Prioritized Trajectory Planning for Networked Vehicles Using Motion Primitives



Patrick Scheffe, Matheus V. A. Pedrosa, Kathrin Flaßkamp, and Bassam Alrifaee

Abstract The computation time required to solve nonconvex, nonlinear optimization problems increases rapidly with their size. This poses a challenge in trajectory planning for multiple networked vehicles with collision avoidance. In the centralized formulation, the optimization problem size increases with the number of vehicles in the networked control system (NCS), rendering the formulation unusable for experiments. We investigate two methods to decrease the complexity of networked trajectory planning. First, we approximate the optimization problem by discretizing the vehicle dynamics with an automaton, which turns it into a graph-search problem. Our search-based trajectory planning algorithm has a limited horizon to further decrease computation complexity. We achieve recursive feasibility by design of the automaton which models the vehicle dynamics. Second, we distribute the optimization problem to the vehicles with prioritized distributed model predictive control (P-DMPC), which reduces the problem size. To counter the incompleteness of P-DMPC, we propose a framework for time-variant priority assignment. The framework expands recursive feasibility to every vehicle in the NCS. We present two time-variant priority assignment algorithms for road vehicles, one to improve vehicle progress and one to improve computation time of the NCS. We evaluate our approach for online trajectory planning of multiple networked vehicles in simulations and experiments.

Code https://github.com/embedded-software-laboratory/p-dmpc

P. Scheffe (✉)
Chair of Embedded Software, RWTH Aachen University, Aachen, Germany
e-mail: scheffe@embedded.rwth-aachen.de

M. V. A. Pedrosa · K. Flaßkamp
Chair of Systems Modeling and Simulation, Systems Engineering, Saarland University, Saarbrücken, Germany
e-mail: matheus.pedrosa@uni-saarland.de

K. Flaßkamp
e-mail: kathrin.flasskamp@uni-saarland.de

B. Alrifaee
Department of Aerospace Engineering, University of the Bundeswehr Munich, Neubiberg, Germany
e-mail: alrifaee@embedded.rwth-aachen.de; bassam.alrifaee@unibw.de

© The Author(s) 2024

C. Stiller et al. (eds.), *Cooperatively Interacting Vehicles*,
https://doi.org/10.1007/978-3-031-60494-2_9

1 Introduction

Networked and autonomous vehicles (NAVs) have the potential to increase the safety and efficiency of traffic [42]. Realizing this potential requires advances in many fields of networked and autonomous vehicles (NAVs), among which is the field of decision making [56]. In decision making, we develop a plan and control the actuators of the vehicle to execute this plan. Planning can be decomposed into three hierarchical layers. The highest layer plans a route through the road network, the middle layer plans behaviors for the vehicle on the road, and the bottom layer plans motions to realize the behavioral plan [46]. The work in this article focuses on the middle and bottom layer of planning for a multi-agent system. We will refer to this area as trajectory planning for multiple NAVs. Section 1.1 motivates our work on networked trajectory planning, Sect. 1.2 presents the state of the art and Sect. 1.3 states our contribution to the state of the art. We introduce our notation in Sect. 1.4 and give an overview of this chapter in Sect. 1.5.

1.1 Motivation

Trajectory planning for multiple NAVs with collision avoidance can be modeled as a nonconvex, nonlinear optimal control problem (OCP). For trajectory planning in changing environments, this OCP must be solved within a duration of tenths of a second. With an increasing amount of controlled vehicles, the OCP grows large, and finding a solution quickly becomes intractable. This chapter investigates two approaches to decrease computation time of networked trajectory planning: simplifying and distributing the OCP.

When simplifying the OCP, a compromise between global optimality and computational efficiency must be found [12]. Trajectory planning approaches can be classified as optimization-based and graph-based [46]. Optimization-based algorithms are often based on convexification of the original nonconvex OCP [5, 6, 28, 52, 58]. The advantage of convexification is a short computation time, which comes at the cost of disregarding nonlinearities in the vehicle model and of disregarding parts of the solution space. Graph-based methods based on motion primitives (MPs) can retain the nonlinearities and the complete solution space. The coarseness of quantization of states and control inputs highly influences the computational complexity and the trajectory quality.

Distributing the centralized OCP, which plans trajectories for all vehicles at once, reduces the computational effort at the cost of global system knowledge. Prioritized trajectory planning for vehicles is first presented in [21]. In a prioritized approach, vehicles with lower priority adjust their objectives and constraints to respect coupled vehicles with higher priority. The core problem of prioritized planning algorithms is their incompleteness. That is, there might exist a priority assignment that leads to

feasible optimization problems of all participating agents, but the algorithm can fail to find it.

1.2 Related Work

This section presents related work on trajectory planning with MPs and on prioritized trajectory planning.

1.2.1 Trajectory Planning with Motion Primitives

The goal of trajectory planning with MPs is to find an optimal sequence and duration of MPs that achieve a desired objective while satisfying constraints. MP consists of a control and state trajectory. Multiple MPs can be concatenated to form a vehicle trajectory plan. There are mainly two kinds of methods to plan trajectories using MPs: methods based on continuous optimization problem formulations, such as mixed integer programming (MIP), and methods with graph-based problem formulations, such as an A* algorithm or a rapidly-exploring random tree algorithm [39]. A literature review on both methods follows.

MIP formulates an OCP with both continuous and discrete variables. MIP can find the optimal sequence and duration of MPs for trajectory planning of a single vehicle [23, 26, 27]. When dealing with multiple NAVs, collision constraints can be modeled with binary decision variables [7]. The ability of MIP to find the optimal solution comes at the cost of high computation time, which rapidly increases with the size of the OCP. Centralized trajectory planning for multiple vehicles with MPs [2, 20, 22] encounters this problem.

A popular search algorithm for trajectory planning using MPs is A* and its variant hybrid A* [1, 19, 49]. When operating on a gridded environment representation, A* associates a cost value with a grid cell center and the cell center's state value, whereas hybrid A* associates a cost value alongside a continuous state value with a grid cell. A computationally demanding task in search algorithms for trajectory planning are edge evaluations, as they incorporate the collision constraints [39]. The number of edge evaluations can be reduced using a lazy approach, in which an edge is only evaluated when the connected vertex is chosen for expansion [17, 18, 43]. The computation time of graph-search algorithms increases with the length of its horizon. Limiting the horizon decreases computation time [9, 36, 45]. Algorithms for graph-based trajectory planning for multiple NAVs include a Monte Carlo tree search [37] and a traditional A* graph search [24, 25]. Graph searches with an infinite horizon suffer from high computation time [17–19, 43, 49]. This challenge can be overcome with a receding horizon at the cost of global optimality guarantees. Graph-based receding horizon approaches do not yet guarantee recursive feasibility [9, 40, 45].

1.2.2 Prioritized Distributed Control

The distributed control strategy for networked control system (NCS) examined in this work is prioritized distributed model predictive control (P-DMPC), in which each vehicle optimizes only its own decision variables. Lower prioritized vehicles consider a communicated optimized solution of coupled higher prioritized vehicles in both in their objective function and their constraints. The benefit of the greedy P-DMPC algorithm is its short computation time [3, 57]. One of the main challenges in P-DMPC is its incompleteness [40]. That means, a priority assignment might lead to an infeasible OCP of a vehicle although the problem is solvable with a different priority assignment. Additionally, the priority assignment influences the solution quality and the computation time.

