Actuator Selection for Cyber-Physical Systems

Ahmad F. Taha[†], Nikolaos Gatsis[†], Tyler Summers[‡], and Sebastian Nugroho[†]
[†]Department of Electrical and Computer Engineering, The University of Texas at San Antonio
[‡]Department of Mechanical Engineering, University of Texas at Dallas
Emails: {ahmad.taha, nikolaos.gatsis, lzg572}@utsa.edu, tyler.summers@utdallas.edu

***Abstract*—In cyber-physical systems (CPS), the problem of controlling resources can be depicted as an actuator selection problem. Given a large library of actuators and a control objective, what is the least number of actuators to be selected, and what is the corresponding optimal control law? These dynamic design questions are inherently coupled. In this paper, we show that a breadth of actuator selection and optimal control problems (stabilizability, robust and LQR control routines, control of uncertain, nonlinear systems) that do not satisfy the submodularity property lead to the formulation of two classes of combinatorial optimization routines for unstable CPSs: mixed-integer semidefinite programs and mixed-integer bilinear matrix inequalities. Branch-and-bound and greedy algorithms are proposed to address the computational complexity, and numerical results are given to illustrate the proposed formulations.**

***Index Terms*—Actuator selection, cyber-physical systems, linear matrix inequalities, controller design, greedy algorithms.**

I. Introduction & Brief Literature Review

The focus of this paper is on the actuator selection problem in CPSs: Given a library of actuators and a qualitative/quantitative control objective, what is the least number of actuators to be selected, and what is the corresponding optimal control law? Enabling more actuators often results in a higher operational cost. Hence, minimizing the number of actuators while maintaining stability and generating optimal control laws can prove to be superior. The two design questions (selection of a set of actuators and the design of a control law given this selection) are inherently coupled, as we demonstrate through this paper.

To quantify controllability of CPSs and dynamic systems, different qualitative and quantitative methods have been developed, especially when the selection of actuators and sensors is considered; see survey paper [1] and the following references [2]–[4]. The reader is referred to [5], [6] for a brief, yet thoroughly informative summary of the most recent research results on actuator selection for dynamic systems.

The actuator selection problem is a combinatorial problem, as the choice is to either activate/deactivate a subset of actuators at a given time to achieve a certain controllability or stabilizability metric. Unfortunately, little is understood about these classes of combinatorial routines [5]. A simple approach to solve these problems is via brute force: test all the combinations of actuators to be used and choose the selection which either satisfies a control energy minimization functional (such as the controllability Gramian) or a stabilizability/controllability metric. Obviously, this brute force approach is infeasible when the number of control nodes is large. Sub-optimal greedy algorithms have been developed to solve this combinatorial challenge [7].

Moreover, the authors in [5] prove that if the metric for the actuator selection is based on the controllability Gramian, which is shown to satisfy the submodular property, then efficient optimization can be implemented to solve the aforementioned combinatorial optimization. Specifically, the authors show that the mapping from possible placements to the trace of the associated Gramian is a modular set function [5]. This implies that a simple optimization can be implemented which computes the metric individually for all possible actuator placement combination, sorts the outcome, and selects the best combination that minimizes the Gramian energy metric [5], [6]. This important result proves to be superior to solve the actuator selection problem when compared with other algorithms. Other controllability Gramian metrics are proved to be submodular [8]. In addition, many other problems featuring supermodularity or submodularity are discussed in [2], [9], [10].

The approaches presented in [5] assume that the system is initially stable with strict applications to linear, time-invariant systems with the controllability Gramian and its derivatives as the control metric. Other formulations also assume a rather limited control objective and dynamical system representation [2], [8], [10]. If the system dynamics are assumed to be unstable, nonlinear or perturbed, the control metric is different from the Gramian (i.e., classical optimal control metrics such as the LQR), and the derivation of a stabilizing control law is simultaneously required with actuator selection, the set function will no longer satisfy the submodular property—as illustrated in [6]. For example, the authors in [6], [11] prove that classical optimal control and estimation routines do not satisfy the submodularity or supermodularity properties—which are essential in reducing the computational burden in the actuator selection problem, as discussed above. Since these properties are not satisfied for generic dynamical systems, the aim of this paper is to address this research challenge via a framework that can be inclusive to a wide range of optimal control formulations while optimizing controllability metrics by optimally selecting actuators and designing control laws for unstable CPSs.

This material is based upon work supported by the Smart Grid Security Research Grant from the UTSA Office of the Vice President of Research.

II. Problem Description & Paper Structure

CPS Model and Problem Formulation In this paper, we consider a general class of nonlinear, interconnected nodes corresponding to a dynamic CPS which can be modeled as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_u\boldsymbol{u}(t) + \boldsymbol{B}_w\boldsymbol{w}(t) + \boldsymbol{\phi}(\boldsymbol{x}). \tag{1}$$

Nodes are connected via a network; suppose that there are N nodes in the network the set $\mathcal{V} = \{1, \dots, N\}$ defines the set of nodes. The global state $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ consists of each nodal agent's states $\boldsymbol{x}_i \in \mathbb{R}^{n_{x_i}}$, $i = 1, \dots, N$; $\boldsymbol{x} = [\boldsymbol{x}_1^\top, \dots, \boldsymbol{x}_N^\top]^\top$. The dynamics of the nodes in a CPS are assumed to be coupled through the state-vector evolution (1). Likewise, each nodal agent has a set of available inputs $\boldsymbol{u}_i \in \mathbb{R}^{n_{u_i}}$. The mapping from the input to state vector can thus be written in the form $\boldsymbol{B}_u\boldsymbol{u} = \text{blkdiag}(\boldsymbol{B}_{u_1}, \dots, \boldsymbol{B}_{u_N})[\boldsymbol{u}_1^\top, \dots, \boldsymbol{u}_N^\top]^\top$.The system nonlinearity can be expressed in terms of the nodal agent nonlinearities as $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{n_x}$. In summary, the system has $n_x = \sum_{i=1}^N n_{x_i}$ states, $n_u = \sum_{i=1}^N n_{u_i}$ control inputs, and a maximum of n_w unknown inputs (UI).

