Synchronization of Local Plans for Cooperative Distributed Decision-Making

Synchronization of Local Plans for Cooperative Distributed Decision-Making

Maximilian Kloock and Bassam Alrifaee

***Abstract*— In cooperative decision-making, agents locally plan for a subset of all agents. Due to only local system knowledge of the agents, these local plans may be inconsistent to local plans of other agents. This inconsistency leads to infeasibility of the plans. This article introduces an algorithm for synchronizing local plans for cooperative distributed decision-making of multi-agent systems. The algorithm consists of two iterative steps: planning and synchronization. In the local planning step, the agents compute local decisions, referred to as plans. Subsequently, consistency of the local plans across agents is achieved using synchronization. The synchronized plans act as reference decisions to the local planning step in the next iteration. In each iteration, the local planning guarantees locally feasible plans, while the synchronization guarantees globally consistent plans in that iteration. The algorithm converges to globally feasible decisions if the coupling topology is feasible. We introduce requirements for the coupling topology to achieve convergence to globally feasible decisions and present the algorithm using a model predictive control example. Our evaluations with car-like robots show that feasible decisions are achieved.**

I. INTRODUCTION

A Multi-Agent System (MAS) consists of multiple agents that should fulfill a common task. As no agent has global information about the overall system, agents share information over a communication network. The use cases of MAS are applications where a central agent would require too many resources for the system requirements, e.g., too high computation time for the timing requirements of the system. MAS also provides an advantage where information sources are naturally distributed [1]. MAS that should fulfill a control task is called Networked Control System (NCS). To this end, each agent implements a controller that makes decisions to minimize the deviation from its reference decision [2], [3]. When the controllers must satisfy constraints, Model Predictive Control (MPC) is often used. For NCS, the communication topology, which is represented as a graph, is important for the control quality and computation time. We classify communication topologies into

a) undirected or directed
b) time-invariant or time-variant

topologies [4].

According to the definitions of [5], NCS make their decisions in a centralized, decentralized, or distributed manner. In centralized decision-making, one agent makes the decisions for all agents on the basis of global system knowledge. Therefore, all agents communicate the required information to the central agent. After decision-making, the central agent communicates the decisions to the single agents' controllers, which apply the decisions. This approach may impose a high computational burden on the central agent. Therefore, the decision-making can be decentralized or distributed to dispense the resource usage in the NCS. In decentralized decision-making, no communication is required. All agents make and implement their own local decisions. Agents need to estimate the decisions of other agents using their local sensors. Distributed decision-making makes use of communication between the local agents. Explicitly shared information improves the control quality at the cost of communication. The actual gain of control quality at communication costs depends on the distribution method. Along these lines, we classify distributed decision-making in

a) cooperative or non-cooperative
b) sequential or parallel
c) iterative or non-iterative

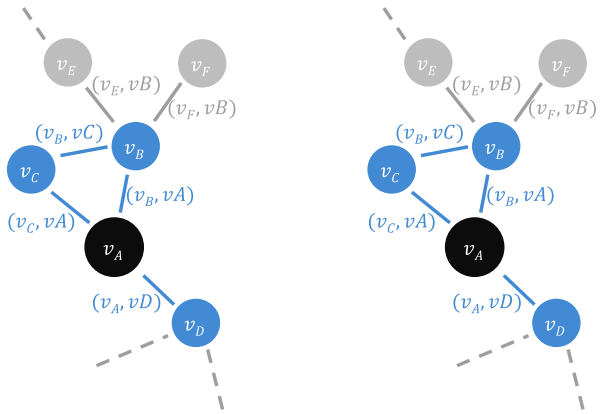
decision-making [4]. According to [6], agents are cooperative if they fulfill two requirements. The first requirement is that the agents pursue their own goals and conflict with other agents. The second requirement is that each agent tries to manage the conflicts. In cooperative decision-making, agents take the current states and objectives of other agents into account and make locally optimal decisions for a subsystem consisting of multiple agents.

A difficulty of distributed decision-making compared to centralized decision-making is that there is no agent with global system knowledge. All agents make their decisions based on local system knowledge that lead to inconsistencies of their local decisions. We use the term *local plans* to refer to local decisions. Fig. 1 illustrates agents' local system knowledge with local system view of Agent v_A in Fig. 1(a) and local system view of Agent v_B in Fig. 1(b). Agent v_A is coupled with agents v_B, v_C, and v_D, while agent v_B is coupled with agents v_A, v_C, v_E, and v_F. The local decisions of agents v_A and v_B are based on objectives of different sets of agents and; hence, the plans of agents v_A and v_B for each other are inconsistent [7].

In distributed decision-making, agents have to communicate to gain consistent plans. We call the property of consistent information about plans *prediction consistency*. We use synchronization inspired by consensus, e.g., in [5] to synchronize local plans of agents. Using the synchronized plans as local reference decisions in the next iteration, the algorithm converges to globally feasible decisions if the coupling topology is feasible. Subsection I-A presents related

This article was submitted on August 26, 2021. This research is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) within the Priority Program SPP 1835 "Cooperative Interacting Automobiles".

All authors are with the Chair for Embedded Software of RWTH Aachen University, Aachen, 52074 Aachen (e-mail: {kloock, alrifaee}@embedded.rwth-aachen.de).



(a) Local system view of agent v_A.

(b) Local system view of agent v_B.

Fig. 1. On each side, the black colored agent considers the blue colored agents in its decision-making and ignores the gray colored agents. This leads to inconsistent plans, since the agents include different sets of agents in their decision-making.

work that addresses the prediction consistency problem in cooperative distributed decision-making. Then, Subsection I-B states the contribution of this article and Subsection I-C introduces the organization of this article.

A. Related Work

Various research has addressed prediction consistency in cooperative distributed decision-making. Examples for sequential approaches are the works in [8] and [9]. The agents make local decisions while each agent considers the goals of coupled agents in its decision-making. The agents make their decisions in a sequential order to achieve prediction consistency. Decisions of the prior agents are known to following agents. Another sequential approach is presented in [10]. The first agent makes multiple options for future decisions and communicates them to the second agent. The second agent makes its own decisions for each of the options. Subsequently, the second agent selects the decisions that are best for both agents. The agents swap their roles in each time step. The authors state that this approach can be extended to more than two agents.

Other approaches make decisions in parallel. The work in [11] addresses the prediction consistency by coupled agents with all agents that have a path to each other in the communication topology. Prediction consistency is achieved by considering the complete graph of active couplings, which results in the central problem, or by considering complete sub-graphs for disjoint coupling graphs. Another idea to address the prediction consistency is to adapt the coupling weights in the coupling graph to model the uncertainty of neighbors depending on the influence of their neighbors' decisions. The work in [4] proposes to choose the weighting factor depending on the node degree in the coupling graph, but did not evaluate this idea. In [12], agents have initial decisions and make proposals to adapt the decisions in favor of their local costs. These changes are acknowledged by other agents. However, this method depends on initial decisions. Other approaches make use of game theory, e.g., of min-max games [13] and coalitional games [14], [15].

Several authors apply cooperative decision-making and propose practice-driven approaches. In [16] external soft constraints extend the optimization problem for networked and autonomous vehicles. These additional constraints increase the distance between the vehicles in order to avoid collisions even if prediction consistency is not achieved. Another example for networked and autonomous vehicles is the work in [17]. The vehicles drive on fixed paths in an unsignalized intersection scenario. In this approach, the vehicles close to the intersection form a complete coupling graph and use consensus to synchronize speed profiles for collision avoidance. Consensus is also used in [18] to synchronize the direction of movement for unmanned aerial vehicles. The authors of [19] use cooperative decision-making in power networks. They use an iterative and sequential approach to converge to a global solution. In [20], the schedule of the power network is optimized using the distributed optimization method of [21].

