Performance Optimization and Stability Guarantees for Multi-tier Real-Time Control Systems

Yehan Ma
Shanghai Jiao Tong University
yehanma@sjtu.edu.cn

Ruijie Fu
Shanghai Jiao Tong University
furj2022@sjtu.edu.cn

An Zou
Shanghai Jiao Tong University
an.zou@sjtu.edu.cn

Jing Li
New Jersey Institute of Technology
jingli@njit.edu

Cailian Chen
Shanghai Jiao Tong University
cailianchen@sjtu.edu.cn

Chenyang Lu
Washington University in St. Louis
lu@wustl.edu

Xinping Guan[§]
Shanghai Jiao Tong University
xpguan@sjtu.edu.cn

***Abstract*—Modern control systems are embracing multi-tier architectures integrating end devices and edge servers. However, due to the distinct control performance demands associated with each control task, it is a formidable challenge to optimize the control performance of multiple control tasks subject to stringent computation resource constraints while guaranteeing stability. Moreover, inherent contradictions exist in the timing aspect between the stability guarantee, which relies on offline analysis, and the run-time control performance, which should be enhanced online. It is essential to bridge the gap between the real-time scheduling of control tasks and their actual control performance. In this paper, we propose a novel real-time scheduling approach for multi-tier control systems, which leverages end devices for executing real-time control tasks and edge devices for run-time coordination. Specifically, we first introduce a new data-driven value function, called *time/state/utility functions (TSUF)*, for modeling control system performance. *TSUF* captures not only timing but also the dynamic states of the physical plants. Subsequently, we propose *value-based control scheduling (VCS)*, which is a multi-granularity scheduling mechanism based on our *TSUF* value function. *VCS* distinguishes the scheduling of *stability jobs* for ensuring system stability and *performance jobs* for optimizing real-time control performance based on run-time physical states. Finally, through realistic case studies involving multiple control loops, we demonstrate the advantages of *VCS* over existing scheduling approaches in terms of both control and real-time performance.**

***Index Terms*—Cyber-physical systems, real-time control, value-based control scheduling, edge computing**

I. Introduction

Modern control systems are evolving into multi-tier systems comprising edge servers and end devices [1]. As shown in Fig. 1, the end layer of industrial control systems comprises a large number of sensing terminals (e.g., thermometers and accelerometers), local controllers (e.g., PLCs and microcontrollers), and actuation terminals (e.g., rolling mills, and valves) to control multiple sub-systems (i.e., control loops), while the the edge layer, which is usually equipped with higher computation capacity, can play roles in monitoring, coordination, and human-computer interaction. Control tasks are usually executed on the local controllers located near physical plants with reliable and wired connections to sensors and actuators. However, as modern control systems continue to integrate an increasing number of control loops along with data-intensive tasks, significant challenges emerge.

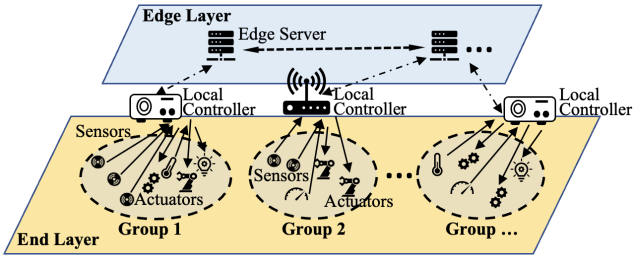


Fig. 1: Multi-tier control system comprising end and edge tiers.

In practice, when control tasks share the same platform [2]–[4], all in the pursuit of meeting stringent real-time requirements, the capacity of local computation platforms is limited [2], [5], thereby restricting their ability to accommodate a multitude of control tasks, especially computation-intensive ones. The gap between the real-time scheduling performance and the control performance further exacerbates this limitation. The gap arises from the intricate relationships among actual control performance, computational execution, and the states of the physical plants. Our preliminary findings (refer to Sec. VI-B for details of the experimental settings), as shown in Fig. 2, reveal the impact of physical state evolution and latency on the practical control performance, i.e., mean absolute error (MAE). The changes of the current physical plant states L_1 and L_2 and the end-to-end delay of the feedback control loop exhibit diverse effects on the control performance over the prediction horizon. Therefore, the value (utility) of control

[§]Corresponding author

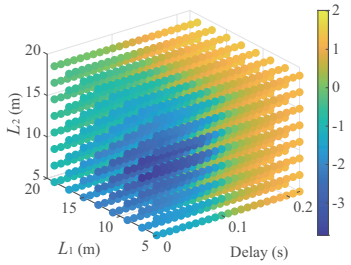


Fig. 2: Varying control performance over the prediction horizon of 10 s, i.e., MAE, impacted by different cyber-physical states, i.e., changing physical states L_1 and L_2, and end-to-end time delay. The colormaps is MAE at log-scale.

task execution dynamically changes according to the varying physical states and the end-to-end delay as plants evolve.

The value(utility)-based task management has been widely explored in the past decades, which mostly focuses on the real-time constraints [6], [7] and some recent work handles the efficiency [8], [9] and reliability demands [10], [11]. However, unlike traditional utility models, the utility for real-time control systems should incorporate control performance, which depends on online cyber-physical states (i.e., the status of cyber and physical components in the control systems) and latency. However, the analytical modeling of control performance over cyber-physical states is challenging and subject to strong assumptions, even for scalar systems [12], let alone the practical control performance, such as MAE, maximum absolute error (MaxAE), and settling time. Deriving practical performance analytically is infeasible if the system order is higher than two [13]. Furthermore, natural conflicts exist between control task scheduling for stability and run-time control performance. Stability is typically ensured by offline designs and analyses. In contrast, dynamic control performance varies at run-time, reflecting the transient processes of the plant control. Hence, dynamic control performance should be improved by online scheduling adjustments. However, coordinating the conflicting elements in scheduling and control across multiple time scales remains a complex and open challenge.

In this work, we propose a multi-tier real-time control approach for performance optimization and stability guarantee. Our approach starts from defining the *time/state/utility functions (TSUF)* that reflects the relationship between the control performance and cyber-physical states. To address the challenge that deriving closed-form TSUF analytically is often infeasible for practical control performance, we employ data-driven modeling on the edge server. Moreover, we propose an innovative *value-based control scheduling (VCS)*, which distinguishes the scheduling of *stability jobs* for system stability and *performance jobs* for run-time control performance optimization based on run-time value prediction of TSUF. VCS enhances control performance at run-time with guaranteed stability for multi-loop control systems. The contributions are four-fold.

- We introduce a novel two-tier real-time control framework that reaps the strengths of end devices for real-time control task execution and the capabilities of edge devices for run-time coordination to improve performance.
- We develop the time/state/utility functions (TSUF) with data-driven modeling methods to formulate the relationship between practical control performance and evolving cyber-physical states within real-time control systems.
- We propose value-based control scheduling (VCS), a multi-granularity real-time scheduling mechanism that dynamically schedules multiple control tasks to enhance the overall control performance while ensuring system stability.
- We demonstrate the advantages of the proposed approach through several realistic case studies involving 12 control loops based on the *Real-Time Control System Simulator (RTCS)*. Results demonstrate the benefits of our approach both in terms of control metrics and scheduling metrics.

The remainder of the article is organized as follows. Sec. II reviews the related work. Sec. III presents an overview of two-tier real-time control system architecture. Sec. IV introduces TSUF and the data-driven approaches to estimate the function. Sec. V presents the VCS mechanism, which integrates the scheduling of stability and performance jobs. Sec. VI describes the case studies in the context of 12-loop industrial control. Sec. VII concludes this paper.

II. Related Work

Real-time control systems often have stringent timing requirements since commands for the plants should be actuated timely on the sensor measurements as plants evolve. In recent years, the co-design of control and scheduling for real-time control systems gains its popularity. The first category is offline scheduling to guarantee static control performance. Maggio et al. [14] and Truong et al. [15] analyze the stability property of closed-loop systems that include a controller that can sporadically miss deadlines or lose packets. Hobbs et al. [16] and Pazzaglia et al. [17] consider the safety properties and linear quadratic regulators, respectively, subject to weakly-hard constraints. Saifullah et al. [18] determine sampling rates for mesh networks to optimize control performance in terms of linear quadratic metrics. Dai et al. [19] co-design sampling periods and poles of controllers by minimizing the settling time while satisfying the schedulability constraint for time-sensitive networks. The second category is run-time scheduling without considering physical states. Wu et al. [20] propose composite resource scheduling in real-time networked control systems with a strict execution order of sensing, computing, and actuating segments, which guarantees the resource utilization of computation and network less than 100%. Lee et al. [21] propose a task model that captures the number of control update misses and a dynamic-priority scheduling mechanism. Yoshimoto et al. [2] propose an optimal arbitration of control tasks by job skipping so that all accepted jobs meet their deadlines under the overloading situations. However, the value of control task execution is varying as the physical state evolves. Moreover, above approaches cannot adapt to run-time

physical disturbances well since they do not consider run-time physical states.

The third category is run-time scheduling to guarantee online control performance. Cervin et al. [22] adapt the period of control tasks so that the overall control costs of the finite-horizon linear quadratic regulator (LQR) for linear physical plants are minimized on a single-CPU platform. Dai et al. [23] adapt the period of the control task at run-time based on historical measurements, considering the task set with one control task and multiple non-control tasks. Chwa et al. [4] prioritizes tasks by stability and state error with weakly-hard real-time constraints. However, all the above studies ignore the specific relationships among latency, physical states, and both run-time and static control performance. Moreover, their run-time scheduling strategies do not differentiate the scheduling mechanism of static and run-time performance, which results in unavoidably pessimistic resource utilization.

