *IEEE Intell. Veh. Symp. 2004*, Jun. 2004, pp. 825–830.

[20] E. Coelingh, A. Eidehall, and M. Bengtsson, "Collision Warning with Full Auto Brake and Pedestrian Detection - a practical example of Automatic Emergency Braking," in *13th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2010, pp. 155–160.

[21] Y. Guo, V. V. Kalidindi, M. Arief, W. Wang, J. Zhu, H. Peng, and D. Zhao, "Modeling Multi-Vehicle Interaction Scenarios Using Gaussian Random Field," in *2019 IEEE Intell. Transp. Syst. Conf. ITSC*, Oct. 2019, pp. 3974–3980.

[22] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks," in *2019 Int. Conf. Robot. Autom. ICRA*. Montreal, QC, Canada: IEEE, May 2019, pp. 2090–2096.

[23] M. Kloock and B. Alrifaee, "Coordinated Cooperative Distributed Decision-Making Using Synchronization of Local Plans," *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1292–1306, 2023.

[24] J. Alonso-Mora, E. Montijano, M. Schwager, and D. Rus, "Distributed multi-robot formation control among obstacles: A geometric and optimization approach with consensus," in *2016 IEEE Int. Conf. Robot. Autom. ICRA*, May 2016, pp. 5356–5363.

[25] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, Feb. 2015.

[26] L. Cohen, T. Uras, T. K. S. Kumar, H. Xu, N. Ayanian, and S. Koenig, "Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding," in *Proc. Twenty-Fifth Int. Jt. Conf. Artif. Intell.* New York, New York, USA: AAAI Press, 2016, p. 8.

[27] L. Cohen, G. Wagner, D. Chan, H. Choset, N. Sturtevant, S. Koenig, and T. K. S. Kumar, "Rapid Randomized Restarts for Multi-Agent Path Finding Solvers," in *Elev. Annu. Symp. Comb. Search*, Jul. 2018.

[28] Y. Ouyang, B. Li, Y. Zhang, T. Acarman, Y. Guo, and T. Zhang, "Fast and Optimal Trajectory Planning for Multiple Vehicles in a Nonconvex and Cluttered Environment: Benchmarks, Methodology, and Experiments," in *2022 Int. Conf. Robot. Autom. ICRA*. Philadelphia, PA, USA: IEEE, May 2022, pp. 10 746–10 752.

[29] W. B. Dunbar and D. S. Caveney, "Distributed Receding Horizon Control of Vehicle Platoons: Stability and String Stability," *IEEE Trans. Autom. Control*, vol. 57, no. 3, pp. 620–633, Mar. 2012.

[30] M. Farina and R. Scattolini, "Distributed predictive control: A non-cooperative algorithm with neighbor-to-neighbor communication for linear systems," *Automatica*, vol. 48, no. 6, pp. 1088–1096, Jun. 2012.

[31] M. Defoort, A. Kokosy, T. Floquet, W. Perruquetti, and J. Palos, "Motion planning for cooperative unicycle-type mobile robots with limited sensing ranges: A distributed receding horizon approach," *Robotics and Autonomous Systems*, vol. 57, no. 11, pp. 1094–1106, Nov. 2009.

[32] G. Demesure, M. Defoort, A. Bekrar, D. Trentesaux, and M. Djemaï, "Decentralized Motion Planning and Scheduling of AGVs in an FMS," *IEEE Trans. Ind. Inform.*, vol. 14, no. 4, pp. 1744–1752, Apr. 2018.

[33] P. Scheffe, G. Dorndorf, and B. Alrifaee, "Increasing Feasibility with Dynamic Priority Assignment in Distributed Trajectory Planning for Road Vehicles," in *IEEE Int. Conf. Intell. Transp. Syst. ITSC*, 2022, pp. 3873–3879.

[34] M. Althoff and S. Magdici, "Set-Based Prediction of Traffic Participants on Arbitrary Road Networks," *IEEE Trans. Intell. Veh.*, vol. 1, no. 2, pp. 187–202, Jun. 2016.

[35] P. Scheffe, M. V. A. Pedrosa, K. Flaßkamp, and B. Alrifaee, "Receding Horizon Control Using Graph Search for Multi-Agent Trajectory Planning," *IEEE Trans. Control Syst. Technol.*, vol. 31, no. 3, pp. 1092–1105, 2023.

[36] C. Pek and M. Althoff, "Fail-Safe Motion Planning for Online Verification of Autonomous Vehicles Using Convex Optimization," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 798–814, Jun. 2021.

[37] R. Rajamani, *Vehicle Dynamics and Control*, ser. Mechanical Engineering Series. New York: Springer Science, 2006.

[38] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT*," in *2011 IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1478–1483.

[39] W. Wachenfeld, P. Junietz, R. Wenzel, and H. Winner, "The worst-time-to-collision metric for situation identification," in *2016 IEEE Intell. Veh. Symp. IV*. Gotenburg, Sweden: IEEE, Jun. 2016, pp. 729–734.

[40] E. L. Johnson, A. Mehrotra, and G. L. Nemhauser, "Min-cut clustering," *Mathematical Programming*, vol. 62, no. 1-3, pp. 133–151, Feb. 1993.

[41] M. Stoer and F. Wagner, "A simple min-cut algorithm," *J. ACM*, vol. 44, no. 4, pp. 585–591, Jul. 1997.

[42] M. Kloock, P. Scheffe, J. Maczijewski, A. Kampmann, A. Mokhtarian, S. Kowalewski, and B. Alrifaee, "Cyber-Physical Mobility Lab: An Open-Source Platform for Networked and Autonomous Vehicles," in *Eur. Control Conf. ECC*, 2021, pp. 1937–1944.