Unifying Control and Verification of Cyber-Physical Systems (UnCoVerCPS)

WP2 Online Decision Making and Control (Task 2.2)
Report on Distributed Model Predictive Control for Cyber-Physical Systems and Efficient Computation

WP2	**D2.2 – Report on Distributed Model Predictive Control for Cyber-Physical Systems and Efficient Computation**
Authors	Maria Prandini - PoliMi Olaf Stursberg, Zonglin Liu, Damian Kontny - UKS Daniel Hess - DLR Joshue Manuel Pérez Rastelli - Tec
Short Description	We present novel techniques for distributed model predictive control of cyber-physical systems, possibly affected by endogenous and exogenous uncertainty. We consider in particular networked systems that are composed by heterogeneous subsystems coupled in their decision making because they are sharing some common resource. The goal is to optimize the overall system performance while also accounting for possible communication constraints and limits in information sharing due to privacy issues. When accounting for uncertainty a-priori will lead to too conservative or even infeasible solutions, on-line (re)computation is needed and model predictive control schemes are adopted. Approximate solutions that enhance computability while preserving feasibility of the solution are developed to cope with reactivity requirements. Transfer of the achieved results to applications is also discussed.
Keywords	Model predictive control, distributed and stochastic optimization, networked systems.
Deliverable Type	Report
Dissemination level	Public
Delivery Date	30 June 2017
Contributions by	—
Internal review by	Daniel Hess Joshue Manuel Pérez Rastelli Mark Burgin
External review by	—
Internally accepted by	Matthias Althoff
Date of acceptance	

Document history:

Version	Date	Author/Reviewer	Description
1.0	28 April 2017	Prandini et al.	Internal review version
1.1	5 June 2017	Prandini, Stursberg, Li, Kontny	Revised version
2.0	15 June 2017	Prandini, Stursberg	Final version

Contents

1 Introduction

Cyber-physical systems are pervasive in many sectors, and in particular in the energy and transportation domains. In order to ensure a safe, reliable, but at the same time high performance operation of the underlying systems, traditional methods need to be revisited, and conceptually new approaches need to be developed.

The challenges are immense since these systems are growing ever more complex: they typically include, not only the interleaving between software logic components and physical continuous elements, but also the interconnection of many subsystems that can interact either physically or via information exchange through a communication network, thus bringing networked cyber-physical systems into the picture.

In this report we address explicitly the interacting and distributed nature of complex cyber-physical systems arising in the energy and transportation application domains, proposing solutions that account for three main complexity features that are prominent in such systems:

i) heterogeneity – the subsystems composing a large scale system may have different parameters, objectives, physical and/or technological constraints,

ii) uncertainty – each subsystem is affected by both endogenous (e.g., incomplete knowledge of some of the underlying processes) and exogenous (e.g., environmental) uncertainties,

iii) locality of information – not all interacting subsystems are willing to share information relevant to their processes and/or not all communication links may be available.

According to the type of application, the interacting subsystems may cooperate to achieve some overall objective, or they may act in a non-cooperative manner, seeking to achieve their individual objectives. Each case poses different information constraints and calls for different algorithmic solutions. Therefore, to capture all types of objectives that may underly decision making problems in complex interacting systems, both cooperative and non-cooperative set-ups have been considered.

In Section 2 of this deliverable, we describe the cooperative solutions developed within work package 2. Studies on a non-cooperative set-up have been performed as well within work package 2. The interested reader is referred to [1, 2] where the Nash equilibrium of the associated game is characterized and a decentralized algorithm is proposed for its computation. Interestingly, as the number of agents grows to infinity, the Nash equilibrium tends to the social optimum of the centralized cooperative version of the problem.

The crucial part of designing a cyber-physical system with (fully or partially) automated behavior is to equip the cyber-component with algorithms for control and planning, which affect

the physical part (through actuation devices) consistent to given specifications. Algorithms for control and planning of a system interacting with a dynamic environment have to be reactive, i.e., specifications have to be satisfied also for time-varying interactions with the environment. In addition, the specifications like goals to be attained or safety restrictions to be enforced may change over time. This is indeed the case of automated driving or human-robot interaction, since not all behavior of the environment possibly occurring during operation can be foreseen and an off-line a-priori solution that accounts for all possible behaviors will be too conservative and even infeasible. This motivates the investigation of online techniques for decision making within work package 2.

For many realizations of cyber-physical systems (partial) modeling or identification of the physical behavior of the system and the environment is possible, thus enabling the use of model-based predictions for decision making. This jointly with the need to account for constraints (e.g., collision avoidance in automated driving or sharing of resources as storage systems in a smart grid) suggests the use of Model Predictive Control (MPC). The investigations in this work package aim at extensions of MPC to networked cyber-physical systems affected by real-time constraints. Distributed MPC solutions are discussed in Section 3 with reference to the class of piecewise-affine systems with time-varying constraints and discrete inputs.

The success of applying an MPC scheme will be inherently bound to the question whether the optimization in each MPC iteration can be completed timely. The computational complexity is largely determined by the combinatorics of the discrete-event part of the hybrid dynamics and the number of constraints for the continuous variables (both depending on the time setting). In Section 4 of this deliverable we illustrate techniques to enhance the computational efficiency in determining suboptimal and approximated solutions. While a certain (small) loss of performance seems acceptable in favor of an enhanced real-time computability, the requirement of maintaining certain key requirements like stability and feasibility has to be kept.

This deliverable presents contributions developed within work package 2 of the UnCoVerCPS project that go beyond the state of the art in several directions:

- the approaches for distributed optimization in multi-agent systems described in Section 2 are able to jointly cope with heterogeneity, locality of information, and uncertainty, and are also resilient to communications failure. They work also for non-differentiable functions and are numerically more stable than sub-gradient based techniques;
- the method described in Section 3 extends existing algorithms for distributed model predictive control to switching affine dynamics;

- Section 4 introduces new methods of distributed model predictive control, in which the effects of interacting subsystems are cast into time-varying constraints, and which are tailored to low online computational effort. More specifically, the method in Section 4.2 extends existing schemes to mixed inputs, while the method in Section 4.3 uses offline designed control laws to react to changing state constraints very fast.

Various examples of application to the case studies in work package 5 on smart grid (and related energy management problems), automated driving, and human-robot interaction are presented alongside the theoretical developments in Sections 2, 3, and 4. The goal is to show the performance of the proposed approaches for distributed optimization and MPC with enhanced computational capabilities on simple, easy to interpret, numerical examples. The concluding Section 5 provides an outlook on possible transfer of the described approaches to more complex instances of the case studies in work package 5.

2 Distributed Optimization

2.1 Introduction

In this section we present the results developed within work package 2 on distributed cooperative optimization for networked systems composed by interacting heterogeneous subsystems, [3–6]. The agents decisions are coupled via common decision vectors (see [3, 4]) or joint constraints like in the case of resource sharing (see [5, 6]), whilst the cost function to be minimized has a separable structure. The communication structure of the network can be time-varying and commute between different topologies, which makes the algorithm resilient to (temporary) failures. In the case of definitive communication failures that break the network in two or more sub-networks, a feasible though sub-optimal solution can be found in the resource sharing case (see Remark 2.4).

Examples of application to energy systems will be presented alongside the theoretical developments, to the purpose of illustrating the effectiveness of the proposed methods. References to the related literature will be given when presenting the specific application. This will also serve the purpose of showing the connection with the smart grid case study in work package 5.

We also address the case where the agents' constraint sets are affected by a possibly common uncertainty vector. Our approach is particularly suited for data-driven optimization where agents are provided with a given set of uncertainty realizations, a setting that is not addressed in the literature. Also, we differentiate from existing approaches in that we do not impose any assumptions on the distribution of the uncertainty neither on geometry of the uncertainty sets. Overall, we offer a unifying framework for distributed optimization, since we take into account time-varying graphs, constraints and uncertainty, features that are typically treated separately in the literature. Results are proven under some assumptions, which are actually standard in the related literature. The only restrictive assumption is convexity, which is however needed not only for proving consensus and optimality but also for computational purposes. Additional assumptions regarding the connectivity of the possibly time varying communication graph are needed to allow information to reach all the agents in the network. Also, parameters in the distributed algorithms need to be appropriately tuned to enforce consensus and optimality at the same time. These latter assumptions are not restrictive at all but part of the design.

Within work package 2, we also studied the cases when the coupling is due to the cost function only or to both cost function and constraints. The interested reader is referred to

the work presented in [7, 8] and [9], where decentralized schemes with a central unit collecting some pieces of information from all agents are introduced to cope with those set-ups.

In all our work on distributed and decentralized optimization, we address static problems. However, problems with discrete-time dynamics can actually be reformulated into static ones by propagating the dynamics in time, thus expressing the system state as a function of the initial condition and the decision variables/inputs, and substituting it in the objective and the constraint functions.

As for continuous time systems, the interested reader is referred to the related literature on distributed optimization, e.g., [10–15], and references therein.

2.2 Coupling via global decision variables

We consider a time-varying network of m agents that communicate to cooperatively solve an optimization problem of the form

$$\mathcal{P}: \min_{x\in\mathbb{R}^n} \sum_{i=1}^{m} f_i(x) \quad \text{subject to } x \in \bigcap_{i=1}^{m} X_i, \tag{1}$$

where $x \in \mathbb{R}^n$ represents a vector of n decision variables. For each $i = 1, \dots, m$, $f_i(\cdot): \mathbb{R}^n \to \mathbb{R}$ is the objective function of agent i, whereas $X_i \subseteq \mathbb{R}^n$ is its constraint set. X_i is supposed to represent all constraints to the decision vector imposed by agent i, including explicit constraints expressed e.g., by inequalities like $h_i(x) \leq 0$ and restrictions to the domain of the objective function f_i.

Since most of the subsequent results are based on $f_i(\cdot)$ and X_i being convex, we formulate the following assumption:

Assumption 2.1 *[Convexity] For each $i = 1, \dots, m$, the function $f_i(\cdot): \mathbb{R}^n \to \mathbb{R}$ and the set $X_i \subseteq \mathbb{R}^n$ are convex.*

Problem $\mathcal{P}$ cannot be solved in a centralized fashion if $f_i(\cdot)$ and X_i represent private information, available only to agent i, and even if all necessary information (objective functions and constraint sets) was available to all agents, imposing all the constraints in one shot, by encoding $\bigcap_{i=1}^{m} X_i$, may result in a computationally intensive program. To alleviate this and account for information privacy, we follow a distributed, iterative approach, where each agent i solves an appropriate, local optimization problem and exchanges information with other agents based on the outcome of this optimization.

Algorithm 1 Distributed algorithm

1: **Initialization**
2: Set $\{a_j^i(k)\}_{k\geq 0}$, for all $i, j = 1, \ldots, m$.
3: Set $\{c(k)\}_{k\geq 0}$.
4: $k = 0$.
5: Consider $x_i(0) \in X_i$, for all $i = 1, \ldots, m$.
6: **For** $i = 1, \ldots, m$ **repeat until convergence**
7: $z_i(k) = \sum_{j=1}^m a_j^i(k) x_j(k)$.
8: $x_i(k+1) = \arg\min_{x_i \in X_i} f_i(x_i) + \frac{1}{2c(k)} \|z_i(k) - x_i\|^2$.
9: $k \leftarrow k + 1$.

We will show that under certain structural and communication assumptions agents reach consensus to an optimal solution of $\mathcal{P}$ (note that $\mathcal{P}$ does not necessarily admit a unique solution). The basic steps of the proposed approach are summarized in Algorithm 1. Initially, each agent i, $i = 1, \ldots, m$, starts with some tentative value $x_i(0)$, which constitutes its estimate of what a minimizer of $\mathcal{P}$ might be (step 5, Algorithm 1). This estimate belongs to the local constraint set X_i of agent i, but not necessarily to $\bigcap_{i=1}^m X_i$. One sensible choice for $x_i(0)$ is to set it such that $x_i(0) \in \arg\min_{x_i \in X_i} f_i(x_i)$. At iteration k, each agent i constructs a weighted average $z_i(k)$ of those solutions $x_j(k)$, $j = 1, \ldots, m$, communicated by the other agents and its local one (step 7, Algorithm 1). Coefficient $a_j^i(k) \in \mathbb{R}_+ \cup \{0\}$ indicates how agent i weights the solution received by agent j at iteration k, and $a_j^i(k) = 0$ encodes the fact that agent i does not require any information from agent j at iteration k, or the communication link between agents i and j is not active at iteration k. Agent i then solves a local minimization problem, seeking the optimal solution within X_i that minimizes a performance criterion, which is defined as a linear combination of the local objective function $f_i(x_i)$ and a quadratic term, penalizing the difference from $z_i(k)$ (step 8, Algorithm 1). The relative importance of these two terms is dictated by $c(k) \in \mathbb{R}_+$. Note that, under Assumption 2.1 and due to the presence of the quadratic penalty term, the resulting problem is strictly convex with respect to x_i, and hence admits a unique solution.

We impose some additional assumptions on the structure of problem $\mathcal{P}$ in (1) and the communication set-up that is considered in this work. In particular, we shall show that Algorithm 1 provides a fully distributed implementation, where agent i uses information only from neighboring agents. This information exchange is time-varying, since the weighting coefficients depend on the iteration index k, and may be occasionally the empty set.

Assumption 2.2 *[Compactness] For each $i = 1, \ldots, m$, $X_i \subseteq \mathbb{R}^n$ is compact.*

Assumption 2.3 *[Interior point] The feasibility region $\bigcap_{i=1}^m X_i$ of $\mathcal{P}$ has a non-empty interior, i.e., there exists $\bar{x} \in \bigcap_{i=1}^m X_i$ and $\rho \in \mathbb{R}_+$ such that $\{x \in \mathbb{R}^n : \|x - \bar{x}\| < \rho\} \subset \bigcap_{i=1}^m X_i$.*

Due to Assumption 2.3, by the Weierstrass' theorem (Proposition A.8, p. 625 in [16]), $\mathcal{P}$ admits at least one optimal solution. Therefore, if we denote by $X^* \subseteq \bigcap_{i=1}^m X_i$ the set of optimizers of $\mathcal{P}$, then X^* is non-empty. Notice also that $f_i(\cdot)$, $i = 1, \ldots, m$, is continuous due to the convexity condition of Assumption 2.1; the addition of Assumption 2.2 is to imply Lipschitz continuity. However, $f_i(\cdot)$, $i = 1, \ldots, m$, is not required to be differentiable.

The reader should note that, differently from other approaches, we require an interior point to exists, but we do not need the agents to actually compute it, which might be impractical in a distributed set-up.

We impose the following assumption on the coefficients $\{c(k)\}_{k\geq 0}$, that appear in step 8 of Algorithm 1. This assumption is similar to those in [17, 18].

Assumption 2.4 *[Coefficient $\{c(k)\}_{k\geq 0}$] Assume that for all $k \geq 0$, $c(k) \in \mathbb{R}_+$ and $\{c(k)\}_{k\geq 0}$ is a non-increasing sequence, i.e., $c(k) \leq c(r)$ for all $k \geq r$, with $r \geq 0$. Moreover,*

1. $\sum_{k=0}^{\infty} c(k) = \infty$,
2. $\sum_{k=0}^{\infty} c(k)^2 < \infty$.

In standard proximal minimization [16] convergence is highly dependent on the appropriate choice of $c(k)$. Assumption 2.4 is in fact needed to guarantee convergence of Algorithm 1. A direct consequence of the last part of Assumption 2.4 is that $\lim_{k\to\infty} c(k) = 0$. One possible choice for $\{c(k)\}_{k\geq 0}$ satisfying Assumption 2.4 is $c(k) = \alpha/(k+1)^\beta$ for some $\alpha \in \mathbb{R}_+$ and $0.5 < \beta \leq 1$. Note that Assumption 2.4 is in a sense analogous to the conditions that the authors of [17, 18] impose on the step-size of their subgradient algorithm. It should be also noted that our set-up is synchronous, using the same $c(k)$ for all agents, at every iteration k. Extension to an asynchronous implementation is a topic for future work.

In line with [19–21] we impose the following assumptions on the information exchange between the agents.

Assumption 2.5 *[Weight coefficients] There exists $\eta \in (0, 1)$ such that for all $i, j \in \{1, \ldots, m\}$ and all $k \geq 0$, $a_j^i(k) \in \mathbb{R}_+ \cup \{0\}$, $a_i^i(k) \geq \eta$, and $a_j^i(k) > 0$ implies that $a_j^i(k) \geq \eta$. Moreover, for all $k \geq 0$,*

1. $\sum_{j=1}^m a_j^i(k) = 1$ *for all* $i = 1, \ldots, m$,
2. $\sum_{i=1}^m a_j^i(k) = 1$ *for all* $j = 1, \ldots, m$.

For each $k \geq 0$ the information exchange between the m agents can be represented by a directed graph (V, E_k), where the nodes $V = \{1, \ldots, m\}$ are the agents and the set E_k of

directed edges is given by

$$E_k = \big\{(j,i):\ a_j^i(k) > 0\big\}, \tag{2}$$

i.e., at time k agent i receives information (estimate $x_j(k)$) from agent j, and this information is weighted by $a_j^i(k)$. From (2), in the absence of communication we set $a_j^i(k) = 0$. If $a_j^i(k) > 0$ we say that j is a neighboring agent of i at time k. Under this set-up, Algorithm 1 provides a fully distributed implementation, where at iteration k each agent $i = 1, \ldots, m$ receives information only from neighboring agents.

Let $E_\infty = \big\{(j,i):\ (j,i) \in E_k \text{ for infinitely many } k\big\}$ denote the set of edges (j,i) that represent agent pairs that communicate directly infinitely often. We then impose the following connectivity and communication assumption.

Assumption 2.6 *[Connectivity and Communication] The graph (V, E_∞) is strongly connected, i.e., for any two nodes there exists a path of directed edges that connects them. Moreover, there exists $T \geq 1$ such that for every $(j,i) \in E_\infty$, agent i receives information from a neighboring agent j at least once every consecutive T iterations.*

Assumption 2.6 guarantees that any pair of agents communicates directly infinitely often, and the intercommunication interval is bounded.

The interpretation of having a uniform lower bound η, independent of k, for the non-zero coefficients $a_j^i(k)$ in Assumption 2.5 is that it ensures that each agent is mixing information received by other agents at a non-diminishing rate in time [17]. Moreover, Conditions 1 and 2 in Assumption 2.5 ensure that this mixing is a convex combination of the other agent estimates, assigning a non-zero weight to its local one due to the fact that $a_i^i(k) \geq \eta$.

For further details on the interpretation of the imposed network structure the reader is referred to [17, 22].

Assumptions 2.5 and 2.6 are identical to Assumptions 2-5 in [17] (the same assumptions are also imposed in [18]), but were reported also here to ease the reader and facilitate the exposition of our results. Note that these are rather standard for distributed optimization and consensus problems; for possible relaxations the reader is referred to [21, 23].

Remark 2.1 (weights computation) *Satisfying Assumption 2.5 requires agents to agree on an infinite sequence of doubly stochastic matrices (double stochasticity arises due to conditions 1 and 2 in Assumption 2.5), where $a_j^i(k)$ would be element (i,j) of the matrix at iteration k. This agreement could be performed prior to the execution of the algorithm in a centralized manner, and the resulting matrices communicated to all agents via some consensus*

scheme; this is standard in distributed optimization algorithms of this type (see also [17, 18, 21]). Alternatively, in the symmetric case when $a_j^i(k) = a_i^j(k)$, $i, j = 1, \ldots, m$, $k \geq 0$, *the distributed algorithm in [24] can be adopted. This requires each pair* (i, j) *of agents that can communicate over a bi-directional link to exchange their planned weighting coefficients* $\hat{a}_j^i(k)$ *and* $\hat{a}_i^j(k)$ *together their solutions* $x_i(k)$ *and* $x_j(k)$ *and then setting* $a_j^i(k) = a_i^j(k) = \min\{\hat{a}_j^i(k), \hat{a}_i^j(k)\}$.

Algorithm 1 terminates if the iterates maintained by all agents converge. From an implementation point of view, agent i, $i = 1, \ldots, m$, will terminate its update process if the absolute difference (relative difference could also be employed) between two consecutive iterates $\|x_i(k+1) - x_i(k)\|$ keeps below some user-defined tolerance for a number of iterations equal to T (see Assumption 2.6) times the diameter of the graph (that is, the greatest distance between any pair of nodes that are connected via an edge in E_∞). This is the worst case number of iterations required for an agent to communicate with all other agents in the network; note that if an agent terminated the process at the first iteration where the desired tolerance is met, then convergence would not be guaranteed since its tentative solution may still change as an effect of other agents updating their solutions.

The proposed iterative methodology resembles the structure of proximal minimization for constrained convex optimization [16, Chapter 3.4.3]. The difference, however, is that our set-up is distributed and the quadratic term in step 8 does not penalize the deviation of x_i from the previous iterate $x_i(k)$, but from an appropriately weighted average $z_i(k)$. Note that, in contrast with the inspiring work in [17, 18, 25] addressing $\mathcal{P}$ under a similar set-up but following a projected subgradient approach, our proximal minimization-based approach allows for an intuitive economic interpretation: at every iteration k we penalize a consensus residual proxy by the time-varying coefficient $1/(2c(k))$, which, due to Assumption 2.4, progressively increases. This can be thought of as a pricing settling mechanism, where the more we delay to achieve consensus the higher the price is.

In the case where $a_j^i(k) = 1/m$ for all $i, j = 1, \ldots, m$, for all $k \geq 0$, that corresponds to a decentralized control paradigm, the solution of our proximal minimization approach coincides with the one obtained when the alternating direction of multipliers [16], [26], is applied to this problem (see eq. (4.72)-(4.74), p. 254 in [16]). In the latter the quadratic penalty term is not added to the local objective function as in step 8 of Algorithm 1, but to the Lagrangian function of an equivalent problem, and the coefficient $c(k)$ is an arbitrary constant independent of k; however, a dual-update step is required. Formal connections between penalty methods and the method of multipliers have been established in [27].

Remark 2.2 (Application to a specific problem structure) *Algorithm 1 can be sim-*

plified when the underlying optimization problem exhibits a specific structure, namely agents need to agree on a common decision vector $y \in \mathbb{R}^{\bar{n}}$, but each of them decides upon a local decision vector $u_i \in \mathbb{R}^{n_i}$, $i = 1, \ldots, m$ as well:

$$\begin{aligned} &\min_{y \in \mathbb{R}^{\bar{n}}, \{u_i \in \mathbb{R}^{n_i}\}_{i=1}^m} \sum_{i=1}^m f_i(y, u_i) \\ &\text{subject to } y \in \bigcap_{i=1}^m Y_i, \ u_i \in U_i, \ i = 1, \ldots, m, \end{aligned} \tag{3}$$

where $Y_i \in \mathbb{R}^{\bar{n}}$ and $U_i \subseteq \mathbb{R}^{n_i}$, for all $i = 1, \ldots, m$. Provided that Assumptions 2.1-2.6 hold for problem (3) with $x = (y, u_1, \ldots, u_m)$ and $X_i = Y_i \times \mathbb{R}^{n_1} \times \cdots \times U_i \times \cdots \times \mathbb{R}^{n_m}$, we can rewrite it as $\min_{y \in \mathbb{R}^{\bar{n}}} \sum_{i=1}^m g_i(y)$ subject to $y \in \bigcap_{i=1}^m Y_i$, where $g_i(y) = \min_{u_i \in U_i} f_i(y, u_i)$ and simplify Algorithm 1 by replacing steps 7-8 with:

$$\begin{aligned} &z_i(k) = \sum_{j=1}^m a_j^i(k) y_j(k), \\ &\big(y_i(k+1), u_i(k+1)\big) = \arg \min_{y_i \in Y_i, u_i \in U_i} f_i(y_i, u_i) + \frac{1}{2c(k)} \|z_i(k) - y_i\|^2. \end{aligned}$$

This entails that agents only need to communicate their local estimates $y_i(k)$, $i = 1, \ldots, m$, of the common decision vector y while the local solutions related to u_i, $i = 1, \ldots, m$, need not be exchanged.

Under the structural assumptions and the communication set-up described above, Algorithm 1 converges and agents reach consensus, in the sense that their local estimates $x_i(k)$, $i = 1, \ldots, m$, converge to some minimizer of problem $\mathcal{P}$. This is formally stated in the following theorem, which constitutes the main contribution of our work in [3, 4].

Theorem 2.1 (Optimality) *Consider Assumptions 2.1-2.6 and Algorithm 1. We have that, for some minimizer $x^* \in X^*$,*

$$\lim_{k \to \infty} \|x_i(k) - x^*\| = 0, \ \textit{for all } i = 1, \ldots, m. \tag{4}$$

As a concluding remark, note that we provided a proximal minimization perspective to the problem of distributed optimization over time-varying networks and in the presence of a possibly different constraint set per agent. Proximal minimization serves as an alternative to gradient methods, where, instead of a gradient (subgradient) step, a penalty term (proxy) is introduced in the objective function of each agents' local decision problem. As observed in [28] with reference to incremental algorithms, the proximal minimization approach leads to numerically more stable algorithms compared to their gradient-based counterparts. This will

be shown in the numerical examples of Section 2.4.1 where Algorithm 1 is applied to power control in cellular networks.

2.3 Coupling via constraints

In this section, we address a specific class of convex optimization problems over time-varying, multi-agent networks, where each agent has its own decision vector, objective function, and constraint set, and is coupled to the others via a constraint expressed as the non-positivity of the sum of convex functions, each function corresponding to one agent.

More precisely, we consider the following optimization program

$$\begin{aligned} \mathcal{P}: \quad \min_{\{x_i \in X_i\}_{i=1}^m} \quad & \sum_{i=1}^m f_i(x_i) \\ \text{subject to:} \quad & \sum_{i=1}^m g_i(x_i) \leq 0, \end{aligned} \tag{5}$$

involving m agents that communicate over a time-varying network. $x_i \in \mathbb{R}^{n_i}$ is the decision vector of agent i, $X_i \subseteq \mathbb{R}^{n_i}$ its local constraint set, and $f_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}$ its objective function. Each agent i, $i = 1, \ldots, m$, is contributing to the coupling constraint $\sum_{i=1}^m g_i(x_i) \leq 0$ via function $g_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^p$. Note that equality linear coupling constraints can be also dealt with by means of $\mathcal{P}$, by means of double-sided inequalities.

We propose a distributed iterative scheme based on a combination of dual decomposition and proximal minimization. Under convexity assumptions and suitable connectivity properties of the communication network, agents reach consensus with respect to the dual variables, without disclosing information about their optimal decision, local objective and constraint functions, nor about the function encoding their contribution to the coupling constraint. The proposed algorithm converges to some optimal dual solution of the centralized problem counterpart, while for the primal variables, we show convergence to the set of optimal primal solutions.

Problem $\mathcal{P}$ could be solved, in principle, in a centralized fashion. However, if the number m of agents is large, this may turn out to be computationally prohibitive. In addition, each agent would be required to share its own information (coded via $f_i(\cdot)$, X_i, and $g_i(\cdot)$) either with the other agents or with a central unit collecting all information, which may be undesirable in some cases, due to privacy issues.

Remark 2.3 (alternative formulation) *Note that the approaches to distributed optimization in [17, 22, 25, 29] and also the one presented in Section 2.2 (see [3, 4]) can be applied to inequality-coupled problems by introducing a common decision vector collecting all local*

decision variables. This, however, immediately leads to the following two main drawbacks:

1. *an increased computational and communication effort that scales as the number of agents in the network since each agent has to generate local copies of the optimization variables of all the other agents, which then are optimized and exchanged. In the approach in this section instead agents need to optimize the local variables only and exchange the estimate of the dual variables, which are as many as the number of coupling constraints. The required local computational effort is thus much smaller. As for the communication effort, the proposed approach is particularly appealing when the number of coupling constraints is low compared to the overall dimensionality of primal decision variables;*

2. *privacy issues since the agents local information on the primal problem (namely, the value of the local optimization variables, the local objective function, the local constraints, and the contribution of the agent to the coupling constraint) should be exchanged and they may represent sensitive data. In the algorithm proposed in this section, only the local estimates of the dual variables are exchanged, and this secures maximum privacy among agents.*

We next formulate a distributed strategy that overcomes both the privacy and computational issues outlined above by resorting to the dual of (5).

Let us consider the Lagrangian function $L(x, \lambda) : \mathbb{R}^n \times \mathbb{R}^p_+ \to \mathbb{R}$ given by

$$L(x, \lambda) = \sum_{i=1}^{m} L_i(x_i, \lambda) = \sum_{i=1}^{m} \Big\{ f_i(x_i) + \lambda^\top g_i(x_i) \Big\},$$

where $x = [x_1{}^\top \cdots x_m{}^\top]^\top \in X = X_1 \times \cdots \times X_m \subseteq \mathbb{R}^n$, with $n = \sum_{i=1}^m n_i$, whereas $\lambda \in \mathbb{R}^p_+$ is the vector of Lagrange multipliers ($\mathbb{R}^p_+$ denotes the p-th dimensional non-negative orthant; in the sequel we shall sometimes write $\lambda \geq 0$ in place of $\lambda \in \mathbb{R}^p_+$).

Correspondingly, we can define the dual function as

$$\varphi(\lambda) = \min_{x \in X} L(x, \lambda), \tag{6}$$

which, due to the separable structure of objective and constraint functions in problem $\mathcal{P}$ (see (5)), can be expressed as

$$\varphi(\lambda) = \sum_{i=1}^{m} \varphi_i(\lambda) = \sum_{i=1}^{m} \min_{x_i \in X_i} L_i(x_i, \lambda), \tag{7}$$

where each $\varphi_i(\cdot)$ is a concave function representing the dual function of agent i.

Algorithm 2 Distributed algorithm

1: **Initialization**
2: $k = 0$.
3: Consider $\hat{x}_i(0) \in X_i$, for all $i = 1, \dots, m$.
4: Consider $\lambda_i(0) \in \mathbb{R}^p_+$, for all $i = 1, \dots, m$.
5: **For** $i = 1, \dots, m$ **repeat until convergence**
6: $\ell_i(k) = \sum_{j=1}^m a_j^i(k)\lambda_j(k)$.
7: $x_i(k+1) \in \arg\min_{x_i \in X_i} f_i(x_i) + \ell_i(k)^\top g_i(x_i)$.
8: $\lambda_i(k+1) = \arg\max_{\lambda_i \geq 0} \left\{ g_i(x_i(k+1))^\top \lambda_i - \frac{1}{2c(k)} \|\lambda_i - \ell_i(k)\|^2 \right\}$
9: $\hat{x}_i(k+1) = \hat{x}_i(k) + \frac{c(k)}{\sum_{r=0}^k c(r)} (x_i(k+1) - \hat{x}_i(k))$.
10: $k \leftarrow k+1$.

Given these definitions, the dual of problem $\mathcal{P}$ in (5) can be expressed as:

$$\mathcal{D}: \quad \max_{\lambda \geq 0} \min_{x \in X} L(x, \lambda),$$

or, equivalently, as

$$\mathcal{D}: \quad \max_{\lambda \geq 0} \sum_{i=1}^m \varphi_i(\lambda). \tag{8}$$

The coupling between agents is given in (8) by the fact that λ is a common decision vector and the agents should agree on its value.

Algorithm 2 is a distributed iterative scheme that aims at reconstructing the solution to both the dual problem (8) and the primal problem (5) by exchanging a minimal amount of information among agents. Its steps are explained hereafter.

Each agent i, $i = 1, \dots, m$, initializes the estimate of its local decision vector with $\hat{x}_i(0) \in X_i$ (step 3 of Algorithm 2), and the estimate of the common dual variables vector with a $\lambda_i(0) \in \mathbb{R}^p_+$ that is feasible for problem $\mathcal{D}$ (step 4 of Algorithm 2). A sensible choice is to set $\hat{x}_i(0) \in \arg\min_{x_i \in X_i} f_i(x_i)$, and $\lambda_i(0) = 0$, $i = 1, \dots, m$, which corresponds to the solution of problem (5) when coupling constraints are neglected.

At every iteration k, $k \geq 1$, each agent i computes a weighted average $\ell_i(k)$ of the dual variables vector based on the estimates $\lambda_j(k)$, $j = 1, \dots, m$, of the other agents and its own estimate (step 6). The weight $a_j^i(k)$ that agent i attributes to the estimate of agent j at iteration k is set equal to zero if agent i does not communicate with agent j at iteration k.

Then, Algorithm 2 alternates between a primal and a dual update step (step 7 and step 8, respectively). In particular, step 7 performs an update of the local primal vector $x_i(k+1)$ by minimizing L_i evaluated at $\lambda = \ell_i(k)$ as in dual decomposition, whereas, differently from dual decomposition, which would consists of the maximization of L_i evaluated at $x_i = x_i(k+1)$, the update of the dual vector in step 8 involves also the proximal term $-\frac{1}{2c(k)} \|\lambda_i - \ell_i(k)\|^2$ to

foster consensus among the agents.

Steps 7 and 8 can be thought of as an approximation of the following proximal maximization step

$$\lambda_i(k+1) = \arg\max_{\lambda_i \geq 0} \min_{x_i \in X_i} \Big\{ L_i(x_i, \lambda_i) - \frac{1}{2c(k)} \|\lambda_i - \ell_i(k)\|^2 \Big\}, \tag{9}$$

which would implement the distributed algorithm of [4] for the dual problem (8). Steps 7 and 8 are however preferred to (9) since the resolution of the $\max - \min$ program in (9) is very hard in general. Moreover, it is perhaps worth mentioning at the outset that step 8 in Algorithm 2 is equivalent to a projected subgradient step. Indeed the constrained maximization of a quadratic function in step 8 can be explicitly solved, leading to

$$\lambda_i(k+1) = [\ell_i(k) + c(k) g_i(x_i(k+1))]_+, \tag{10}$$

where $[\,\cdot\,]_+$ denotes the projection of its argument onto $\mathbb{R}^p_+$. Then, it can be shown that $g_i(x_i(k+1))$ is a subgradient of the dual function $\varphi_i(\cdot)$ evaluated at $\ell_i(k)$ (see the proof of Theorem 2.2 for more details), while $c(k)$ can be thought of as the subgradient step-size. Hence, steps 7 and 8 can be also seen as an application of the distributed subgradient algorithm of [17], which was originally developed for primal problems though, to the dual problem (8).

Unfortunately, the local primal vector $x_i(k)$ does not converge to the optimal solution x_i^* to (5) in general. Therefore, the auxiliary primal iterates $\hat{x}_i(k+1)$, defined as the weighted running average of $\{x_i(r+1)\}_{r=0}^k$

$$\hat{x}_i(k+1) = \frac{\sum_{r=0}^k c(r) x_i(r+1)}{\sum_{r=0}^k c(r)}, \tag{11}$$

is computed in step 9 of Algorithm 2 in a recursive fashion. Such an auxiliary variable shows better convergence properties as compared to $x_i(k)$, and is often constructed in the so-called primal recovery procedure of dual decomposition methods, [18, 30, 31].

Note that in Algorithm 2 no local information related to the primal is exchanged between the agents (as a matter of fact only the estimates of the dual vector are communicated) so that our algorithm is well suited to account for privacy requirements.

The proposed distributed algorithm shows properties of convergence and optimality, which hold under the following assumptions on the structure of the problem and on the same communication features of the time-varying multi-agent network as in Section 2.2.

Assumption 2.7 *[Convexity] For each $i = 1, \ldots, m$, the function $f_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}$ and each*

component of $g_i(\cdot) : \mathbb{R}^{n_i} \to \mathbb{R}^p$ *are convex; for each* $i = 1, \dots, m$ *the set* $X_i \subseteq \mathbb{R}^{n_i}$ *is convex.*

Assumption 2.8 *[Compactness] For each* $i = 1, \dots, m$*, the set* $X_i \subseteq \mathbb{R}^{n_i}$ *is compact.*

Assumption 2.9 *[Interior point] There exists* $\tilde{x} = [\tilde{x}_1 \cdots \tilde{x}_m]^\top \in \text{relint}(X)$*, where* $\text{relint}(X)$ *is the relative interior of the set* X*, such that* $\sum_{i=1}^m g_i(\tilde{x}_i) \leq 0$ *for those components of* $\sum_{i=1}^m g_i(x_i)$ *that are linear in* x*, if any, while* $\sum_{i=1}^m g_i(\tilde{x}_i) < 0$ *for all other components.*

The reader should note that as in Section 2.2, we require an interior point to exists, but we do not need the agents to actually compute it.

