Event-Driven Joint Mobile Actuators Scheduling and Control in Cyber-Physical Systems

Lei Mo, *Member, IEEE*, Pengcheng You, *Student Member, IEEE*, Xianghui Cao, *Senior Member, IEEE*, Yeqiong Song, *Member, IEEE*, and Angeliki Kritikakou

***Abstract*—In cyber-physical systems, mobile actuators can enhance system's flexibility and scalability, but at the same time incurs complex couplings in the scheduling and controlling of the actuators. In this paper, we propose a novel event-driven method aiming at satisfying a required level of control accuracy and saving energy consumption of the actuators, while guaranteeing a bounded action delay. We formulate a joint-design problem of both actuator scheduling and output control. To solve this problem, we propose a two-step optimization method. In the first step, the problem of actuator scheduling and action time allocation is decomposed into two subproblems. They are solved iteratively by utilizing the solution of one in the other. The convergence of this iterative algorithm is proved. In the second step, an online method is proposed to estimate the error and adjust the outputs of the actuators accordingly. Through simulations and experiments, we demonstrate the effectiveness of the proposed method.**

***Index Terms*—Actuation delay, control, cyber-physical systems (CPSs), experiments, joint design, mobile actuator, scheduling.**

I. Introduction

CYBER-PHYSICAL systems (CPS) bridge the cyber world with the physical world using sensors and actuators connected through a wireless communication network. Sensing information and controlling actions through CPSs remove the limitations of wired connections and fixed network structures, and thus, the flexibility and the scalability of the system can be further enhanced. CPSs are characterized by sensor–actuator coordination, heterogeneous information communication, and intelligent decision/actuation [1]. These characteristics enable the use of CPSs to several domains such as smart building [2], [3], power grids [4], mobile charging [5]–[8], and industrial and environment control [9]–[12].

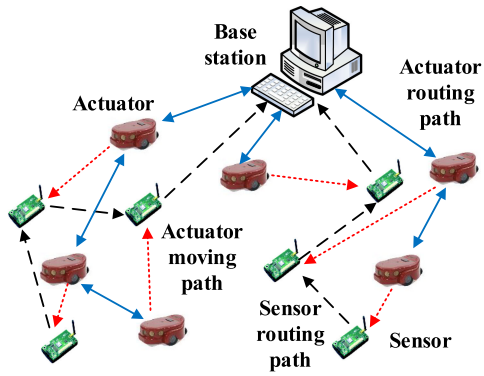


Fig. 1. Example of a CPS with mobile actuators.

To support the requirements of the applications through the CPS, sensors and actuators have to efficiently coordinate with each other. For example, in mobile charging application [5], as shown in Fig. 1, sensors perform environment monitoring, while actuators (i.e., mobile chargers) are responsible for charging the sensors. Each sensor reports its energy level periodically to the base station (BS). When the energy of a sensor is lower than a predefined threshold, an event is triggered at the BS. Based on the energy information collected from the sensors and the charging abilities of actuators, the BS coordinates the actuators on how to perform energy charging tasks. The coordination includes both the scheduling and the control of the actuators. The scheduling problem refers to the relocation of actuators [5], [11], [13]. On the other hand, the control problem refers to the output adjustment of actuators [4], [9], [12], which directly influence the physical variables under consideration. Since different actuator scheduling schemes lead to different actuator control decisions, to find the optimal solution, the actuator scheduling and control problems should be jointly addressed.

The design of the scheduling of mobile actuators and the control of their outputs should meet: 1) control accuracy requirement [6], e.g., to ensure that the sensors have enough energy to work until being charged again, the replenished energy of the

Manuscript received November 13, 2018; accepted March 7, 2019. Date of publication March 25, 2019; date of current version November 5, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61403340 and Grant 61573103, in part by the AppRoximaTivE Flexible Circuits and Computing for IoT Project under Grant ANR-15-CE25-0015, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20180012, and in part by the Lorraine Regional Project under Grant AME SATELOR. Paper no. TII-18-3022. *(Corresponding author: Xianghui Cao.)*

L. Mo and A. Kritikakou are with the University of Rennes, Institut National de Recherche en Informatique et en Automatique (INRIA), Centre National de la Recherche Scientifique (CNRS), Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), 35042 Rennes, France (e-mail: lei.mo@inria.fr; angeliki.kritikakou@irisa.fr).

P. You is with the Whiting School of Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: pcyou@jhu.edu).

X. Cao is with the School of Automation, Southeast University, Nanjing 210096, China (e-mail: xhcao@seu.edu.cn).

Y. Song is with the Lorraine Laboratory of IT Research and Applications, University of Lorraine, CNRS, INRIA, 54000 Nancy, France (e-mail: ye-qiong.song@loria.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2906061

TABLE I
CLASSIFICATION OF NODE COORDINATION METHODS

References		*With stationary actuators*									*With mobile actuators*								
		[2]	[3]	[4]	[9]	[10]	[12]	[15]	[16]	[17]	[5]	[6]	[7]	[8]	[11]	[13]	[14]	[18]	This paper
Actuator	Single											√	√	√	√	√			
	Multiple	√	√	√	√	√	√	√	√	√	√						√	√	√
Objective	Control accuracy	√	√	√	√	√	√	√	√	√		√		√			√		√
	Task real-time		√	√			√	√		√	√	√	√	√	√		√		√
	Energy efficiency	√	√				√		√		√	√	√	√	√	√	√	√	√
Solution	Non-optimal							√			√	√	√	√	√	√	√	√	
	Optimal	√	√	√	√	√	√		√	√									√

sensors should reach to some certain levels; 2) real-time requirement [8], e.g., the tasks assigned to the actuators (relocating and controlling) should be finished within a predefined threshold so as to generate a correct result; and 3) energy efficiency requirement [14]. Although actuators are more powerful than sensors, they still have limited energy budgets: 1) actuators cannot be easily recharged after deployment; and 2) relocating and controlling tasks consume much more energy than computation and communication tasks. To meet the above system requirements, the efficient coordination of actuators is crucial.

The existing work on actuator coordination can be classified according to whether: 1) the system runs with multiple actuators; 2) the actuator is mobile; 3) the problem is multiobjective; and 4) the solution is optimal. Table I provides a summary of some representative papers from the literature. The extension from the single-actuator (stationary actuators or single objective) case to the multiactuator (mobile actuators or multiobjective) case is not straightforward, since additional variables and constraints need to be added into the problem formulation to satisfy new requirements. When taking multiobjective such as the multiactuator scheduling and control, the real-time performance, and the energy efficiency into account, optimization variables are coupled with each other nonlinearly. Heuristics [5], [14] and evolutionary approaches [8] are popular methods to solve complex optimization problems. However, the solution qualities of these methods are hard to guarantee. Finding an optimal solution is also very important. Only by doing so, we can find out how far the nonoptimal solution is from the optimal one, and how to improve nonoptimal approaches based on the optimal solution.

Complementary to the state of the art, this paper jointly optimizes the multiactuator scheduling and control under the control accuracy, energy, and real-time constraints and provides an optimal solution with a reduced computing time. Our main contributions are summarized as follows.

1) To jointly optimize the scheduling and the control of actuators under multiple system requirements, we introduce an actuator scheduling and action time allocation (ASATA) problem. The ASATA problem is a mixed-integer program (MIP) problem. To reduce the computational complexity, the MIP-based ASATA problem is relaxed to a mixed-integer linear program (MILP). Instead of periodically updating the actuator scheduling and action time decisions, decisions are updated only when events occur.
2) To efficiently solve the MILP-based ASATA problem, we divide this problem into two correlated subproblems: an integer linear program (ILP)-based subproblem, dedicated to the actuator scheduling (master problem—MP), and a linear program (LP)-based subproblem, dedicated to the action time allocation (slave problem—SP). The correlated subproblems are solved in an iterative way, where the solution of the one subproblem is propagated to the other. We prove that through limited iterations between the MP and the SP, the algorithm converges to the global optimal solution.
3) Using the solution of the ASATA problem to schedule the actuators and adjust their action time, an error is introduced, since the actuators are mobile and the system states are dynamic. To enhance the control accuracy, as well as to eliminate performance degradation caused by the problem relaxation from MIP to MILP, we propose an online method to estimate the introduced error. On this basis, we introduce an LP-based actuator output adjustment (AOA) problem to adjust the outputs of the actuators.
4) We evaluate the performance of the proposed method by both simulations and experiments that are based on a physical testbed.

The remainder of this paper is organized as follows. Section II presents the related work. Section III introduces the system model and formulates the problem under study. Section IV presents the joint-design algorithm. Section V shows the simulation and experimental results. Section VI concludes this study.

II. RELATED WORK

With respect to the actuator control problem, requirements on the control accuracy should be satisfied [2]. On this basis, the control quality can be further enhanced by reducing the action delay [12] and packet-loss rate [10] or resisting the disturbances [16] and inaccurate system parameters [9]. Some works consider the network delay, energy efficiency, and control accuracy joint optimization [3] or the communication protocol and control accuracy joint optimization [4], [15], [17]. However, the aforementioned studies mainly focus on the control of stationary actuators.

With respect to the actuator scheduling problem, studies exist for single-actuator and multiactuator cases. For the single-actuator scheduling case, the location and the emergency of an event are estimated by the maximum likelihood estimation, and the actuator is scheduled by the Markov decision processes to handle this event in [11]. The traveling salesman problem (TSP) [13] and the orienteering problem (OP) [7] can be used to formulate the actuator scheduling problems. The basic idea of the TSP (or OP) is utilizing a graph to model the system, where

vertices are usually associated with profits (e.g., priorities of events), while edges are usually associated with costs (e.g., the moving time or moving energy of the actuator). The aim of the problem is to maximize profits (or minimize the costs), under the constraint that all the vertices must be visited (or the moving time is limited). For the multiactuator scheduling case, the TSP can be extended to the multiple TSP [5]. In [18], to balance the workloads (i.e., travel cost and data collection cost) of actuators, sensors are divided into several groups, and actuators are scheduled to visit these groups in sequence. However, the actuator output control problem is not taken into account in these studies, since actuators are mainly used to visit the event area [11] or collect data from sensors [13], [18] or are required to arrive at sensors before the energy of sensors is lower than a predefined threshold [5], [7].

In [14], considering the energy consumed by actuators to move and perform action, the actuator scheduling and control problem is formulated by a mixed-integer nonlinear program. In [6] and [8], the actuator is scheduled to visit the sensors in sequence and changes their energy to some certain levels. However, the dynamics of system states is not taken into account in [6], [8], and [14]. During the movement of actuators, system states will change, and thus, the actuator control decision should be updated accordingly. For the dynamic system, it will introduce a nonlinear coupling among the optimization variables (i.e., the actuator scheduling and control decisions). The mobile actuator scheduling and/or control problems are usually NP-hard. The common solutions include: 1) heuristics, e.g., greedy and approximation algorithms [5], [7], [13], [14], [18]; 2) evolutionary approaches, e.g., genetic algorithm (GA) [8]; and 3) problem relaxation, e.g., under specific conditions or assumptions, the nonlinear program problem can be transformed to an MILP problem [6]. Although heuristics are able to find the feasible solution in a short time, they do not provide bounds on solution quality and are sensitive to changes in problem structures.

III. System Model and Problem Formulation

In this section, we first present the system model with stationary sensors and mobile actuators. Then, we formulate the joint-design problem of multiple actuator scheduling and control. The main notations are summarized in Table II.

A. System Model

We consider that n_s static sensors $\{\mathbb{S}_1, \ldots, \mathbb{S}_{n_s}\}$ and n_a mobile actuators $\{\mathbb{A}_1, \ldots, \mathbb{A}_{n_a}\}$ are randomly deployed in a region of interest (ROI) to monitor n_p system states $\{x_1, \ldots, x_{n_p}\}$ and take necessary actions to deal with the events in that area, respectively. System states represent physical variables under consideration. We assume that: 1) the velocity of the actuator is constant and no obstacle exists in the ROI, similar to the previously published works [6], [13], [14]; 2) there are n_l working points $\{\mathbb{L}_1, \ldots, \mathbb{L}_{n_l}\}$ that the actuators can stay and perform actions; and 3) the actuator $\mathbb{A}_j$ starts performing actions only when it arrives at the designed working point.