To measure the productivity of a computing platform, a number of studies in real-time society focus on the importance of task/job execution, which is usually characterized by the time/utility function (TUF) [24]. TUF determines the utility resulting from the execution of a job as a function of its completion time. The commonly used patterns of the TUF in the computing domain are monotonic TUF [6], [25], [26], which monotonically decreases until zero, and threshold-based TUF [7], [8], which is constant within the hard time threshold and monotonically decreases from the soft time threshold to the hard time threshold. Various value-based heuristics have been explored for TUF [8], [9], [11], [27]. Kumbhare et al. [8] take the system-wide power constraint into account and design a series of power-aware value-based algorithms to improve productivity and resource utilization. Chen et al. [9] present an optimal algorithm to maximize the quality of service (QoS) represented by value for a DVFS-enabled nonvolatile processor under resource constraints. Considering the reliability demands, Shelton [27] designed the fault-sensitive utility based on the criticality of system failure. Accordingly, Emberson et al. [11] used this utility as part of the objective function to find the fault-tolerance task allocations. However, for real-time scheduling of control applications, which have distinct concerns of overall closed-loop control performance, such as MAE and system stability, the TUF cannot represent their unique metrics of control performance well.

The impact of computation and communication on real-time control performance has been explored by analytical methods, where control performance is modeled as functions of network latency, rate, and reliability. However, the theoretical analyses are challenging and subject to strong assumptions, even for scalar systems [12], [13]. In addition, the studies are limited to observability [28], stability [29], and LQR [30] so that existing theoretical tools, such as the Popov-Belevitch-Hautus test, Lyapunov equation, and Riccati equation etc., can be utilized. To overcome the limitations of analytical approaches, data-driven techniques provide new tools for establishing models of real-time control. Machine learning models have been applied to controller design [31], safety verification [32], and decision making [33]. In this work, we exploit data-driven TSUF, which has not been addressed in the aforementioned exploration of data-driven models.

Edge computing is gaining rapid adoption among multiple applications with low-latency requirements, such as health monitoring [34], video analytics [35], and autonomous driving [36]. However, few edge computing frameworks have been applied to real-time control systems thus far because of their unique concerns [37] of overall closed-loop control performance. Skarin et al. design and implement a control system on local/edge/cloud platforms [38], and compare the impact of the different platforms on MPC [39]. Ma et al. propose an edge offloading approach to exploit edge computing in multi-loop control systems under varying network conditions [40]. However, all the above work assumes that the computation platform is a black box and ignores the potential benefits introduced by edge-enabled computation task scheduling. In this work, we explore an end-edge coordination methodology that leverages the capabilities of end devices for real-time control tasks and that of edge devices for run-time coordination. The overall control performance of multiple loops is optimized while guaranteeing stability.

III. System Architecture

We present the system architecture of our approach in this section and describe our proposed techniques including TSUF and VCS in detail in the following sections.

A. Multi-tier Real-Time Control Architecture

We introduce an edge-enabled real-time control system, as illustrated in Fig. 3. This system employs a two-tier architecture that operates at different timescales to effectively manage the control task execution of multiple control loops.

- The edge coordination operates at *large timescales*. At the *offline* stage, the edge server establishes the time/state/utility function (TSUF) based on data-driven approaches (①) and analyses the stability conditions of all control loops (②). During each run-time *coordination period*, the edge server assesses the impact (i.e., value) of each control task in terms of control performance based on TSUF and run-time cyber-physical states. It then assigns *priorities* to each control task based on *value-based control scheduling* mechanism (③). The priorities are used for the scheduling of control tasks on the local controller in the subsequent coordination period.
- At *small timescales*, the system relies on the capabilities of the local controller for real-time scheduling and reliable execution of control tasks (④). The local controller is interconnected with sensors and actuators across multiple physical plants through dependable wired connections. It concurrently executes real-time control policies (i.e., tasks) for multiple physical plants with their own *sampling/control periods*.

This two-tier architecture effectively manages control tasks and optimizes control performance without sacrificing stability for the real-time control system.

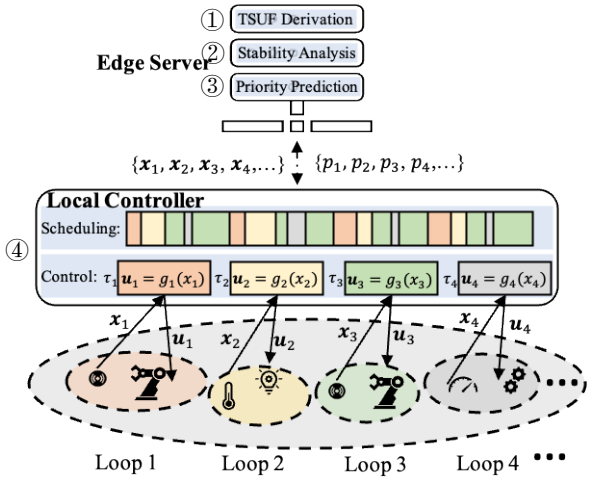


Fig. 3: Two-tier architecture for real-time control systems.

B. Control and Scheduling on Local Controller

1) Control System Modeling: The following equations show the modeling of a control loop i considered in this work.

$$\begin{aligned} \boldsymbol{x}_i(k+1) &= f_i(\boldsymbol{x}_i(k), \boldsymbol{u}_i(k), \boldsymbol{\theta}_i) \\ \boldsymbol{u}_i(k) &= g_i(\boldsymbol{x}_i(k)) \end{aligned} \tag{1}$$

where $i \in \{1, 2, ..., M\}$ is the control loop (i.e., control task) index, k is the time index of control periods T_i, M is the number of plants controlled by the local controller, $\boldsymbol{x}_i(k) \in \mathbb{R}^n$ is the physical state vector, $\boldsymbol{u}_i(k) \in \mathbb{R}^m$ is the control command vector, $\boldsymbol{\theta}_i$ is the configurable model parameter vector, $f_i(\cdot)$ is the dynamic and kinematic model of the physical plant, and $g_i(\cdot)$ is the control policy. The equations reflect that the physical state $\boldsymbol{x}_i(k)$ evolves based on $f_i(\cdot)$ of the physical plants and the control commands $\boldsymbol{u}_i(k)$ derived by executing the control policies $g_i(\cdot)$.

As in Fig. 3, the local controller is connected to sensors and actuators of multiple physical plants via reliable wired connections [41], [42]. The physical states $\boldsymbol{x}_i(k)$ of the plants are collected from sensors and sent to the local controller periodically. The local controller operates control policies $\boldsymbol{u}_i(k) = g_i(\boldsymbol{x}_i(k))$ based on $\boldsymbol{x}_i(k)$ in each period T_i, and generates actuation commands $\boldsymbol{u}_i(k)$ for actuators to actuate the physical plants. In this work, we assume all control loops are independent of each other in terms of physical states and control policies [19].

2) Real-Time Scheduling: The local controller executes real-time control tasks of all physical plants connected to it. It executes control tasks τ_i *periodically* [37] with period T_i and executed with the priority that the edge server decides for each coordination period. Control tasks have implicit deadlines, where the deadlines are equal to their period, $D_i = T_i$ [43]. We assume the actual execution time of τ_i is consistent with the worst-case execution time (WCET) and known accurately [44], [45]. The M control tasks share the processors on the local controller. Focusing on the mainstream multicore processors, the local controller applies preemptive global priority-based scheduling. In this work, tailored for the overall control performance for multi-loop real-time control systems, we present a multi-granularity real-time scheduling mechanism, which dynamically schedules multiple control tasks based on run-time cyber-physical states to enhance performance while ensuring system stability, which is detailed in Sec. V.

C. Coordination on Edge Server

As shown in Fig. 3, the coordination on the edge server is divided into three parts. The first part is the offline TSUF derivation tailored for real-time control systems. The edge server applies data-driven approaches to derive TSUF, which defines the relationship between the importance of each control task on the overall control performance given the current cyber-physical states. The data-driven approaches overcome the challenges of modeling the coupling relationship analytically. The importance of the control task can be interpreted by its impact on the practical control performance metrics, such as MAE, MaxAE, and settling time. The cyber-physical states could be but not limited to end-to-end latency, control period, and states of the physical plants. We discuss TSUF in more detail in Sec. IV-B1.

The second part is the offline stability analysis and guarantee. We deliberately classify the jobs of control task τ_i into two categories: *stability jobs* for guaranteeing stability and *performance jobs* for improving the run-time performance. We convert the stability guarantee of the multi-loop control system to the schedulability guarantee for stability jobs on a multi-core platform, which is detailed in Sec. V-A.

The third part is the run-time task priority prediction for performance jobs based on TSUF and cyber-physical states. In each coordination period, the edge server leverages the TSUF to evaluate the values (i.e., utilities) of executing control jobs based on run-time cyber and physical states and give higher priorities to jobs that benefit control performance more, which is introduced in Sec. IV-B2.

D. Control Performance

It is critical to ensure the *stability* of the closed-loop system, which means that the physical states are bounded, for the sake of safety. To be more specific, every bounded initial physical state approaches a bounded state as time approaches infinity. Therefore, stability is determined by mathematical models of the control system. Stability analysis can be performed offline before the control system is implemented if the mathematical models are fixed [14], [46].

In addition, we take the following three practical control metrics, i.e., MAE, MaxAE, and settling time, as examples

- MAE $= \frac{1}{T_h/T_i+1} \sum_{k=0}^{T_h/T_i} |\boldsymbol{x}_e(k)|$,
- MaxAE $= \max(\{|\boldsymbol{x}_e(0)|, |\boldsymbol{x}_e(1)|, ..., |\boldsymbol{x}_e(T_h/T_i)|\})$,
- Settling time: the time required by the response to reach and stay within the range of 1% of its steady value.

where $\boldsymbol{x}_e(k)$ is the error between the actual physical states and their reference values, and T_h is the horizon of performance evaluation. The above practical performance represents the run-time information of the transition process from the current

physical states under the actuation commands. However, the analytical expressions for these run-time performance indicators cannot be found if the system order is larger than two [13], which is infeasible for complex industrial control systems.

IV. The Data-driven Time/State/Utility Functions

This section presents our TSUF tailored for real-time control systems. To overcome the challenges for deriving practical control performance, we develop a *data-driven* approach on the edge server to derive TSUF and predict the value of control tasks based on run-time cyber-physical states.