As a consequence of Assumptions 2.7-2.9, we have that strong duality holds and an optimal primal-dual pair $(x^\star, \lambda^\star)$ exists, where $x^\star = [x_1^\star \cdots x_m^\star]^\top$. In the following we will denote by $X^\star$ the set of all primal minimizers, and by $\Lambda^\star$ the set of all dual maximizers.

The time-varying coefficient $c(k)$ satisfy Assumption 2.4. The communication network is required to satisfy the connectivity and communications conditions of the previous section, specified in Assumptions 2.5 and 2.6.

If Assumptions 2.4-2.9 are satisfied, then Algorithm 2 converges and agents agree to a common vector of Lagrange multipliers. Specifically, their local estimates $\lambda_i(k)$ converge to some optimal vector of Lagrange multipliers, while the vector $\hat{x}(k) = [\hat{x}_1(k)^\top \cdots \hat{x}_m(k)^\top]^\top$ approaches $X^\star$, the set of minimizers of the primal problem.

These results are formally stated in the following theorems, whose proofs can be found in [6].

Theorem 2.2 *[Dual Optimality] Under Assumptions 2.4-2.9, there exists a* $\lambda^\star \in \Lambda^\star$ *such that*

$$\lim_{k\to\infty} \|\lambda_i(k) - \lambda^\star\| = 0, \text{ for all } i = 1, \dots, m. \tag{12}$$

Theorem 2.3 *[Primal Optimality] Under Assumptions 2.4-2.9, we have that*

$$\lim_{k\to\infty} \text{dist}(\hat{x}(k), X^\star) = 0, \tag{13}$$

where $\text{dist}(y, Z)$ *denotes the distance between* y *and the set* Z*, i.e.,* $\text{dist}(y, Z) = \min_{z\in Z} \|y - z\|$.

Remark 2.4 (network breaking into not-connected sub-networks) *If communication failures cause the network to break into not-connected components, then a feasible, though sub-optimal, solution where each sub-network optimizes its own a-priori assigned share of resource can be obtained by integrating the distributed algorithm presented in [24] (and recalled in Remark 2.1) for recomputing the weights in the average* $\ell_i(k)$ *of the dual variables. Indeed,*

each sub-network will end up solving a problem like the original one in (5) *but where the sum of the objective functions* $f_i(\cdot)$ *and constraint functions* $g_i(\cdot)$ *is confined to the set of agents in the sub-network.*

2.4 Numerical examples

2.4.1 Power control in cellular networks

We now address the problem of power control in cellular networks, which is an energy management problem like the one addressed in the smart grid use case of work package 5, the main difference being that here power consumption of different interfering wireless devices is set so to optimize the transmission quality, whereas in the smart grid context it is set so as to minimize costs while satisfying the load request.

Consider a wireless cellular network where each cell is associated with a base station, and base stations exchange data via wired communication (see Figure 1 for a pictorial view).

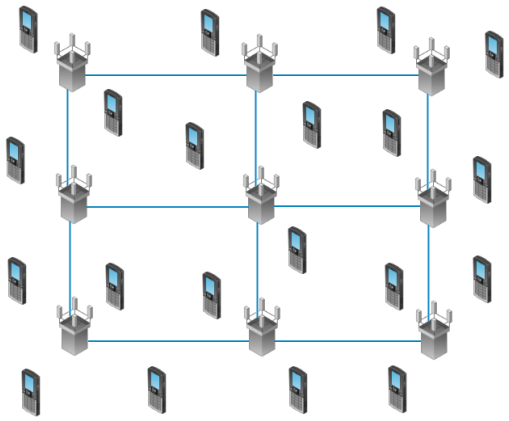

Figure 1: Example of a cellular network.

Mobile users adopt Code Division Multiple Access (CDMA, [32]) to communicate on the same channel with the base station in their cell. If the same communication channel is used in all cells, then, interference occurs among all mobile users in the network, thus causing a decrease of the quality of transmission as measured by the Signal to Interference plus Noise Ratio (SINR). Consequently, the throughput of the device can deteriorate, since, as the SINR decreases, the number of retransmissions increases, with a reduction of the effective data transmission rate.

An adjustment of the transmission power is then needed to guarantee a certain transmission rate and throughput. Due to the wireless channel resource sharing, this leads to a network-wide constrained optimization problem where the transmission powers of all devices in the cellular

network have to be jointly set so as to maximize the sum of their throughput while not exceeding the power transmission capabilities of each one of them. The solution is far from being trivial like, e.g., "use the maximum transmission power per device" because this would have the positive effect of increasing the signal component but, at the same time, it would also increase the interference.

Solving the problem in a centralized way can be computationally challenging and also requires the transmission of much information among base stations, possibly overloading the wired communication network. Most of the approaches proposed in the literature are actually based on a distributed scheme and they can be classified into two main categories, i.e., autonomous and non-autonomous, the distinguishing feature being that, while in the former communications occur only between base station and mobile users, in the latter base stations collaborate and exchange information.

The approach described in [33] and further studied in [34], belongs to the first class and aims at minimizing the total transmission power subject to constraints on the SINR via a distributed iterative algorithm. At every iteration, each mobile user sets its transmission power at a certain level, communicates using that power level, and then refines it based on the information on the SINR provided by its reference base station. Autonomous distributed approaches to power control based on games involving non-cooperative users are proposed in [35] and [36]. A further autonomous approach in the literature consists in formulating power control as an open loop global optimization problem where the SINR is not a constraint but has to be optimized via a distributed scheme, [37, 38].

The solution of the power control problem in a distributed non-autonomous fashion involves communication among neighboring base stations and information sharing on their local solutions. The involved additional communication overhead is not an issue though, since, typically, base stations are connected via a wired backbone. In turn, the exchange of information between base stations leads to a faster optimization and to a more robust scheme.

We here apply Algorithm 1 as a distributed non-autonomous algorithm for power control in a single channel wireless cellular network. In [39], the same problem is addressed via sub-gradient based distributed optimization, which is adapted here to our modeling set-up where possibly multiple mobile users are linked to the same base station. A comparative analysis via an extensive simulation study is performed between the two methods, and reveals the advantage of proximal minimization in the case when the cost function is non-differentiable and the sub-gradient has to be computed.

We consider a cellular network with m mobile users MU_j, $j = 1, \ldots, m$, served by n base stations BS_i, $i = 1, \ldots, n$. A common wireless channel is used for data exchange between

mobile users and base stations, so that each mobile user introduces some interference in the data exchange of the others with their reference base station. The channel is static, and the number of mobile users is assumed constant in the time scale of interest. We suppose that each base station, say BS_i, has an accurate estimate of the amplitude gains (called also channel coefficients) $h_{i,j}$ of the links from all mobile users MU_j, $j = 1, 2, \dots m$. In particular, if MU_j is in a cell that is not an immediate neighbor of BS_i, then, it causes negligible interference and $h_{i,j}$ will be close to zero. If MU_j is either in cell i or in an immediate neighboring cell, BS_i can estimate the channel coefficient from pilot signals that are sent by the mobile user MU_j. Each mobile user MU_j is served by a single base station and exchange data with it via wireless communication using a transmission power p_j.

Let us consider base station BS_i. Denote with $\mathcal{J}_i \subset \{1, 2, \dots, m\}$ be the set of indices of the mobile users that communicate with BS_i. We can then compute the SINR of MU_j with $j \in \mathcal{J}_i$ at its reference base station BS_i as follows:

$$\varrho_j(p) = \frac{p_j h_{i,j}^2}{\sigma_i^2 + \sum_{s \neq j} p_s h_{i,s}^2}, \tag{14}$$

where $p = \begin{bmatrix} p_1 & p_2 & \dots & p_m \end{bmatrix}^\top$ is the transmission power vector, $h_i = \begin{bmatrix} h_{i,1} & h_{i,2} & \dots & h_{i,m} \end{bmatrix}^\top$ is the vector containing the channel coefficients of all communication uplinks from MU_j, $j = 1, \dots, m$ to BS_i, and σ_i^2 is the receiver noise variance at the base station BS_i. Shadow fading can also be included by introducing a re-scaling lognormal distributed factor in the channel coefficient along each uplink, [40, 41].

The SINR ϱ_j in (14) is strongly affecting the quality of the transmission, and, in particular, the throughput achieved by the mobile user MU_j, which can be modeled (see e.g. [42]) as proportional to $\mathcal{U}_j(p) = \log(1 + \eta \varrho_j(p))$ where η is a constant that depends on the modulation scheme. The expression above further simplifies to $\mathcal{U}_j(p) \approx \log(\eta \varrho_j(p))$ in high SINR regime. This implies that some minimum transmission power $p_{\min} > 0$ should be adopted on each link.

From the perspective of base station BS_i, a certain quality of transmission should be guaranteed to all mobile users that BS_i serves, while avoiding too costly use of power. This can be achieved by choosing p so as to maximize the worst performance index

$$\min_{j \in \mathcal{J}_i} \{\log(\varrho_j(p)) - \mathcal{V}_j(p)\}, \tag{15}$$

where $\mathcal{V}_j(\cdot)$ is a convex and differentiable function that represents the power cost for the mobile user MU_j and is then increasing as a function of p_j. The contribution of η to the cost can be neglected since it does not affect the solution.

Note that the above performance index depends on the transmission power of all mobile users in the network and not only on those that are served by BS_i. This entails that all base stations need to cooperate to guarantee the best minimum quality of the transmission for all users by jointly solving the following constrained optimization problem

$$\max_{p} \sum_{i=1}^{n} \min_{j \in \mathcal{J}_i} \{\log(\varrho_j(p)) - \mathcal{V}_j(p)\} \tag{16}$$
$$\text{subject to: } p_{\min} \leq p_j \leq p_{\max},\ j = 1, \ldots, m,$$

where $p_{\max} > p_{\min}$ denotes a maximum transmission power and is assumed to be given.

We next propose a distributed approach to the solution of (16) that does not require the introduction of any central unit, not for the base stations to share the channel coefficients of their mobile users, which are kept as a sensitive information. To this purpose, we reformulate the problem so that it fits the framework of the distributed Algorithm 1 for solving convex optimization problems with separable cost and local constraints on a global decision vector.

Let us express vector p of the mobile users transmission powers as a function of vector $x = \begin{bmatrix} x_1 & \ldots & x_m \end{bmatrix}^\top$ through the change of variables $p = e^x$, where $e^x = \begin{bmatrix} e^{x_1} & e^{x_2} & \ldots & e^{x_m} \end{bmatrix}^\top$.

We then obtain the following reformulation of (16)

$$\min_{x \in X} \sum_{i=1}^{n} f_i(x) \tag{17}$$

where the cost function f_i is given by

$$f_i(x) = \max_{j \in \mathcal{J}_i} J_{ij}(x) \tag{18}$$

with $J_{ij}(x) = \log\left(\sigma_i^2 h_{i,j}^{-2} e^{-x_j} + \sum_{s \neq j} h_{i,s}^2 h_{i,j}^{-2} e^{x_s - x_j}\right) + \mathcal{V}_j(e^x)$ and the constraint set is

$$X = \{x : \log(p_{\min}) \leq x_j \leq \log(p_{\max}), j = 1, \ldots, m\}.$$

If we let $x^\star$ denote a solution to (17), then, the optimal transmission power vector $p^\star$ can be recovered as $p^\star = \log(x^\star)$ where $\log(\cdot)$ applied to a vector should be interpreted as the log function applied to each component of the vector, i.e., $p^\star = \begin{bmatrix} \log(x_1^\star) & \log(x_2^\star) & \ldots & \log(x_m^\star) \end{bmatrix}^\top$. This finally leads to Algorithm 3.

Remark 2.5 *Note that function $J_{ij}(x)$ is convex and differentiable in x since it is the sum of two terms: the* log *of the sum of exponentials in x and the convex and differentiable function $\mathcal{V}_j(e^x)$. Then, $f_i(\cdot)$ is convex since it is the maximum of convex functions. In the case when*

there are multiple mobile users served by the base station i, $f_i(\cdot)$ is not guaranteed to be differentiable. Also, $f_i(\cdot)$ is known only to the base station BS_i because it depends on the channel coefficients h_i. Note finally that set X is convex, nonempty ($p_{\min} < p_{\max}$), and also compact since $p_{\min} > 0$, which guarantees that the set $X^\star$ of minimizers of (17) is non-empty (see the interior point Assuption 2.3).

Algorithm 3 Distributed power control proximal algorithm

1: **Initialization**
2: $k = 0$
3: Consider $x^i(0) \in X$, for all $i = 1, ..., n$
4: **For** $i = 1, ..., n$ **repeat until convergence**
5: $\hat{x}^i(k) = \sum_{j=1}^{n} a_j^i x^j(k)$
6: $x^i(k+1) = \arg\min_{x^i \in X} \left\{ f_i(x^i) + \frac{1}{2c(k)} \|x^i - \hat{x}^i(k)\|^2 \right\}$
7: $p^i(k) = \log(x^i(k))$
8: $k \leftarrow k + 1$

The weights $a_j^i \geq 0$, $j = 1, \ldots, n$, at step 5 of Algorithm 3 encode the wired network (directed) communication graph, in that if $a_j^i = 0$, then, BS_i does not receive any information from BS_j. Those base stations BS_j such that $a_j^i > 0$ are the neighbors of BS_i.

Initially, each base station BS_i, $i = 1, \ldots, n$, makes its own guess on some tentative value $x^i(0)$ for the solution to (17) (step 3, Algorithm 3). One possible choice is $x^i(0) \in \arg\min_{x^i \in X} f_i(x^i)$. At iteration k, each BS_i constructs a weighted average $\hat{x}^i(k)$ of the tentative solutions $x^j(k)$, $j = 1, \ldots, n$, received by its neighbors and its local one (step 5, Algorithm 3). Step 6 of Algorithm 3 is a proximal minimization computation, where BS_i solves a local minimization problem, updating its tentative solution with a value within X that minimizes the sum of its local objective function $f_i(\cdot)$ and a weighted quadratic term, which accounts for the difference of its current tentative solution from the average $\hat{x}_i(k)$. The relative importance of the two terms is determined by $c(k) \in \mathbb{R}_+$. Note that since $f_i(\cdot)$ is convex (see Remark 2.5) and the quadratic penalty term is strictly convex, the resulting minimization problem admits a unique solution at each iteration k.

We next present some assumptions that are needed for Algorithm 3 to converge to an optimal solution $p^\star$ of the constrained optimization problem (16).

Assumption 2.10 (Connectivity and Communication) *Let (V, E) be the directed graph with nodes $V = \{1, \ldots, n\}$ and edges $E = \{(j, i) : a_j^i > 0\}$. We assume that $(V; E)$ is strongly connected, i.e., for any two nodes there exists a path of directed edges that connects them.*

Coefficients $\{c(k)\}_{k \geq 0}$ in step 6 of Algorithm 3 are chosen so as to satisfy Assumption 2.4,

which is the same assumption imposed in [39] on the step-size of their subgradient algorithm. The weights a_j^i, $i, j = 1, \ldots, n$, of the average tentative solution satisfy Assumption 2.5, where we recall that the last two conditions imply that the matrix with element (i, j) equal to a_j^i is doubly stochastic.

Theorem 2.4 (Optimality) *Consider Assumptions 2.10-2.5 and Algorithm 3. We have that for some maximizer $p^\star$ of problem* (16)*, all base stations reach consensus to $p^\star$, i.e.,*

$$\lim_{k\to\infty} \|p^i(k) - p^\star\| = 0,\ i = 1, \ldots, n.$$

Proof 1 *Given the convexity of $f_i(\cdot)$ and the convexity and compactness of X (see Remark 2.5), by applying Theorem 2.1, we get that for some minimizer $x^\star$ of problem* (17) *the following convergence property holds*

$$\lim_{k\to\infty} \|x^i(k) - x^\star\| = 0,\ i = 1, \ldots, n.$$

Since the mapping $p^i(k) = \log(x^i(k))$ from x^i to p^i is a monotonically increasing bijective function and the cost that is minimized in (17) *is obtained by multiplying by -1 the performance index that is maximized in* (16)*, by defining $p^\star = \log(x^\star)$ the statement in the theorem follows immediately.*

Remark 2.6 (Resilience to failures) *The optimality result in Theorem 2.4 is preserved when temporary failure of communication links occurs. This is because the asymptotic optimality result in Theorem 2.1 holds with time-varying weight coefficients, under suitable long run connectivity conditions as specified in Assuption 2.6. Also, if one of the base stations definitely breaks down, its mobile users can be re-assigned to the closest base station, and the system will automatically adapt to the new configuration, if the distributed algorithm for transmission power control adjusts the weight coefficients so as to comply with Assumption 2.5.*

We next show the performance of the proposed distributed non-autonomous algorithm for power control in a single channel wireless cellular network, and compare it with the sub-gradient based alternative proposed in [39], which is here presented in Algorithm 4 to comply with our notation.

In Algorithm 4, $\Pi_X[\cdot]$ denotes the projection onto set X (which is straightforward since X is a box), and $\nabla f_i(\cdot)$ the sub-gradient of function $f_i(\cdot)$, which satisfies

$$\nabla f_i(x)^\top (y - x) \leq f_i(y) - f_i(x),\ x, y \in X.$$

Algorithm 4 Distributed power control sub-gradient algorithm

1: **Initialization**
2: $k = 0$
3: Consider $x^i(0) \in X$, for all $i = 1, ..., n$
4: For $i = 1, ..., n$ **repeat until convergence**
5: $\hat{x}^i(k) = \sum_{j=1}^{n} a_j^i x^j(k)$
6: $x^i(k+1) = \Pi_X[\hat{x}^i(k) - c(k)\nabla f_i(\hat{x}^i(k))]$
7: $p^i(k) = \log(x^i(k))$
8: $k \leftarrow k + 1$

Given that $f_i(\cdot)$ in (18) is convex (see Remark 2.5), the sub-gradient at $x \in X$ is well-defined. However, it is not differentiable in the case when the base station is serving multiple users. The sub-gradient computation involves the choice of the mobile user with the worst throughput at every iteration, given the current value x of the iterated weighted average $\hat{x}^i$. More specifically, we have that

$$\nabla f_i(x) = \nabla \bar{J}_i(x), \tag{19}$$

where $\nabla \bar{J}_i(\cdot)$ is the standard gradient of function

$$\bar{J}_i(\cdot) = J_{ij_x}(\cdot) \text{ with } j_x = \arg\max_{j \in \mathcal{J}_i} J_{ij}(x), \tag{20}$$

which is convex and differentiable.

The same result in Theorem 2.4 holds for Algorithm 4 (see [39] for a proof). The two algorithms work in the same set-up and involve the same communication structure, both avoiding local information sharing. The main difference is how the tentative solution is updated at every iteration, and, in particular, the methods used by each base station to solve its local optimization problem.

A clear advantage of Algorithm 4 compared with Algorithm 3 proposed in this paper is that it is computationally less demanding since it does not involve any optimization over X. However, if we consider the multi-user case that is indeed the most common in practice, Algorithm 4 is typically affected by oscillations of the solution before convergence is reached. This is caused by the non-differentiability of $f_i(\cdot)$ and the fact that the sub-gradient calculation in (19) involves the identification of the mobile user with the worst throughput given the current average tentative solution (see (20)).

In the simulation results, we consider cellular networks with n base stations that are located on a regularly gridded square area with grid parameter equal to 5. Each base station

Algorithm 5 Sinkhorn-Knopp algorithm

1: $a_j^i \leftarrow 1$ if $(i,j) \in E$
2: $a_j^i \leftarrow 0$ if $(i,j) \notin E$
3: **repeat**
4: $\quad a_j^i \leftarrow a_j^i / \sum_{c=1}^m a_c^i$, $\forall i = 1, \ldots, m$
5: $\quad a_j^i \leftarrow a_j^i / \sum_{r=1}^m a_j^r$, $\forall j = 1, \ldots, m$
6: **until** Assumption 2.5 is satisfied within a given tolerance

is at the vertex of one or multiple squares and is wired connected to all the base stations that belong to its same squares, either on the same edge and or on the diagonal. Each mobile user is served by the closest base station.

The channel coefficient $h_{i,j}$ of the communication link from the mobile user j to the base station i is set to zero if their distance is larger than or equal to 200, otherwise it decays to zero as the fourth power of the distance. A shadow fading factor modeled as lognormal with mean 1.05 and variance 0.1 is introduced along each uplink. The receiver noise variance is set equal to 10^{-4} at each base station. We set $\log(p_{\min}) = 0$ and $\log(p_{\max}) = 7$ when defining the set X. The term penalizing the transmission power cost in (15) is given by $\mathcal{V}_j(p) = 10^{-3} p_j$.

Parameter $\{c(k)\}_{k\geq 0}$ appearing in both algorithms is set equal to $c(k) = \frac{\alpha}{(k+1)^\beta}$, where $\alpha = n$ and $\beta = 0.7$. The initial value for x^i, $i = 1, \ldots, n$, in both algorithms is set equal to the local optimal solution: $x^i(0) \in \arg\min_{x^i \in X} f_i(x^i)$.

Lastly, we set the weights coefficients so as to satisfy Assumption 2.5. To this end, we use the procedure described in [43] and formulate Algorithm 5, which is initialized with the adjacency matrix $\mathcal{A}$ of the graph (V, E) (i.e., the matrix which has 1 in position (i,j) if $(i,j) \in E$ and zero otherwise) and returns a matrix with the same sparsity pattern of $\mathcal{A}$. Note that since we require $a_i^i > 0$ in Assumption 2.5, then, $(i,i) \in E$. If the communication graph is undirected, then the adjacency matrix has full support (see [43] for a definition) and Algorithm 5 is guaranteed to converge to a doubly stochastic matrix.

Given that each base station is in charge of setting the transmission power of the mobile users that it is serving, when plotting the performance at time k we shall refer to the transmission power vector $\hat{p}(k)$ that contains as elements the transmission powers computed at iteration k by the base stations for the mobile users that they are serving.

Performance evaluation and comparative analysis of the distributed Algorithms 3 and 4 are performed in terms of either the cost

$$\sum_{i=1}^{n} f_i(\log(\hat{p}(k))), \;\; k = 0, 1, \ldots,$$

or the normalized relative error cost

$$\frac{\sum_{i=1}^{n} f_i(\log(\hat{p}(k))) - \min_{x\in X} \sum_{i=1}^{n} f_i(x)}{\min_{x\in X} \sum_{i=1}^{n} f_i(x)}, \ k = 0, 1, \dots .$$

All the numerical simulations were run using Matlab with CVX, [44], as optimization interface and Mosek™, [45], as solver. In the first simulation study, we consider a cellular network with $n = 16$ base stations on a 4 by 4 squared grid and $m = 16$ mobile users, one per base station.

We performed 100 runs of both Algorithms 3 and 4, where the position of each mobile user is extracted at random in a square of size 5 by 5 centered in a base station. The resulting histograms of the normalized relative costs at iterations $k = 10$, 50, 100, and 200, are plotted in Figure 2. The relative error decreases progressively for both, after an initial transient phase Algorithm 4 performs slightly better than Algorithm 3. This is also apparent from Figure 3 where the plots of the cost obtained with the two algorithms are reported for one of the extracted configurations. Note that both plots are smooth curves. As for the overshoot in the cost for Algorithm 4, $c(k)$ is large initially and makes the algorithm take a large step in the wrong direction, thus increasing the cost.

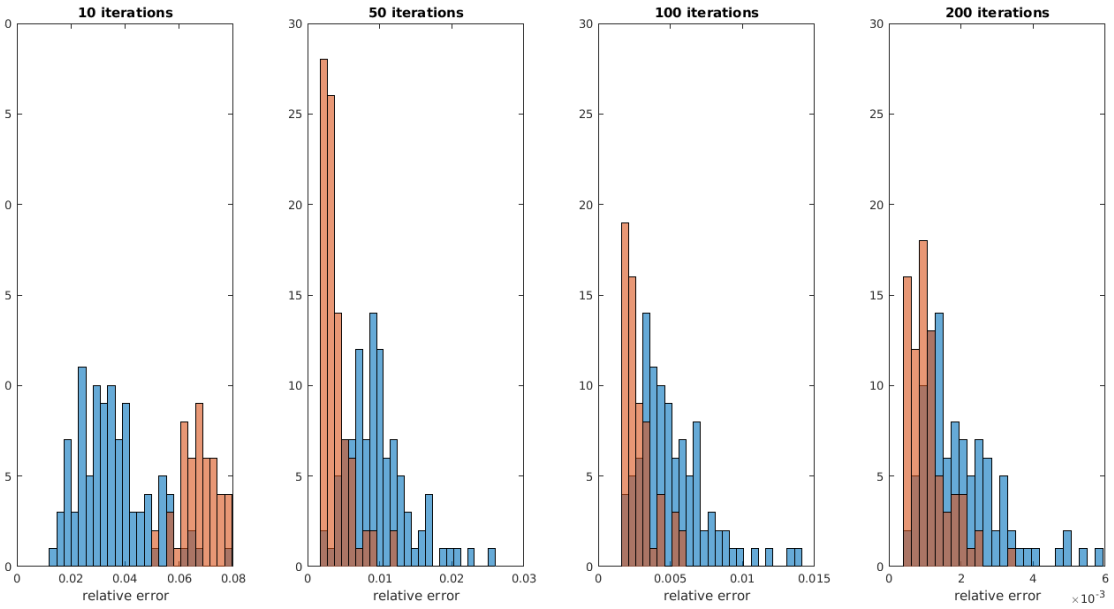

Figure 2: Histograms of the normalized relative cost for the cellular network with 16 base stations and 16 mobile users at iterations $k = 10$, 50, 100, and 200, of Algorithms 3 (in blue) and 4 (in red).

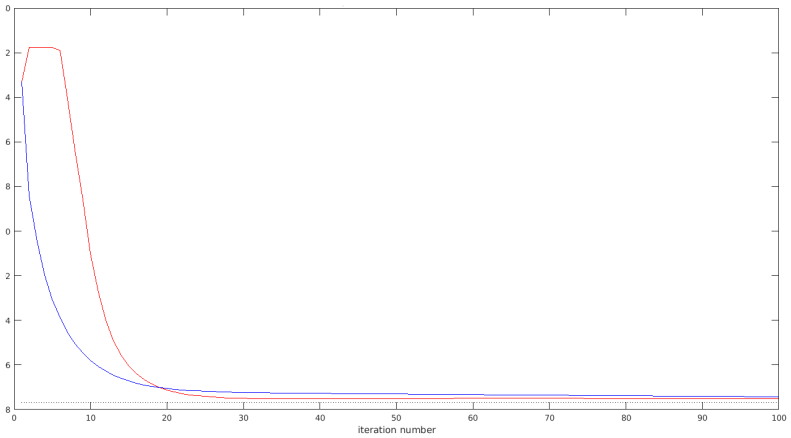

Figure 3: Cost function for a cellular network with 16 base stations and 16 mobile users for Algorithms 3 (in blue) and 4 (in red).

If we now consider the multi-user set-up, with $n = 16$ base stations but $m = 20$ mobile users, and perform the same kind of experiments, we get the histograms of the normalized relative cost and the plots for the cost in Figures 4 and 5, respectively. Convergence is slower and oscillations deteriorate the performance of the solution obtained by Algorithm 4.

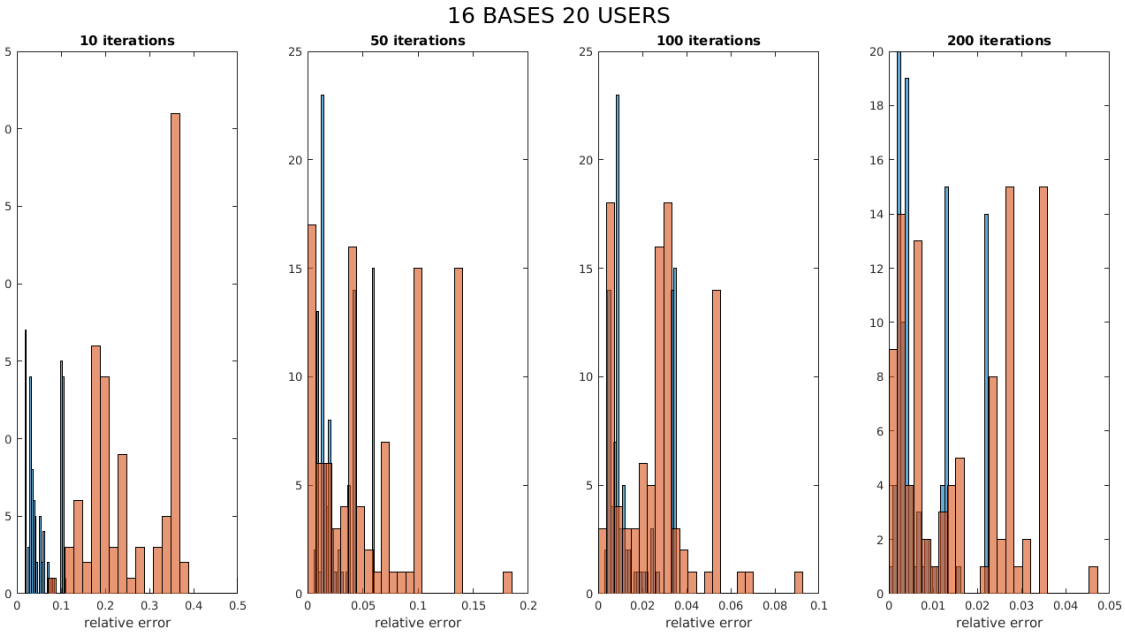

Figure 4: Histograms of the normalized relative cost for the cellular network with 16 base stations and 20 mobile users at iterations $k = 10$, 50, 100, and 200, of Algorithms 3 (in blue) and 4 (in red).

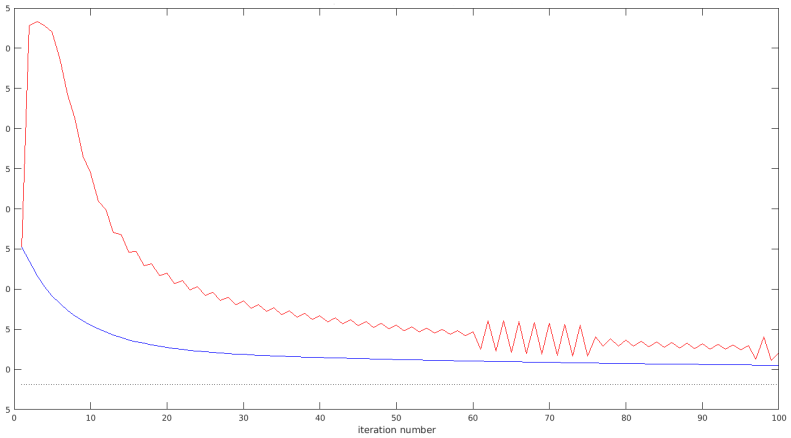

Figure 5: Cost function for a cellular network with 16 base stations and 20 mobile users for Algorithms 3 (in blue) and 4 (in red). The dotted constant line is the optimal cost.

In Figure 6, we report the time needed to run 100 iterations of both algorithms for different instances of 5 different single-user cellular networks. As expected Algorithm 4 is less time consuming.

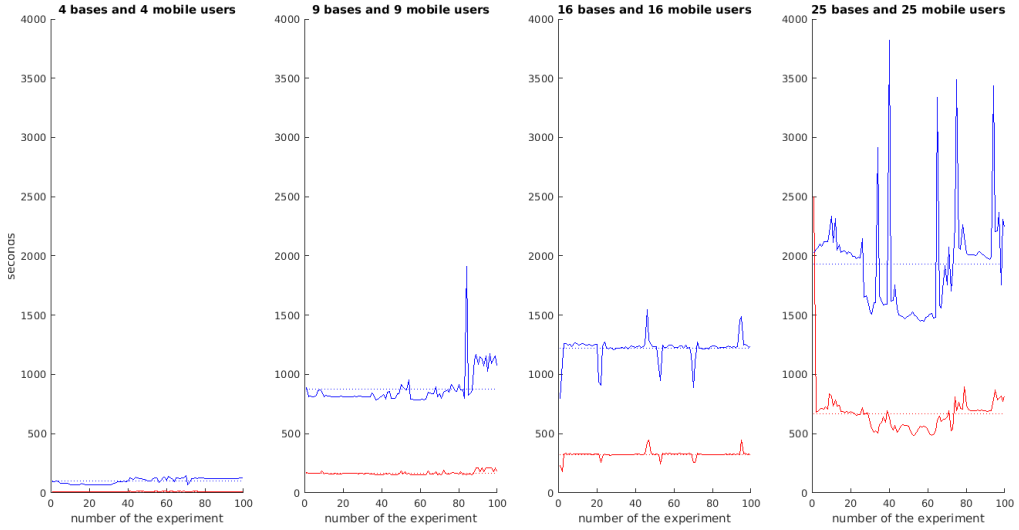

Figure 6: Simulation time for running 100 iterations of Algorithms 3 (in blue) and 4 (in red) in 100 tests.

In summary, we proposed a distributed transmission power control algorithm for a wireless cellular network. The algorithm is based on proximal minimization and represents an alternative to a gradient-based distributed algorithm that works in the same set-up, with the same guarantees on convergence and optimality. The two approaches show similar performance in the case when each base station serves a single mobile user in the cellular network. Admittedly, the proposed approach is computationally more demanding. However, in the relevant case of

multiple mobile users per base station, it shows better performance than the gradient-based approach since, as an effect of the cost being non-differentiable, using a sub-gradient induces an oscillatory behavior.

2.4.2 Building energy management

We now present an example that is part of the case study on smart grids of work package 5.

Consider a network of m buildings, each of them equipped with a different chiller plant, that share a common storage so as to avoid usage inefficiencies and increase the return on investment of the storage resource, which might be expensive to afford at an individual level (see Figure 7). Our objective is to minimize the total electric energy cost for the m building network, across a horizon of n_t steps of duration Δ. To this purpose, we need first to discrete-time dynamic model of the system that includes the building cooling energy request, the chiller plant, and the storage unit. As explained in the introduction of this section, by piling up in a vector the values of the variables of interest along the control horizon and expressing them as a function of the control input along that time horizon, we finally get a static description of the dynamical system evolution that fits our (static) optimization framework.

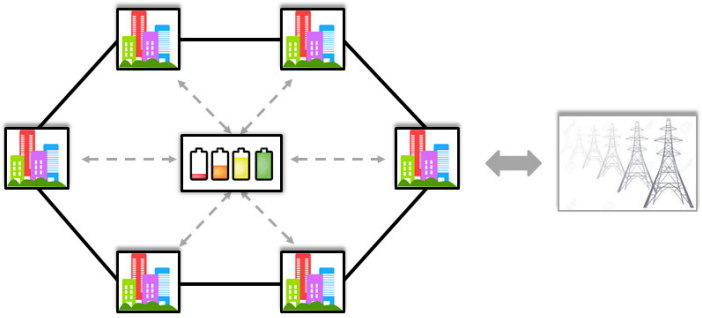

Figure 7: Example of a building network with a shared storage. The district is connected to the main grid that provides electrical energy.