TABLE II
Main Notations

n_s	number of sensors
n_a	number of actuators
n_p	number of system states
n_l	number of working points
$\mathbb{L}_i$	the i^{th} working point
$\mathbb{A}_j$	the j^{th} actuator
$\mathbb{S}_l$	the l^{th} sensor
$\boldsymbol{x}(k)$	system states at the k^{th} step
$\boldsymbol{u}(k)$	actuators' outputs at the k^{th} step
$\boldsymbol{z}(k)$	sensors' measurements at the k^{th} step
Δ_s	system's sampling period
τ	action delay
$[\underline{\boldsymbol{x}}_{th}, \overline{\boldsymbol{x}}_{th}]$	event trigger threshold
$[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$	user's requirement on $\boldsymbol{x}$
$[\underline{\boldsymbol{u}}, \overline{\boldsymbol{u}}]$	bounds of actuators' outputs
$\boldsymbol{x}(k\|k)$	a posteriori estimate of $\boldsymbol{x}(k)$
$\boldsymbol{x}(k+1\|k)$	a priori estimate of $\boldsymbol{x}(k)$
$d_{ij}(k)$	distance between $\mathbb{L}_i$ and $\mathbb{A}_j$ at the k^{th} step
v	mean velocity of $\mathbb{A}_j$
u_j	output amplitude of $\mathbb{A}_j$ during the action time before output adjustment
t_j^m	maximum action time of $\mathbb{A}_j$
k_d	energy consumed by actuator to move a unit distance
k_u	coefficient of actuator's action energy
s_{ij}	$=\begin{cases}1 & \text{if } \mathbb{A}_j \text{ is scheduled to } \mathbb{L}_i\\ 0 & \text{otherwise}\end{cases}$
t_{ij}	action time of $\mathbb{A}_j$ at $\mathbb{L}_i$
$\Delta u_j(k)$	output adjustment of $\mathbb{A}_j$ at the k^{th} step

Since the information exchange among sensors and actuators is carried out by discrete data packets, we consider a linear discrete-time model for the dynamic physical system as

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}(k)\boldsymbol{u}(k) + \boldsymbol{\omega}(k) \tag{1}$$

where $\boldsymbol{x}(k) = [x_1(k), \ldots, x_{n_p}(k)]'$. $\boldsymbol{A}$ and $\boldsymbol{B}(k)$ are the system matrix and the input matrix with appropriate dimensions, respectively. Note that the input matrix $\boldsymbol{B}(k)$ may vary in different time steps depending on the scheduling of actuators. This is because the actuators may be scheduled to move to different working points and only those active ones that arrive at the designated working points start performing control actions. The system noises $\boldsymbol{\omega}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$ and the outputs of the actuators are bounded by $\underline{\boldsymbol{u}} \preceq \boldsymbol{u}(k) \preceq \overline{\boldsymbol{u}}$. $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ represents that the random variables $\boldsymbol{y}$ follow a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\sigma}$. $\boldsymbol{p} \preceq \boldsymbol{q}$ represents $p_i \leq q_i$ $(1 \leq i \leq m)$, where $\boldsymbol{p} = [p_1 \ldots, p_m]'$ and $\boldsymbol{q} = [q_1 \ldots, q_m]'$.

Sensors measure the system states $\boldsymbol{x}(k)$ as follows:

$$\boldsymbol{z}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{\nu}(k) \tag{2}$$

where $\boldsymbol{z}(k) = [z_1(k), \ldots, z_{n_s}(k)]'$ are the measurements of sensors, $\boldsymbol{C}$ is the measurement matrix, and the measurement noises $\boldsymbol{\nu}(k) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R})$.

For the system described by (1) and (2), a Kalman filter (KF) is an optimal estimator, since it provides a minimum variance

unbiased estimate of system states $\boldsymbol{x}(k)$. Therefore, we run the KF at each step k to estimate system states $\boldsymbol{x}(k)$ from the noisy measurements $\boldsymbol{z}(k)$

$$\begin{aligned}\boldsymbol{x}(k+1|k) &= \boldsymbol{A}\boldsymbol{x}(k|k) + \boldsymbol{B}(k)\boldsymbol{u}(k)\\ \boldsymbol{P}(k+1|k) &= \boldsymbol{A}\boldsymbol{P}(k+1|k)\boldsymbol{A}' + \boldsymbol{Q}\\ \boldsymbol{G}(k) &= \boldsymbol{P}(k+1|k)\boldsymbol{C}'\left(\boldsymbol{C}\boldsymbol{P}(k|k-1)\boldsymbol{C}' + \boldsymbol{R}\right)^{-1}\\ \boldsymbol{x}(k|k) &= \boldsymbol{x}(k|k-1) + \boldsymbol{G}(k)\left(\boldsymbol{z}(k) - \boldsymbol{C}\boldsymbol{x}(k|k-1)\right)\\ \boldsymbol{P}(k|k) &= \boldsymbol{P}(k|k-1) - \boldsymbol{G}(k)\boldsymbol{C}\boldsymbol{P}(k|k-1)\end{aligned}$$

where $\boldsymbol{G}(k)$ is the Kalman gain, $\boldsymbol{x}(k|k)$ and $\boldsymbol{x}(k+1|k)$ are the *a posteriori* and *a priori* estimates of system states $\boldsymbol{x}(k)$, respectively. $\boldsymbol{P}(k|k)$ and $\boldsymbol{P}(k+1|k)$ are the error covariance matrices with respect to the estimates $\boldsymbol{x}(k|k)$ and $\boldsymbol{x}(k+1|k)$, respectively. The communication delay is not considered in this paper.

Definition 3.1 (Event): If the state estimate $\boldsymbol{x}(k|k)$ is out of a predefined threshold $[\underline{\boldsymbol{x}}_{\text{th}}, \overline{\boldsymbol{x}}_{\text{th}}]$, while the state estimate $\boldsymbol{x}(k-1|k-1)$ was within that threshold, an event occurs at the kth step.

Definition 3.2 (Action delay): For an event, the action delay τ is the number of sampling periods after the event occurrence, during which the user's requirement on the system states, i.e., $[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$, remains unsatisfied.

Definition 3.3 (Action time): The action time of actuator $\mathbb{A}_j$ is the number of sampling periods, during which the $\mathbb{A}_j$'s output is not equal to zero, i.e., $u_j(k) \neq 0$.

The sensors $\{\mathbb{S}_1, \ldots, \mathbb{S}_{n_s}\}$ transmit their measurements $\boldsymbol{z}(k)$ to the BS at each step k to estimate the system states $\boldsymbol{x}(k)$. Without losing generality, we assume that an event occurs at the kth step. Therefore, the BS schedules the actuators to the designed working points and adjusts their outputs $\boldsymbol{u}(k)$ to control the system states $\boldsymbol{x}(k+\tau|k+\tau-1)$ to meet user's requirement $[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$. Based on different applications, the ranges $[\underline{\boldsymbol{x}}_{\text{th}}, \overline{\boldsymbol{x}}_{\text{th}}]$ and $[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$ may be different.

B. Preliminaries

Equation (1) shows that the system states $\boldsymbol{x}(k+\tau)$ are determined by the input matrices $\{\boldsymbol{B}(k), \ldots, \boldsymbol{B}(k+\tau-1)\}$ and the outputs of the actuators $\{\boldsymbol{u}(k), \ldots, \boldsymbol{u}(k+\tau-1)\}$. Due to the product of the variables $\boldsymbol{B}(l)\boldsymbol{u}(l)$ $(k \leq l \leq k+\tau-1)$, it is difficult to solve $\boldsymbol{B}(l)$ and $\boldsymbol{u}(l)$ directly. Alternatively, we fix the output amplitude of the actuators as $\boldsymbol{u} = [u_1, \ldots, u_{n_a}]'$ during the action time and introduce binary matrix $\boldsymbol{S} = [s_{ij}]_{n_l \times n_a}$ $(s_{ij} \in [0,1])$ and integer matrix $\boldsymbol{T} = [t_{ij}]_{n_l \times n_a}$ $(t_{ij} \in \mathbb{Z}^+)$ to schedule the actuators and adjust their action time during the steps $[k, k+\tau-1]$. This problem is called the ASATA problem, which is detailed in Section III-C. The benefit of introducing variables $\boldsymbol{S}$ and $\boldsymbol{T}$ to the problem formulation is that they are coupled with each other linearly, which makes the problem easier to solve. On the other hand, to enhance the control accuracy, based on the solution of the ASATA problem, we introduce a continuous vector $\Delta\boldsymbol{u}(l) = [\Delta u_1(l), \ldots, \Delta u_{n_a}(l)]'$ $(\Delta u_j(l) \in \mathbb{R})$ to adjust the outputs of the actuators at the lth step $(k \leq l \leq k+\tau-1)$. This problem is called the AOA problem, which is detailed in Section III-D. If the values of variables $\boldsymbol{S}, \boldsymbol{T}$, and $\Delta\boldsymbol{u}(l)$ are determined, the input matrix $\boldsymbol{B}(k)$ and the outputs of the actuators $\boldsymbol{u}(k)$ during the steps $[k, k+\tau-1]$ are calculated as follows.

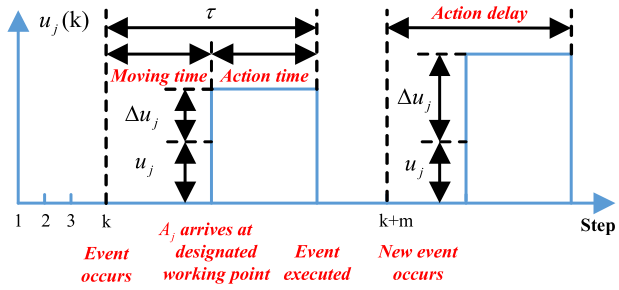


Fig. 2. Control sequence of the actuator $\mathbb{A}_j$.

1) Following the scheduling decision $\boldsymbol{S}$ to move the actuators, the location of the actuator $\mathbb{A}_j$ at the lth step $(k \leq l \leq k+\tau-1)$ is known. We assume that at the lth step, the relocation of the actuators $\{\mathbb{A}_1, \ldots, \mathbb{A}_j\}$ has been completed, while the actuators $\{\mathbb{A}_{j+1}, \ldots, \mathbb{A}_{n_a}\}$ are still moving. With the given sampling period Δ_s, we obtain an $n_p \times j$ matrix $\hat{\boldsymbol{B}}$ by using the methods in [2] and [10], and thus, the input matrix $\boldsymbol{B}(l) = [\hat{\boldsymbol{B}}, \boldsymbol{0}_{n_p \times (n_a - j)}]$. This matrix remains constant until a new actuator (e.g., $\mathbb{A}_{j+1}$) arrives at its designed working point.
2) Denote $d_{ij}^s(k) = \lceil \frac{d_{ij}(k)}{v\Delta_s} \rceil$ as the number of steps required for the actuator to move $d_{ij}(k)$ distance, where $\lceil y \rceil \triangleq \min\{n \in \mathbb{Z} | y \leq n\}$. According to the scheduling decision $\boldsymbol{S}$ and the action time decision $\boldsymbol{T}$, the actuator $\mathbb{A}_j$ takes $\sum_{i=1}^{n_l} s_{ij} d_{ij}^s(k)$ steps to move and $\sum_{i=1}^{n_l} t_{ij}$ steps to perform the actions. Therefore, the output of the actuator $\mathbb{A}_j$ at the lth step is

$$u_j(l) = \begin{cases} u_j, & \text{if } k + \sum_{i=1}^{n_l} s_{ij} d_{ij}^s(k) \leq l \leq k \\ & \quad + \sum_{i=1}^{n_l} s_{ij} d_{ij}^s(k) + \sum_{i=1}^{n_l} t_{ij} \\ 0, & \text{otherwise.} \end{cases}$$

3) Based on the control decisions $\boldsymbol{u}(l)$ and $\Delta\boldsymbol{u}(l)$, the outputs of actuators at the lth step are given by $\boldsymbol{u}(l) + \Delta\boldsymbol{u}(l)$, as shown in Fig. 2.

From the above statement, we can see that the ASATA problem and the AOA problem are solved in sequence. In the following sections, we explain how to formulate the actuator scheduling and control joint-design problem through new variables $\boldsymbol{S}$, $\boldsymbol{T}$, and $\Delta\boldsymbol{u}(l)$.

C. ASATA Problem

The problem consists of an objective function that minimizes the moving and the action energy consumption of actuators subject to the control accuracy and the action real-time constraints. Under these constraints, we determine: 1) which

working point should the actuator be assigned to (scheduling); and 2) when the actuators start and end the actions (action time allocation). Let $\mathcal{I} \triangleq \{1, \ldots, n_l\}$, $\mathcal{J} \triangleq \{1, \ldots, n_a\}$ and $\mathcal{M} \triangleq \{1, \ldots, n_p\}$. Since each actuator moves toward at most one working point, the scheduling variable s_{ij} should satisfy the inequality

$$\sum_{i=1}^{n_l} s_{ij} \leq 1 \quad \forall j \in \mathcal{J}. \tag{3}$$

The maximum action time of the actuator $\mathbb{A}_j$ is limited by its residual energy [6]. Hence, the action time variable t_{ij} is bounded by

$$0 \leq t_{ij} \leq t_j^m s_{ij} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}. \tag{4}$$

To finish the relocating and the controlling tasks within the action delay τ, we have

$$s_{ij} d_{ij}^s(k) + t_{ij} \leq \tau \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}. \tag{5}$$

Note that the system states are hard to precisely estimate when the scheduling and the action time decisions $\boldsymbol{S}$ and $\boldsymbol{T}$ are unknown. In the ASATA problem, we let the system matrix $\boldsymbol{A} = \boldsymbol{I}$. To meet the control requirement $[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$, we have

$$\underline{x}_m \leq x_m(k|k) + \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} u_j t_{ij} \tilde{b}_{mi} \leq \overline{x}_m \quad \forall m \in \mathcal{M}. \tag{6}$$

$\tilde{b}_{mi}$ is the (m, i)th element of input matrix $\tilde{\boldsymbol{B}}$, and it evaluates the influence of the actuators on the system state x_m, where these actuators are placed to the working point $\mathbb{L}_i$. Note that matrices $\tilde{\boldsymbol{B}}$ and $\boldsymbol{B}$ are different. First, the dimension of $\tilde{\boldsymbol{B}}$ is $n_p \times n_l$, while the dimension of $\boldsymbol{B}$ is $n_p \times n_a$. Second, $\tilde{\boldsymbol{B}}$ is fixed and given in advance, while $\boldsymbol{B}$ is time varying and determined by the scheduling decision. In Section III-D, we explain the necessity to introduce constraint (6) and how to extend it to the case $\boldsymbol{A} \neq \boldsymbol{I}$ through the AOA problem.