A. Time/state/utility Functions and Value of Control Tasks

As detailed in Sec. II, traditional value-based scheduling does not consider control performance. In contrast, we incorporate the value of executing real-time control tasks into the formulation and design TSUF. As shown in Fig. 2, the value of executing a certain real-time control task varies at run-time, not only depending on the time delay but also on the state evolution of the physical plants as time passes.

In this section, we define the comprehensive cyber-physical states targeted by our designs. Controlling multiple physical plants involves managing various static properties, denoted as $\boldsymbol{s}_i$, which tend to remain constant throughout the operation of real-time control systems. These static properties include the control period T_i, model parameters $\boldsymbol{\theta}_i$, and control policies $g_i(\cdot)$, which are defined in Sec. III-B, among others. Additionally, the control process is featured by dynamic cyber and physical states, including latency t_i^d and physical states $\boldsymbol{x}_i$. The latency t_i^d with superscript d indicating the delay is defined as the end-to-end latency between the time the physical states $\boldsymbol{x}_i(k)$ are measured and the time the control commands $\boldsymbol{u}_i(k)$ for states $\boldsymbol{x}_i(k)$ are actuated on the physical plants. For instance, at time $t = kT_i$, the physical state $\boldsymbol{x}_i(k)$ is measured. It takes t_i^d to transmit $\boldsymbol{x}_i(k)$ to the local controller, execute the control task to derive control commands $\boldsymbol{u}_i(k)$, and transmit $\boldsymbol{u}_i(k)$ to the actuator of the physical plant i. At time $t = kT_i + t_i^d$, $\boldsymbol{u}_i(k)$ is actuated. As the value of executing control tasks depends on both dynamic and static states, we define time/state/utility function (TSUF) as

$$V_i = \mathcal{F}_i(t_i^d, \boldsymbol{x}_i, \boldsymbol{s}_i). \tag{2}$$

Here, V_i indicates the control performance (provided in Sec. III-D) prediction of τ_i given the dynamic and static states t_i^d, $\boldsymbol{x}_i$, and $\boldsymbol{s}_i$.

Since physical states $\boldsymbol{x}_i$ of the physical plants evolve at run-time, TSUF reflects the run-time control performance. Practical control performance metrics described in Sec. III-D can be used in the utility. Furthermore, the ***overall control performance*** of M control tasks can be calculated by

$$V = \sum_{i=1}^{M} w_i V_i = \sum_{i=1}^{M} w_i \mathcal{F}_i(t_i^d, \boldsymbol{x}_i, \boldsymbol{s}_i), \tag{3}$$

where w_i indicates the specified importance of loop i [19]. In the evaluation, we set $w_i = 1, \forall i \in \{1, 2, ..., M\}$.

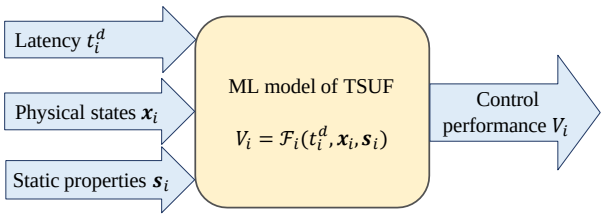


Fig. 4: ML model of TSUF.

B. Data-Driven Derivation and Prediction

Deriving run-time control performance over cyber-physical states analytically is challenging and subject to strong assumptions, even for scalar systems [12], which is infeasible for practical performance metrics if the system order is higher than two [13]. Therefore, we propose to adopt well-established machine learning (ML) models to construct the TSUF for real-time control systems, as shown in Fig. 4. The inputs of the model are latency t_i^d, physical states $\boldsymbol{x}_i$, and static properties $\boldsymbol{s}_i$. The outputs are the utility in terms of the control performance of running the control tasks.

1) Offline TSUF Derivation: Instead of leaving the utility formulation up to the application developer as in existing works, we use the offline training of the ML model in Fig. 4 to learn the relationship between V_i and cyber-physical states and derive the TSUF $\mathcal{F}_i(\cdot)$. Given the data of various input-output combinations, we construct and select the ML models for TSUF. Note that the data-driven approach does *not* require any analytical models, which could be extremely complicated to be known or derived explicitly.

Regarding data collection, we conduct control system simulations by sampling the input values over their feasible ranges. In Sec. IV-C, we adopt several well-established ML methods and evaluate their accuracy by a case study for a better illustration. However, note that the TSUF derivation is not limited to those listed ML methods. Here, we consider offline model training and TSUF derivation. Online refinement and transfer learning can be applied to TSUF model training as well, which is not the focus of this work. We will work on refining the TSUF in our future work.

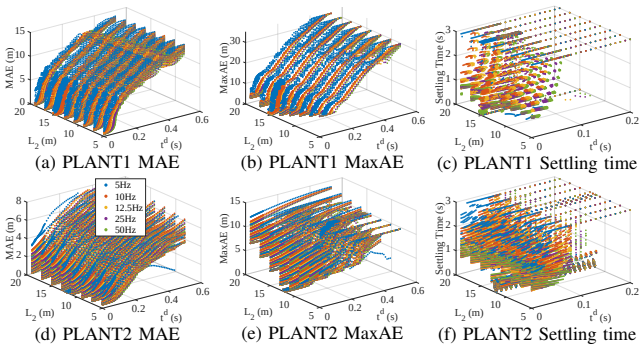
2) Run-time Value Prediction: The ML model inference provides the online value prediction for TSUF in the process of real-time control, which predicts the utilities $V_i(k)$ of executing control tasks over $[kT_i, kT_i + T_h]$ based on run-time cyber-physical conditions $t_i^d(k)$, $\boldsymbol{x}_i(k)$, and $\boldsymbol{s_i}$.

$$V_i(k) = \mathcal{F}_i(t_i^d(k), \boldsymbol{x}_i(k), \boldsymbol{s}_i) \tag{4}$$

The predicted values are used to advise the real-time scheduling of control tasks based at run-time.

C. Case Study

Firstly, we run simulations for two different control systems PLANT1 and PLANT2 with different $f(\cdot)$ and $g(\cdot)$ to collect the control performance under different cyber-physical states. The detailed experimental settings are the same as in Sec. VI, and we omit the subscript i here for brevity. For training, we



(a) PLANT1 MAE (b) PLANT1 MaxAE (c) PLANT1 Settling time

(d) PLANT2 MAE (e) PLANT2 MaxAE (f) PLANT2 Settling time

Fig. 5: Practical control performance, i.e., MAE, MaxAE, and settling time, under varying cyber-physical states, i.e., t^d and one of the physical states L_2.

generate traces with the length of T_h in each simulation round. To train a data-driven TSUF model to predict the control performance over a prediction horizon of T_h, we use t^d, $\boldsymbol{x}(0) = [L_1, L_2]^\top$, and T as inputs to predict MAE, MaxAE, and settling time over $[0, T_h]$ during training. T_h depends on time constants [13] (5-10 times the time constant) for the physical plants in order to properly represent both transient and steady performance. At the run-time, we invoke TSUF periodically for performance prediction. For each invocation at time kT_i, we use t^d, $\boldsymbol{x}(k)$, and T as inputs to predict certain control performance over $[kT_i, kT_i + T_h]$.

We collect data from simulating control loops of two different physical plants PLANT1 and PLANT2 (refer to Sec. VI-B for details of the experimental settings) covering the sets of input variable values specified in Table I. We sample the data point near the equilibrium points, $(8.44, 15)$ for PLANT1 and $(6.75, 12)$ for PLANT2, in smaller granularities since the systems work at them most of the time. Therefore, for each plant, $92 \times 64 \times 5 \times 2 = 58880$ simulations are executed, and 58880 $\left(\boldsymbol{x}_i(0), V_i([0, T_h])\right)$ data pairs are collected. Fig. 5 shows data collection examples when $T_h = 3$ s. The x and y axes are partial TSUF inputs. Each data point indicates one round of simulation for 3 s. The corresponding control metrics for each input, i.e., MAE, MaxAE, and settling time, over T_h under different control periods T_i are indicated by the value of z axes. We can see that when t^d is smaller and L_2 is closer to equilibrium points, the control performance is better.

TABLE I: The range and granularity of each input variable used in the simulations for data collection.

	Min	***Max***	***Granularity***
t^d	0	500 ms	10 ms
	0	80 ms	2 ms
L_1 of PLANT1	5 m	19 m	2 m
	8.1 m	8.8 m	0.1 m
L_2 of PLANT1	5 m	19 m	2 m
	14.7 m	15.4 m	0.1 m
L_1 of PLANT2	5 m	19 m	2 m
	6.4 m	7.1 m	0.1 m
L_2 of PLANT2	5 m	19 m	2 m
	11.7 m	12.4 m	0.1 m
T	$\{20, 40, 80, 100, 200\}$ ms		

TABLE II: Testing accuracy of ML models.

	Normalized Error (unit:m,m,s)					
	PLANT1			**PLANT2**		
	MAE	MaxAE	Settling	MAE	MaxAE	Settling
Polynomial	0.244	0.196	0.132	0.181	0.189	0.148
SVM	0.139	0.115	0.081	0.186	0.517	0.082
LSBoost	0.072	0.076	0.047	0.054	0.066	0.061
NN	0.105	0.086	0.047	0.074	0.069	0.068

Next, we use the following four well-established ML models to train and predict the TSUF utilities.

- Polynomial regression model with all terms up to the power of 5 for all input variables;
- Support vector machine (SVM) regression model, we use the Gaussian kernel function, box constraint and kernel scale that minimize five-fold cross-validation loss;
- LSBoost, we train an ensemble of boosted regression trees using least-squares boosting;
- Neural network (NN) with a hidden layer size of 50 (since accuracy improvement is limited when the hidden layer size exceeds 50 based on experimental observation).