Building cooling energy request: Consider a building composed of n_z thermally conditioned zones. For all $t = 1, \dots, n_t$, $z = 1, \dots, n_z$, let $E_{B,z}(t) \in \mathbb{R}$ be the cooling energy request of building zone z during time slot t. Denote then by $E_B(t) = \sum_{z=1}^{n_z} E_{B,z}(t)$ the energy request of the building over the time slot t, and by

$$E_B = \sum_{t=1}^{n_t} E_B(t) = \sum_{t=1}^{n_t} \sum_{z=1}^{n_z} E_{B,z}(t), \tag{21}$$

the energy request of the building over the entire horizon. For all $t = 1, \dots, n_t,$, $z = 1, \dots, n_z$, $E_{B,z}(t)$ constitutes of four energy contributions, namely

$$E_{B,z}(t) = E_{\text{walls},z}(t) + E_{\text{people},z}(t) + E_{\text{internal},z}(t) + E_{\text{inertia},z}(t), \tag{22}$$

where $E_{\text{walls},z}(t) \in \mathbb{R}$ is the amount of thermal energy exchanged between walls and zone z over the time slot t, $E_{\text{people},z}(t) \in \mathbb{R}$ and $E_{\text{internal},z}(t) \in \mathbb{R}$ is the thermal energy produced by people and by other internal sources of heat in zone z, respectively, and $E_{\text{inertia},z}(t) \in \mathbb{R}$ is the energy contribution of the thermal inertia of zone z, over the time slot t. A detailed description of each of these terms can be found in Deliverable 5.1. Derivation of the first contribution involves modeling walls as the composition of vertical layers ('slices') that may differ in width and material composition. The resulting model is dynamic with the slice and wall temperatures as state variable.

Chiller plant: A chiller plant converts electric energy into cooling energy. The cooling energy is then transferred to the building via, e.g., the chilled water circuit. Denote by $E_{\text{chiller},e}(t) \in \mathbb{R}$ the electric energy absorbed by the chiller to provide cooling energy $E_{\text{chiller},c} \in \mathbb{R}$ over a time slot of duration t, $t = 1, \dots, n_t$.

To facilitate the solution to the optimal energy management of the building district, we employ the following convex approximation of the Ng-Gordon model in [46], which relates the electric and the cooling energy (see Deliverable 5.1 for more details):

$$E_{\text{chiller},e}(t) = c_2(T_o(t\Delta))E^4_{\text{chiller},c}(t) + c_1(T_o(t\Delta))E^2_{\text{chiller},c}(t) + c_0(T_o(t\Delta)), \tag{23}$$

where, for each $t = 1, \dots, n_t$, the functions $c_0(\cdot), c_1(\cdot), c_2(\cdot) : \mathbb{R} \to \mathbb{R}$ depend on the ambient temperature $T_o(t\Delta)$.

Energy storage: Thermal energy storage is becoming widely used since it represents the most effective way, and often the only way, to take advantage of renewable energy sources. There are many different technical solutions to store thermal energy; the most widely used are fluid tanks. In buildings, and more generally in a smart grid context, they can be used as energy buffers for unbinding energy production from energy consumption. In particular, thermal energy storage for cooling energy can shift the production of cooling energy to off-peak hours of electrical energy consumption, make chillers operate in high-efficiency conditions, and smoothen peaks of electrical energy request with benefits both for power production and distribution network systems.

As discuss in Deliverable 5.1, we can adopt the following first order AutoRegressive eXogenous (ARX) model:

$$E_{\text{storage}}(t+1) = aE_{\text{storage}}(t) - \sum_{i=1}^{m} e_s^i(t), \tag{24}$$

where $E_{\text{storage}}(t) \in \mathbb{R}$ is the amount of cooling energy stored. In view of the multi-building problem considered in the next section we assume that the storage device is shared among m buildings, and denote by $e_s^i(t) \in \mathbb{R}$ the cooling energy exchanged ($e_s^i(t) > 0$ if the storage is discharged, and $e_s^i(t) < 0$ if it is charged), with building i, $i = 1, \ldots, m$, in time slot t, $t = 1, \ldots, n_t$. The coefficient $a \in (0, 1)$ is introduced to model energy losses. Note that (24) can be thought of as a discrete-time integrator, where the stored energy $E_{\text{storage}}(t+1)$ at time $t+1$ is computed by accumulating the cooling energy exchanged with all buildings up to time t, i.e., $\sum_{i=1}^{m} e_s^i(t')$ for all $t' \leq t$.

Optimal Energy management of a building district: We are now in a position to formulate the optimal energy management problem for a district of m buildings, each of them equipped with a different chiller plant, that share a common storage and are cooperatively aiming at minimize the electrical energy cost across a horizon of n_t steps.

To this end, append to all quantities introduced in the previous section the superscript i, to denote that they correspond to building i, $i = 1, \ldots, m$, e.g., $E^i_{\text{chiller},e}(t)$ denotes the cooling energy of the chiller at building i at time slot t, $\widetilde{T}^i$ denotes the vector of zone temperatures at building i, etc. For each $i = 1, \ldots, m$, $t = 1, \ldots, n_t$, the electric energy request of building i over the time slot t is given by the chiller electric energy request $E^i_{\text{chiller},e}(t)$. The latter is in turn related to the cooling energy exchange terms via (23).

For each building i, $i = 1, \ldots, m$, we will schedule the zone temperature set-points $\widetilde{T}^i(t\Delta)$, the energy exchange $e_s^i(t)$ with the storage, and the initial conditions for the temperature vector $T^i(\Delta)$ (including slice and wall temperatures), and the storage level $E_{\text{storage}}(1)$ by solving the following minimization problem:

$$\min_{\substack{\left\{\left\{\widetilde{T}^i(t\Delta)\in\mathbb{R}^{n_z}, e_s^i(t)\in\mathbb{R}\right\}_{t=1}^{n_t}\right\}_{i=1}^{m}, \\ \left\{T^i(\Delta)\in\mathbb{R}^{n_s n_w}\right\}_{i=1}^{m}, E_{\text{storage}}(1)\in\mathbb{R}}} \sum_{i=1}^{m}\sum_{t=1}^{n_t} \psi^i(t) E^i_{\text{chiller},e}(t), \tag{25}$$

where $\psi^i(t) \in \mathbb{R}$ is the electric energy price for building i, $i = 1, \ldots, m$, over the time slot t, $t = 1, \ldots, n_t$. This minimization is subject to the following constraints.

1. Electric energy request: For each $t = 1, \ldots, n_t$, the electric energy request $E^i_{\text{chiller},e}(t)$

of building i, $i = 1, \ldots, m$, is given by (23) as a function of the chiller cooling energy request $E^i_{\text{chiller},c}(t)$. The latter denotes the net cooling energy request, and is given by the difference between the total energy requested by the building minus the energy exchanged with the storage, i.e.,

$$E^i_{\text{chiller},c}(t) = E^i_B(t) - e^i_s(t), \tag{26}$$

where $E^i_B(t)$ is as shown in (21), and $e^i_s(t)$ is the energy exchange between building i and the storage.

2. Electric energy limits: For each $i = 1, \ldots, m$, $t = 1, \ldots, n_t$, the electric energy drawn from the network is limited to $E^i_{\max} \in \mathbb{R}$, as an effect of the chiller unit size and maximum capability, thus giving rise to

$$E_{\text{chiller},e}(t) \leq E^i_{\max}. \tag{27}$$

3. Cooling energy limits: For each $i = 1, \ldots, m$, $t = 1, \ldots, n_t$, the cooling energy request $E^i_B(t)$ of building i over time-slot t, as given by (21), is non-negative, i.e.,

$$E^i_B(t) \geq 0, \tag{28}$$

4. Comfort constraints: For each $i = 1, \ldots, m$, $t = 1, \ldots, n_t$, the zone temperature set-points is within certain limits, i.e.,

$$\widetilde{T}^i(t\Delta) \in [\widetilde{T}^i_{\min}(t\Delta),\ \widetilde{T}^i_{\max}(t\Delta)], \tag{29}$$

where $\widetilde{T}^i_{\min}(t\Delta) \in \mathbb{R}^{n_z}$, $\widetilde{T}^i_{\max}(t\Delta) \in \mathbb{R}^{n_z}$ denote the minimum and maximum, respectively, temperature limits, so that comfort is maintained. These limits may differ according to the type of each building.

5. Storage energy limits: For each $t = 1, \ldots, n_t$, the amount of cooling energy stored at time slot t should be non-negative and within a prescribed energy storage limit $E_{s,\max} \in \mathbb{R}$, i.e.,

$$E_{\text{storage}}(t) \in [0,\ E_{s,\max}]. \tag{30}$$

6. Storage energy exchange limits: For each $i = 1, \ldots, m$, $t = 1, \ldots, n_t$, the energy

exchanged with the storage is subject to

$$e_s^i(t) \in [-e_{s,\max}^i,\ e_{s,\max}^i], \tag{31}$$

where $e_{s,\max}^i \in \mathbb{R}$ denotes the maximum value of energy that can be exchanged with the storage for building i. Notice that we use symmetric limits for positive and negative energy exchanges.

7. Final value constraints: For each $i = 1, \dots, m$, the zone temperature, and the wall-slice temperature, at the beginning and at the end of the planning horizon should be equal:

$$\begin{aligned} \widetilde{T}^i(n_t\Delta) &= \widetilde{T}^i(\Delta), \\ T^i(n_t\Delta) &= T^i(\Delta). \end{aligned} \tag{32}$$

To ensure that at the end of the horizon the storage is sufficiently charged we impose the constraint

$$E_{\text{storage}}(n_t) \geq E_{\text{storage}}(1), \tag{33}$$

where we optimize with respect to $E_{\text{storage}}(1)$ (see (36)). Constraint (33) is of particular importance in case of a receding horizon implementation of the proposed scheme.

Note that, even though it is not shown explicitly to ease notation, $E_{\text{chiller},e}^i(t)$ in (25) is a function of the decision variables $\left\{\left\{\widetilde{T}^i(t\Delta), e_s^i(t)\right\}_{t=1}^{n_t}\right\}_{i=1}^m, \left\{T^i(\Delta)\right\}_{i=1}^m, E_{\text{storage}}(1)$; this can be verified by tracing the representation of $E_{\text{chiller},e}^i(t)$ via (23), (26), (21), where the energy terms in the latter equation depend on the decision variables according to the analysis of all contributions to the building cooling energy request.

For each $i = 1, \dots, m$, denote by

$$u_i = \left[\widetilde{T}^i(\Delta), \dots, \widetilde{T}^i(n_t\Delta), T^i(\Delta)\right]^\top \in \mathbb{R}^{n_t n_z + n_s n_w}, \tag{34}$$

all temperature related decision variables that correspond to building i.

Let $\bar{e}_s^i = \left[e_s^i(1), \dots, e_s^i(n_t), E_{\text{storage}}(1)\right]^\top \in \mathbb{R}^{n_t+1}$, and denote by

$$x = \left[\bar{e}_s^1, \dots, \bar{e}_s^m\right]^\top \in \mathbb{R}^{m(n_t+1)}, \tag{35}$$

the vector including all decision variables related to the energy exchange between the buildings and the storage, and the initial energy storage value. Note that u_i is indexed by i, $i = 1, \dots, m$,

Algorithm 6 Distributed algorithm

1: **Initialization**
2: $k = 0$.
3: Consider $x_i(0)$, $u_i(0)$,
such that $(x_i(0), u_i(0)) \in V_i$ for all $i = 1, \ldots, m$.
4: **For** $i = 1, \ldots, m$ **repeat until convergence**
5: $\bar{x}_i(k) = \sum_{j=1}^{m} a_j^i(k) x_j(k)$.
6: $(x_i(k+1), u_i(k+1))$
$\in \arg\min_{(x_i,u_i)\in V_i} f_i(x_i, u_i) + \frac{1}{2c(k)}||\bar{x}_i(k) - x_i||^2$.
7: $k \leftarrow k + 1$.

and can be thus thought of as a local decision vector related to the comfort and actuation constraints of each chiller plant, that can be enforced locally. On the other hand, x is treated as a global decision vector which is related to the energy exchange of the building network with the common storage device. Under the variable assignment in (34), (35), the energy management in (25)-(33) can be represented in a more compact notation by

$$\mathcal{P}: \min_{x\in\mathbb{R}^n, \{u_i\in\mathbb{R}^{n_i}\}_{i=1}^m} \sum_{i=1}^{m} f_i(x, u_i) \tag{36}$$

subject to

$$(x, u_i) \in V_i, \text{ for all } i = 1, \ldots, m, \tag{37}$$

where $f_i(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^{n_i} \to \mathbb{R}$ and $V_i \subseteq \mathbb{R}^{n+n_i}$, for all $i = 1, \ldots, m$. Note that x couples the individual decision vectors u_i via the objective in (36) and the constraints in (37).

In the part to follow we will occasionally refer to buildings as agents. We provide a distributed iterative procedure to solve $\mathcal{P}$, where, at every iteration, each agent i solves an appropriate local optimization problem and then exchanges information with other agents only regarding the temporarily obtained value for the common decision vector x. In this way, one can account for information privacy, because agents are not required to share the objective function f_i, the constraint set V_i, and their local decision vector u_i, $i = 1, \ldots, m$. In a building energy management context, this specifically means that each building does not need to reveal constraints that are related to its local consumption patterns or to occupants' preferences, nor to reveal information about its individual utility function, which may constitute private information in case buildings participate in a demand response program. Moreover, even though all the necessary information could be exchanged, solving $\mathcal{P}$ in a centralized fashion may result computationally intensive and our distributed algorithm is also a means to alleviate this issue. The pseudo-code of the proposed distributed procedure is given in Algorithm 6.

Initially, each agent i, $i = 1, \ldots, m$, starts with some tentative values $u_i(0)$ and $x_i(0)$ for

its local decision vector and the global decision vector, respectively. The latter constitutes an estimate of agent i (this justifies the subscript i in x_i) of what the value of the global decision vector might be. Those tentative values are chosen arbitrarily from the set of feasible solutions, i.e., $(x_i(0), u_i(0)) \in V_i$ (step 3). One sensible choice for $(x_i(0), u_i(0))$ is to set it such that $(x_i(0), u_i(0)) \in \arg\min_{(x_i,u_i)\in V_i,} f_i(x_i, u_i)$. At iteration $k+1$, each agent i constructs a weighted average $\bar{x}_i(k)$ of the solutions $x_j(k),\ j = 1, \ldots, m$ communicated by the other agents and its own one (step 5). Coefficient $a_j^i(k) \geq 0$, indicates how agent i weights the solution received by agent j at iteration k, and $a_j^i(k) = 0$ encodes the fact that agent i does not receive any information from agent j at iteration k (i.e. the communication link between agents i and j is not active at iteration k). Agent i solves then a local minimization problem, seeking the optimal solution pair (x_i, u_i) within V_i that minimizes a performance criterion, which is defined as a linear combination of the local objective function $f_i(x_i, u_i)$ and a quadratic term, penalizing the difference from $\bar{x}_i(k)$ (step 6). The relative importance of these two terms is dictated by $c(k) > 0$. Since multiple minimizers may exist, we assume that at every iteration the same deterministic tie-break rule (as e.g. that implemented by a deterministic numerical solver) is used.

Algorithm 6 is closely related to the distributed Algorithm 1. However, in Algorithm 6 neighboring agents need to exchange at every iteration their tentative estimates for the value of the global decision vector only, while the distributed Algorithm 1 requires to exchange both the global and the local decision vectors. When the dimension of the local decision vector is high compared to the global one, as it is the case in the building energy management problem, this would unnecessarily increase the amount of information that needs to be exchanged. Algorithm 6 alleviates this issue by exploiting the particular structure of $\mathcal{P}$, where the objective functions and the constraint sets are coupled only by means of x. This requires some further extension of the theory to prove that Algorithm 6 converges, and agents reach consensus to a common value for the global decision vector x that, together with the converged values for the local decision vectors u_i, $i = 1, \ldots, m$, forms an optimal solution of $\mathcal{P}$. The interested reader is referred to [47].

Case study: Consider a network of $m = 3$, identical, three-storey buildings, each one with a 20m by 20m base, a total height of 9m, flat rooftop and half glazed lateral surfaces. Each building is divided into $n_z = 3$ thermal zones (one per floor) and is equipped with its own chiller, namely, building 1 has a medium-size chiller, building 2 a small one, and building 3 a large one.

The structure of the buildings is schematically illustrated in Figure 8 and the COP curves

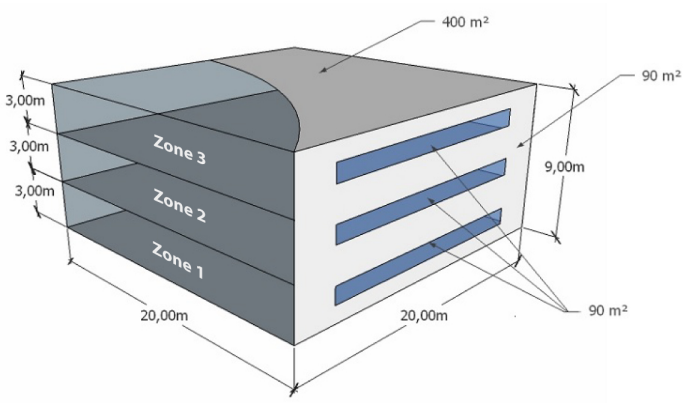

Figure 8: Structure of each building.

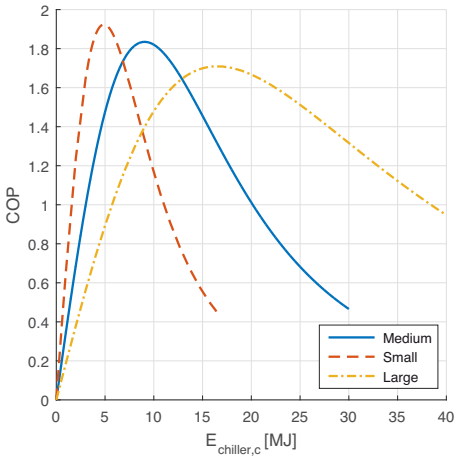

Figure 9: Chiller COP curves for each building.

i	Size	c_2^i	c_1^i	c_0^i	$E_{\max}^i$ [MJ]
1	Medium	$3.79 \cdot 10^{-5}$	$2.77 \cdot 10^{-2}$	2.46	30
2	Small	$2.49 \cdot 10^{-4}$	$4.98 \cdot 10^{-2}$	1.26	18
3	Large	$3.56 \cdot 10^{-6}$	$1.58 \cdot 10^{-2}$	5.11	40

Table 1: Chiller coefficients and maximal cooling energy.

as a function of the cooling energy request $E_{\text{chiller},c}$ are shown in Figure 9. The parameters of the biquadratic approximations are assumed to be constant for simplicity. Their value is reported in Table 1 together with the maximal admissible cooling energy request. Index i, $i = 1, \ldots, 3$, in Table 1 is used to denote the building.

The external disturbances affecting the buildings are reported in Figure 10. Longwave and shortwave solar radiation are depicted with square and circle markers respectively. The outside temperature and the occupancy are plotted with triangles and diamonds respectively. Note that the three buildings are supposed to be subject to the same disturbance profiles, and the occupancy shall be intended per building and equally partitioned among the zones. The

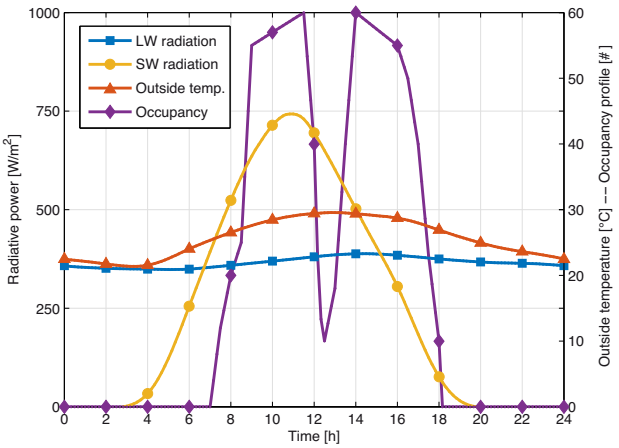

Figure 10: Disturbance profiles: Longwave (LW) and shortwave (SW) solar radiation, outdoor temperature and occupancy.

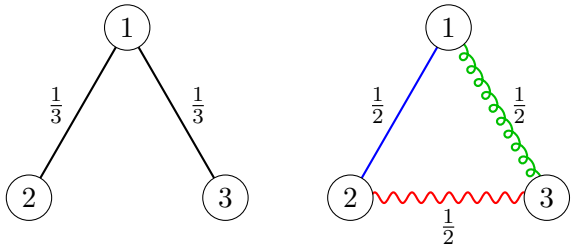

Figure 11: Communication structure: Fixed (left panel) and time-varying (right panel).

period in which the occupancy is greater than zero is referred to as "occupancy period" and it is within the "working hours" range 7AM to 6PM. In all buildings, temperature constraints are set to $\widetilde{T}^i_{\min} = 20°\text{C}$ and $\widetilde{T}^i_{\max} = 24°\text{C}$ during working hours and to $\widetilde{T}^i_{\min} = 16°\text{C}$ and $\widetilde{T}^i_{\max} = 30°\text{C}$ otherwise.

For the control problem, we considered a time horizon of 24 hours discretized in $n_t = 144$ time slots of $\Delta = 10\text{min}$ each. We tested the proposed algorithm with two different types of bi-directional communication topologies. The first one (Figure 11, left panel) is a connected topology, in which buildings 1 and 3 exchange information only with building 2 but not with each other, and the communication scheme is kept fixed across iterations. The second one (Figure 11, right panel) is a time-varying periodic topology in which, at each iteration k, only two buildings communicate. The order in which the links are activated within the period is the following: $(1, 2)$ (blue straight), $(2, 3)$ (red wavy), and $(1, 3)$ (green spring). In Figure 11 we also report the coefficients $a^i_j(k)$ for $j \neq i$ near the corresponding edge (i, j). The $a^i_i(k)$ coefficients, $i = 1, \ldots, m$, are not reported but they can be easily retrieved so that Assumption 2.5 is satisfied.

We applied Algorithm 6 to the two communication structures and in both cases the proposed distributed approach was able to retrieve the optimal solution.

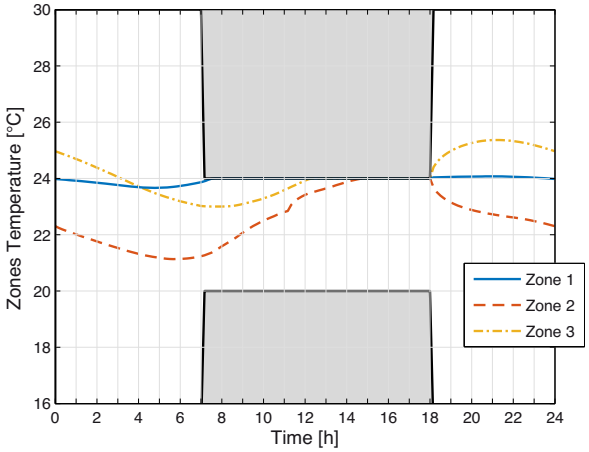

Figure 12: Optimal zone temperature profiles of building 1. The temperature of zone 2 (at the middle) is always lower than the other two, since it acts as a passive thermal storage to drain the heat of the other zones through floor and ceiling.

Figure 12 shows the optimal temperature profiles for the three zones of building 1. It can be observed that, while the profiles of zones 1 and 2 are kept close to the maximum temperature bound of the working hours comfort range (outside the grey area), the temperature of zone 2 is always lower than the other two. Zone 2 is indeed subject to a pre-cooling phase before the occupancy period so as to cool down the building, acting as an additional passive thermal storage to drain the heat of the other zones through floor and ceiling. The temperature profiles of the other two buildings are very similar to that of building 1, and hence are not reported here.

In Figures 13 and 14 we report the storage profiles of building 1 at iteration $k = 1$ and at consensus (when Algorithm 6 converges), respectively. From Figure 13 it is clear that, at the beginning, building 1 acts in a "selfish" manner and its optimal strategy is to constantly withdraw cooling energy from the storage ($e_s^1 > 0$, solid line), thus forcing buildings 2 and 3 to charge the storage ($e_s^2 < 0$ and $e_s^3 < 0$, dashed and dot-dashed lines, respectively). The stored energy is shown with the black dotted line. The consensus solution depicted in Figure 14 is instead cooperative. Building 3, which has the biggest chiller, is constantly providing cooling energy ($e_s^3 < 0$) to the shared storage; building 2, which has the smallest chiller, is constantly withdrawing energy ($e_s^2 > 0$) from it; and building 1 provides/retrieves energy to/from the storage depending on the time slot. In this way, differences in the chiller sizes are compensated through the storage.

Figure 15 and 16 show the COP coefficient of the chillers of the three buildings (resulting from the optimization of building 1) at $k = 1$ and at consensus, respectively. In Figure 15 building 1 is clearly optimizing the efficiency of its own chiller disregarding completely the

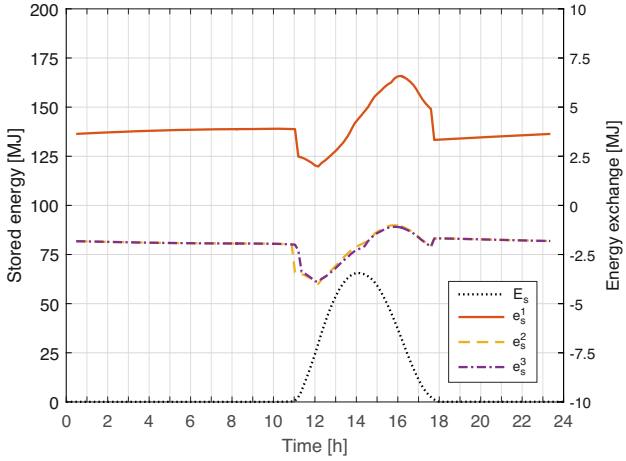

Figure 13: Storage profiles at iteration $k = 1$. Building 1 acts in a "selfish"manner and its optimal strategy is to constantly withdraw cooling energy from the storage ($e_s^1 > 0$, solid line), thus forcing buildings 2 and 3 to charge the storage ($e_s^2 < 0$ and $e_s^3 < 0$, dashed and dot-dashed lines, respectively). The stored energy is shown with the black dotted line.

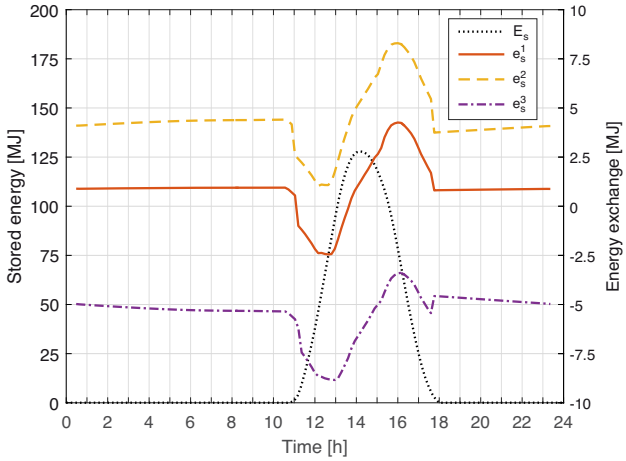

Figure 14: Storage profiles at consensus. Cooperative solution, with building 3, which has the biggest chiller, is constantly providing cooling energy ($e_s^3 < 0$) to the shared storage; building 2, which has the smallest chiller, is constantly withdrawing energy ($e_s^2 > 0$) from it; and building 1 provides/retrieves energy to/from the storage depending on the time slot. The stored energy is shown with the black dotted line.

efficiency of the other two, whereas the consensus solution reported in Figure 16 shows that the efficiency of the two other chillers is increased significantly at the expense of a slight deterioration in the one of building 1, thus resulting in an overall benefit for the building district.

The number of iterations needed to achieve consensus are 278 for the fixed topology and 1032 for the time-varying topology, where we considered the solution to be at consensus if either the absolute or the relative difference between the solutions of the agents across two consecutive iterations was less than a given threshold, which was taken to be 10^{-3}.

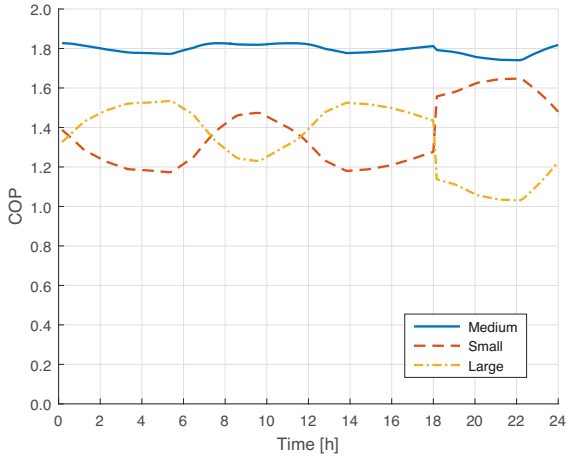

Figure 15: COP profiles at iteration $k = 1$. Building 1 is clearly optimizing the efficiency of its own chiller disregarding completely the efficiency of the other two.

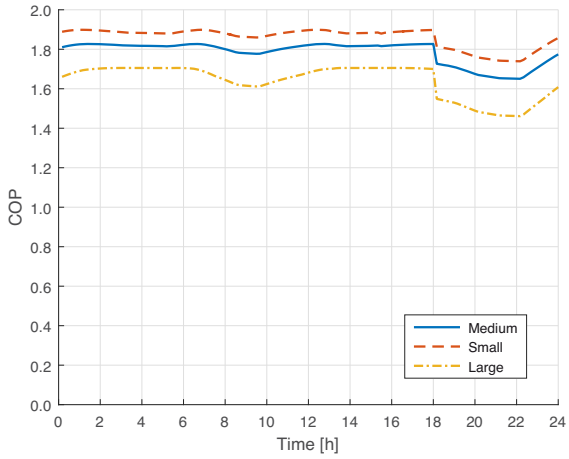

Figure 16: COP profiles at consensus. The efficiency of the chillers of buildings 2 and 3 is increased significantly at the expense of a slight deterioration in the one of building 1, thus resulting in an overall benefit for the building district.

2.4.3 Electric vehicles charging control

In this section we demonstrate the efficacy of the approach in Section 2.3 on an energy management system application that is part of the smart grid case study in work package 5 and involves electric vehicles. More specifically, we shall consider a modified version of the Plug-in Electric Vehicles (PEVs) charging problem described in [48]. This problem consists in finding an optimal overnight charging schedule for a fleet of m vehicles, which has to be compatible with local requirements and limitations (e.g., desired final state of charge and maximum charging power for each vehicle), and must satisfy some network-wide constraints (e.g., maximum power that the network can deliver).

Specifically, we hereby consider a slight modification of the "only charging" problem in [48], in that we allow for the optimization of the vehicles charging rate at each time slot, instead

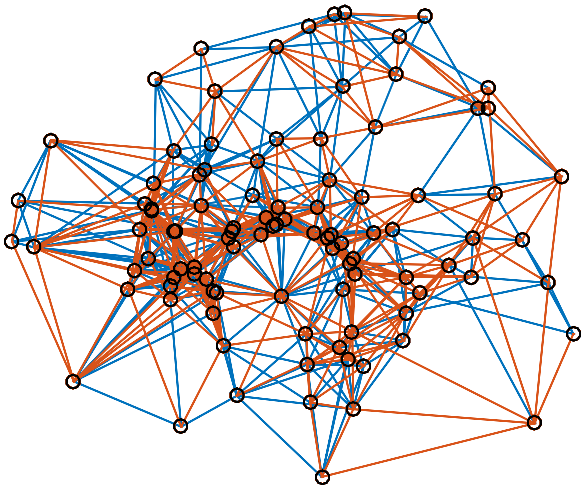

Figure 17: Network of $m = 100$ agents.

of deciding whether to charge or not to charge the vehicle at some fixed charging rate. The overall charging problem can be formalized as the following optimization program

$$\begin{aligned} \min_{\{x_i \in X_i\}_{i=1}^m} \quad & \sum_{i=1}^m c_i^\top x_i \\ \text{subject to:} \quad & \sum_{i=1}^m \left(A_i x_i - \frac{b}{m} \right) \leq 0 \end{aligned} \tag{38}$$

which is a linear program (X_i are indeed bounded convex polytopic sets) having the same structure of (5) and satisfying the assumptions for Theorems 2.2 and 2.3 to hold. In (38) the components of the optimization vector x_i represent the charging rate for vehicle i in given time slots, vector c_i gives the costs for charging vehicle i with unitary charging rate, X_i expresses local requirements and limitations for vehicle i such as desired final state of charge and battery rated capacity, while $\sum_{i=1}^m (A_i x_i - b/m) \leq 0$ encodes network-wide power constraints. We refer the reader to [48] for the precise formulation of all quantities in (38).

In our simulation we considered a fleet of $m = 100$ vehicles. According to the "only charging" set-up in [48], each vehicles has $n_i = 24$ decision variables and a local constraint set defined by 197 inequalities. There are $p = 48$ coupling inequalities, and therefore we have 48 Lagrange multipliers to optimize for the dual problem.

The communication network is depicted in Figure 17 and corresponds to a connected graph, whose edges are divided into two groups: the blue and the red ones, which are activated alternatively; this way Assumption 2.6 is satisfied with a period of $T = 2$. For each set of edges we created a doubly stochastic matrix so as to satisfy Assumption 2.5. Finally, we

selected $c(k) = \frac{1}{k+1}$.

We ran Algorithm 2 for 1000 iterations. Figure 18 shows the evolution of the agents' estimates $\lambda_i(k)$, $i = 1, \ldots, m$ across iterations. As expected, all agents gradually reach consensus on the optimal Lagrange multipliers of (38) (red triangles). Note that, for the

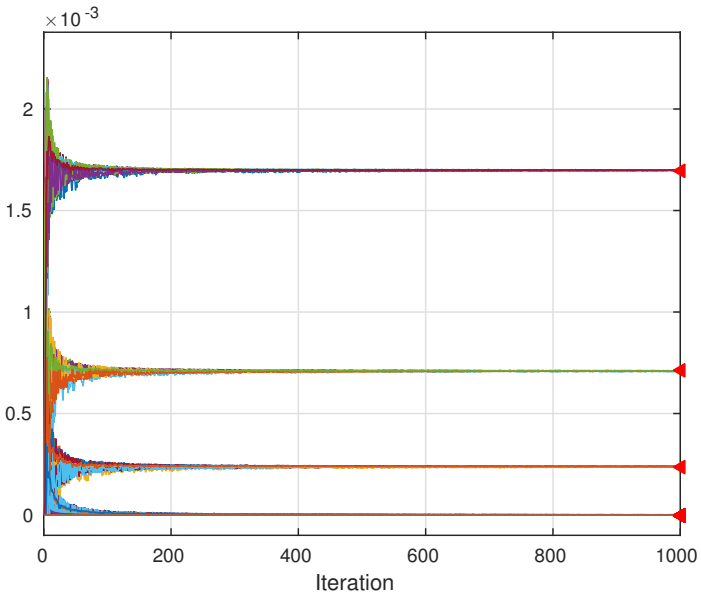

Figure 18: Evolution of the agents' estimates $\lambda_i(k)$, $i = 1, \ldots, m$. Red triangles represent the optimal dual solution.

problem at hand, only 3 multipliers are positive, while all the remaining 45 are equal to zero (in the figure, there are 45 red triangles in 0 each one on top of the other). Figure 19 instead shows the evolution of the primal objective value $\sum_{i=1}^{m} c_i^\top x_i$ (upper plot), and constraint violation in terms of $\max\{\sum_{i=1}^{m}(A_i x_i - b/m), 0\}$ (lower plot), where x_i is replaced by two different sequences: $\hat{x}_i(k)$ (dashed lines), and $\tilde{x}_i(k)$ (solid lines), $\tilde{x}_i(k)$ being defined as

$$\tilde{x}_i(k+1) = \begin{cases} \hat{x}_i(k+1) & k < k_{s,i} \\ \dfrac{\sum_{r=k_{s,i}}^{k} c(r) x_i(r+1)}{\sum_{r=k_{s,i}}^{k} c(r)} & k \geq k_{s,i} \end{cases} \tag{39}$$

where $k_{s,i} \in \mathbb{N}_+$ is the iteration index related to a specific event, namely, the "practical" convergence of the Lagrange multipliers, as detected by agent i. Specifically, in the proposed example $k_{s,i}$ is the first iteration step at which the quantity $\|\lambda_i(k+1) - \ell_i(k)\|_2$ has kept below a certain threshold (10^{-5} in our simulation) for $m = 100$ consecutive iterations. Given that $\tilde{x}_i(k)$ is a refresh of $\hat{x}_i(k)$, it is easy to show via the same argument used for $\hat{x}_i(k)$ that Theorem 2.3 holds also for $\{\tilde{x}_i(k)\}_{k\geq 0}$, $i = 1, \ldots, m$.

