Automatica 106 (2019) 124–133

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Brief paper

Algorithms for joint sensor and control nodes selection in dynamic networks☆

Sebastian A. Nugroho [a], Ahmad F. Taha [a,*], Nikolaos Gatsis [a], Tyler H. Summers [b], Ram Krishnan [a]

[a] Department of Electrical and Computer Engineering, The University of Texas at San Antonio, 1 UTSA Circle, San Antonio, TX 78249, USA
[b] Department of Mechanical Engineering, The University of Texas at Dallas, 800 W Campbell Rd, Richardson, TX, 75080, USA

ARTICLE INFO

Article history:
Received 17 May 2018
Received in revised form 7 March 2019
Accepted 15 April 2019
Available online 17 May 2019

Keywords:
Sensor and control nodes selection
Static output feedback control
Mixed-integer nonlinear programming
Combinatorial heuristics

ABSTRACT

The problem of placing or selecting sensors and control nodes plays a pivotal role in the operation of dynamic networks. This paper proposes optimal algorithms and heuristics to solve the Simultaneous Sensor and Actuator Selection Problem (SSASP) in linear dynamic networks. In particular, a sufficiency condition of static output feedback stabilizability is used to obtain the minimal set of sensors and control nodes needed to stabilize an unstable network. We then show that SSASP can be written as a mixed-integer nonconvex problem. To solve this nonconvex combinatorial problem, three methods based on *(i)* mixed-integer nonlinear programming, *(ii)* binary search algorithms, and *(iii)* simple heuristics are proposed. The first method yields optimal solutions to SSASP—given that some constants are appropriately selected. The second method requires a database of binary sensor/actuator combinations, returns optimal solutions, and necessitates no tuning parameters. The third approach is a heuristic that yields suboptimal solutions but is computationally attractive. The theoretical properties of these methods are discussed and numerical tests on dynamic networks showcase the trade-off between optimality and computational time.

1. Introduction

Consider an unstable dynamic network of N interconnected nodes

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \tag{1}$$

where $\boldsymbol{A}$ has at least one unstable eigenvalue; $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$, and $\boldsymbol{y}(t)$ collect the state, input, and output vectors for all N nodes. This paper studies the joint problems of *(i)* stabilization of dynamic network (1) through *static output feedback control* (SOFC) while simultaneously *(ii)* selecting/placing minimal number of sensors and control nodes.

Problem *(i)* corresponds to finding a control law $\boldsymbol{u}(t) = \boldsymbol{F}\boldsymbol{y}(t)$ such that the closed loop system eigenvalues of $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{F}\boldsymbol{C}$ are in the left-half plane (Astolfi & Colaneri, 2000). This type of control is advantageous in the sense that it only requires output measurements rather than full state information, is analogous to the simple proportional controller, and can be implemented without an observer or an augmented dynamic system. Problem *(ii)* corresponds to finding minimal number of *sensors and actuators* (SA) yielding a feasible solution for the static output feedback (SOF) stabilization problem. The joint formulations of Problems *(i)*–*(ii)* can be abstracted through this high-level optimization routine

$$\min_{\boldsymbol{\Pi},\boldsymbol{\Gamma},\boldsymbol{F}} \quad \sum_{k=1}^{N} \pi_k + \gamma_k \tag{2a}$$

$$\text{s.t.} \quad \text{Real}(\text{eig}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\Pi}\boldsymbol{F}\boldsymbol{\Gamma}\boldsymbol{C})) < 0, \quad \pi_i, \gamma_j \in \{0, 1\} \tag{2b}$$

where π_i and γ_j are binary variables selecting the ith actuator and jth sensor; $\boldsymbol{\Pi}$ and $\boldsymbol{\Gamma}$ are diagonal matrices containing all π_i and γ_j. These binary variables post- and pre-multiply $\boldsymbol{B}$ and $\boldsymbol{C}$, thereby activating the optimal sensors and control nodes while designing a SOFC law. Even for small to mid-size dynamic networks, problem (2) is difficult to solve as the SOFC problem—without the SA selection—is known to be nonconvex (Crusius & Trofino, 1999) (conjectured to be NP-hard (Peretz, 2016)), and the SA selection introduces binary variables thereby accentuating

☆ We also acknowledge the financial support from National Science Foundation through Grants CMMI-DCSD-1728629 and 1728605. The material in this paper was presented at the 2018 American Control Conference, June 27–29, 2018, Milwaukee, WI, USA. This paper was recommended for publication in revised form by Associate Editor Andrey V. Savkin under the direction of Editor Ian R. Petersen.

* Corresponding author.
E-mail addresses: sebastian.nugroho@my.utsa.edu (S.A. Nugroho), ahmad.taha@utsa.edu (A.F. Taha), nikolaos.gatsis@utsa.edu (N. Gatsis), tyler.summers@utdallas.edu (T.H. Summers), ram.krishnan@utsa.edu (R. Krishnan).

https://doi.org/10.1016/j.automatica.2019.04.047
0005-1098/

the nonconvexity. To that end, the objective of this paper is to develop optimal algorithms and heuristics to solve Problem (2). Next, we summarize the recent literature on solving variants of (2).

Hundreds of studies have investigated the separate problem of minimally selecting/placing sensors *or* actuators while performing state estimation *or* state-feedback control. This paper studies the joint SA selection in the sense that an observer-based controller—which invokes the separation principle and requires a dynamic system module to perform state estimation—is not needed. For this reason, we do not delve into the literature of separate sensor or actuator selection. Interested readers are referred to our recent work (Nugroho, Taha, Summers, & Gatsis, 2018; Taha, Gatsis, Summers, & Nugroho, 2018) for a summary on methods that solve the separate SA selection problems. The literature of addressing the *simultaneous sensor and actuator selection problem* (SSASP) is summarized next.

Several attempts have been made to address variants of the SSASP in dynamic networks through the more general *dynamic output feedback control* (DOFC) framework. Specifically, the authors in De Oliveira and Geromei (2000) investigate the $\mathcal{H}_2$ minimization via DOFC with SA selection, in which a reformulated suboptimal problem in the form of mixed-integer semi-definite program (MI-SDP) is proposed and solved using a coordinate descent algorithm. In Argha, Su, Savkin, and Celler (2017), the SSASP for multi-channel $\mathcal{H}_2$ DOFC with regional pole placement is addressed. In particular, the authors develop a semi-definite program (SDP) framework and propose a sparsity-promoting algorithm to obtain sparse row/column feedback control matrices. This approach ultimately yields binary SA selection, without needing binary variables. The same algorithm is then employed in Singh, Swevers, and Pipeleers (2018) for SSA selection with simpler $\mathcal{H}_2/\mathcal{H}_\infty$ formulations. The SA selection with control configuration selection problem is formulated in Pequito, Kar, and Pappas (2015) using structural design and graph theory, which is proven to be NP-hard. Although this particular problem is similar to the SSASP with SOFC given in (2), the problem proposed in Pequito et al. (2015), along with the algorithm, are based on the information of structural pattern of the dynamic matrix. The limitations of these studies are discussed next.