Among 58880 data tuples, we randomly select 37600 tuples as training set and the remaining 21280 as testing set. We compare the regression models on testing datasets. The normalized errors for different control metrics are shown in Table II, where the absolute errors are normalized by actual control performance. We can see that LSBoost and NN have comparable accuracy, while LSBoost models are slightly better.

Since ML models are trained offline, the training time differences between models do not affect their deployment. In comparison, the control performance prediction based on the ML model is online. Thus, we measure the execution time of LSBoost and NN prediction 5000 times in MATLAB on an edge server (8 Intel Core i9-14900K CPUs@3.2 GHz, 128 GB RAM). The (mean, median, and standard deviation) tuples of prediction time (for one loop) with NN and LSBoost are (2.3, 2.2, 0.39) ms and (2.0, 2.0, 0.20) ms, respectively. The time cost of LSBoost is slightly shorter than that of NN.

V. Value-based Real-Time Scheduling for Control Performance with Stability Guarantee

This section presents our real-time scheduling that seamlessly achieves dynamic control performance optimization based on run-time states and system stability guarantee. The scheduling mechanism integrates three parts: (1) establishing the *stability jobs* for guaranteeing stability, (2) deriving run-time value of the *performance jobs* for improving the overall control performance, and (3) a *real-time scheduling approach* for coordinating stability and performance jobs.

The jobs $J_{i,q}$ of the periodic control task τ_i (defined in Sec. III-B2) are classified into two categories: stability jobs $J^s_{i,j}$ and performance jobs $J^p_{i,r}$, as shown in Fig. 6, where q, j, r are corresponding job indexes. The stability jobs $J^s_{i,j}$ are jobs of control tasks that guarantee the stability of the control system, and performance jobs $J^p_{i,r}$ are used to improve

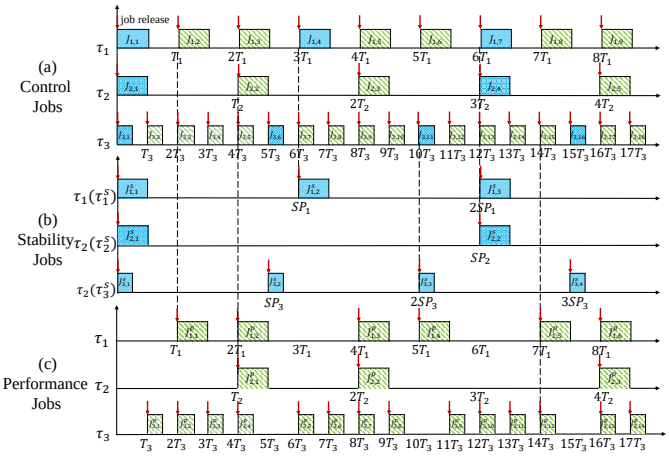


Fig. 6: The job release patterns of value-based control scheduling. The jobs $J_{i,q}$ of control task τ_i in Sub-fig. (a) are classified into stability jobs $J^s_{i,j}$ in (b) and performance jobs $J^p_{i,r}$ in (c).

the practical control performance at run-time by enhancing the control/actuation granularity subjected to limited computational resources. We present the determination and scheduling properties of $J^s_{i,j}$ and $J^p_{i,r}$ in the rest of this section.

A. Determination of Stability Jobs for System Stability

The core of VCS is to improve dynamic control performance based on TSUF and run-time physical states under a stability guarantee. Therefore, we establish the minimum computation resources that guarantee system stability such that we can enhance the resource for dynamic control performance. The stability of control loops can be characterized by the *stability period* SP_i, which is the maximum control period that guarantees the stability of control loop i. SP_i depends on the models of the physical plants and types of controllers employed. These controllers encompass state-feedback controllers [14], [46], linear quadratic regulators [47], model predictive controllers [48], and sub-optimal controllers [49], [50]. The stability period SP_i is multiples of control period T_i, which indicates that, facing consecutive misses/losses of control, the system will become unstable. For instance, Maggio et al. investigated the stability conditions of the linear time invariant (LTI) system by a weakly hard task model [14], which derives that the stability can be guaranteed when the control task τ_i experiences a bounded number of n_i consecutive deadline misses, where $n_i \in \mathbb{N}$. Based on this, we can compute SP_i and establish the stability job $J^s_{i,j}$ from the jobs of control task τ_i with the period of T_i. SP_i is derived by the maximum deadline missing periods with stability guarantee, i.e., $SP_i = (n_i + 1)T_i$.

To ensure the stability of the closed-loop system, we designate the first job within each stability period SP_i ***of task*** τ_i ***as the stability jobs, as shown in Fig. 6b, i.e.,*** $J^s_{i,j} = J_{i,(j-1)\frac{SP_i}{T_i}+1}, j \in \mathbb{N}^+$. Stability jobs form *stability task* τ^s_i with period of SP_i and constrained deadline of T_i. Furthermore, we map the stability guarantee of the control system to a constrained deadline scheduling problem, which is detailed and proved in Prop. V.1.

Proposition V.1. *The stability of the control system with control period T_i and stability period SP_i can be guaranteed by the schedulability of stability task set τ^s_i with the period of SP_i and the constrained deadline of T_i, $\forall i \in \{1, 2, 3, ..., M\}$.*

Proof. If the stability can be guaranteed when the control task τ_i experiences a bounded number of n_i consecutive deadline misses, i.e., $SP_i = (n_i+1)T_i$, we designate the first job within each SP_i of task τ_i as stability job, $J^s_{i,j} = J_{i,(j-1)\frac{SP_i}{T_i}+1}, j \in \mathbb{N}^+$. Stability jobs form stability task τ^s_i with period of SP_i and constrained deadline of T_i. As a result, if all τ^s_i is completed within its deadline T_i, the maximum number of consecutive uncontrolled periods can be easily bounded to n_i. Therefore, the stabilizing problem is converted to a hard real-time scheduling problem for τ^s_i with its period SP_i and constrained deadline T_i, and the stability of the control system can be guaranteed by ensuring the schedulability of τ^s_i. □

B. Run-time Priority Prediction for Performance Jobs

The rest of the jobs of τ_i ***except stability jobs*** $J^s_{i,j}$ ***are defined as performance jobs*** $J^p_{i,r}$***, as shown in Fig. 6c.*** As complementary to the stability jobs, the performance jobs are the jobs between consecutive stability jobs to enhance the control performance. Since the performance of a controller has a monotonically decreasing relationship in regards to the control period [18], the plant will gain performance as more performance job are released. The priorities of performance jobs are derived at run-time based on TSUF derived in Sec. IV and the run-time cyber-physical states. The priority indicates the value of executing a certain performance job, which can be interpreted by the impact of a performance job on the overall control performance. To better describe the value of executing the performance job, we utilize *control value density* $\delta_i(k')$, which scales the control performance (utility) gain of a performance job by its execution time and task frequency, deriving the utility gain per scheduling unit.

$$\delta_i(k') = \frac{\mathcal{F}_i(0, \boldsymbol{x}_i(k'), \boldsymbol{s}_i) - \mathcal{F}_i(\Delta, \boldsymbol{x}_i(k'), \boldsymbol{s}_i)}{C_i/T_i}, \quad (5)$$

where $\mathcal{F}_i(\cdot)$ is the TSUF in Eq. (2), C_i the worst-case execution time for τ_i, k' the time index for the priority prediction. And Δ the end-to-end latency interval for density prediction.

Since the scheduling strategy based on value density determines the actual end-to-end latency, δ_i and Δ are coupled. In order to decouple scheduling and value density prediction to derive $\delta_i(k')$ for scheduling, we apply the same time interval Δ for all control loops for fairness and simplicity. In this way, δ_i indicates the sensitivity of executing $J^p_{i,r}$ to latency given run-time physical states $\boldsymbol{x}_i(k')$. Moreover, any physical disturbances or noises that can be captured by the variance of $\boldsymbol{x}_i(k')$ are reflected by $\delta_i(k')$, indicating the resiliency of physical disturbances of our scheduling strategy.

The online performance $V_i = \mathcal{F}_i(\cdot)$ for all control loops is predicated on the edge server based on Eq. (4) in Sec. IV-B2,

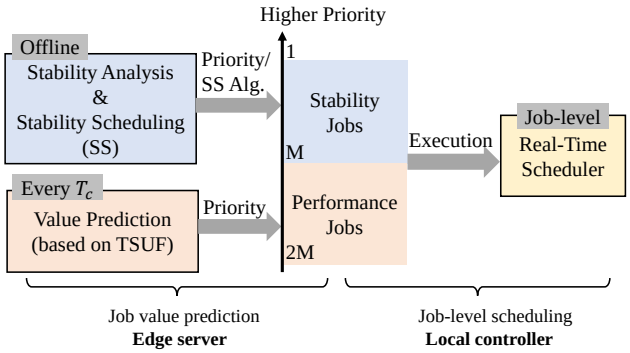


Fig. 7: Value-based control scheduling mechanism. (Integration of control jobs with multi-granularity coordination of offline stability analysis, run-time value prediction, and real-time scheduling by job classification and priority assignments)

$\boldsymbol{x}_i(k')$, and the static cyber-physical settings $\boldsymbol{s}_i$, such as control period T_i and plant model parameters $\boldsymbol{\theta}_i$, in each coordination period of T_c. Then the edge server predicts the control value density $\delta_i(k')$ over $[k'T_c, (k'+1)T_c]$ according to Eq. (5). The weighted (w_i, which is consistent with Eq. (3)) ranking of $\delta_i(k')$ are interpreted as the priority $p_i(k')$ of performance jobs $J^p_{i,r}$ released during $[k'T_c, (k'+1)T_c]$. We define the priority set for M control loops as $P(k') = [p_1(k'), p_2(k'), ..., p_M(k')]$. The explicit priority level assignment for performance jobs is discussed in Sec. V-C in order to cooperate with stability jobs.

The local controller operates priority-based real-time scheduling based on $P(k')$ provided by the edge server in the following T_c. Note that the priorities for $J^p_{i,r}$ are adapted based on TSUF and run-time cyber-physical states with a large time granularity of T_c. The priorities remain the same as in the previous priority set $P(k'-1)$ until they receive the update.