As can be seen from Figure 19, the rate of convergence of the cost and the constraints

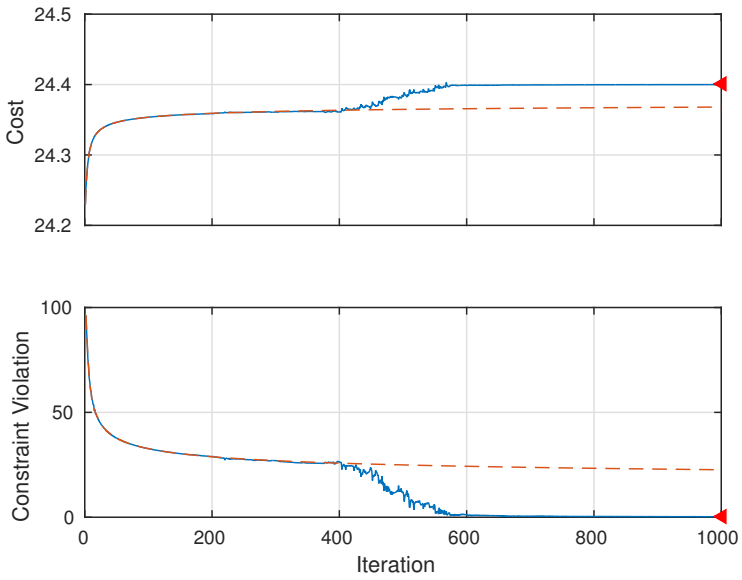

Figure 19: Evolution of primal objective $\sum_{i=1}^{m} c_i^\top x_i$ (upper plot) and constraint violation $\max\{\sum_{i=1}^{m}(A_i x_i - b/m), 0\}$ (lower plot) as a function of $\hat{x}_i(k)$ (dashed lines), and $\tilde{x}_i(k)$ (solid lines).

violation computed with the $\{\hat{x}_i(k)\}_{k\geq 0}$ sequence appears to be slow. In comparison with the rate of $O(1/k)$ in [49], a lower bound for our auxiliary sequence to converge is in fact given by $1/\sum_{i=0}^{\infty} c(k) \sim O(1/\log(k))$ (see the discussion in [6] on the rate of convergence for the dual objective value). We therefore believe that the difference in the convergence rate between Algorithm 2 and [49] might be primarily due to the constant vs. vanishing step-size. Having a vanishing step-size, however, allows us to provide optimality guarantees, while in [49] only convergence to a neighbourhood of the optimal solution is guaranteed. The motivation for introducing the modified auxiliary sequence (which has the same asymptotic convergence rate of the original one) is mainly to counteract the fact that the convergence of $\hat{x}_i(k)$ is also adversely affected by the bad estimates of the Lagrange multipliers obtained at the early stages of the algorithm. By the re-initialization mechanism, $\tilde{x}_i(k)$ for $k \geq k_{s,i}$ depends only on estimates of the Lagrange multipliers that are very close to $\lambda^\star$ and, as such, it presents a much better numerical behavior than $\hat{x}_i(k)$.

2.5 Extension to the stochastic case

Of particular interest is the case where agents cooperate seeking a solution to an optimization program where constraints depend on an uncertain parameter and should be robustly satisfied for all values that this parameter may take. This poses additional challenges when devising a distributed solution methodology. Here, we extend the approach in Section 2.2 (which can be applied also to solve problem (5) in Section 2.3, see Remark 2.3) to the case where the agents'

constraint sets are affected by a possibly common uncertainty vector by exploiting results on scenario-based optimization [50–55].

In particular, we assume that each agent is provided with a given set of uncertainty realizations (scenarios) and enforces the constraints only on these scenarios. We then show that the distributed algorithm presented in Section 2.2 is applicable and that the converged solution is feasible in a probabilistic sense for the constraints of the centralized problem, i.e., it satisfies with high probability all agents' constraints when an unseen uncertainty instance is realized. To achieve this we rely on the novel contribution of [56], which leads to a sharper result compared to the one that would be obtained by a direct application of the basic scenario theory [51]. Our approach can be thought of as the data-driven counterpart of robust or worst-case optimization paradigms, enabling us to provide a priori guarantees on the probability of constraint satisfaction without imposing any assumptions on the underlying distribution of the uncertainty and its moments, and/or the geometry of the uncertainty sets (e.g., [57, Chapters 6, 7]); however, providing the overall feasibility statement with a certain confidence. A key feature of our work is that we provide a distributed implementation of the scenario approach, which is typically performed in a centralized fashion, allowing agents to consider a different set of scenarios in their local minimization problems. This reduces the communication effort and, at the same time, allow to preserve privacy of local information.

Let us consider problem $\mathcal{P}$ in (1) in the more general case where the constraint set X_i of each agent $i = 1, \dots, m$, depends on an uncertain parameter $\delta \in \Delta$. This leads to the following optimization problem $\mathcal{P}_\delta$, where the subscript δ is introduced to emphasize the dependency with respect to the uncertainty:

$$\begin{aligned} \mathcal{P}_\delta : \min_{x \in \mathbb{R}^n} & \sum_{i=1}^{m} f_i(x) \\ \text{subject to } & x \in \bigcap_{\delta \in \Delta} \bigcap_{i=1}^{m} X_i(\delta). \end{aligned} \tag{40}$$

Note that problem $\mathcal{P}_\delta$ in (40) differs from problem $\mathcal{P}$ in (1) in that the constraint sets depend on the uncertainty δ, which allows to model the presence of disturbances affecting the system performance. The fact that uncertainty appears only in the constraints and not in the objective functions is without loss of generality; in the opposite case, an epigraphic reformulation would recast the problem in the form of $\mathcal{P}_\delta$.

Problem (40) is a robust program, in that any feasible solution x should belong to $\bigcap_{i=1}^{m} X_i(\delta)$ for all realizations $\delta \in \Delta$ of the uncertainty.

The following modifications to Assumptions 2.1-2.3 are imposed:

Assumption 2.11 *1) For each $i = 1, \dots, m$, $X_i(\delta)$ is a convex set for any $\delta \in \Delta$; 2) For each $i = 1, \dots, m$, and for any finite set S of values for δ, $\bigcap_{\delta \in S} X_i(\delta)$ is compact; and 3) For any finite set S of values for δ, $\bigcap_{i=1}^m \bigcap_{\delta \in S} X_i(\delta)$ has a non-empty interior.*

Given that it is generally difficult to solve problem (40) when Δ is a continuous set, and motivated by data-driven considerations, we assume that each agent i, $i = 1, \dots, m$, is provided with a fixed number of realizations of δ, referred to as scenarios, extracted according to the underlying probability measure $\mathbb{P}$ with which δ takes values in Δ. According to the information about the scenarios that agents possess, two cases are distinguished in the sequel and the properties of the corresponding scenario programs are analyzed.

Scenarios as a common resource

We first consider the case where all agents are provided with the same scenarios of δ, i.e., scenarios can be thought of as a common resource for the agents. This is the case if all agents have access to the same set of historical data for δ, or if agents communicate the scenarios with each other. The latter case, however, increases the communication requirements.

Let $\bar{N} \in \mathbb{N}_+$ denote the number of scenarios, and $\bar{S} = \{\delta^{(1)}, \dots, \delta^{(\bar{N})}\} \subset \Delta$ be a set of scenarios available to all agents. The scenarios are independently and identically distributed (i.i.d.) according to $\mathbb{P}$. Consider then the following optimization program $\mathcal{P}_{\bar{N}}$, where the subscript $\bar{N}$ is introduced to emphasize the dependency with respect to the uncertainty scenarios.

$$\mathcal{P}_{\bar{N}} : \min_{x \in \mathbb{R}^n} \sum_{i=1}^m f_i(x)$$
$$\text{subject to } x \in \bigcap_{\delta \in \bar{S}} \bigcap_{i=1}^m X_i(\delta). \tag{41}$$

Clearly, $x \in \bigcap_{\delta \in \bar{S}} \bigcap_{i=1}^m X_i(\delta)$ is equivalent to $x \in \bigcap_{i=1}^m \bigcap_{\delta \in \bar{S}} X_i(\delta)$, and $\mathcal{P}_{\bar{N}}$ is amenable to be solved via the distributed Algorithm 1 with $\bigcap_{\delta \in \bar{S}} X_i(\delta)$ in place of X_i, for all $i = 1, \dots, m$. Let $X^*_{\bar{N}} \subseteq \bigcap_{i=1}^m \bigcap_{\delta \in \bar{S}} X_i(\delta)$ be the set of minimizers of $\mathcal{P}_{\bar{N}}$. We then have the following corollary of Theorem 2.1.

Corollary 2.1 *Consider Assumptions 2.1-2.6 and the additional conditions in Assumption 2.11. We have that, for some $x^*_{\bar{N}} \in X^*_{\bar{N}}$,*

$$\lim_{k \to \infty} \|x_{i,\bar{N}}(k) - x^*_{\bar{N}}\| = 0, \text{ for all } i = 1, \dots, m, \tag{42}$$

where $x_{i,\bar{N}}(k)$ denotes the solution generated at iteration k, step 8 of Algorithm 1, when X_i is replaced by $\bigcap_{\delta\in\bar{S}} X_i(\delta)$.

We address the problem of quantifying the robustness of the minimizer $x^*_{\bar{N}}$ of $\mathcal{P}_{\bar{N}}$ to which our iterative scheme converges according to Corollary 2.1. In the current set-up a complete answer is given by the scenario approach theory [50, 51], which shows that $x^*_{\bar{N}}$ is feasible for $\mathcal{P}_\delta$ up to a quantifiable level $\bar{\varepsilon}$. This result is based on the notion of support constraints (see also Definition 4 in [50]), and in particular on the notion of support set [56] (also referred to as compression scheme in [55]). Given an optimization program, we say that a subset of the constraints constitutes a support set, if it is the minimal cardinality subset of the constraints such that by solving the optimization problem considering only this set of constraints, we obtain the same solution with the original problem that includes all constraints. As a consequence, all constraints that do not belong to the support set are in a sense redundant since their removal leaves the optimal solution unaffected.

By Theorem 3 of [50], for any convex optimization program the cardinality of the support set is at most equal to the number of decision variables n, whereas in [58] a refined bound is provided. The subsequent result is valid for any given bound on the cardinality of the support set. Therefore, and since $\mathcal{P}_{\bar{N}}$ is convex, let $d \in \mathbb{N}_+$ be a known upper-bound for the cardinality of its support set. A direct application of the scenario approach theory in [50] leads then to the following result.

Theorem 2.5 *Fix $\beta \in (0,1)$ and let*

$$\bar{\varepsilon} = 1 - \sqrt[\bar{N}-d]{\frac{\beta}{\binom{\bar{N}}{d}}}. \tag{43}$$

We then have that

$$\mathbb{P}^{\bar{N}}\Big\{\bar{S}\in\Delta^{\bar{N}}:\ \mathbb{P}\Big\{\delta\in\Delta:\ x^*_{\bar{N}}\notin\bigcap_{i=1}^{m} X_i(\delta)\Big\}\le\bar{\varepsilon}\Big\}\ge 1-\beta. \tag{44}$$

In words, Theorem 2.5 implies that with confidence at least $1-\beta$, $x^*_{\bar{N}}$ is feasible for $\mathcal{P}_\delta$ apart from a set of uncertainty instances with measure at most $\bar{\varepsilon}$. Notice that $\bar{\varepsilon}$ is in fact a function of $\bar{N}$, β and d. We suppress this dependency though to simplify notation. Note that even though $\mathcal{P}_{\bar{N}}$ does not necessarily have a unique solution, Theorem 2.5 still holds for the solution returned by Algorithm 1 (assuming convergence), since it is a deterministic algorithm and hence serves as a tie-break rule to select among the possibly multiple minimizers.

Following [51], (43) could be replaced with an improved $\bar{\varepsilon}$, obtained as the solution of

$\sum_{k=0}^{d-1} \binom{\bar{N}}{k} \bar{\varepsilon}^k \big(1-\bar{\varepsilon}\big)^{\bar{N}-k} = \beta$. However, we often use (43) since it gives an explicit relation expression for $\bar{\varepsilon}$, and also renders (44) directly comparable with the results provided in the next subsection.

In case $\bar{\varepsilon}$ exceeds one, the result becomes trivial. However, note that Theorem 2.5 can be also reversed (as in experiment design) to compute the number $\bar{N}$ of scenarios that is required for (44) to hold for given $\bar{\varepsilon}, \beta \in (0,1)$. This can be determined by solving (43) with respect to $\bar{N}$ with the chosen $\bar{\varepsilon}$ fixed (e.g., using numerical inversion). The reader is referred to Theorem 1 of [50] for an explicit expression of $\bar{N}$.

Scenarios as a private resource

We now consider the case where the information carried by the scenarios is distributed, that is, each agent has its own set of scenarios, which constitute agents' private information. Assume that each agent i, $i = 1, \ldots, m$, is provided with a set $S_i = \{\delta_i^{(1)}, \ldots, \delta_i^{(N_i)}\} \subset \Delta$ of $N_i \in \mathbb{N}_+$ i.i.d. scenarios of δ, extracted according to the underlying probability measure $\mathbb{P}$. Here, $\delta_i^{(j)}$ denotes scenario j of agent i, $j = 1, \ldots, N_i$, $i = 1, \ldots, m$. The scenarios across the different sets S_i, $i = 1, \ldots, m$, are independent from each other. The total number of scenarios is $N = \sum_{i=1}^m N_i$. Consider then the following optimization program $\mathcal{P}_N$, where each agent has its own scenario set.

$$\begin{aligned} \mathcal{P}_N : \min_{x \in \mathbb{R}^n} & \sum_{i=1}^m f_i(x) \\ \text{subject to } & x \in \bigcap_{i=1}^m \bigcap_{\delta \in S_i} X_i(\delta). \end{aligned} \tag{45}$$

Program $\mathcal{P}_N$ can be solved via the distributed Algorithm 1, so that a solution is obtained without exchanging any private information regarding the scenarios. In fact, one can apply Algorithm 1 with $\bigcap_{\delta \in S_i} X_i(\delta)$ in place of X_i, for all $i = 1, \ldots, m$.

Similarly to Corollary 2.1, letting $X_N^* \subseteq \bigcap_{i=1}^m \bigcap_{\delta \in S_i} X_i(\delta)$ be the set of minimizers of $\mathcal{P}_N$, we have the following corollary of Theorem 2.1.

Corollary 2.2 *Consider Assumptions 2.1-2.6 and the additional conditions in Assumption 2.11. We have that, for some $x_N^* \in X_N^*$,*

$$\lim_{k \to \infty} \|x_{i,N}(k) - x_N^*\| = 0, \text{ for all } i = 1, \ldots, m, \tag{46}$$

where $x_{i,N}(k)$ denotes the solution generated at iteration k, step 8 of Algorithm 1, when X_i is replaced by $\bigcap_{\delta \in S_i} X_i(\delta)$.

As in the previous section, we show that the minimizer x_N^* of $\mathcal{P}_N$ to which our iterative scheme converges according to Corollary 2.2 is feasible in a probabilistic sense for $\mathcal{P}_\delta$. Here, a difficulty arises, since we seek to quantify the probability that x_N^* satisfies the global constraint $\bigcap_{i=1}^m X_i(\delta)$, where δ is a common parameter to all $X_i(\delta)$, $i = 1, \ldots, m$, while x_N^* has been computed considering $X_i(\delta)$ for uncertainty scenarios that are independent from those of $X_j(\delta)$, $j \neq i$, $i = 1, \ldots, m$.

Let $S = \{S_i\}_{i=1}^m$ be a collection of the scenarios of all agents. Similarly to the previous case, we denote by $d \in \mathbb{N}_+$ a known upper-bound for the cardinality of the support set of $\mathcal{P}_N$. However, the way the constraints of this set are split among the agents depends on the specific scenarios S employed. Therefore, for each set of scenarios S, denote by $d_{i,N}(S) \in \mathbb{N}$ (possibly equal to zero) the number of constraints that belong to the support set of $\mathcal{P}_N$ and correspond to S_i, $i = 1, \ldots, m$, i.e., that belong to the constraints of agent i. We then have that $\sum_{i=1}^m d_{i,N}(S) \leq d$, for any $S \in \Delta^N$. For short we will write $d_{i,N}$ instead of $d_{i,N}(S)$ and make the dependency on S explicit only when necessary.

For any collection of agents' scenarios, it clearly holds that $d_{i,N} \leq d$ for all $i = 1, \ldots, m$, for any scenario set. Thus, for each $i = 1, \ldots, m$, Theorem 2.5 can be applied conditionally to the scenarios of all other agents to obtain a local, in the sense that it holds only for the constraints of agent i, feasibility characterization. Fix $\beta_i \in (0, 1)$ and let

$$\widetilde{\varepsilon}_i = 1 - \sqrt[N_i - d]{\frac{\beta_i}{\binom{N_i}{d}}}. \tag{47}$$

We then have that

$$\mathbb{P}^N\Big\{S \in \Delta^N : \ \mathbb{P}\Big\{\delta \in \Delta : \ x_N^* \notin X_i(\delta)\Big\} \leq \widetilde{\varepsilon}_i\Big\} \geq 1 - \beta_i. \tag{48}$$

By the subadditivity of $\mathbb{P}^N$ and $\mathbb{P}$, (48) can be used to quantify the probabilistic feasibility of x_N^* with respect to the global constraint $\bigcap_{i=1}^m X_i(\delta)$. Following the proof of Corollary 1 in [59], where a similar argument is provided, in [4] the following proposition is proven.

Proposition 2.1 *Fix $\beta \in (0, 1)$ and choose β_i, $i = 1, \ldots, m$, such that $\sum_{i=1}^m \beta_i = \beta$. For each $i = 1, \ldots, m$, let $\widetilde{\varepsilon}_i$ be as in (47) and set $\widetilde{\varepsilon} = \sum_{i=1}^m \widetilde{\varepsilon}_i$. We then have that*

$$\mathbb{P}^N\Big\{S \in \Delta^N : \ \mathbb{P}\Big\{\delta \in \Delta : \ x_N^* \notin \bigcap_{i=1}^m X_i(\delta)\Big\} \leq \widetilde{\varepsilon}\Big\} \geq 1 - \beta. \tag{49}$$

Proposition 2.1 implies that with confidence at least $1 - \beta$, x_N^* is feasible for $\mathcal{P}_\delta$ apart from a set with measure at most $\widetilde{\varepsilon}$. This result, however, tends to be very conservative thus

prohibiting its applicability to problems with a high number of agents. This can be seen by comparing $\widetilde{\varepsilon}$ with $\bar{\varepsilon}$, where the latter corresponds to the case where scenarios are treated as a common resource. To this end, consider the particular set-up where $N_i = \bar{N}$ and $\beta_i = \beta/m$, for all $i = 1, \ldots, m$. By (43) and (47), it follows that $\widetilde{\varepsilon} = m\widetilde{\varepsilon}_i \approx m\bar{\varepsilon}$, thus growing approximately (we do not have exact equality since $\beta_i = \beta/m$) linearly with the number of agents. The issue with Proposition 2.1 is that it accounts for a worst-case setting, where $d_{i,N} = d$ for all $i = 1, \ldots, m$; however, this can not occur, since $\sum_{i=1}^{m} d_{i,N} \leq d$ implies that if $d_{i,N} = d$ for some i, then $d_{j,N} = 0$, for all $j \neq i$, $i = 1, \ldots, m$.

To alleviate the conservatism of Proposition 2.1, and exploit the fact that $\sum_{i=1}^{m} d_{i,N} \leq d$, we apply the following reasoning, which strongly depends on the recent results of [56].

For each $i = 1, \ldots, m$, fix $\beta_i \in (0, 1)$ and consider a function $\varepsilon_i(\cdot)$ defined as follows:

$$\varepsilon_i(k) = 1 - \sqrt[N_i - k]{\frac{\beta_i}{(d+1)\binom{N_i}{k}}}, \text{ for all } k = 0, \ldots, d. \tag{50}$$

Notice that $\varepsilon_i(\cdot)$ is also a function of N_i, β_i and d, but this dependency is suppressed to simplify notation. For each $i = 1, \ldots, m$, working conditionally with respect to the scenarios $S \setminus S_i$ of all other agents, Theorem 1 of [56] entails that

$$\mathbb{P}^N\Big\{S \in \Delta^N : \mathbb{P}\Big\{\delta \in \Delta : x_N^* \notin X_i(\delta)\Big\} \leq \varepsilon_i(d_{i,N})\Big| \left\{S \setminus S_i \in \Delta^{N-N_i}\right\}\Big\} \geq 1 - \beta_i. \tag{51}$$

Integrating (51) with respect to the probability of realizing the scenarios $S \setminus S_i$, if $\varepsilon_i(\cdot)$ is set according to (50), we have that

$$\mathbb{P}^N\Big\{S \in \Delta^N : \mathbb{P}\Big\{\delta \in \Delta : x_N^* \notin X_i(\delta)\Big\} \leq \varepsilon_i(d_{i,N})\Big\} \geq 1 - \beta_i. \tag{52}$$

The statement in (52) implies that for each agent $i = 1, \ldots, m$, with confidence at least $1 - \beta_i$, the probability that x_N^* does not belong to the constraint set $X_i(\delta)$ of agent i is at most equal to $\varepsilon_i(d_{i,N})$.

Note, however, that (52) is very different from (48), which is obtained by means of the basic scenario approach theory, since $d_{i,N}$ is not known a-priori but depends on the extracted scenarios. Using (52) in place of (48), by the subadditivity of $\mathbb{P}^N$ and $\mathbb{P}$ in [4] it is proven that

$$\mathbb{P}^N\Big\{S \in \Delta^N : \mathbb{P}\Big\{\delta \in \Delta : x_N^* \notin \bigcap_{i=1}^{m} X_i(\delta)\Big\} \leq \sum_{i=1}^{m} \varepsilon_i(d_{i,N})\Big\} \geq 1 - \sum_{i=1}^{m} \beta_i. \tag{53}$$

Unlike (44) and (49), (53) is an a-posteriori statement due to the dependency of $\varepsilon_i(d_{i,N})$ on the extracted scenarios. However, the sought a-priori result can be obtained by considering

the worst-case value for $\sum_{i=1}^m \varepsilon_i(d_{i,N})$, with respect to the different combinations of $d_{i,N}$, $i = 1, \ldots, m$, satisfying $\sum_{i=1}^m d_{i,N} \leq d$. This can be achieved by means of the following maximization problem:

$$\begin{aligned} \varepsilon = \max_{\{d_i \in \mathbb{N}_+\}_{i=1}^m} & \sum_{i=1}^m \varepsilon_i(d_i) \\ \text{subject to } & \sum_{i=1}^m d_i \leq d, \end{aligned} \tag{54}$$

Problem (54) is an integer optimization program. It can be solved numerically to obtain ε. The optimal value ε of the problem above depends on $\{N_i, \beta_i\}_{i=1}^m$ and d, but this dependency is suppressed to simplify notation. Notice the slight abuse of notation, since $\{d_i\}_{i=1}^m$ in (54) are integer decision variables and should not be related to $\{d_{i,N}\}_{i=1}^m$.

Theorem 2.6 *Fix $\beta \in (0,1)$ and choose β_i, $i = 1, \ldots, m$, such that $\sum_{i=1}^m \beta_i = \beta$. Set ε according to* (54)*. We then have that*

$$\mathbb{P}^N\Big\{S \in \Delta^N : \ \mathbb{P}\Big\{\delta \in \Delta : \ x_N^* \notin \bigcap_{i=1}^m X_i(\delta)\Big\} \leq \varepsilon\Big\} \geq 1 - \beta. \tag{55}$$

The result of Theorem 2.6 can be significantly less conservative compared to that of Proposition 2.1, since we explicitly account for the fact that $\sum_{i=1}^m d_{i,N} \leq d$ in the maximization problem in (54). This can be also observed by means of the numerical example of Fig. 20, where we investigate how $\bar{\varepsilon}$, $\widetilde{\varepsilon}$ and ε change as a function of the number of agents m. We consider a particular case where $d = 50$, $\beta = 10^{-6}$, $N_i = \bar{N} = 4500$ and $\beta_i = \beta/m$, for all $i = 1, \ldots, m$. For this set-up, where β is split evenly among agents and all agents have the same number of scenarios, it turned out that the maximum value ε in (54) is achieved for $d_i = d/m$, $i = 1, \ldots, m$.

In certain cases (e.g., when the number of agents is high) ε may still exceed one and hence the result of Theorem 2.6 becomes trivial (the same for Proposition 2.1 in such cases). Theorem 2.6 can be reversed to compute the number of scenarios N_i that need to be extracted by agent i, $i = 1, \ldots, m$, for a given value of $\varepsilon, \beta \in (0,1)$. This can be achieved by numerically seeking for values of N_i, $i = 1, \ldots, m$, that lead to a solution of (54) that attains the desired ε.

2.6 Concluding remarks

We developed distributed optimization approaches for a large scale system composed of many subsystems (agents) whose decisions are coupled. The proposed approaches can cope with *i)*

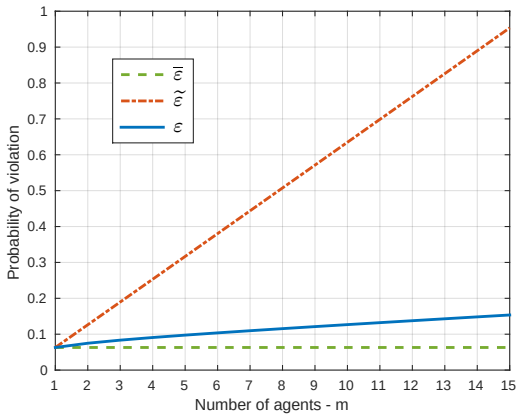

Figure 20: Probability of constraint violation as a function of the number of agents m, for the case where $d = 50$, $\beta = 10^{-6}$, $N_i = \bar{N} = 4500$ and $\beta_i = \beta/m$, for all $i = 1, \ldots, m$. Comparison among the probability of violation $\bar{\varepsilon}$ (green dashed line) in (43), $\widetilde{\varepsilon}$ (red dotted-dashed line) in (47), and ε (blue solid line) obtained by solving problem (54).

heterogeneity of the subsystems by allowing different local objectives and constraints coding physical and/or technological limits, *ii)* locality of information and privacy, in that each agent communicates with its neighbors and shares with them only its tentative value for some global optimization variable that needs to be agree upon, and *iii)* uncertainty affecting each agent dynamics. We hence capture the compositional nature of cyber-physical systems sharing some common resource.

Inspired by [60], we are currently addressing decentralized optimization for solving Mixed Integer Linear Program (MILP). This is useful for control of switched linear systems with discrete and continuous inputs modeling smart grids in the energy management applications described in Deliverable 5.1 and [61], which are part of the case studies of work package 5. Companies involved in smart grid energy management can then benefit from the results of work package 2.

Note that the proposed methods can solve problems involving a discrete-time dynamics by reformulating them as static problems where the dynamics is propagated in time and the system state is expressed as a function of the initial condition and the decision variables/inputs. In such a case, one can adopt a receding horizon implementation of the distributed optimization strategy, by re-initializing at each step the system at the current state and re-running the distributed optimization algorithm. This would lead to a distributed MPC approach to the optimal constrained control of a networked discrete-time system. However, it will also bring in challenges related to real-time computability, as discussed in Section 4 of this deliverable.

3 Distributed Model Predictive Control (DMPC)

The present section and Section 4 consider settings of distributed model predictive control (DMPC), in which the local optimization problems are carried out in a decentralized fashion, while information on planned trajectories (including costs) and/or on the interaction of the subsystem dynamics is communicated through the network. A number of variants of DMPC have been proposed in the past, mainly focussing on linear dynamics, see. e.g. [62–67]. Aspects which have been hardly considered in research of DMPC so far, but which are particularly important in the context of cyber-physical systems are the following:

- to account for heterogenous dynamics in the sense that the state evolution depends on the specific mode of operation (which changes over time), thus requiring the consideration of hybrid dynamics,
- to model that the constraints imposed by the environment on the evolution of the subsystem state may vary over time,
- and that the optimization problems to be solved within the DMPC require sufficiently small computation times to allow online execution.

These are the challenges which are addresses in work package 2 of *UnCoVerCPS*, and which determine the content of this section and Section 4 of this report. Accordingly, we start from a method for DMPC of piecewise-affine systems as described in [68] in an initial form, and extend the method to cover time-varying constraints, to include discrete inputs, and to embed mechanisms for reducing computation times, and thus to enhance the scalability. The method described in the following is the first one to propose a DMPC scheme for switching affine system such that stabilization is ensured.

3.1 Problem Setup

According to [68], consider N_l dynamically decoupled hybrid subsystems Σ_i, $i \in \{1, \dots, N_l\} =: \mathcal{N}$ with discrete-time affine dynamics on a polyhedral partition of the continuous state space:

$$x_{k+1}^i = \underbrace{A_{p^i}^i x_k^i + B_{p^i}^i u_k^i + f_{p^i}^i, \quad \text{if } x_k^i \in \mathcal{P}_{p^i}^i}_{=:g^i(x_{k+l}^i, u_{k+l}^i)}, \tag{56}$$

where $x_k^i \in \mathbb{R}^{n^i}$ and $u_k^i \in \mathbb{U}^i \subseteq \mathbb{R}^{m^i}$ are the local states and inputs of Σ_i; $\mathcal{P}_{p^i}^i$ is a convex polyhedral region of the partitioned state-space of Σ_i, and the affine dynamics parametrized by $A_{p^i}^i$, $B_{p^i}^i$ and $f_{p^i}^i$ is valid on the region with index $p^i \in \{1, \dots, N_p^i\}$. The global state and input

vectors are given by $x_k = [(x_k^1)^T, \ldots, (x_k^{N_l})^T]^T \in \mathbb{R}^n$ and $u_k = [(u_k^1)^T, \ldots, (u_k^{N_l})^T]^T \in \mathbb{R}^m$. Coupling is induced by a global cost function for a finite horizon of $N \geq 2$ time steps:

$$J(\mathbf{x}_k, \mathbf{u}_k) = \|x_{k+N|k}\|_P + \sum_{l=0}^{N-1} \|x_{k+l|k}\|_Q + \|u_{k+l|k}\|_R,$$

where $x_{k+l|k}$ denotes the state at time $k+l$ predicted at time k and $\mathbf{u}_k^i = [(u_{k|k}^i)^T, \ldots, (u_{k|k+N-1}^i)^T]^T \in \mathbb{R}^{Nm^i}$ and $\mathbf{x}_k^i = [(x_{k|k}^i)^T, \ldots, (x_{k|k+N}^i)^T]^T \in \mathbb{R}^{(N+1)n^i}$ denote the sequence of inputs and states of the subsystem Σ_i over the horizon. The global state and input vector over the horizon are given by $\mathbf{x}_k = [(\mathbf{x}_k^1)^T, \ldots, (\mathbf{x}_k^{N_l})^T]^T$ and $\mathbf{u}_k = [(\mathbf{u}_k^1)^T, \ldots, (\mathbf{u}_k^{N_l})^T]^T$. Let $\|x_{k+N|k}\|_P = x_{k+N|k}^T P x_{k+N|k}$, and the weighting matrices satisfy $Q = Q^T > 0$, $R = R^T > 0$, and $P = P^T > 0$. Further coupling is induced by interconnected state constraints $x_k \in \mathbb{X} \subseteq \mathbb{R}^n$. The polyhedral regions of the state space are defined by:

$$\mathcal{P}_{p^i}^i := \{x_k^i \in \mathbb{R}^{n^i} | C_{p^i}^i x_k^i \leq b_{p^i}^i\}, \tag{57}$$

with $C_{p^i}^i \in \mathbb{R}^{c_{p^i}^i \times n^i}$, $b_{p^i}^i \in \mathbb{R}^{c_{p^i}^i}$. The input constraints are given by:

$$\mathbb{U}^i := \{u_k^i \in \mathbb{R}^m | F^i u_k^i \leq h^i\}, \tag{58}$$

where $H^i \in \mathbb{R}^{n_h^i \times m^i}$, $h^i \in \mathbb{R}^{n_h^i}$ and $\mathbb{U} := \mathbb{U}^i \times \ldots \times \mathbb{U}^{N_l}$. In order to define the state constraints, let $X := \{x_k \in \mathbb{R}^n | E x_k \leq o\}$, with $E \in \mathbb{R}^{n_e \times n}$ and $o \in \mathbb{R}^{n_e}$ denote coupled convex state constraints. Furthermore, let $\tilde{X}_j := \{x_k \in \mathbb{R}^n | \tilde{E}_j x_k \leq \tilde{o}_j\}$, with $\tilde{E}_j \in \mathbb{R}^{n_{\tilde{e}_j} \times n}$ and $\tilde{o}_j \in \mathbb{R}^{n_{\tilde{e}_j}}$ denote regions which are excluded from the feasible set. Then, the overall state constraint is given by:

$$\mathbb{X} := (\cup_{p^1=1}^{N_p^1} \mathcal{P}_{p^1}^1 \times \ldots \times \cup_{p^{N_l}=1}^{N_p^{N_l}} \mathcal{P}_{p^{N_l}}^{N_l}) \cap (X \setminus \cup_j \tilde{X}_j). \tag{59}$$

Notice, that this formulation allows to approximate arbitrary non-convex constraints.

The following assumptions are made in order to establish feasibility and stability of the scheme:

Assumption 3.1 *It is assumed that:*

1. *Let $\mathcal{B}_r^n$ denote a closed ball of dimension n with radius $r > 0$ and center 0. $\mathcal{B}_\epsilon^m \subseteq \mathbb{U}$, $\mathcal{B}_\epsilon^n \subseteq \mathbb{X}$, and $\mathcal{B}_\epsilon^{n^i} \subseteq \mathcal{P}_1^i$, $\forall i \in \mathcal{N}$.*

2. *The pair (A_1^i, B_1^i) is stabilizable for all $i \in \mathcal{N}$.*

3. There exists a terminal control law:

$$u_k = K_1 x_k, \tag{60}$$

with $K_1 = blkdiag(K_1^1, \ldots, K_1^{N_l})$, $K_1^i \in \mathbb{R}^{m^i \times n^i}$, *and a decoupled terminal set* $\mathbb{X}_f = \mathbb{X}_f^1 \times \ldots \times \mathbb{X}_f^{N_l}$*:*

$$\mathbb{X}_f \subseteq (\mathcal{P}_1^1 \times \ldots \times \mathcal{P}_1^{N_l}) \cap (X \setminus \cup_j \tilde{X}_j), \tag{61}$$

and $P = blkdiag(P^1, \ldots, P^{N_l})$*, such that the following holds for all* $x_k \in \mathbb{X}_f$*:*

(i) $\|x_k\|_P - \|(A_1 + B_1 K_1)x_k\|_P \geq \|x_k\|_Q + \|K_1 x_k\|_R$,

(ii) $(A_1 + B_1 K_1)x_k \in \mathbb{X}_f$, $K_1 x_k \in \mathbb{U}$.

A1.1 and A1.2 ensure that the problem is well posed, and they ensure that a terminal control law and constraint satisfying A1.3 can be constructed.

The centralized MPC Problem is given by:

$$\begin{aligned}
(\mathbf{x}_k^*, \mathbf{u}_k^*) = \quad & \operatorname{argmin}_{\mathbf{x}_k, \mathbf{u}_k} J(\mathbf{x}_k, \mathbf{u}_k) \\
\text{s.t. } & x_{k+l+1|k}^i = g^i(x_{k+l|k}^i, u_{k+l|k}^i), \forall i \in \mathcal{N}, \forall l \in \{0, \ldots, N-1\}, \\
& u_{k+l|k} \in \mathbb{U},\ x_{k+l|k} \in \mathbb{X}, \qquad \forall l \in \{0, \ldots, N-1\}, \\
& x_{k+N|k} \in \mathbb{X}_f.
\end{aligned} \tag{62}$$

By introducing $N \sum_{i \in \mathcal{N}} N_p^i$ binary variables, which encode the dependancy of the dynamics $g^i(x_k^i, u_k^i)$ on the condition $x_k^i \in \mathcal{P}_{p^i}^i$, problem (62) can be formulated as a mixed-integer quadratic program (MIQP). While the resulting MIQP can be solved by techniques such as branch and bound, the computational complexity is high due to a large number of binary variables arising from the combinatorial nature of the dynamics (56). Thus, in order to apply MPC to distributed PWA systems, both the number of binary variables and the number of continuous variables have to be reduced by decomposing (62) into a set of smaller subproblems.