First, the majority of works (Argha et al., 2017; De Oliveira & Geromei, 2000; Singh et al., 2018) consider the $\mathcal{H}_2/\mathcal{H}_\infty$ control framework in conjunction with dynamic output feedback which requires an additional block of dynamical systems to construct the control action (which is not the case in SOFC). Second, the work in De Oliveira and Geromei (2000) assumes that the number of SA to be selected is known *a priori*, which for certain cases is not very intuitive. Third, the sparsity-promoting algorithm proposed in Argha et al. (2017) and Singh et al. (2018) is based on convex relaxation of the l_0 norm—called re-weighted l_1 norm—which is then solved iteratively until the solution converges, thus making it less suitable for larger dynamic networks. The other drawback of this method is that arbitrary convex constraints on the binary selection variables are not intuitive to include. Finally, the algorithm proposed in Pequito et al. (2015)—which interestingly runs in polynomial-time if the structure of the dynamic matrix is irreducible—only computes the structure and the corresponding costs of the feedback matrix (along with the sets of selected SA).

As an alternative to the aforementioned methods, this paper proposes algorithms and heuristics to solve the SSASP for unstable dynamic networks via SOFC. Specifically, we use a sufficiency condition for SOFC from Crusius and Trofino (1999) which reduces the SOF control problem—without the SSA selection—from a nonconvex problem into a simple linear matrix inequality (LMI) feasibility problem. The developed approaches are based on MI-SDP, binary search algorithms, and simple heuristics that use the problem structure to find good suboptimal solutions. A preliminary version of this work appeared in Nugroho et al. (2018) where we focus mainly on the MI-SDP approach. Here, we significantly extend this approach with the addition of binary search algorithms, heuristics, thorough analytical discussion of the properties of the developed methods, and comprehensive numerical experiments. The paper contributions and organization are discussed as follows.

- First, we formally introduce the SOF stabilizability problem (Section 2). The SSASP through SOFC is then formulated and shown to be a nonconvex problem with mixed-integer nonlinear matrix inequality (MI-NMI) constraints (Section 3). We prove that the SSASP can be formulated as a MI-SDP, and the equivalence between the two is shown (Section 4). The MI-SDP, if solved using combinatorial optimization techniques, yields an optimal solution to the SSASP.
- As a departure from the MI-SDP approach, we introduce a routine akin to binary search algorithms that computes an optimal solution for SSASP—the proof of optimality is given. The routine requires a database of binary SA combinations (Section 5).
- A heuristic that scales better than the first two approaches is also introduced. The heuristic is based on constructing a simple logic of infeasible or suboptimal combinations of SA, while offering flexibility in terms of the tradeoff between the computational time and distance to optimality (Section 6). A brief discussion on the computational complexity as well as thorough numerical tests showcasing the applicability of the proposed algorithms are provided (Sections 7 and 8).

The presented algorithms in this paper have their limitations which are all discussed with future work and concluding marks (Section 9). The online preprint of this paper (Nugroho, Taha, Gatsis, Summers, & Krishnan, 2018) provides a more thorough discussion on the proposed heuristic algorithm presented in Section 6.

2. Static output feedback control review

Consider a dynamic network consisting of N nodes/sub-systems with $\mathcal{N} = \{1, \ldots, N\}$ defining the set of nodes. The network dynamics are given as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t). \tag{3}$$

The state, input, and output vectors on each node $i \in \mathcal{N}$ are represented by $\boldsymbol{x}_i(t) \in \mathbb{R}^{n_{x_i}}$, $\boldsymbol{u}_i(t) \in \mathbb{R}^{n_{u_i}}$, and $\boldsymbol{y}_i(t) \in \mathbb{R}^{n_{y_i}}$, in which $\boldsymbol{u}_i(t)$ and $\boldsymbol{y}_i(t)$ at node i only correspond to that particular node. The matrices $\boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}$ and $\boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$ are constructed as $\boldsymbol{B} \triangleq \text{Blkdiag}(\boldsymbol{B}_1, \ldots, \boldsymbol{B}_N)$ and $\boldsymbol{C} \triangleq \text{Blkdiag}(\boldsymbol{C}_1, \ldots, \boldsymbol{C}_N)$ (Blkdiag$(\cdot)$ forms a block diagonal matrix). This assumption enforces the coupling among nodes to be represented in the state evolution matrix $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$. In this paper, we consider the following assumption and definition for SOF stabilizability.

Assumption 1. The system (3) satisfies the following conditions: (a) The pair $(\boldsymbol{A}, \boldsymbol{B})$ is stabilizable; (b) the pair $(\boldsymbol{A}, \boldsymbol{C})$ is detectable; (c) $\boldsymbol{B}$ and $\boldsymbol{C}$ are full rank.

Definition 1. The dynamical system (3) is stabilizable via SOF if there exists $\boldsymbol{F} \in \mathbb{R}^{n_u \times n_y}$ with control law given as $\boldsymbol{u}(t) = \boldsymbol{F}\boldsymbol{y}(t)$ such that $\bar{\lambda}_{\text{Re}}(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{F}\boldsymbol{C}) < 0$.

The above definition and assumption are standard in the SOF control literature (Astolfi & Colaneri, 2000; Kučera & Souza, 1995; Syrmos, Abdallah, Dorato, & Grigoriadis, 1997) ($\bar{\lambda}_{\text{Re}}(\cdot)$ computes the maximum value of the real part of eigenvalues of a square

matrix). The SOF stabilizability problem is equivalent to the following feasibility problem.

Proposition 1 (*From Syrmos et al., 1997*)**.** *The dynamical system (3) is SOF stabilizable with output feedback gain $\boldsymbol{F} \in \mathbb{R}^{n_u \times n_y}$ if and only if there exists $\boldsymbol{P} \in \mathbb{S}^{n_x}_{++}$ such that*

$$\boldsymbol{A}^\top \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^\top \boldsymbol{F}^\top \boldsymbol{B}^\top \boldsymbol{P} + \boldsymbol{P}\boldsymbol{B}\boldsymbol{F}\boldsymbol{C} \prec 0. \tag{4}$$

In Proposition 1, $\mathbb{S}^{n_x}_{++}$ denotes the set of positive-definite matrices of size n_x. Realize that the matrix inequality (4) is nonconvex due to bilinearity in terms of $\boldsymbol{P}$ and $\boldsymbol{F}$. Thus, instead of using (4), we consider a sufficient condition proposed in Crusius and Trofino (1999) as it renders the SOF problem into an LMI framework—presented in the following proposition.

Proposition 2 (*From Crusius & Trofino, 1999*)**.** *The dynamical system (3) is SOF stabilizable if there exist $\boldsymbol{M} \in \mathbb{R}^{n_u \times n_u}$, $\boldsymbol{P} \in \mathbb{S}^{n_x}_{++}$, $\boldsymbol{N} \in \mathbb{R}^{n_u \times n_y}$, and $\boldsymbol{F} \in \mathbb{R}^{n_u \times n_y}$ such that the following LMIs are feasible*

$$\boldsymbol{A}^\top \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^\top \boldsymbol{N}^\top \boldsymbol{B}^\top + \boldsymbol{B}\boldsymbol{N}\boldsymbol{C} \prec 0, \quad \boldsymbol{B}\boldsymbol{M} = \boldsymbol{P}\boldsymbol{B}, \tag{5}$$

with SOF gain computed as $\boldsymbol{F} = \boldsymbol{M}^{-1}\boldsymbol{N}$.