Our objective is to minimize the moving and action energy consumption of the actuators. Specifically, the moving and action energies are assumed to be proportional to the moving distance and the action time, respectively [6]. Therefore, the objective function is $\Phi(\boldsymbol{S}, \boldsymbol{T}) = \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} (k_d s_{ij} d_{ij}(k) + k_u |u_j| t_{ij})$. Summarizing the objective and the aforementioned constraints, the ASATA problem is formulated as

$$\mathbf{P1}: \min_{\boldsymbol{S}, \boldsymbol{T}} \Phi(\boldsymbol{S}, \boldsymbol{T})$$

$$\text{s.t.} \begin{cases} (3)\text{–}(6), \\ s_{ij} \in \{0, 1\}, t_{ij} \in \mathbb{Z}^+, 0 \leq t_{ij} \leq t_j^m \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}. \end{cases} \tag{7}$$

Since $s_{ij} \in \{0, 1\}$ and $t_{ij} \in \mathbb{Z}^+$ $(\forall i \in \mathcal{I}, \forall j \in \mathcal{J})$, P1 is an MIP problem. To reduce the computational complexity, we relax P1 to the following MILP problem:

$$\mathbf{P2}: \min_{\boldsymbol{S}, \hat{\boldsymbol{T}}} \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} \left(k_d s_{ij} d_{ij} + k_u |u_j| \hat{t}_{ij}\right)$$

$$\text{s.t.} \begin{cases} (3), \\ \hat{t}_{ij} \leq t_j^m s_{ij} \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ s_{ij} d_{ij}^s(k) + \hat{t}_{ij} \leq \tau \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ \underline{x}_m \leq x_m(k|k) + \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} u_j \hat{t}_{ij} \tilde{b}_{mi} \leq \overline{x}_m \; \forall m \in \mathcal{M} \\ s_{ij} \in \{0, 1\}, \hat{t}_{ij} \in \mathbb{R}^+, 0 \leq \hat{t}_{ij} \leq t_j^m \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}. \end{cases} \tag{8}$$

Finding an optimal scheduling decision is the most important step to solve P2. Since if the value of binary variables $\boldsymbol{S}$ is determined, P2 reduces to an LP problem, which has a simpler structure and is more easy to solve. In contrast, for P1, even if the value of $\boldsymbol{S}$ is given, we still need to solve an ILP problem. Solving an ILP problem is more complex than solving an LP problem, especially when the problem size is large [19].

Equations (1) and (2) show that system is discrete, and thus, each actuator maintains the output stepwise. We define $\tilde{\boldsymbol{T}} = [\tilde{t}_{ij}]_{n_l \times n_a}$ as the real action time matrix, where $\tilde{t}_{ij} = \lfloor \hat{t}_{ij} \rfloor$ and $\lfloor y \rfloor \triangleq \max\{n \in \mathbb{Z} | n \leq y\}$. Hence, the actuator $\mathbb{A}_j$ takes $\sum_{i=1}^{n_l} \tilde{t}_{ij}$ steps to perform actions. Since $\tilde{t}_{ij} \leq \hat{t}_{ij} \leq t_{ij}$ $(\forall i \in \mathcal{I}, \forall j \in \mathcal{J})$, the real action time decision $\tilde{\boldsymbol{T}}$ does not violate constraints (4) and (5) except constraint (6). In Section III-D, we explain how to mitigate this influence through the AOA problem.

D. AOA Problem

When $\boldsymbol{A} \neq \boldsymbol{I}$, constraint (6) should be constructed by using the predicted system states. For example, at the kth step, by solving P2, the actuator $\mathbb{A}_1$ is scheduled to the working point $\mathbb{L}_1$ to control the system state x_1, and the moving time takes k_1 steps. When the actuator $\mathbb{A}_1$ arrives at the designed working point $\mathbb{L}_1$ and starts acting, the state of x_1 has been changed from $x_1(k)$ to $x_1(k + k_1)$. If the scheduling decision $\boldsymbol{S}$ and the action time decision $\tilde{\boldsymbol{T}}$ are given, we obtain the input matrices $\{\boldsymbol{B}(k), \ldots, \boldsymbol{B}(k + \tau - 1)\}$ and the outputs of the actuators $\{\boldsymbol{u}(k), \ldots, \boldsymbol{u}(k + \tau - 1)\}$. Therefore, the system states at the lth step $(k + 1 \leq l \leq k + \tau)$ are estimated by

$$\boldsymbol{x}(l|k) = \boldsymbol{A}^{l-k} \boldsymbol{x}(k|k) + \sum_{i=k}^{l-1} \boldsymbol{A}^{l-1-i} \boldsymbol{B}(i) \boldsymbol{u}(i).$$

Note that $\boldsymbol{x}(l|k)$ is a KF-based *a priori* estimate. The *a posteriori* estimates $\{\boldsymbol{x}(k + 1|k + 1), \ldots, \boldsymbol{x}(k + \tau|k + \tau)\}$ are unavailable, since the future measurements $\{\boldsymbol{z}(k + 1), \ldots, \boldsymbol{z}(k + \tau)\}$ are unknown at the current step k. In contrast, if the scheduling decision $\boldsymbol{S}$ and the action time decision $\tilde{\boldsymbol{T}}$ are unknown, the input matrix $\boldsymbol{B}(l)$, the outputs of the actuators $\boldsymbol{u}(l)$, and the *a priori* estimate $\boldsymbol{x}(l|k)$ $(k + 1 \leq l \leq k + \tau)$ are hard to derive. This is because different $\boldsymbol{S}$ and $\hat{\boldsymbol{T}}$ decisions lead to different values of $\boldsymbol{B}(l)$, $\boldsymbol{u}(l)$, and $\boldsymbol{x}(l|k)$. In addition, it is difficult to formulate the functions of $\boldsymbol{B}(l)$, $\boldsymbol{u}(l)$, and $\boldsymbol{x}(l|k)$ by using the

variables $\boldsymbol{S}$ and $\hat{\boldsymbol{T}}$, i.e., it is hard to construct constraint (6) through the state estimate $\boldsymbol{x}(l|k)$.

Since the *a posteriori* estimate $\boldsymbol{x}(k|k)$ is known at the current step k, we can construct the constraint (6) through $\boldsymbol{x}(k|k)$, while satisfying the real-time and control accuracy requirements through another way, e.g., adjusting the outputs of the actuators during the steps $[k, k+\tau-1]$. Therefore, we introduce an AOA problem. The basic idea is similar to the closed-loop control: we 1) fix $\boldsymbol{u}$ to determine the decisions $\boldsymbol{S}$ and $\tilde{\boldsymbol{T}}$; 2) estimate the error under the given decisions $\boldsymbol{S}$ and $\tilde{\boldsymbol{T}}$; and 3) adjust the outputs of the actuators $\{\boldsymbol{u}(k), \ldots, \boldsymbol{u}(k+\tau-1)\}$ based on the estimated error. The details are as follows.

1) At the kth step, we obtain the scheduling decision $\boldsymbol{S}$ and the action time decision $\tilde{\boldsymbol{T}}$ by solving P2; furthermore, we derive the input matrix $\boldsymbol{B}(l)$ and the outputs of the actuators $\boldsymbol{u}(l)$ at the lth step $(k+1 \le l \le k+\tau)$. We define $\Delta\mathcal{U} = [\Delta\boldsymbol{u}(k), \ldots, \Delta\boldsymbol{u}(k+\tau-1)]$ as the output adjustment decision during the steps $[k, k+\tau-1]$. To drive the *a priori* estimate $\boldsymbol{x}(k+\tau|k)$ toward the user's requirement $[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$, we let the output adjustment variables $\Delta\mathcal{U}$ to satisfy the following constraint:

$$\underline{\boldsymbol{x}} \preceq \boldsymbol{x}(k+\tau|k) = \boldsymbol{A}^{\tau}\boldsymbol{x}(k|k) + \sum_{l=k}^{k+\tau-1} \boldsymbol{A}^{k+\tau-1-l}\boldsymbol{B}(l)(\boldsymbol{u}(l)+\Delta\boldsymbol{u}(l)) \preceq \overline{\boldsymbol{x}}. \quad (9)$$

2) Note that: 1) the output adjustment variables $\Delta\boldsymbol{u}(l)$ are bounded by $\underline{\boldsymbol{u}} \preceq \boldsymbol{u}(l)+\Delta\boldsymbol{u}(l) \preceq \overline{\boldsymbol{u}}$ $(k+1 \le l \le k+\tau)$; and 2) the output amplitude vector $\boldsymbol{u}$ is fixed and given. The AOA problem is formulated as

$$\mathbf{P3}: \min_{\Delta\mathcal{U}} \sum_{l=k}^{k+\tau-1}\sum_{j=1}^{n_a} |\Delta u_j(l)|$$
$$\text{s.t.} \begin{cases} (9), \\ \underline{\boldsymbol{u}} \preceq \boldsymbol{u}(l)+\Delta\boldsymbol{u}(l) \preceq \overline{\boldsymbol{u}},\ k \le l \le k+\tau-1. \end{cases} \quad (10)$$

Equation (9) shows that the real-time and control accuracy constraints are actually determined by P3. Constraints (5) and (6) can be removed from P2. However, P2 with these constraints can provide more accurate scheduling and action time decisions, i.e., the solution of P3 is more easy to find out (the constraints of P3 are more easy to satisfy) if constraints (5) and (6) are included in P2. Therefore, P1 can be replaced by P2 due to the introduction of P3.

IV. Algorithm Design and Analysis

In this section, we propose a joint actuator scheduling and control algorithm to solve P2 and P3. As shown in Fig. 3, the proposed algorithm contains two steps. In the first step, based on Benders decomposition [20], we divide P2 into two subproblems and then find the global optimal solution by iterating the subproblems. In the second step, based on the solution of the first step, we solve P3 in an online manner and adjust the outputs of the actuators accordingly.

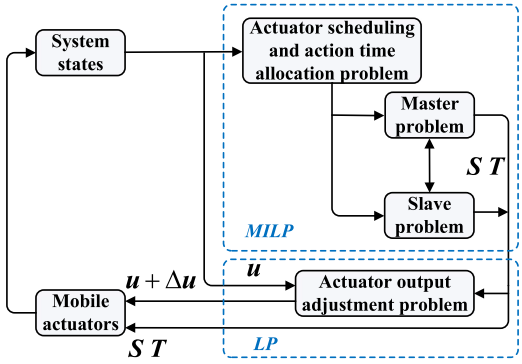


Fig. 3. Joint multiactuator scheduling and control algorithm.

A. Decomposition-Based ASATA Algorithm

As we mentioned before, finding an optimal scheduling decision $\boldsymbol{S}$ is the most important step to solve P2. Based on this idea, we divide P2 into two subproblems with less variables and constraints: an MP and an SP. Instead of considering all variables and constraints simultaneously, P2 is solved by iterating the MP and the SP. By doing so, the computing time can be further reduced. The MP accounts for all the binary variables and the associated portion of the objective function and the constraints of P2. It also includes the information regarding the SP via a set of constraints called *Benders cuts*. The SP includes all the continuous variables and the associated constraints of P2. Solving the SP provides some information regarding the SP portion of the P2, and this information is included in the MP through Benders cut.