To the best of our knowledge, there is no work that addresses the prediction consistency problem in cooperative decision-making for agents that make their decisions in parallel. Recent methods depend on coupling topologies with a high number of links or initial decisions. Other methods are application-specific and depend on a specific scenario.

B. Contribution of this Article

This article introduces an algorithm for cooperative distributed decision-making to achieve prediction consistent decisions over sparse coupling topologies, without depending on pre-computations, e.g., of initial decisions. Agents make their decisions in parallel over directed and time-varying network topologies. The agents' decision-making works time-triggered. The algorithm synchronizes local plans of coupled agents and iteratively converges to consistent decisions. Furthermore, we present theoretical analyses and numerical simulations of the algorithm. We present our algorithm using a Distributed Model Predictive Control (DMPC) example and apply it to car-like robots.

C. Organization of this Article

This article is structured as follows. We introduce the problem formulation in Section II. Section III presents our cooperative decision-making algorithm using a DMPC approach. The algorithm is evaluated in Section IV in trajectory planning scenarios for networked car-like robots. Finally, Section V concludes this article.

II. PROBLEM FORMULATION

The objective function for agent v_i in cooperative DMPC at time t is

$$J^{(i)^\star} = \min_{\Delta\hat{U}^{(i)}(\cdot)} \sum_{\substack{j \\ v_j \in \mathcal{V}^{(i)} \cup \{v_i\}}} \alpha^{(i,j)} \Bigg(\sum_{k=1}^{H_p-1} l^{(j)}\big(x_i^{(j)}(t+k),\, r^{(j)}(t+k)\big) + l_{H_p}^{(j)}\big(x_i^{(j)}(t+H_p),\, r^{(j)}(t+H_p)\big) + \sum_{k=0}^{H_u-1} l_u^{(j)}\big(\Delta u_i^{(j)}(t+k)\big) \Bigg) + \sum_{\substack{j \\ v_j \in \mathcal{V}^{(i)}}} \sum_{k=1}^{H_p} \Bigg(c_o^{(i,j)}\big(x_i^{(i)}(t+k), x_i^{(j)}(t+k)\big) + \sum_{\substack{q \\ v_q \in \mathcal{V}^{(i)} \\ q>j}} c_o^{(j,q)}\big(x_i^{(j)}(t+k), x_i^{(q)}(t+k)\big) \Bigg),\ \forall i : v_i \in \mathcal{V}, \tag{1}$$

where $\mathcal{V} = v_1, \ldots, v_N$ is the set of agents, N is the number of agents, H_p denotes the prediction horizon, $x_i^{(j)}(t+k)$ and $\Delta u_i^{(j)}(t+k),\ \forall k \in [1, H_p]$ denote the decision made by agent v_i for agent v_j and the resulting system input variations, respectively, and $\Delta\hat{U}^{(i)}(\cdot) = \Big((\Delta u^{(i)}(\cdot)\ \Delta u^{(i_1)}(\cdot)\ \ldots\ \Delta u^{(i_{N^{(i)}})}(\cdot)\Big)^T$. We use $\square(\cdot)$ to refer to the full prediction horizon H_p. $x_i^{(-)}(\cdot)$ denotes the decisions of all agents for agent v_i and $x_-^{(i)}(\cdot)$ denotes all decisions made by agent v_i. $r^{(i)}(\cdot)$ denotes the reference decision of agent v_i. $l(\cdot)$, $l_u(\cdot)$, and $l_{H_p}(\cdot)$ denote costs for reference deviation, costs for system input variations, and costs for the last step at the end of the prediction horizon H_p, respectively. $\mathcal{V}^{(i)} = \{v_{i_1}, \ldots, v_{N^{(i)}}\}$ is the set of neighbors of agent v_i. Neighboring agents are coupled with coupling weight $\alpha^{(i,j)}$, with $\alpha^{(i,i)} = 1, \forall i : v_i \in \mathcal{V}$. $c_o^{(i,j)}(\cdot)$ denotes coupling objective between agents v_i and v_j.

In order to avoid conflicts, the DMPC algorithm has to satisfy the prediction consistency property as stated in Definition 1 for each time step t.

Definition 1 (Prediction consistency): Prediction consistency is the property that the predicted decision $x_j^{(i)}(\cdot)$ of agent v_i for agent v_j and the predicted decision $x_j^{(j)}(\cdot)$ of agent v_j for itself coincide, i.e., $x_j^{(i)}(\cdot) = x_j^{(j)}(\cdot),\ \forall i, j : v_i, v_j \in \mathcal{V}$.

If the prediction consistency is not satisfied, conflicts between decisions are not avoided, i.e., if agent v_i guarantees conflict freeness between decisions $x_j^{(i)}(\cdot)$ and $x_i^{(i)}(\cdot)$ and implements decision $x_i^{(i)}(\cdot)$, but agent v_j implements decision $x_j^{(j)}(\cdot) \neq x_j^{(i)}(\cdot)$. In this case, the planned decisions are locally feasible in each agent, but not globally feasible within the NCS. We introduce synchronization in order to achieve prediction consistency.

In order to achieve global feasibility of local plans, the plans have to be locally feasible, prediction consistent, and be planned using a topology with a diameter of at most 2, as stated in Theorem 1.

Theorem 1: Agents that implement a cooperative distributed decision-making algorithm achieve globally feasible decisions if and only if

- C1 all decisions $x_-^{(i)}(\cdot),\ \forall i : v_i \in V$ are locally feasible, and
- C2 the decisions are prediction consistent $x_i^{(i)}(\cdot) = x_j^{(i)}(\cdot),\ \forall i, j : v_i, v_j \in \mathcal{V}$, and one of the following two statements is satisfied:
- C3 agents v_i, v_j that may conflict with each other are coupled by $v_i \in \mathcal{V}^{(j)}$ or $v_j \in \mathcal{V}^{(i)}$, or
- C4 agents v_i, v_j that may conflict with each other have a common neighbor, i.e., $\exists v_h : v_i, v_j \in \mathcal{V}^{(h)}$.

Proof: See Appendix V-A. ∎

We now introduce our algorithm for synchronized DMPC in Section III.

III. SYNCHRONIZED COOPERATIVE DISTRIBUTED MODEL PREDICTIVE CONTROL

Consider a MAS consisting of N dynamically decoupled agents $\mathcal{V} = \{v_1\ \ldots\ v_N\}$. A weighted and directed coupling topology graph $G = (\mathcal{V}, E)$ models coupling links between agents, where E is the set of coupling links. A denotes the weighted adjacency matrix of G and $\alpha^{(i,j)}$ is the element of the ith row and jth column of A. The set of neighbors for an agent v_i is defined as $\mathcal{V}^{(i)} = \{v_j \,|\, (v_j, v_i) \in E\}$ with cardinality $N^{(i)} = |\mathcal{V}^{(i)}|$. Hence, $\mathcal{V}^{(i)}$ is the set of all nodes with incoming edges to node v_i in the topology. Let $u^{(i)}(\cdot)$, $\Delta u^{(i)}(\cdot)$, and $x^{(i)}(\cdot)$ denote the inputs, input variations, and decision of agent v_i, respectively and $r^{(i)}(\cdot)$ represents its reference decision. $x_j^{(i)}(\cdot)$ denotes the decision of agent v_i for agent v_j. The vector $\hat{u}^{(i)}(\cdot) = \big(\hat{u}_i^{(i)}(\cdot)\ \hat{u}_{j_1}^{(i)}(\cdot) \ldots \hat{u}_{j_{N^{(i)}}}^{(i)}(\cdot)\big)^T$ contains the system inputs predicted by v_i for itself and all of its neighbors. Analogously, $\Delta\hat{u}^{(i)}(\cdot)$ and $\hat{x}^{(i)}(\cdot)$ hold the corresponding predictions for input variations and plans, respectively. $\hat{x}_j^{(i)}(\cdot)$ denotes the local plan of agent v_i for agent v_j. $\hat{x}_-^{(i)}(\cdot)$ denotes all plans made by agent v_i, while $\hat{x}_i^{(-)}(\cdot)$ denotes all plans made by different agents for agent v_i. $\hat{x}_-^{(-)}(\cdot)$ denotes all plans of the corresponding time step.