3. Problem formulation

To formulate the SSASP, let $\gamma_i \in \{0, 1\}$ and $\pi_i \in \{0, 1\}$ be two binary variables that represent the selection of SA at node i of the dynamic networks. We consider that $\gamma_i = 1$ if the sensor of node i is selected (or activated) and $\gamma_i = 0$ otherwise. Similarly, $\pi_i = 1$ if the actuator of node i is selected and $\pi_i = 0$ otherwise. By defining $\boldsymbol{\Pi} \triangleq \text{Blkdiag}(\pi_1 \boldsymbol{I}_{n_{u_1}}, \ldots, \pi_N \boldsymbol{I}_{n_{u_N}})$ and $\boldsymbol{\Gamma} \triangleq \text{Blkdiag}(\gamma_1 \boldsymbol{I}_{n_{y_1}}, \ldots, \gamma_N \boldsymbol{I}_{n_{y_N}})$, the system dynamics with SA selection can be formulated as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{\Pi}\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = \boldsymbol{\Gamma}\boldsymbol{C}\boldsymbol{x}(t). \tag{6}$$

The SSASP via SOF stabilizability can be expressed as in (7), which optimization variables are $\{\boldsymbol{\pi}, \boldsymbol{\gamma}, \boldsymbol{N}, \boldsymbol{M}, \boldsymbol{P}\}$ with $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_N]^\top$, $\boldsymbol{\gamma} = [\gamma_1, \gamma_2, \ldots, \gamma_N]^\top$.

$$\text{SSASP} \quad \underset{\substack{\boldsymbol{\pi},\boldsymbol{\gamma},\boldsymbol{N}\\ \boldsymbol{M},\boldsymbol{P}}}{\text{minimize}} \ \sum_{k=1}^{N} \pi_k + \gamma_k \tag{7a}$$

$$\text{subject to} \quad \boldsymbol{A}^\top \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^\top \boldsymbol{\Gamma}\boldsymbol{N}^\top \boldsymbol{\Pi}\boldsymbol{B}^\top + \boldsymbol{B}\boldsymbol{\Pi}\boldsymbol{N}\boldsymbol{\Gamma}\boldsymbol{C} \prec 0 \tag{7b}$$

$$\boldsymbol{B}\boldsymbol{\Pi}\boldsymbol{M} = \boldsymbol{P}\boldsymbol{B}\boldsymbol{\Pi} \tag{7c}$$

$$\Phi \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\gamma} \end{bmatrix} \leq \boldsymbol{\phi} \tag{7d}$$

$$\boldsymbol{P} \succ 0, \ \boldsymbol{\pi} \in \{0, 1\}^N, \ \boldsymbol{\gamma} \in \{0, 1\}^N. \tag{7e}$$

Remark 1. The solution of SSASP guarantees that the dynamic network is stabilized using the minimal number of SA, as the closed loop stability is ensured by the sufficient condition for the existence of SOFC given in Proposition 2. This entails that the closed-loop eigenvalues are all in the left side of $j\omega$-axis. If it is desired to move the closed-loop eigenvalues further away from the $j\omega$-axis, the matrix inequality (7b) can be upper bounded by $-\epsilon \boldsymbol{I}$ where $\epsilon > 0$.

Remark 2. Our focus here is to find a SOF control gain that stabilizes dynamical system (6) with minimum number of SA. With that in mind, performance metrics such as robustness and energy cost functions are not considered in SSASP.

In the next sections, $\boldsymbol{\pi}$ and $\boldsymbol{\gamma}$ will be used interchangeably with $\boldsymbol{\Pi}$ and $\boldsymbol{\Gamma}$. Constraints (7b) and (7c) in SSASP are obtained by simply applying the sufficient condition for SOF stabilizability. Constraint (7d) is an additional linear logistic constraint which can be useful to model preferred activation or deactivation of SA on particular nodes and to define the desired minimum and maximum number of activated SA. After solving (7), the selected SA are obtained and represented by $\{\boldsymbol{\pi}^*, \boldsymbol{\gamma}^*\}$. Due to SSA selection, the matrix $\boldsymbol{B}\boldsymbol{\Pi}$ will most likely not be full column rank, hence the existence of an invertible matrix $\boldsymbol{M}$ is not assured. This is not the case when solving (5) due to the fact that $\boldsymbol{B}$ being full column rank and $\boldsymbol{P} \succ 0$ ensure $\boldsymbol{M}$ to be nonsingular—see Lemma 1. However, if (7) returns $\boldsymbol{M}$ that is invertible, then the SSASP is solved with SOF gain $\boldsymbol{F}$ to be computed as $\boldsymbol{M}^{-1}\boldsymbol{N}$. Otherwise, $\boldsymbol{F}$ can be computed as $\hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{N}}$ where $\hat{\boldsymbol{M}}$ and $\hat{\boldsymbol{N}}$ are the submatrices of $\boldsymbol{M}$ and $\boldsymbol{N}$ that correspond to activated SA. Proposition 3 ensures the SOF stabilizability with minimal SA after solving SSASP.

Lemma 1. *Let $\boldsymbol{M} \in \mathbb{R}^{m \times m}$ and $\boldsymbol{P} \in \mathbb{S}^{n_x}$ be the solution of $\boldsymbol{B}\boldsymbol{M} = \boldsymbol{P}\boldsymbol{B}$ where $\boldsymbol{B} \in \mathbb{R}^{n_x \times m}$ and $m \leq n_u$. If $\boldsymbol{P} \succ 0$ and $\text{Rank}(\boldsymbol{B}) = m$, then $\boldsymbol{M}$ is invertible.*

Proposition 3. *Let $\boldsymbol{P}$, $\boldsymbol{M}$, $\boldsymbol{N}$, $\boldsymbol{\Pi}^*$, and $\boldsymbol{\Gamma}^*$ be the solution of SSASP with appropriate dimensions. Also, let $\hat{\boldsymbol{B}} \in \mathbb{R}^{n_x \times m}$, $\hat{\boldsymbol{C}} \in \mathbb{R}^{r \times n_x}$, $\hat{\boldsymbol{M}} \in \mathbb{R}^{m \times m}$, and $\hat{\boldsymbol{N}} \in \mathbb{R}^{m \times r}$, where $m \leq n_u$ and $r \leq n_y$, be the matrices (or submatrices) representing the nonzero components of $\boldsymbol{B}\boldsymbol{\Pi}^*$, $\boldsymbol{\Gamma}^*\boldsymbol{C}$, $\boldsymbol{\Pi}^*\boldsymbol{M}$, and $\boldsymbol{\Pi}^*\boldsymbol{N}\boldsymbol{\Gamma}^*$ that correspond to activated SA. Then, the closed loop system $\boldsymbol{A} + \hat{\boldsymbol{B}}\boldsymbol{F}\hat{\boldsymbol{C}}$ is stable with SOF gain $\boldsymbol{F} = \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{N}}$.*

See Appendices A and B respectively for the proofs of Lemma 1 and Proposition 3. SSASP (7) is nonconvex due to the presence of MI-NMI in the form of $\boldsymbol{\Pi}\boldsymbol{N}\boldsymbol{\Gamma}$ and mixed-integer bilinear matrix equality in (7c). Therefore, it cannot be solved by any general-purpose mixed-integer convex programming solver. To that end, we present three approaches that solve or approximate SSASP.

4. SSASP as a MI-SDP

In this section, we present the first approach to solve (7), which transforms SSASP from a mixed-integer nonconvex problem to a MI-SDP. The following theorem presents this result.