For simplicity and generality, we remove the step index k from the equations. Based on the structure of P2, the MP and the SP are formulated as

$$\mathbf{MP}: \Phi_L(l) = \min_{\boldsymbol{S},\hat{\Phi}} \hat{\Phi}$$
$$\text{s.t.} \begin{cases} (3), \\ C_1: \hat{\Phi} \ge \Gamma(\boldsymbol{S}, \boldsymbol{\alpha}(\varsigma), \boldsymbol{\beta}(\varsigma), \boldsymbol{\gamma}(\varsigma), \boldsymbol{\psi}(\varsigma))\ \forall \varsigma \in \mathcal{A} \\ C_2: 0 \ge \Lambda(\boldsymbol{S}, \hat{\boldsymbol{\alpha}}(\vartheta), \hat{\boldsymbol{\beta}}(\vartheta), \hat{\boldsymbol{\gamma}}(\vartheta), \hat{\boldsymbol{\psi}}(\vartheta))\ \forall \vartheta \in \mathcal{B}. \end{cases} \quad (11)$$

where

$$\Gamma(\boldsymbol{S}, \boldsymbol{\alpha}(\varsigma), \boldsymbol{\beta}(\varsigma), \boldsymbol{\gamma}(\varsigma), \boldsymbol{\psi}(\varsigma)) = \sum_{i=1}^{n_l}\sum_{j=1}^{n_a}[k_d s_{ij} d_{ij} - t_j^m s_{ij}\alpha_{ij}(\varsigma) + (s_{ij}d_{ij}^s - \tau)\beta_{ij}(\varsigma)] + \sum_{m=1}^{n_p}[(x_m - \overline{x}_m)\psi_m(\varsigma) - (x_m - \underline{x}_m)\gamma_m(\varsigma)]$$

$$\Lambda(\boldsymbol{S}, \hat{\boldsymbol{\alpha}}(\vartheta), \hat{\boldsymbol{\beta}}(\vartheta), \hat{\boldsymbol{\gamma}}(\vartheta), \hat{\boldsymbol{\psi}}(\vartheta)) = \Gamma(\boldsymbol{S}, \hat{\boldsymbol{\alpha}}(\vartheta), \hat{\boldsymbol{\beta}}(\vartheta), \hat{\boldsymbol{\gamma}}(\vartheta), \hat{\boldsymbol{\psi}}(\vartheta)) - \sum_{i=1}^{n_l}\sum_{j=1}^{n_a} k_d s_{ij} d_{ij}.$$

C_1 and C_2 are the sets of the feasibility constraints (FCs) and the infeasibility constraints (ICs), respectively. $\mathcal{A}$ and $\mathcal{B}$ are

the sets of iterations that the dual slave problem (DSP) (13) has the bounded and the unbounded solutions, respectively. $(\boldsymbol{\alpha}(\varsigma), \boldsymbol{\beta}(\varsigma), \boldsymbol{\gamma}(\varsigma), \boldsymbol{\psi}(\varsigma))$ is the solution of the DSP at the ςth iteration. $(\hat{\boldsymbol{\alpha}}(\vartheta), \hat{\boldsymbol{\beta}}(\vartheta), \hat{\boldsymbol{\gamma}}(\vartheta), \hat{\boldsymbol{\psi}}(\vartheta))$ is the solution of the dual feasibility check problem (DFCP) (20) at the ϑth iteration. Note that the objective function of P2 contains the binary variables $\boldsymbol{S}$, as well as the continuous variables $\hat{\boldsymbol{T}}$, while MP only considers the binary variables $\boldsymbol{S}$. We introduce an auxiliary variable $\hat{\Phi}$ into the MP as the objective function, where $\hat{\Phi}$ has the same physical meaning as Φ, as

$$\mathbf{SP}: \Phi_U(l) = \min_{\hat{\boldsymbol{T}} \succeq 0} \Phi(\boldsymbol{S}(l), \hat{\boldsymbol{T}})$$
$$\text{s.t.} \begin{cases} \hat{t}_{ij} \le t_j^m s_{ij}(l) \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ s_{ij}(l) d_{ij}^s + \hat{t}_{ij} \le \tau \Delta_s \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ \underline{x}_m \le x_m + \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} u_j \hat{t}_{ij} \tilde{b}_{mi} \le \overline{x}_m \quad \forall m \in \mathcal{M} \end{cases} \tag{12}$$

where $(\boldsymbol{S}(l), \hat{\Phi}(l))$ is the solution of the MP at the lth iteration. Comparing the SP with P2, we observe that their formulations are the same with the exception that the binary variables $\boldsymbol{S}$ in the SP are fixed.

For the MP, since the constraints regarding the action time variables $\hat{\boldsymbol{T}}$ are relaxed, solving this problem yields a lower bound $\Phi_L(l)$. On the other hand, since the scheduling decision $\boldsymbol{S}(l)$ may be just a feasible solution (not optimal yet), solving the SP yields an upper bound $\Phi_U(l)$ (the proofs are provided in Appendix A). Denote $(\boldsymbol{S}^*, \hat{\boldsymbol{T}}^*)$ as the optimal solution of P2, and $\Phi^* = \Phi(\boldsymbol{S}^*, \hat{\boldsymbol{T}}^*)$. Hence, we have $\Phi_L(l) \le \Phi^* \le \Phi_U(l)$. At each iteration l, a new FC or IC (Benders cuts) is generated and added into the MP to reduce the gap between the bounds $\Phi_L(l)$ and $\Phi_U(l)$ (the proof is provided in Appendix B). The iteration process is summarized as follows.

1) Step 1 (Initialization): Initialize the iteration counter $l = 0$, the solution $\boldsymbol{S}(0)$ of the MP, the lower bound $\Phi_L(0) = -\infty$, and the upper bound $\Phi_U(0) = +\infty$. The sets of FCs and ICs, i.e., C_1 and C_2, are set to null. The initial solution $\boldsymbol{S}(0)$ is given arbitrarily, as long as it satisfies constraint (3).

2) Step 2 (Solving the SP): In this paper, rather than solving the SP directly, we solve its dual problem. This is because the SP is an LP problem; the optimal objective function values of the SP and its dual problem are equivalent due to the strong duality [21]. In addition, the FC (14) and the IC (15) can be constructed through the solution of the DSP. To develop the dual of the SP, the positive Lagrange multipliers $\boldsymbol{\alpha} = [\alpha_{ij}]$, $\boldsymbol{\beta} = [\beta_{ij}]$, $\boldsymbol{\gamma} = [\gamma_m]$, and $\boldsymbol{\psi} = [\psi_m]$ $(\forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall m \in \mathcal{M})$ are introduced to the SP. Hence, the Lagrangian is

$$\mathcal{L}_1(\boldsymbol{S}(l), \hat{\boldsymbol{T}}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi}) = \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} [k_d s_{ij}(l) d_{ij}$$
$$+ k_u |u_j| \hat{t}_{ij} + (\hat{t}_{ij} - t_j^m s_{ij}(l)) \alpha_{ij} + (s_{ij}(l) d_{ij}^s + \hat{t}_{ij} - \tau) \beta_{ij}]$$
$$- \sum_{m=1}^{n_p} \left(\sum_{i=1}^{n_l} \sum_{j=1}^{n_a} u_j \hat{t}_{ij} \tilde{b}_{mi} + x_m - \underline{x}_m \right) \gamma_m$$
$$+ \sum_{m=1}^{n_p} \left(\sum_{i=1}^{n_l} \sum_{j=1}^{n_a} u_j \hat{t}_{ij} \tilde{b}_{mi} + x_m - \overline{x}_m \right) \psi_m .$$

The dual function [21] is defined as the minimum value of Lagrangian $\mathcal{L}_1(\boldsymbol{S}(l), \hat{\boldsymbol{T}}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi})$ with respect to the variables $\hat{t}_{ij}$, i.e.,

$$\mathcal{D}(\boldsymbol{S}(l), \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi}) = \min_{\hat{\boldsymbol{T}} \succeq 0} \left\{ \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} [k_u |u_j| + \alpha_{ij} + \beta_{ij} \right.$$
$$\left. + \sum_{m=1}^{n_p} u_j \tilde{b}_{mi} (\psi_m - \gamma_m) \right] \hat{t}_{ij} + \Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi}) \Bigg\}.$$

Since the variables $\hat{t}_{ij}$ are positive, they are finite only when $k_u |u_j| + \alpha_{ij} + \beta_{ij} + \sum_{m=1}^{n_p} u_j \tilde{b}_{mi} (\psi_m - \gamma_m) \ge 0$ $(\forall i \in \mathcal{I}, \forall j \in \mathcal{J})$. Therefore, the DSP is formulated as

$$\mathbf{DSP}: \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi} \succeq 0} \Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi})$$
$$\text{s.t.} \quad k_d |u_j| + \alpha_{ij} + \beta_{ij} + \sum_{m=1}^{n_p} u_j \tilde{b}_{mi} (\psi_m - \gamma_m) \ge 0$$
$$\forall i \in \mathcal{I}, \forall j \in \mathcal{J} \tag{13}$$

3) Step 3 (Solving the MP): Based on the solution of the DSP, two different types of new constraints are generated and added into the MP at the next iteration.

1) If the DSP is *infeasible*, the SP has an unbounded solution. Hence, P2 has no feasible solution.
2) If the DSP has a *bounded solution* $(\boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))$, $\mathcal{A} \leftarrow \{l\} \cup \mathcal{A}$. The upper bound is updated by $\Phi_U(l) = \min\{\Phi_U(l-1), \Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))\}$. Due to the strong duality, the SP is feasible. Denote $\hat{\boldsymbol{T}}(l)$ as the solution of the SP at the lth iteration. Note that $\hat{\Phi}(l) < \Phi(\boldsymbol{S}(l), \hat{\boldsymbol{T}}(l)) = \Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))$. To avoid selecting the nonoptimal solution $\boldsymbol{S}(l)$ again, a new FC
$$\hat{\Phi} \ge \Gamma(\boldsymbol{S}, \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l)) \tag{14}$$
is generated and added into C_1 at the $(l+1)$th iteration.
3) If the DSP has an *unbounded solution*, i.e., $\Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l)) = +\infty$, due to the strong duality, the SP has no feasible solution under the given $\boldsymbol{S}(l)$, and $\mathcal{B} \leftarrow \{l\} \cup \mathcal{B}$. To exclude the infeasible solution $\boldsymbol{S}(l)$, we construct a feasibility check problem (FCP) (19) and solve its dual problem. Based on the solution of the DFCP (20), i.e., $(\hat{\boldsymbol{\alpha}}(l), \hat{\boldsymbol{\beta}}(l), \hat{\boldsymbol{\gamma}}(l), \hat{\boldsymbol{\psi}}(l))$, a new IC
$$0 \ge \Lambda(\boldsymbol{S}, \hat{\boldsymbol{\alpha}}(l), \hat{\boldsymbol{\beta}}(l), \hat{\boldsymbol{\gamma}}(l), \hat{\boldsymbol{\psi}}(l)) \tag{15}$$
is generated and added into C_2 at the $(l+1)$th iteration.

When the MP is solved, the iteration counter l increases, and steps 2 and 3 are repeated. The iteration stops when $|\Phi_U(l) - \Phi_L(l)| \le \varepsilon$, where ε is a small positive value. Note that the SP and P2 have the same objective functions, as well as the fact that the SP and the DSP are equivalent due to the strong duality. From inequation (14), we can see that the auxiliary variable $\hat{\Phi}$ has the same physical meaning as Φ.

Lemma IV.1: The lower bound $\Phi_L(l)$ (upper bound $\Phi_U(l)$) on the optimal objective function value Φ^* can be derived from the solution of the MP (SP) at the lth iteration.

Proof: See Appendix A for the proof. ■

Theorem IV.1: At each iteration with FC (14) or IC (15) added into the MP, the solution converges to the global optimal value within a finite number of iterations.

Proof: See Appendix B for the proof. ■

B. Estimation-Based AOA Algorithm

Since the objective function of P3 contains the absolute value $|\Delta u_j(l)|$, this formulation is not a standard LP. To transform P3 into a standard LP problem, we introduce two new variables $\Delta\tilde{u}_j(l)$ and $\Delta\bar{u}_j(l)$ to replace the original variable $\Delta u_j(l)$, where $\Delta\tilde{u}_j(l) = \frac{|\Delta u_j(l)|+\Delta u_j(l)}{2}$ and $\Delta\bar{u}_j(l) = \frac{|\Delta u_j(l)|-\Delta u_j(l)}{2}$. With the new output adjustment variables $\Delta\tilde{\mathcal{U}} = [\Delta\tilde{\boldsymbol{u}}(k), \ldots, \Delta\tilde{\boldsymbol{u}}(k+\tau-1)]$ and $\Delta\bar{\mathcal{U}} = [\Delta\bar{\boldsymbol{u}}(k), \ldots, \Delta\bar{\boldsymbol{u}}(k+\tau-1)]$, P3 is rewritten as

$$\mathbf{P4}: \min_{\Delta\tilde{\mathcal{U}},\Delta\bar{\mathcal{U}}} \sum_{l=k}^{k+\tau-1}\sum_{j=1}^{n_a}(\Delta\tilde{u}_j(l)+\Delta\bar{u}_j(l))$$

$$\text{s.t.}\begin{cases} \underline{\boldsymbol{x}} \preceq \boldsymbol{x}(k+\tau|k) = \boldsymbol{A}^{\tau}\boldsymbol{x}(k|k)+ \\ \sum_{l=k}^{k+\tau-1}\boldsymbol{A}^{k+\tau-1-l}\boldsymbol{B}(l)(\boldsymbol{u}(l)+\Delta\tilde{\boldsymbol{u}}(l)-\Delta\bar{\boldsymbol{u}}(l)) \preceq \overline{\boldsymbol{x}}, \\ \underline{\boldsymbol{u}} \preceq \boldsymbol{u}(l)+\Delta\tilde{\boldsymbol{u}}(l)-\Delta\bar{\boldsymbol{u}}(l) \preceq \overline{\boldsymbol{u}},\ k \le l \le k+\tau-1. \end{cases} \tag{16}$$

Following the solution of P4 (i.e., $\Delta\mathcal{U} = \Delta\tilde{\mathcal{U}} + \Delta\bar{\mathcal{U}}$) to adjust the actuators' outputs during the steps $[k, k+\tau-1]$, we have $\underline{\boldsymbol{x}} \preceq \boldsymbol{x}(k+\tau|k) \preceq \overline{\boldsymbol{x}}$. However, $\underline{\boldsymbol{x}} \preceq \boldsymbol{x}(k+\tau|k) \preceq \overline{\boldsymbol{x}}$ cannot guarantee that $\underline{\boldsymbol{x}} \preceq \boldsymbol{x}(k+\tau|k+\tau-1) \preceq \overline{\boldsymbol{x}}$. The state estimate $\boldsymbol{x}(k+\tau|k+\tau-1)$ may be out of the range $[\underline{\boldsymbol{x}}, \overline{\boldsymbol{x}}]$ due to the system noises $\boldsymbol{\omega}(k)$. To achieve $\underline{\boldsymbol{x}} \preceq \boldsymbol{x}(k+\tau|k+\tau-1) \preceq \overline{\boldsymbol{x}}$, the only way is based on $\boldsymbol{x}(k+\tau-1|k+\tau-1)$ to adjust $\boldsymbol{u}(k+\tau-1)$. Therefore, we propose an online output adjustment method. The details are as follows.