The objective of the methods presented in this paper is two-fold, namely, to model the actuator selection problem and to propose scalable optimization methods to solve this problem in large-scale CPSs. A typical CPS comprises a very large number of networked dynamical systems. In order to formally state the actuator selection problem, define binary variables π_i, $i = 1, \dots, N$, where $\pi_i = 1$ if the actuator of the i-th nodal agent is selected, and 0 otherwise. Variables π_i are organized in a vector $\boldsymbol{\pi}$. It is supposed for now that the selection of actuators remains constant over the period starting with the system initialization until the steady state is reached. This assumption is not restrictive as a multi-period actuator selection and optimal control problem can be formulated. This formulation can incorporate the changes in the state-space parameters of the system, and the varying selection of actuators.

Another aim of the paper is to simultaneously obtain an optimal control law $\boldsymbol{u}^*(t)$ and an actuator selection $\boldsymbol{\pi}^*$ from a given, fixed library of actuators to optimize a control metric.

Leveraging SDP Formulations The common instrument that we use to mathematically formulate the array of problems within the previously outlined methods is matrix inequalities and semidefinite programs (SDPs). The formulations in this paper are building upon semidefinite programming formulations for robust control and stabilizability routines. This idea is natural: SDP solvers have become efficient and robust over the past 10–20 years.

The second major reason that makes SDP formulations of optimal control for CPSs lucrative is that SDPs facilitate the inclusion of operational and logistic constraints on the control law or the generated gain matrices. To set the stage, we first succinctly list control and stabilizability formulations in terms of LMIs in Table I. For each entry, the system dynamics, the controller form, the optimization variables, and the optimization problem are stated. The listed formulations are instrumental in formalizing the actuator selection problem.

Paper Structure Section III introduces a formulation for the actuator selection of an LTI system. Section IV investigates the selection problem for other classes of dynamic systems resulting in complex combinatorial problem mixed integer bilinear matrix inequality constraints. To address this computational complexity, we propose a branch-and-bound algorithm in Section IV and a greedy, sub-optimal algorithms in Section V. Section VI presents numerical results.

III. Actuator Selection For Stabilizability

Here, we focus on the simplest stabilizability problem listed in the first row of Table I. With the introduction of the actuator selection variables, the system dynamics evolve as follows:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_u\boldsymbol{\Pi}\boldsymbol{u}(t), \tag{2}$$

where $\boldsymbol{\Pi} = \text{blkdiag}(\pi_1\boldsymbol{I}_{n_{u_1}}, \dots, \pi_N\boldsymbol{I}_{n_{u_N}})$ is placing vector $\boldsymbol{\pi}$ in a block diagonal matrix of appropriate dimensions (variables $\boldsymbol{\Pi}$ and $\boldsymbol{\pi}$ are thus used interchangeably). Following the first entry in Table I, stabilizability is ensured if the matrix inequality[1]

$$\boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top \preceq \boldsymbol{B}_u\boldsymbol{\Pi}\boldsymbol{\Pi}\boldsymbol{B}_u^\top$$

is solved in the variables $\boldsymbol{S}$ and $\boldsymbol{\pi}$. Note that the stabilizing state feedback controller can be obtained as follows: $\boldsymbol{u}(t) = \boldsymbol{K}\boldsymbol{x}(t) = -\frac{1}{2}\boldsymbol{B}_u^\top\boldsymbol{S}^{-1}\boldsymbol{x}(t)$. In practice it may be desirable to minimize the number of active actuators. Noticing that $\boldsymbol{\Pi}^2 = \boldsymbol{\Pi}$, we obtain:

$$\begin{aligned}
\underset{\boldsymbol{S},\boldsymbol{\Pi}}{\text{minimize}} \quad & \sum_{i=1}^N \pi_i && \text{(3a)}\\
\text{subject to} \quad & \boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top \preceq \boldsymbol{B}_u\boldsymbol{\Pi}\boldsymbol{B}_u^\top && \text{(3b)}\\
& \boldsymbol{S} = \boldsymbol{S}^\top \succ \boldsymbol{O};\ \boldsymbol{\pi} \in \{0,1\}^N. && \text{(3c)}
\end{aligned}$$

The previous problem is a mixed-integer SDP (MISDP). Specifically, the problem is an SDP *jointly* in the variables $\boldsymbol{S}$ and $\boldsymbol{\Pi}$, if the integrality constraints ($\pi_i \in \{0, 1\}$) are ignored. The widely used convex optimization modeling toolboxes CVX and YALMIP have been expanded to incorporate modeling of mixed-integer convex programs [16], [17] and interface corresponding general-purpose SDP solvers combined with implementations of branch-and-bound (BB) methods. The BB method is essentially a smart and efficient way to run an exhaustive search over all possible 2^N combinations of the binary variables π_i's. Notice that if the π_i's are fixed, the resulting problem is an SDP. At most 2^N SDPs are then solved in the worst case run of a BB method. Nevertheless, the empirical complexity of BB algorithms is much smaller than the worst-case one, and thus, they are among the state-of-the-art in solving mixed-integer convex programs.

A BB algorithm can find the global solution of (3), but may require an impractically long time. This motivates the development of approximations or suboptimal methods to solve (3). One such immediate possibility is to relax the binary constraint $\pi_i \in \{0, 1\}$ to $0 \leq \pi_i \leq 1$, which yields an

[1]Originally, the stabilizability LMI is written as $\boldsymbol{A}\boldsymbol{S}+\boldsymbol{S}\boldsymbol{A}^\top \preceq \sigma\boldsymbol{B}_u\boldsymbol{B}_u^\top$ where $\sigma > 0$ is a constant variable to be solved for simultaneously with $\boldsymbol{S}$. However, it is customary to assume that $\sigma = 1$ as discussed in [14, Ch.7.2].