Theorem 1. *SSASP is equivalent to*

$$\underset{\substack{\boldsymbol{\pi},\boldsymbol{\gamma},\boldsymbol{N},\boldsymbol{M}\\ \boldsymbol{P},\boldsymbol{\Theta}}}{\text{minimize}} \ \sum_{k=1}^{N} \pi_k + \gamma_k \tag{8a}$$

subject to

$$\boldsymbol{A}^\top \boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} + \boldsymbol{C}^\top \boldsymbol{\Theta}^\top \boldsymbol{B}^\top + \boldsymbol{B}\boldsymbol{\Theta}\boldsymbol{C} \prec 0 \tag{8b}$$

$$\Psi_1(\boldsymbol{N}, \boldsymbol{\Theta}) \leq L_1 \boldsymbol{\Delta}_1(\boldsymbol{\Gamma}, \boldsymbol{\Pi}) \tag{8c}$$

$$\Psi_2(\boldsymbol{M}, \boldsymbol{\Omega}(\boldsymbol{P})) \leq L_2 \boldsymbol{\Delta}_2(\boldsymbol{\Pi}) \tag{8d}$$

$$\Psi_3(\boldsymbol{\Xi}(\boldsymbol{P})) \leq L_3 \boldsymbol{\Delta}_3(\boldsymbol{\Pi}) \tag{8e}$$

$$\boldsymbol{P} \succ 0, \ \Phi \begin{bmatrix} \boldsymbol{\pi} \\ \boldsymbol{\gamma} \end{bmatrix} \leq \boldsymbol{\phi}, \ \boldsymbol{\pi} \in \{0, 1\}^N, \ \boldsymbol{\gamma} \in \{0, 1\}^N, \tag{8f}$$

where (8c),(8d),(8e) are linear constraints in which each function is specified as

$$\Psi_1(\boldsymbol{N}, \boldsymbol{\Theta}) \triangleq \begin{bmatrix} \text{Vec}(\boldsymbol{\Theta}) \\ -\text{Vec}(\boldsymbol{\Theta}) \\ \text{Vec}(\boldsymbol{\Theta}) \\ -\text{Vec}(\boldsymbol{\Theta}) \\ \text{Vec}(\boldsymbol{\Theta} - \boldsymbol{N}) \\ -\text{Vec}(\boldsymbol{\Theta} - \boldsymbol{N}) \end{bmatrix} \tag{9a}$$

$$\boldsymbol{\Delta}_1(\boldsymbol{\Gamma},\boldsymbol{\Pi}) \triangleq \begin{bmatrix} \mathrm{Diag}(\boldsymbol{I}_{n_y}\otimes\boldsymbol{\Pi}) \\ \mathrm{Diag}(\boldsymbol{I}_{n_y}\otimes\boldsymbol{\Pi}) \\ \mathrm{Diag}(\boldsymbol{\Gamma}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(\boldsymbol{\Gamma}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(2\boldsymbol{I}_{n_u\times n_y}-\boldsymbol{I}_{n_y}\otimes\boldsymbol{\Pi}-\boldsymbol{\Gamma}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(2\boldsymbol{I}_{n_u\times n_y}-\boldsymbol{I}_{n_y}\otimes\boldsymbol{\Pi}-\boldsymbol{\Gamma}\otimes\boldsymbol{I}_{n_u}) \end{bmatrix} \tag{9b}$$

$$\boldsymbol{\Psi}_2(\boldsymbol{M},\boldsymbol{\Omega}(\boldsymbol{P})) \triangleq \begin{bmatrix} \mathrm{Vec}(\boldsymbol{M}) \\ -\mathrm{Vec}(\boldsymbol{M}) \\ \mathrm{Vec}(\boldsymbol{\Omega}(\boldsymbol{P})) \\ -\mathrm{Vec}(\boldsymbol{\Omega}(\boldsymbol{P})) \\ \mathrm{Vec}(\boldsymbol{M}-\boldsymbol{\Omega}(\boldsymbol{P})) \\ -\mathrm{Vec}(\boldsymbol{M}-\boldsymbol{\Omega}(\boldsymbol{P})) \end{bmatrix} \tag{9c}$$

$$\boldsymbol{\Delta}_2(\boldsymbol{\Pi}) \triangleq \begin{bmatrix} \mathrm{Diag}(\boldsymbol{I}_{n_u^2}-\boldsymbol{I}_{n_u}\otimes\boldsymbol{\Pi}+\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(\boldsymbol{I}_{n_u^2}-\boldsymbol{I}_{n_u}\otimes\boldsymbol{\Pi}+\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(\boldsymbol{I}_{n_u^2}+\boldsymbol{I}_{n_u}\otimes\boldsymbol{\Pi}-\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(\boldsymbol{I}_{n_u^2}+\boldsymbol{I}_{n_u}\otimes\boldsymbol{\Pi}-\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(2\boldsymbol{I}_{n_u^2}-\boldsymbol{I}_{n_u}\otimes\boldsymbol{\Pi}-\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_u}) \\ \mathrm{Diag}(2\boldsymbol{I}_{n_u^2}-\boldsymbol{I}_{n_u}\otimes\boldsymbol{\Pi}-\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_u}) \end{bmatrix} \tag{9d}$$

$$\boldsymbol{\Psi}_3(\boldsymbol{\Xi}(\boldsymbol{P})) \triangleq \begin{bmatrix} \mathrm{Vec}(\boldsymbol{\Xi}(\boldsymbol{P})) \\ -\mathrm{Vec}(\boldsymbol{\Xi}(\boldsymbol{P})) \end{bmatrix} \tag{9e}$$

$$\boldsymbol{\Delta}_3(\boldsymbol{\Pi}) \triangleq \begin{bmatrix} \mathrm{Diag}(\boldsymbol{I}_{n_x\times n_u}-\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_x}) \\ \mathrm{Diag}(\boldsymbol{I}_{n_x\times n_u}-\boldsymbol{\Pi}\otimes\boldsymbol{I}_{n_x}) \end{bmatrix}, \tag{9f}$$

$\boldsymbol{\Omega}$ and $\boldsymbol{\Xi}$ are functions defined as

$$\boldsymbol{\Omega}(\boldsymbol{P}) \triangleq (\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{B} \tag{10a}$$

$$\boldsymbol{\Xi}(\boldsymbol{P}) \triangleq (\boldsymbol{I}-\boldsymbol{B}(\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top)\boldsymbol{P}\boldsymbol{B}, \tag{10b}$$

$\boldsymbol{\Theta}\in\mathbb{R}^{n_u\times n_y}$, is an additional optimization variable, and $L_{1,2,3}>0$ are three predefined, sufficiently large constants.

The proof of Theorem 1 is given in Appendix C. In (9a)–(9f), $\mathrm{Vec}(\boldsymbol{X})$ returns a stacked $pq\times 1$ column vector of entries of $\boldsymbol{X}\in\mathbb{R}^{p\times q}$, $\mathrm{Diag}(\boldsymbol{Y})$ returns a $n\times 1$ column vector of diagonal entries of square matrix $\boldsymbol{Y}\in\mathbb{R}^{n\times n}$, $\otimes$ denotes the Kronecker product, and $\boldsymbol{I}_n$ is an $n\times n$ identity matrix. Theorem 1 allows the SSASP to be solved as a MI-SDP, which can be handled using a variety of optimization methods such as branch-and-bound algorithms (Gally, Pfetsch, & Ulbrich, 2017; Gamrath et al., 2016), outer approximations (Lubin, Yamangil, Bent, & Vielma, 2016), or branch-and-cut algorithm (Kobayashi & Takano, 2018). The next section presents a departure from MI-SDP to an algorithm that returns optimal solutions to SSASP, without requiring $L_{1,2,3}$.

Remark 3. Although (8) is equivalent to SSASP, the quality of the solution that comes out of (8) is very dependent on the choice of $L_{1,2,3}$. This observation is corroborated by numerical test results discussed in Section 8.