The iterative Synchronized Cooperative Distributed Model Predictive Control (SC-DMPC) algorithm works as follows. Algorithm 1 shows this method as pseudo code. First, in line 4, each agent v_i makes plans $\hat{x}_-^{(i)}(\cdot) = \hat{x}_j^{(i)}(\cdot), \forall j : v_j \in \mathcal{V}^{(i)} \cup \{v_i\}$ in parallel. Second, all coupled agents share their plans $\hat{x}_-^{(i)}(\cdot)$ in line 5. Third, if the plans differ (line 6), the agents synchronize them (line 7) to achieve synchronized plans, denoted as $\overline{x}_-^{(i)}$. If the synchronized plans are not conflict-free (line 3), the agents plan again using the synchronized plans as local reference decisions. The next two subsections explain the planning and synchronization in more detail.

A. Optimization of Plans

At every time step, each agent v_i solves a local optimization problem. The optimization problem is composed

Algorithm 1 SC-DMPC for agent v_i.

Input: $r^{(j)}(\cdot), J^{(j)}, \mathcal{V}^{(i)}, \quad \forall j : v_j \in V^{(i)} \cup \{v_i\}$
Output: $x_j^{(i)}(\cdot), \quad \forall j : v_j \in V^{(i)} \cup \{v_i\}$

while $\neg$ isConflictFree($r^{(-)}(\cdot)$) **do**
  $\hat{x}_-^{(i)}(\cdot) \leftarrow$ optimize($J^{(i)}$)$|_{r^{(-)}(\cdot)}$
  communicate $\hat{x}_j^{(i)}(\cdot)$ to $v_j, \ \forall j : v_j \in V^{(i)}$
  if $\neg$ isPredictionConsistent($\hat{x}_-^{(i)}(\cdot)$) **then**
    $\overline{x}_-^{(i)}(\cdot) \leftarrow$ synchronize($\hat{x}_-^{(-)}(\cdot), \mathcal{V}^{(i)}$)
    $r^{(-)}(\cdot) = \overline{x}_-^{(i)}(\cdot)$
  else
    $r^{(-)}(\cdot) = \hat{x}_-^{(i)}(\cdot)$
  end if
end while
$x_-^{(i)}(\cdot) = r^{(-)}(\cdot)$

of the individual cost functions and constraints of v_i and all its neighbors $v_j \in \mathcal{V}^{(i)}$. Furthermore, the optimization considers coupling objectives and constraints between v_i and each neighbor as well as between each pair of neighbors $v_j, v_k \in \mathcal{V}^{(i)}$ with $j \neq k$. The main objective for each agent is to stay on its reference decision while keeping input variations to a minimum. This property is defined as local cost function over the finite prediction horizon H_p in Eq. 1.

The complete optimization problem for agent v_i is formulated as follows:

$$\hat{u}^{(i)^\star}(\cdot) = \min_{\Delta\tilde{U}^{(i)}(\cdot)} J^{(i)}, \tag{2}$$

Subject to ($\forall v_j, v_q \in \mathcal{V}^{(i)}$):

$$x^{(i)}(t+1+k) = f^{(i)}(x^{(i)}(t+k), u^{(i)}(t+k)), \quad k = 0, \dots, H_p - 1 \tag{3}$$
$$x^{(i)}(t+k) \in \mathcal{X}^{(i)}, \ k = 1, \dots, H_p - 1 \tag{4}$$
$$x^{(i)}(t+H_p) \in \mathcal{X}_{H_p}^{(i)} \tag{5}$$
$$u^{(i)}(t+k) \in \mathcal{U}^{(i)}, \ k = 0, \dots, H_u - 1 \tag{6}$$
$$\Delta u^{(i)}(t+k) \in \Delta\mathcal{U}^{(i)} \ k = 0, \dots, H_u - 1 \tag{7}$$
$$x^{(j)}(t+1+k) = f^{(j)}(x^{(j)}(t+k), u^{(j)}(t+k)), \quad k = 0, \dots, H_p - 1 \tag{8}$$
$$x^{(j)}(t+k) \in \mathcal{X}^{(j)}, \ k = 1, \dots, H_p - 1 \tag{9}$$
$$x^{(j)}(t+H_p) \in \mathcal{X}_{H_p}^{(j)} \tag{10}$$
$$u^{(j)}(t+k) \in \mathcal{U}^{(j)}, \ k = 0, \dots, H_u - 1 \tag{11}$$
$$\Delta u^{(j)}(t+k) \in \Delta\mathcal{U}^{(j)}, \ k = 0, \dots, H_u - 1 \tag{12}$$
$$c_c^{(i,j)}(x^{(i)}(t+k), x^{(j)}(t+k)) \leq 0, \ k = 1, \dots, H_p \tag{13}$$
$$c_c^{(j,q)}(x^{(j)}(t+k), x^{(q)}(t+k)) \leq 0, \quad k = 1, \dots, H_p, q > j \tag{14}$$

where $\mathcal{X}^{(i)}$ is the set of feasible states of agent v_i, $\mathcal{X}_{H_p}^{(i)}$ is the set of feasible states of agent v_i at time H_p, $\mathcal{U}^{(i)}$ is the set of feasible control inputs of agent v_i, $\Delta\mathcal{U}^{(i)}$ is the set of feasible control input variations of agent v_i, $f^{(i)}(x^{(i)}(\cdot), u^{(i)}(\cdot))$ denotes the dynamics of agent v_i, and $c_c^{(i,j)}(x^{(i)}(\cdot), x^{(j)}(\cdot))$ denotes the coupling constraints between agents v_i and v_j. Eq. (2) is the objective function of Eq. (1). Eq. (3) constrains the solution space to the dynamics. Eq. (4)-(5) ensure feasible states, while Eq. (6)-(7) ensure feasible changes of the system inputs. The solution space of coupled agents are constrained in the same way in Eq. (8)-(12). Eq. (13) represents coupling constraints to neighbors and Eq. (14) represents the coupling constraints between the neighbors.

B. Synchronization of Plans

The local plans $\hat{x}_-^{(i)}, \ \forall i : v_i \in \mathcal{V}$ resulting from the optimization made by agent v_i are locally feasible, but may not be prediction consistent, i.e., $\hat{x}_j^{(i)}(\cdot) \neq \hat{x}_j^{(j)}(\cdot)$ for $i \neq j$ and $\alpha^{(i,j)} > 0$. We synchronize the plans using distributed local average, inspired by, e.g., consensus in [5]. For synchronizing the different plans $\hat{x}_i^{(-)}(\cdot), \ \forall i : v_i \in \mathcal{V}$, homogeneous dynamics are assumed. For agents with heterogeneous dynamics, the heterogeneous dynamics are considered in the optimization of each agent, see Eq. (3) and Eq. (8). Hence, different plans for the same agent use the same dynamics, i.e., $\hat{x}_i^{(i)}(\cdot)$ and $\hat{x}_i^{(j)}(\cdot)$ both were optimized considering $f^{(i)}(x^{(i)}(\cdot), u^{(i)}(\cdot))$. The synchronization topology is represented by the synchronization graph that is related to the adjacency matrix A as stated in Definition 2.