3.2 Scheme using decentralized optimization and communication

In a typical sequential scheme, subsystem Σ_i solves (62) for its own inputs $\mathbf{u}_{k|k}^i$ while keeping the inputs and states of the interconnected subsystems Σ_j, $\forall j \in \mathcal{N}_i^0 \setminus i$ constant. Then Σ_i communicates the resulting input sequence to all subsystems Σ_j, $\forall j \in \mathcal{N}_i^0 \setminus i$. Subsequently, the next subsystem in the sequence repeats the procedure (still in time k). Let τ_s denote an upper

bound on the computation time of one subsystem and τ_c an upper bound on the communication delay. Assuming that data can be communicated to all interconnected subsystems at once, the time required to optimize over the whole sequence is $N_l(\tau_s + \tau_c)$. Thus, one has to choose a sampling time $T > N_l(\tau_s + \tau_c)$, i.e. if a large number of subsystems is considered, the overall computation time may be problematic. In the scheme proposed by [69], some groups of subsystems may update their control inputs in parallel if the interconnections are sparse. However, each subsystem needs to send four messages to neighboring subsystems per time step in this scheme. Furthermore, it is not clear how to efficiently use the communication network and avoid simultaneous access to a shared network. For instance, in a wireless multi-hop network communication frequencies / channels are shared by the nodes / subsystems and collisions may arise if multiple subsystems access the network at the same time. These collisions can be avoided by suitable scheduling or arbitration protocols, which ensure that only one subsystem can access the network at a time. In the following, we will consider an arbitration protocol which grants network access to the subsystem with the largest decrease in the cost function.

Let $\mathcal{N}_i^0 \subseteq \mathcal{N}$ denote an index set containing the indices of the subsystem Σ_i and of all subsystems with which Σ_i is directly interconnected with, either by costs or constraints. In the distributed scheme, each subsystem Σ_i optimizes over its own inputs and that of directly interconnected subsystems, i.e. Σ_i optimizes over $\mathbf{u}_k^j$ for $j \in \mathcal{N}_i^0$. Hence, the resulting problem also depends on the state and input sequences of the subsystems Σ_s, $s \in \mathcal{N}_i^1 \setminus \mathcal{N}_i^0$ for $\mathcal{N}_i^1 := \cup_{j \in \mathcal{N}_i^0} \mathcal{N}_j^0$. These sequences are held constant during the local optimization. Thus, each subsystem needs to assume values for the inputs and states of other subsystems. For this purpose, we introduce local variables $u_{k+l|k}^{i,j}$ and $v_{k+l|k}^{i,j}$, where e.g. $u_{k+l|k}^{i,j}$ is the local value that Σ_i uses for $u_{k+l|k}^j$, $v_{k+l|k}^{i,j}$ denotes an optimized candidate input sequence, and we write $u_{k+l|k}^i = u_{k+l|k}^{i,i}$. The main idea is to solve the local problems in parallel and to synchronize the assumed and actual inputs of the subsystems by communication. Consider the following local optimization problem of subsystem Σ_i, which leads to a candidate solution $(\mathbf{z}_k^{i^*}, \mathbf{v}_k^{i^*})$, containing a locally optimized state and input sequence:

$$
\begin{aligned}
(\mathbf{z}_k^{i^*}, \mathbf{v}_k^{i^*}) = \quad & \operatorname{argmin}_{\mathbf{z}_k^i, \mathbf{v}_k^i} J(\mathbf{z}_k, \mathbf{v}_k) && (63)\\
\text{s.t. } z_{k+l+1|k}^{i,j} = & g^i(z_{k+l|k}^{i,j}, v_{k+l|k}^{i,j}), \forall j \in \mathcal{N}_i^0, \forall l \in \{0, \dots, N-1\},\\
& v_{k+l|k}^i \in \mathbb{U},\ z_{k+l|k}^i \in \mathbb{X}^i, && \forall l \in \{0, \dots, N-1\},\\
& z_{k+l|k}^{i,s} = x_{k+l|k}^{i,s},\ v_{k+l|k}^{i,s} = u_{k+l|k}^{i,s}, && \forall s \in \mathcal{N}_i^1 \setminus \mathcal{N}_i^0, \forall l \in \{0, \dots, N-1\},\\
& z_{k+l|k}^{i,q} = 0,\ v_{k+l|k}^{i,q} = 0, && \forall q \notin \mathcal{N}_i^1, \forall l \in \{0, \dots, N-1\},
\end{aligned}
$$

$$z_{k+N|k} \in \mathbb{X}_f,\ v^{i,j}_{k|k} = u^{i,j}_{k|k}, \qquad \forall j \in \mathcal{N}_i^0,$$

Here, $\mathbb{X}^i$ collects all constraints from $\mathbb{X}$ which involve $x_k^j,\ \forall j \in \mathcal{N}_i^0$. In this problem, inputs and states of $\Sigma_s,\ s \in \mathcal{N}_i^1$ are fixed to the previous solution. Because only the hybrid dynamics $g^i(x_k^i, u_k^i),\ i \in \mathcal{N}_j^0$ have to be considered the number of binary variables is reduced to $N \sum_{j \in \mathcal{N}_j^0} N_p^j$. This results in a large reduction of computational complexity, if the interactions are sparse. Furthermore, the optimizer $(\mathbf{z}_k^{i^*}, \mathbf{v}_k^{i^*})$ does not depend on states and inputs of $\Sigma_\zeta,\ \zeta \notin \mathcal{N}_i^1$. Thus, these states and inputs are set to 0 (which is feasible by Assumption 3.1.1 and the construction of $\mathbb{X}^i$) and do not need to be communicated to subsystem Σ_i. Let $\mathbf{v}_k^{i,j} := [(v^{i,j}_{k+l|k})^T, \ldots, (v^{i,j}_{k+N-1|k})^T]^T$ and $\mathbf{z}_k^{i,j} := [(z^{i,j}_{k+l|k})^T, \ldots, (z^{i,j}_{k+N|k})^T]^T$ again denote sequences over the prediction horizon, and define $\mathbf{v}^i := [(\mathbf{v}_k^{i,1})^T, \ldots, (\mathbf{v}_k^{i,N_l})^T]^T$ and $\mathbf{z}^i := [(\mathbf{z}_k^{i,1})^T, \ldots, (\mathbf{z}_k^{i,N_l})^T]^T$. In order to compensate for computation and communication delays, the last constraint fixes the current input to the initial solution, i.e. the current input is not optimized. After the optimization is performed in parallel, each subsystem Σ_i computes whether the cost decrease by the local candidate solution is larger than a threshold $\gamma > 0$, and which subsystem offers the largest decrease. Note that γ can be used to establish a trade-off between communication and closed-loop performance, since updates which improve the cost by less than γ are not communicated and are discarded. The cost before a communication event by Σ_i is given by $J(\bar{\mathbf{x}}_k^i, \bar{\mathbf{u}}_k^i)$, where $\bar{\mathbf{x}}_k^i$ and $\bar{\mathbf{u}}_k^i$ are defined the same way as $\mathbf{v}^i$ and $\mathbf{z}^i$, and the components are given by:

$$\begin{aligned} &\bar{x}^{i,s}_{k+l|k} = x^{i,s}_{k+l|k}, \quad \bar{u}^{i,s}_{k+l|k} = u^{i,s}_{k+l|k}, && \forall s \in \mathcal{N}_i^1, \\ &\bar{x}^{i,\zeta}_{k+l|k} = 0, \qquad\ \ \bar{u}^{j,\zeta}_{k+l|k} = 0, && \forall \zeta \notin \mathcal{N}_i^1. \end{aligned} \tag{64}$$

The cost after a communication event by Σ_i is given by $J(\hat{\mathbf{x}}_k^i, \hat{\mathbf{u}}_k^i)$, and the components of $\hat{\mathbf{x}}_k^i$ and $\hat{\mathbf{u}}_k^i$ are:

$$\begin{aligned} &\hat{x}^{i,j}_{k+l|k} = z^{i,j}_{k+l|k}, \quad \hat{u}^{i,j}_{k+l|k} = v^{i,j}_{k+l|k}, && \forall j \in \mathcal{N}_i^0, \\ &\hat{x}^{i,s}_{k+l|k} = x^{i,s}_{k+l|k}, \quad \hat{u}^{i,s}_{k+l|k} = u^{i,s}_{k+l|k}, && \forall s \in \mathcal{N}_i^1 \setminus \mathcal{N}_i^0, \\ &\hat{x}^{i,\zeta}_{k+l|k} = 0, \qquad\ \ \hat{u}^{i,\zeta}_{k+l|k} = 0, && \forall \zeta \notin \mathcal{N}_i^1. \end{aligned} \tag{65}$$

These costs are compared according to:

$$J_d^i = \begin{cases} J(\bar{\mathbf{x}}_k^i, \bar{\mathbf{u}}_k^i) - J(\hat{\mathbf{x}}_k^i, \hat{\mathbf{u}}_k^i) & \text{if } \hat{\mathbf{x}}_k^i \in \mathbb{X}^i \times \ldots \times \mathbb{X}^i, \\ -\infty & \text{otherwise,} \end{cases} \tag{66}$$

Algorithm 7 DMPC Algorithm

Initialization: $\mathbf{u}_0^i$, $\mathbf{x}_0^i$, for all $i \in \mathcal{N}$.
while $k \geq 0$ **do**
  Each subsystem Σ_i applies $u_{k|k}^{i,i}$.
  Solve (63) in parallel for all $i \in \mathcal{N}$, and set $\Omega_k := \mathcal{N}$.
  while $t < (k+1)T - \tau_c$ and $stop = 0$ **do**
    Each Σ_i computes J_d^i according to (66) and arbitration leads to i^* based on (67).
    if $i^* = \emptyset$ **then**
      $stop = 1$
    else
      Σ_{i^*} sends $(\mathbf{z}_{k|k}^{i^*}, \mathbf{v}_{k|k}^{i^*})$ to all Σ_j, $j \in \mathcal{N}_{i^*}^1$.
      for $j \in \mathcal{N}$ and all $r \in \mathcal{N}$ **do**
        $u_{k+l|k+1}^{j,r} := v_{k+l|k}^{i^*,r}$, $\forall l \in \{1, \ldots, N-1\}$,
        $x_{k+l|k+1}^{j,r} := z_{k+l|k}^{i^*,r}$, $\forall l \in \{2, \ldots, N-2\}$.
      end for
      Set $\Omega_k := \Omega_k \setminus i^*$.
    end if
  end while
  Each Σ_i computes for all $r \in \mathcal{N}_i^1$:

$$
\begin{aligned}
u_{k+l|k+1}^{i,r} &:= u_{k+l|k}^{i,r}, \ \forall l \in \{1, \ldots, N-2\} \\
x_{k+l|k+1}^{i,r} &:= x_{k+l|k}^{i,r}, \ \forall l \in \{1, \ldots, N-1\} \\
u_{k+N-1|k+1}^{i,r} &:= K_1^r x_{k+N-1|k+1}^{i,r}, \\
x_{k+N|k+1}^{i,r} &:= (A_1^r + B_1^r K_1^r) x_{k+N-1|k}^{i,r},
\end{aligned}
$$

  and sets $k := k+1$.
end while

i.e. an infeasible update results in infinite costs. Subsequently, the subsystem Σ_{i^*} with

$$
i^* := \underset{i \in \Omega_k, J_d^i \geq \gamma}{\operatorname{argmax}} \ J_d^i, \tag{67}
$$

is granted access to the network. The set Ω_k is the index set of subsystems which have not yet communicated in time k. This procedure may be repeated, if enough time is left before the next sampling instant.

The overall scheme is given by Algorithm 7, where t denotes the current time, and k the current time step. If i^* is not unique, a given priority may be used for arbitration. The subsystem Σ_{i^*} communicates its optimized input and state sequence to Σ_j, $j \in \mathcal{N}_{i^*}^1$, which update the local input and state sequences $u_{k+l|k}^{i,j}$ and $x_{k+l|k}^{i,j}$ and repeat the arbitration scheme. Note that this scheme is similar to the well known try-once-discard (TOD) protocol and can be implemented in a wide range of shared and distributed communication networks. For instance, [70] proposed a method to implement such an arbitration scheme in multi-hop

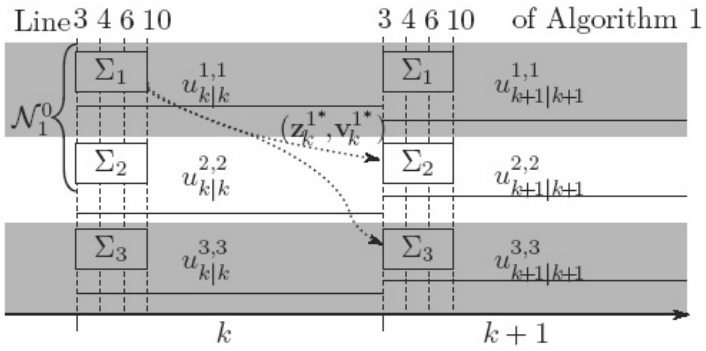

Figure 21: Steps from Algorithm 7 (dashed) and communication (dotted, shown for Σ_1).

wireless networks with shared communication channels.

Note that the set Ω_k is not required to actually implement the scheme; it is only used to specify that a system Σ_i does not participate in the arbitration phase, if it already communicated its candidate solution. The scheme is illustrated in Figure 21. Similarly to sequential schemes, a feasible initialization for all Σ_j, $j \in \mathcal{N}_i^1$ has to be assigned to each subsystem Σ_i. Within the scheme, feasibility of a combination of the current state and input sequences and local candidate solutions is checked and combinations which are either not feasible or do not lead to a cost reduction are discarded (similar to ([69])). It should be noted, that two subsystems Σ_i and Σ_j with $\mathcal{N}_i^1 \cap \mathcal{N}_j^1 = \emptyset$ could update at the same time without causing infeasibility. In particular, this implies that Σ_i may communicate a feasible update as long as $\mathcal{N}_i^1 \cap \mathcal{N}_j^1 = \emptyset$ for all $j \in \Omega_k$.

Theorem 1 *Suppose that Assumption 3.1 is satisfied and a feasible initialization exists. Then, Algorithm 7 ensures feasibility for all times, and the overall system is asymptotically stable in closed loop with the DMPC-Scheme.*

The reader is referred to [68] for the proof of this result.

3.3 Numerical Example

As an example, consider the problem of controlling a platoon of $N_l = 7$ identical vehicles, a version of the automotive use case in UnCoVerCPS. The vehicles are modeled by switched second order dynamics with three regions $\mathcal{P}_{p^i}^i$, modeling approximately changes of the dynamics due to shifting gears and nonlinear effects. The states are given by $x_k^i = \begin{bmatrix} \nu^i - S_d(i-1) & \dot{\nu}^i \end{bmatrix}^T$, where ν^i is the position, $\dot{\nu}^i$ the velocity, and S_d the desired spacing between vehicles. Thus, $\nu_k^{i+1} - \nu_k^i = S_d$ is equal to $x_k^{i+1} - x_k^i = 0$. We choose $S_d = 5$ which corresponds to a constant spacing of 5m. The cost is formulated such that the distance of the first vehicle from the desired

position and the spacing between each vehicle and its follower are penalized. Furthermore, the problems are interconnected by collision avoidance constraints between subsequent vehicles, i.e. $x_k^{i+1} - x_k^i > -5$. The coupling structure is given by e.g. $\mathcal{N}_1^0 := \{1, 2\}$, $\mathcal{N}_2^0 := \{1, 2, 3\}$, $\mathcal{N}_3^0 := \{2, 3, 4\}$, etc. The dynamics is parametrized by:

$$A_1^i = \begin{bmatrix} 1 & 1 \\ 0 & 0.95 \end{bmatrix},\ B_1^i = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix},\ f_1^i = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$A_2^i = \begin{bmatrix} 1 & 1 \\ 0 & 0.75 \end{bmatrix},\ B_2^i = \begin{bmatrix} 0.4 \\ 0.8 \end{bmatrix},\ f_2^i = \begin{bmatrix} 0 \\ 1.11 \end{bmatrix},$$
$$A_3^i = \begin{bmatrix} 1 & 1 \\ 0 & 0.75 \end{bmatrix},\ B_3^i = \begin{bmatrix} 0.25 \\ 0.5 \end{bmatrix},\ f_3^i = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

with $\mathcal{P}_1^i = \{x_k^i \in \mathbb{R}^{n^i} | -0.278 \leq [0\ 1] x_k^i \leq 5.55\}$, $\mathcal{P}_2^i = \{x_k^i \in \mathbb{R}^{n^i} | 5.55 \leq [0\ 1] x_k^i \leq 27.78\}$, and $\mathcal{P}_3^i = \{x_k^i \in \mathbb{R}^{n^i} | -5.55 \leq [0\ 1] x_k^i \leq -0.278\}$ for all Σ_i, $i \in \mathcal{N}$. Simulation results for $N = 30$ and $\gamma = 1$ are shown in Figure 22, where the subplots show the position, velocity, and input of each vehicle. The threshold $\gamma = 1$ was chosen to closely emulate the behavior of time-triggered communication and the scheme is initialized with a suboptimal centralized solution. The last plot shows the number of communication events per time step, which was limited to three. It can be seen that the vehicles cooperate to reach the desired spacing. For example, Σ_1 (blue) does not accelerate strongly until ca. $k = 7$, thereby allowing Σ_2 (green) to reach the desired spacing. For $k \leq 11$ the subsystems communicate frequently, but for $k > 11$ the local solutions do not sufficiently improve the overall cost and are no longer communicated. Nonetheless, the vehicles reach the desired position and no collisions between vehicles occur. Using CPLEX 12 on an AMD Phenom II X4 920 with 4 GB RAM, the computation times for the local problems (63) range from 0.1s to 3s, the sum of local computation times is between 1s and 10s, and a comparable centralized problem typically required between 3s and 100s. In order to allow for real-time operation, shorter prediction horizons, or longer sampling intervals of the subproblems may be used.

Note that, in comparison to other existing techniques for vehicle platooning, the consideration of switching dynamics as well as the distributed online optimization of the driving behavior are new contributions. Two aspects should be observed, however, in the modeling of this example (and the method illustrated before): firstly, the change of gears is bound to the partitioning of the state space, i.e. the system switches autonomously between the different affine dynamics. An interesting question is, how the switching can be handled if it constitutes a degree of freedom of the system evolution and thus needs to be determined by

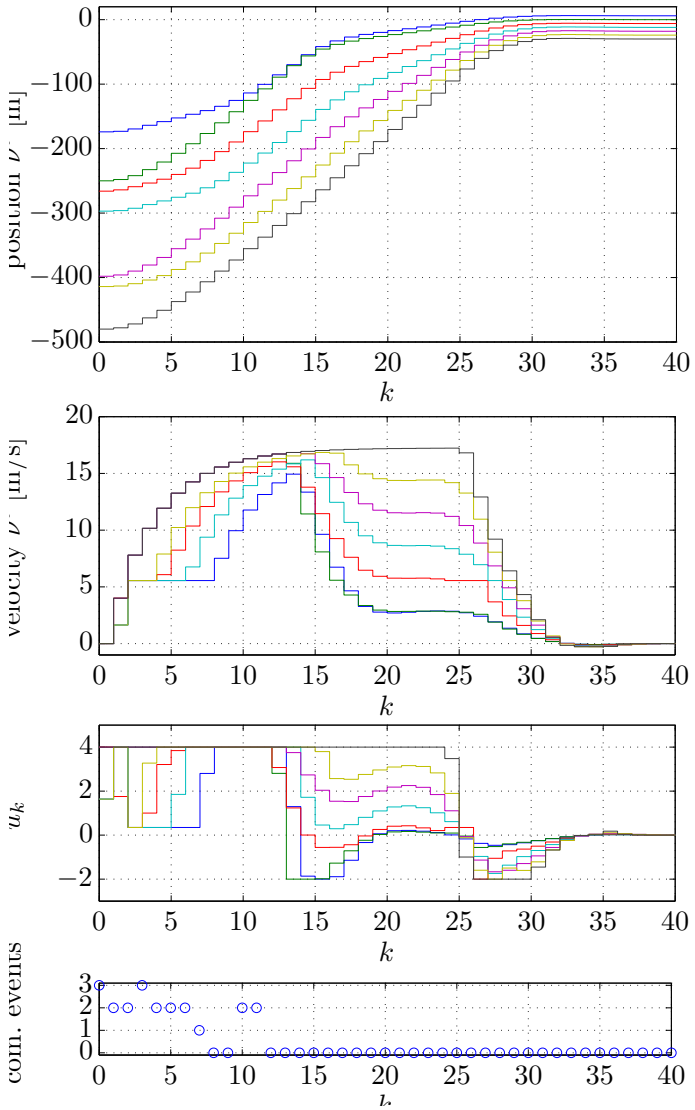

Figure 22: Simulation results for a platoon of vehicles.

a controller. Secondly, the partitioning of the state space and hence the state constraints are time-invariant. Particularly for CPS which operate in uncertain environments, the state constraints will arise from current measurements, i.e. they will be modeled as time-varying constraints. These two extensions were developed within work package 2 and are investigated in the following section.

4 Enhancing Real Time Computability

4.1 Introduction

The effectiveness of a Model Predictive Control (MPC) scheme for online control design is inherently bound to the question whether the optimization in each MPC iteration can be completed timely. This question is particularly relevant to distributed MPC for cyber-physical systems. The idea is therefore to explore different techniques to enhance the computational efficiency in determining suboptimal and approximated solutions. While a certain (small) loss of performance seems acceptable in favor of an enhanced real-time computability, the requirement of maintaining stability obviously has to be kept.

For those variants of CPS, in which the switching of the dynamics exists as a degree of freedom of the controller and/or the state constraints are varying over time, the control task is obviously more challenging than the instance of DMPC for switching systems, as covered in the previous section. This, in turn, makes the provision of techniques that provide the control inputs with low computational effort (and thus timely in a real-time setting) even more important. The next section, addresses this problem for a DMPC setting, in which the subsystem dynamics comprises discrete and continuous degrees of freedom. The optimization within the local MPC of any subsystem fixes these degrees of freedom (subject to constraints imposed / communicated by other subsystems) such that the optimal solution is approximated by relatively low effort. Thus, this approach extends existing variants of DMPC with coupling over constraints (as the method presented in the previous section) to mixed inputs, while at the same time a solution to the problem of increased complexity is proposed. With respect applications, the use of mixed inputs is valuable for any system in which actuators coexist that partly operate on continuous ranges, and partly on a discrete set of choices. An example for this case is a platoon of vehicles for which acceleration and gear can be controlled simultaneously (as described in the following part).

Section 4.3 then presents an alternative method tailored to low computation times specifically for time-varying state constraints (while focussing on continuous controls only). The main idea there is to determine in very short time for the local subsystem control an optimized state trajectory that stays feasible with respect to the changing constraints. This technique is based on the concept of homotopic functions, and it derives suitable trajectories online from optimal solutions computed prior to start of system operation. The method is suitable for applications in which the space available for trajectory planning may change rapidly. An example (related to the robotic use case of the project) is the control of a robotic manipulator

which operates in the same space as a human, such that collision avoidance is mandatory implying that the controller has to adapt fastly to the motion of the human.

4.2 Fast DMPC for Switched Systems with Mixed Inputs

As in any constrained optimization problem, the complexity of the local optimization of a subsystem in a DMPC scheme (as the one in Section 3) is determined by the number and type of the degrees of freedom, the number of constraints, and the properties of the functions occuring in the cost functional and the constraints. For CPS with hybrid dynamics, the combinatorics arising from the discrete-event part often has a dominating effect on the complexity and thus the computational effort. If the local dynamics of the subsystems are specified by switched dynamic systems, optimization problems of the class mixed-integer (non)linear programming (MINLP) are typically obtained, which are known to be NP hard, see e.g. [71, 72]. The use of relaxations (i.e. temporarily treating an integer variable $v \in \{0, 1\}$ as a continuous one $v \in [0, 1]$) is an established method to generate lower cost bounds when exploring and pruning the tree of integer variables, and it has been used in a number of approaches for solving control problems for switched systems [73–76]. While for some problem instances relaxations may lead to satisfactory results (in terms of efficiently pruning the search tree), the opposite effect can occur for switched systems in some cases: using $v \in [0, 1]$ instead of $v \in \{0, 1\}$ may mean to average between two distinct dynamics, leading to a system evolution which is not possible for the switched system. A gross under-estimation of a bound may result, i.e. many nodes of the search tree may be explored which later turn out to be infeasible. If the solution is approached by existing solvers for MINLP problems, such as 'BONMIN' [77] or 'BNB' [78], it can be observed that typically the computation times increase quickly with a growing number of integer variables, and, in addition, there is no guarantee that global optima are determined. These shortcomings motivate the investigation of techniques that do not not rely on relaxations of binary variables.

In this work, local online predictive controllers for switched systems are synthesized. The finite horizon optimal control problems consider time-varying state constaints, which stem from the predicted and communicated trajectories of interacting subsystems. In addition, input constraints are considered, and the method aims at a suitable compromise between control performance and computational effort. Several examples illustrate that by applying the proposed technique, real-time computability is significantly enhanced compared to standard mixed-integer solutions.

4.2.1 Problem Formulation

The class of models studied in this section are switched systems as, e.g., defined in [79]. For $k \in \mathbb{N}^{\geq 0}$, consider the discrete-time model:

$$\begin{aligned} x_{k+1} &= A_{v_k} x_k + B_{v_k} u_k \\ x_k &\in X_k,\ u_k \in U,\ \ v_k \in V \end{aligned} \tag{68}$$

where $x_k \in \mathbb{R}^{n_x \times 1}$ is the continuous state bound to time-varying constraints X_k, The vector $u_k \in \mathbb{R}^{n_u \times 1}$ denotes the continuous input selected from an invariant input space U, and v_k is the discrete input determining the parametrization of the continuous dynamics. The latter is chosen from the set $V = \{1, \ldots, n_v\}$, and any $v_k \in V$ corresponds to a pair of matrices (A_{v_k}, B_{v_k}). For a finite $N \in \mathbb{N}$, a given $x_k \in X_k$, let a sequence of state constraints $(X_k, X_{k+1}, \ldots, X_{k+N})$ be given. Let sequences of continuous inputs $\phi^u_{k,N} = (u_k, \ldots, u_{k+N-1})$ and discrete inputs $\phi^v_{k,N} = (v_k, \ldots, v_{k+N-1})$ be selected, for which a resulting sequence of continuous states is $\phi^x_{k,N} = (x_{k+1}, \cdots, x_{k+N})$ satisfies for $j \in \{0, \cdots, N-1\}$:

$$\begin{aligned} &x_{(k+j+1)} = \\ &\prod_{l=0}^{j} A_{v_{(k+l)}} \cdot x_k + \sum_{l=0}^{j} [(\prod_{t=0}^{j-l-1} A_{v_{(k+j-t)}}) \cdot B_{v_{(k+l+1)}} \cdot u_{(k+l)}]. \end{aligned} \tag{69}$$

The control objective is to drive x_k into a target state $x_f \in X_T = X_N$ within the N steps, while minimizing the cost function:

$$\begin{aligned} &\Omega(x_k, x_f, \phi^v_{k,N}, \phi^u_{k,N}) := (x_{k+N} - x_f)^{\mathrm{T}}_N Q_f (x_{k+N} - x_f) \\ &+ \sum_{j=0}^{N-1} \underbrace{(x_{k+j} - x_f)^{\mathrm{T}} Q_1 (x_{k+j} - x_f) + u^{\mathrm{T}}_{k+j} Q_2 u_{k+j}}_{\textit{Step cost } \mathcal{L}(x_{k+j}, u_{k+j})} \\ &\text{subject to: } (69), u_{k+j} \in U, v_{k+j} \in V, x_{k+j+1} \in X_{k+j+1}\ \forall j \in \{0, \ldots, N-1\}, \end{aligned} \tag{70}$$

with weighting matrices $Q_1 = {Q_1}^{\mathrm{T}} \geq 0$, $Q_2 = {Q_2}^{\mathrm{T}} > 0$ and $Q_f = {Q_f}^{\mathrm{T}} > 0$. If N has to be chosen large in order to enable $x_f \in X_N$, a large combinatorial complexity arises from the discrete inputs $({n_v}^N)$. If, in addition, the state constraints X_k can only be measured online and predicted a few time steps ahead, the solution on a moving and comparatively short horizon (as in MPC), is a self-evident option.

The substitute problem to be solved for any $k \geq 0$ until $x_k \in X_T$ and for a prediction horizon

$H \in \mathbb{N}$, $H < N$ is then:

$$\begin{aligned} &\min_{\begin{bmatrix}\phi^v_{k+j,H}\\ \phi^u_{k+j,H}\end{bmatrix}} \quad \Omega(x_k, x_f, \phi^v_{k,k+H}, \phi^u_{k,k+H}) \\ \text{s.t.:} \quad &(69), x_{k+j|k} \in X_{k+j|k},\ j \in \{1, \dots, H\} \\ &u_{k+j|k} \in U, v_{k+j|k} \in V,\ j \in \{0, \dots, H-1\}, \end{aligned} \tag{71}$$

where $u_{k+j|k}$, $v_{k+j|k}$ denote the continuous/discrete input computed at time k for time $k+j$. Note that the predicted state constraint $X_{k+j|k}$ may be different to X_{k+j} in the original problem. As the predicted horizon is selected to be shorter than in the original problem, a terminal weighting matrix $Q_H = {Q_H}^{\mathrm{T}} > 0$, $Q_H > Q_f$ with larger entries (compared to Q_f in (71)) is applied, in order to guarantee the following stability condition to hold:

$$\Omega^*(x_k, x_f, \phi^{v,*}_{k,H}, \phi^{u,*}_{k,H}) \geq \Omega^*(x_{k+1}, x_f, \phi^{v,*}_{k+1,H}, \phi^{u,*}_{k+1,H}). \tag{72}$$

As the continuous dynamics can be selected in any time step of the horizon, the set of choices exponentially increase with H, i.e., a total number of ${n_v}^H$ sequences ϕ_v are possible. Clearly, while complete enumeration would lead to the optimal solution, it renders the online solution infeasible in most cases. The optimization of ϕ^v (in conjunction to ϕ^u) can be treated by MINLP solvers. These provide locally optimal result, but the solution is typically not feasible in real-time for large H – solutions with small H may, however, violate the stability condition (72).

Hence, as an alternative, a tree search algorithm is proposed here to solve (71). It will be shown that the procedure limits the suboptimality and that it incurs computation times that are sufficiently low for many applications. The structure of the search algorithm is motivated by the one in [80], which operates on a tree in which a layer with index j represents the states reachable at time step $k+j$. For a node representing state $x_{k+j|k}$, one outgoing edge each for any $v_{k+j|k} \in V$ is introduced, leading to a successor node on the next layer. The following sections describe methods to reduce the share of the tree to be explored by pruning over cost bounds and by using dynamic programming.

4.2.2 Lower and Upper Cost Bounds for Pruning a Search Tree

Lower Bound Identification A lower cost bound for a given state can be obtained by ignoring the constraints on the continuous states and the continuous inputs of (71). With

this assumption, the following variant of the optimization problem (71) is obtained:

$$\begin{aligned}&\min_{\phi_u,\phi_v} \Omega^{un}(H,\phi_u,\phi_v,x_k,x_f)\\&\text{s. t. } (69), v_{(k+j|k)} \in V, j \in \{0,\cdots,H-1\}.\end{aligned} \tag{73}$$

As Ω^{un} has the same form as Ω in (71), (73) can be seen as an extension of (71) to a larger feasible set, providing a lower bound:

$$\Omega^{un,*}(x_k) \leq \Omega^*(x_k). \tag{74}$$

The difference between $\Omega^{un,*}(x_k)$ and $\Omega^*(x_k)$ depends on the effects of the constraints. The focus of the rest of this subsection is on computing $\Omega^{un,*}(x_k)$ efficiently.