1) At the lth step, we construct P4 based on the output adjustment variables $\Delta\tilde{\mathcal{U}} = [\Delta\tilde{\boldsymbol{u}}(l), \ldots, \Delta\tilde{\boldsymbol{u}}(k+\tau-1)]$ and $\Delta\bar{\mathcal{U}} = [\Delta\bar{\boldsymbol{u}}(l), \ldots, \Delta\bar{\boldsymbol{u}}(k+\tau-1)]$, and the current step state estimate $\boldsymbol{x}(l|l)$, where the control accuracy constraint is given by $\underline{\boldsymbol{x}} \preceq \boldsymbol{x}(k+\tau|l) \preceq \overline{\boldsymbol{x}}$.
2) We solve P4 through the LP and follow its solution, i.e., $\boldsymbol{u}(l)+\Delta\boldsymbol{u}(l) = \boldsymbol{u}(l)+\Delta\tilde{\boldsymbol{u}}(l)+\Delta\bar{\boldsymbol{u}}(l)$, to adjust the outputs of the actuators at the lth step.

The above two steps are repeated until $l = k+\tau-1$. For the online output adjustment method, we need to solve P4 $\tau-1$ times during the steps $[k, k+\tau-1]$. However, with step number increasing, the dimension of variables $\Delta\mathcal{U}$ reduces.

V. Performance Evaluation

A. Simulation Setup

We consider a wireless rechargeable sensor network (WRSN) as the simulation and experimental case study. Eight sensors and eight mobile actuators are randomly deployed in a 20 m × 20 m ROI to perform the environment sensing and the sensor energy charging tasks ($n_s = n_a = 8$). Mobile robots and sensors (e.g., Mica2) that equipped with Powercast chargers and receivers (see Fig. 4) are served as mobile chargers (actuators) and rechargeable sensors, respectively. Therefore, the system states $\boldsymbol{x}$ represent the residual energy of the sensors ($n_p = 8$). The system parameters are summarized in Table III. The sensor model is adopted from [6], where r_i is the energy consumption rate of the sensor $\mathbb{S}_i$. f_{ij} (f_{ib}) is the flow rate from $\mathbb{S}_i$ to $\mathbb{S}_j$ (from $\mathbb{S}_i$ to BS), ρ and η_{ij} (or η_{ib}) are the rate of energy consumption for receiving a unit of data rate and transmitting a unit of data rate from $\mathbb{S}_i$ to $\mathbb{S}_j$ (or BS), respectively. λ_1 and λ_2 are the distance-independent and distance-dependent constant terms, respectively. d_{ij} is the distance between $\mathbb{S}_i$ and $\mathbb{S}_j$. θ is the path loss index. Here, we consider that r_i is invariant with time. For a regular AA battery, its nominal cell voltage and the quantity of electricity is 1.2 V/2.5 Ah. Since two AA batteries provide an average voltage 2.4 V for the Mica2 and the operating limit is 2.1 V [22], we have $\underline{x}_{th,i} = 2.1 \times 2.5 \times 3600 = 18\,900$ J and $\overline{x}_{th,i} = 2.4 \times 2.5 \times 3600 = 21\,600$ J [6].

As shown in the experiments in [23], when a charger is placed 10 cm away from the receiver, the charging efficiency reduces to 1.5%. To enhance the charging efficiency, we consider that each sensor has only one working point, which is very close to this sensor ($n_l = 8$). Hence, the charging efficiency increases to 6% [23]. Since $n_p = n_a$, all the sensors can be charged in one round. If $n_p > n_a$, based on the lifetime of sensors, we can divide the charging process into several rounds, and then, in each round, we have $n_p \le n_a$. Note that different sensor and actuator parameters lead to different values of parameters for P2 and P4. However, structures of problems under different values

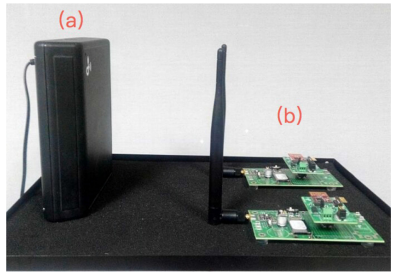


Fig. 4. (a) Powercast charger. (b) Powercast receivers.

TABLE III
System Parameters

Sensor $\mathbb{S}_i$ characteristics				
$r_i = \rho\sum_{k=1,k\neq i}^{n_s} f_{ki} + \sum_{j=1,j\neq i}^{n_s}\eta_{ij}f_{ij} + \eta_{ib}f_{ib}$				
$\eta_{ij} = \lambda_1 + \lambda_2(d_{ij})^{\theta}$		$\rho = 50$ nJ/b	$f_{ij} \in [1, 10]$ kb/s	
$\lambda_1 = 50$ nJ/b		$\lambda_2 = 0.0013$ pJ/(b.m^4)		$\theta = 4$
Actuator $\mathbb{A}_j$ characteristics				
$b_{ij} = 6$ %	$u_j = 3$ W	$\underline{u}_j = 0$ W	$\bar{u}_j = 5$ W	$v = 0.5$ m/s
System state characteristics				
$n_a = 8$	$n_s = 8$	$\Delta_s = 1$ s	$\tau = 1000$ s	
$\underline{x}_{th,i} = 18900$ J		$\underline{x}_i = 21580$ J		$n_p = 8$
$\overline{x}_{th,i} = 21600$ J		$\overline{x}_i = 21600$ J		$n_l = 8$

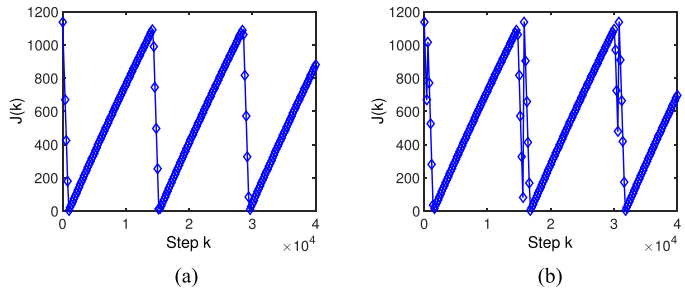


Fig. 5. System performance to deal with sequential and random events. (a) Sequential events. (b) Random events.

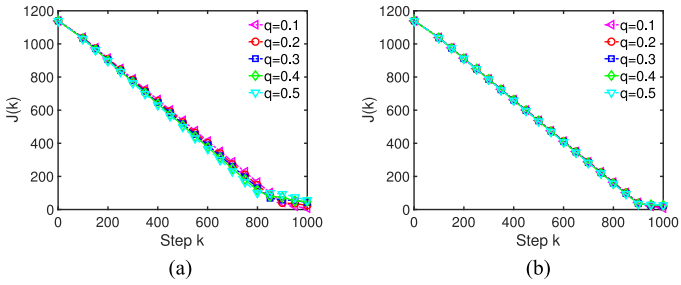


Fig. 6. Control errors with online and offline output adjustment methods. (a) With the offline adjustment method. (b) With the online adjustment method.

of parameters are the same, and thus, the proposed methods are still applicative. The simulations are performed on a PC with a dual-core 2.6-GHz Intel i5 processor and 12-GB RAM, and the algorithms are implemented in MATLAB 2013a.

B. System Performance

The initial system states $\boldsymbol{x}(1)$ are assumed to be in the range [18 700, 18 900] J. Hence, an event occurs at step $k = 1$. We use the root mean square error between lower bound $\underline{\boldsymbol{x}}$ and *a posteriori* estimate $\boldsymbol{x}(k|k)$, i.e., $J(k) = \sqrt{\frac{(\underline{\boldsymbol{x}}-\boldsymbol{x}(k|k))'(\underline{\boldsymbol{x}}-\boldsymbol{x}(k|k))}{n_p}}$, to evaluate the control error. Following the solutions of P2 and P4, i.e., the matrices $\boldsymbol{S}$, $\tilde{\boldsymbol{T}}$, and $\Delta\mathcal{U}$, to schedule the actuators and adjust their outputs, the dynamic change of control error $J(k)$ is shown in Fig. 5(a). At the beginning, actuators take several steps to relocate. When actuators arrive at the designed working points and start charging the sensors, the control error $J(k)$ converges to δ gradually, where δ is a small positive value. After the energy charging process is completed, the control error $J(k)$ will gradually increase. When a new event occurs, the actuator scheduling and control decisions should be updated again.

Fig. 5(b) evaluates the system performance (control error) to deal with random events, where the sensors $\{\mathbb{S}_1, \mathbb{S}_3\}$, $\{\mathbb{S}_4, \mathbb{S}_8\}$, $\{\mathbb{S}_2, \mathbb{S}_7\}$, and $\{\mathbb{S}_5, \mathbb{S}_6\}$ require charging at steps $k = 1$, 500, 15 800, and 30 800, respectively. From Fig. 5(b), we observe that new events occur when actuators are still handling the previous events. In this case, the scheduling decision $\boldsymbol{S}$ and the action time decision $\tilde{\boldsymbol{T}}$ should be updated again, even if the previous tasks have not finished yet. Note that there is no need to update decisions $\boldsymbol{S}$ and $\tilde{\boldsymbol{T}}$ at each step k. This is because solving an MILP problem at each step k is time-consuming and changing scheduling decision $\boldsymbol{S}$ at each step k will cause actuators to move in a zigzag way. Since the decision update process is event driven rather than time driven, we can handle random event efficiently, while avoiding high computational complexity.

Fig. 6 compares the changes of control error $J(k)$ with the online and offline output adjustment methods. The offline method solves P4 only once at step $k = 1$. We set $\boldsymbol{Q} = q \cdot \boldsymbol{I}_{n_p \times n_p}$ and change the value of q from 0.1 to 0.5 with a step of 0.1. With the offline method, since $\boldsymbol{x}(k+\tau|k)$ is a multistep prediction, the control error $J(k+\tau)$ increases with the value of q. In contrast, with the online method, the influence of q on the control error $J(k+\tau)$ is limited. This is because the output adjustment

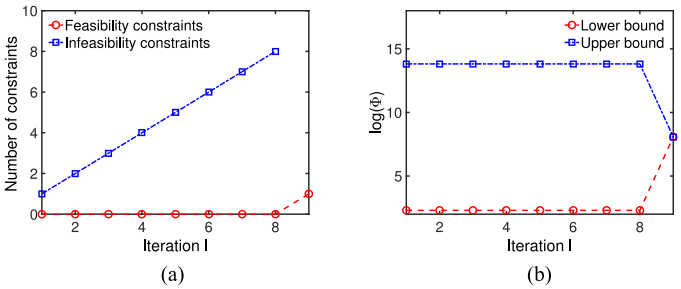


Fig. 7. Convergence of the decomposition-based ASATA algorithm. (a) Number of FCs and ICs. (b) Gap between the lower and upper bounds.

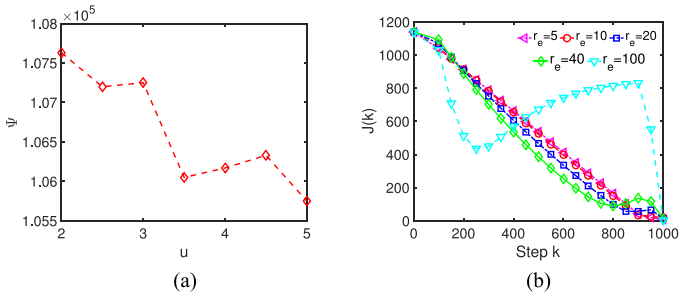


Fig. 8. System performance with system parameter varying. (a) Actuator energy consumption with $\boldsymbol{u}$ varying. (b) Control error with r_e varying.

matrix $\Delta\mathcal{U}$ needs to be updated at each step k, and the update process is based on the *a posteriori* estimate $\boldsymbol{x}(k|k)$ of the current step. Although the online method needs to solve P4 multiple times, P4 can be solved very fast using polynomial-time algorithms as it is an LP problem. Using the online method, we can get a better robustness against the system noises $\boldsymbol{\omega}(k)$.