5. Binary search algorithm for SSA selection

In this section we present an algorithm that is similar, in spirit, to binary search algorithms. The presented algorithm here seeks optimality for SSASP while not requiring any tuning parameters such as $L_{1,2,3}$; see Theorem 1.

5.1. Definitions and preliminaries

In what follows, we provide some needed definitions.

Definition 2. Let $\mathcal{S}_\pi$ and $\mathcal{S}_\gamma$ be two N-tuples representing the selection of SA, i.e., $\mathcal{S}_\pi \triangleq (\pi_1,\ldots,\pi_N)$ and $\mathcal{S}_\gamma \triangleq (\gamma_1,\ldots,\gamma_N)$. Then, the selection of SA can be defined as $\mathcal{S}\triangleq(\mathcal{S}_\pi,\mathcal{S}_\gamma)$ such that $\{\boldsymbol{\Pi},\boldsymbol{\Gamma}\}=\mathcal{G}(\mathcal{S})$, $\boldsymbol{\Pi}=\mathcal{G}_\pi(\mathcal{S})$, and $\boldsymbol{\Gamma}=\mathcal{G}_\gamma(\mathcal{S})$ where $\mathcal{G}:\{0,1\}^{2N}\to\mathbb{R}^{n_u\times n_u}\times\mathbb{R}^{n_y\times n_y}$, $\mathcal{G}_\pi:\{0,1\}^{2N}\to\mathbb{R}^{n_u\times n_u}$, and $\mathcal{G}_\gamma:\{0,1\}^{2N}\to\mathbb{R}^{n_y\times n_y}$ are linear mappings. The number of nodes with activated SA is defined as $\mathcal{H}(\mathcal{S})\triangleq\sum_{k=1}^N \pi_k+\gamma_k$ where $\mathcal{H}:\mathcal{S}\to\mathbb{Z}_+$.

Algorithm 1: Binary Search Algorithm (BSA)

```
initialize: S* = (1)^{2N}, p = 1
input: S_p, A, B, C
while S_p ≠ ∅ do
    compute: σ ← |S_p|, q ← ⌈σ/2⌉, S_q ∈ S_p
    if S_q is feasible for (5) then
        S* ← S_q, S_p ← S_p \ {S ∈ S_p | H(S) ≥ H(S_q)}
    else
        S_p ← S_p \ {S ∈ S_p | S_q ∨ S = S_q}
    p ← p + 1
output: S*
```

Definition 3. Let $\boldsymbol{\mathcal{S}}\triangleq\{\mathcal{S}_q\}_{q=1}^{\sigma}$ be the candidate set such that it contains all possible combinations of SA where σ denotes the number of total combinations, i.e., $\sigma\triangleq|\boldsymbol{\mathcal{S}}|$. Then, the following conditions hold: (1) for every $\mathcal{S}\in\boldsymbol{\mathcal{S}}$, $\mathcal{S}\in\left\{\mathcal{S}\in\{0,1\}^{2N}\mid\mathcal{G}(\mathcal{S})\text{ is feasible for (7d)}\right\}$, (2) for every q where $1\le q\le\sigma$, $\mathcal{H}(\mathcal{S}_{q-1})\le\mathcal{H}(\mathcal{S}_q)$.

Definition 4. For any $\mathcal{S}_q\in\boldsymbol{\mathcal{S}}$ such that $\{\boldsymbol{\Pi}_q,\boldsymbol{\Gamma}_q\}=\mathcal{G}(\mathcal{S}_q)$, $\boldsymbol{B}_q$ and $\boldsymbol{C}_q$ are defined as the matrices containing the nonzero components of $\boldsymbol{B}\boldsymbol{\Pi}_q$ and $\boldsymbol{\Gamma}_q\boldsymbol{C}$ that correspond to the activated SA. Then, we say that $\mathcal{S}_q$ is feasible for (5) if and only if the triplet $(\boldsymbol{A},\boldsymbol{B}_q,\boldsymbol{C}_q)$ is feasible for (5).

The following example shows how the candidate set $\boldsymbol{\mathcal{S}}$ is constructed for a given simple logistic constraint.

Example 1. Suppose that the dynamical system has two nodes. If the logistic constraint dictates that $1\le\mathcal{H}(\mathcal{S})=\sum_{k=1}^2\pi_k+\gamma_k<4$, then $\boldsymbol{\mathcal{S}}_1$ can be constructed as

$$\begin{aligned}\boldsymbol{\mathcal{S}}_1=\{&(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1),(1,1,0,0),\\&(1,0,1,0),(1,0,0,1),(0,1,1,0),(0,1,0,1),(0,0,1,1),\\&(1,1,1,0),(1,1,0,1),(1,0,1,1),(0,1,1,1)\}.\end{aligned}$$

5.2. Binary search algorithm to solve SSASP

The objective of this algorithm is to find *an* optimal solution $\mathcal{S}^*\in\boldsymbol{\mathcal{S}}$ such that $\mathcal{H}(\mathcal{S}^*)\le\mathcal{H}(\mathcal{S})$ for all $\mathcal{S}\in\boldsymbol{\mathcal{V}}$ where $\boldsymbol{\mathcal{V}}\triangleq\{\mathcal{S}\in\boldsymbol{\mathcal{S}}\mid\mathcal{S}\text{ is feasible for (5)}\}$—that is, for any $\mathcal{S}\in\boldsymbol{\mathcal{V}}$, there exist a corresponding feedback gain $\boldsymbol{F}$ that stabilizes dynamical system (6). Realize that any $\mathcal{S}_q\in\boldsymbol{\mathcal{V}}$, $(\boldsymbol{A},\boldsymbol{B}_q,\boldsymbol{C}_q)$ is feasible for SSASP with objective function value equal to $\mathcal{H}(\mathcal{S}_q)$.

The routine to solve SSASP based on binary search algorithm is now explained and summarized in Algorithm 1. Let p be the index of iteration and q be the index of position in the ordered set $\boldsymbol{\mathcal{S}}$. Hence at iteration p, the candidate set containing all possible combinations of SA can be represented as $\boldsymbol{\mathcal{S}}_p$, with $\sigma=|\boldsymbol{\mathcal{S}}_p|$ (the cardinality of $\boldsymbol{\mathcal{S}}_p$), and any element of $\boldsymbol{\mathcal{S}}_p$ at position q can be represented by $\mathcal{S}_q$. Also, let $\mathcal{S}^*$ be the solution at iteration p.

Next, obtain $\mathcal{S}_q$ where $\mathcal{S}_q\in\boldsymbol{\mathcal{S}}_p$, $q=\lceil\sigma/2\rceil$, and $\lceil\cdot\rceil$ denotes the ceiling function. In this step, we need to determine whether system (6) is SOF stabilizable with the particular combination of SA $\{\boldsymbol{\Pi}_q,\boldsymbol{\Gamma}_q\}=\mathcal{G}(\mathcal{S}_q)$. To that end, we solve the LMIs (5). If $\mathcal{S}_q$ is feasible for (5), then update $\mathcal{S}^*\leftarrow\mathcal{S}_q$. Since $\mathcal{S}_q$ is feasible, then we can discard all combinations that have more **or** equal number of activated SA, i.e., the combinations that are suboptimal. Otherwise, if $\mathcal{S}_q$ is infeasible for (5), $\mathcal{S}_q$ can be discarded along with all combinations that (a) have less number of activated SA than $\mathcal{S}_q$

and (b) the activated SA are included in $\mathcal{S}_q$. Realize that the above method reduces the size of $\boldsymbol{\mathcal{S}}_p$ in every iteration since one or more elements of $\boldsymbol{\mathcal{S}}_p$ are discarded. The algorithm now continues and terminates whenever $\boldsymbol{\mathcal{S}}_p = \emptyset$. The details of this algorithm are given in Algorithm 1. Example 2 gives an illustration how $\boldsymbol{\mathcal{S}}_p$ is constructed in every iteration.