The following works have designed priority assignments for robots and NAVs with the goal of feasibility and solution quality. In our work [32] the ordering is based on rules, i.e., we assign time-variant priorities to multiple vehicles competing on a racetrack based on their race position. Constraint-based heuristics increase the priority of a vehicle with the number of constraints it has [13, 16, 41, 48, 60]. The goal of these heuristics is to maintain feasibility of the control problems. In our work [35], we assign priorities to vehicles based on the time remaining before they enter an intersection. In our work [31], we assign priorities to vehicles based on the crowdedness of their goal location. Objective-based heuristics assign priorities to improve the solution quality of the NCS [15, 59]. A randomized priority assignment with hill-climbing is proposed in [10]. In [8], all priority assignments are considered to find the optimal one. Both approaches achieve higher solution quality with higher computation time. In [61], priorities are assigned based on machine learning and achieve results competitive to heuristics. The priority assignment can also influence computation time [4]. The number of simultaneous computations in prioritized planning is maximized in [38]. Despite the number of priority assignment strategies, the incompleteness of P-DMPC remains. Many works assign time-invariant priorities for a specific scenario [13, 16, 38, 41, 48, 59, 60]. Time-variant priority assignments improve feasibility in changing operating conditions over time-invariant priority assignments [10, 15]. In [38], time-invariant priorities are shown to produce recursively feasible solutions. Similarly, this property needs to be shown for time-variant priorities.

1.3 Contribution

The contribution of this chapter is twofold. First, we present our method of receding horizon graph search (RHGS), a search-based trajectory planning algorithm for road vehicles. We reduce the computation time by limiting the planning horizon. We prove that our method fulfills recursive feasibility by design of the motion primitive automaton (MPA) [55]. Second, we present a framework for distributed reprioritiza-

tion of vehicles. We prove that it fulfills recursive NCS-feasibility for P-DMPC with any time-variant priority assignment algorithm [51].

We present two priority assignment algorithms, one for vehicle progression using a constraint-based heuristic, and a one for computation time reduction of the NCS using graph coloring. We demonstrate the effectiveness of the presented approach in a simulative case study of P-DMPC for trajectory planning.

1.4 Notation

A variable x is marked with a superscript $x^{(j)}$ if belonging to agent j, and with $x^{(-j)}$ if belonging to the neighbors of agent j. The actual value of a variable x at time k is written as $x(k)$, while values predicted for time $k+i$ at time k are written as $x_{k+i|k}$. A trajectory is denoted by substituting the time argument with $\cdot$ as in $x_{\cdot|k}$. An agent equals a vehicle in our application of prioritized trajectory plannning. In this chapter, we use the terms vehicle, road vehicle and NAV interchangeably.

1.5 Structure

The remainder of this chapter is structured as follows. Section 2 presents our vehicle model, our RHGS for trajectory planning, and our proof of recursive agent-feasibility. Section 3 presents the distribution of RHGS with P-DMPC for trajectory planning. We show recursive NCS-feasibility of our reprioritization framework before presenting two time-variant priority assignment algorithms, one for vehicle progression and one for computation time reduction. In Sect. 4, we evaluate both the RHGS and the P-DMPC in experiment, before combining both in a simulative case study.

2 Receding Horizon Graph Search for Trajectory Planning

This section presents how we transfer a receding horizon control (RHC) approach to graph-based trajectory planning. The content is based on our previous publication [55]. Section 2.1 states the RHC trajectory planning problem that we subsequently map to graph search based on an MPA. Section 2.2 presents our approximation of the vehicle dynamics as an MPA, Sect. 2.3 shows the graph-based optimization in our RHGS algorithm. In Sect. 2.4, we prove that our RHGS produces recursively agent-feasible trajectories by design of the MPA.

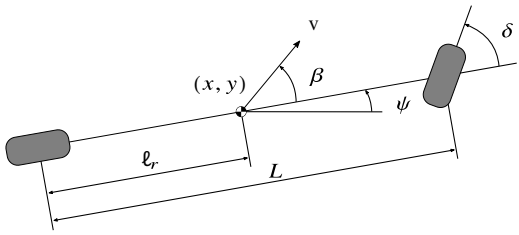

Fig. 1 Kinematic single-track model of a vehicle [55]

2.1 Trajectory Planning Problem

This section presents the ordinary differential equations describing the vehicle dynamics and our cost function before both are incorporated in a RHC problem for trajectory planning.

Figure 1 shows an overview of the variables for the nonlinear kinematic single-track model [47]. Assuming low velocities, we model no slip on the front and rear wheels, and no forces acting on the vehicle. The resulting equations are

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

with

$$\beta(t) = \tan^{-1}\left(\frac{\ell_r}{L}\tan(\delta(t))\right), \tag{2}$$

where $x \in \mathbb{R}$ 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 $u_{\mathrm{v}} \in \mathbb{R}$ 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. The position of the CG and the orientation together form the pose $\boldsymbol{p}$.

The system dynamics defined in (1) are compactly written as

$$\dot{\boldsymbol{x}}(t) := \frac{d}{dt}\boldsymbol{x}(t) = f\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \tag{3}$$

with the state vector

$$\boldsymbol{x} = \big(x \;\; y \;\; \psi \;\; \mathrm{v} \;\; \delta\big)^{\mathrm{T}} \in \mathbb{R}^5, \tag{4}$$

the control input

$$\boldsymbol{u} = \begin{pmatrix} u_{\mathrm{v}} & u_{\mathrm{v}} \end{pmatrix}^{\mathrm{T}} \in \mathbb{R}^2 \tag{5}$$

and the vector field f defined by (1). Transferring (3) to a discrete-time nonlinear system representation yields

$$\boldsymbol{x}_{k+1} = f_d\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \tag{6}$$

with $k \in \mathbb{N}$, the vector field $f_d \colon \mathbb{R}^5 \times \mathbb{R}^2 \to \mathbb{R}^5$, the state vector $\boldsymbol{x} \in \mathbb{R}^5$ and the input vector $\boldsymbol{u} \in \mathbb{R}^2$.

We define the cost function to minimize in our trajectory planning problem as

$$J_{k \to k+N|k} = \sum_{i=1}^{N} \left(\boldsymbol{x}_{k+i|k} - \boldsymbol{x}_{\mathrm{ref},k+i|k}\right)^{\mathrm{T}} \boldsymbol{Q} \left(\boldsymbol{x}_{k+i|k} - \boldsymbol{x}_{\mathrm{ref},k+i|k}\right) \tag{7}$$

with the planning horizon length N, the positive semi-definite, block diagonal matrix

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{I}_2 & \boldsymbol{0}_{2\times 3} \\ \boldsymbol{0}_{3\times 2} & \boldsymbol{0}_3 \end{pmatrix} \in \mathbb{R}^{5\times 5} \tag{8}$$

and a reference trajectory $\boldsymbol{x}_{\mathrm{ref},\cdot|k} \in \mathbb{R}^5$.

We combine the system model (6) and the cost function (7) to an OCP

$$\underset{U_{k\to k+N|k}}{\text{minimize}} \quad J_{k\to k+N|k} \tag{9a}$$

subject to

$$\boldsymbol{x}_{k+i+1|k} = f_d\left(\boldsymbol{x}_{k+i|k}, \boldsymbol{u}_{k+i|k}\right), \quad i = 0, \ldots, N-1 \tag{9b}$$

$$\boldsymbol{u}_{k+i|k} \in \mathcal{U}, \quad i = 0, \ldots, N-1 \tag{9c}$$

$$\boldsymbol{x}_{k+i|k} \in \mathcal{X} \quad i = 1, \ldots, N-1 \tag{9d}$$

$$\boldsymbol{x}_{k+N|k} \in \mathcal{X}_f \tag{9e}$$

$$\boldsymbol{x}_{k|k} = \boldsymbol{x}(k) \tag{9f}$$

with the vector $U_{k\to k+N|k}$ of stacked control inputs $(\boldsymbol{u}_{k|k}, \boldsymbol{u}_{k+1|k}, \ldots, \boldsymbol{u}_{k+N-1|k})$, the input constraint set $\mathcal{U} \subseteq \mathbb{R}^2$, the state constraint set $\mathcal{X} \subseteq \mathbb{R}^5$ and the terminal set $\mathcal{X}_f \subseteq \mathbb{R}^5$. We assume a full measurement or estimate of the state $\boldsymbol{x}(k)$ is available at the current time k. The OCP (9) is solved repeatedly after a timestep duration T_s and with updated values for the states and constraints, which establishes the RHC.