C. Value-based Control Scheduling Mechanism

To summarize, as shown in Fig. 7, on the local controller, VCS coordinates stability and performance jobs. The schedulability of stability jobs guarantees the stability of the control system. Stability job scheduling is designed offline to satisfy the deadline constraints. Performance jobs are scheduled dynamically based on priorities provided by TSUF, which aim to optimize the control performance.

The jobs of control tasks are scheduled by a priority-based scheduler. Since the stability jobs $J^s_{i,j}$ should meet their hard deadlines to guarantee stability, the M highest priorities (i.e., priority range of $[1, M]$) are assigned to the stability jobs of M control tasks τ_i with $i \in \{1, 2, 3, ..., M\}$. For the performance job $J^p_{i,r}$, the edge server provides the priorities $p_i(k')$ for performance jobs of τ_i sorted by the online prediction of control value densities $\delta_i(k')$ in each coordination period T_c. The priority levels of $p_i(k')$ are within the range of $[M+1, 2M]$. As all the stability jobs $J^s_{i,j}$ always have a higher priority than the performance jobs $J^p_{i,r}$, they take precedence over $J^p_{i,r}$. The execution of $J^p_{i,r}$ will not impact (i.e., interrupt or block) that of $J^s_{i,j}$. If all stability jobs are schedulable, i.e., the stability task set τ^s_i, $\forall i \in \{1, 2, 3, ..., M\}$, passes the schedulability test, given certain scheduling approaches, the control systems are stable, as we proved in Sec. V-A. The performance jobs $J^p_{i,r}$ leverage the remaining computation resources to enhance the performance without interfering with the stability and schedulability of the stability jobs.

The assignment of each priority (from 1 to M) to one stability job is not limited to any priority assignment policy. Thus, deadline monotonic (DM), optimal priority assignment [51], or dynamic priority assignments, such as earliest deadline first (EDF), can be applied. The only prerequisite is that the stability task set can pass the schedulability tests mentioned in Sec. V-A. For example, if the scheduling of stability tasks follows DM, the corresponding schedulability test could be response time analysis [52]. In addition, if the scheduling of stability tasks follows EDF, the corresponding schedulability test could be processor-demand analysis [53].

VI. Case Study and Evaluation

In this section, we present an evaluation of the real-time scheduling approach and the functionalities of TSUF in multi-loop real-time control systems scenarios. To accomplish this, we have devised a real-time control system simulator (RTCS) employing MATLAB/Simulink. This simulator facilitates real-time scheduling on multi-core processors and allows for real-time control of numerous physical plants. Our investigation encompasses performance assessments of the proposed approach across 12 heterogeneous control loops.

A. Real-Time Control System Simulator (RTCS)

To evaluate systematically on various cases of real-time control systems, we developed a real-time control system simulator (RTCS), as depicted in Fig. 8. The RTCS includes the simulation of multiple physical plants, real-time scheduling of multiple control tasks on the local controller with multicore processors, and coordination on the edge server. As the RTCS simulates the physical plants and computing processors on the fly, it can accurately reflect the impact of real-time scheduling on run-time physical states and control performance.

In RTCS, we consider a practical control system setup [54], [55] with (1) clock-driven sensors that sample the plant outputs periodically every T_i, (2) event-driven controllers which calculate the actuation commands as soon as the sensor data arrives, and (3) event-driven actuators. This setup enables actuators to respond to updated actuation commands immediately. Sensors sample periodically every T_i. The event-driven local controller discovers the sensing updates and releases the control task τ_i, which is scheduled on the local controller together with other control tasks by the real-time scheduler. Once τ_i is finished, the control command for the actuator is generated and sent to the actuator i. The actuator updates the actuation once it discovers an update of control commands. We incorporate multi-rate simulation by setting the frequencies of edge server $1/T_c$ Hz, sensor sampling frequency $1/T_i$ Hz, and those of the multi-processor local controller and physical plants to 1000 Hz for

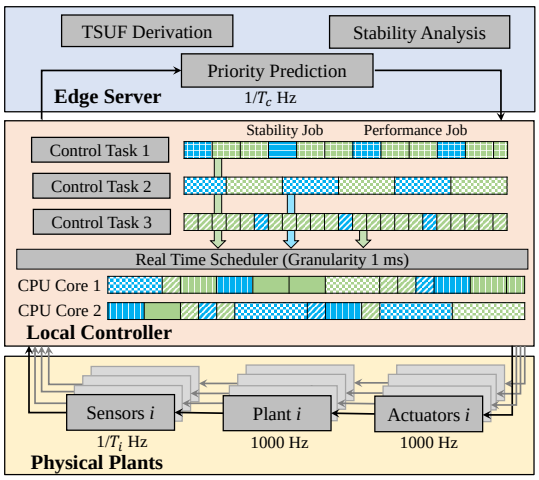


Fig. 8: Architecture of RTCS.

TABLE III: System parameters $\boldsymbol{\theta}_i$.

PLANT1				PLANT2			
par	value	par	value	par	value	par	value
A_1	0.01	R_1	0.0006	A_1	0.12	R_1	0.0006
A_2	0.006	R_2	0.0008	A_2	0.007	R_2	0.0008
A_R	1	α	10	A_R	1	α	10

simulation of the high-granularity real-time scheduling and the continuous nature of the physical plants.

B. Experimental Setups

1) Physical Plants: Consider 12 independent 3-state double water-tank systems, each of which is modeled as [56]

$$\begin{aligned}\dot{L}_1 &= \frac{1}{\rho A_1}(\alpha u - \frac{\sqrt{\rho g}}{\rho R_1}\sqrt{L_1})\\ \dot{L}_2 &= \frac{1}{\rho A_2}(\frac{\sqrt{\rho g}}{\rho R_1}\sqrt{L_1} - \frac{\sqrt{\rho g}}{\rho R_2}\sqrt{L_2})\\ \dot{L}_R &= \frac{1}{\rho A_R}(\frac{\sqrt{\rho g}}{\rho R_2}\sqrt{L_2} - \alpha u)\end{aligned} \tag{6}$$

where L_1, L_2, L_R are the liquid levels of the upper tank, lower tank, and the basin, respectively; A_1, A_2, A_R are the cross-sectional areas of the tanks; and R_1, R_2 are the resistance parameters of pipes of upper and lower tanks. We discretize the continuous-time model (6) using the Euler method.

There are two types of plants, denoted by PLANT1 and PLANT2, that have different system parameters $\boldsymbol{\theta}_i$, shown in Table III. Control loops 1, 3, 5, 7, 9, and 11 are of type PLANT1, and loops 2, 4, 6, 8, 10, and 12 are of type PLANT2. For 12 loops, we design state feedback controllers for reference tracking. We apply the performance metrics in Sec. IV-B1 in order to illustrate the intuition and advantages of the proposed value-based control scheduling. We evaluate the scheduling strategy under pulse physical disturbances. We add pulse physical disturbance to the plant of loops 3 and 7 at $t = 5$ s, loops 6 and 12 at $t = 6$ s, and loops 2 and 8 at $t = 9$ s.

2) Control Task and Computation Platform: In order to show the generality of real-time scheduling and the RTCS simulator, in this work, we decouple the control policies and the computation resources. We apply fixed feedback control policies for all control loops, and we synthesize the computation resources for each control task in RTCS. If control tasks miss their deadlines, they are regarded as outdated for current physical states and are terminated to save the computation resources for control tasks based on new state measurements. In our case studies, the computing platform incorporates two identical CPU cores. Each control loop is featured by its control period, worse-case execution time on the processor. We evaluate four different cases with different execution times in order to explore the impact of workloads. The control task settings of Case 1 are in Table IV. Case 1 represents an overload scenario with the utilization rate of 1.4500. The WCET for control tasks of Case 2 to Case 4 is set to 5 ms, 10 ms, and 13 ms, respectively, while the utilization rates are 0.4875, 0.9759, and 1.2675, respectively.

TABLE IV: Experimental settings for control tasks of Case 1.

Loop	Period	WCET	Loop	Period	WCET
1	50ms	10ms	2	80ms	20ms
3	50ms	15ms	4	100ms	30ms
5	50ms	10ms	6	80ms	20ms
7	40ms	10ms	8	80ms	15ms
9	50ms	10ms	10	100ms	15ms
11	50ms	15ms	12	80ms	25ms

Meanwhile, on the edge server, we set $T_h = 3$ s, which is around 10 times of time constants of both loops in this case study, and default $T_c = 0.25$ s. We discuss the effects of T_c in Sec. VI-D. The (mean, median, standard deviation) tuple of TSUF computation time for 12 loops is (24.23, 24.17, 0.47) ms with the same edge server settings as in Sec. IV-C. We assume the latency (overhead) of priority prediction is 30 ms, since a round-trip communication delay of less than 5 ms over km-level Ethernet based on experimental measurements. As the local controller adopts priority-based scheduling, the overhead of priority-based scheduling is $10 \times \mu s$ on the multi-core platform [57], which is negligible compared with WCET.

Remark VI.1. *In this work, since we focus on scheduling computational control tasks, we assume the latency of value prediction on the edge is constant. The latency is shorter than one control period in our case study. Please note that this overhead does not affect the overhead of the priority-based scheduling on the local controller since the performance scheduling sticks to the previous priorities until it receives the updated priorities. Besides, as discussed in Fig. 13, moderate priority update interval has limited impact on overall control performance. However, if the control systems work at faster control rates or the end-edge communication is based on low-rate network protocols, the latency of value prediction would have larger impact, which we will study in the future.* □

3) Comparison Groups: We compare the control performance and resultant control task scheduling using different state-of-the-art approaches.

- The proposed value-based control scheduling (VCS).