As proposed in [73,74], *difference riccati equation* are used to handle problem (73). Specifically, the *H-step value function* $\mathcal{V}(x_{(k+j|k)}) = (x_{(k+j|k)} - x_f)^{\mathrm{T}}\mathcal{P}_j^*(x_{(k+j|k)} - x_f)$ for $\forall j \in \{0,\cdots,H\}$ is defined first to formulate the cost-to-go from step j to step H in (73) . A set of positive-definite symmetric matrices $\mathcal{P}_j^*$ for $j \in \{0,\cdots,H\}$ are introduced, satisfying the property:

$$\begin{aligned}&\mathcal{V}^*(x_{(k+j|k)}) = (x^{un,*}_{(k+j|k)} - x_f)^{\mathrm{T}}\mathcal{P}_j^*(x^{un,*}_{(k+j|k)} - x_f)\\&= \min_{\phi_u^j,\phi_v^j}\{\sum_{i=j}^{H-1}\mathcal{L}(x_{(k+i|k)},u_{(k+i|k)})\\&+(x_{(k+H|k)} - x_f)^{\mathrm{T}}Q_H(x_{(k+H|k)} - x_f)\}\\&\text{s. t. } (69),\ v_{(k+i|k)} \in V,\ i \in \{j,\cdots,H-1\}.\end{aligned} \tag{75}$$

Here, $\phi_u^j = (u_{(k+j|k)},\cdots,u_{(k+H-1|k)})$, $\phi_v^j = (v_{(k+j|k)},\cdots,v_{(k+H-1|k)})$, and $x^{un,*}_{(k+j|k)} \in \phi_x^{un,*}$ denotes the optimal continuous state at step $k+j$ in (73). Similar to the above, there also exists an *H-step value function* $\mathcal{V}^{*,c}(x_{(k+j|k)})$ in problem (71) to denote the cost-to-go from j to H, where for $j \in \{0,\cdots,H\}$:

$$\begin{aligned}&\mathcal{V}^{*,c}(x_{(k+j|k)}) = \min_{\phi_u^j,\phi_v^j}\{\sum_{i=j}^{H-1}\mathcal{L}(x_{(k+i|k)},u_{(k+i|k)}) + (x_{(k+H|k)} - x_f)^{\mathrm{T}}Q_H(x_{(k+H|k)} - x_f)\}\\&\text{s. t. } (69),\ v_{(k+i|k)} \in V; u_{(k+i|k)} \in U,\ i \in \{j,\cdots,H-1\}\\&x_{(k+i|k)} \in X_{(k+i|k)},\ x_{(k+H|k)} \in X_f^{k+H|k}.\end{aligned} \tag{76}$$

Since (75) always provides a larger feasible set than (76) for the same state $x_{(k+j|k)}$, the relation $\mathcal{V}^{*,c}(x_{(k+j|k)}) \geq \mathcal{V}^*(x_{(k+j|k)})$ always holds. Moreover, as proved in [74], the following relation between $\mathcal{P}_H^*$ and Q_H exists:

- by substituting $j = H$ into (75):

$$\begin{aligned}
&\mathcal{V}^*(x_{(k+H|k)}) = (x^{un,*}_{(k+H|k)} - x_f)^{\mathrm{T}} \mathcal{P}^*_H (x^{un,*}_{(k+H|k)} - x_f) \\
&= \min_{x_{(k+H|k)} \in \phi^{un}_x} (x_{(k+H|k)} - x_f)^{\mathrm{T}} Q_H (x_{(k+H|k)} - x_f)\} \\
&= (x^{un,*}_{(k+H|k)} - x_f)^{\mathrm{T}} Q_H (x^{un,*}_{(k+H|k)} - x_f)\} \\
&\text{and thus:} \\
&\mathcal{P}^*_H = Q_H.
\end{aligned} \tag{77}$$

- by substituting $j = 0$ into (75):

$$\begin{aligned}
&\mathcal{V}^*(x_k) = (x^{un,*}_k - x_f)^{\mathrm{T}} \mathcal{P}^*_0 (x^{un,*}_k - x_f) \\
&= \min_{\phi_u, \phi_v} \{ \sum_{i=0}^{H-1} \mathcal{L}(x_{(k+i|k)}, u_{(k+i|k)}) \\
&+ (x_{(k+H|k)} - x_f)^{\mathrm{T}} Q_H (x_{(k+H|k)} - x_f)\} \\
&= \Omega^{un,*}(x_k).
\end{aligned} \tag{78}$$

Since x_k is known from measurements, (78) indicates that $\Omega^{un,*}(x_k)$ of (73) can be determined by determining $\mathcal{P}^*_0$. Following [74], the computation of $\mathcal{P}^*_0$ can traced by iterative enumeration of the *difference Riccati equation*: starting from H, for each $v_{(k+H-1|k)} = v_q \in V$ the value of $\mathcal{P}_{H-1}$ follows from:

$$\begin{aligned}
&\mathcal{P}_{H-1} = Q_1 + A^{\mathrm{T}}_{v_q} \mathcal{P}_H A_{v_q} + 2\mathcal{K}^{\mathrm{T}}_q B^{\mathrm{T}}_{v_q} \mathcal{P}_H A_{v_q} \\
&+ \mathcal{K}^{\mathrm{T}}_{H-1} (B^{\mathrm{T}}_{v_q} \mathcal{P}_H B_{v_q} + Q_2) \mathcal{K}_{H-1}
\end{aligned} \tag{79}$$

$$\mathcal{K}_{H-1} = -(B^{\mathrm{T}}_{v_q} \mathcal{P}_H B_{v_q} + Q_2)^{-1} B^{\mathrm{T}}_{v_q} \mathcal{P}_H A_{v_q}.$$

Since Q_2 and $\mathcal{P}_H$ are defined positive-definite and only real-valued entries exist in B_{v_q}, it applies that $B^{\mathrm{T}}_{v_q} \mathcal{P}_H B_{v_q} + Q_2 > 0$, and the matrix is invertible. As $\mathcal{P}^*_H = Q_H$ is given in (77), we can substitute each $v_{(k+H-1|k)} = v_q \in V$ into (79), and thus a number of n_v different matrices $\mathcal{P}_{H-1}$ are obtained in step $k + H - 1$. We use the symbol $\mathbb{P}_{H-1}$ to denote the set of these matrices. By continuing the backward computation, a set $\mathbb{P}_0$ is finally obtained for step k, containing the ${n_v}^H$ matrices $\mathcal{P}_0$, as shown in Fig. 23. Since the optimal value $\mathcal{P}^*_0$ is contained in $\mathbb{P}_0$, it can be identified by:

$$\mathcal{P}^*_0 = \underset{\mathcal{P}_{0,q} \in \mathbb{P}_0}{\operatorname{argmin}} (x_k - x_f)^{\mathrm{T}} \mathcal{P}_{0,q} (x_k - x_f), \ , \tag{80}$$

and thus $\Omega^{un,*}(x_k)$ is obtained, and provides a lower bound of $\Omega^*(x_k)$. However, the explicit enumeration induces exponential complexity, and thus rapidly becomes intractable. As

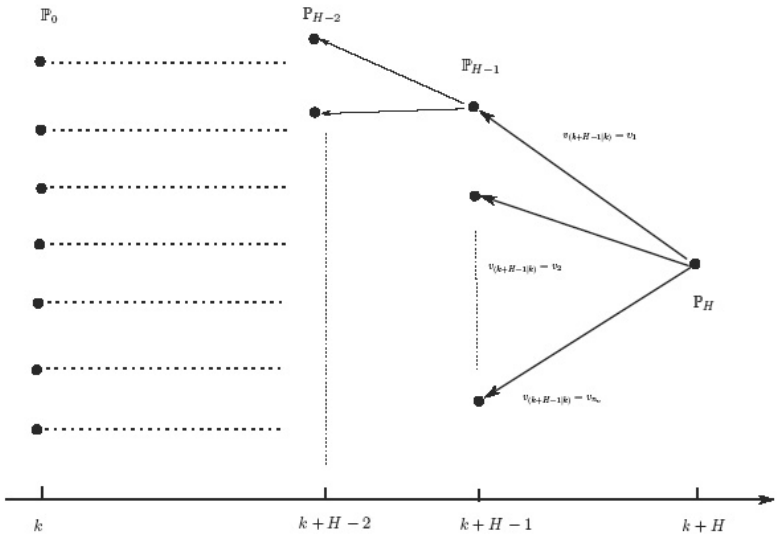

Figure 23: An explicit backward enumeration procedure

countermeasure, [73, 74] proposed to multiply the step cost $\mathcal{L}$ with a relaxation factor, leading to an efficient reduction of the set $\mathbb{P}_j$. However, selecting the relaxation factor is diffcult, and no constructive rule seems to exist. The pruning of the set seems to extremely sensitive to the factor, as indicated in [74], where increasing the factor from 1.0 to 1.0001, reduces $|\mathbb{P}_0|$ from 386 to 22. While problem (73) is very similar to the one in [73, 74], the objective is not exactly the same: while the reference aims at an explicit and efficient approximation of $\Omega^{un,*}(x_k)$, we go for approximating the minimum of $\Omega^*(x_k)$ instead of $\Omega^{un,*}(x_k)$, where $\Omega^{un,*}(x_k)$ only provides a lower bound of the original variant. Thus, computing a lower bound of $\Omega^{un,*}(x_k)$ is just an intermediate step, i.e. approximating $\Omega^{un,*}(x_k)$ should not incur significant effort.

Recalling (80) and Fig. 23, the reason for using the explicit enumeration up to the first step is the missing information of $\phi_x^{un,*}$: only after $\mathcal{P}_0^*$ is identified, one is able to forwardly apply (79) to calculate $\phi_x^{un,*}$. As a work-around, [81, 82] assumes that for all $j \in \{1, \cdots, H\}$, the deviation from the final state $(x_{(k+j|k)}^{un,*} - x_f)$ follows a Gaussian distribution with expected value $\mathbf{E}((x_{(k+j|k)}^{un,*} - x_f)) = 0$ and covariance matrix $\mathbf{E}((x_{(k+j|k)}^{un,*} - x_f)^{\mathrm{T}}(x_{(k+j|k)}^{un,*} - x_f)) = \mathbb{I}$. Then, the trace of $\mathcal{P}_j$ can be used to represent the expected value of $(x_{(k+j|k)}^{un,*} - x_f)^{\mathrm{T}}\mathcal{P}_j(x_{(k+j|k)}^{un,*} - x_f)$. By employing this assumption and the fact that x_f is given, the following algorithm 8 is proposed, which is executed at step $k + j$:

This algorithm provides a 'Best-First' like method in which the trace of each $\mathcal{P}_j \in \mathbb{P}_j$ is compared only to the the matrix with the minimal trace found so far. Thus, $\mathbb{P}_j$ constantly contains one element over all steps. Finally, a sequence $\phi_{\mathcal{P}}^{apx} = (\mathcal{P}_0^{apx}, \mathcal{P}_1^{apx}, \cdots, \mathcal{P}_H^{apx})$ is obtained and used as approximation of $\phi_{\mathcal{P}}^* = (\mathcal{P}_0^*, \mathcal{P}_1^*, \cdots, \mathcal{P}_H^*)$. Meanwhile, as each $\mathcal{P}_j^{apx}$ relates to a certain v_q, a discrete input sequence $\phi_v^{apx} = (v_0^{apx}, v_1^{apx}, \cdots, v_{H-1}^{apx})$ is also obtained.

Algorithm 8 Pruning Procedure in step $k+j$

```
Given: ℙ_j;
for m = 1 : |ℙ_j| do
  for n = 1 : |ℙ_j| do
    if m ≠ n then
      if trace(𝒫_{j,m}) < trace(𝒫_{j,n}) then
        𝒫_{j,n} is pruned from the tree
      end if
    end if
  end for
end for
```

In comparison with full enumeration, the search reduces from $\sum_{j=1}^{H} {n_v}^j$ to $H \cdot n_v$. Moreover, the tuning procedure for a relaxation factor is not necessary, and numeric studies for examples show that the obtained lower bounds to $\Omega^*(x_k)$ are suitable for pruning the search tree. The obtained lower bound $\Omega^{apx}(x_k)$ of $\Omega^*(x_k)$ takes the following form:

$$\begin{aligned} \Omega^{apx}(x_k) &= (x_k - x_f)^{\mathrm{T}} \mathcal{P}_0^{apx} (x_k - x_f) \\ &\approx (x_k - x_f)^{\mathrm{T}} \mathcal{P}_0^{*} (x_k - x_f) \leq \Omega^*(x_k). \end{aligned} \tag{81}$$

Furthermore, based on (75) and (76), the relation (81) can be extended to a more general case ($\forall j \in \{0, \cdots, H\}$):

$$\begin{aligned} &\underline{\mathcal{V}^*}(x_{(k+j|k)}) = (x_{(k+j|k)} - x_f)^{\mathrm{T}} \mathcal{P}_j^{apx} (x_{(k+j|k)} - x_f) \\ &\approx (x_{(k+j|k)} - x_f)^{\mathrm{T}} \mathcal{P}_j^{*} (x_{(k+j|k)} - x_f) \leq \mathcal{V}^{*,c}(x_{(k+j|k)}). \end{aligned} \tag{82}$$

Here, $\underline{\mathcal{V}^*}(x_{(k+j|k)})$ denotes the obtained lower bound. Relation (82) suggests that, once $x_{(k+j|k)}$ and $\mathcal{P}_j^{apx}$ are known, the lower bound of the cost-to-go starts from state $x_{(k+j|k)}$ and can be immediately computed.

Upper Bound Identification Referencing again problem (71), the state sequence $\phi_x = F_c(\phi_v, \phi_u, x_k)$ obtained for ϕ_v^*, ϕ_u^* should obviously satisfy the relation $\Omega^*(H, \phi_u^*, \phi_v^*, x_k, x_f) \leq \Omega(H, \phi_u, \phi_v, x_k, x_f)$. By using the obtained sequence ϕ_v^{apx} in problem (71), the evolution of state $x_{(k+j|k)}$ in (69) only depends on ϕ_u. Thus, in the original MINLP problem (71) the binary variables will be fixed and a simple QP results, which can be solved efficiently. Assume that $\phi_u^{apx,*}$ and $\phi_x^{apx,*}$ determine the solution for ϕ_v^{apx} to be obtained from:

$$\phi_u^{apx,*} = \underset{\phi_u}{\operatorname{argmin}}\, \Omega(H, \phi_u, \phi_v^{apx}, x_k, x_f) \tag{83}$$

$$\text{s. t.: (69)},\ v_{(k+j|k)} := v_j^{apx},\ u_{(k+j|k)} \in U, x_{(k+j|k)} \in X_{(k+j|k)},\ j \in \{0, \cdots, H-1\}$$

$$x_{(k+H|k)} \in X_f^{k+H|k}$$

Then, the following relation holds:

$$\Omega^*(H, \phi_u^*, \phi_v^*, x_k, x_f) \leq \Omega^*(H, \phi_u^{apx,*}, \phi_v^{apx}, x_k, x_f). \tag{84}$$

If no feasible solution exists to (83), the value of $\Omega^*(H, \phi_u^{apx,*}, \phi_v^{apx}, x_k, x_f)$ is infinite. Similarly as in (81), the above criteria can also be used for the upper bound $\mathcal{V}^{*,c}(x_{(k+j|k)}) \leq \overline{\mathcal{V}^*}(x_{(k+j|k)})$ for $\forall j \in \{0, \cdots, H\}$:

$$\overline{\mathcal{V}^*}(x_{(k+j|k)}) := \min_{\phi_u^j, \phi_v^j := \phi_v^{apx,j}} \{\sum_{i=j}^{H-1} \mathcal{L}(x_{(k+i|k)}, u_{(k+i|k)}) + (x_{(k+H|k)} - x_f)^{\mathrm{T}} Q_H (x_{(k+H|k)} - x_f)\} \tag{85}$$

$$\text{s. t.: (69)},\ u_{(k+i|k)} \in U;\ x_{(k+i|k)} \in X_{(k+i|k)},\ x_{(k+H|k)} \in X_f^{k+H|k},\ i \in \{j, \cdots, H-1\} \tag{86}$$

$$\phi_v^{apx,j} = (v_j^{apx}, \cdots, v_{H-1}^{apx}) \subseteq \phi_v^{apx}. \tag{87}$$

Now, the value of $\mathcal{V}^{*,c}(x_{(k+j|k)}$ for $j \in \{0, \cdots, H\}$ is bound to the range:

$$\underline{\mathcal{V}^*}(x_{(k+j|k)}) \leq \mathcal{V}^{*,c}(x_{(k+j|k)} \leq \overline{\mathcal{V}^*}(x_{(k+j|k)}). \tag{88}$$

The determination of the bounds of $\mathcal{V}^{*,c}(x_{(k+j|k)})$ according to the procedures described before can be accomplished with the following complexity:

- for the lower bound, the computation of $\phi_{\mathcal{P}}^{apx}$ only involves $H \cdot n_v$ matrix trace computations; In addition, as the computation of $\phi_{\mathcal{P}}^{apx}$ in (81) does not involve the initial state x_k, the same $\phi_{\mathcal{P}}^{apx}$ is applicable for any initial state in the MPC procedure, as long as the other parameters like the prediction horizon, and the weighting matrices do not change;
- for the upper bound, the optimization problem (85) has to be solved; such QP problems can be solved in negligible time compared to the other steps.

Graph Search Procedure In order to determine an approximation of problem (71) by using the bounds, a graph search procedure is used. The root node represents x_k, and one edge originates from a node for each $v_q \in V$ and leads to a new state on the next layer. The condition for pruning a branch is established by comparing the costs for the new states to corresponding cost bounds. To this end, consider the following fact resulting from (88):

Lemma 4.1 *Starting from a state $x_{(k+j|k)}$, let the pair of states x_m, x_n ($x_m \neq x_n$) by reachable at step $k+j+1$. If then $\underline{\mathcal{V}^*}(x_m) > \overline{\mathcal{V}^*}(x_n)$ applies, the relation $\mathcal{V}^{*,c}(x_m) > \mathcal{V}^{*,c}(x_n)$ holds.*

The implication of this fact is that x_m can not be part of the optimal input sequence, and the node representing x_m need not to be explored further.

As a further means to reduce the search graph, we employ the concept of adjacency of states, as introduced in [80]. The two states x_m and x_n (both again reachable in step $k+j+1$), are said to be adjacent if:

$$\|x_m - x_n\| \leq \gamma,\ \gamma > 0, \tag{89}$$

holds for an appropriate small choice of γ. Only that state of an adjacent pair is further explored, for which the corresponding cost-to-go ($\mathcal{V}^{*,c}(x_m)$, or $\mathcal{V}^{*,c}(x_n)$) is smaller.

Now, in order to evaluate the cost of a node $x_{(k+j|k)}$ of the graph, we can turn to the problem:

$$\begin{aligned}
&\mathcal{V}^{*,c}(x_{(k+j|k)}) := \\
&\min_{u_{(k+j|k)}, v_{(k+j|k)}} \{\underbrace{\mathcal{L}(x_{k+j|k}, u_{(k+j|k)})}_{Step\,cost} + \underbrace{\mathcal{V}^{*,c}(x_{(k+j+1|k))})}_{Cost\,to\,go}\} \\
&\text{s. t.: (69)},\ x_{(k+j+1|k)} \in X_{(k+j+1|k)},\ u_{(k+j|k)} \in U,\ v_{(k+j|k)} \in V.
\end{aligned} \tag{90}$$

Equation (90) is based on the principle of *Dynamic Programming* and if $j = 0$, the relation $\mathcal{V}^{*,c}(x_k) = \Omega^*(x_k)$ exists. If the mixed input $u_{(k+j|k)} \in U$, $v_{(k+j|k)} \in V$ are leading to the states x_m and x_n, where $x_m, x_n \in X_{(k+j+1|k)}$, then the step cost for both cases can be identified, and the suboptimal costs $\mathcal{V}^c(x_{(k+j|k)})$ (as obtained when reaching x_m and x_n at step $k+j+1$) are:

$$\mathcal{V}^{*,c,x_{j+1}=x_m}_{(x_j)} = \underbrace{\mathcal{L}(x_{(k+j|k)}, u^m_{(k+j|k)})}_{Known} + \underbrace{\mathcal{V}^{*,c}(x_m)}_{Unknown}, \tag{91}$$

and

$$\mathcal{V}^{*,c,x_{j+1}=x_n}_{(x_j)} = \underbrace{\mathcal{L}(x_{(k+j|k)}, u^n_{(k+j|k)})}_{Known} + \underbrace{\mathcal{V}^{*,c}(x_n)}_{Unknown}. \tag{92}$$

Here, $\mathcal{V}^{*,c,x_{j+1}=x_m}_{(x_j)}$ denotes the optimal cost of $\mathcal{V}^c(x_{(k+j|k)})$ by reaching x_m at step $k+j+1$. For the unknown parts $\mathcal{V}^{*,c}(x_m)$ and $\mathcal{V}^{*,c}(x_n)$, a range is determined by (88):

$$\underbrace{\mathcal{L} + \underline{\mathcal{V}^*}(x_m)}_{Known} \leq \mathcal{V}^{*,x_{j+1}=x_m}_{(x_j)} \leq \underbrace{\mathcal{L} + \overline{\mathcal{V}^*}(x_m)}_{Known}, \tag{93}$$

and

$$\underbrace{\mathcal{L} + \underline{\mathcal{V}^*}(x_n)}_{Known} \leq \mathcal{V}^{*,x_{j+1}=x_n}_{(x_j)} \leq \underbrace{\mathcal{L} + \overline{\mathcal{V}^*}(x_n)}_{Known}. \tag{94}$$

In these equations, $\mathcal{L}$ is a short notation for the step cost. Now, if the lower cost bound for reaching x_n is higher than the upper cost bound for reaching x_m, choosing x_m ensures a better performance than x_n. In the cases in which this relation does not apply, the adjacency condition (89) is checked. If satisfied, only one node (the one with better performance) is further considered.

The graph search procedure starts from node x_k, and in order to select $v_{(k|k)} = v_q \in V$ (and thus to determine the successor $x_{(k+1|k)}$), the following subproblem is solved for any $v_q \in V$:

$$u^*_{(k|k)} := \underset{u_{(k|k)} \in U}{\operatorname{argmin}} \{ \underbrace{\mathcal{L}(x_k, u_{(k|k)}) + \underline{\mathcal{V}^*}(x_{(k+1|k)})}_{Lower\ Bound} + \underbrace{\mathcal{L}(x_k, u_{(k|k)}) + \overline{\mathcal{V}^*}(x_{(k+1|k)})}_{Upper\ Bound} \} \quad (95)$$

$$\text{s.t.: } (69), (82), (85). \quad (96)$$

Then, the new node for each $v_{(k|k)} = v_q \in V$ is obtained from:

$$x^{*,v_q}_{(k+1|k)} = A_{v_q} x_k + B_{v_q} u^*_{(k|k)}. \quad (97)$$

With respect to the optimality condition (90) with $j = 0$, the optimal cost value $\mathcal{V}^{*,c}(x_k)$ should satisfy:

$$\begin{aligned} \mathcal{V}^{*,c}(x_k) = & \min_{u_{(k|k)}, v_{(k|k)}} \{ \underbrace{\mathcal{L}(x_k, u_{(k|k)})}_{Step\ cost} + \underbrace{\mathcal{V}^{*,c}(x_{(k+1|k))})}_{Cost\ to\ go} \} \\ & \text{s.t.: } (69),\ x_{(k+1|k)} \in X_{(k+1|k)},\ u_{(k|k)} \in U,\ v_{(k|k)} \in V. \end{aligned} \quad (98)$$

As neither the lower nor upper bound represents the truly optimal value $\mathcal{V}^{*,c}(x_{(k+1|k)})$, problem (95) minimizes the upper/lower-bound of $\mathcal{V}^{*,c}(x_{(k+1|k)})$ simultaneously, and is an approximation to (98). Notice that solving (95) is relative easy, since it represents a QP problem only. After executing the above procedure, at most n_v successor states $x^{*,v_q}_{(k+1|k)}$ are obtained – let $\mathcal{X}_{(k+1|k)}$ denot this set. Overall, Alg. 9 determines the candidate state sequence $\phi_{\mathcal{X}} = (\mathcal{X}_{(k+0|k)}, \cdots, \mathcal{X}_{(k+H|k)})$ of all H steps.

Let $\phi_V = (V_{(k|k)}, \cdots, V_{(k+H-1|k)})$ denote the set of discrete input sequences generating $\phi_{\mathcal{X}}$. The optimal discrete input sequence $\phi^{*,s}_v$ from this set is obtained as the one leading to $\Omega^*(H, \phi^{*,s}_u, \phi^s_v, x_k, x_f)$, and which approximates the solution to the original problem (71) best.

Algorithm 9 Determination of $\phi_{\mathcal{X}}$

Given: $\mathcal{X}_{(k+0|k)} = \{x_k\}$, U, V;
for $j = 0 : H-1$ **do**
  for $m = 1 : |\mathcal{X}_{(k+j|k)}|$ **do**
    for $q = 1 : n_v$ **do**
      Compute $u^*_{(k+j|k)}$, $x^{*,v_q}_{(k+j+1|k)}$ under given $v_q \in V$ and $x^m_{(k+j|k)} \in \mathcal{X}_{(k+j|k)}$ by solving (95) and record $\underline{\mathcal{V}^*}(x^{*,v_q}_{(k+j+1|k)})$ and $\overline{\mathcal{V}^*}(x^{*,v_q}_{(k+j+1|k)})$
    end for
  end for
  eliminate $x^{*,v_q}_{(k+j+1|k)} \in \mathcal{X}_{(k+j+1|k)}$ if its lower cost bound is higher than the upper bound of any other state in $\mathcal{X}_{(k+j+1|k)}$
  check pairwise adjacency of the remaining states and only keep the one with lowest costs
  update $\mathcal{X}_{(k+j+1|k)}$ to the remaining states
end for

4.2.3 Numerical Example

Numerical Comparison The first numerical evaluation evaluates the proposed technique for 30 randomly generated switched systems. The results are compared exemplarily to the solvers *bnb.solver*&*fmincon*, which exist in Matlab and are able to solve the underlying MINLP problems. The comparison comprises systems with $n_x = 10$, $n_u = 8$, $n_v = 4$ and $H = 8$. For an adjacency parameter $\gamma = 5$, Fig. 24 shows the relative deviation of the optimization results for the two approaches – while the proposed method is better in this respect in most cases, it is stressed that the average computation time is $0.0651s$ with the new method, compared to $8.0467s$ for *bnb.solver*&*fmincon*, i.e. a drastic reduction in computation time is obtained.

The comparison to globally optimal solution (obtained from full enumeration) is shown in Fig. 25. The figures 26 and 27 show the average computation as well as the average deviation from the global optimum for varying values of H.

Vehicle Platoon The second part of the numerical evaluation refers to the control of a vehicle platoon - compared to the version considered in Sec. 3 it is extended to mixed inputs: each of the $M = 4$ vehicles is here modeled by a different switched system with a discrete input (modeling the gear), which changes the mode of continuous dynamics (acceleration). The local constraints of each vehicle models different ranges for velocity and acceleration. Furthermore, a coupling constraint between neighbored vehicles is used to model that the distance must not decrease below $d = 25$ to avoid collision. In addition, the local cost functions of each vehicle are differently parametrized by the weighting matrices.

A number of 36 subsystem combinations over all vehicles would be possible in a single

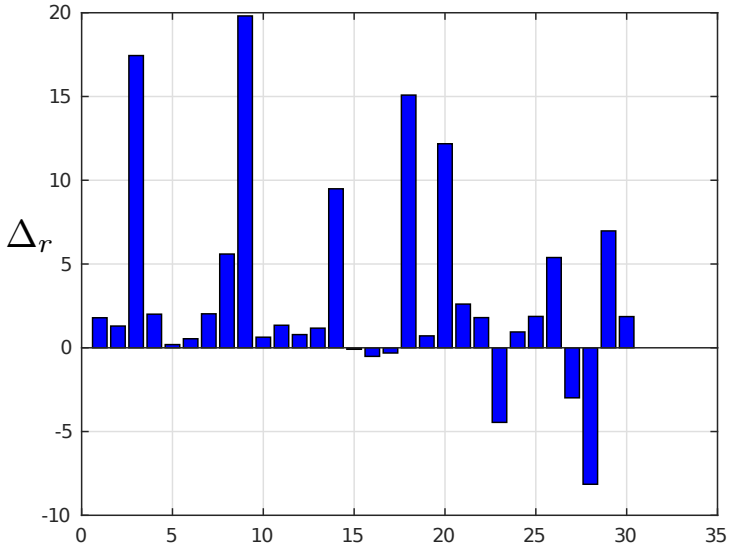

Figure 24: Relative deviation $\Delta_r = \frac{\Omega^{bf} - \Omega^{*,sub}}{\Omega^{*,sub}}\%$ between the optimization result $\Omega^{*,sub}$ for the proposed method and the Ω^{bf} obtained by $bnb.solver\&fmincon$; the abscissa refers to the experiment index.

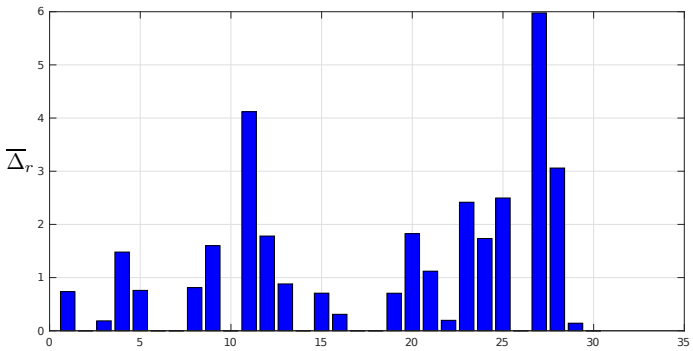

Figure 25: Relative deviation $\bar{\Delta}_r = \frac{\Omega^{*,sub} - \Omega^{*}}{\Omega^{*}}\%$ of the costs $\Omega^{*,sub}$ obtained with the proposed method compared to the globally optimal solution Ω^{*}; the abscissa is again the experiment index. The average deviation for $\gamma = 5$ is 1.10%, while for $\gamma = 2$ an average deviation of 1.16% and an average computation time of $0.08020s$ is obtained.

decision step of a centralized controller. The prediction horizon H is selected to be 15, which would lead to 540 integer variables if mixed-integer programming were used, leading to $36^{15} \approx 2.2107 \times 10^{23}$ possible discrete input sequences in any k. For a chosen parametrization and initialization, the proposed method provides a solution in which the vehicles reach their target after 17 time steps, and the compuation takes in average $1.22s$ per time step k. The following figures show the courses of position, gear, velocity, and acceleration of each vehicle (s_4 in black, s_3 in magenta, s_2 in blue, s_1 in red). It can be seen that the vehicles reached their target while satisfying all constraints and the computation time in each step is acceptable for

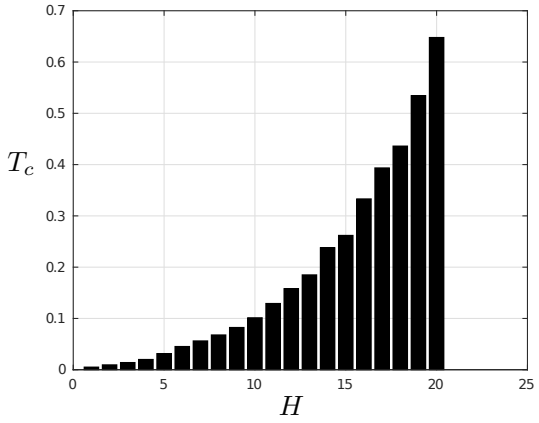

Figure 26: Average computation time T_c over the prediction horizon H.

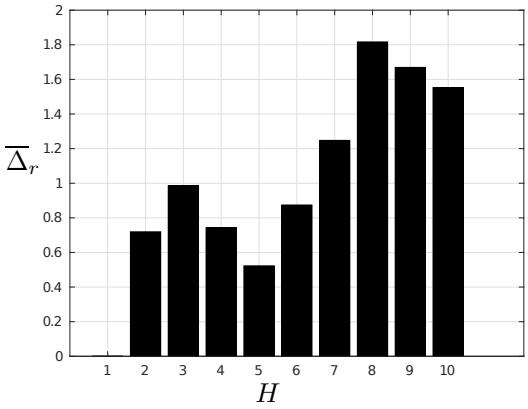

Figure 27: Average deviation $\bar{\Delta}_r$ over the prediction horizon H.

real-time computation.

4.3 Online Adaption of Motion Paths to Time-Varying Constraints Using Homotopic Functions

While the method described before is applicable also to CPS with time-varying state-constraints in principle, this subsection focusses on handling such constraints by an alternative procedure. Again, the underlying understanding is that the state constaints result either from the communicated planning of another subsystem, or from the monitoring of an obstacle (or non-cooperative subsystem). To make the notation simpler, we here refer to subsystem dynamics which is purely continuous. More specifically, this part addresses the problem of controlling a linear discrete time system from an initial to a final state, while minimizing a cost function and considering input constraints as well as non-convex time-varying state constraints. A practical instance of such a problem is human-robot-cooperation, in which the robot has to avoid collisions with a human worker while maintaining the nominal operation close to optimal. Another example from the field of automated driving is to prevent accidents with pedestrians or other cars. A difficulty of solving such problems is the large online computational effort, rendering standard formulations of optimal or predictive control often not applicable. The goal of this work is to derive a method which provides optimized and efficient solutions for

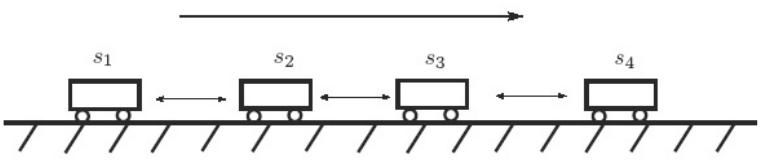

Figure 28: Platoon of 4 vehicles with switched dynamics driving in one lane.

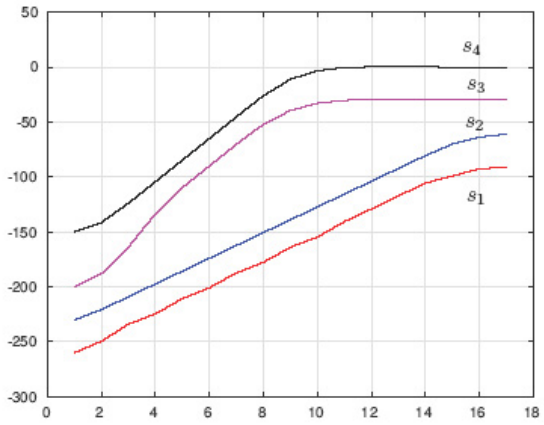

Figure 29: Position of each vehicle over time.

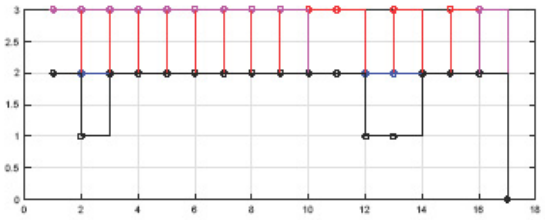

Figure 30: Gear of each vehicle over time.

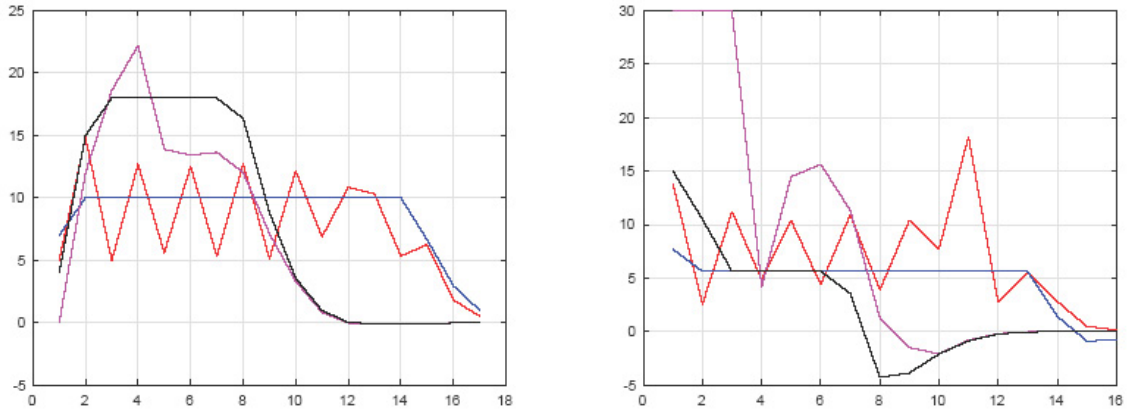

Figure 31: Velocity (left plot) and acceleration (right plot) of each vehicle over time.

such scenarios.

Different MPC methods were proposed in the literature to optimized point-to-point transition with constraints while reducing the online effort, like move-blocking strategies [83], soft-constrained formulations by penalty terms or barrier functions [84], [85], and direct multiple shooting [86]. While for convex problems, the computation times typically are sufficiently low for online operation, the situation for non-convex problems is different. MPC with mixed-integer-programming (MIP) as discussed in [87] or [88] for a human-robot scenario, suffers from large amounts of binary variables for encoding convex partitioned regions over which a suitable sequence can be determined by tree search. Even for small problems, the combinatorial complexity limits the real-time applicability. Therefore, also methods like *explicit MPC* [89], were developed.

In the field of human-robot-interaction *potential field methods* [90], cell-decomposition based on graph search [91], and sampling-based methods like *rapidly exploring random trees* (RRT), and RRT* [92] were developed. Most of them do not consider the system dynamics, are

not applicable for higher dimensions, or do not treat prediction informations of the obstacle/ other subsystems.

Here, we describe an approach that adopts homotopic trajectories to determine optimized point-to-point trajectories in the presence of input constraints and state constraints given by time-varying moving obstacles. The proposed method identifies time steps at which collisions with a moving obstacle may occur. These are used in a collision avoidance scheme based on tree-search. The homotopic trajectories are computed offline, which avoids time-consuming online computation of complete trajectories and makes the online implementation efficient. The main advantage is that the homotopy control algorithm performs its computations over an entire trajectory (which is coded by the homotopy parameter) and not as usually done in trajectory optimization for each time step. This reduces the optimization problem to a low-dimensional problem. Furthermore, due to the homotopic space spanned by the base trajectories, a subset of time steps is identified which contains time steps which are relevant for collision avoidance. By considering only these in the optimization, the efficiency of the proposed method increases significantly. The simulations show that the proposed method computes close to optimal trajectories with significantly lower computational effort than standard MPC. This work has been extend to time-varying final states and to nonlinear systems, as given for a nonlinear robotic manipulator with the abstraction techniques described in Deliverable 1.2.

4.3.1 Homotopic Functions

Let a linear discrete-time system be given by:

$$x_{k+1} = Ax_k + Bu_k, \tag{99}$$

with time $k \in \mathbb{N}_0$, state vector $x_k \in \mathbb{R}^{n_x}$, and inputs from the feasible set $u_k \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$. (Compared to (68), here we do not consider switched dynamics for simplicity.)

Definition 4.1 *Given the system dynamics* (99)*, a system is said to be r-step controllable, if there exists a control input $u_k \in \mathbb{R}^{n_u}$ such that for any initial state x_0, the state can be driven to a final state x_f in at least r time-steps.*

This is fulfilled if the controllability matrix $R = (B, AB, A^2B, ..., A^{n_x-1}B) \in \mathbb{R}^{n_x \times n_u n_x}$ has full row rank i.e. $\texttt{rank}(R) = n_x$. Constraints on inputs or states are irrelevant for this property.