2.2 Motion Primitive Automaton as System Model

This section presents how we model the state-continuous system (6) as an MPA, a type of maneuver automaton [23]. The MPA incorporates the constraints on system

dynamics (9b), on control inputs (9c), and on both the steering angle and the speed (9d) and (9e).[1]

From the system dynamics (1), we derive a finite state automaton which we call MPA and define as follows.

Definition 1 (*Motion primitive automaton*) An MPA is a 5-tuple (Q, S, γ, q_0, Q_f) composed of:

- Q is a finite set of automaton states q;
- S is a finite set of transitions π, also called motion primitives;
- $\gamma : Q \times S \times \mathbb{N} \to Q$ is the update function defining the transition from one automaton state to another, dependent on the timestep in the horizon;
- $q_0 \in Q$ is the initial automaton state;
- $Q_f \subseteq Q$ is the set of final automaton states.

An automaton state is characterized by a specific speed v and steering angle δ. An MP is characterized by an input trajectory and a corresponding state trajectory which starts and ends with the speed and steering angle of an automaton state. It has a fixed duration which we choose equal to the timestep duration T_s. MPs can be concatenated to vehicle trajectories by rotation and translation. Our MPA discretizes both the state space with the update function γ and the time space with a fixed duration T_s for all MPs. This MPA replaces the system representation (6). Note that the dynamics model on which our MPA is based is exchangeable. Its complexity is irrelevant computation-wise for trajectory planning since MPs are computed offline.

2.3 Receding Horizon Graph Search Algorithm

This section demonstrates how our RHGS incorporates the constraints on the pose, which are included in (9d) and (9e), while minimizing the cost function (9a).

Our RHGS algorithm constructs a search tree $\mathcal{T}$ up to a limited depth N. A level i in the tree directly corresponds to the timestep $k + i$ in the OCP (9). The information contained in each vertex v of the tree is a tuple $\langle q, \boldsymbol{p}, i, J \rangle$, whose elements are the automaton state, the vehicle pose, the distance to the root vertex, and the value of the cost function, respectively. When the algorithm finds the leaf vertex with the minimal cost value at the horizon $k + N$, it returns the path from the root vertex to this leaf vertex. The algorithm ensures optimality of the returned path with an admissible and underestimating cost estimation, similar to A*.

[1] A detailed explanation of modeling with MPAs is found in this book's chapter "Designing Maneuver Automata of Motion Primitives for Optimal Cooperative Trajectory Planning".

Algorithm 1 Receding Horizon Graph Search

Input: initial vertex v_0, MPA, goal set $\mathcal{X}_f$
Output: path from v_0 to best vertex v_p

```
1:  L_open ← v_0
2:  while L_open ≠ ∅ do
3:      Sort L_open ascending by J = J_CTC + J_CTG
4:      v_p ← L_open[0]
5:      L_open ← L_open \ v_p
6:      if not IsValid(v_p) then
7:          continue
8:      if i_{v_p} = N and x_{v_p} ∈ X_f then
9:          return path from v_0 to v_p
10:     successors← Expand(v_p)
11:     for all v_s ∈ successors do
12:         LazyEval(v_s)
13:         L_open ← L_open ∪ v_s
14: return failure
```

Algorithm 1 shows the main steps of our RHGS algorithm. At the beginning of the control loop at time k, the algorithm determines the search tree's root vertex v_0 from the state vector $\boldsymbol{x}(k)$ and initializes the open list with this root vertex (Line 1). Sorting the open list by the cost function value brings the vertex with the lowest cost v_p to the front (Line 3). It is removed from the open list (Line 5). We evaluate the edge to the selected vertex by checking inter-vehicle collisions and obstacle collisions (Line 6). If there is a collision, the algorithm continues to the next vertex in the open list. If the vertex is collision-free, satisfies the constraint (9e), and is at the planning horizon N, it is optimal (Line 8). The algorithm returns the path to the vertex (Line 9). Otherwise, the algorithm expands the vertex based on its automaton state q, the update function γ, and its state vector $\boldsymbol{x}$ (Line 10). The algorithm evaluates edges to successors lazily by computing only the cost function without collision checks to reduce computation time (Lines 11 to 12). In informed graph-search algorithms, the cost function consists of the cost-to-come (CTC) and the cost-to-go (CTG). Our algorithm minimizes (7) as the CTC is equal to (7) and the CTG is an underestimation of (7). We underestimate the cost from a vertex v at depth i_v by moving a vehicle towards its reference position at each subsequent timestep with maximum speed in a straight line

$$J_{\mathrm{CTG}}(i_v) = \sum_{i=i_v+1}^{N} \max\Big(0, \\ \left(\boldsymbol{x}_{k+i|k} - \boldsymbol{x}_{\mathrm{ref},k+i|k}\right)^{\mathrm{T}} \boldsymbol{Q} \left(\boldsymbol{x}_{k+i|k} - \boldsymbol{x}_{\mathrm{ref},k+i|k}\right) - i \cdot v_{\max} \cdot T_s\Big) \tag{10}$$

with the same $\boldsymbol{Q}$ as in (7). At the end of the loop, all successor vertices are added to the open list (Line 13).

2.4 Recursive Agent-Feasibility

This section proves recursive agent-feasibility of our RHGS. The property is commonly known as recursive feasibility or persistent feasibility. We design the time-variant update function γ of our MPA such that an equilibrium state can always be reached within the horizon N from expanded successors (Line 10).

A set $C_{\text{inv}} \subseteq \mathcal{X}$ is a control invariant set for the system (6) subject to constraints (9b)–(9f) if

$$\begin{aligned} \boldsymbol{x}(k) \in C_{\text{inv}} \implies &\exists \boldsymbol{u}(k) \in \mathcal{U} \text{ such that} \\ &\boldsymbol{x}(k+1) \in C_{\text{inv}}, \forall k \in \mathbb{N}. \end{aligned} \tag{11}$$

Lemma 1 *If $\mathcal{X}_f$ is a control invariant set of the system (9) with $N > 1$, then (9e) ensures recursive agent-feasibility of the RHC.*

Proof The proof is given in [11]. □

We reformulate the condition of control invariant sets for MPAs as follows.

Definition 2 (*Control invariant set for an MPA*) A set $C_{\text{inv}} \subseteq \mathcal{X}$ is a control invariant set for the system (6) given by an MPA if

$$\begin{aligned} \boldsymbol{x}(k) \in C_{\text{inv}} \text{ with } q(k) \in Q_f \implies &\exists \pi \in S \text{ such that} \\ &\boldsymbol{x}(k+1) \in C_{\text{inv}} \text{ with } q(k+1) \in Q_f \text{ and} \\ &\gamma(q(k), \pi, k) = q(k+1), \forall k \in \mathbb{N}. \end{aligned} \tag{12}$$

Note that the automaton state q follows from the state vector $\boldsymbol{x}$.

Theorem 1 *RHGS achieves recursive agent-feasibility if the generated sequence of transitions ends in an automaton state and a state vector that together form a control invariant set.*

Proof Follows directly from Lemma 1 with Definition 2 of control invariant sets for MPAs. □

In an equilibrium of the system, it holds that $f_d\big(\boldsymbol{x}(k), \boldsymbol{u}(k)\big) = \boldsymbol{x}(k)$. If a sequence of transitions ends in an automaton state from where there exists a transition which keeps the system at an equilibrium, $\boldsymbol{x}(k)$ represents a control invariant set. Such an automaton state in our MPA has a speed $\mathrm{v} = 0\,\mathrm{m\,s^{-1}}$. Figure 2 depicts a simple example of an MPA with a time-invariant update function. This MPA can generate sequences of transitions which are not recursively feasible. We design a time-variant update function which only generates recursively feasible sequences, as shown in an example MPA in Fig. 3.