TABLE I
THE TABLE SHOWS HOW DESIGN OF CONTROLLERS FOR VARIOUS SYSTEMS CAN BE PERFORMED VIA SDP FORMULATIONS [12]–[15]. MANY OTHER FORMULATIONS FOLLOW SIMILAR STRUCTURE, AND ARE OMITTED FOR BREVITY.

CPS Dynamics & Design Objective	Control Design via SDPs
Stabilizability $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_u\boldsymbol{u}$ $\boldsymbol{u} = -\frac{1}{2}\sigma\boldsymbol{B}_u^\top\boldsymbol{S}^{-1}\boldsymbol{x}$ Variables: $\boldsymbol{S}, \sigma$ (can assume $\sigma = 1$ [14])	$\boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top \preceq \sigma\boldsymbol{B}_u\boldsymbol{B}_u^\top$
Robust control of perturbed systems $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_u\boldsymbol{u} + \boldsymbol{B}_w\boldsymbol{w}$ $\boldsymbol{z} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}_w\boldsymbol{w}$ $\boldsymbol{u} = -\boldsymbol{K}x = -\boldsymbol{Z}\boldsymbol{S}^{-1}\boldsymbol{x}$ Variables: $\boldsymbol{Z}, \boldsymbol{S}, \zeta$	$\min\ \zeta$ s.t. $\begin{bmatrix} \boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top - \boldsymbol{B}_u\boldsymbol{Z} - \boldsymbol{Z}^\top\boldsymbol{B}_u^\top + 2\alpha\boldsymbol{S} & \boldsymbol{B}_w \\ \boldsymbol{B}_w^\top & -2\alpha\boldsymbol{I} \end{bmatrix} \preceq \boldsymbol{O}$ $\begin{bmatrix} -\boldsymbol{S} & \boldsymbol{O} & \boldsymbol{S}\boldsymbol{C}^\top \\ \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{D}_w^\top \\ \boldsymbol{C}\boldsymbol{S} & \boldsymbol{D}_w & -\zeta\boldsymbol{I} \end{bmatrix} \preceq \boldsymbol{O}$
LQR Control—minimal cost $\min\ \int_{t_0}^{\infty} \boldsymbol{x}(\tau)\boldsymbol{Q}\boldsymbol{x}(\tau) + \boldsymbol{u}(\tau)\boldsymbol{R}\boldsymbol{u}(\tau)d\tau$ s.t. $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_u\boldsymbol{u}$ $\boldsymbol{u} = -\boldsymbol{R}^{-1}\boldsymbol{B}_u^\top\boldsymbol{S}^{-1}\boldsymbol{x}$ Variables: $\boldsymbol{Y}, \boldsymbol{S}$	$\min\ \ \mathbf{trace}(\boldsymbol{S}^{-1})$ s.t. $\begin{bmatrix} \boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top + \boldsymbol{B}_u\boldsymbol{Y} + \boldsymbol{Y}^\top\boldsymbol{B}_u^\top & \boldsymbol{S} & \boldsymbol{Y} \\ \boldsymbol{S} & -\boldsymbol{Q}^{-1} & 0 \\ \boldsymbol{Y}^\top & 0 & -\boldsymbol{R}^{-1} \end{bmatrix} \preceq \boldsymbol{O}$
Stabilizability of nonlinear systems $\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}_u\boldsymbol{u} + \boldsymbol{f}(\boldsymbol{x})$ $\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x}$ $\boldsymbol{u} = \boldsymbol{K}\boldsymbol{x} = \boldsymbol{Z}\boldsymbol{S}^{-1}\boldsymbol{x}$ Variables: $\boldsymbol{Z}, \boldsymbol{S}, \zeta, \kappa_S, \kappa_Z$ $\boldsymbol{f}(\boldsymbol{x})$ bounded by: $\boldsymbol{f}^\top(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}) \le \alpha^2\boldsymbol{x}^\top\boldsymbol{F}^\top\boldsymbol{F}\boldsymbol{x}$	$\min\ \zeta + \kappa_S + \kappa_Z$ s.t. $\begin{bmatrix} \boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top + \boldsymbol{B}_u\boldsymbol{Z} + \boldsymbol{Z}^\top\boldsymbol{B}_u^\top & \boldsymbol{I} & \boldsymbol{S}\boldsymbol{F}^\top \\ \boldsymbol{I} & -\boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{F}\boldsymbol{S} & \boldsymbol{O} & -\zeta\boldsymbol{I} \end{bmatrix} \prec \boldsymbol{O}$ $\begin{bmatrix} -\kappa_Z\boldsymbol{I} & \boldsymbol{Z}^\top \\ \boldsymbol{Z} & -\boldsymbol{I} \end{bmatrix} \prec \boldsymbol{O},\ \begin{bmatrix} \boldsymbol{Z} & \boldsymbol{I} \\ \boldsymbol{I} & \kappa_S\boldsymbol{I} \end{bmatrix} \succ \boldsymbol{O},\ \zeta - \frac{1}{\bar{\alpha}^2} < 0$

SDP. The number of actuated systems would then be given by rounding the optimal value $\sum_{i=1}^N \pi_i^*$ up to the nearest integer, say $\tilde{N}$, and picking the actuators corresponding to the $\tilde{N}$ highest values of π_i^*'s. If this procedure does not guarantee the feasibility of (3b), then additional actuators must be added according to their ranking in $\boldsymbol{\pi}$.