Let so-called *base trajectories* of (99) be given by $\hat{x}^i = (x_0^i, ..., x_N^i)$ and $\hat{u}^i = (u_0^i, ..., u_{N-1}^i)$, where $i \in \mathcal{M} := \{0, 1, ..., n_c\}$ is indexing the i-th base trajectory of $n_c + 1$ many. They are chosen to span a region, in which the online circumvention of an obstacle is possible. All base trajectories are defined in the same time domain, i.e. the same number of time steps with final time $N \in \mathbb{N}$, and the same initial and final states $x_0^i = x_s$, $x_N^i = x_f$, $i \in \mathcal{M}$. They are designed in such a way, that sufficient reserves in the input space are available, to later transition from one to another. Let the set $\mathcal{X} := \{\hat{x}^0, ..., \hat{x}^{n_c}\}$ contain all base trajectories, which are computed offline and are thus known in the online procedure. When choosing the base trajectories, knowledge about possible position occurrence of the obstacles should be considered.

A trajectory $\hat{x}^i$ can be interpreted as the image of a function: $\hat{x}^i = F^i(\hat{u}^i)$. The following definition introduces *homotopic* trajectories which are in between the trajectories $\hat{x}^i$:

Definition 4.2 *For a set of $n_c + 1$ continuous functions $F^i : \mathbb{R}^{n_u \times T} \to \mathbb{R}^{n_x \times T}$, $i \in \mathcal{M}$, a vectorized homotopy is defined by: $H : \left(\mathbb{R}^{n_u \times T}\right)^{n_c} \times [0,1]^{n_c} \to \mathbb{R}^{n_x \times T}$. The second argument is a vector of homotopy parameters $\boldsymbol{\lambda} = (\lambda^1, ..., \lambda^{n_c})^T$ with:*

$$\lambda^i \in [0,1], \ \sum_{i=1}^{n_c} \lambda^i \leq 1. \tag{100}$$

With $F = (F^1(\hat{u}^1) - F^0(\hat{u}^0), ..., F^{n_c}(\hat{u}^{n_c}) - F^0(\hat{u}^0))^T$ and $\lambda^0 := 1 - \sum_{i=1}^{n_c} \lambda^i$, the linear vectorized homotopy function is given by:

$$\begin{aligned} H(\hat{u}^0, \ldots, \hat{u}_c^n, \boldsymbol{\lambda}) &= \sum_{i=0}^{n_c} \lambda^i \cdot F^i(\hat{u}^i) \\ &= F^0(\hat{u}^0) + \sum_{i=1}^{n_c} (F^i(\hat{u}^i) - F^0(\hat{u}^0)) \cdot \lambda^i = F^0(\hat{u}^0) + F \cdot \boldsymbol{\lambda} \end{aligned} \tag{101}$$

This definition states that the homotopic functions are linear interpolations of the base trajectories, see Fig. 32 for illustration. Since the dynamics of (99) is also linear, the homotopic trajectories share the same characteristics and satisfy the same convex input or state constraints as the base trajectories. Since (101) relates to complete trajectories, a homotopic state $x_k(\boldsymbol{\lambda}_k)$ for a single point of time k lies in between the states x_k^i, $i \in \mathcal{M}$, and is identified by the homotopy value $\boldsymbol{\lambda}_k$. This leads to the following formulation of a homotopic state and input at time k:

$$x_k(\boldsymbol{\lambda}_k) := x_k^0 + D_{x_k} \boldsymbol{\lambda}_k, \ u_k(\boldsymbol{\lambda}_k) := u_k^0 + D_{u_k} \boldsymbol{\lambda}_k, \tag{102}$$

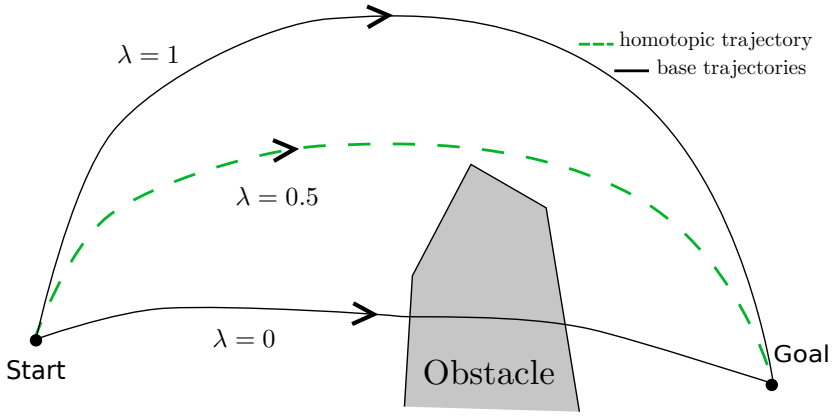

Figure 32: Illustration of two base trajectories (black), and a homotopic trajectory (green), given by $\boldsymbol{\lambda} = 0.5$, which circumvents the obstacle

with the homotopy vector $\boldsymbol{\lambda}_k := (\lambda_k^1, ..., \lambda_k^{n_c})^T \in \mathbb{R}^{n_c}$ at time k, matrices $D_{x_k} = (x_k^1 - x_k^0, ..., x_k^{n_c} - x_k^0) \in \mathbb{R}^{n_x \times n_c}$, and $D_{u_k} = (u_k^1 - u_k^0, ..., u_k^{n_c} - u_k^0) \in \mathbb{R}^{n_u \times n_c}$ for $k \in \mathcal{K} : \{0, ..., N\}$. A trajectory with constant homotopy vector $\bar{\boldsymbol{\lambda}}$ is denoted by $\hat{x}(\bar{\boldsymbol{\lambda}})$, and $\hat{u}(\bar{\boldsymbol{\lambda}})$ respectively.

Since for an *r-step controllable* system (see Def. 4.1) only every r-th time $k = t \cdot r$, with $t \in \mathcal{T} = \{0, ..., \lfloor N/r \rfloor\} \subseteq \mathbb{N}_0$, can satisfy the homotopic state equation (102), new state notations are introduced. The notation is required since in Section 4.3.3, an auxiliary system is introduced which describes the transitioning behavior between homotopic trajectories depending on this timeline. Each state x_{tr} for $t \in \mathcal{T}$ corresponds to the new notation $z_t \hat{=} x_{tr}$, $z_{t+1} \hat{=} x_{(t+1)r}$ etc. Consequently, the homotopy state equation (102) can be rewritten to:

$$z_t(\boldsymbol{\lambda}_{tr}) = z_t^0 + D_{z_t} \boldsymbol{\lambda}_{tr}, \tag{103}$$

which satisfies (102) for every r-th time step. Simultaneously, a new homotopy state notation $\boldsymbol{\mu} \in \mathbb{R}^{n_c}$ describing every r-th homotopy state $\boldsymbol{\lambda}_{tr}$ is introduced. The new homotopy state notation $\boldsymbol{\mu}$ is connected as follows with the homotopy state $\boldsymbol{\lambda}$: $\boldsymbol{\mu}_t \hat{=} \boldsymbol{\lambda}_{tr}$, $\boldsymbol{\mu}_{t+1} \hat{=} \boldsymbol{\lambda}_{(t+1)r}$. Thus (103) can be described by:

$$z_t(\boldsymbol{\mu}_t) = z_t^0 + D_{z_t} \boldsymbol{\mu}_t. \tag{104}$$

Definition 4.3 *Let the set $\mathcal{L}$ contain all base trajectories $\hat{x}^i \in \mathcal{X}$ described by their homotopy values $\bar{\boldsymbol{\mu}}^i \in \mathcal{L} := \{\bar{\boldsymbol{\mu}}^0, ..., \bar{\boldsymbol{\mu}}^{n_c}\}$, with $\bar{\boldsymbol{\mu}}^i \in \{0, 1\}^{n_c}$ and let the sum over all components of $\boldsymbol{\mu}^i$ be $\sum_{i=1}^{n_c} \mu^i \leq 1$.*

4.3.2 Problem Definition

The task is to bring the system (99) from an initial state $x_{k_0} = x_s$ at time $k = k_0$ to a final state x_f, while avoiding collisions with a moving obstacle $\mathcal{P}_{x_k}$ in the state space and satisfying input constraints $u_k \in \mathcal{U} \subset \mathbb{R}^{n_u}$. The obstacle is considered to be known for a prediction horizon of H steps, and is defined as a polytopic region $\mathcal{P}_{x_{k+j}} := \{x_{k+j} |\ C_{k+j} x_{k+j} \leq d_{k+j}\} \subseteq \mathbb{R}^{n_x}$, with $j \in \mathcal{J} := \{0, ..., H\}$, $C \in \mathbb{R}^{c \times n_x}$, and $d \in \mathbb{R}^c$. The position prediction maybe obtained from estimation using an appropriate model or for communicated information on the planned motion of $\mathcal{P}_{x_{k+j}}$. The goal is to find a feasible and optimized trajectory $\hat{x}^*, \hat{u}^*$ by solving the optimization problem:

$$\begin{aligned} &\min_{x_{k_0+j},\ u_{k_0+j}} J(x_{k_0+j},\ u_{k_0+j}) \\ =&\min_{x_{k_0+j},\ u_{k_0+j}} \sum_{j=0}^{H} (x_{k_0+j} - x_f)^T Q (x_{k_0+j} - x_f) + (u_{k_0+j} - u_f)^T R (u_{k_0+j} - u_f) \qquad (105) \\ &\text{s.t.} \quad (99),\ u_{k+j} \in \mathcal{U},\ x_{k_0+j} \notin \mathcal{P}_{x_{k_0+j}}, \forall j \in \mathcal{J},\ x_{k_0} = x_s, \end{aligned}$$

for the prediction horizon H. The performance function is quadratic with positive-definite weighting matrices $Q \in \mathbb{R}^{n_x \times n_x}$, and $R \in \mathbb{R}^{n_u \times n_u}$. The control problem is implemented in a receding horizon fashion. Within this implementation, only the first step of the control strategy is applied, and the calculation is repeated for an incremented initial time $k_0 := k_0 + 1$. The initial state for the next iteration is then given by the actual measured state. Due to the non-convex obstacle avoidance constraint, a possible approach to solve the considered problem is by MIP, or specifically mixed-integer quadratic programming (MIQP). This leads typically to a large number of binary variables used to formulate, whether x_{k_0+j} lies outside of any bounding plane of $\mathcal{P}_{\S_{\|\prime+|}}$ for each point of time $k_0 + j$, $j \in \mathcal{J}$. Even if the obstacle and x_{k_0+j} are far away from each other in the considered time horizon, the problem must nevertheless be formulated as a MIQP, leading to a large share of unnecessary computations.

The method presented here is based on computing offline trajectories to build a homotopy function, which spans a region for circumventing the obstacle. This enables a fast online procedure to bring the system to a homotopic trajectory which passes the moving obstacle. In order to enable that a solution of the problem can be found by this method, the following assumption is introduced:

Assumption 4.1 *Let the set of trajectories $\mathcal{X}$ contain at least one trajectory $\hat{x}^i$, $i \in \mathcal{M}$, such that for any $\hat{x}^i_k \in \hat{x}^i$ a collision with the obstacle can be avoided: $x^i_k \notin \mathcal{P}_{x_k}$, $\forall k \in \{0, ..., N\}$.*

While the violation of this assumption implies an ill-posed problem, the assumption alone is not sufficient to guarantee a trajectory free of collision , since the transition from the current trajectory to a homotopic feasible one must be also realized without collisions.

4.3.3 Transitions between Homotopic Trajectories

Consider the task of steering (99) from a state $z_{t_0} \hat{=} x_{k_0}$ satisfying (104) to a future state on a homotopic trajectory. Let z_{t_0} be identified by $\boldsymbol{\mu}_{t_0}$, and the new trajectory (i.e. the goal of the transition) be referenced by $\bar{\boldsymbol{\mu}}$. While the inputs for the currently executed trajectory and the targeted one are known from the homotopy function (102), the transition from the actually executed trajectory to the final one requires additional inputs δu_k, leading to the overall input:

$$\tilde{u}_k(\boldsymbol{\lambda}_k) := u_k(\boldsymbol{\lambda}_k) + \delta u_k. \tag{106}$$

To restrict the speed of changing between homotopic trajectories, constraints on δu_k are formulated:

$$\delta u_{min} \leq \delta u_k \leq \delta u_{max}, \tag{107}$$

for all $k \in \mathcal{K}$, with $\delta u_{min}, \delta u_{max} \in \mathbb{R}^{n_u}$.

Now since every r-th state $x_{tr} \hat{=} z_t$ is reachable according to Def. 4.1 (with $\texttt{rank}(R) = n_x$), an auxiliary linear time-varying (LTV) system is defined, which describes the change of the homotopy state $\boldsymbol{\mu}_t$, $t \in \mathcal{T}$ to a final value denoted by $\bar{\boldsymbol{\mu}}$ as follows:

$$\boldsymbol{\mu}_{t+1} = \tilde{A}_t(\boldsymbol{\mu}_t - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}. \tag{108}$$

Since the transitioning behavior is freely parameterizable, the matrix $\tilde{A}_t$ is chosen to be time-varying and diagonal $\tilde{A}_t = diag(a_{i,t}) \in \mathbb{R}^{n_c \times n_c}$ with elements $a_{i,t}$, where index i denotes the i-th element, and t specifies the point of time. Additionally the variables $a_{i,t}$ have to satisfy $|a_{i,t}| \leq 1$ for all $i \in \{1, ..., n_c\}$ and $t \in \mathcal{T}$, for reasons of stability. Thus, (108) describes how the homotopy values $\boldsymbol{\mu}_t$ (or respectively every r-th value $\boldsymbol{\lambda}_{tr}$) evolves from its current value to $\bar{\boldsymbol{\mu}}$ in terms of the matrices $\tilde{A}_t$.

Lemma 4.2 *The input constraints* (107) *hold for every time step $k \in \mathcal{K}$, if the following inequality holds for all $t \in \mathcal{T}$:*

$$\Delta U_{min} \leq (B, AB, ..., A^{r-1}B)^{-1} D_{z_{t+1}} (\tilde{A}_t(\boldsymbol{\mu}_t - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_t) \leq \Delta U_{max},$$

with $\Delta U_{min} = (\delta u_{min}^T, ..., \delta u_{min}^T)^T \in \mathbb{R}^{n_u r}$, *and* $\Delta U_{max} = (\delta u_{max}^T, ..., \delta u_{max}^T)^T \in \mathbb{R}^{n_u r}$.

Proof 2 *Starting from an initial state* $z_{t_0}(\boldsymbol{\mu}_{t_0})$, *the state vector* $z_t(\boldsymbol{\mu}_t)$ *can be described by the following equation:*

$$z_t(\boldsymbol{\mu}_t) = A^{tr-t_0 r} z_{t_0}(\boldsymbol{\mu}_{t_0}) + \sum_{l=t_0 r}^{tr-1} A^{tr-l-1} B \tilde{u}_l(\boldsymbol{\lambda}_l). \tag{109}$$

Starting from time $t-1$, (109) *determines* $z_t(\boldsymbol{\mu}_t)$ *by:*

$$z_t(\boldsymbol{\mu}_t) = A^r z_{t-1}(\boldsymbol{\mu}_{t-1}) + \sum_{l=tr-r}^{tr-1} A^{tr-l-1} B \tilde{u}_l(\boldsymbol{\lambda}_l). \tag{110}$$

The value $\boldsymbol{\lambda}_{tr-r}$ *in* (110) *corresponds to* $\boldsymbol{\lambda}_{tr-r} \mathrel{\hat{=}} \boldsymbol{\mu}_{t-1}$. *The other intermediate values of* $\boldsymbol{\lambda}_l$ *for* $l \in \{tr-r+1, ..., tr-1\}$ *are located in between* $\boldsymbol{\mu}_{t-1}$ *and* $\boldsymbol{\mu}_t \mathrel{\hat{=}} \boldsymbol{\lambda}_{tr}$. *Since this intermediate values are not existing for an* r*-step controllable system, these values are set to:* $\boldsymbol{\mu}_{t-1}$, *corresponding to a zero-order hold of* $\boldsymbol{\mu}_{t-1}$. *Furthermore, the states* $z_{t-1}(\boldsymbol{\mu}_{t-1})$ *on the right side and* $z_t(\boldsymbol{\mu}_t)$ *on the left are replaced according to* (104), *one gets:*

$$\begin{aligned} z_t^0 + D_{z_t}\boldsymbol{\mu}_t = A^r(z_{t-1}^0 + D_{z_{t-1}}\boldsymbol{\mu}_{t-1}) + \\ (B, AB, ..., A^{r-1}B) \begin{pmatrix} u_{tr-1}^0 + D_{u_{tr-1}}\boldsymbol{\mu}_{t-1} + \delta u_{tr-1} \\ u_{tr-2}^0 + D_{u_{tr-2}}\boldsymbol{\mu}_{t-1} + \delta u_{tr-2} \\ \vdots \\ u_{tr-r}^0 + D_{u_{tr-r}}\boldsymbol{\mu}_{t-1} + \delta u_{tr-r} \end{pmatrix}. \end{aligned}$$

Replacing $\boldsymbol{\mu}_t$ *on the left side according to* (108) *and rewriting the terms leads to:*

$$\begin{aligned} z_t^0 + D_{z_t}(\tilde{A}_{t-1}(\boldsymbol{\mu}_{t-1} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}) = \underbrace{A^r z_{t-1}^0 + (B, AB, ..., A^{r-1}B) \begin{pmatrix} u_{tr-1}^0 \\ u_{tr-2}^0 \\ \vdots \\ u_{tr-r}^0 \end{pmatrix}}_{z_t^0} + \\ \underbrace{\left(A^r D_{z_{t-1}} + (B, AB, ..., A^{r-1}B) \begin{pmatrix} D_{u_{tr-1}} \\ D_{u_{tr-2}} \\ \vdots \\ D_{u_{tr-r}} \end{pmatrix} \right)}_{D_{z_t}} \boldsymbol{\mu}_{t-1} + (B, AB, ..., A^{r-1}B) \begin{pmatrix} \delta u_{tr-1} \\ \delta u_{tr-2} \\ \vdots \\ \delta u_{tr-r} \end{pmatrix}. \end{aligned} \tag{111}$$

Incrementing $t := t+1$, *solving* (111) *for the vector of the additional inputs, and constraining*

it from the left and right side yields:

$$\Delta U_{min} \leq \begin{pmatrix} \delta u_{tr+r-1} \\ \delta u_{tr+r-2} \\ \vdots \\ \delta u_{tr} \end{pmatrix} \leq \Delta U_{max}, \tag{112}$$

which equals (4.2) *and therefore satisfies* (107).

For the scenario as described in Sec. 4.3.2, it is assumed that the system should transition as fast as possible to a collision-free and optimized trajectory (described by the transition from $\boldsymbol{\mu}_t$ to $\bar{\boldsymbol{\mu}}$) after detection of the obstacle $\mathcal{P}_x$. Selecting a suitable candidate $\bar{\boldsymbol{\mu}}$ is discussed in the next section. In order to make the best possible use of the constrained input signal (4.2), the matrices $\tilde{A}_t$, are chosen to be time-varying, since $D_{z_{t+1}}$ in (4.2) is also changing over time.

For fast transitioning between homotopic trajectories, the eigenvalues of the matrices $\tilde{A}_t$ must be minimized subject to the input constraints. As given in ([93]), the eigenvalue problem (EVP) is to minimize the maximum eigenvalue of $\tilde{A}_t$. The matrix $\tilde{A}_t$ depends affinely on the variables $a_{i,t}$, such that $\tilde{A}_t = W_{0,t} + \sum_{i=1}^{n_c} a_{i,t} W_{i,t}$ is a linear matrix. This allows to formulate the convex EVP with the input constraints as a semi-definite program (SDP):

$$\min_{a_{i,t},\gamma} \quad \gamma \tag{113}$$

$$\text{s.t.} \quad \tilde{A}_t - \gamma I \leq 0, \ \forall t \in \mathcal{T} \tag{114}$$

$$(4.2), \ \forall \{\boldsymbol{\mu}_t, \bar{\boldsymbol{\mu}}\} \in \{0,1\}^{n_c}, \ \forall t \in \mathcal{T}. \tag{115}$$

The constraint (4.2) has to be satisfied for all possible combinations resulting from $\boldsymbol{\mu}_t \in \{0,1\}^{n_c}$ and $\bar{\boldsymbol{\mu}} \in \{0,1\}^{n_c}$, and for all times $t \in \mathcal{T}$. Because $\tilde{A}_t$ is chosen to be diagonal, (113)-(115) reduces to a general linear programming problem. The dynamics (108) specifies the transitioning behavior between homotopic trajectories such that the transition is as fast as possible with respect to the input constraints.

4.3.4 Online collision avoidance

The online procedure starts by determining the cost optimal homotopic goal trajectory encoded by $\bar{\boldsymbol{\mu}}^*$ for a given prediction horizon H, while first neglecting the obstacle. If the resulting trajectory fails to be free of collision, a fast online procedure is initiated, which determines an optimized value for $\bar{\boldsymbol{\mu}}$ with respect to the performance function in (105), such that the system passes the obstacle. The advantage of restricting the problem of collision avoidance

to homotopic trajectories is that time steps with collisions in the homotopy space can be identified quickly, and that only these time steps are relevant for the circumvention procedure.

Determining the Best Homotopic Trajectory If for a certain time no obstacles are present, the obvious choice of the controller is to select the best homotopic trajectory, i.e. the task is to find an optimal value of $\bar{\boldsymbol{\mu}}$. It can be shown that the cost function (105) can be reformulated from $J(x_{k+j}, u_{k+j})$ to $J(\bar{\boldsymbol{\mu}})$, $j \in \mathcal{J}$. This essentially reduces the search space of the problem from $(H+1) \cdot n_u \cdot n_x$ to n_c variables. The initial homotopy vector is $\boldsymbol{\mu}_{t_0} = \boldsymbol{\mu}_{init}$ at time $t = t_0$, and the corresponding state x_{k_0} is known with $k_0 = t_0 \cdot r$ according to (104). The prediction of up to H time steps leads to a considered time span $k \in \mathcal{K}_H = \{k_0, ..., k_0 + H\}$, beginning from k_0, or respectively $t \in \mathcal{T}_H = \{t_0, ..., t_0 + \lfloor H/r \rfloor\}$. The optimization problem can be reformulated to:

$$\min_{\bar{\boldsymbol{\mu}}} \quad \sum_{j=0}^{H} (x_{t_0 r+j} - x_f)^T Q (x_{t_0 r+j} - x_f)$$

$$+ (u_{t_0 r+j} - u_f)^T R (u_{t_0 r+j} - u_f) \tag{116}$$

s.t. :

$$x_{k+1} = A x_k + B \tilde{u}_k, \ \forall k \in \{tr, ..., tr + r - 1\} \tag{117}$$

$$\tilde{u}_k = u_k^0 + D_{u_k} \boldsymbol{\mu}_t + \delta u_k, \ \forall k \in \{tr, ..., tr + r - 1\} \tag{118}$$

$$(B, AB, ..., A^{r-1}B)^{-1} D_{z_{t+1}} (\tilde{A}_t (\boldsymbol{\mu}_t - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} - \boldsymbol{\mu}_t) = \begin{pmatrix} \delta u_{tr+r-1} \\ \delta u_{tr+r-2} \\ \vdots \\ \delta u_{tr} \end{pmatrix} \tag{119}$$

$$\boldsymbol{\mu}_{t+1} = \tilde{A}_t (\boldsymbol{\mu}_t - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}} \tag{120}$$

$$z_t(\boldsymbol{\mu}_t) = z_t^0 + D_{z_t} \boldsymbol{\mu}_t \ t \in \mathcal{T}_H \tag{121}$$

The values of $\tilde{A}_t$ are determined according to Sec. 4.3.3. As noticed, the optimization problem contains no constraints on the input signal, while these have already been integrated in the matrices $\tilde{A}_t$ by the solution of (113)-(115). The optimal solution of the homotopy value, based on the initial time t_0, is given by $\bar{\boldsymbol{\mu}}^*_{|t_0}$. The index denotes that the optimal homotopy value is computed based on time t_0 and is constant for the future times of the prediction horizon. The resulting optimal state trajectory is denoted by $x^*_{t_0 r+j|t_0 r}$, $j \in \mathcal{J}$. The computation of (116)-(121) can be performed very fast because the quadratic optimization problem is low-dimensional, and only subject to equality constrains.

Transforming the Moving Obstacle into the Homotopy Space Now, the resulting state trajectory is checked against collision, i.e. $x^*_{t_0r+j|t_0r} \notin \mathcal{P}_{x_{t_0r+j}}$ is evaluated $\forall j \in \mathcal{J}$. If collisions are found, the state space obstacle $\mathcal{P}_{x_{t_0r+j}}$ is transformed for each j into the homotopy space, denoted by $\mathcal{P}_{\boldsymbol{\lambda}_{t_0r+j}}$. Since it was shown that an *r-step controllable* system conforms only every r-th time step to the homotopy function (102), a collision check may only be allowed to any r-th step. The related polytope is $\mathcal{P}_{x_{t_0r+sr}} := \{x_{t_0r+sr} |\ C_{t_0r+sr}\ x_{t_0r+sr} \leq d_{t_0r+sr}\} \subseteq \mathbb{R}^{n_x}$, or respectively with $s \in \{0, ..., \lfloor H/r \rfloor\}$:

$$\mathcal{P}_{z_t} := \{z_t |\ C_{tr}\ z_t \leq d_{tr}\},\ t \in \mathcal{T}_H. \tag{122}$$

The polytope $\mathcal{P}_{z_t}$ mapped into the homotopy space is denoted by $\mathcal{P}_{\boldsymbol{\mu}_t}$ and determined by inserting (104) into (122):

$$C_{tr}(z_t^0 + D_{z_t}\boldsymbol{\mu}_t) \leq d_{tr} \tag{123}$$

$$\underbrace{C_{tr}D_{z_t}}_{C_{\boldsymbol{\mu}_t}}\boldsymbol{\mu}_t \leq \underbrace{d_{tr} - C_{tr}z_t^0}_{d_{\boldsymbol{\mu}_t}} \tag{124}$$

$$\Rightarrow \mathcal{P}_{\boldsymbol{\mu}_t} := \{\boldsymbol{\mu}_t | C_{\boldsymbol{\mu}_t}\ \boldsymbol{\mu}_t \leq d_{\boldsymbol{\mu}_t}\}. \tag{125}$$

The polytope $\mathcal{P}_{\boldsymbol{\mu}_t}$ exists for time t if the intersection:

$$\mathcal{P}_{\boldsymbol{\mu}_t} \cap z_t(\boldsymbol{\mu}_t) \neq \emptyset \tag{126}$$

is non-empty. This can be easily checked if the following linear optimization problem has a feasible solution:

$$\min_{\boldsymbol{\mu}_t} \quad b^T\ \boldsymbol{\mu}_t \tag{127}$$

$$\text{s.t.} \quad C_{\boldsymbol{\mu}_t}\ \boldsymbol{\mu}_t \leq d_{\boldsymbol{\mu}_t} \tag{128}$$

$$\boldsymbol{\mu}_t \in [0,1]^{n_c},\ \sum_{i=1}^{n_c} \mu_t^i \leq 1, \tag{129}$$

where $b \neq 0 \in \mathbb{R}^{n_c}$ can be chosen arbitrarily. The set of all times t for which (127)-(129) is satisfied, is denoted by:

$$\mathcal{T}_I := \{t | (127) - (129) \text{ satisfied}\} \subseteq \mathcal{T}_H. \tag{130}$$

If $\mathcal{T}_I = \emptyset$, no collision of the moving obstacle and the system occurs for $t \in \mathcal{T}_H$.

The transformation of $\mathcal{P}_{z_t}$ into $\mathcal{P}_{\boldsymbol{\mu}_t}$ may lead to redundant half-planes in $\mathcal{P}_{\boldsymbol{\mu}_t}$, which can be removed by linear programming, as described in [94]. Let the w-th row of $C_{\boldsymbol{\mu}_t}$ be denoted by $C^w_{\boldsymbol{\mu}_t}$, and the w-th component of $d_{\boldsymbol{\mu}_t}$ by $d^w_{\boldsymbol{\mu}_t}$. Further, let $\mathcal{W}_t := \{1, ..., c_t\}$ be the set of all inequalities at time t. The w-th row is redundant, iff the linear program:

$$\max_{\boldsymbol{\mu}_t} \quad C^w_{\boldsymbol{\mu}_t}\, \boldsymbol{\mu}_t \tag{131}$$

$$\text{s.t.} \quad C^i_{\boldsymbol{\mu}_t}\, \boldsymbol{\mu}_t \leq d^i_{\boldsymbol{\mu}_t},\ \forall i \in \mathcal{W}_t \backslash \{w\} \tag{132}$$

has an optimal value less or equal to $d^w_{\boldsymbol{\mu}_t}$. Solving the problem c_t-times yields for $t \in \mathcal{T}_I$, the simplified polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu}_t} := \{\boldsymbol{\mu}_t | \tilde{C}_{\boldsymbol{\mu}_t} \boldsymbol{\mu}_t \leq \tilde{d}_{\boldsymbol{\mu}_t}\}$ with $\tilde{C}_{\boldsymbol{\mu}_t} \in \mathbb{R}^{\tilde{c}_t \times n_c}$, $\tilde{d}_{\boldsymbol{\mu}_t} \in \mathbb{R}^{\tilde{c}_t}$, and w-th rows $\tilde{C}^w_{\boldsymbol{\mu}_t}$ and $\tilde{d}^w_{\boldsymbol{\mu}_t}$, $w \in \tilde{\mathcal{W}}_t := \{1, ..., \tilde{c}_t\}$.

Homotopy Control Algorithm (`HCA`) If a collision is possible, the goal is to select an optimized homotopy value $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$, with initial state $z_{t_0} \hat{=} x_{k_0}$ and corresponding $\boldsymbol{\mu}_{t_0}$ for which the resulting trajectory satisfies $x^*_{t_0 r+j|t_0 r} \notin \mathcal{P}_{x_{t_0 r+j}}$, $j \in \mathcal{J}$. The for computing $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$, see Alg. 10 can be separated into four parts.

1. Optimal trajectory, without obstacle (Alg. 10: line 0)

The algorithm 10 computes the optimal value $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$ for the prediction horizon H by solving the optimization problem (116)-(121). The optimal homotopy value $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$ defines the optimal state sequence $x^*_{t_0 r+j|t_0 r}$, $j \in \mathcal{J}$. The resulting trajectory is checked for collisions with $\mathcal{P}_{x_{t_0 r+j}}$ (line 1). If no collisions are found, the algorithm terminates successfully. If the trajectory collides with $\mathcal{P}_{x_{t_0 r+j}}$, the following collision avoidance procedure starts.

2. Best Free Base Trajectory (Alg. 10: line 2) First, the costs for all $\bar{\boldsymbol{\mu}}^i \in \mathcal{L}$ are computed according to (116)-(121), and the transitions from the actually executed trajectory towards the base trajectories (encoded by $\bar{\boldsymbol{\mu}}^i$) are checked for collisions. The least costly solution $\bar{\boldsymbol{\mu}}^{i^*}$ within the set $\mathcal{L}$ is selected, leading to the best free resulting trajectory when transitioning to a base trajectory. This step prevents the following step from running into an infeasible branch, since the selected base trajectory is always feasible.

3. Tree-Search (Alg. 10: line 3-15) A best-first strategy is executed to decide how the moving obstacle should be passed, and to guarantee that the resulting trajectory of $z_t(\boldsymbol{\mu}_t)$ determined from (108), for all $t \in \mathcal{T}_I$ is free of collisions. A node of the tree models an element $g_p \in \mathcal{T}_I$, with p indexing the tree level starting with $p := 1$. The node g_p is said to be explored, if all successors $w_{g_p} \in \tilde{\mathcal{W}}_{g_p}$ are visited. If a node g_p is explored, the node is removed from the

set $\mathcal{T}_I$:

$$\mathcal{T}_I := \mathcal{T}_I \setminus g_p, \tag{133}$$

The best node based on g_p is denoted by $w^*_{g_p}$ and is stored in the set:

$$\mathcal{O} := \mathcal{O} \cup w^*_{g_p}. \tag{134}$$

The best-first search selects for node p one polytope $\tilde{\mathcal{P}}_{\boldsymbol{\mu}_{g_p}}$ with $g_p \in \mathcal{T}_I$. Then, each successor $w_{g_p} \in \tilde{\mathcal{W}}_{g_p}$ is determined by solving the optimization problem (116)-(121) with the additional constraints:

$$\tilde{C}^{w_{g_p}}_{\boldsymbol{\mu}_{g_p}} \bar{\boldsymbol{\mu}} \geq \tilde{d}^{w_{g_p}}_{\boldsymbol{\mu}_{g_p}} \tag{135}$$

$$\tilde{C}^{w_{g_p}}_{\boldsymbol{\mu}_{g_p}} \boldsymbol{\mu}_{g_p} \geq \tilde{d}^{w_{g_p}}_{\boldsymbol{\mu}_{g_p}} \tag{136}$$

$$\tilde{C}^{w_{g_p}}_{\boldsymbol{\mu}_{g_p}} \bar{\boldsymbol{\mu}}^{i^*} \geq \tilde{d}^{w_{g_p}}_{\boldsymbol{\mu}_{g_p}}. \tag{137}$$

Constraint (135) requires to select a goal trajectory $\bar{\boldsymbol{\mu}}$, which generally passes the obstacle $\tilde{\mathcal{P}}_{\boldsymbol{\mu}_{g_p}}$ along the w_{g_p}-th half-space. Since this does not ensure that the transient behavior during the change of the trajectories is also free of collision, constraint (136) is introduced, which prevents the system to collide at time g_p with $\tilde{\mathcal{P}}_{\boldsymbol{\mu}_{g_p}}$. The homotopy state $\boldsymbol{\mu}_{g_p}$ can be computed recursively from (108):

$$\boldsymbol{\mu}_{g_p} = \prod_{i=g_p-1}^{t_0} \tilde{A}_i(\boldsymbol{\mu}_{t_0} - \bar{\boldsymbol{\mu}}) + \bar{\boldsymbol{\mu}}. \tag{138}$$

The last constraint (137) guarantees, that the w_{g_p}-th half-space also contains the previously selected homotopy value of the best base trajectory $\bar{\boldsymbol{\mu}}^{i^*}$ from part (*2.*). This guarantees that a feasible solution of the tree-search can always be found. After the exploration of all successors $w_{g_p} \in \tilde{\mathcal{W}}_{g_p}$ at time g_p, the one producing the lowest costs is denoted by $w^*_{g_p}$ and stored in the set (134). The set $\mathcal{T}_I$ is updated according to (133).

The procedure is repeated by selecting a new $g_p \in \mathcal{T}_I$ and computing the successors $w_{g_p} \in \tilde{\mathcal{W}}_{g_p}$ according to (116)-(121) with the constraints (135)-(137), and additionally (135)-(137) for all $w_{g_p} \in \mathcal{O}$.

The tree-search terminates immediately if no more collisions between $z_t(\boldsymbol{\mu}_t)$ and $\mathcal{P}_{z_t}$, for all $t \in \mathcal{T}_I$ can be found (line 9-13), or if the set $\mathcal{T}_I$ is empty. The best-first search terminates with optimized homotopy value $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$.