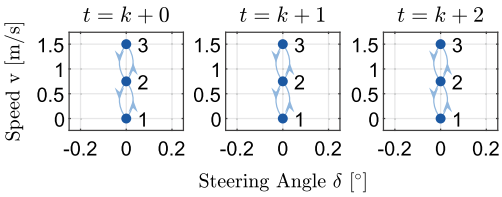

Fig. 2 MPA which does not guarantee recursive agent-feasibility, rolled out over a planning horizon with length $N = 3$

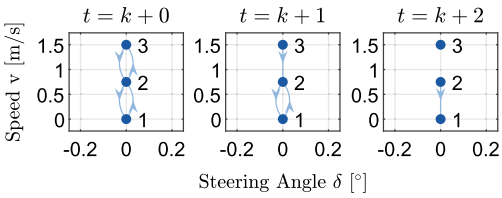

Fig. 3 MPA which guarantees recursive agent-feasibility by only allowing a speed of $0\,\mathrm{m\,s^{-1}}$ at the end of the horizon, rolled out over a planning horizon with length $N = 3$

3 Prioritized Trajectory Planning

This section presents our approach for distributed trajectory planning with distributed reprioritization while guaranteeing recursive NCS-feasibility. It is based on our publications [51, 53]. Our P-DMPC loop consists of the steps coupling, prioritization, trajectory planning, and communication of trajectories. We couple agents if they potentially interact during their planning horizon N. We represent couplings between agents with a coupling graph. Denote by $\mathcal{V} = \{1, \ldots, N_A\}$ the set of agents and by $N_A = |\mathcal{V}| \in \mathbb{N}$ its cardinality.

Definition 3 (*Coupling graph*) A coupling graph $G = (\mathcal{V}, \mathcal{E})$ is a graph that represents the interaction between agents. Vertices represent agents and edges denote coupling objectives or constraints in the OCP associated with the vertex.

The agents connected to agent j are called its neighbors $\mathcal{V}^{(j)}$. Introducing priorities results in clear responsibilities to satisfy collision constraints. We direct edges in the coupling graph from a higher prioritized agent to a lower prioritized agent.

Definition 4 (*Directed coupling graph*) A directed coupling graph $G' = (\mathcal{V}, \mathcal{E}')$ results from a coupling graph $G = (\mathcal{V}, \mathcal{E})$ by keeping all vertices $\mathcal{V}$ and a subset of edges $\mathcal{E}' \subset \mathcal{E}$ of G. In a directed coupling graph, a directed edge denotes a coupling objective or constraint in the OCP associated with the ending vertex.

Vehicles determine their priorities using a priority assignment algorithm. A time-variant priority assignment algorithm yields an injective priority assignment function $p\colon \mathcal{V} \times \mathbb{N} \to \mathbb{N}$, which assigns a unique priority to each vehicle in the NCS at every timestep. If $p(l, k) < p(j, k)$, then vehicle l has a higher priority than vehicle j at timestep k. At each timestep k, every vehicle groups its current neighbors $\mathcal{V}^{(j)}(k)$ in a set of higher prioritized neighbors $\hat{\mathcal{V}}^{(j)}(k)$ and lower prioritized neighbors $\check{\mathcal{V}}^{(j)}(k)$. When a vehicle j has received the planned trajectories of all vehicles in $\hat{\mathcal{V}}^{(j)}(k)$, it plans its own trajectory while avoiding collisions with the received trajectories. It

communicates its own trajectory to vehicles in $\check{\mathcal{V}}^{(j)}(k)$. Each vehicle j adds constraint functions $c^{(j,l)}$ to its OCP (9) to ensure collision-free trajectories with vehicles in $\hat{\mathcal{V}}^{(j)}(k)$

$$c^{(j,l)}\left(\boldsymbol{x}^{(j)}_{k+i|k}, \boldsymbol{x}^{(l)}_{k+i|k}\right) \leq 0, \quad \forall i = 1, \ldots, N, \quad \forall l \in \hat{\mathcal{V}}^{(j)}(k). \tag{13}$$

3.1 Reprioritization Framework for Recursive NCS-Feasibility

One of the main challenges for P-DMPC is its incompleteness: even though there exists a priority assignment that results in an NCS-feasible P-DMPC problem, a specific priority assignment might fail to produce a solution. Changing the priority assignment during runtime can prevent such a failure, but loses recursive NCS-feasibility of the P-DMPC problem.

Definition 5 (*NCS-feasible*) A P-DMPC problem is NCS-feasible if every agent in the NCS finds a feasible solution to its OCP.

A P-DMPC problem is recursively NCS-feasible if from NCS-feasibility at time k we can guarantee NCS-feasibility for all future times. Figure 4 illustrates our distributed reprioritization framework to maintain NCS-feasible P-DMPC trajectory planning problems while using a time-variant priority assignment function. At the beginning of every timestep k, each agent attempts to plan its trajectory given the priorities from time k. If any agent fails to find a feasible solution, it notifies all other agents. All agents stay on their recursively agent-feasible trajectory. At any point, if the P-DMPC problem is NCS-feasible, the corresponding input is applied.

A proof for recursive NCS-feasibility of time-invariant priorities is given in [38]. We need to prove recursive NCS-feasibility with time-variant priorities and our distributed reprioritization framework. We assume an initially NCS-feasible problem and bounded disturbances which an underlying controller can compensate.

Theorem 2 *A P-DMPC problem with our distributed reprioritization framework, the OCP (9) with coupling constraints (13), and any time-variant priority assignment function p is recursively NCS-feasible.*

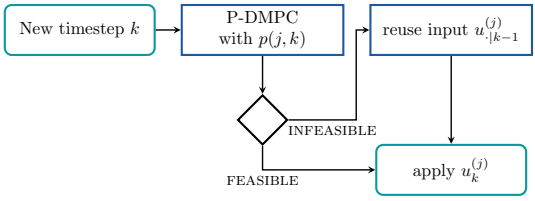

Fig. 4 Distributed reprioritization framework which guarantees recursive NCS-feasibility, as seen from agent j. Figure adapted from [51]

Proof Without loss of generality, assume the computation order resulting from the priority assignment function $p(j,k)$ to be $1,\ldots,N_A$. Assume an NCS-feasible solution $\left(u^{(j)}_{\cdot|k}, \boldsymbol{x}^{(j)}_{\cdot|k}\right), \forall j \in \mathcal{V}$ at timestep k. Because of bounded disturbances which an underlying controller can compensate, we have

$$\boldsymbol{x}^{(j)}(k+1) \approx \boldsymbol{x}^{(j)}_{k+1|k}, \quad \forall j \in \mathcal{V}. \tag{14}$$

Every agent shifts and extends the feasible solution of the previous timestep

$$\begin{aligned} \boldsymbol{x}^{(j)}_{k+1+i|k+1} &= \boldsymbol{x}^{(j)}_{k+1+i|k}, \quad \forall j \in \mathcal{V}, \quad \forall i = 1,\ldots,N-1 \\ \boldsymbol{x}^{(j)}_{k+1+N|k+1} &= \boldsymbol{x}^{(j)}_{k+N|k}, \quad \forall j \in \mathcal{V} \end{aligned} \tag{15}$$

For agent 1, who does not consider other agents, recursive feasibility is given by Theorem 1. For any agent $2 \leq j \leq N_A$, the coupling constraints (13) must also be considered. Substituting (15) in (13) yields

$$c^{(j,l)}\left(\boldsymbol{x}^{(j)}_{k+1+i|k+1}, \boldsymbol{x}^{(l)}_{k+1+i|k+1}\right) = c^{(j,l)}\left(\boldsymbol{x}^{(j)}_{k+1+i|k}, \boldsymbol{x}^{(l)}_{k+1+i|k}\right), \tag{16}$$