Definition 2 (Synchronization graph): A synchronization graph represents the information flow during the synchronization phase. If an agent v_i considers agent v_j in the planning, there is an edge $(v_j, v_i) \in E$ in the coupling graph G. The resulting plan $x_j^{(i)}(\cdot)$ is then communicated to v_j to be considered in the synchronization process. Therefore, the synchronization graph is defined as $G_s = (\mathcal{V}_s, E_s)$, where $\mathcal{V}_s = \mathcal{V}$ and $(v_i, v_j) \in E_s, \ \forall (v_j, v_i) \in E$. Therefore, the relation between the optimization adjacency matrix A and the synchronization adjacency matrix A_s is

$$A_s = A^T. \tag{15}$$

The synchronization graph consists of reduced synchronization graphs as introduced in Definition 3.

Definition 3 (Reduced synchronization graph): Reduced synchronization graphs are sub-graphs of synchronization graphs. The reduced synchronization graph $G_s^{(i)}$ for agent v_i consists of the nodes of agent v_i and all its neighbors and all edges between these nodes. It is defined as $G_s^{(i)} = (\{\mathcal{V}^{(i)} \cup v_i\}, E_s^{(i)}), \ \forall i : v_i \in \mathcal{V}_s$, where $E_s^{(i)} = \{(v_j, v_k) | v_j, v_k \in V^{(i)} \wedge (v_j, v_k) \in E_s\}$.

Agent v_i synchronizes the plan of agent v_j as

$$\overline{x}_j^{(i)}(\cdot) = \sum_{\substack{k \\ v_k \in \mathcal{V}^{(i)} \cup \{v_i\}}} \alpha^{(k,i)} \hat{x}_j^{(k)}(\cdot), \quad \forall j : v_j \in \mathcal{V}^{(i)}, \tag{16}$$

where $\alpha^{(k,i)}$ is the transposed weighting factor resulting from Eq. (15). The agents communicate the locally averaged plans $\overline{x}_-^{(-)}(\cdot)$ and iteratively perform local average steps and communicate the intermediate results to their neighbors until the plans are synchronized, i.e., $\overline{x}_k^{(i)}(\cdot) = \overline{x}_k^{(j)}(\cdot), \ \forall i, j :$

$v_i, v_j \in \mathcal{V}^k \cup \{v_k\}$. Synchronization is achieved, if the requirements of Theorem 2 are satisfied.

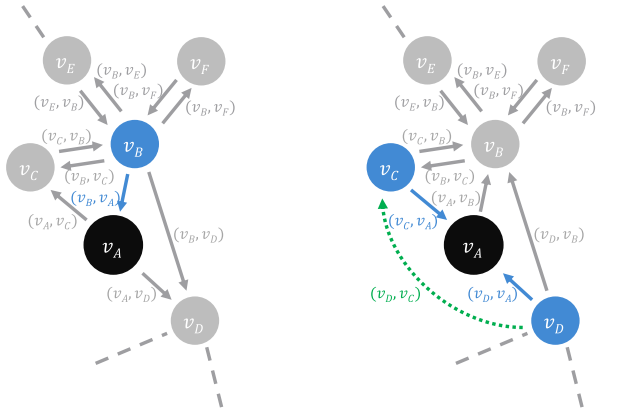
Theorem 2: The synchronization converges to a solution if and only if all reduced synchronization graphs $G_s^{(i)}$ contain a spanning tree and the coupling weights are feasible.

Proof: According to [22], the distributed local average converges to a solution if and only if the topology contains a spanning tree. We synchronize the plan of each agent using distributed local average. To achieve global synchronization, all plans must be synchronized. Hence, for each plan, a spanning tree is required. The reduced synchronization graph $G_s^{(i)}$ represents the synchronization topology for the decision $x^{(i)}(\cdot)$. Therefore, the synchronization converges to a solution if and only if all reduced synchronization graphs $G_s^{(i)}$ contain spanning trees except for the case that the coupling weights are not feasible. Infeasible coupling weighs are weights that lead to synchronized plans that result in the reference decisions. In this case, the optimization will always produce the same result. Hence, if the coupling weights are feasible, the synchronization converges to a solution if and only if all reduced synchronization graphs $G_s^{(i)}$ contain a spanning tree. ■

We now introduce an example to illustrate the synchronization process.

Example 1: Fig. 2(a) shows an example coupling graph. The coupling graph contains a spanning tree. Agent v_A plans for agents v_A, v_B, and v_D. The agents v_A, v_C, and v_D plan for agent v_A. Hence, agent v_A synchronizes its plans with agents v_C and v_D, i.e., agent v_B is irrelevant for the synchronization of the plan of agent v_A. Therefore, the coupling graph for v_A can be reduced to agents v_A, v_C, and v_D and their couplings to one another. The blue colored nodes and edges in Fig. 2(b) represent the resulting reduced synchronization graph $G_s^{(A)}$ for agent v_A. However, $G_s^{(A)}$ does not contain a spanning tree. Agent v_A synchronizes its plan $x_A^{(A)}$ to $x_A^{(C)}$ and $x_A^{(D)}$. There is no path from agent v_C to v_D or vice versa. Therefore, if $x_A^{(C)}$ and $x_A^{(D)}$ do not match, the synchronization will not converge to a solution. An edge from agent v_D to agent v_C, shown in green in Fig. 2(b), would solve this problem by completing a spanning tree. Please note that any edge that completes a spanning tree is sufficient for convergence of the synchronization. We assume feasible coupling weights in this example.

Due to in general non-convex solution space of Eq. (2)-(14), plans that are synchronized using distributed local average may conflict with each other. Therefore, if the synchronized plans are not conflict-free (line 3 of Algorithm 1), they are optimized again. The optimization (line 4 of Algorithm 1) ensures the satisfaction of all constraints. For convergence, the optimization uses the synchronized plans as reference decisions (line 8 of Algorithm 1). Hence, the deviation from the resulting plans to the synchronized plans is minimized. Algorithm 1 will converge to conflict-free decisions and achieve prediction consistency by iterating optimization and synchronization. However, the rate of convergence and; therefore, the number of iterations required for convergence depends on the coupling topology. For some



(a) Cooperation graph G.

(b) Synchronization graph G_s.

Fig. 2. The blue nodes and edges influence the optimization (a) and synchronization (b) of agent v_A. The blue colored nodes and egdes in (b) represent the reduced synchronization graph $G_s^{(A)}$.

coupling topologies, this algorithm cannot converge at all, see theorems 1 and 2. However, according to Theorem 3, if a solution exists for centralized planning, there exists a coupling topology for which Algorithm 1 will find a solution.

Theorem 3: If a solution of centralized decision-making exists, there is a coupling topology for synchronized cooperative decision-making to find a globally feasible solution.

Proof: Cooperative decision-making corresponds to centralized decision-making for a full topology, i.e., a complete coupling graph with

$$\mathcal{V}^{(i)} = \mathcal{V} \backslash v_i, \ \forall i : v_i \in \mathcal{V}. \tag{17}$$

In this case, each agent solves the central optimization problem for all agents, considering their objectives and constraints. Therefore, all agents generate the same plans, which are prediction consistent and feasible, if a solution for a centralized decision-maker exists. Since no synchronization is required, one optimization step is enough to generate these decisions. ■

The coupling topology has to meet some requirements in order to ensure conflict-free decisions and fast computation times. We will introduce these requirements in Section III-C.