Example 2. Consider again the dynamic system from Example 1. Let (1, 0, 0, 1) be the starting combination and assume that it is infeasible for (5). Then, by Algorithm 1, the set {(1, 0, 0, 0), (0, 0, 0, 1)} is removed from $\boldsymbol{\mathcal{S}}_1$, which gives $\boldsymbol{\mathcal{S}}_2 = \boldsymbol{\mathcal{S}}_1 \backslash \{(1, 0, 0, 0), (0, 0, 0, 1)\}$. Let (0, 1, 0, 1) be the new starting point. If we assume that it is feasible for (5), then by Algorithm 1, the suboptimal candidates of SA can be discarded such that $\boldsymbol{\mathcal{S}}_3 = \{(0, 1, 0, 0), (0, 0, 1, 0)\}$. This algorithm continues in a fashion similar to the above routine. If none of these combinations in $\boldsymbol{\mathcal{S}}_3$ is feasible, then Algorithm 1 returns $\mathcal{S}^* = (0, 1, 0, 1)$ as the solution.

In what follows, we discuss the optimality of Algorithm 1 through Theorem 2—see Appendix D for the proof.

Theorem 2. *Algorithm 1 yields* ***an*** *optimal solution of* SSASP.

Remark 4. $\mathcal{S}^*$ from Algorithm 1 might not be unique. This is the case since there could be multiple binary combinations of SA yielding the same number of activated SA and hence the same objective function value $\sum_{k=1}^{N} \pi_k + \gamma_k$.

6. Heuristics to solve SSASP

The binary search algorithm in the previous section requires the construction of the candidate set $\boldsymbol{\mathcal{S}}$ in an off-line database, while leading to an optimal solution for the SSASP. Seeking optimality and constructing an off-line database might be impractical for large-scale dynamic networks. Moreover, the other approach presented in Section 4 entails solving (8), a MI-SDP, which might consume large computational resources. This motivates the development of a heuristic for the SSASP that forgoes optimality. In short, the heuristic builds a dynamic, virtual database of all possible combinations—not by generating all of these combinations, but by having a procedure that identifies suboptimal/infeasible candidates—while attempting to find a SA combination that has the least number of activated SA that makes system (6) SOF stabilizable.

We now introduce the details of the heuristic. Define $\boldsymbol{\mathcal{W}} \triangleq \{\mathcal{S} \in \{0, 1\}^{2N} \mid \mathcal{G}(\mathcal{S}) \text{ is not feasible for (7d)}\}$. Since we are interested in finding a candidate $\mathcal{S}$ that is feasible for (5), we just need to check whether $\mathcal{S} \notin \boldsymbol{\mathcal{W}}$ using the logistic constraint (7d). Next, from the logistic constraint (7d), we define $\underline{w} \triangleq \min(\mathcal{H}(\mathcal{S}))$ and $\bar{w} \triangleq \max(\mathcal{H}(\mathcal{S}))$ for all $\mathcal{S} \notin \boldsymbol{\mathcal{W}}$. That is, $\underline{w}$ and $\bar{w}$ represent the required minimum and maximum number of activated SA so that any candidate $\mathcal{S} \notin \boldsymbol{\mathcal{W}}$ must satisfy $\underline{w} \leq \mathcal{H}(\mathcal{S}) \leq \bar{w}$. More importantly, $\underline{w}$ and $\bar{w}$ can also be used to bound the search space of a potential candidate $\mathcal{S}$. In contrast with Algorithm 1, the heuristic constructs and updates—in each iteration—a set that contains combinations of SA that are known to be infeasible for (7). This finite set is referred to as the *forbidden set* and symbolized by $\boldsymbol{\mathcal{Z}}$. Clearly, $\boldsymbol{\mathcal{W}} \subseteq \boldsymbol{\mathcal{Z}}$. Thus, any candidate $\mathcal{S}$ must not belong in $\boldsymbol{\mathcal{Z}}$ because any $\mathcal{S} \in \boldsymbol{\mathcal{Z}}$ is infeasible for (5) and/or $\mathcal{S} \in \boldsymbol{\mathcal{W}}$. To get a potential candidate $\mathcal{S}$ for this heuristic, we can randomly generate $\mathcal{S}$ such that $\mathcal{S} \notin \boldsymbol{\mathcal{Z}}$.

The heuristic is described as follows and summarized in Algorithm 2. First, from the logistic constraint, $\underline{w}$, $\bar{w}$, and $\boldsymbol{\mathcal{Z}}$ are initialized. Let p denote the iteration index and $q = \lceil(\underline{w} + \bar{w})/2\rceil$

Algorithm 2: Heuristic to Solve SSASP

initialize: $\mathcal{S}^* = (1)^{2N}$, $\boldsymbol{\mathcal{Z}} = \boldsymbol{\mathcal{W}}$, $\underline{w}$, and $\bar{w}$
set: $t = 1$, $p = 1$, $q = \lceil(\underline{w} + \bar{w})/2\rceil$
input: maxIter, maxInfeasibility
while $p \leq$ maxIter **and** $\underline{w} \leq q \leq \bar{w}$ **do**
  while $p \leq$ maxIter **and** $t \leq$ maxInfeasibility **do**
    compute: $\mathcal{S}_p^{(q)} \notin \boldsymbol{\mathcal{Z}}$ from Algorithm 3
    if $\mathcal{S}_p^{(q)} \neq (0)^{2N}$ **then**
      if (5) *is feasible* **then**
        $\mathcal{S}^* \leftarrow \mathcal{S}_p^{(q)}$, $\bar{w} \leftarrow q - 1$, $t \leftarrow 1$, $p \leftarrow p + 1$
        break
      else
        $\boldsymbol{\mathcal{Z}} \leftarrow \boldsymbol{\mathcal{Z}} \cup \{\mathcal{S}_p^{(q)}\}$, $t \leftarrow t + 1$, $p \leftarrow p + 1$
    else
      $t \leftarrow 1$
      break
  if $t >$ maxInfeasibility **then**
    $q \leftarrow \lceil(q + \bar{w})/2\rceil$, $t \leftarrow 1$
  else
    $q \leftarrow \lceil(\underline{w} + \bar{w})/2\rceil$
output: $\mathcal{S}^*$