- The state-aware scheduling (SAS) [4]: the critical jobs for stability are scheduled by DM. Non-critical jobs that pass online schedulability test ordered by physical state errors are executed and scheduled by DM.
- The dynamic optimal period assignment (DPA) [22]: Task periods are adapted every T_c to minimize the overall control costs. All tasks are scheduled under rate monotonic. Since the cost function of DPA is designed for the linear continuous-time systems, we apply TSUF instead.
- The static optimal period assignment (PA) [18], [19]: PA selects the optimal sampling periods which minimize the overall mean absolute state error (MAE) offline. All tasks are scheduled under rate monotonic scheduling.
- Earliest deadline first (EDF),
- Rate monotonic (RM),
- Traditional value-based scheduling (TVB) [6], [25]: The value is monotonically decreasing until zero at deadline.

Stepping into state-of-the-art control-scheduling co-design approaches, we incorporate offline scheduling targeting safety constraints into stability scheduling in VCS. Besides, we compare VCS against offline scheduling (PA) that optimizes control performance, online scheduling and utility-based scheduling without considering physical states (e.g., EDF, RM, and TVB), and online scheduling (SAS and DPA) to improve control performance. Please note that the default scheduling approach of the stability tasks τ_i^s in VCS applies DM scheduling. We discuss different scheduling approaches of the stability tasks in Sec. VI-D. For fixed priority scheduling of SAS, DPA, and PA, we apply response time analysis [52] to establish the schedulability. The data-driven TSUF applied here is LSBoost model since it has higher accuracy and shorter execution time, as indicated in Sec. IV-C.

C. Stability Analysis and Stability Scheduling

As illustrated in Sec. V-A, the stability period SP_i can be determined by the maximum number (n_i) of consecutive deadline misses between two success control tasks. We derive n_i that satisfies the joint spectral radius condition [14], and $SP_i = (n_i + 1)T_i$. The resultant SP_i of each control loop is shown in Table V.

TABLE V: Stability period of loops.

Loop	SP_i	Loop	SP_i	Loop	SP_i	Loop	SP_i
1	150ms	2	320ms	3	150ms	4	300ms
5	150ms	6	320ms	7	160ms	8	320ms
9	150ms	10	300ms	11	150ms	12	320ms

Here we apply the DM and EDF real-time scheduling on these stability tasks and adopts the response time analysis [52] for DM and processor-demand analysis [53] to verify the schedulability of τ_i^s, respectively. Under the DM and EDF real-time scheduling, all the stability tasks can pass the schedulability test, which further indicates that the stability of the control systems can be guaranteed.

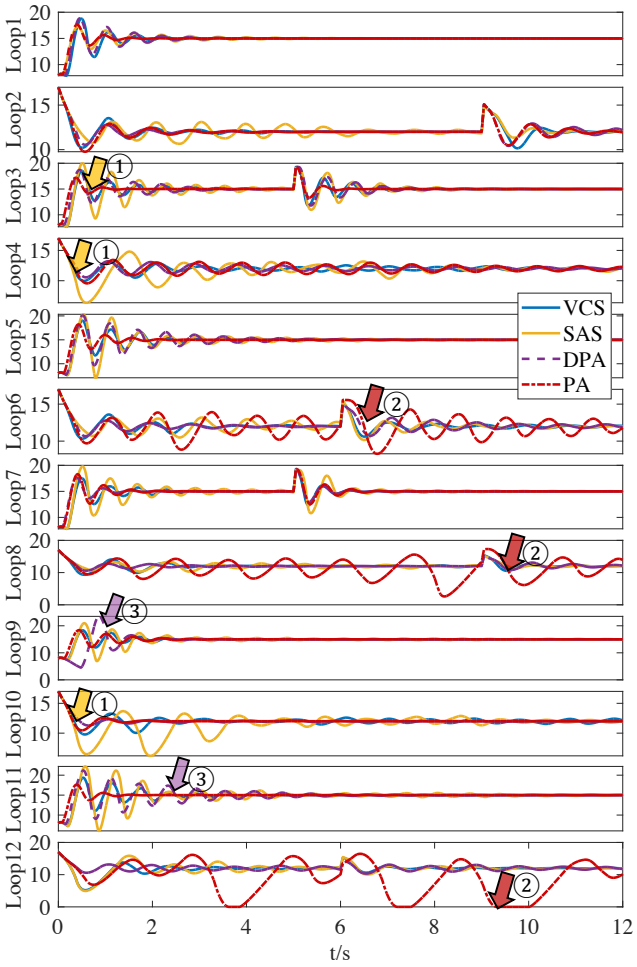


Fig. 9: Response curves of L_2 (m) for twelve loops under different scheduling approaches.

D. Control Performance

We evaluate the control performance under different scheduling approaches. First, we illustrate how VCS works using control response curves of 12-loop control system described in Sec. VI-B1. Then, we evaluate the statistical results of the overall control performance. Moreover, we discuss the impact of scheduling mechanisms for stability tasks and the impact of T_c on control performance.

1) Response Curve Analysis: We take Case 1 in Table IV as an instance. The response curves of L_2 for all 12 control loops are shown in Fig. 9. We can see that the control loops of PLANT1 (i.e., loops 1, 3, 5, 7, 9, 11) are featured by relatively shorter time constants than the control loops of PLANT2 (i.e., loops 2, 4, 6, 8, 10, 12) since PLANT1 has smaller sectional areas as indicated in Table III. This is also the reason why the control rates for PLANT1 are higher than those of PLANT2, as indicated in Table IV. The control performance metric of TSUF in VCS here is MAE. VCS performs much better than control-independent scheduling approaches. Therefore, we do not show the response curves of EDF, RM, and TVB. We discuss their statistic results

in Sec. VI-D2. The control performance of the proposed VCS shows its superiority in all loops for all time, which demonstrates the efficiency of TSUF and our scheduling strategy. SAS performs worse than VCS near equilibrium points (reference values) since the performance metric of state error cannot reflect comprehensive cyber-physical states of the control loop (shown in ①). Besides, its requirement of schedulability for all admitted tasks is pessimistic in resource utilization. Although PA selects the optimal control periods offline, it performs the worst since it cannot adapt to the run-time conditions and physical disturbance (shown in ②). DPA improves the control performance compared to PA since it adapts the periods based on TSUF. However, guaranteeing schedulability results in prolonged control periods, leading to poorer control performance sometimes (shown in ③).

In addition to the overhead measurements of VCS in Sec. VI-B2, we measure the overheads of SAS and DPA 5000 times on the edge server. The (mean, median, standard deviation) tuples of overheads of SAS, and DPA are (0.2, 0.1, 1.5) ms and (1577.2, 1576.6, 24.1) ms, respectively. The overhead of SAS is shorter than VCS since it makes decisions based on state error instead of data-driven TSUF. The overhead of DPA is much longer than VCS since it calculates the control performance metrics and schedulability tests by traversing all candidate periods. For fair comparison, we assume DPA runs on a powerful machine and shares the same end-to-end latency of 30 ms as VCS for controlled experiments.

2) Statistical Results of Control Performance: Furthermore, we study the statistical results of control performance under different scheduling approaches. Each boxplot in Figs. 10 to 12 is derived by 36 rounds of simulation under various initial physical states. Each sample of the boxplot is the overall MAEs for 12 loops over the simulation interval of 12 s. We compare the control performance MAE with different scheduling mechanisms listed above. When the workloads for the local controller are large, VCS can effectively improve the overall control performance (sum of MAE of all loops) to a large extent compared with other scheduling approaches. The missing boxplot of RM in Fig. 10d indicates unstable control loops with poor control performance that we do not record in the figure. When the workload is moderate, VCS can also improve the control performance by shortening the latency for critical control loops.

To verify the efficacy of TSUF with different control metrics besides MAE, we evaluate the performance when the control metrics in Eq. (3) are settling time and MaxAE during training TSUF, as shown in Fig. 11. The same happens to settling time, as shown in Fig. 11a. MaxAEs reflect the worst-case state error in each response curve. The improvement in MaxAE is limited, as shown in Fig. 11b, since the MaxAEs mainly happen at the first transient process, which depends more on the initial physical states than latency. In addition, we look into the response curves of all simulation rounds for VCS, SAS, DPA, PA, EDF, RM, and TVB. The numbers of unstable (nonconvergent) curves are 0, 0, 0, 2, 10, 7, and 8 among 36-round simulations in Case 1, respectively. Similarly, for Case 4, they are 0, 0, 0, 1, 4, 9, and 5. VCS is stable throughout the simulations performed, which is consistent with the theoretical analysis in Sec. VI-C.

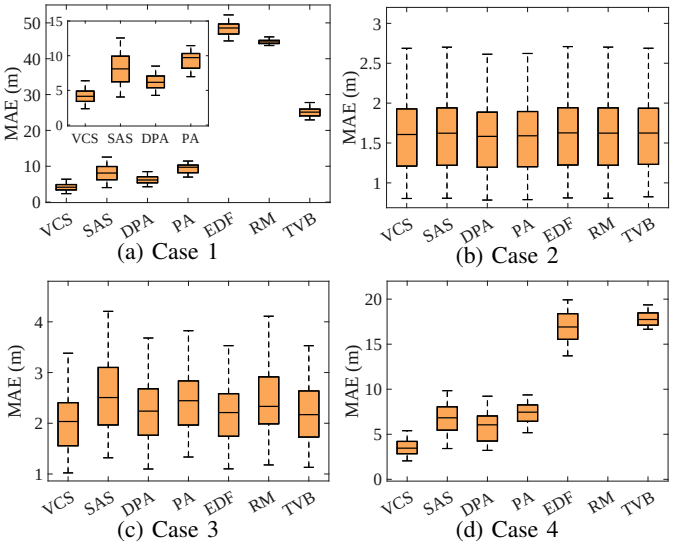


Fig. 10: MAEs of VCS and baselines.

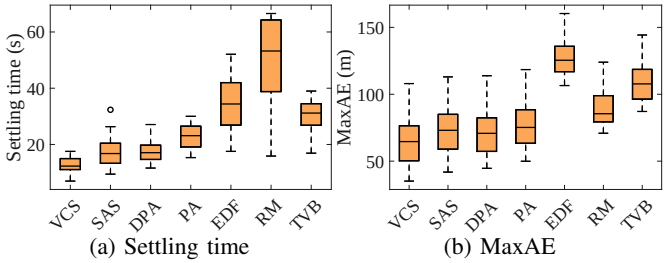


Fig. 11: Comparisons of other performance (Case 1).