C. Topology Requirements

The following paragraphs discuss coupling topology requirements:

a) Physical Requirements: Each application has its definition for decision conflicts. To guarantee conflict-free decisions, Theorem 1 states that the topology has to contain a path of length 2 or smaller for each pair of agents that may have conflicting decisions.

b) Spanning Tree: For the convergence of Algorithm 1 to prediction consistent decisions, the synchronization requires the coupling topology to contain a spanning tree. According to [23], in case of time-variant topologies, it is sufficient for distributed local average to converge if there is a topology that contains a spanning tree that is applied frequently enough. Convergence of distributed local average

is also achieved if the union graph contains a spanning tree and if the agents are connected frequently enough [24]. Since we do not expect the network to vary during a SC-DMPC iteration, we require a spanning tree for all reduced synchronization graphs $G_s^{(i)}$ as stated in Theorem 2.

c) Maximum Node Degree: A higher node degree results in higher optimization times. To achieve low computation times, the node degree should be as small as possible. Since the agents plan in parallel, the agent with the highest computation time limits the algorithm.

d) Diameter of Reduced Synchronization Graphs: The synchronization process depends on the connectivity of the coupling graph [23]. A higher connectivity results in higher convergence rates. The connectivity of the coupling graph is represented by the second smallest eigenvalue of its Laplacian matrix $\mathcal{L}$. According to [25], the lower bound of the second smallest eigenvalue of $\mathcal{L}$ is given as

$$\frac{4}{N \cdot d} \ , \tag{18}$$

where N is the number of agents and d is the diameter of the graph. The diameter of a graph is defined as the length of the longest of all shortest paths between two nodes. For applications with a fixed number of agents N, the lower bound depends on the diameter. A small diameter results in a high lower bound for the connectivity and, therefore, fast convergence is achieved. However, a small diameter may result in high node degrees. High node degrees increase the computation times of the planning steps in line 4 of Algorithm 1, while small diameters of the synchronization graphs decrease the convergence times of the synchronization in line 7 of Algorithm 1. Hence, there is a tradeoff between small node degrees and small diameters of the reduced synchronization graphs.

D. Algorithm Enhancements

In order to improve Algorithm 1, we present enhancements in this subsection. We adapt the synchronization process to also forward information about the plans of neighboring agents in each iteration. Hence, the plans are not only provided to the coupled neighbors, but also to agents that have a path of length 2 in the coupling graph. These agents use the forwarded plans as non-cooperative constraints in their planning to avoid conflicts. To this end, we introduce the set of indirectly known agents $\mathcal{V}_o^{(i)} \subseteq \mathcal{V} \backslash \mathcal{V}^{(i)}$ for each agent v_i that consists of all agents that have a minimum path of length 2 to v_i in the coupling graph G. If the plans are locally feasible in each agent, it holds that no conflicts occur between v_i and v_j nor between v_i and v_o for any pair $v_j \in \mathcal{V}^{(i)}$ and $v_o \in \mathcal{V}_o^{(i)}$.

Algorithm 1 does not depend on the forwarding of synchronized plans, see Theorem 1. Nevertheless, the forwarding of synchronized plans to the neighbors of the neighbors improves the convergence time. The forwarded plans are prediction consistent, since the synchronization is repeated until no plans change in the last step of the synchronization in Eq. (16).

IV. NUMERICAL EVALUATION

This section presents our evaluation results. We evaluate the SC-DMPC method in a networked trajectory planning simulation for networked and autonomous vehicles. Networked and autonomous vehicles make driving decisions without driver interaction and share information over a communication network. For networked and autonomous vehicles, the driving path has to be two times continuously differentiable [26]. This is due to the continuity of the change of the steering angle. Hence, the reference trajectories should be two times continuously differentiable to be feasible to the vehicle's dynamics.

Theorem 4: The synchronized trajectories are feasible to dynamics.

Proof: Assumption: f and g are two times continuously differentiable. Proof that $h = \alpha_1 f + \alpha_2 g$ is two times continuously differentiable.

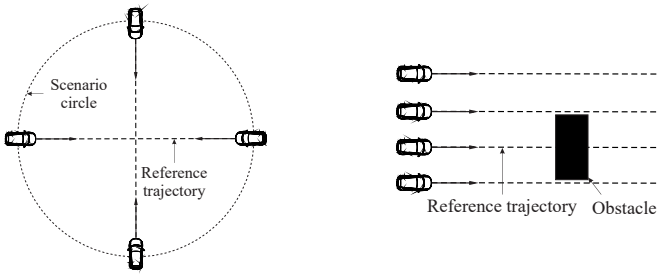
f and g are two times continuously differentiable. Therefore, for all $\lambda \in \mathbb{R}$, $\phi = \lambda f$ and $\psi = \lambda g$ are two times continuously differentiable. Since ϕ and ψ are two times continuously differentiable, ϕ'' and ψ'' are continuous. Then, $\theta'' = \phi'' + \psi''$ is continuous. Therefore, $h = \alpha_1 f + \alpha_2 g$ is continuous.

Since all system inputs lie in their bounds, i.e., $u_{min} \leq u^{(i)} \leq u_{max}, \ \forall i : v_i \in \mathcal{V}$, the synchronized system inputs lie within the same bounds u_{min} and u_{max}. ■

According to Theorem 4, the synchronized trajectories are feasible to the vehicles' dynamics. We now present our evaluation methodology in Subsection IV-A.

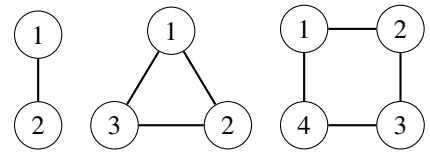
A. Methodology

We simulate two different scenarios, a circle and a parallel scenario. Fig. 3 shows a sketch of both evaluation scenarios for 4 vehicles. In the circle scenario, the vehicles start on a circle while driving through the center. In order to avoid collisions in the center, the vehicles have to adjust their trajectories. The vehicles in the parallel scenario drive in parallel until they reach an obstacle. To avoid collisions, the vehicles have to adjust their trajectories. The speed is fixed in both scenarios, so that the vehicles can only change their steering. Please note that the number of vehicles in each scenario is variable. The scenarios are referred to as N-Parallel and N-Circle, where N denotes the number of vehicles.



(a) 4-Circle scenario. (b) 4-Parallel scenario.

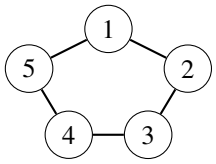
Fig. 3. Sketch of the circle and parallel evaluation scenarios with 4 vehicles.



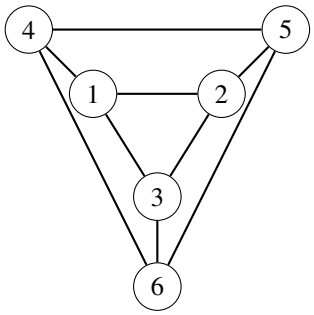
(a) 2 vehicles

(b) 3 vehicles

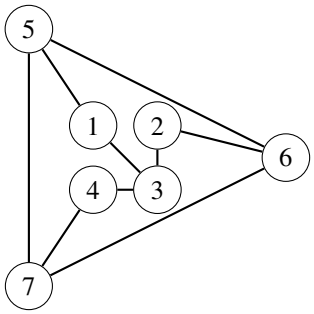
(c) 4 vehicles



(d) 5 vehicles



(e) 6 vehicles



(f) 7 vehicles

Fig. 4. Evaluation topologies for the circle scenario for different numbers of vehicles.