An alternative approach is to use greedy algorithms, which are the theme of Section V. The next section deals with control objectives beyond stabilizability and demonstrates that one needs to work in a more general optimization class than MISDPs.

IV. ACTUATOR SELECTION BEYOND STABILIZABILITY

The optimization complexity increases significantly for all the remaining controllers of Table I, and necessitates the development of advanced optimization algorithms with capabilities beyond available general-purpose solvers. We consider the controller design for the second system in Table I which includes UIs that are inevitably present in modern large-scale CPSs. The actuator selection variables are defined as before. The system dynamics are given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}_u\boldsymbol{\Pi}\boldsymbol{u}(t) + \boldsymbol{B}_w\boldsymbol{w}(t), \tag{4a}$$
$$\boldsymbol{z}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}_w\boldsymbol{w}(t), \tag{4b}$$

where $\boldsymbol{\Pi} = \text{blkdiag}(\pi_1\boldsymbol{I}_{n_{u_1}}, \ldots, \pi_N\boldsymbol{I}_{n_{u_N}})$. Stabilizability under UIs can be obtained from the solution of the following optimization problem with variables $\boldsymbol{S}$, $\boldsymbol{Z}$, $\zeta \in \mathbb{R}$, and $\boldsymbol{\pi}$:

$$\underset{\boldsymbol{S},\boldsymbol{Z},\zeta,\boldsymbol{\pi}}{\text{minimize}} \qquad \zeta + \beta\sum_{i=1}^N \pi_i \tag{5a}$$

$$\text{subject to} \qquad \begin{bmatrix} \boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top + 2\alpha\boldsymbol{S} & \\ -\boldsymbol{B}_u\boldsymbol{\Pi}\boldsymbol{Z} - \boldsymbol{Z}^\top\boldsymbol{\Pi}\boldsymbol{B}_u^\top & \boldsymbol{B}_w \\ \boldsymbol{B}_w^\top & -2\alpha\boldsymbol{I} \end{bmatrix} \preceq \boldsymbol{O} \tag{5b}$$

$$\begin{bmatrix} -\boldsymbol{S} & \boldsymbol{O} & \boldsymbol{S}\boldsymbol{C}^\top \\ \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{D}_w^\top \\ \boldsymbol{C}\boldsymbol{S} & \boldsymbol{D}_w & -\zeta\boldsymbol{I} \end{bmatrix} \preceq \boldsymbol{O} \tag{5c}$$

$$\boldsymbol{S} = \boldsymbol{S}^\top \succ \boldsymbol{O};\ \boldsymbol{\pi} \in \mathcal{P}. \tag{5d}$$

The constant β is the relative weight between the objectives of minimizing ζ and the number of selected actuators. A particular advantage of this formulation is that it can accommodate constraints on the actuator selection through the set constraint $\{\pi_i\}_{i=1}^N \in \mathcal{P} \subseteq \{0,1\}^N$. For instance, regulatory constraints may require certain actuators to be always on. Likewise, system operators may wish to impose empirically determined constraints, e.g., "always activate actuator x if you don't activate actuator y."

The previous optimization problem includes a mixed-integer bilinear matrix inequality (MIBMI) due to the term $\boldsymbol{\Pi}\boldsymbol{Z}$. The bilinearity stems from the fact that the matrix inequality is linear in $\boldsymbol{Z}$ (with $\boldsymbol{\Pi}$ fixed) and vice-versa, but not linear in both optimization variables [18]. The bilinearity together with the integrality constraints bring about the need for specialized optimization methods. It should be emphasized that (5) is *not* a mixed-integer convex program. For this to be the case, the problem must be convex in all optimization variables if the integrality constraints are ignored; this condition is not satisfied due to the bilinearity. Therefore, general-purpose mixed-integer convex programming solvers are not applicable.

Interestingly, the design of the remaining controllers in

Table I largely shares the optimization complexity of problem (5). It can be easily observed that *all* design problems from Row 2 onwards feature a MIBMI with the form $\boldsymbol{B}_u\boldsymbol{\Pi}\boldsymbol{Z} + \boldsymbol{Z}^\top\boldsymbol{\Pi}\boldsymbol{B}_u^\top$ or a very similar one. Using (5) as a model, custom optimization algorithms are proposed next.

A. Branch-and-Bound Algorithm

Here, we describe a custom basic branch-and-bound (BB) algorithm to solve (5). It is worth noting that general-purpose BB algorithms as the one in e.g. [18] are not applicable here, because they need compact regions where the variables $\boldsymbol{S}$ and $\boldsymbol{Z}$ lie in—this is not doable here. An essential subroutine of the BB algorithm is a method that computes a lower bound for the optimal value of (5). We present such a method next.

1) Lower Bound: Recall that the dimensions of $\boldsymbol{u}(t)$ and $\boldsymbol{x}(t)$ are n_u and n_x, respectively; notice that $\boldsymbol{Z} \in \mathbb{R}^{n_u \times n_x}$. First, change the variables π_i, $i = 1, \dots, N$ to a vector $\boldsymbol{\rho} \in \mathbb{R}^{n_u}$ such that $\mathrm{diag}(\boldsymbol{\rho}) = \boldsymbol{\Pi}$. Clearly, equalities must be enforced among entries of $\boldsymbol{\rho}$ (e.g., the first n_{u_1} entries must be equal to each other); these equality constraints are summarized in $\boldsymbol{H}\boldsymbol{\rho} = \boldsymbol{0}$. Introduce further a basis on the space of $n_u \times n_x$ matrices defined by matrices $\boldsymbol{E}_{jk}$ with a single entry 1 at position (j,k), $(j \in \{1,\dots,n_u\}, k \in \{1,\dots,n_x\})$, and 0 otherwise. Given this construction, we can write

$$\boldsymbol{B}_u\boldsymbol{\Pi}\boldsymbol{Z} - \boldsymbol{Z}^\top\boldsymbol{\Pi}\boldsymbol{B}_u^\top = \sum_{j=1}^{n_u}\sum_{k=1}^{n_x} \rho_j Z_{jk}[\boldsymbol{B}_u\boldsymbol{E}_{jk} + \boldsymbol{E}_{jk}^\top\boldsymbol{B}_u^\top].$$