Fig. 7 shows the convergence of the decomposition-based ASATA algorithm. With the IC (14) and the FC (15) added into the MP during the iterations $l = 1 \sim 8$, and $l = 9$, respectively, the upper bound $\Phi_U(l)$ and the lower bound $\Phi_L(l)$ quickly converge to the optimal value Φ^*.

C. Scalability and Robustness Evaluation

Denote Ψ as the total energy consumption of the actuators during the steps $[1, 40\,000]$. Fig. 8(a) evaluates the values of Ψ under different actuator output amplitudes $\boldsymbol{u}$. We set $n_a = n_p =$

TABLE IV
ASATA CONVERGENCE ITERATION WITH n_p VARYING

n_p	5	10	15	20	25	30	35	40	45	50
Iterations	4	9	14	19	27	30	32	35	41	49

8 and $u_i = u_j$ $(i \neq j, 1 \leq i, j \leq n_a)$ and change the value of u_j $(1 \leq j \leq n_a)$ from 2 to 5 with a step of 0.5. Fig. 8(a) shows that the differences are small. This is because when the output amplitude $\boldsymbol{u}$ increases, the actuator action time $\tilde{\boldsymbol{T}}$ will reduce, while Φ represents the total energy consumption. In addition, the elements of output adjustment matrix $\Delta\mathcal{U}$ are usually very small, since the aim of P4 is to minimize them. However, if the value of u_j $(1 \leq j \leq n_a)$ is too small, e.g., $u_j = 1$, P2 or P4 may be infeasible.

Denote $r_e = \frac{\sum_{i=1}^{n_p} r_i}{n_p}$ as the average energy consumption rate of the sensors. Fig. 8(b) shows the values of control error $J(k)$ under different r_e. We set $n_a = n_p = 8$ and $u_j = 3$ $(1 \leq j \leq n_a)$ and change the value of r_e between [5, 100]. Note that: 1) r_e only influences the system matrix $\boldsymbol{A}$; and 2) we assume that $\boldsymbol{A} = I$ in P2. The solution of P2, i.e., the matrices $\boldsymbol{S}$ and $\tilde{\boldsymbol{T}}$, will not change with r_e. To compensate the error introduced by constraint (6), based on the real system matrix $\boldsymbol{A}$ $(\boldsymbol{A} \neq \boldsymbol{I})$, we construct P4 through the estimate of system states $\boldsymbol{x}(k+\tau)$. Hence, r_e will influence the solution of P4, i.e., the output adjustment matrix $\Delta\mathcal{U}$. Fig. 8(b) shows that the larger r_e is, the larger the error overshoot is. This is because before the actuator relocation is completed, a large r_e leads to a fast change of the residual energy $\boldsymbol{x}(k)$. Fig. 8(b) also shows that at the beginning, under different average energy consumption rates, the control error $J(k)$ could either increase or decrease. However, when the actuator relocation is completed, the control error $J(k)$ converges to δ within the action delay τ.

We define the ASATA convergence iteration as the number of iterations to achieve $|\Phi_U(l) - \Phi_L(l)| \leq \varepsilon$. Table IV shows that the ASATA convergence iteration almost linearly increases with the value of n_p, where we set $n_a = n_p$ and $u_j = 3$ $(1 \leq j \leq n_a)$ and change the value of n_p from 5 to 50 with a step of 5. The parameters in the simulations with $n_p = i$ $(i = 10, \ldots, 50)$ and $n_p = i - 5$ are correlated, i.e., the sensors and the actuators in the simulation with $n_p = i$ are extended based on the sensors and the actuators in the simulation with $n_p = i - 5$. Higher dimension of the system states usually involves more variables and constraints into the problem. Hence, more iterations are required to search for the optimal solution.

Since actuator scheduling and control problems are jointly addressed in P2 and the input matrix is determined by the scheduling decision, the input matrix is unknown until P2 is solved. This is in contrast to the traditional system stability analysis that the system model is usually given in advance. Hence, we evaluate the influence of the sampling period Δ_s through the experiments. We define the error convergence speed as the number of steps to achieve $J(k) \leq \delta$. Table V shows the changes of error convergence speed under different Δ_s. We set $n_a = n_p = 8$, $u_j = 3$ $(1 \leq j \leq n_a)$ and change the value of Δ_s between [1, 120] s. If the sampling period is too large, e.g., $\Delta_s = 100$ s, the control error $J(k)$ is hard to converge since P2 or P4 is infeasible.

TABLE V
ERROR CONVERGENCE SPEED WITH Δ_s VARYING

Δ_s (s)	1	20	40	60	80	100	120
Steps	1000	1000	1000	1000	1000	$+\infty$	$+\infty$

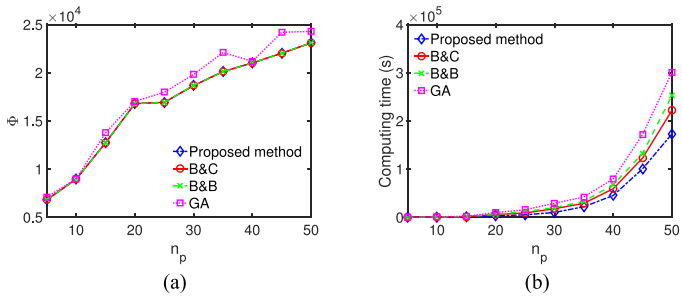
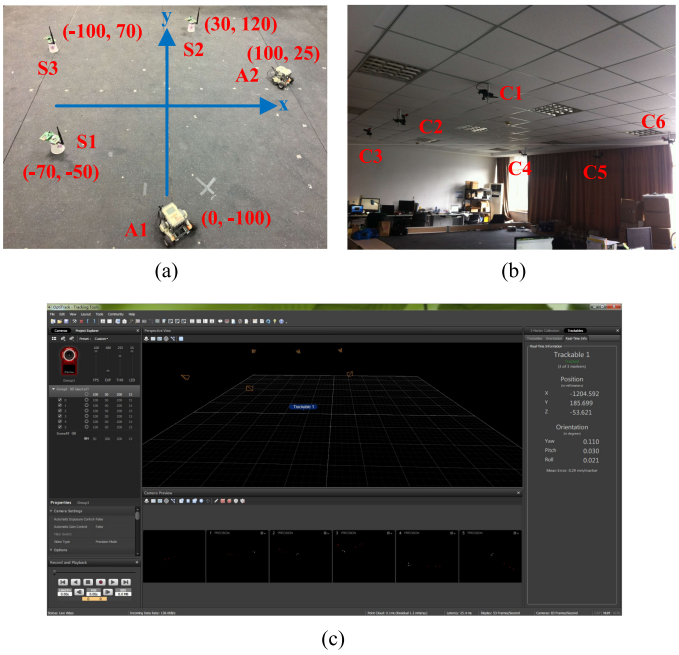


Fig. 9. Comparison between the proposed method and other methods. (a) Objective function. (b) Computing time.

D. Comparison With Existing Algorithms

Fig. 9(a) compares the performance (the objective function value and computation time) of the proposed decomposition-based method with: 1) optimal approaches: branch and bound (B&B) method [24] and branch and cut (B&C) method [25], which are known to provide the optimal solution for the MILP problems such as P2; and 2) evolutionary approach: GA [8], [26]. We set $n_a = n_p$ and $u_j = 3$ $(1 \leq j \leq n_a)$ and change the value of n_p from 5 to 50 with a step of 5. Fig. 9(a) shows that the solutions given by the proposed, B&B, and B&C methods are the same, since the proposed method is able to find the global optimal solution. In addition, the proposed method achieves a lower value for the objective function than the GA, since P2 is a minimization problem and there is no guarantee of convergence to a global optimum for the GA [19]. The convergence of the GA is sensitive to the choice of the genetic operators, the mutation probability, and the selection criteria, while fine-tuning of these parameters is often required.

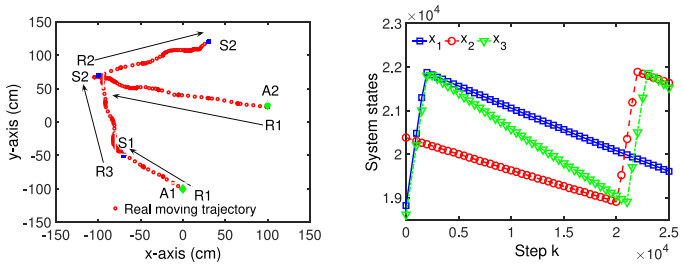
Fig. 9(b) shows that when the value of n_p is increased, the computing time of all the four methods increases, since more variables and constraints are involved into the problem, and thus, the problem size is enlarged. However, the proposed method takes a shorter computing time than the others. Compared with the proposed method, the GA is more complex, since it needs to generate new populations in each iteration by applying several procedures, such as selection, reproduction, mutation, and crossover. B&C, which combines the benefits of B&B and Gomory cutting schemes, can better explore the tradeoff between optimality, efficiency, and stability. Usually, B&C has a faster convergence speed than B&B [25]. For an optimization problem, the computing time increases significantly with the number



(a) (b)

(c)

Fig. 10. Overview of the testbed. (a) LEGO NXT robots and Mica2 nodes with Powercast receivers. (b) OptiTrack tracking system. (c) GUI.



(a) (b)

Fig. 11. Experimental results. (a) Real robot moving trajectory. (b) Dynamic sensor energy.

of variables and constraints. Hence, solving smaller problems (i.e., MP and SP) iteratively is more efficient than solving a single large problem. This result is in line with the comparison in [27], where the decomposition-based method is faster than B&B and B&C (computing time) methods when solving large problem instances.

E. Experiment

In this section, we evaluate the performance of the proposed method through the experiments, where three sensors (Mica2 equipped with Powercast receiver) are randomly deployed in a 3 m $\times$ 3 m ROI, and two LEGO NXT wheeled robots are employed to serve as the actuators (mobile chargers). The initial locations of the sensors and the robots are shown in Fig. 10(a). The constrained environment (e.g., collision/obstacle avoidance) is not considered in this paper. We use the OptiTrack system [28] to detect and track the movements of the robots. This system contains six cameras (with 0.3-MP resolution at 100 frames/s), which are installed on the ceiling [see Fig. 10(b)]. The cameras are connected to the BS (i.e., PC) via a USB hub, and the real-time positions of the mobile targets are shown through a graphical user interface (GUI) on the PC [see Fig. 10(c)]. The parameters of sensors and actuators are listed in Table III. The BS collects the real-time positions of the robots obtained from the OptiTrack, constructs P2 and P4, and solves the problems in MATLAB to obtain the actuator scheduling and control decisions and simulates the dynamic energy change of the sensors. At the beginning of the experiment, the BS simulates the dynamic change of the system states $\boldsymbol{x}(k)$ without the system inputs $\boldsymbol{u}(k)$. If an event is detected, the BS updates the scheduling and control decisions, and the mobile robots are relocated accordingly. When the mobile robots arrive at the designed working points and start performing actions, the BS calculates the dynamic change of system states $\boldsymbol{x}(k)$ under the influences of system inputs $\boldsymbol{u}(k)$.

When the start and goal points of each robot are determined, the BS performs the path tracking control through the quadratic curve method [29]. The aim of the path tracking control is to control the system states of a mobile robot $(p_x, p_y, p_\theta)'$ by using two parameters (φ_r, φ_l), where (p_x, p_y) and p_θ are the position and the posture angle of a robot, respectively. φ_r and φ_l are the angular velocities of the right and left wheels, respectively. The quadratic curve method contains two steps. In the first step, a quadratic curve that links the reference position (p_x^*, p_y^*) and the robot position (p_x, p_y) is calculated. In the second step, by using the error vector $(e_x, e_y, e_\theta)'$, the control inputs to the robot (φ_r, φ_l) are determined so that the robot can move along with the calculated quadratic curve. The control command (φ_r, φ_l) is sent from the BS to the robot via Bluetooth with a period of 0.2 s. Since this command only contains the angular velocities of the left and right wheels, the packet size of one command for a robot is usually smaller than 100 bytes. The control of the robot is carried out in a closed-loop manner. Based on the received control command, a classic proportional–integral–differential algorithm [30] is implemented in the robot to adjust the velocities of two driving wheels. Although there exist some errors between the designed and real goal points, these errors can be further reduced by adjusting the parameters of the algorithms, e.g., the period for sending commands.