We also use different couplings to evaluate the impact on the SC-DMPC. In the circle coupling, all vehicles may collide. Hence, topologies with a diameter of 2 or less are required. Figure 4 show our evaluation topologies for the circle scenario. In the parallel scenario we make use of the fact that only neighboring vehicles may collide. Therefore, the topology may have a higher diameter than 2. We use two different groups of couplings in the parallel scenario. First, the complete cluster coupling in which the vehicles are grouped into clusters that are internally completely coupled and externally coupled by one link. The corresponding adjacency matrix A_{cc} is defined as

$$A_{cc} = \begin{pmatrix} 1 & \cdots & 1 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & 1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & \cdots & 1 & 1 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 1 & 1 & \cdots & 1 \\ 0 & \cdots & 0 & 0 & 1 & 1 & \cdots & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & 1 & \cdots & 1 \end{pmatrix}.$$

The second coupling is the nearest neighbor coupling in which each vehicle communicates with its left and right neighbor. The adjacency matrix is given as

$$A_{nn} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 & 1 \\ 1 & \ddots & 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \ldots\ldots & 0 & 1 & 1 & \end{pmatrix}.$$

When we use weighted couplings, we define the weights as

$$\alpha^{(i,j)} = 0.1 + (N - i) \cdot 0.3 + \mod (i, 2) \cdot 0.6, \quad (19)$$

i.e., all outgoing edges of each agent have the same weights. We evaluate the reference deviation, computation time, and number of iterations of the SC-DMPC method in both scenarios with different network topologies. For comparison, we also evaluate the Centralized Model Predictive Control (CMPC) approach. In the following, we introduce our evaluation metrics. Table I lists our simulation parameters.

a) Reference Deviation: We evaluate the reference deviation as the cumulative distance of the final trajectories and the original reference trajectories of each time step. The synchronized trajectories in each iteration do not influence the reference deviation in this evaluation. The distance metric is the euclidean distance between the sample points. The reference deviation metric D is given as

$$D = \sum_{t=1}^{N_t} \sum_{i=1}^{N} \sum_{k=1}^{H_p} ||x^{(i)}(t+k) - r^{(i)}(t+k)||_2^2, \quad (20)$$

where N_t denotes the number of simulation steps and $x^{(i)}(\cdot) = x_j^{(i)}(\cdot),\ \forall j : v_j \in \mathcal{V}^{(i)}$. We compare the reference deviation of SC-DMPC and CMPC as

$$\frac{D_{SC-DMPC}}{D_{CMPC}}, \quad (21)$$

where $D_{SC-DMPC}$ is the reference deviation of the SC-DMPC and D_{CMPC} is the reference deviation of the CMPC.

TABLE I
EVALUATION PARAMETERS

Parameter	Description	Value
T_s	Sample time	0.4s
H_p	Prediction horizon	14
H_u	Control horizon	3
$Q^{(i)}(k)$	Cost of tracking error	$1m \cdot q^{(i)}(k)$
$Q^{(i)}(H_p)$	Cost of terminal tracking error	$10m \cdot q^{(i)}(k)$
$R^{(i)}(k)$	Cost of steering angle variation	*Circle*: $1200\ {}^{1}/_{rad^2}$ *Parallel*: $800\ {}^{1}/_{rad^2}$
$u_{max}^{(i)}$	Steering angle limit	3°
$\Delta u_{max}^{(i)}$	Steering angle variation limit	6°

b) Computation Time: The computation time is measured for the optimization only, i.e., no time for initializing variables or similar is included in the computation time. For CMPC, there is only one optimization, hence the computation time for the optimization step is the overall computation time

$$ct_{CMPC} = t_{opt}. \tag{22}$$

In SC-DMPC, there are multiple optimizations per time step, i.e., one for each vehicle and iteration. Since the vehicles optimize their trajectories in parallel in each iteration, we measure the sum of all maximum optimization times per iteration as overall computation time as follows:

$$ct_{SC-DMPC} = \sum_{k=1}^{n} \max_{i}(t_{opt}^{(i)}(k)), \tag{23}$$

where n is the number of iterations and $t_{opt}^{(i)}(k)$ is the optimization time of vehicle v_i at the k'th iteration.

c) Number of Iterations: The number of optimizations represents the number of iterations of the optimization and synchronization process. The number of synchronization iterations corresponds to the number of iterations of the synchronization process itself.

B. Reference Deviation

Fig. 5 shows the trajectory deviation according to Eq. (21) in the circle scenario. For up to four vehicles, the trajectories have about the same cumulative deviation in the SC-DMPC and CMPC approach. For higher numbers of vehicles, the trajectory deviation in the SC-DMPC scenario is higher than the deviation of the CMPC trajectories. The maximum reference deviation is about 20% higher for the SC-DMPC than for CMPC for 6 vehicles. This trend is also true for the parallel scenario with complete cluster coupling shown in Fig. 6. For two vehicles, the topology is complete and, hence, the trajectory deviation is the same for SC-DMPC and CMPC.

C. Computation Time

Fig. 7 shows the median and maximum computation times of SC-DMPC and CMPC in the circle scenario on a logarithmic scale. The computation time of both approaches

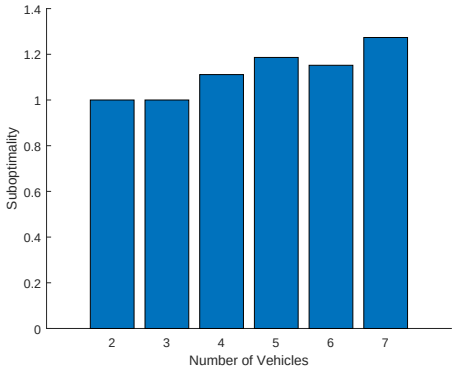


Fig. 5. Reference deviation of the SC-DMPC algorithm compared to CMPC in the circle scenario with different numbers of vehicles.

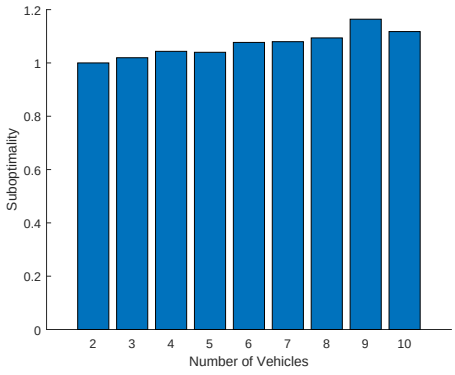


Fig. 6. Reference deviation of the SC-DMPC algorithm with complete cluster coupling compared to CMPC in the parallel scenario with different numbers of vehicles.

are similar for up to four vehicles. For more vehicles, the computation time of CMPC increases to over 200 s in the median and over 3000 s in the maximum for seven vehicles. The computation time of SC-DMPC increases to about 500 ms in the median and about 10 s in the maximum. Hence, in the 7-Circle scenario, a speedup of about 300 is achieved.

Figs. 8 and 9 show the computation time in the parallel scenario with complete cluster coupling and nearest neighbor coupling. For complete cluster coupling, the median computation time is not much lower for SC-DMPC than for CMPC, even for 10 vehicles. However, the maximum computation time is higher for the SC-DMPC approach. This may be due to high number of iterations until convergence is achieved. For nearest neighbor coupling, nevertheless, the computation time is higher than for complete cluster coupling. For 10 vehicles, the SC-DMPC did not converge for the nearest neighbor coupling. This shows that the topology has an impact on the overall computation time and feasibility of the SC-DMPC.

D. Number of Iterations

We also evaluated the number of iterations for the SC-DMPC approach. Fig. 10 shows the number of iterations for the circle scenario. It shows the average and maximum

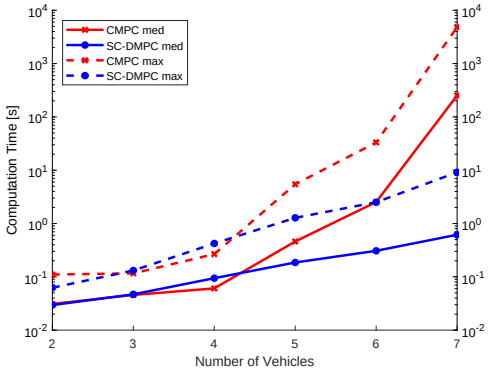


Fig. 7. Median and maximum computation time of the SC-DMPC and CMPC algorithms in the circle scenario with different numbers of vehicles.