Now, making a change of variables

$$\boldsymbol{q} = [\boldsymbol{\rho}^\top, \mathrm{vec}(\boldsymbol{Z})^\top]^\top \in \mathbb{R}^{n_u + n_u n_w},$$

and denoting the first n_u entries of $\boldsymbol{q}$ as $\boldsymbol{q}_{1:n_u}$. The previous summation can thus be written as $\sum_{(l,m)\in\mathcal{I}} q_l q_m F_{lm}$, where $\mathcal{I}$ is an appropriate set of indices and F_{lm} are all known matrices given by $\boldsymbol{B}_u\boldsymbol{E}_{jk} + \boldsymbol{E}_{jk}^\top\boldsymbol{B}_u^\top$. Problem (5) becomes equivalent to an optimization problem with $\boldsymbol{S}$, $\boldsymbol{q}$, and ζ as variables—as shown in (6).

$$\underset{\boldsymbol{S},\boldsymbol{q},\zeta}{\text{minimize}} \quad \zeta + \beta \sum_{i\in\{1,1+n_{u_1},\dots,1+n_{u_{N-1}}\}} q_i \tag{6a}$$

subject to

$$\begin{bmatrix} \boldsymbol{A}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{A}^\top + 2\alpha\boldsymbol{S} & \\ -\sum_{(l,m)\in\mathcal{I}} q_l q_m F_{lm} & \boldsymbol{B}_w \\ \boldsymbol{B}_w^\top & -2\alpha\boldsymbol{I} \end{bmatrix} \preceq \boldsymbol{O} \tag{6b}$$

$$\begin{bmatrix} -\boldsymbol{S} & \boldsymbol{O} & \boldsymbol{S}\boldsymbol{C}^\top \\ \boldsymbol{O} & -\boldsymbol{I} & \boldsymbol{D}_w^\top \\ \boldsymbol{C}\boldsymbol{S} & \boldsymbol{D}_w & -\zeta\boldsymbol{I} \end{bmatrix} \preceq \boldsymbol{O} \tag{6c}$$

$$\boldsymbol{S} = \boldsymbol{S}^\top \succ \boldsymbol{O};\ \boldsymbol{H}\boldsymbol{q}_{1:n_u} = \boldsymbol{0};\ \boldsymbol{q}_{1:n_u} \in \{0,1\}^{n_u}. \tag{6d}$$

Notice that (6b) is still a BMI due to the presence of the products $q_l q_m$. The previous manipulations enable us to develop an SDP relaxation of the BMI [19]. Specifically, introduce additional optimization variables $v_{lm} = q_l q_m$ $(l, m = 1, \dots, n_u + n_u n_w)$, which are organized in matrix $\boldsymbol{V}$. The equalities $v_{lm} = q_l q_m$ are equivalent to

$$\begin{bmatrix} \boldsymbol{V} & \boldsymbol{q} \\ \boldsymbol{q}^\top & 1 \end{bmatrix} \succeq \boldsymbol{O},\ \mathrm{rank} \begin{bmatrix} \boldsymbol{V} & \boldsymbol{q} \\ \boldsymbol{q}^\top & 1 \end{bmatrix} = 1.$$

By dropping the rank constraint and relaxing the binary variables $(q_{1:m})$ to the interval $[0,1]$, a lower bound on the optimal value of (5) is obtained.

2) Branching: Returning to the BB algorithm, and after a lower bound on the optimal value of (5) is obtained as described, an upper bound can be obtained by fixing arbitrarily the binary variables π_i, and solving the resulting SDP. Define as L and U the previous bounds. The BB algorithm forms a tree corresponding to the choices for π_i.

First, a random index $i \in \{1, \dots, N\}$ is selected. Then, the lower and upper bounds on (5) with $\pi_i = 0$ and $\pi_i = 1$ are computed, using the methods previously described; this yields two children nodes on the tree. A node on the tree whose lower bound is greater than U is pruned, and the corresponding value of π_i is not further considered. The procedure continues by further branching. If the problem is convex, this procedure is guaranteed to result in a node where $L = U$, and then, global optimality is guaranteed. But for the problem at hand there is no guarantee that $L = U$ at any point, because the problem involves a MIBMI which is not convex. In particular, there is no guarantee that the lower bounding process described previously results will yield zero duality gap. The overall solution in this case is suboptimal.

3) Improvements: The previous BB algorithm can be improved in several fronts. First, it is not hard to see that the number of binary variables was increased here with the introduction of variable $\boldsymbol{\rho}$ and the constraints $\boldsymbol{H}\boldsymbol{\rho} = \boldsymbol{0}$ in order to clarify the exposition. A more economical representation of (6) is thus possible. Second, the branching process was based on random selection of the index $i = 1, \dots, N$; more elaborate methods that are guided by the value of the optimized relaxed π_i are possible. Finally, a tighter lower bounding procedure benefits the practical convergence properties of the algorithm, based on the discussion about sub-optimality in Section IV-A2. Note that the proposed BB algorithm can seamlessly be applied to other SDP formulations as in Table I.

V. Actuation Selection via Suboptimal Routines

An alternative to the mixed-integer optimization approach described above is combinatorial greedy algorithms. Greedy algorithms for actuator selection are studied in [20] for Gramian-based metrics and in [6] for optimal feedback control performance metrics in discrete time. We briefly summarize and extend this approach. By focusing on an LQR optimal control metric for concreteness, though many other metrics are possible.