$\forall i = 1,\ldots,N-1$ and $\forall l \in \hat{\mathcal{V}}^{(j)}(k)$. Since the agents stand still at the horizon, we have for the last timestep $k+N+1$

$$c^{(j,l)}\left(\boldsymbol{x}^{(j)}_{k+N+1|k+1}, \boldsymbol{x}^{(l)}_{k+N+1|k+1}\right) = c^{(j,l)}\left(\boldsymbol{x}^{(j)}_{k+N|k}, \boldsymbol{x}^{(l)}_{k+N|k}\right) \tag{17}$$

$\forall l \in \hat{\mathcal{V}}^{(j)}(k)$. This establishes recursive NCS-feasibility of the P-DMPC at time k. Because of a time-variant directed coupling graph, the set of higher prioritized agents $\hat{\mathcal{V}}^{(j)}(k+1)$ might differ from $\hat{\mathcal{V}}^{(j)}(k)$. Still, all coupling constraints are fulfilled. Our coupling constraints are symmetric, i.e., $c^{(j,l)} = c^{(l,j)}$. A new coupling constraint is guaranteed to be satisfied, as there was no collision possibility in timestep k. A vanished coupling constraint cannot interfere with feasibility. Since all constraints are satisfied at timestep $k+1$, the P-DMPC problem with time-variant priorities is recursively NCS-feasible with our reprioritization framework. □

3.2 Priority Assignment Algorithms

This section introduces two priority assignment functions. Section 3.2.1 describes a constraint-based heuristic which aims at assigning priorities for NCS-feasibility. Section 3.2.2 presents a priority assignment function based on coloring of the coupling graph which reduces computation time.

3.2.1 Constraint-Based Heuristic

The goal of the priority assignment function presented in this subsection is to reduce the risk of standstill of the NCS due to infeasible OCPs of vehicles. We propose a distributed, time-variant priority assignment algorithm for road vehicles on road networks based on our previous work [51]. Each vehicle j first plans a trajectory without inter-vehicle collision constraints (13), which we call the free trajectory $\boldsymbol{y}_{\text{free}}^{(j)}$. Then, each vehicle j counts the number of collisions N_c with other free trajectories $\boldsymbol{y}_{\text{free}}^{(-j)}$ and possibly already planned, optimal trajectories $\boldsymbol{y}^{(-j)^*}$. Vehicle w with most collisions receives the next priority and plans its trajectory considering already planned, optimal trajectories $\boldsymbol{y}^{(-w)^*}$ by solving OCP (9) with coupling constraints (13). The loop repeats until all vehicles have planned their optimal trajectories. If a vehicle cannot find a feasible solution, all vehicles use the previous input as illustrated in Fig. 4. This algorithm results in a time-variant priority assignment function $p_{\text{fca}} \colon \mathcal{V} \times \mathbb{N} \to \mathbb{N}$. The index "future collision assessment (FCA)" reflects the inspiration of this approach from [41].

3.2.2 Graph Coloring

In P-DMPC, if there is no path between two vehicles in the coupling DAG, they can compute in parallel [4]. We call the number of necessary sequential computations the number of computation levels. This section presents a priority assignment function which minimizes the number of computation levels by vertex coloring based on our previous work [53]. Figure 5 illustrates the proposed problem solution with an example. From an example undirected graph, a baseline approach which assigns priorities equal to the vertex number results in four computation levels. Assigning

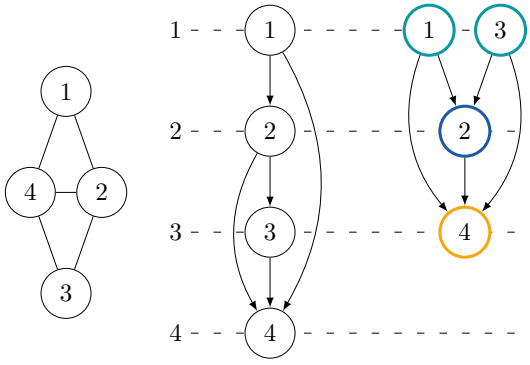

Fig. 5 Example of computation levels from graph coloring compared to baseline. Left: Undirected coupling graph. Middle: Coupling DAG with computation levels from baseline priorities equal to vertex number. Right: Coupling DAG with computation levels from priorities based on graph coloring. Figure adapted from [53]

priorities with our coloring approach reduces the number of computation levels to three, as each color corresponds to a computation level.

In vertex coloring, we map vertices $i \in \mathcal{V}(G)$ to colors $c \in C \subset \mathbb{N}_{>0}$ with the function $\varphi\colon \mathcal{V}(G) \to C$. In order to produce a valid coloring, φ has to satisfy

$$\varphi(i) \neq \varphi(j), \quad \forall i, j \in \mathcal{V}(G), \forall e_{ij} \in \mathcal{E}(G), i \neq j. \tag{18}$$

Our distributed graph coloring algorithm must produce the same coloring φ in every vehicle and must be fast enough for online execution. We propose a combination of saturation degree ordering, largest degree ordering and first-fit to achieve a deterministic coloring as detailed in [53]. We translate our graph coloring function φ to a priority assignment function p. Let $\mathcal{V}_c$ be all vertices of color c

$$\mathcal{V}_c = \{v \mid v \in \mathcal{V}, \varphi(v) = c\}. \tag{19}$$

We can generate a coupling DAG from an undirected coupling graph colored with φ with an injective priority assignment function p that fulfills the requirement

$$p(i) < p(j) \iff c_1 < c_2, \quad \forall i \in \mathcal{V}_{c_1}, \forall j \in \mathcal{V}_{c_2}. \tag{20}$$

4 Numerical and Experimental Results

This section describes the evaluation platform, our Cyber-Physical Mobility Lab (CPM Lab).[2] It presents the evaluation of our RHGS algorithm for recursive agent-feasibility and of our reprioritization framework for recursive NCS-feasibility. Our algorithms are implemented in MATLAB R2023a and openly available online.[3]

4.1 Cyber-Physical Mobility Lab

The evaluation hardware for this work is our 1:18 model-scale CPM Lab [34]. It is a remotely accessible open-source platform consisting of 20 networked and autonomous vehicles (μCars) [54]. Our trajectory planning algorithms run on a PC with an AMD Ryzen 5 5600X 6-core 3.7 GHz CPU and 32 GB of RAM. This PC communicates with the other components in the CPM Lab via the data distribution service standard over WLAN [33]. Figure 6 illustrates the road network in the CPM Lab. It replicates a wide variety of common traffic scenarios with a 16-lane urban intersection, a highway, highway on-ramps, and highway off-ramps.

[2] https://cpm.embedded.rwth-aachen.de.

[3] https://github.com/embedded-software-laboratory/p-dmpc.

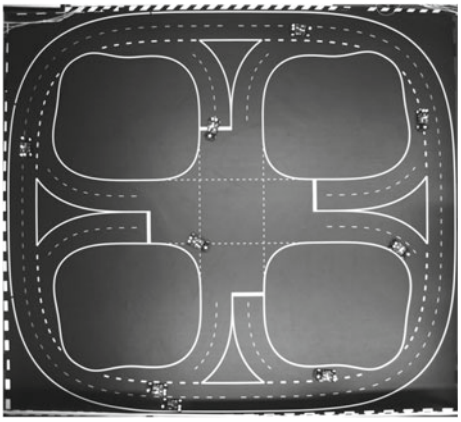

Fig. 6 1:18 model-scale road network in the CPM Lab with an intersection, a highway, highway on-ramps, and highway off-ramps

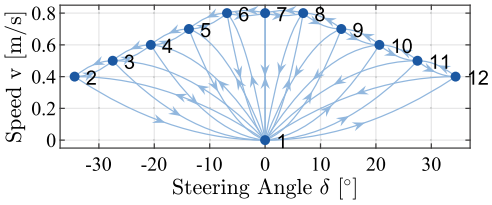

Fig. 7 The MPA for our experiments. The position of a state marks its speed v and its steering angle δ. For clarity of presentation, the figure omits the time dependency of transitions to ensure recursive feasibility

Our algorithm plans trajectories using the MPA shown in Fig. 7. It is based on a kinematic bicycle model (1) of our μCars with $\ell_r = 7.5\,\text{cm}$ and $L = 15\,\text{cm}$. It is designed such that transitions between automata states respect input constraints of the μCars used in the experiments. The transitions change the control inputs linearly over the duration of the sampling time $T_s = 0.2$ s. The planning horizon is $N = 8$.