Algorithm 10 Homotopy Control Algorithm (`HCA`)

Given: (99), $\hat{x}^i \in \mathcal{X}$, $\tilde{A}_t$, t_0, $k_0 = t_0 r$, H, $\mathcal{P}_{x_{t_0 r+j}}$, $\mathcal{T}_H$
Define: $\mathcal{O} := \emptyset$
0. compute $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$ according to (116)-(121)
1. **if** $\exists j \in \mathcal{J} : x_{t_0 r+j} \in \mathcal{P}_{x_{t_0 r+j}}$ **do**
2. compute the best free base trajectory $\bar{\boldsymbol{\mu}}^{i^*}$
3. transform state obstacles $\mathcal{P}_{x_{t_0 r+j}}$ into the homotopy space $\tilde{\mathcal{P}}_{\boldsymbol{\mu}_t}$ as described in Sec. 4.3.4 and determine intersection times $t \in \mathcal{T}_I$.
4. **while** $\mathcal{T}_I \neq \emptyset$ **do**
5. select $q_p \in \mathcal{T}_I$
6. compute for each each successor $w_{q_p} \in \tilde{\mathcal{W}}_{g_p}$ the corresponding optimal $\bar{\boldsymbol{\mu}}^*_{g_p}$ acc. (116)-(121) and (135)-(137), and additionally (135)-(137) for all elements of $\mathcal{O}$
7. store best successor $w^*_{q_p}$ into $\mathcal{O} := \mathcal{O} \cup w^*_{q_p}$
8. compute $z_t(\boldsymbol{\mu}_t)$ with the obtained $\bar{\boldsymbol{\mu}}^*_{g_p}$ and (108)
9. **if** $\exists t \in \mathcal{T}_H : z_t(\boldsymbol{\mu}_t) \in \mathcal{P}_{z_t}$ **do**
10. update $\mathcal{T}_I$ acc. to (133)
11. **else**
12. **break while**
13. **end if**
14. **end while**
15. optimal homotopy value $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$, and corresponding optimal trajectory $x^*_{t_0 r+j|t_0 r}$, with $j \in \mathcal{J}$
16. **if** $\exists j \in \mathcal{J} : x_{t_0 r+j} \in \mathcal{P}_{x_{t_0 r+j}}$ **do**
17. push $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$ towards base trajectory $\bar{\boldsymbol{\mu}}^{i^*}$ acc. to (139)
18. **end if**
19. $t_0 := t_0 + 1$
20. **end if**
Return $(\bar{\boldsymbol{\mu}}^*_{\cdot|t_0},\ x^*_{t_0 r+j|t_0 r})$ obtained from executing $u^*_{t_0 r+j|t_0 r}$

4. Pushing (Alg. 10: line 16-18) The pushing procedure is executed if a collision is detected for the intermediate states $x_{t_0 r+j}$, $j \in \mathcal{J}$. Hence, for time steps which are not considered during the collision avoidance in Step *3.*. This part then pushes the trajectory out of the obstacle. Therefore, the solution of the optimized homotopy value $\bar{\boldsymbol{\mu}}^*_{\cdot|t_0}$ is changed piecewise towards the known collision free trajectory encoded by $\bar{\boldsymbol{\mu}}^{i^*}$ of step *2.*, according to:

$$\bar{\boldsymbol{\mu}}^*_{\cdot|t_0} := \bar{\boldsymbol{\mu}}^*_{\cdot|t_0} + \alpha(\bar{\boldsymbol{\mu}}^{i^*} - \bar{\boldsymbol{\mu}}^*_{\cdot|t_0}), \tag{139}$$

by iteratively increasing $\alpha \in [0, 1]$.

The complete procedure is shown in Alg. 10. After computation according to Alg. 10, the first step of the control strategy is applied to the system, the prediction horizon is shifted one step forward, and the computation is repeated.

4.3.5 Example

The proposed homotopy control method is applied to an abstracted version of the robotic use case: assume that the motion of the robotic end-effector is represented as a point mass moving in a 3-D space, which must not collide with a moving obstacle represented by an over-approximating box. The dynamic model of the point mass is specified as:

$$\ddot{x}(t) = -\frac{c}{m}\dot{x}(t) + \frac{1}{m}F_x(t) \tag{140}$$

$$\ddot{y}(t) = -\frac{c}{m}\dot{y}(t) + \frac{1}{m}F_y(t) \tag{141}$$

$$\ddot{z}(t) = -\frac{c}{m}\dot{z}(t) + \frac{1}{m}F_z(t), \tag{142}$$

with an input vector $\boldsymbol{u}(t) = (F_x(t), F_y(t), F_z(t))$, the mass m, and a friction constant c. The variables x, y, z describe the cartesian coordinates of the 3-D space in this example. The state vector of system (140)-(142) is given by $\boldsymbol{x}(t) = (x(t), \dot{x}(t), y(t), \dot{y}(t), z(t), \dot{z}(t))^T$. The system is discretized (ZOH) with discretization time $T = 0.1$ seconds (s). Using parameters $m = 2$ and $c = 1$, the discrete-time system $\dot{\boldsymbol{x}}_{k+1} = A\boldsymbol{x}_k + B\boldsymbol{u}_k$ has the following matrices:

$$A = \begin{pmatrix} 1 & 0.0975 & 0 & 0 & 0 & 0 \\ 0 & 0.9512 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.0975 & 0 & 0 \\ 0 & 0 & 0 & 0.9512 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0.0975 \\ 0 & 0 & 0 & 0 & & 0.9512 \end{pmatrix}, \; B = 1e^{-2} \begin{pmatrix} 0.25 & 0 & 0 \\ 4.88 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 4.88 & 0 \\ 0 & 0 & 0.25 \\ 0 & 0 & 4.88 \end{pmatrix}. \tag{143}$$

The system is 2-step controllable according to Def. 4.1. Totally three pairs of base trajectories $(\hat{x}^i, \hat{u}^i)$ are chosen to span a region for circumvention, i.e. $n_c = 2$. The final time for all $i \in \mathcal{M}$, is $N = 60$. All state trajectories start at the common state $x_0^i = x_s = [0, 0, 0, 0, 0, 0]^T$, and end in the final state $x_N^i = x_s = [5, 0, 5, 0, 5, 0]^T$. The base trajectories are shown in Fig. 33 by the three black dotted lines. The upper and lower bound on the additional input δu_k, which limit the speed of motion between the homotopic trajectories, is given by $\delta u_{max} = [100, 100, 100]^T$ and $\delta u_{min} = [-100, -100, -100]^T$. The obstacle $\mathcal{P}_x$ starts at time $k = 0$ at the upper right area in the state space (x, y, z), shown by the transparent polytope with dashed edges in Fig. 33. It moves towards the lower left region until time $k = 10$, where it remains for the rest of the time. The final position is shown by the green polytope. For better illustration, the cost function (116) is parametrized by $Q \in \mathbb{R}^{3\times3}$ chosen as the identity matrix and $R \in \mathbb{R}^{2\times2}$ as diagonal matrix with values $1e^{-3}$, implying that trajectories with lower costs reach the final state along a straighter trajectory.

The computation is performed in Matlab using a PC with an Intel Core i7 (3.4GHz). The blue trajectory (see Fig. 33) is determined by the homotopy approach starting at initial time $k_0 = 0$ with a prediction horizon of $H = 10$. It can be seen that the trajectory first follows an optimal homotopic trajectory, and from $k = 5$ reacts to the moving polytope by transitioning to another homotopic trajectory. This can also be seen in Fig. 34 by the yellow colored bars, which show the computation times for the homotopy control method with receding horizon. It can be identified that during the first steps the computation time is very low, with approx. 3 milliseconds (ms). When the obstacle becomes relevant during the prediction horizon, the computation time raises up to 18 ms, and finally falls again when the obstacle is passed. Since the system is 2-step controllable in $r = 2$ steps, the computation has to be performed only in every second time step, which results in a total amount of 30 time steps of computation, as shown in Fig. 34.

For the same scenario, but with a prediction horizon of $H = 60$, the solution of the homotopy control method is given by the red trajectory in Fig. 33. Here, it can be seen that the system reacts at the very beginning to the moving obstacle, and finds a solution which is 2% better in costs, compared to the first case with smaller prediction horizon. The computation time (see Fig. 34 blue bars) is approx. 35 ms at the beginning. Compared to the smaller prediction horizon, the computational load for the entire time is larger. However, for comparison, the computation times of a standard MPC based on MIP solved with CPLEX yields solver times of approx. 800 ms at each iteration (while the costs are 37% better for the second scenario).

Thus, the computation times are drastically reduced by approx. 96% overall, when using

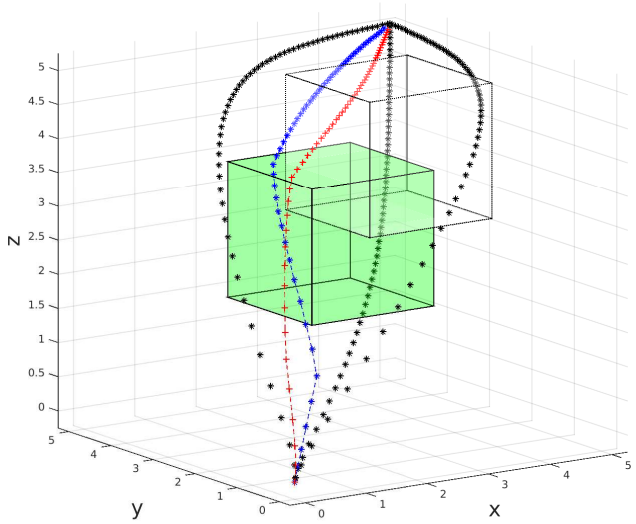

Figure 33: Simulation of the base trajectories $\hat{x}^1$, $\hat{x}^2$, $\hat{x}^3$ (black stars), the obtained trajectory (blue stars) by the homotopy control method with $H = 10$, and the trajectory (red crosses) for $H = 60$. The obstacle $\mathcal{P}_{x_k}$ moves from initial position (transparent dashed polytope) to the final position (green polytope).

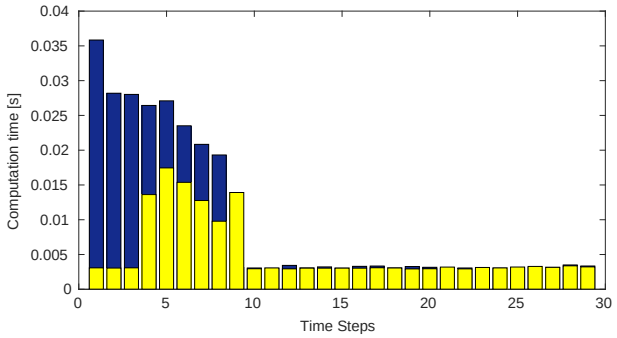

Figure 34: Comparison between computation times of Alg. 10 with prediction horizon $H = 10$ (yellow bars), and $H = 60$ (blue bars). The computation times are shown for the entire simulation of the receding horizon scheme.

the proposed scheme.

5 Concluding Remarks and Transfer to Applications

In this report, we described the developments under work package 2 of the *UnCoVerCPS* project regarding distributed optimization and MPC solutions with enhanced real time computability. We showed their performance in various simulation examples, most of which refer to the application domains studied within the *UnCoVerCPS* project. The plan is now to transfer these techniques to the case studies in work package 5, dealing with the realization of cyber-physical systems. This is expected to require some modeling effort, and some adaptation of the techniques described in this report.

We next outline briefly what the foreseen applications are, with reference to the case studies on smart grid, automated driving, and human-robot interaction.

Smart grid

Scalability of the methods in Sections 2.2 and 2.3 for smart grid energy management of a district network will be tested. To this purpose the compositional modeling framework described in Deliverable 5.1 and further developed in [95] will be used. We shall then implement the stochastic extension in Section 2.5 of the distributed optimization scheme in Section 2.2 to achieve probabilistic robustness for a smart grid energy management problem, where renewable power generators like solar plants are present.

Automated driving

The method proposed in Section 4.2 for DMPC of systems with mixed inputs will be applied to and evaluated for the case study of cooperative automated driving, as investigated in Task 5.3 of work package 5. The planning of a driving path of a single automated vehicle (under consideration of the motion of other neighbored cars) can be cast into the MPC setting addressed in Section 4.2: if within the planning at time k, the vehicles first communicate their driving plans as computed in the previous step, the constraints for the local optimization problems can be derived, leading to the problem formulation as specified in Section 4.2. The solution according to the proposed procedure will determine a feasible driving path over the prediction horizon, including corresponding trajectories of continuous (e.g. steering angle) and discrete inputs (e.g. selection of the gear, or a driving decision such as changing lane). Future work will evaluate the DMPC scheme for different driving maneuvers, as e.g. overtaking procedures, and it will investigate how cooperative behavior can be obtained by appropriate choice of the local cost functions.

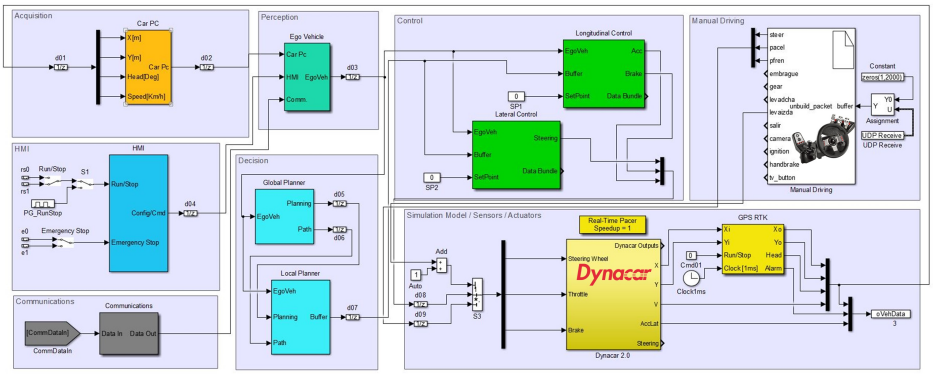

Figure 35: Architecture framework for automated Driving in Simulink

Dynacar simulator

We shall test the real time capabilities of the MPC method proposed in Section 4.2 for the design of a driving path for a single automated vehicle in the simulation tool Dynacar developed by Tecnalia. Dynacar focuses on vehicle dynamics and provides a high-fidelity vehicle physics simulation, which is combined with a Pacejka tyre model and sub-models for elements like the engine, transmission, steering system, braking system, aerodynamics, among other. Moreover, it enables to integrate components and subsystems of the Electric-Electronic architecture of the vehicle, such es ECUs (electronic control units) and power propulsion elements.

Figure 36: Example of the double lane change maneuvers in three phases, with Dynacar

Dynacar allows real-time and accelerated-time simulations. The real-time capability is very valuable, as, combined with its notable modularity and interfacing options, it permits to execute tests with driver-in-the-loop (DiL) and hardware-in-the-loop (HiL) setups, for instance for ECU (Electronic Control Unit) development, integrated into Simulink blocks (see Figure 35). A testing methodology for the validation of control algorithms for automated vehicles is available, which is modular and can be adapted to a general control architecture for automated driving. It enables a good trajectory definition, cooperative maneuvers and virtual validation with different kinds of vehicles and scenarios. It can be adapted to test cooperative maneuvers where vehicles communicate their driving plans. The MPC solution in Section 4.2 for a double lane change maneuver can be implemented in Matlab-Simulink, which permits to integrate

C-code functions and provide convenient interfacing capabilities. The double lane change maneuver is defined by a series of constrains and conditions given by the perception systems, as detection of the environment and obstacles, trajectory considerations and communication with another vehicles. Figure 36 shows the three phases of an overtaking with another vehicle coming in front.

Extension to lane changing and merging of multiple vehicles

We shall consider a multi-lane scenario with a mixture of communicating automated vehicles and non-communicating manually driven vehicles as depicted in Figure 37. The automated vehicles in the given scenario have to communicate in order to cooperatively control the longitudinal motion of the cars along their lane. In this way gaps can be cooperatively opened and the alignment to gaps and other cars can be regulated in preparation to, as well as during execution of a lane-change. Optimal control inputs are calculated in a distributed manner. Each vehicle computes its own current and future control inputs in such a way that constraints imposed by other vehicles are satisfied.

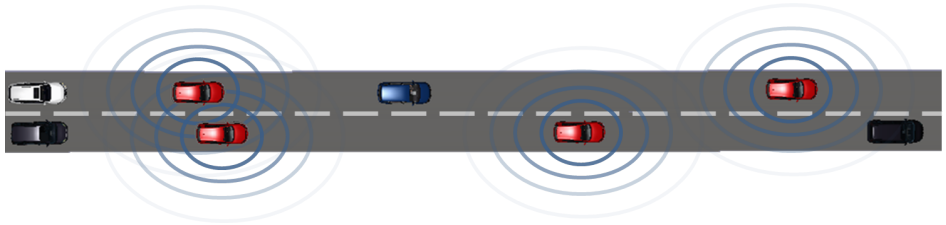

Figure 37: Cooperative lane-changing in mixed traffic

There are two possible ways to integrate the DMPC method into the overall safety concept for automated driving, which is developed in Task 5.3 of work package 5.

The first possibility is as a stand-alone application: in this case, the DMPC method would be supplied with an environment-model and ego-vehicle state derived from the vehicle sensors. It then directly calculates control inputs, which are sent to the actuators. In this case, it would have to guarantee invariant safety of the selected control inputs at all points of time. This could require guarantees for termination in a fixed time-frame from the employed optimization algorithm. Additionally, the DMPC module would have to control longitudinal and lateral motion of a vehicle and make discrete decisions, such as which gap to merge into.

The second possible deployment strategy is to integrate the DMPC approach into a hierarchical control architecture such as described in [96] or a different approach in [97]. This would allow other components to provide lateral control, maneuver switching and invariant safety guarantees. The robust, cooperative lane-changing in mixed traffic scenario has been investigated and demonstrated in simulation in [97] for two cooperating, automated vehicles

and four un-cooperative, simulated vehicles on a high-resolution map of the ring-road of Braunschweig, see Figure 38.

Figure 38: Demonstration of use case without DMPC and with focus only on safety

The architecture of the approach is visualized in Figure 39: Each vehicle is equipped with four layers. The High-Level Behaviors at the top-most layer compute reference set points, which enable the vehicle to follow lanes and to execute lane-changes in normal driving situations. The Supervisor layer guarantees invariant safety, by switching from the desired reference set points to emergency reference set points, if the execution of the desired reference set points would endanger invariant safety. Car-to-car communication is initiated by the High-level Behaviors and is passed through the Supervisor layer, as negotiations between cars can be safety-critical. The Low-level Control layer stabilizes the dynamics of the Physical Vehicle according to the requested set points and is considered in a closed-loop vehicle model by the Supervisor and the High-Level Behaviors.

The previous demonstration and the investigations in Task 5.3 of work package 5 are focused on guaranteeing invariant safety and do not consider optimality of the cooperative control. For example the High-Level Behavior layer was realized as a simple, linear, hybrid automaton.

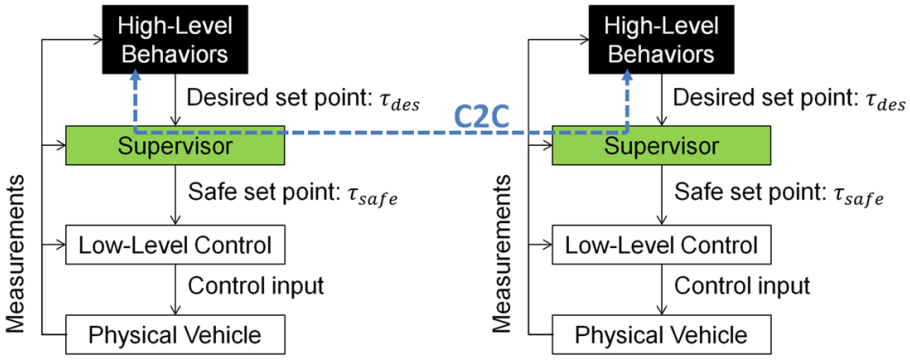

Figure 39: Hierarchical control architecture for safe, cooperative control

The automated driving application could therefore gain significantly by replacing parts of the existing High-Level Behavior by DMPC techniques described in this report. As a first step, a preliminary study should evaluate whether execution time constraints and communication bandwidth constraints can be satisfied for realistic problem-sizes. The next step could be an integration into the previously employed, sub-microscopic traffic simulation and a subsequent analysis of robustness and impact on individual and global objectives.

Human-robot interaction

The technique described in Section 4.3 appears to be particularly amenable to (and is motivated by) the use case of human-robot interaction as investigated in Task 5.4 of work package 5: the optimized trajectories established offline in the technique refer to the nominal operation of the robot, i.e. the behavior to be established in free operating space. If a human is monitored to be within this space during online operation, its position is mapped into the polytopic obstacle region. The online control procedure then identifies a feasible homotopic trajectory of the robotic end-effector, i.e. one which is free of collision with the moving human. It will be investigated for different scenarios up to which speed of human motion the control procedure will reliably provide suitable motion paths for the robot.

References

[1] L. Deori, K. Margellos, and M. Prandini, "On the connection between Nash equilibria and social optima in electric vehicle charging control games," in *20th World Congress of the International Federation of Automatic Control (IFAC 2017), Toulouse, France*, July 2017.

[2] L. Deori, K. Margellos, and M. Prandini, "Nash equilibria in electric vehicle charging control games: Decentralized computation and connection with social optima," 2016. Submitted. Online preprint arXiv:1612.01342.

[3] K. Margellos, A. Falsone, S. Garatti, and M. Prandini, "Proximal minimization based distributed convex optimization," in *2016 American Control Conference (ACC)*, pp. 2466–2471, July 2016.

[4] K. Margellos, A. Falsone, S. Garatti, and M. Prandini, "Distributed constrained optimization and consensus in uncertain networks via proximal minimization," *IEEE Transactions on Automatic Control*, 2016. Submitted. Online preprint arXiv:1603.02239.

[5] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Distributed constrained convex optimization and consensus via dual decomposition and proximal minimization," in *55th IEEE Conference on Decision and Control (CDC)*, pp. 1889–1894, Dec 2016.

[6] A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Dual decomposition for multi-agent distributed optimization with coupling constraints," *Automatica*, 2017. Accepted. Online preprint arXiv:1607.00600.

[7] L. Deori, K. Margellos, and M. Prandini, "On decentralized convex optimization in a multi-agent setting with separable constraints and its application to optimal charging of electric vehicles," in *55th IEEE Conference on Decision and Control (CDC)*, pp. 6044–6049, Dec 2016.

[8] L. Deori, K. Margellos, and M. Prandini, "Regularized jacobi iteration for decentralized convex optimization with separable constraints," 2017. Submitted. Online preprint arXiv:1604.07814.

[9] D. Ioli, L. Deori, A. Falsone, and M. Prandini, "A two-layer decentralized approach to the optimal energy management of a building district with a shared thermal storage," in *20th World Congress of the International Federation of Automatic Control (IFAC 2017), Toulouse, France*, July 2017.

[10] J. Cortes and F. Bullo, "Coordination and Geometric Optimization via Distributed Dynamical Systems," *SIAM Journal on Control and Optimization*, vol. 44, no. 5, pp. 1543–1574, 2005.

[11] J. Cortes and F. Bullo, "Nonsmooth Coordination and Geometric Optimization via Distributed Dynamical Systems," *SIAM Review*, vol. 51, no. 1, pp. 163–189, 2009.

[12] J. Wang and N. Elia, "A control perspective for centralized and distributed convex optimization," *IEEE Conference on Decision and Control*, vol. 21, no. 5, pp. 1–6, 2011.

[13] S. Kia, "A distributed dynamical solver for an optimal resource allocation problem over networked systems," *IEEE Conference on Decision and Control*, pp. 1–6, 2015.

[14] S. Kia, J. Cortes, and S. Martinez, "Dynamic average consensus under limited control authority and privacy requirements," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 13, pp. 1946–1966, 2015.

[15] A. Cherukuri and J. Cortes, "Initialization-free distributed coordination for economic dispatch under varying loads and generator commitment," *Automatica, under review*, pp. 1–15, 2015.

[16] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation: Numerical methods*. Athena Scientific (republished in 1997), 1989.

[17] A. Nedic, A. Ozdaglar, and P. Parrilo, "Constrained consensus and optimization in multi-agent networks," *IEEE Transactions on Automatic Control*, vol. 55, no. 4, pp. 922–938, 2010.

[18] M. Zhu and S. Martinez, "On distributed convex optimization under inequality and equality constraints," *IEEE Transactions on Automatic Control*, vol. 57, no. 1, pp. 151–164, 2012.

[19] J. Tsitsiklis, *Problems in decentralized decision making and computation*. Ph.D. Dissertation, MIT, Cambridge, MA, 1984.

[20] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, 1986.

[21] A. Olshevsky and J. Tsitsiklis, "Convergence speed in distributed convergence and averaging," *SIAM Review*, vol. 53, no. 4, pp. 747–772, 2011.

[22] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[23] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2506–2517, 2009.

[24] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, pp. 48–61, Jan 2009.

[25] S. Lee and A. Nedic, "Distributed random projection algorithm for convex optimization," *IEEE Journal on Selected Topics in Signal Processing*, vol. 7, no. 2, pp. 221–229, 2013.

[26] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[27] D. Bertsekas, "Multiplier methods: A survey," *Automatica*, vol. 12, no. 2, pp. 133–145, 1976.

[28] D. Bertsekas, "Incremental proximal methods for large scale convex optimization," *Mathematical Programming*, vol. 129, no. 2, pp. 163–195, 2011.

[29] S. Ram, A. Nedic, and V. Veeravalli, "A new class of distributed optimization algorithm: Application of regression of distributed data," *Optimization Methods & Software*, vol. 27, no. 1, pp. 71–88, 2012.

[30] A. Nedic and A. Ozdaglar, "Approximate primal solutions and rate analysis for dual subgradient methods," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1757–1780, 2009.

[31] T. Chang, A. Nedic, and A. Scaglione, "Distributed constrained optimization by consensus-based primal-dual perturbation method," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1524–1538, 2014.

[32] D. Zhao, "Cdma-based wireless cellular networks," in *Power Distribution and Performance Analysis for Wireless Communication Networks*, pp. 17–37, Springer, 2012.

[33] G. J. Foschini and Z. Miljanic, "A simple distributed autonomous power control algorithm and its convergence," *Vehicular Technology, IEEE Transactions on*, vol. 42, no. 4, pp. 641–646, 1993.

[34] R. D. Yates, "A framework for uplink power control in cellular radio systems," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 7, pp. 1341–1347, 1995.

[35] T. Alpcan, T. Başar, R. Srikant, and E. Altman, "Cdma uplink power control as a noncooperative game," *Wireless Networks*, vol. 8, no. 6, pp. 659–670, 2002.

[36] D. Famolari, N. Mandayam, D. Goodman, and V. Shah, "A new framework for power control in wireless data networks: Games, utility, and pricing," in *Wireless Multimedia Network Technologies*, pp. 289–309, Springer, 2002.

[37] P. Hande, S. Rangan, M. Chiang, and X. Wu, "Distributed uplink power control for optimal sir assignment in cellular data networks," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 6, pp. 1420–1433, 2008.

[38] J. Huang, R. A. Berry, and M. L. Honig, "Distributed interference compensation for wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 5, pp. 1074–1084, 2006.

[39] S. S. Ram, V. V. Veeravalli, and A. Nedić, "Distributed non-autonomous power control through distributed convex optimization," in *INFOCOM 2009, IEEE*, pp. 3001–3005, IEEE, 2009.

[40] S. Grandhi and J. Zander, "Constrained power control in cellular radio systems," in *Proceedings of IEEE Vehicular Technology Conference (VTC)*, pp. 824–828 vol.2, Jun 1994.

[41] K.-L. Hsiung, S.-J. Kim, and S. Boyd, "Power control in lognormal fading wireless channels with uptime probability specifications via robust geometric programming," in *Proceedings of the 2005, American Control Conference, 2005.*, pp. 3955–3959 vol. 6, June 2005.

[42] M. Chiang, "Geometric programming for communication systems," *Foundations and Trends® in Communications and Information Theory*, vol. 2, no. 1–2, pp. 1–154, 2005.

[43] R. Sinkhorn and P. Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, no. 2, pp. 343–348, 1967.

[44] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1." `http://cvxr.com/cvx`, Mar. 2014.

[45] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 28).*, 2015.

[46] M. Gordon and K. C. Ng, *Cool thermodynamics: The Engineering and Physics of Predictive, Diagnostic and Optimization Methods for Cooling Systems*. Cambridge International Science Publishing, 2000.

[47] F. Belluschi, A. Falsone, K. Margellos, S. Garatti, and M. Prandini, "Energy management for building district cooling: a distributed approach to resource sharing," *IEEE Transactions on Control Systems Technology*, 2016. Submitted. Online preprint arXiv:1610.06332.

[48] R. Vujanic, P. Mohajerin, P. Goulart, S. Mariethoz, and M. Morari, "A decomposition method for large scale MILPs, with performance guarantees and a power system application," *Automatica*, vol. 67, no. 5, pp. 144–156, 2016.

[49] A. Simonetto and H. Jamali-Rad, "Primal recovery from consensus-based dual decomposition for distributed convex optimization," *Journal of Optimization Theory and Applications*, vol. 168, no. 1, pp. 172–197, 2016.

[50] G. Calafiore and M. Campi, "The scenario approach to robust control design," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 742–753, 2006.

[51] M. Campi and S. Garatti, "The exact feasibility of randomized solutions of uncertain convex programs," *SIAM Journal on Optimization*, vol. 19, no. 3, pp. 1211–1230, 2008.

[52] M. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, vol. 33, no. 2, pp. 149 – 157, 2009.

[53] M. Campi and S. Garatti, "A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality," *Journal of Optimization Theory and Applications*, vol. 148, no. 2, pp. 257–280, 2011.

[54] S. Garatti and M. Campi, "Modulating Robustness in Control Design," *IEEE Control Systems*, vol. 33, no. 2, pp. 36 – 51, 2013.

[55] K. Margellos, M. Prandini, and J. Lygeros, "On the connection between compression learning and scenario based single-stage and cascading optimization problems," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2716–2721, 2015.

[56] M. Campi, S. Garatti, and F. Ramponi, "Non-convex scenario optimization with application to system identification," *IEEE Conference on Decision and Control, to appear*, pp. 1–8, 2015.

[57] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[58] G. Schildbach, L. Fagiano, and M. Morari, "Randomized Solutions to Convex Programs with Multiple Chance Constraints," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2479 – 2501, 2013.

[59] N. Kariotoglou, K. Margellos, and J. Lygeros, "On the computational complexity and generalization properties of multi-stage and stage-wise coupled scenario programs," *Systems and Control Letters*, vol. 94, pp. 63–69, 2016.

[60] R. Vujanic, P. M. Esfahani, P. J. Goulart, S. Mariéthoz, and M. Morari, "A decomposition method for large scale milps, with performance guarantees and a power system application," *Automatica*, vol. 67, pp. 144–156, 2016.

[61] D. Ioli, A. Falsone, S. Schuler, and M. Prandini, "A compositional framework for energy management of a smart grid: A scalable stochastic hybrid model for cooling of a district network," in *12th IEEE International Conference on Control and Automation (ICCA)*, pp. 389–394, 2016.

[62] D. Jia and B. Krogh, "Min-max feedback model predictive control for distributed control with communication," in *Proc. IEEE Amer. Contr. Conf.*, pp. 4507–4512, 2002.

[63] M. D. Doan, T. Keviczky, and B. De Schutter, "A distributed optimization-based approach for hierarchical MPC of large-scale systems with coupled dynamics and constraints," in *Proc. IEEE Conf. on Dec. and Contr. and Eur. Contr. Conf.*, pp. 5236 –5241, 2011.

[64] P. Trodden and A. Richards, "Distributed model predictive control of linear systems with persistent disturbances," *Int. Journal of Contr.*, vol. 83, no. 8, pp. 1653–1663, 2010.

[65] A. Venkat, I. Hiskens, J. Rawlings, and S. Wright, "Distributed MPC strategies with application to power system automatic generation control," *IEEE Trans. Contr. Sys. Tech.*, vol. 16, no. 6, pp. 1192–1206, 2008.

[66] P. Giselsson and A. Rantzer, "Distributed model predictive control with suboptimality and stability guarantees," in *Proc. IEEE Conf. on Dec. and Contr.*, pp. 7272–7277, 2010.

[67] D. Groß and O. Stursberg, *Distributed predictive control of communicating decentralized systems*, pp. 139–156. Springer, 2014.

[68] D. Groß and O. Stursberg, "Distributed predictive control for a class of hybrid systems with event-based communication," *IFAC Proceedings Volumes*, vol. 46, no. 27, pp. 383–388, 2013.

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

[70] D. Christmann, R. Gotzhein, and S. Rohr, "The arbitrating value transfer protocol (AVTP) - deterministic binary countdown in wireless multi-hop networks," in *Int. Conf. on Computer Communications and Networks*, pp. 1–9, 2012.

[71] M. Bussieck and A. Pruessner, "Mixed-integer nonlinear programming," *SIAG/OPT Newsletter: Views & News*, vol. 14, no. 1, pp. 19–22, 2003.

[72] J. Lee and S. Leyffer, *Mixed integer nonlinear programming*, vol. 154. Springer Science & Business Media, 2011.

[73] B. Lincoln and Rantzer, "Relaxing dynamic programming," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1249–1260, 2006.

[74] D. Gorges and S. Izák, Mand Liu, "Optimal control and scheduling of switched systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 1, pp. 135–140, 2011.

[75] W. Zhang, A. Abate, J. Hu, and M. Vitus, "Exponential stabilization of discrete-time switched linear systems," *Automatica*, vol. 45, no. 11, pp. 2526–2536, 2009.

[76] T. Westenbroek and H. Gonzalez, "Optimal control of hybrid systems using a feedback relaxed control formulation," *arXiv preprint arXiv:1510.09127*, 2015.

[77] P. Bonami and J. Lee, "Bonmin user's manual," *Numer Math*, vol. 4, pp. 1–32, 2007.

[78] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*, pp. 284–289, IEEE, 2005.

[79] D. Liberzon, *Switching in systems and control*. Springer Science & Business Media, 2012.

[80] O. Stursberg, "A graph search algorithm for optimal control of hybrid systems," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 2, pp. 1412–1417, IEEE, 2004.

[81] K. Åström and B. Wittenmark, *Computer-controlled systems: theory and design*. Courier Corporation, 2013.

[82] G. Seber and A. Lee, *Linear regression analysis*, vol. 936. John Wiley & Sons, 2012.

[83] R. Gondhalekar and J. Imura, "Least-restrictive move-blocking model predictive control," *Automatica*, vol. 46, no. 7, pp. 1234 – 1240, 2010.

[84] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Tr. on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[85] A. Richards, "Fast model predictive control with soft constraints," *European Journal of Control*, vol. 25, pp. 51 – 59, 2015.

[86] C. Kirches, L. Wirsching, H. Bock, and J. Schlöder, "Efficient direct multiple shooting for nonlinear model predictive control on long horizons," *J. of Process Control*, vol. 22, no. 3, pp. 540 – 550, 2012.

[87] L. Blackmore and B. Williams, "Optimal manipulator path planning with obstacles using disjunctive programming," in *American Control Conference*, pp. 3200–3202, 2006.

[88] H. Ding, G. Reissig, and O. Stursberg, "Increasing Efficiency of Optimization-based Path Planning for Robotic Manipulators," in 50^{th} *IEEE Conf. on Decision and Control*, pp. 1399–1404, 2011.

[89] P. Tøndel, T. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, vol. 39, no. 3, pp. 489 – 497, 2003.

[90] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *J. of Robotics Research*, vol. 5, pp. 90–98, 1986.

[91] W. Newman and M. Branicky, "Real-time configuration space transforms for obstacle avoidance," *J. of Robotics Research*, vol. 6, pp. 650–667, 1991.

[92] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *J. of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[93] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994.

[94] R. J. Caron, J. F. McDonald, and C. M. Ponic, "A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary," *Journal of Optimization Theory and Applications*, vol. 62, no. 2, pp. 225–237, 1989.

[95] D. Ioli, A. Falsone, A. V. Papadopoulos, and M. Prandini, "A compositional modeling framework for the optimal energy management of a district network," *International Journal of Process Control*, 2016. Submitted.

[96] J. Eilbrecht and O. Stursberg, "Auction-based cooperation of autonomous vehicles using mixed-integer planning," in *Proc. of the AAET Automatisiertes & Vernetztes Fahren*, pp. 267–286, ITS automotive nord e.V., 2017.

[97] D. Heß, C. Löper, and T. Hesse, "Safe cooperation of automated vehicles," in *Proc. of the AAET Automatisiertes & Vernetztes Fahren*, pp. 309–334, ITS automotive nord e.V., 2017.