The real robot moving trajectory (from OptiTrack) and the simulated sensor energy (from MATLAB) are shown in Fig. 11(a) and (b), respectively. Since the velocities of the robots are not constant, the mean velocity in P2 is set to $v = 6$ cm/s, which is an empirical value. The charging process contains three rounds: $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$. Table VI summarizes the experimental details, where I_r, L_i, L_f, T_m, and T_c are the scheduled robot, the initial and final positions of robots, and the moving and charging time of robots, respectively. The experimental results show that for the robots depending on different initial orientations, their movements are not always a straight line. In addition, the moving time of the robots is much shorter than the charging time of the sensors, and thus, the approximation of the mean velocity in P2 is acceptable. The experiments based on the testbed

TABLE VI
ROBOT SCHEDULING AND CHARGING TIME

	I_r	L_i	L_f	T_m (s)	T_c (s)
$\mathcal{R}_1$	$\mathbb{A}_1$	$\mathbb{A}_1$	$\mathbb{S}_1$	3.5	1859.8
$\mathcal{R}_1$	$\mathbb{A}_2$	$\mathbb{A}_2$	$\mathbb{S}_3$	6.3	2115.4
$\mathcal{R}_2$	$\mathbb{A}_2$	$\mathbb{S}_3$	$\mathbb{S}_2$	4.9	1792.3
$\mathcal{R}_3$	$\mathbb{A}_1$	$\mathbb{S}_1$	$\mathbb{S}_3$	4.8	1928.9

provide some demonstrative results of our proposed method for the multiactuator coordination. Based on different applications, e.g., fire monitoring and extinguishing [11], boundary and zone control [31], or multivehicle formation control [32], the LEGO robots can be replaced by appropriate types of robots. In these applications, the temperature or the gas concentration at some points of interest (POIs) or the distances between the leader and the followers can be viewed as the system states. Therefore, the aim of the problem is to schedule the mobile actuators and adjust their outputs to control the system states in order to meet the system requirements, such as control accuracy, real-time performance, and energy efficiency.

F. Discussion

In the simulations and experiments, we consider the WRSN as an illustrative example. However, the proposed method is not limited to the WRSN. Similar joint-design problems can be found in many other areas, such as the task mapping problem [33] in CPS devices. The CPS devices could be sensors, actuators, or BSs, depending on different applications, and they have different computing abilities. An application (e.g., image processing or target tracking) usually consists of a set of tasks. These tasks could be executed on different single-core devices (e.g., sensors or actuators) and/or on different cores of a multicore device (e.g., BS). For example, in the target tracking application, instead of collecting and sending all data to the BS, a part of the processing is done on site with sensors to reduce the network traffic. Therefore, only a small part of preprocessed data needs to be sent. This model of computation is known as "fog/edge computing."

Generally, the task mapping problem includes the task allocation subproblem with the aim to decide task-to-device and/or task-to-core allocation and the task scheduling subproblem with the aim to determine the start and the end time of each task. Since different task allocation schemes lead to different task scheduling decisions, the correlated subproblems should be optimized simultaneously to find the optimal solution. Usually, the task mapping is performed under a set of real-time and energy constraints, since the tasks need to be executed before a deadline to generate a correct result and some devices (e.g., sensors and actuators) have limited energy budgets. The energy consumption of the CPS devices is determined by the task mapping decision, which is a dynamic process. In this context, the approach presented in this paper can be easily extended to formulate the task mapping problem. Some existing works, e.g., [34], [35], have already considered the task mapping problem with the MILP structure. Such a problem can be optimally solved by the proposed decomposition-based method.

VI. CONCLUSION

In this paper, we solved the actuator scheduling and control joint-design problem in CPSs. To enhance system real-time and energy efficiency performances, as well as to reduce the error introduced by the actuator movement and dynamic system states, we formulated a joint-design problem that consists of an ASATA problem and an AOA problem. On this basis, we proposed a decomposition-based method and an estimation-based method to solve these problems efficiently. Finally, we analyzed the convergence and the control accuracy of the proposed methods. Simulation results demonstrated that the proposed methods are able to deal with the sequential and random events and achieve a tradeoff between the control accuracy and the computing time. The proposed method is also implemented to the real system. The experimental results showed that the desired control requirements are satisfied by scheduling the robots and controlling their actions.

APPENDIX A
PROOF OF LEMMA 4.1

Proof: Although the MP is composed of binary variables $\boldsymbol{S}$ and a continuous variable $\hat{\Phi}$, this problem can be solved by only considering the binary variables $\boldsymbol{S}$. Equation (11) shows that the effective constraints are (3) and C_2, while C_1 can be treated as the objective function. We assume that at the ςth $(1 \le \varsigma \le l)$ iteration, the DSP has a bounded solution $(\boldsymbol{\alpha}(\varsigma), \boldsymbol{\beta}(\varsigma), \boldsymbol{\gamma}(\varsigma), \boldsymbol{\psi}(\varsigma))$. Comparing the MP with the following ILP problem:

$$\hat{\Phi}_r(\varsigma) = \min_{\boldsymbol{S}} \Gamma(\boldsymbol{S}, \boldsymbol{\alpha}(\varsigma), \boldsymbol{\beta}(\varsigma), \boldsymbol{\gamma}(\varsigma), \boldsymbol{\psi}(\varsigma)) \quad \text{s.t.} \begin{cases} \sum_{i=1}^{n_l} s_{ij} \le 1 & \forall j \in \mathcal{J} \\ 0 \ge \Lambda(\boldsymbol{S}, \hat{\boldsymbol{\alpha}}(\vartheta), \hat{\boldsymbol{\beta}}(\vartheta), \hat{\boldsymbol{\gamma}}(\vartheta), \hat{\boldsymbol{\psi}}(\vartheta)) & \forall \vartheta \in \mathcal{B} \end{cases} \tag{17}$$

we have $\Phi_L(l) = \hat{\Phi}(l) = \max_{\forall \varsigma \in \mathcal{A}}\{\hat{\Phi}_r(\varsigma)\}$. Without loss of generality, we assume that $\hat{\Phi}(l) = \hat{\Phi}_r(\rho)$ $(\rho \in \mathcal{A})$. Thus, we have

$$\begin{aligned} \hat{\Phi}(l) &= \min_{\boldsymbol{S}} \Gamma(\boldsymbol{S}, \boldsymbol{\alpha}(\rho), \boldsymbol{\beta}(\rho), \boldsymbol{\gamma}(\rho), \boldsymbol{\psi}(\rho)) \\ &\le \Gamma(\boldsymbol{S}^*, \boldsymbol{\alpha}(\rho), \boldsymbol{\beta}(\rho), \boldsymbol{\gamma}(\rho), \boldsymbol{\psi}(\rho)) \qquad (18a) \\ &\le \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi} \succeq 0} \Gamma(\boldsymbol{S}^*, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\psi}) = \Phi^* \qquad (18b) \end{aligned}$$

where inequation (18a) holds, since $\boldsymbol{S}^*$ is not the optimal solution under the given multipliers $(\boldsymbol{\alpha}(\rho), \boldsymbol{\beta}(\rho), \boldsymbol{\gamma}(\rho), \boldsymbol{\psi}(\rho))$. Equation (18b) shows that $\Phi_L(l)$ is a lower bound of Φ^*. Depending on the solution of the DSP, its objective function value can be either finite or infinite. It is obvious that $+\infty$ is an upper bound of Φ^*. Thus, we focus on the case when the DSP has a

bounded solution $(\boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))$. Since

$$\Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l)) = \min_{\hat{\boldsymbol{T}} \succeq 0} \Phi(\boldsymbol{S}(l), \hat{\boldsymbol{T}}) \geq \min_{\hat{\boldsymbol{T}} \succeq 0} \Phi(\boldsymbol{S}^*, \hat{\boldsymbol{T}}) = \Phi^*$$

$\Phi_U(l) = \min\{\Phi_U(l-1), \Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))\} = \min_{1 \leq i \leq l}\{\Gamma(\boldsymbol{S}(i), \boldsymbol{\alpha}(i), \boldsymbol{\beta}(i), \boldsymbol{\gamma}(i), \boldsymbol{\psi}(i))\}$ is an upper bound of Φ^*. ■

APPENDIX B
PROOF OF THEOREM 4.1

Proof: At the lth iteration, if the DSP has a bounded solution $(\boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))$, the SP is feasible. The nonoptimal solution $\boldsymbol{S}(l)$ is excluded by the FC: $\hat{\Phi} \geq \Gamma(\boldsymbol{S}, \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))$. On the other hand, if the DSP has an unbounded solution, the SP is infeasible. For the SP, its feasibility is related to the constraints rather than the objective function. This problem may be feasible if the positive variables $\boldsymbol{\xi}^1 = [\xi_{ij}^1]$, $\boldsymbol{\xi}^2 = [\xi_{ij}^2]$, $\boldsymbol{\xi}^3 = [\xi_m^3]$, and $\boldsymbol{\xi}^4 = [\xi_m^4]$ $(\forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall m \in \mathcal{M})$ are introduced to relax the constraints. Therefore, we construct an FCP as

$$\mathbf{FCP}: \min_{\hat{\boldsymbol{T}}, \boldsymbol{\xi} \succeq 0} \Theta(\boldsymbol{\xi}) = \sum_{k=1}^{4} \left(\sum_{i=1}^{n_l} \sum_{j=1}^{n_a} \xi_{ij}^k + \sum_{m=1}^{n_p} \xi_m^k \right)$$

$$\text{s.t.} \begin{cases} \hat{t}_{ij} \leq t_j^m s_{ij}(l) + \xi_{ij}^1 & \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ \hat{t}_{ij} + s_{ij}(l) d_{ij}^s \leq \tau + \xi_{ij}^2 & \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ \underline{x}_m - \xi_m^3 \leq x_m + \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} u_j \hat{t}_{ij} \tilde{b}_{mi} \\ \quad \leq \overline{x}_m + \xi_m^4 & \forall m \in \mathcal{M}. \end{cases} \tag{19}$$

Introducing the positive Lagrange multipliers $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_{ij}]$, $\hat{\boldsymbol{\beta}} = [\hat{\beta}_{ij}]$, $\hat{\boldsymbol{\gamma}} = [\hat{\gamma}_m]$, and $\hat{\boldsymbol{\psi}} = [\hat{\psi}_m]$ $(\forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall m \in \mathcal{M})$ to the FCP, the Lagrangian is

$$\begin{aligned} &\mathcal{L}_2(\boldsymbol{S}(l), \hat{\boldsymbol{T}}, \boldsymbol{\xi}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\psi}}) \\ &= \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} \left\{ \left[\hat{\alpha}_{ij} + \hat{\beta}_{ij} + \sum_{m=1}^{n_p} u_j \tilde{b}_{mi} (\hat{\psi}_m - \hat{\gamma}_m) \right] \hat{t}_{ij} \right. \\ &\qquad \left. + (1 - \hat{\alpha}_{ij}) \xi_{ij}^1 + (1 - \hat{\beta}_{ij}) \xi_{ij}^2 \right\} \\ &\quad + \sum_{m=1}^{n_p} [(1 - \hat{\gamma}_m) \xi_m^3 + (1 - \hat{\psi}_m) \xi_m^4] \\ &\quad + \sum_{i=1}^{n_l} \sum_{j=1}^{n_a} [-t_j^m s_{ij}(l) \hat{\alpha}_{ij} + (s_{ij}(l) d_{ij}^s - \tau) \hat{\beta}_{ij}] \\ &\quad + \sum_{m=1}^{n_p} [(x_m - \overline{x}_m) \hat{\psi}_m - (x_m - \underline{x}_m) \hat{\gamma}_m]. \end{aligned}$$

Therefore, the DFCP is formulated as

$$\mathbf{DFCP}: \max_{\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\psi}} \succeq 0} \Lambda(\boldsymbol{S}(l), \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\psi}})$$

$$\text{s.t.} \begin{cases} \hat{\alpha}_{ij} + \hat{\beta}_{ij} + \sum_{m=1}^{n_p} u_j \tilde{b}_{mi} (\hat{\psi}_m - \hat{\gamma}_m) \geq 0 \, \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ 1 - \hat{\alpha}_{ij} \geq 0, 1 - \hat{\beta}_{ij} \geq 0 \quad \forall i \in \mathcal{I}, \forall j \in \mathcal{J} \\ 1 - \hat{\gamma}_m \geq 0, 1 - \hat{\psi}_m \geq 0 \quad \forall m \in \mathcal{M}. \end{cases} \tag{20}$$

Denote $\boldsymbol{\xi}(l)$ and $(\hat{\boldsymbol{\alpha}}(l), \hat{\boldsymbol{\beta}}(l), \hat{\boldsymbol{\gamma}}(l), \hat{\boldsymbol{\psi}}(l))$ as the solutions of the FCP and the DFCP at the lth iteration, respectively. If the SP is infeasible, the related relax variables are nonzero, while the others are zero. Hence, we have $\Theta(\boldsymbol{\xi}(l)) > 0$. Since the FCP is an LP, the strong duality is guaranteed between the FCP and its dual problem, i.e., $\Theta(\boldsymbol{\xi}(l)) = \Lambda(\boldsymbol{S}(l), \hat{\boldsymbol{\alpha}}(l), \hat{\boldsymbol{\beta}}(l), \hat{\boldsymbol{\gamma}}(l), \hat{\boldsymbol{\psi}}(l)) > 0$. Therefore, the infeasible solution $\boldsymbol{S}(l)$ is excluded by the IC: $0 \geq \Lambda(\boldsymbol{S}, \hat{\boldsymbol{\alpha}}(l), \hat{\boldsymbol{\beta}}(l), \hat{\boldsymbol{\gamma}}(l), \hat{\boldsymbol{\psi}}(l))$. The reason why we construct the IC by the solution of the DFCP rather than the solution of the FCP is that $\Lambda(\boldsymbol{S}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\psi}})$ is a function with respect to the binary variables $\boldsymbol{S}$ but not $\Theta(\boldsymbol{\xi})$, i.e., $0 \geq \Theta(\boldsymbol{\xi}(l))$ is an invalid constraint for the MP.