4.2 Evaluation of Receding Horizon Graph Search

In our RHGS algorithm, we achieve recursive agent-feasibility by design of the MPA, as illustrated in Fig. 3. The recursive agent-feasibility is verified in [55].

In [55], we compare our RHGS planner with a state-of-the-art graph search (SGS) planner. The SGS planner computes the trajectory once at the beginning of the experiment with a horizon spanning the whole experiment duration. The test scenario contains moving obstacles with known future trajectories. Both planners manage to avoid the obstacles. In the specific test scenario, the RHGS planner stops in front of

the obstacles, while the SGS avoids the obstacles by steering early enough. Consequently, the cost function value is lower for the SGS than for the RHGS. However, in the worst case, the computation effort increases exponentially with the horizon length. A video of an experiment using RHGS with multiple vehicles in the CPM Lab is available online.[4]

4.3 Evaluation of Time-Variant Priority Assignment

This section presents P-DMPC trajectory planning with time-variant priority assignment using our reprioritization framework depicted in Fig. 4 to guarantee recursive NCS-feasibility. A time-invariant priority assignment algorithm and a time-variant random priority assignment algorithm represent state-of-the-art priority assignment algorithms for our evaluation. In the time-invariant priority assignment algorithm, each vehicle receives a unique priority corresponding to its unique number $j \in \mathcal{V}$ at the beginning of the experiment. The priority assignment function $p_{\text{const}} \colon \mathcal{V} \times \mathbb{N} \to \mathbb{N}$ is

$$p_{\text{const}}(j, k) = j. \tag{21}$$

In the random priority assignment algorithm, each vehicle receives a random priority in each timestep. The priority assignment function $p_{\text{rand}} \colon \mathcal{V} \times \mathbb{N} \to \mathbb{N}$ is

$$p_{\text{rand}}(j, k) = r(k). \tag{22}$$

The evaluation focuses on two criteria: (i) the ability to maintain progress of the vehicles, i.e., to avoid a standstill, and (ii) the ability to reduce computation time. We call the absence of progress a standstill, which we define as a situation where two or more vehicles stop for the rest of the experiment.

Our evaluation spans 720 numerical experiments with an individual duration of 180 s, a combination of the four priority assignment functions (p_{fca}, p_{color}, p_{rand}, and p_{const}) with vehicle amounts from 1 to 20 in 9 random scenarios. All scenarios are based on the map shown in Fig. 6. The vehicle starting positions and their reference paths in the map are determined randomly to replicate various traffic situations. We use the Mersenne Twister algorithm [44] with a manually set random seed for reproducible experiments.

Figure 8 depicts the performance on a scale of 0 to 1 of the four priority assignments in three aspects. The first aspect is the number of vehicles up to which all vehicles in all scenarios could maintain progress over the experiment duration. The functions p_{const} and p_{fca} are able to move up to 10 and 9 vehicles respectively, whereas p_{rand} and p_{color} produce a standstill with already 6 and 5 vehicles respectively. The second aspect is the percentage of scenarios from all scenarios with all numbers of vehicles, for which the corresponding priority assignment function successfully

[4] https://youtu.be/7LB7I5SOpQE.

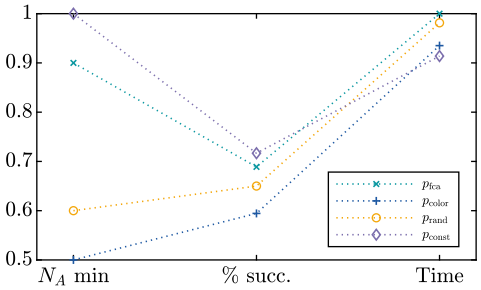

Fig. 8 Performance of priority assignment functions scaled from 0 to 1: N_A min: standstill-free up to number of agents ($\times 10$), % succ.: percentage of standstill-free scenarios ($\times 100$), Time: average time until standstill ($\times 145.1$ s)

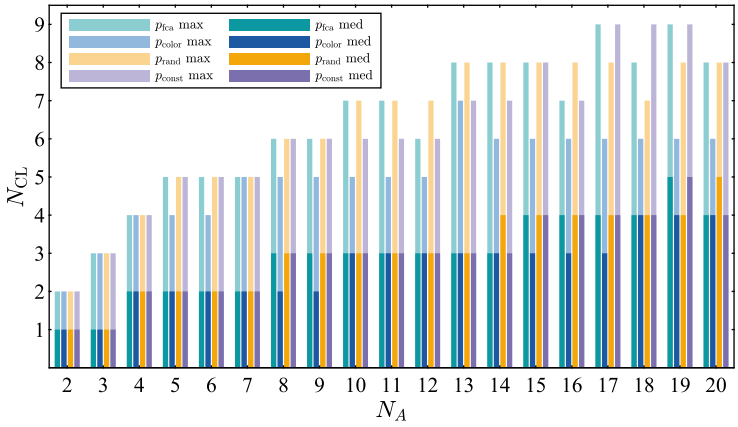

Fig. 9 Median and maximum number of computation levels N_{CL} in all timesteps of all standstill-free scenarios per priority assignment function over the number of vehicles N_A

maintained progress over the full experiment duration. The performance tendency is similar to the first aspect. Both aspects indicate that a change in the priority assignment can decrease NCS-feasibility. A constant priority might not be ideal in all situations, but can help maintaining NCS-feasibility and avoid standstills. The third aspect is the average time until standstill, in which p_{fca} performs best with an average time of 145.1 s. These results indicate that changing priorities might harm the systems performance. A better approach might be to change priorities only when the P-DMPC problem becomes NCS-infeasible.

The computation time in P-DMPC is mainly determined by number of computation levels, i.e., the minimum number of sequential computations of the NCS [4]. Figure 9 shows the median and maximum number of computation levels per priority assignment function in experiments without standstills. A scenario will develop differently for different priority assignment functions. To mitigate the effect of this difference, we consider each timestep from all experiments on its own. In every

timestep, we assign priorities with all four priority assignment functions and analyze the resulting number of computation levels. The strength of the priority assignment function p_{color} lies in this criterion, as it produces the lowest amount of median and maximum computation levels for all experiments. In the scenarios with 17 to 19 vehicles, it reduces the number of computation levels by up to 33 %.

A video of an experiment in the CPM Lab is available online.[5] It presents the priority assignment function p_{fca} with our distributed reprioritization framework.

5 Conclusion

This chapter presented two approaches to deal with the complexity of a nonconvex trajectory planning problem: discretization of control inputs using motion primitives and distribution of the control problem using prioritization. We showed recursive agent-feasibility for our receding horizon graph search using motion primitives, making it a viable alternative to receding horizon approaches using optimization. The efficiency of the informed search algorithm is highly dependent on the quality of the cost-estimating heuristic. We showed recursive NCS-feasibility for time-variant priority assignment functions in prioritized planning. We presented and evaluated two priority assignment functions for road vehicles, one for maintaining progress of vehicles and one for reduced computation time. Changing the priorities during an experiment affects NCS-feasibility of the P-DMPC problem, as it alters the constraints of the vehicles' OCPs. Experiments with up to 17 vehicles in our CPM Lab showed efficient computation and effective results for networked trajectory planning problems.