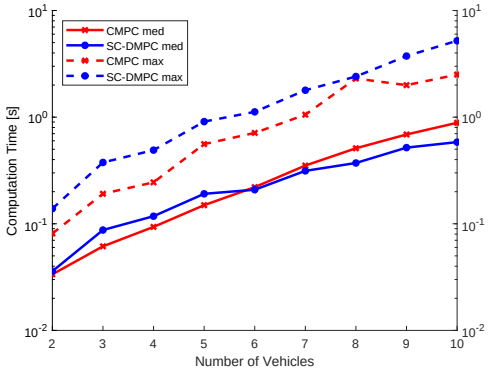


Fig. 8. Median and maximum computation time of the SC-DMPC and CMPC algorithms in the parallel scenario with complete cluster couplings and different numbers of vehicles.

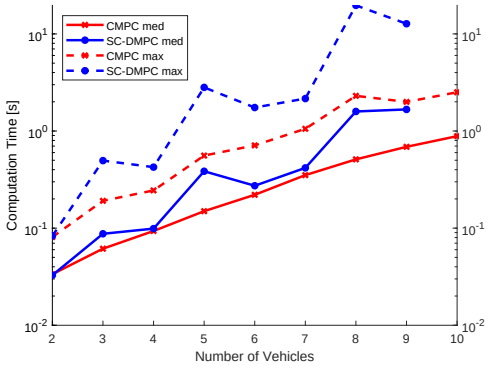


Fig. 9. Median and maximum computation time of the SC-DMPC and CMPC algorithms in the parallel scenario with nearest neighbor couplings and different numbers of vehicles.

number of iterations for optimization and the average number of iterations for synchronization. For two and three vehicles, only one optimization is required and the trajectories are instantly synchronized. This is because of the topology structure, since the evaluated topologies in Fig. 4 are complete for up to three vehicles. The scenarios with more than three vehicles require at most 7 optimizations per time step, while the average is about 1.5 optimizations. The number of required synchronization steps increases with increasing number of vehicles.

Figs. 11 and 12 show the number of iterations required in the parallel scenario with the complete cluster coupling and nearest neighbor coupling. The complete cluster coupling shows a maximum number of optimizations of 3 for most number of vehicles. In the parallel scenario with 7 vehicles, 4 optimizations were required at maximum per time step. The average number of optimizations is about 1.5 for most numbers of vehicles. The average number of synchronizations increases with increasing numbers of vehicles. The SC-DMPC with the nearest neighbor coupling shows a high maximum number of optimizations, while the average is almost constant. The maximum number of optimizations is in the scenario with 8 vehicles, where 45 optimizations were required in one time step. However, for the 10 vehicles scenario the SC-DMPC did not converge at all. This is because the requirements of Theorem 2 are not fulfilled. The average number of iterations for synchronization is almost constant.

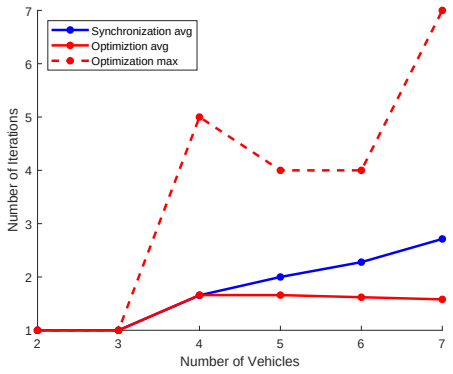


Fig. 10. Average and maximum number of optimization steps and average number of synchronization iterations per time step of the SC-DMPC algorithm in the circle scenario with different numbers of vehicles.

E. Weighting

Fig. 13 shows the computation time of the parallel scenario with a weighted and unweighted nearest neighbor coupling compared to CMPC. The unweighted scenario did not converge to a solution for 10 vehicles. However, the SC-DMPC converges in the parallel scenario for 10 vehicles with weighted nearest neighbor couplings. The weightings enabled convergence for all scenarios. Furthermore, the median and maximum computation times of the SC-DMPC with weighted nearest neighbor couplings are lower for most numbers of vehicles. This shows that the weightings of

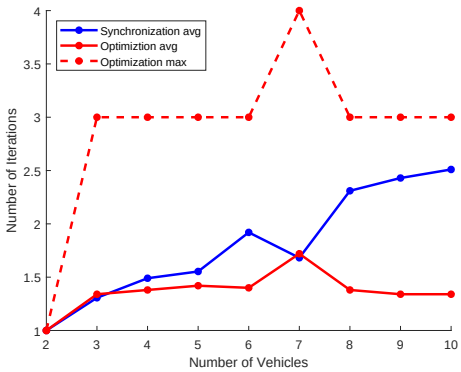


Fig. 11. Average and maximum number of optimization steps and average number of synchronization iterations per time step of the SC-DMPC algorithm in the parallel scenario with complete cluster couplings and different numbers of vehicles.

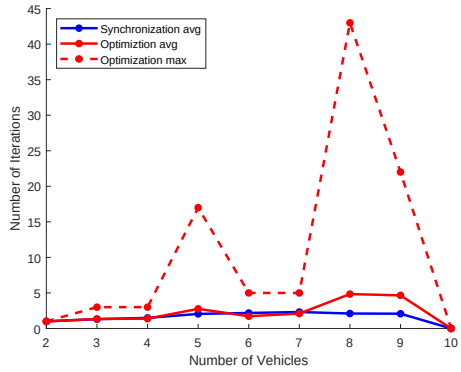


Fig. 12. Average and maximum number of optimization steps and average number of synchronization iterations per time step of the SC-DMPC algorithm in the parallel scenario with nearest neighbor couplings and different numbers of vehicles.

couplings not only have an impact on the feasibility, but also on the computation time.

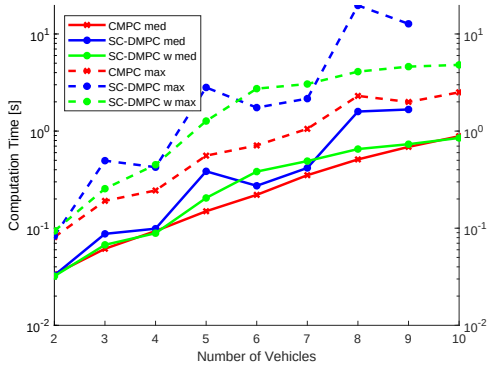


Fig. 13. Computation time of SC-DMPC with unweighted nearest neighbor couplings in blue and weighted nearest neighbor coupling in green in the parallel scenario compared to CMPC in red.

V. CONCLUSION

This article presents an algorithm for cooperative decision-making. Multiple time-triggered agents are coupled over directed links in a time-variant topology and make their decisions in parallel. The agents synchronize their decisions at each time step to achieve prediction consistency. Synchronization is achieved using iterative distributed average of local plans. Our algorithm alternates planning and synchronization steps to iteratively converge to a globally feasible solution.

Our evaluations show that the computation time of our algorithm for 7 agents is up to 300 times faster than in CMPC approaches. The reference deviation increased by 20%. Nevertheless, in some scenarios there is no improvement of computation time compared to CMPC. We showed that the topology has a high impact on our algorithm's computation time and that some topologies may lead to infeasibility. We presented requirements and goals for the topologies to guarantee feasible trajectories and low computation times. The scalability of computation time show that this algorithm can be real-time capable. Future work will contain methods for topology generation and tests on real vehicle hardware.