Since the nonoptimal values of the binary variables $\boldsymbol{S}$ found by previous l iterations have been excluded, and, as l increases, more constraints are added into the MP (i.e., the feasible region of the MP will shrink), $\Phi_L(l+1)$ is larger than the previous lower bounds $\{\Phi_L(0), \ldots, \Phi_L(l)\}$. On the other hand, since the upper bound is updated by $\Phi_U(l) = \min\{\Phi_U(l-1), \Gamma(\boldsymbol{S}(l), \boldsymbol{\alpha}(l), \boldsymbol{\beta}(l), \boldsymbol{\gamma}(l), \boldsymbol{\psi}(l))\}$, $\Phi_U(l+1)$ is smaller than the previous upper bounds $\{\Phi_U(0), \ldots, \Phi_U(l)\}$. According to the characteristics of these two sequences, the global optimal solution of P2 is guaranteed when $|\Phi_U(l) - \Phi_L(l)| \leq \varepsilon$. Note that at each iteration, a new constraint is added into the MP to exclude those nonoptimal or infeasible values of the binary variables $\boldsymbol{S}$. In addition, the dimension of the binary variables $\boldsymbol{S}$ is finite. The gap between the lower and upper bounds converges to ε in a finite number of iterations. ■

REFERENCES

[1] Y. Liu, Y. Peng, B. Wang, S. Yao, and Z. Liu, "Review on cyber-physical systems," *IEEE/CAA J. Autom. Sinica*, vol. 4, no. 1, pp. 27–40, 2017.

[2] L. Yeh, C. Lu, C. Kou, Y. Tseng, and C. Yi, "Autonomous light control by wireless sensor and actuator networks," *IEEE Sens. J.*, vol. 10, no. 6, pp. 1029–1041, Jun. 2010.

[3] A. E. D. Mady and G. Provan, "Co-design of wireless sensor-actuator networks for building controls," in *Proc. 50th IEEE Conf. Decis. Control/Eur. Control Conf.*, 2011, pp. 5266–5273.

[4] H. Li, L. Lai, and H. V. Poor, "Multicast routing for decentralized control of cyber physical systems with an application in smart grid," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 6, pp. 1097–1107, Jul. 2012.

[5] C. Wang, J. Li, F. Ye, and Y. Yang, "NETWRAP: An NDN based real-time wireless recharging framework for wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 6, pp. 1283–1297, Jun. 2014.

[6] L. Xie, Y. Shi, T. Hou, W. Lou, H. D. Sherali, and S. F. Midkiff, "Multi-node wireless energy charging in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 437–450, Apr. 2015.

[7] X. Ye and W. Liang, "Charging utility maximization in wireless rechargeable sensor networks," *Wireless Netw.*, vol. 23, pp. 2069–2081, 2017.

[8] J. Jia, J. Chen, Y. Deng, X. Wang, and A.-H. Aghvami, "Joint power charging and routing in wireless rechargeable sensor networks," *Sensors*, vol. 17, no. 10, pp. 1–17, 2017.

[9] J. Chen, X. Cao, P. Cheng, Y. Xiao, and Y. Sun, "Distributed collaborative control for industrial automation with wireless sensor and actuator networks," *IEEE Trans. Ind. Electron.*, vol. 57, no. 12, pp. 4219–4230, Dec. 2010.

[10] X. Cao, J. Chen, Y. Xiao, and Y. Sun, "Building-environment control with wireless sensor and actuator networks: Centralized versus distributed," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3596–3605, Nov. 2010.
[11] K. Ota, M. Dong, Z. Cheng, J. Wang, X. Li, and X. S. Shen, "ORACLE: Mobility control in wireless sensor and actor networks," *Comput. Commun.*, vol. 35, no. 9, pp. 1029–1037, 2012.
[12] L. Mo, X. Cao, Y. Song, and A. Kritikakou, "Distributed node coordination for real-time energy-constrained control in wireless sensor and actuator networks," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 4151–4163, Oct. 2018.
[13] H. Nakayama, Z. M. Fadlullah, N. Ansari, and N. Kato, "A novel scheme for WSAN sink mobility based on clustering and set packing techniques," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2381–2389, Oct. 2011.
[14] T. Melodia, D. Pompili, and I. F. Akyldiz, "Handling mobility in wireless sensor and actor networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 2, pp. 160–173, Feb. 2010.
[15] X. Cao, P. Cheng, J. Chen, and Y. Sun, "An online optimization approach for control and communication co-design in networked cyber-physical systems," *IEEE Trans. Ind. Inform.*, vol. 9, no. 1, pp. 439–450, Feb. 2013.
[16] A. Aminifar, P. Eles, Z. Peng, and A. Cervin, "Control-quality driven design of cyber-physical systems with robustness guarantees," in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2013, pp. 1093–1098.
[17] D. Goswami, R. Schneider, and S. Chakraborty, "Co-design of cyber-physical systems via controllers with flexible delay constraints," in *Proc. 16th Asia South Pacific Des. Autom. Conf.*, 2011, pp. 225–230.
[18] C. Wu, G. S. Tewolde, W. Sheng, B. Xu, and Y. Wang, "Distributed multi-actuator control for workload balancing in wireless sensor and actuator networks," *IEEE Trans. Autom. Control*, vol. 56, no. 10, pp. 2462–2467, Oct. 2011.
[19] K. Genova and V. Guliashki, "Linear integer programming methods and approaches—A survey," *Cybern. and Inf. Technol.*, vol. 11, no. 1, pp. 1–23, 2011.
[20] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numer. Math.*, vol. 4, no. 1, pp. 238–252, 1962.
[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[22] O. Landsiedel, K. Wehrle, and S. Gotz, "Accurate prediction of power consumption in sensor networks," in *Proc. 2nd IEEE Workshop Embedded Netw. Sens.*, 2005, pp. 37–44.
[23] Y. Peng, Z. Li, W. Zhang, and D. Qiao, "Prolonging sensor network lifetime through wireless charging," in *Proc. 31st IEEE Real-Time Syst. Symp.*, 2010, pp. 129–139.
[24] S. Boyd and J. Mattingley, "Branch and bound methods," Notes for EE364b, Stanford Univ., Stanford, CA, USA, 2007, pp. 1–11.
[25] S. Albert, "Solving mixed integer linear programs using branch and cut algorithm," master's thesis, North Carolina State Univ., Raleigh, NC, USA, 1999.
[26] E. Rothberg, "An evolutionary algorithm for polishing mixed integer programming solutions," *INFORMS J. Comput.*, vol. 19, no. 4, pp. 534–541, 2007.
[27] C. Randazzo and H. P. L. Luna, "A comparison of optimal methods for local access uncapacitated network design," *Ann. Oper. Res.*, vol. 106, no. 1, pp. 263–286, 2001.
[28] 2013. [Online]. Available: http://optitrack.com/
[29] K. Yoshizawa, H. Hashimoto, M. Wada, and S. Mori, "Path tracking control of mobile robots using a quadratic curve," in *Proc. Conf. Intell. Veh.*, 1996, pp. 58–63.
[30] 2013. [Online]. Available: http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.ht ml/
[31] K. L. Moore, Y. Chen, and Z. Song, "Diffusion-based path planning in mobile actuator-sensor networks (MAS-Net): Some preliminary results," *Proc. SPIE*, vol. 5421, pp. 58–69, 2004.
[32] M. Marin-Perianu, S. Bosch, R. Marin-Perianu, H. Scholten, and P. Havinga, "Autonomous vehicle coordination with wireless sensor and actuator networks," *ACM Trans. Auton. Adapt. Syst.*, vol. 5, no. 4, pp. 13.1–13.29, 2010.
[33] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proc. 50th ACM/EDAC/IEEE Des. Autom. Conf.*, 2013, pp. 1–10.
[34] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3, 2014, Art. no. 111.
[35] K. Cao, G. Xu, J. Zhou, T. Wei, M. Chen, and S. Hu, "QoS-adaptive approximate real-time computation for mobility-aware IoT lifetime optimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, to be published, doi: 10.1109/TCAD.2018.2873239.

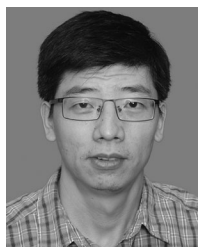
Lei Mo (S'13–M'17) received the B.S. degree from the College of Telecom Engineering and Information Engineering, Lanzhou University of Technology, Lanzhou, China, in 2007, and the Ph.D. degree from the College of Automation Science and Engineering, South China University of Technology, Guangzhou, China, in 2013.

He is currently a Postdoctoral Fellow with the Institut National de Recherche en Informatique et en Automatique (INRIA) Rennes Research Center, University of Rennes, Rennes, France. From 2013 to 2015, he was a Research Fellow with the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China. From 2015 to 2017, he was a Research Fellow with the INRIA Nancy Research Center, France. His current research interests include networked estimation and control in wireless sensor and actuator networks, cyber-physical systems, task mapping, and resource allocation in multicore systems.

Dr. Mo is an Associate Editor for the *KSII Transactions on Internet and Information Systems*, the *Journal of Computer*, and the *Journal of Electrical and Electronic Engineering*. He is a Guest Editor for the IEEE ACCESS and the *Journal of Computer Networks and Communications* and a technical program committee member for several international conferences.

Pengcheng You (S'14) received the B.S. (Hons.) degree in electrical engineering and the Ph.D. degree in control from Zhejiang University, Hangzhou, China, in 2013 and 2018, respectively.

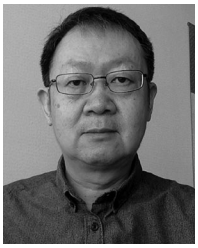
He is currently a Postdoctoral Fellow with the Whiting School of Engineering, Johns Hopkins University, Baltimore, MD, USA. Previously, he was a Visiting Student with the California Institute of Technology, Pasadena, CA, USA, and also a Research Intern with the Pacific Northwest National Laboratory, Richland, WA, USA. His research interests include smart grids, especially electric vehicles and power market.



Xianghui Cao (S'08–M'11–SM'16) received the B.S. and Ph.D. degrees in control science and engineering from Zhejiang University, Hangzhou, China, in 2006 and 2011, respectively.

From 2012 to 2015, he was a Senior Research Associate with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. He is currently an Associate Professor with the School of Automation, Southeast University, Nanjing, China. His current research interests include cyber-physical systems, wireless network performance analysis, wireless networked control, and network security.

Dr. Cao served as the Organization Chair for The Youth Academic Annual Conference of Chinese Association of Automation in 2018, the Symposium Co-Chair for the 2017 International Conference on Computing, Networking and Communications, and the Publicity Co-Chair for the 2015 ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc) and the 2015 IEEE/CIC International Conference on Communications in China. He was a recipient of the Best Paper Runner-Up Award from ACM MobiHoc in 2014 and the First Prize of Natural Science Award of Ministry of Education of China in 2017. He also serves as an Associate Editor for *ACTA Automatica Sinica* and the IEEE/CAA JOURNAL OF AUTOMATICA SINICA.



Yeqiong Song (M'15) received the B.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 1984, the master's degree from the Ecole National Supérieure des Télécommuications, Paris, France, in 1987, the M.Sc. degree (DEA) from the University of Paris 6, Paris, in 1988, the Ph.D. degree from the Institut National Polytechnique de Lorraine (INPL), Nancy, France, in 1991, and the Habilitation to Lead Research degree from the University of Henri Poincaré Nancy 1, Nancy, in 2004, all in telecommunications and computer science.

In 1988, he joined the Lorraine Laboratory of IT Research and Applications, Nancy, France. From 1992 to 2005, he was an Associate Professor with the University of Henri Poincaré Nancy 1. From 2001 to 2003, he was a Full-Time Researcher with INRIA Lorraine. Since 2005, he has been a full Professor of Computer Science with the INPL and now with the University of Lorraine, Nancy. He authored/coauthored more than 100 scientific papers. His research interests include modeling and performance evaluation of networks and real-time distributed systems including networked control systems, and the implementation of real-time quality-of-service mechanisms in industrial networks, in-vehicle networks, and wireless Internet of Things and sensor networks.

Angeliki Kritikakou received the Ph.D. degree in electrical and computer engineering from the University of Patras, Patras, Greece, in collaboration with the IMEC Research Center, Leuven, Belgium, in 2013.

She is currently an Associate Professor with the Institut de Recherche en Informatique et Syst`mes Aléatoires—INRIA Rennes Research Center, University of Rennes 1, Rennes, France. She worked for one year as a Postdoctoral Research Fellow with the Department of Modelling and Information Processing, ONERA, in collaboration with the Laboratory of Analysis and Architecture of Systems and the University of Toulouse, Toulouse, France. Her research interests include embedded systems, real-time systems, mixed-critical systems, hardware/software codesign, mapping methodologies, design space exploration methodologies, memory management methodologies, low-power design and fault tolerance.