The actuator selection problem can be described as a set function optimization problem. For any given $\boldsymbol{\pi}$, let $\Gamma = \{i \in \{1, ..., N\} \mid \pi_i = 1\}$; note that $\Gamma \subseteq \{1, ..., N\}$. Let $\boldsymbol{B}_{u,\Gamma} = \boldsymbol{B}_u\boldsymbol{\Pi}$, the input matrix associated with the selection Γ, and $\boldsymbol{R}_\Gamma$ the input cost submatrix of rows and columns associated with Γ. Let $\boldsymbol{X}_\Gamma$ be the solution to the algebraic Riccati equation

$$\boldsymbol{A}^T\boldsymbol{X}_\Gamma + \boldsymbol{X}_\Gamma\boldsymbol{A} - \boldsymbol{X}_\Gamma\boldsymbol{B}_{u,\Gamma}\boldsymbol{R}_\Gamma\boldsymbol{B}_{u,\Gamma}^T\boldsymbol{X}_\Gamma + \boldsymbol{Q} = 0. \tag{7}$$

Let $f : 2^{\{1,\dots,N\}} \to \mathbb{R}$ be the set function that maps a given actuator selection to the optimal LQR cost that it achieves,

i.e., $f(\Gamma) = \mathbf{trace}(\boldsymbol{X}_\Gamma)$. Then the actuator selection problem of selecting k actuators to optimize LQR performance is the cardinality constrained set function optimization problem

$$\underset{\Gamma\subset\{1,\ldots,N\}}{\text{minimize}} \quad \mathbf{trace}(\boldsymbol{X}_\Gamma); \quad \text{subject to} \quad |\Gamma| = k. \tag{8}$$

A common heuristic for solving this problem is a greedy algorithm; see Algorithm 1, which simply adds at each step the actuator that provides the best marginal LQR performance given the actuators which have already been added.

Algorithm 1 A greedy heuristic for set function optimization.

$\Gamma \leftarrow \emptyset$
while $|\Gamma| \leq k$ **do**
  $e^\star = \underset{e\in\mathcal{V}\setminus\Gamma}{\text{argmin}} \quad f(\Gamma\cup\{e\})$
  $\Gamma \leftarrow \Gamma \cup \{e^\star\}$
end while
$\Gamma^\star \leftarrow \Gamma$

However, when $\boldsymbol{A}$ is unstable, it is possible that small actuator subsets may fail to provide closed-loop stability, in which case the optimal value of (8) may be infinite for each actuator in an iteration. In this case, one can instead work with an alternative Riccati equation involving $\boldsymbol{X}_\Gamma^{-1}$. Multiplying (7) on the left and right by $\boldsymbol{X}_\Gamma^{-1}$ and then by -1, while setting $\boldsymbol{X}_\Gamma^{-1} = \boldsymbol{P}_\Gamma$ yields the Riccati equation

$$-\boldsymbol{P}_\Gamma\boldsymbol{A}^\top - \boldsymbol{A}\boldsymbol{P}_\Gamma - \boldsymbol{P}_\Gamma\boldsymbol{Q}\boldsymbol{P}_\Gamma + \boldsymbol{B}_{u,\Gamma}\boldsymbol{R}_\Gamma\boldsymbol{B}_{u,\Gamma}^\top = 0 \tag{9}$$

associated with the linear dynamical system $\dot{z}(t) = -\boldsymbol{A}^\top\boldsymbol{z}(t) + \boldsymbol{v}(t)$ with cost function $\int_0^\infty[\boldsymbol{z}(\tau)^\top\boldsymbol{B}_{u,\Gamma}\boldsymbol{R}_\Gamma\boldsymbol{B}_{u,\Gamma}^\top\boldsymbol{z}(\tau) + \boldsymbol{v}(\tau)Q^{-1}\boldsymbol{v}(\tau)]d\tau$. When an actuator subset fails to stabilize the system, (7) will not have a stabilizing solution, whereas (9) always has a stabilizing solution since the input matrix of the associated system is identity. The null space of inverse cost matrix $\boldsymbol{P}_\Gamma$ coincides with the unstabilizable subspace, causing $\boldsymbol{X}_\Gamma$ to be infinite in these directions. In the stabilizable subspace, $\boldsymbol{P}_\Gamma$ gives valuable quantitative information about the effectiveness of a particular actuator subset for controlling the system, even when it fails to stabilize. For unstable systems, the greedy algorithm can thus be modified to minimize the set function $\mathbf{trace}((\boldsymbol{P}_\Gamma)^\dagger)$, where $(\boldsymbol{P}_\Gamma)^\dagger$ denotes the Moore-Penrose pseudo-inverse of the solution to (9); see Algorithm 2. In

Algorithm 2 A greedy heuristic for (8) with unstable $\boldsymbol{A}$.

$\Gamma \leftarrow \emptyset$
while $|\Gamma| \leq k$ **do**
  $e^\star = \underset{e\in\mathcal{V}\setminus\Gamma}{\text{argmin}} \quad \mathbf{trace}((\boldsymbol{P}_\Gamma)^\dagger)$
  $\Gamma \leftarrow \Gamma \cup \{e^\star\}$
end while
$\Gamma^\star \leftarrow \Gamma$

other problems, it may be desired to add actuators until some constraint is met, rather than adding a fixed number of actuators. For example, one may want to add a set of actuators to optimize LQR performance and stabilize the system:

$$\underset{\Gamma\subset\{1,\ldots,N\}}{\text{minimize}} \quad \mathbf{trace}(\boldsymbol{P}_\Gamma^{-1}) \tag{10a}$$

$$\text{subject to} \quad (\boldsymbol{A}, \boldsymbol{B}_{u,\Gamma}) \quad \text{is stabilizable.} \tag{10b}$$

The greedy algorithm can be modified to add actuators until stabilizability is achieved by replacing the condition in the while loop in Algorithm 2 with $\mathrm{rank}(\boldsymbol{P}_\Gamma) < n_x$.