The priority assignment function offers potential for improvement. A strategy that might be worth examining is the application of game theory to assign priorities [30]. Our framework for distributed reprioritization achieves recursive NCS-feasibility through standstill at the end of the prediction horizon. While ensuring safety, this counteracts the goal to maintain progress in traffic. Some of the scenarios we evaluated resulted in a standstill which could not be resolved through the priority assignment function. In these situations, the priority assignment function could be altered to explore different priority permutations. The trajectory planner could also switch to a cooperative or centralized trajectory planning algorithm, which is more flexible, but has higher computation time [29]. The minimum number of computation levels and thus the expected computation time in our P-DMPC is decided by the coupling graph. If the allowed computation time is fixed and the number vehicles increases, less computation time for each vehicle is available. This issue will be addressed in our future work. Another topic to explore is the cooperation of our distributed trajectory planning algorithm with others such as [14], and the cooperation with human-driven vehicles [50].

[5] https://youtu.be/RqwbHUwip10.

Acknowledgements We thank Julius Kahle and Georg Dorndorf for their contributions. This research is supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Program SPP 1835 “Cooperative Interacting Automobiles” (grant number: KO 1430/17-1).

References

1. Ajanovic, Z., Lacevic, B., Shyrokau, B., Stolz, M., Horn, M.: Search-based optimal motion planning for automated driving. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 4523–4530 (2018). https://doi.org/10.1109/IROS.2018.8593813
2. Alonso-Mora, J., Beardsley, P., Siegwart, R.: Cooperative collision avoidance for nonholonomic robots. IEEE Trans. Rob. **34**(2), 404–420 (2018). https://doi.org/10.1109/TRO.2018.2793890
3. Alrifaee, B.: Networked Model Predictive Control for Vehicle Collision Avoidance. Ph.D. Thesis, RWTH Aachen University (2017)
4. Alrifaee, B., Heßeler, F.J., Abel, D.: Coordinated non-cooperative distributed model predictive control for decoupled systems using graphs. IFAC-PapersOnLine **49**(22), 216–221 (2016). https://doi.org/10.1016/j.ifacol.2016.10.399
5. Alrifaee, B., Maczijewski, J.: Real-time trajectory optimization for autonomous vehicle racing using sequential linearization. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 476–483 (2018). https://doi.org/10.1109/IVS.2018.8500634
6. Alrifaee, B., Maczijewski, J., Abel, D.: Sequential convex programming MPC for dynamic vehicle collision avoidance. In: 2017 IEEE Conference on Control Technology and Applications (CCTA), pp. 2202–2207 (2017). https://doi.org/10.1109/CCTA.2017.8062778
7. Alrifaee, B., Mamaghani, M.G., Abel, D.: Centralized non-convex model predictive control for cooperative collision avoidance of networked vehicles. In: 2014 IEEE International Symposium on Intelligent Control (ISIC), pp. 1583–1588. IEEE, Juan Les Pins, France (2014). https://doi.org/10.1109/ISIC.2014.6967623
8. Altché, F., Qian, X., de La Fortelle, A.: Time-optimal coordination of mobile robots along specified paths. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5020–5026 (2016). https://doi.org/10.1109/IROS.2016.7759737
9. Andersson, O., Ljungqvist, O., Tiger, M., Axehill, D., Heintz, F.: Receding-horizon lattice-based motion planning with dynamic obstacle avoidance. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 4467–4474. IEEE, Miami Beach, FL (2018). https://doi.org/10.1109/CDC.2018.8618964
10. Bennewitz, M., Burgard, W., Thrun, S.: Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. Robot. Auton. Syst. **41**(2), 89–99 (2002). https://doi.org/10.1016/S0921-8890(02)00256-7
11. Borrelli, F., Bemporad, A., Morari, M.: Predictive Control for Linear and Hybrid Systems. Cambridge University Press (2017)
12. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge, UK; New York (2004)
13. Buckley, S.: Fast motion planning for multiple moving robots. In: 1989 International Conference on Robotics and Automation Proceedings, pp. 322–326 vol.1 (1989). https://doi.org/10.1109/ROBOT.1989.100008
14. Burger, C., Fischer, J., Bieder, F., Taş, Ö.Ş., Stiller, C.: Interaction-aware game-theoretic motion planning for automated vehicles using bi-level optimization. In: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), pp. 3978–3985 (2022). https://doi.org/10.1109/ITSC55140.2022.9922600

15. Chalaki, B., Malikopoulos, A.A.: A priority-aware replanning and resequencing framework for coordination of connected and automated vehicles. IEEE Control Syst. Lett. **6**, 1772–1777 (2022). https://doi.org/10.1109/LCSYS.2021.3133416
16. Clark, C., Bretl, T., Rock, S.: Applying kinodynamic randomized motion planning with a dynamic priority system to multi-robot space systems. In: Proceedings, IEEE Aerospace Conference, vol. 7, pp. 7–7 (2002). https://doi.org/10.1109/AERO.2002.1035338
17. Cohen, B., Phillips, M., Likhachev, M.: Planning single-arm manipulations with n-arm robots. In: Robotics: Science and Systems X, pp. 226–. Robotics: Science and Systems Foundation (2014). https://doi.org/10.15607/RSS.2014.X.033
18. Dellin, C.M., Srinivasa, S.S.: A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In: Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS), p. 9 (2016)
19. Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J.: Practical search techniques in path planning for autonomous driving. In: AAAI Conference on Artificial Intelligence, p. 6 (2008)
20. Eilbrecht, J., Stursberg, O.: Optimization-based maneuver automata for cooperative trajectory planning of autonomous vehicles. In: 2018 European Control Conference (ECC), pp. 82–88 (2018). https://doi.org/10.23919/ECC.2018.8550422
21. Erdmann, M., Lozano-Pérez, T.: On multiple moving objects. Algorithmica **2**(1), 477 (1987). https://doi.org/10.1007/BF01840371
22. Frazzoli, E.: Maneuver-based motion planning and coordination for multiple UAVs. In: Proceedings. The 21st Digital Avionics Systems Conference, vol. 2, pp. 8D3–8D3 (2002). https://doi.org/10.1109/DASC.2002.1052947
23. Frazzoli, E., Dahleh, M., Feron, E.: Maneuver-based motion planning for nonlinear systems with symmetries. IEEE Trans. Rob. **21**(6), 1077–1091 (2005). https://doi.org/10.1109/TRO.2005.852260
24. Frese, C.: Cooperative motion planning using branch and bound methods. In: Proceedings of the 2009 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory, p. 15. KIT Scientific Publishing (2010)
25. Frese, C., Beyerer, J.: Planning cooperative motions of cognitive automobiles using tree search algorithms. In: Dillmann, R., Beyerer, J., Hanebeck, U.D., Schultz, T. (eds.) KI 2010: Advances in Artificial Intelligence, vol. 6359, pp. 91–98. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16111-7_10
26. Gray, A., Gao, Y., Lin, T., Hedrick, J.K., Tseng, H.E., Borrelli, F.: Predictive control for agile semi-autonomous ground vehicles using motion primitives. In: 2012 American Control Conference (ACC), pp. 4239–4244 (2012). https://doi.org/10.1109/ACC.2012.6315303
27. Ioan, D., Prodan, I., Olaru, S., Stoican, F., Niculescu, S.I.: Mixed-integer programming in motion planning. Annu. Rev. Control. **51**, 65–87 (2021). https://doi.org/10.1016/j.arcontrol.2020.10.008
28. Kapania, N.R., Subosits, J., Christian Gerdes, J.: A sequential two-step algorithm for fast generation of vehicle racing trajectories. J. Dyn. Syst. Meas. Control **138**(9) (2016). https://doi.org/10.1115/1.4033311
29. Kloock, M., Alrifaee, B.: Coordinated Cooperative Distributed Decision-Making using Synchronization of Local Plans. IEEE Transaction on Intelligent Vehicles **8**(2), 1292–1302 (2023). https://doi.org/10.1109/TIV.2023.3234189
30. Kloock, M., Dirksen, M., Kowalewski, S., Alrifaee, B.: Generation of coupling topologies for multi-agent systems using non-cooperative games. In: 2022 IEEE Intelligent Vehicles Symposium (IV), pp. 1–8 (2022). https://doi.org/10.1109/IV51971.2022.9827431
31. Kloock, M., Kragl, L., Maczijewski, J., Alrifaee, B., Kowalewski, S.: Distributed model predictive pose control of multiple nonholonomic vehicles. In: 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 1620–1625 (2019). https://doi.org/10.1109/IVS.2019.8813980
32. Kloock, M., Scheffe, P., Botz, L., Maczijewski, J., Alrifaee, B., Kowalewski, S.: Networked model predictive vehicle race control. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 1552–1557 (2019). https://doi.org/10.1109/ITSC.2019.8917222