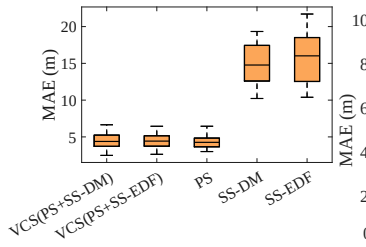


Fig. 12: Impact of τ_i^s scheduling (Case 1).

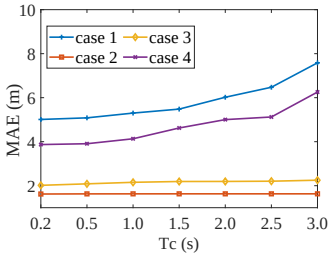


Fig. 13: Impact of T_c.

3) Impact of τ_i^s scheduling and T_c: As shown in Fig. 12, we separate the performance jobs and stability jobs to analyse their functionalities and compare their impact on the overall performance of VCS.

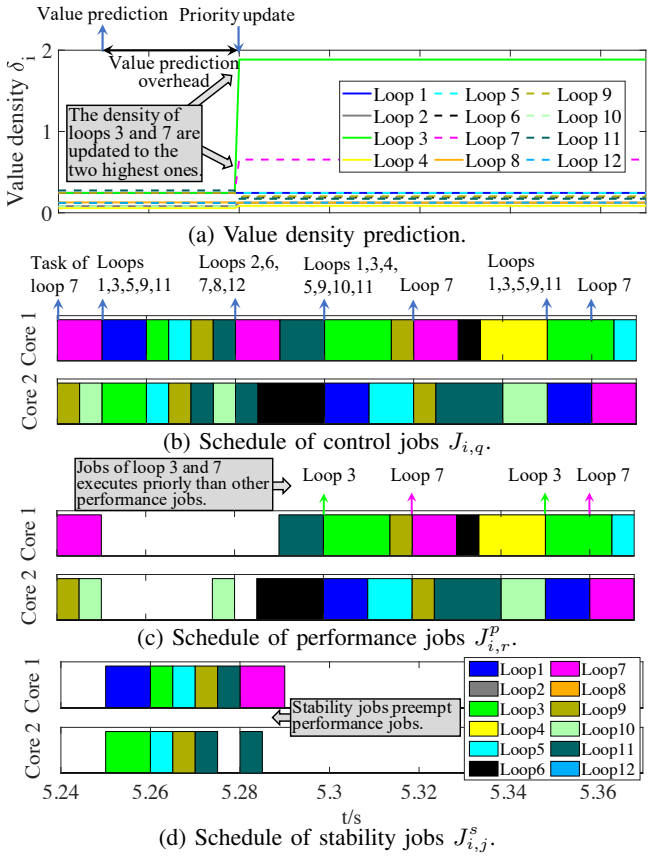
- Value-based control scheduling with only performance scheduling, named as VCS(PS): all the control tasks are scheduled based on value predictions by TSUF, i.e., let $J_{i,r}^p = J_{i,r}$.
- Pure stability job scheduling (SS): only schedule τ_i^s with the period of SP_i and deadline of T_i. And SS-DM

and SS-EDF are the different scheduling mechanisms for stability jobs, i.e., DM and EDF.

- VCS(PS+SS-DM) and VCS(PS+SS-EDF): mean value-based control scheduling with both performance and stability jobs, and stability scheduling mechanisms are DM and EDF, respectively.

We can see that although SS-DM and SS-EDF guarantee the stability of the control systems, the MAEs are inferior. On the other hand, all the VCS mechanisms perform quite well, which demonstrate the importance of improving run-time control performance besides guaranteeing the stability.

We analyze the impact of T_c on the overall control performance, as shown in Fig. 13. As T_c increases, the control performance deteriorates, especially for the cases of heavy workload, since the value-based control scheduling cannot react to frequently changing physical conditions in time.



(a) Value density prediction.

(b) Schedule of control jobs $J_{i,q}$.

(c) Schedule of performance jobs $J^p_{i,r}$.

(d) Schedule of stability jobs $J^s_{i,j}$.

Fig. 14: A zoom-in view of the resultant schedule (Case 1) during the time interval of $[5.24s, 5.37s]$. Sub-fig. (a) shows the resultant δ_i based on TSUF and physical states, and (b-d) shows the resultant schedules of $J_{i,q}$, $J^p_{i,r}$, and $J^s_{i,j}$.

E. Scheduling Mechanisms Analysis

A zoom-in view of the resultant schedule for Case 1 during the time interval of $[5.24s, 5.37s]$ is shown in Fig. 14 as an illustrative instance. We differentiate the resultant schedules of performance jobs and the stability jobs, as shown in Figs. 14c and 14d. We can see that the performance jobs are preempted by the stability jobs since they have higher priorities. In addition, control tasks for loops 3 and 7 are mostly executed once they are released since they have higher value density, as indicated in Fig. 14a. This can be explained by the fact that loops 3 and 7 are under physical disturbances around $t = 5$ s.

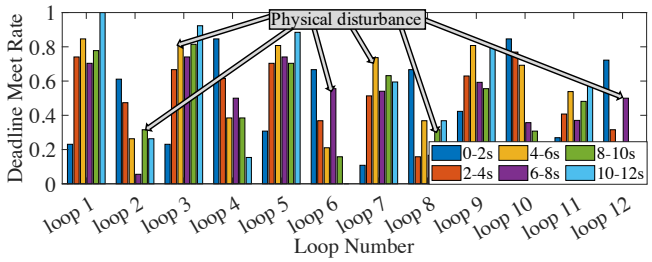


Fig. 15: Acceptance ratios of control tasks for performance jobs over different time intervals (Case 1). Each arrow points to the interval when a physical disturbance occurred to each control loop.

We evaluate the deadline meet rates (acceptance ratios) of performance and stability jobs for each control loop over different time intervals (Case 1). It is not surprising that all stability jobs meet their deadlines, i.e., deadline meet rates are all 1, since they are proven to be schedulable in Sec. VI-C. Looking into the deadline meet rates for performance jobs in Fig. 15, the ratios drop obviously because of limited computation resources. It is worth noting that the deadline meet rates of loops 2, 3, 6, 7, 8, and 12 obviously increase during the time intervals of 8-10 s, 4-6 s, 6-8 s, 4-8 s, 8-10 s, and 6-8 s, respectively, which are the exact time intervals with the physical disturbances. Therefore, VCS can adapt well to the physical conditions of each control loop.

VII. Conclusion

In this paper, we present a two-tier real-time control system with performance optimization and stability guarantee. We propose TSUF tailored for real-time control systems, which indicates the coupling relationship among control performance, physical states, and latency. And we develop a data-driven approach to derive TSUF in terms of control performance of MAE, settling time, and MaxAE. We establish value-based control scheduling mechanisms that can improve the control performance by dynamically updating the schedule according to TSUF and run-time physical states while guaranteeing stability. Through extensive evaluation, we demonstrate the effectiveness of performance predictions based on TSUF. Five case studies with 12 control loops validate the significant improvement of the overall control performance by our two-tier real-time control solution.

VIII. Acknowledgement

This work is supported in part by the NSF of China under Grants 62103268, 92167205, 62202287, 62473254, 61933009, 62025305, and Shanghai Chenguang Program 21CGA11.

REFERENCES

[1] G. Nain, K. Pattanaik, and G. Sharma, "Towards edge computing in intelligent manufacturing: Past, present and future," *Journal of Manufacturing Systems*, vol. 62, pp. 588–611, 2022.

[2] T. Yoshimoto and T. Ushio, "Optimal arbitration of control tasks by job skipping in cyber-physical systems," in *IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2011.

[3] M. Hosseinzadeh, B. Sinopoli, I. Kolmanovsky, and S. Baruah, "Rotec: Robust to early termination command governor for systems with limited computing capacity," *Systems & Control Letters*, vol. 161, p. 105142, 2022.

[4] H. S. Chwa, K. G. Shin, and J. Lee, "Closing the gap between stability and schedulability: a new task model for cyber-physical systems," in *RTAS*. IEEE, 2018, pp. 327–337.

[5] H. Choi, Y. Xiang, and H. Kim, "Picas: New design of priority-driven chain-aware scheduling for ros2," in *2021 IEEE 27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2021, pp. 251–263.

[6] S. Li, M. Song, P.-J. Wan, and S. Ren, "A 2-approximation algorithm for scheduling parallel and time-sensitive applications to maximize total accrued utility value," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 1864–1878, 2015.

[7] B. Khemka, R. Friese, L. D. Briceno, H. J. Siegel, A. A. Maciejewski, G. A. Koenig, C. Groer, G. Okonski, M. M. Hilton, R. Rambharos *et al.*, "Utility functions and resource management in an oversubscribed heterogeneous computing environment," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2394–2407, 2014.

[8] N. Kumbhare, A. Marathe, A. Akoglu, H. J. Siegel, G. Abdulla, and S. Hariri, "A value-oriented job scheduling approach for power-constrained and oversubscribed hpc systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1419–1433, 2020.

[9] W.-M. Chen, T.-S. Cheng, P.-C. Hsiu, and T.-W. Kuo, "Value-based task scheduling for nonvolatile processor-based embedded devices," in *IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2016, pp. 247–256.

[10] C. P. Shelton, P. Koopman, and W. Nace, "A framework for scalable analysis and design of system-wide graceful degradation in distributed embedded systems," in *Proceedings of the Eighth International Workshop on Object-Oriented Real-Time Dependable Systems, 2003.(WORDS 2003)*. IEEE, 2003, pp. 156–163.