VI. Numerical Tests

The stabilizability formulation for actuator selection and design are tested on a network with a $N = 15$ unstable nodes [21]. The 15 nodes are randomly placed on a square of dimensions 3×3 units. Fig. 1-(a) shows the location of the nodes used here. The number of states per node is $n_{x_i} = 2$, and the number of inputs per node is $n_{u_i} = 1$, that is, each column of $\boldsymbol{B}_u$ corresponds to a node. The actuator selection problem is to select the columns of so that the pair $(\boldsymbol{A}, \boldsymbol{B}_u\boldsymbol{\Pi})$ stabilizable. The formulation (3) is tested first. In order to ensure that the resulting matrix $\boldsymbol{S}$ is positive definite, the constraint $\boldsymbol{S} \succeq 10^{-5}\boldsymbol{I}_{30}$ is enforced. The problem is solved with YALMIP's BB algorithm combined with SeDuMi as an SDP solver [22]. Table II lists the selected actuators.

The optimal selection is compared with the simple greedy algorithm where the columns of $\boldsymbol{B}_u$ are considered one after the other until the LMI (3b) with $\boldsymbol{S} \succeq 10^{-5}\boldsymbol{I}_{30}$ is satisfied (greedy algorithm with linear selection). An alternative greedy algorithm is to randomly choose columns of $\boldsymbol{B}_u$ until the previous inequalities are satisfied. This algorithm was run three times and the run with the smallest number of selected actuators is reported in Table II. For this algorithm, the selected columns of $\boldsymbol{B}_u$ are entered sorted in the LMI (that is, not with the random order).

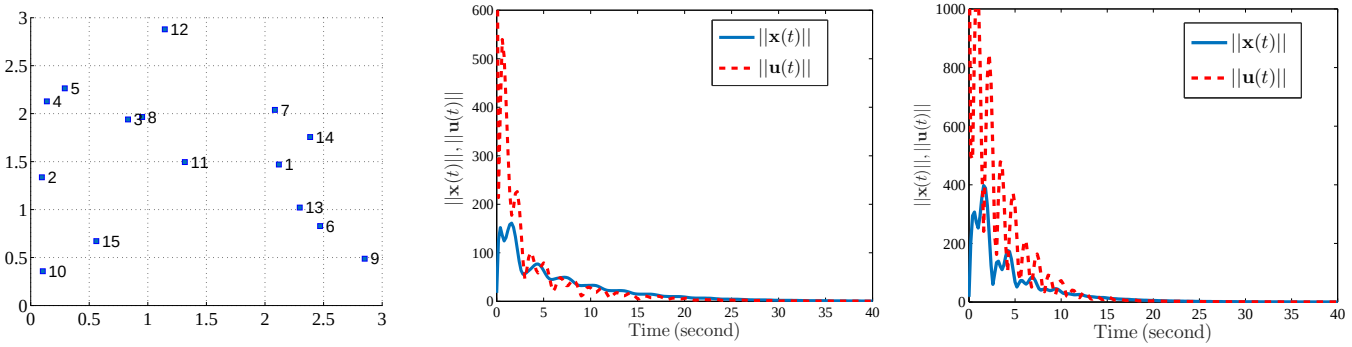
The optimal selection method reveals that the smallest number of actuators that makes the pair $(\boldsymbol{A}, \boldsymbol{B}_u\boldsymbol{\Pi})$ stabilizable is only 9 out of the available 15. The greedy algorithm with linear selection must consider the first 10 columns of $\boldsymbol{B}_u$ in order to satisfy the stabilizability condition. The greedy algorithm with randomization yields also 10 actuators; see Table II.

TABLE II
Selected actuators using optimal and greedy algorithms

Method	Actuator Selection	$\sum_i \pi_i$
Optimal	2, 4, 6–12	**9***
Greedy with linear selection	1–10	10
Greedy with 3 randomizations	1, 3–7, 9, 13–15	10

Fig. 1-(b) and Fig. 1-(c) depict a comparison between the norms of of control actions and state trajectories for the 15 control inputs and 30 states. Specifically, at each time-instant, the state norm $\|\boldsymbol{x}(t)\|_2$ and input $\|\boldsymbol{u}(t)\|_2$ are computed for the optimal solution of the MISDP and the greedy algorithms (with linear selection of the columns of $\boldsymbol{B}_u$). The figures illustrate the state-transients and the control actions are both smaller for the optimal case which is expected. The state and control plots for the greedy algorithm with randomization is omitted as it returns results nearly identical to the greedy case.

Table III shows numerical results as σ (the positive scalar optimization variable) changes; see Section III and Table I.



(a) (b) (c)

Fig. 1. (a) Placement of 15 nodes. (b) Norm of the optimal state and control input trajectory using the optimal solution to the MISDP (3) with $\sum_i \pi_i = 9$. The state-feedback control input drives the system to stability. (c) Norm of the optimal state and control input trajectory using the greedy algorithm for solving the MISDP (3)with $\sum_i \pi_i = 10$. The state-feedback control input drives the system to stability, although the control input magnitude is higher than the optimal solution.

TABLE III
COMPARISON BETWEEN THE OPTIMAL SOLUTION (COLUMNS 2–4) AND GREEDY SOLUTION (COLUMNS 5–7) TO THE STABILIZABILITY PROBLEM WITH ACTUATOR SELECTION FOR LTI SYSTEMS FOR DIFFERENT σ.