REFERENCES

[1] K. P. Sycara, "Multiagent systems," *AI magazine*, vol. 19, no. 2, pp. 79–79, 1998.

[2] R. A. Gupta and M.-Y. Chow, "Networked control system: Overview and research trends," *IEEE transactions on industrial electronics*, vol. 57, no. 7, pp. 2527–2535, 2009.

[3] A. Bemporad, M. Heemels, M. Johansson *et al.*, *Networked control systems*. Springer, 2010, vol. 406.

[4] B. Alrifaee, "Networked model predictive control for vehicle collision avoidance," Ph.D. dissertation, RWTH Aachen University, 2017.

[5] J. Lunze, *Control theory of digitally networked dynamic systems*. Springer, 2014, vol. 1.

[6] J. Hoc, "Towards a cognitive approach to human-machine cooperation in dynamic situations," *Int. J. Hum. Comput. Stud.*, vol. 54, pp. 509–540, 2001.

[7] 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.

[8] M. A. Müller, M. Reble, and F. Allgöwer, "Cooperative control of dynamically decoupled systems via distributed model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 22, no. 12, pp. 1376–1397, 2012.

[9] P. Trodden and A. Richards, "Cooperative distributed mpc of linear systems with coupled constraints," *Automatica*, vol. 49, no. 2, pp. 479–487, 2013.

[10] S. Blasi, M. Kögel, and R. Findeisen, "Distributed model predictive control using cooperative contract options," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 448–454, 2018.

[11] P. Trodden and A. Richards, "Adaptive cooperation in robust distributed model predictive control," in *2009 IEEE Control Applications,(CCA) & Intelligent Control,(ISIC)*. IEEE, 2009, pp. 896–901.

[12] J. Maestre, D. M. De La Peña, E. Camacho, and T. Alamo, "Distributed model predictive control based on agent negotiation," *Journal of Process Control*, vol. 21, no. 5, pp. 685–697, 2011.

[13] A. Maxim, J. M. Maestre, C. F. Caruntu, and C. Lazar, "Min-max coalitional model predictive control algorithm," in *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*. IEEE, 2019, pp. 24–29.

[14] F. Fele, E. Debada, J. M. Maestre, and E. F. Camacho, "Coalitional control for self-organizing agents," *IEEE Transactions on Automatic Control*, vol. 63, no. 9, pp. 2883–2897, 2018.

[15] P. Chanfreut, J. Maestre, F. J. Muros, and E. F. Camacho, "A coalitional control scheme with topology-switchings convexity guarantees," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1096–1101.

[16] C. Kloock and H. Werner, "Prediction mismatch in distributed model predictive control," in *2020 European Control Conference (ECC)*. IEEE, 2020, pp. 1224–1229.

[17] A. Mirheli, M. Tajalli, L. Hajibabai, and A. Hajbabaie, "A consensus-based distributed trajectory control in a signal-free intersection," *Transportation research part C: emerging technologies*, vol. 100, pp. 161–176, 2019.
[18] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Autonomous Robots*, vol. 43, no. 5, pp. 1079–1100, 2019.
[19] R. Carli and M. Dotoli, "Cooperative distributed control for the energy scheduling of smart homes with shared energy storage and renewable energy source," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 8867–8872, 2017.
[20] Y. Zhang, N. Rahbari-Asr, J. Duan, and M.-Y. Chow, "Day-ahead smart grid cooperative distributed energy scheduling with renewable and storage integration," *IEEE Transactions on Sustainable Energy*, vol. 7, no. 4, pp. 1739–1748, 2016.
[21] M. Zhu and S. Martínez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2011.
[22] W. Ren and R. W. Beard, "Consensus seeking in multiagent systems under dynamically changing interaction topologies," *IEEE Transactions on automatic control*, vol. 50, no. 5, pp. 655–661, 2005.
[23] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control systems magazine*, vol. 27, no. 2, pp. 71–82, 2007.
[24] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.
[25] B. Mohar, Y. Alavi, G. Chartrand, and O. Oellermann, "The laplacian spectrum of graphs," *Graph theory, combinatorics, and applications*, vol. 2, no. 871-898, p. 12, 1991.
[26] D. Meek and D. Walton, "Clothoid spline transition spirals," *Mathematics of computation*, vol. 59, no. 199, pp. 117–133, 1992.

APPENDIX

A. Proof of Theorem 1

We proof global feasibility $\Leftrightarrow \Big(C1 \wedge C2 \wedge \big(C3 \vee C4\big)\Big)$ by showing $\Big(C1 \wedge C2 \wedge \big(C3 \vee C4\big)\Big) \Rightarrow$ global feasibility and global feasibility $\Rightarrow \Big(C1 \wedge C2 \wedge \big(C3 \vee C4\big)\Big)$.

a) To prove $\Big(C1 \wedge C2 \wedge \big(C3 \vee C4\big)\Big) \Rightarrow$ global feasibility, we show global infeasibility $\Rightarrow \neg\Big(C1 \wedge C2 \wedge \big(C3 \vee C4\big)\Big)$.
Assume globally infeasible decisions.

- Case 1: The decisions of agent v_i and v_j are locally feasible (C1) and $x_i^{(i)}(\cdot) = x_i^{(j)}(\cdot)$ and $x_j^{(i)}(\cdot) = x_j^{(j)}(\cdot),\ \forall i,j : v_i, v_j \in \mathcal{V}$, i.e., the decisions are prediction consistent (C2). Since the decisions are globally infeasible, there are agents v_i and v_j with conflicting decisions, but are not coupled with each other (C3) and there is no agent $h \in V^{(i)} \cap V^{(j)}$, i.e., the agents v_i and v_j have no common neighbor (C4). Otherwise, the decisions would be globally feasible.

- Case 2: The decisions of agents v_i and v_j are locally feasible (C1) and each pair of agents v_i, v_j that may conflict with each other are coupled by $v_i \in \mathcal{V}^{(j)}$ or $v_j \in \mathcal{V}^{(i)}$ (C3) or have a common neighbor (C4). There is a pair of agents v_i, v_j with $x_i^{(i)}(\cdot) \neq x_i^{(i)}(\cdot)$, i.e., the decisions are not prediction consistent (C2). Otherwise, the decisions would be globally feasible.

- Case 3: The decisions are prediction consistent $x_i^{(i)}(\cdot) = x_i^{(i)}(\cdot),\ \forall i,j : v_i, v_j \in \mathcal{V}$ and each pair of agents v_i, v_j that may conflict with each other are coupled by $v_i \in \mathcal{V}^{(j)}$ or $v_j \in \mathcal{V}^{(i)}$ (C3) or have a common neighbor (C4). There are agents that have no locally feasible decisions (C1). Otherwise, the decisions would be globally feasible.

b) To prove global feasibility $\Rightarrow \Big(C1 \wedge C2 \wedge \big(C3 \vee C4\big)\Big)$, we show $\neg\Big(C1 \wedge C2 \wedge \big(C3 \vee C4\big)\Big) \Rightarrow$ global infeasibility.

- Case 1: Assume locally infeasible decisions. These decisions can not be globally feasible.
- Case 2: Assume prediction inconsistent decisions. These decisions can not be globally feasible.
- Case 3: Assume a pair of agents v_i, v_j that may conflict are not coupled, i.e., $v_i \notin \mathcal{V}^{(j)}$ and $v_j \notin \mathcal{V}^{(i)}$ and they have no common neighbor $\nexists v_h : v_i, v_j \in \mathcal{V}^{(h)}$. These agents do not consider each other in their decision-making, i.e., they do not resolve conflicts. Therefore, decisions can not be globally feasible.