Algorithm 3: Candidate Generation

initialize: $\mathcal{S}_p^{(q)} = (0)^{2N}$, $r = 1$
input: p, q, $\underline{w}$
while $r \leq$ maxRandom **do**
  Randomly generate $\mathcal{S}$ with $\mathcal{H}(\mathcal{S}) = q$
  if $\mathcal{S} \notin \boldsymbol{\mathcal{Z}}$ **then**
    $\mathcal{S}_p^{(q)} \leftarrow \mathcal{S}$
    break
  else
    $r \leftarrow r + 1$
    if $r >$ maxRandom **then**
      $\underline{w} \leftarrow q + 1$
output: $\mathcal{S}_p^{(q)}$, $\underline{w}$

denote the desired number of activated SA for the candidate $\mathcal{S}$ such that $\mathcal{H}(\mathcal{S}) = q$. Then, a candidate at iteration p with q number of activated SA can be denoted by $\mathcal{S}_p^{(q)}$. The next step is to generate a candidate $\mathcal{S}_p^{(q)}$ such that $\mathcal{S}_p^{(q)} \notin \boldsymbol{\mathcal{Z}}$. As mentioned earlier, one simple method to obtain $\mathcal{S}_p^{(q)}$ is to randomly generate $\mathcal{S}$ with q number of activated SA such that $\mathcal{S} \notin \boldsymbol{\mathcal{Z}}$—see Algorithm 3 for the detailed steps. If such candidate cannot be obtained after some combinations of SA have been randomly generated, then $\mathcal{S}_p^{(q)} \leftarrow (0)^{2N}$. When this happens, we can assert that the majority of combinations of SA with q or less than q number of activated SA most likely belong to the forbidden set $\boldsymbol{\mathcal{Z}}$. Given this condition, the required minimum number of activated SA can then be increased and updated. If $\mathcal{S}_p^{(q)}$ is nonzero, then we must check whether $\mathcal{S}_p^{(q)}$ is feasible for (5). If $\mathcal{S}_p^{(q)}$ is feasible for (5), we update $\mathcal{S}^* \leftarrow \mathcal{S}_p^{(q)}$; otherwise, update the forbidden set so that $\boldsymbol{\mathcal{Z}} \leftarrow \boldsymbol{\mathcal{Z}} \cup \{\mathcal{S}_p^{(q)}\}$. Unlike Algorithm 1, here we define maxInfeasibility that allows (5) to be solved repeatedly with different candidates while having the same number of activated SA. This process is repeated until there exists a candidate that makes (5) feasible or maxInfeasibility is reached. If (5) is still infeasible, we increase the required number of activated SA for the next candidate,

hoping that adding more activated SA will increase the chance for (5) being feasible. The algorithm continues and terminates when maximum iteration, denoted by maxIter, is reached or there is no more candidates that can be generated. At the end of Algorithm 2, the best suboptimal combination of SA is given as $\mathcal{S}^*$.

The algorithm in its nature allows the trade-off between the computational time and distance to optimality. This trade-off can be designed via selecting large values for maxRandom, maxInfeasibility, and maxIter. The parameter maxIter depends on how the user is willing to wait before the algorithm terminates, maxRandom imposes an upper bound on how many times a random SA candidate $\mathcal{S}$ is generated such that it is does not belong to the forbidden set. Finally, maxInfeasibility defines how many LMI feasibility problems are solved with a fixed number of activated SA.

7. Discussion on computational complexity

To discuss the computational complexity of the developed approaches, we start by discussing that of LMIs/SDPs. Primal–dual interior-point methods for SDPs have a worst-case complexity estimate of $\mathcal{O}\left(m^{2.75}L^{1.5}\right)$, where m is the number of variables and L is the number of constraints (Boyd, El Ghaoui, Feron, & Balakrishnan, 1994). The number of variables and constraints in (5) are

$$m = \underbrace{0.5n_x(n_x+1)}_{\text{entries in } \boldsymbol{P}} + \underbrace{n_u n_y}_{\text{entries in } \boldsymbol{N}} + \underbrace{n_u^2}_{\text{entries in } \boldsymbol{M}} \tag{11a}$$

$$L = \underbrace{0.5n_x(n_x+1)}_{\text{matrix inequality constraints}} + \underbrace{n_x n_u}_{\text{equality constraints}}. \tag{11b}$$

In various problems arising in control systems studies, it is shown that the complexity estimate is closer to $\mathcal{O}\left(m^{2.1}L^{1.2}\right)$ which is significantly smaller than the worst-case estimate $\mathcal{O}\left(m^{2.75}L^{1.5}\right)$ (Boyd et al., 1994). Hence, the complexity estimate for the LMIs is $\mathcal{O}\left(n_x^{4.2}n_x^{2.4}\right) = \mathcal{O}\left(n_x^{6.6}\right)$, since typically $n_x > n_u$ and $n_x > n_y$ in dynamic networks. Admittedly, these figures are outdated now as many advancements in interior-point methods are often implemented within newer versions of SDP solvers. Given that, the first approach in Theorem 1 entails solving the MI-SDP. Unfortunately, the worst-case complexity of solving MI-SDPs through branch-and-bound is $\mathcal{O}\left(2^{2N}n_x^{6.6}\right)$ as there are $2N$ SSA decision variables (for a network comprising N nodes). However, as branch-and-bound solvers almost always terminate way before trying all combinations, it is very difficult to obtain the best-case performance for this approach.

The second approach, namely Algorithm 1, entails solving a LMI feasibility problem at each iteration. The best case performance of this algorithm occurs when a feasible solution is always obtained for the LMI at each iteration of Algorithm 1. This yields a logarithmic reduction in the number of candidate optimal solutions. Hence, the best case complexity of running Algorithm 1 is of $\mathcal{O}\left(\log(2^{2N})n_x^{6.6}\right) = \mathcal{O}\left(N\,n_x^{6.6}\right) = \mathcal{O}\left(n_x^{7.6}\right)$. The worst-case performance of Algorithm 1 occurs when the LMI returns an infeasible solution at each iteration, which is hard to quantify. Unfortunately, it is virtually impossible to upper or lower bound the number of these combinations as this is system-dependent. Algorithm 2 also entails either solving an LMI feasibility problem, while depending on a maximum iteration number and thresholds. The computational complexity hence depends on the user-defined maximum iteration number and thresholds which is hard to estimate here.

8. Numerical experiments

Numerical experiments are presented here to test the proposed approaches on two dynamic networks. The first system is a random dynamic network adopted from Motee and Jadbabaie (2008) and Jovanović (0000), whereas the second is a mass–spring system (Lin, Fardad, & Jovanović, 2013). Both systems are initially unstable; the latter has a sparser structure than the former. All simulations are performed using MATLAB R2016b running on a 64-bit Windows 10 with 3.4 GHz Intel Core i7-6700 CPU and 16 GB of RAM. All optimization problems are solved using MOSEK version 8.1 (Andersen & Andersen, 2000) with YALMIP (Löfberg, 2004). All the MATLAB codes used in the paper are available for the interested reader upon request.

8.1. Comparing the proposed algorithms

In the first part of our numerical experiment, we focus on testing the performance of the approaches to solve SSASP on a relatively small dynamic network where optimality for the SSASP can be determined. Specifically, we consider the aforementioned random dynamic network with 10 subsystems, with two states per subsystem, so that 10 sensors and 10 actuators are available. Each sensor measures the two states per subsystem. We impose a logistic constraint so that there are at least one sensor and one actuator to be activated: $\sum_i^N \pi_i \geq 1$ and $\sum_j^N \gamma_j \geq 1$. In this particular experiment, the following scenarios are considered.

1. *MI-SDP-1*: The first scenario uses the results from Theorem 1 that is solved via YALMIP's MI-SDP branch and bound (BnB) solver (Integer Programming, 2015). We set $L_1 = 10^4$, $L_2 = 5 \times 10^6$, and $L_3 = 5 \times 10^6$.
2. *MI-SDP-2*: The second scenario is identical to the first one with the exception that $L_1 = 10^4$, $L_2 = 10^7$, and $L_3 = 10^7$. This scenario shows the impact of $L_{1,2,3}$ on the performance of the MI-SDP approach.
3. *BSA*: The third scenario directly follows Algorithm 1 and solves (5) in each iteration to check the feasibility of the given SA combinations, while also computing the SOF gain matrix from the solution of LMI (5).
4. *HEU*: The fourth scenario implements the heuristic (Algorithm 2). The parameters of the heuristic in this scenario are maxRandom $= 10^4$, maxInfeasibility $= 10$, and maxIter $= 50$. Since the proposed heuristic entails randomizations to generate SA candidates, we perform 500 randomizations. Then, from these randomizations, the mean values are computed.