σ	$\Sigma\pi_i$	CL-St.	$\Delta t(s)$	$\Sigma\pi_i$	CL-St.	$\Delta t(s)$
0.6	8	✗	602.1	10	✓	12.5
0.55	8	✗	668.9	10	✓	9.7
0.5	8	✗	828.3	10	✓	9.3
0.45	9	✓	1527.5	10	✓	9.3
0.4	9	✓	669.5	10	✓	9.7
0.35	9	✓	627.7	10	✓	9.7
0.3	9	✓	462.2	10	✓	9.8
0.25	9	✓	363.2	10	✓	9.4
0.2	9	✗	542.1	11	✓	11.0
0.15	10	✓	812.6	12	✓	11.6
0.1	10	✓	3925.8	12	✓	11.3

Columns 2–4 reflect the solution from the optimal MISDP, whereas Columns 5–7 show the solution via the greedy algorithm with linear selection—as σ changes. The table also includes the simulation time $\Delta t(s)$ and whether the closed-loop system is stable (Cl-St.) or not. Due to numerical errors, and even though the LMIs are satisfied, the closed-loop system may not be stable unfortunately. Hence, it is necessary to check whether the closed-loop system is stable or not, even if a feasible solution is obtained. Most importantly, the optimal algorithm returned 9 actuators as the minimum number of actuators needed, assuming that the closed loop stability is required. Note that the computational time for greedy algorithms (as expected) is much smaller than the MISDP solution. For large-scale systems, the computational time will increase if sparsity is not considered, but that will generate a smaller number of actuators (and a better transient state performance). In other words, one can trade optimality for tractability via sub-optimal, greedy routines.

REFERENCES

[1] M. Van De Wal and B. De Jager, "A review of methods for input/output selection," *Automatica*, vol. 37, no. 4, pp. 487–510, 2001.

[2] F. Pasqualetti, S. Zampieri, and F. Bullo, "Controllability metrics, limitations and algorithms for complex networks," *Control of Network Systems, IEEE Transactions on*, vol. 1, no. 1, pp. 40–52, 2014.

[3] I. Rajapakse, M. Groudine, and M. Mesbahi, "Dynamics and control of state-dependent networks for probing genomic organization," *Proceedings of the National Academy of Sciences*, vol. 108, no. 42, pp. 17 257–17 262, 2011.

[4] W.-X. Wang, X. Ni, Y.-C. Lai, and C. Grebogi, "Optimizing controllability of complex networks by minimum structural perturbations," *Physical Review E*, vol. 85, no. 2, p. 026115, 2012.

[5] T. H. Summers and J. Lygeros, "Optimal sensor and actuator placement in complex dynamical networks," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 3784–3789, 2014.

[6] T. Summers, "Actuator placement in networks using optimal control performance metrics," in *to appear, IEEE Conference on Decision and Control*. IEEE, 2016.

[7] F. P. Y. Zhao and J. Cortés, "Scheduling of control nodes for improved network controllability," in *Submitted to Conference on Decisions and Controls*. IEEE, 2016.

[8] V. Tzoumas, M. A. Rahimian, G. J. Pappas, and A. Jadbabaie, "Minimal actuator placement with bounds on control effort," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 67–78, 2016.

[9] A. Clark, B. Alomair, L. Bushnell, and R. Poovendran, "Minimizing convergence error in multi-agent systems via leader selection: A supermodular optimization approach," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, pp. 1480–1494, 2014.

[10] T. Summers, I. Shames, J. Lygeros, and F. Dörfler, "Topology design for optimal network coherence," in *Control Conference (ECC), 2015 European*. IEEE, 2015, pp. 575–580.

[11] H. Zhang, R. Ayoub, and S. Sundaram, "Sensor selection for optimal filtering of linear dynamical systems: Complexity and approximation," in *IEEE Conference on Decision and Control (CDC)*, 2015.

[12] K. Zhou, J. C. Doyle, K. Glover *et al.*, *Robust and optimal control*. Prentice hall New Jersey, 1996, vol. 40.

[13] C. A. Crusius and A. Trofino, "Sufficient lmi conditions for output feedback control problems," *Automatic Control, IEEE Transactions on*, vol. 44, no. 5, pp. 1053–1057, 1999.

[14] S. P. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear matrix inequalities in system and control theory*. SIAM, 1994, vol. 15.

[15] D. D. Siljak, D. M. Stipanovic, and A. I. Zecevic, "Robust decentralized turbine/governor control using linear matrix inequalities," *IEEE Transactions on Power Systems*, vol. 17, no. 3, pp. 715–722, 2002.

[16] "Mixed-integer support in CVX 2.0," CVX Research, 2012 Aug. 31. [Online]. Available: http://cvxr.com/news/2012/08/midcp/

[17] "Integer programming," YALMIP Wiki, Sept. 1 2015. [Online]. Available: http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Tutorials.IntegerProgramming

[18] J. G. VanAntwerp and R. D. Braatz, "A tutorial on linear and bilinear matrix inequalities," *Journal of Process Control*, vol. 10, no. 4, pp. 363–385, 2000.

[19] S. Boyd and L. Vandenberghe, "Semidefinite programming relaxations of non-convex problems in control and combinatorial optimization," in *Communications, Computation, Control and Signal Processing: A Tribute to Thomas Kailath*, A. Paulraj, V. Roychowdhuri, and C. Schaper, Eds., 1997, ch. 15, pp. 279–288.

[20] T. H. Summers, F. L. Cortesi, and J. Lygeros, "On submodularity and controllability in complex dynamical networks," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 1, pp. 91–101, March 2016.

[21] M. Jovanovic, "Network with 100 unstable nodes." [Online]. Available: http://people.ece.umn.edu/~mihailo/software/lqrsp/dist_sys.html

[22] J. Löfberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *Proc. IEEE Int. Symp. Computer Aided Control Systems Design*. IEEE, 2004, pp. 284–289.