[11] P. Emberson and I. Bate, "Extending a task allocation algorithm for graceful degradation of real-time distributed embedded systems," in *2008 Real-Time Systems Symposium*. IEEE, 2008, pp. 270–279.

[12] W. Liu, G. Nair, Y. Li, D. Nesic, B. Vucetic, and H. V. Poor, "On the latency, rate and reliability tradeoff in wireless networked control systems for IIoT," *IEEE Internet of Things J.*, 2020.

[13] N. S. Nise, *Control systems engineering*. John Wiley & Sons, 2020.

[14] M. Maggio, A. Hamann, E. Mayer-John, and D. Ziegenbein, "Control-system stability under consecutive deadline misses constraints," in *32nd euromicro conference on real-time systems (ECRTS)*, 2020.

[15] T. H. Truong, P. Seiler, and L. E. Linderman, "Analysis of networked structural control with packet loss," *IEEE Transactions on Control Systems Technology (TCST)*, vol. 30, no. 1, pp. 344–351, 2021.

[16] C. Hobbs, B. Ghosh, S. Xu, P. S. Duggirala, and S. Chakraborty, "Safety analysis of embedded controllers under implementation platform timing uncertainties," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2022.

[17] P. Pazzaglia, L. Pannocchi, A. Biondi, and M. Di Natale, "Beyond the weakly hard model: Measuring the performance cost of deadline misses," in *30th Euromicro Conference on Real-Time Systems (ECRTS)*, 2018.

[18] A. Saifullah, C. Wu, P. B. Tiwari, Y. Xu, Y. Fu, C. Lu, and Y. Chen, "Near optimal rate selection for wireless control systems," *ACM Trans. on Embedded Computing Systems*, vol. 13, no. 4s, p. 128, 2014.

[19] X. Dai, S. Zhao, Y. Jiang, X. Jiao, X. S. Hu, and W. Chang, "Fixed-priority scheduling and controller co-design for time-sensitive networks," in *International Conference on Computer-Aided Design (ICCAD)*, 2020.

[20] P. Wu, C. Fu, T. Wang, M. Li, Y. Zhao, C. J. Xue, and S. Han, "Composite resource scheduling for networked control systems," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 162–175.

[21] J. Lee and K. G. Shin, "Development and use of a new task model for cyber-physical systems: A real-time scheduling perspective," *Journal of Systems and Software*, vol. 126, pp. 45–56, 2017.

[22] A. Cervin, M. Velasco, P. Martí, and A. Camacho, "Optimal online sampling period assignment: Theory and experiments," *IEEE Transactions on Control Systems Technology*, vol. 19, no. 4, pp. 902–910, 2010.

[23] X. Dai and A. Burns, "Period adaptation of real-time control tasks with fixed-priority scheduling in cyber-physical systems," *Journal of Systems Architecture*, vol. 103, p. 101691, 2020.

[24] M. Kargahi and A. Movaghar, "Performance optimization based on analytical modeling in a real-time system with constrained time/utility functions," *IEEE Transactions on Computers*, vol. 60, no. 8, 2010.

[25] U. Balli, H. Wu, B. Ravindran, J. S. Anderson, and E. D. Jensen, "Utility accrual real-time scheduling under variable cost functions," *IEEE Transactions on Computers*, vol. 56, no. 3, pp. 385–401, 2007.

[26] G. Buttazzo, M. Spuri, and F. Sensini, "Value vs. deadline scheduling in overload conditions," in *Proceedings 16th IEEE Real-Time Systems Symposium*. IEEE, 1995, pp. 90–99.

[27] C. Shelton and P. Koopman, "Using architectural properties to model and measure graceful degradation," in *Architecting dependable systems*. Springer, 2003, pp. 267–289.

[28] G. Pang, W. Liu, Y. Li, and B. Vucetic, "Drl-based resource allocation in remote state estimation," *IEEE Transactions on Wireless Communications*, 2022.

[29] F. Mager, D. Baumann, C. Herrmann, S. Trimpe, and M. Zimmerling, "Scaling beyond bandwidth limitations: Wireless control with stability guarantees under overload," *ACM Transactions on Cyber-Physical Systems (TCPS)*, vol. 6, no. 3, pp. 1–30, 2022.

[30] D. Maity, M. H. Mamduhi, S. Hirche, and K. H. Johansson, "Optimal lqg control of networked systems under traffic-correlated delay and dropout," *IEEE Control Systems Letters*, vol. 6, pp. 1280–1285, 2021.

[31] I. Mitsioni, P. Tajvar, D. Kragic, J. Tumova, and C. Pek, "Safe data-driven model predictive control of systems with complex dynamics," *IEEE Transactions on Robotics*, 2023.

[32] M. Hibbard, M. Cubuktepe, M. Shubert, K. Lang, U. Topcu, and S. Phillips, "Trajectory synthesis for the coordinated inspection of a spacecraft with safety guarantees," *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 12, pp. 2245–2264, 2023.

[33] H. Liao, Y. He, X. Wu, Z. Wu, and R. Bausys, "Reimagining multi-criterion decision making by data-driven methods based on machine learning: A literature review," *Information Fusion*, p. 101970, 2023.

[34] M. Chen, W. Li, Y. Hao, Y. Qian, and I. Humar, "Edge cognitive computing based smart healthcare system," *Future Generation Computer Systems*, vol. 86, pp. 403–411, 2018.

[35] C.-C. Hung, G. Ananthanarayanan, P. Bodik, L. Golubchik, M. Yu, P. Bahl, and M. Philipose, "Videoedge: Processing camera streams using hierarchical clusters," in *SEC*. IEEE, 2018, pp. 115–131.

[36] M. Cui, S. Zhong, B. Li, X. Chen, and K. Huang, "Offloading autonomous driving services via edge computing," *IEEE Internet of Things J.*, vol. 7, no. 10, pp. 10 535–10 547, 2020.

[37] P. Park, S. C. Ergen, C. Fischione, C. Lu, and K. H. Johansson, "Wireless network design for control systems: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 978–1013, 2017.

[38] P. Skarin, W. Tärneberg, K.-E. Årzen, and M. Kihl, "Towards mission-critical control at the edge and over 5g," in *EDGE*. IEEE, 2018.

[39] P. Skarin, J. Eker, M. Kihl, and K.-E. Årzén, "Cloud-assisted model predictive control," in *EDGE*. IEEE, 2019, pp. 110–112.

[40] Y. Ma, C. Chen, S. Zeng, X. Guan, and C. Lu, "Data-driven edge offloading for wireless control systems," *IEEE Internet of Things J.*, 2023.

[41] X. Hou, Z. Ren, J. Wang, W. Cheng, Y. Ren, K.-C. Chen, and H. Zhang, "Reliable computation offloading for edge-computing-enabled software-defined iov," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7097–7111, 2020.

[42] V. K. Akram, O. Dagdeviren, and B. Tavli, "Distributed k-connectivity restoration for fault tolerant wireless sensor and actuator networks: Algorithm design and experimental evaluations," *IEEE Transactions on Reliability*, vol. 70, no. 3, pp. 1112–1125, 2020.

[43] A. Cervin and J. Eker, "Control-scheduling codesign of real-time systems: The control server approach," *Journal of Embedded Computing*, vol. 1, no. 2, pp. 209–224, 2005.

[44] Y. Xu, T. He, R. Sun, Y. Ma, Y. Jin, and A. Zou, "Shape: Scheduling of fixed-priority tasks on heterogeneous architectures with multiple cpus and many pes," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–9.

[45] J.-J. Chen, G. Nelissen, W.-H. Huang, M. Yang, B. Brandenburg, K. Bletsas, C. Liu, P. Richard, F. Ridouard, N. Audsley *et al.*, "Many

suspensions, many problems: a review of self-suspending tasks in real-time systems," *Real-Time Systems*, vol. 55, no. 1, pp. 144–207, 2019.
[46] Y. Ma, C. Lu, and Y. Wang, "Efficient holistic control: Self-awareness across controllers and wireless networks," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 4, pp. 1–27, 2020.
[47] S. A. A. Rizvi and Z. Lin, "Output feedback q-learning control for the discrete-time linear quadratic regulator problem," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 5, 2018.
[48] K. Worthmann, M. Reble, L. Grune, and F. Allgower, "The role of sampling for stability and performance in unconstrained nonlinear model predictive control," *SIAM Journal on Control and Optimization*, 2014.
[49] S. Wang, S. Wen, K. Shi, X. Zhou, and T. Huang, "Approximate optimal control for nonlinear systems with periodic event-triggered mechanism," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.
[50] B. Luo, T. Huang, and D. Liu, "Periodic event-triggered suboptimal control with sampling period and performance analysis," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1253–1261, 2019.
[51] N. C. Audsley, "Optimal priority assignment and feasibility of static priority tasks with arbitrary start times," Department of Computer Science, University of York, UK, Tech. Rep., 1991.
[52] M. Bertogna and M. Cirinei, "Response-time analysis for globally scheduled symmetric multiprocessor platforms," in *28th IEEE International Real-Time Systems Symposium (RTSS)*. IEEE, 2007, pp. 149–160.
[53] F. Zhang and A. Burns, "Schedulability analysis for real-time systems with edf scheduling," *IEEE Transactions on Computers*, 2009.
[54] L. An and G.-H. Yang, "Data-based distributed sensor scheduling for multiple linear systems with H-∞ performance preservation," *IEEE Transactions on Automatic Control*, vol. 67, no. 12, 2021.
[55] D. Soudbakhsh, A. Annaswamy, and H. Voit, "Adaptation in network control systems with hierarchical scheduling," *IET Control Theory & Applications*, vol. 13, no. 17, pp. 2775–2782, 2019.
[56] B. Li, L. Nie, C. Wu, H. Gonzalez, and C. Lu, "Incorporating emergency alarms in reliable wireless process control," in *ACM/IEEE International Conference on Cyber-Physical Systems*. IEEE, 2015.
[57] A. K. Singh, P. Dziurzanski, H. R. Mendis, and L. S. Indrusiak, "A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–40, 2017.