33. Kloock, M., Scheffe, P., Greß, O., Alrifaee, B.: An Architecture for Experiments in Networked Control Systems. IEEE Open Journal of Intelligent Transportation Systems **4**, 175–186 (2023). https://doi.org/10.1109/OJITS.2023.3250951
34. Kloock, M., Scheffe, P., Maczijewski, J., Kampmann, A., Mokhtarian, A., Kowalewski, S., Alrifaee, B.: Cyber-physical mobility lab: An open-source platform for networked and autonomous vehicles. In: 2021 European Control Conference (ECC), pp. 1937–1944 (2021). https://doi.org/10.23919/ECC54610.2021.9654986
35. Kloock, M., Scheffe, P., Marquardt, S., Maczijewski, J., Alrifaee, B., Kowalewski, S.: Distributed model predictive intersection control of multiple vehicles. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 1735–1740 (2019). https://doi.org/10.1109/ITSC.2019.8917117
36. Korf, R.E.: Real-time heuristic search. Artif. Intell. **42**(2–3), 189–211 (1990). https://doi.org/10.1016/0004-3702(90)90054-4
37. Kurzer, K., Zhou, C., Marius Zöllner, J.: Decentralized cooperative planning for automated vehicles with hierarchical monte carlo tree search. In: 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 529–536 (2018). https://doi.org/10.1109/IVS.2018.8500712
38. Kuwata, Y., Richards, A., Schouwenaars, T., How, J.P.: Distributed robust receding horizon control for multivehicle guidance. IEEE Trans. Control Syst. Technol. **15**(4), 627–641 (2007). https://doi.org/10.1109/TCST.2007.899152
39. LaValle, S.M.: Planning Algorithms, 1stedn. Cambridge University Press (2006). https://doi.org/10.1017/CBO9780511546877
40. Li, J., Tinka, A., Kiesel, S., Durham, J.W., Kumar, T.K.S., Koenig, S.: Lifelong multi-agent path finding in large-scale warehouses. Proceed. AAAI Conf. Artif. Intell. **35**(13), 11272–11281 (2021)
41. Luo, W., Chakraborty, N., Sycara, K.: Distributed dynamic priority assignment and motion planning for multiple mobile robots with kinodynamic constraints. In: 2016 American Control Conference (ACC), pp. 148–154. IEEE, Boston, MA, USA (2016). https://doi.org/10.1109/ACC.2016.7524907
42. Malikopoulos, A.A.: Connected and Integrated Transportation Systems (2022). https://doi.org/10.48550/arXiv.2211.08600. ArXiv, Preprint
43. Mandalika, A., Salzman, O., Srinivasa, S.: Lazy Receding horizon A* for efficient path planning in graphs with expensive-to-evaluate edges. In: International Conference on Automated Planning and Scheduling, p. 9 (2018)
44. Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998). https://doi.org/10.1145/272991.272995
45. Mettler, B., Kong, Z.: Receding horizon trajectory optimization with a finite-state value function approximation. In: American Control Conference, p. 7. Seattle, WA, USA (2008)
46. Paden, B., Čáp, M., Yong, S.Z., Yershov, D., Frazzoli, E.: A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Trans. Intell. Veh. **1**(1), 33–55 (2016). https://doi.org/10.1109/TIV.2016.2578706
47. Rajamani, R.: Vehicle Dynamics and Control. Mechanical Engineering Series. Springer Science, New York (2006)
48. Regele, R., Levi, P.: Cooperative multi-robot path planning by heuristic priority adjustment. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5954–5959 (2006). https://doi.org/10.1109/IROS.2006.282480
49. Richards, N., Sharma, M., Ward, D.: A hybrid A*/automaton approach to on-line path planning with obstacle avoidance. In: AIAA 1st Intelligent Systems Technical Conference, Infotech@Aerospace Conferences. American Institute of Aeronautics and Astronautics (2004). https://doi.org/10.2514/6.2004-6229
50. Scheffe, P., Alrifaee, B.: A scaled experiment platform to study interactions between humans and CAVs. In: 2023 IEEE Intelligent Vehicles Symposium (IV), pp. 1–6. IEEE, Anchorage, AK, USA (2023). https://doi.org/10.1109/IV55152.2023.10186623

51. Scheffe, P., Dorndorf, G., Alrifaee, B.: Increasing feasibility with dynamic priority assignment in distributed trajectory planning for road vehicles. In: 2022 IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 3873–3879 (2022). https://doi.org/10.1109/ITSC55140.2022.9922028
52. Scheffe, P., Henneken, T.M., Kloock, M., Alrifaee, B.: Sequential convex programming methods for real-time optimal trajectory planning in autonomous vehicle racing. IEEE Trans. Intell. Veh. 1–1 (2022). https://doi.org/10.1109/TIV.2022.3168130
53. Scheffe, P., Kahle, J., Alrifaee, B.: Reducing Computation Time with Priority Assignment in Distributed Control (2022). https://doi.org/10.36227/techrxiv.20304015.v1. Preprint
54. Scheffe, P., Maczijewski, J., Kloock, M., Kampmann, A., Derks, A., Kowalewski, S., Alrifaee, B.: Networked and autonomous model-scale vehicles for experiments in research and education. IFAC-PapersOnLine **53**(2), 17332–17337 (2020). https://doi.org/10.1016/j.ifacol.2020.12.1821
55. Scheffe, P., Pedrosa, M.V.A., Flaßkamp, K., Alrifaee, B.: Receding horizon control using graph search for multi-agent trajectory planning. IEEE Trans. Control Syst. Technol. 1–14 (2022). https://doi.org/10.1109/TCST.2022.3214718
56. Schwarting, W., Alonso-Mora, J., Rus, D.: Planning and decision-making for autonomous vehicles. Ann. Rev. Control Robot. Autonom. Syst. **1**(1), 187–210 (2018). https://doi.org/10.1146/annurev-control-060117-105157
57. Silver, D.: Cooperative pathfinding. Proceed. AAAI Conf. Artif. Intell. Inter. Digit. Entertain. **1**(1), 117–122 (2005). https://doi.org/10.1609/aiide.v1i1.18726
58. Tran, D.Q., Diehl, M.: An application of sequential convex programming to time optimal trajectory planning for a car motion. In: Proceedings of the 48h IEEE Conference on Decision and Control (CDC) Held Jointly with 2009 28th Chinese Control Conference, pp. 4366–4371 (2009). https://doi.org/10.1109/CDC.2009.5399823
59. van den Berg, J., Overmars, M.: Prioritized motion planning for multiple robots. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 430–435 (2005). https://doi.org/10.1109/IROS.2005.1545306
60. Wu, W., Bhattacharya, S., Prorok, A.: Multi-robot path deconfliction through prioritization by path prospects. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 9809–9815. IEEE, Paris, France (2020). https://doi.org/10.1109/ICRA40945.2020.9196813
61. Zhang, S., Li, J., Huang, T., Koenig, S.: Learning a priority ordering for prioritized planning in multi-agent path finding. Proceed. Symp. Combinat. Sear. **15**(1), 208–216 (2022)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