Table 1 presents the result of this test. All scenarios successfully return solutions with stable closed-loop system eigenvalues. The MI-SDP approach yields the optimal solution as discussed in Theorem 1 and confirmed by BSA and Theorem 2. While MI-SDP-1 returns an optimal solution with a smaller computational time in comparison with BSA, the former is dependent on the choice of $L_{1,2,3}$. MI-SDP-2 solves the same problem for different values of $L_{1,2,3}$ and yields 5 activated SA—clearly a suboptimal solution. BSA here is advantageous in the sense that it does not require tuning to find the appropriate constants $L_{1,2,3}$. Also, BSA requires only an LMI solver, instead of a BnB solver for the MI-SDP. Out of the 500 randomizations, the heuristic yields an average of 3.42 activated SA as shown in Table 1. The HEU hence surprisingly yields optimal solutions with 3 activated SA in the majority of randomizations, while requiring a smaller computational time in comparison with BSA and a comparable computational time with the MI-SDP while not requiring any tuning for $L_{1,2,3}$.

Secondly, to find out how our methods perform on a larger system, we test our MI-SDP and heuristic (Algorithm 2) to solve

Table 1
Numerical test results on random network with $N = 10$, $n_x = 20$, $n_u = 10$ and $n_y = 20$. The symbol $\Delta t(s)$ represents the computational time measured in seconds. All methods successfully return a stable closed loop system, albeit close to the $j\omega$-axis; see Remark 1. The number of iterations for the MI-SDP corresponds to the BnB solver in YALMIP.

Scenario	$\bar{\lambda}_{\text{Re}}(\boldsymbol{A}+\boldsymbol{B\Pi}^*\boldsymbol{F\Gamma}^*\boldsymbol{C})$	$\Delta t(s)$	Iterations	$\sum_k \pi_k + \gamma_k$	$\boldsymbol{\gamma}^*$ and $\boldsymbol{\pi}^*$
MI-SDP-1	-4.78×10^{-3}	24.2	187	3	$\boldsymbol{\gamma}^* = \{0,0,0,0,0,0,1,0,0,0\}$, $\boldsymbol{\pi}^* = \{0,0,0,0,0,1,0,0,0,1\}$
MI-SDP-2	-6.71×10^{-3}	13.3	82	5	$\boldsymbol{\gamma}^* = \{1,1,0,0,0,0,0,0,0,0\}$, $\boldsymbol{\pi}^* = \{0,1,0,0,1,0,0,0,0,1\}$
BSA	-2.69×10^{-3}	61.8	198	3	$\boldsymbol{\gamma}^* = \{0,0,0,0,1,0,0,0,0,0\}$, $\boldsymbol{\pi}^* = \{1,0,0,0,0,1,0,0,0,0\}$
HEU[a]	-1.65×10^{-3}	27.4	50	3.42[b]	$\boldsymbol{\gamma}^* = \{1,0,0,0,0,0,0,0,0,0\}$, $\boldsymbol{\pi}^* = \{1,0,0,0,0,1,0,0,0,0\}$

[a]The displayed values are mean values of 500 randomizations. The corresponding binary configurations of SA are taken from the first randomization. All of the 500 randomizations return stable closed loop system.
[b]Out of the 500 randomizations, 294 return 3 activated SA, 198 return 4, and 8 randomizations return 5.

Table 2
Results for the MI-SDP and heuristic with $N = 50$, $n_x = 100$, $n_u = 50$, $n_y = 100$.

Method	$\bar{\lambda}_{\text{Re}}(\boldsymbol{A}+\boldsymbol{B\Pi}^*\boldsymbol{F\Gamma}^*\boldsymbol{C})$	$\Delta t(s)$	$\sum_k \pi_k + \gamma_k$
MI-SDP	-7.59×10^{-3}	13749.9	9
Heuristic	-1.68×10^{-2}	6181.5	17.39[a]

[a]Out of the 10 randomizations, 4 return 3 activated SA while the remaining 6 return various number of activated SA each ranging from 11 to 23.

a larger random dynamic network consisting 50 subsystems with $n_x = 100$, $n_u = 50$, and $n_y = 100$. For MI-SDP, we set a maximum of 1000 iterations for the BnB solver. As for heuristic, we perform 10 randomizations with maxRandom $= 10^6$, maxInfeasibility $= 10$, and maxIter $= 200$. The result of this experiment is summarized in Table 2. The MI-SDP via BnB algorithm returns 9 activated SA in total (3 sensors and 6 actuators) while requiring 112 iterations and 1.38×10^4 s $\approx$ 3.8 h of computational time which we presume is the optimal solution. On the other hand, we obtain an average of 17.39 activated SA with 6.18×10^3 s $\approx$ 1.7 h of average computational time. Realize that the total available SA are 50 sensors and 50 actuators, which implies that there are $2^{100} \approx 1.26 \times 10^{30}$ combinations of SA. With the average number of activated SA being relatively small compared to the number of available SA, we conclude that the heuristic produces a reasonably good solution—in comparison with the MI-SDP solution. This experiment demonstrates the tradeoff between MI-SDP and heuristic.

8.2. Comparative study with dynamic output feedback

In the second part of the numerical experiment, we consider the mass–spring systems from Lin et al. (2013) of various sizes to measure the performance of our heuristic (Algorithm 2) relative to the other method by comparing it with the sparsity promoting algorithm (SPA) for $\mathcal{H}_\infty$ dynamic output feedback developed in Singh et al. (2018). The heuristic algorithm is configured in a way such that maxRandom $= 10^6$, maxInfeasibility $= 10$, and maxIter $= 200$. For each number of nodes N, we perform 10 randomizations for the heuristic in which the mean value of maximum real part of closed loop eigenvalues, computational time, and the number of activated SA are computed accordingly. The SPA is set up so that the maximum iteration number is 50 and the convergence tolerance is 0.5. The results are given in Table 3.

From this experiment, we observe that both methods are able to give stable closed-loop systems with the heuristic returning fewer activated SA than SPA. The computational time of the heuristic is significantly faster than SPA's and the latter returns many more activated SA. For example, if we consider the particular case of $N = 50$, the heuristic returns 2.30 of activated SA on average, while SPA returns 26. These findings, however, should not conflated with the objectives of the $\mathcal{H}_\infty$ and SOFC methods, seeing that both methods consider different control metrics

Table 3
Numerical comparison results between the heuristic and SPA with mass–spring systems. Notations: $\boldsymbol{A}_{cl} = \boldsymbol{A} + \boldsymbol{B\Pi}^*\boldsymbol{F\Gamma}^*\boldsymbol{C}$; $\tilde{\boldsymbol{A}}_{cl}$ denotes the overall closed-loop dynamics matrix of the plant with DOFC; and $\boldsymbol{\Sigma} = \sum_{k=1}^{N} \pi_k + \gamma_k$.

N	Heuristic			SPA		
	$\bar{\lambda}_{\text{Re}}(\boldsymbol{A}_{cl})$	$\Delta t(s)$	$\boldsymbol{\Sigma}$	$\bar{\lambda}_{\text{Re}}(\tilde{\boldsymbol{A}}_{cl})$	$\Delta t(s)$	$\boldsymbol{\Sigma}$
10	-3.3×10^{-3}	6.9	2	-2.7×10^{-1}	10.4	10
20	-4.6×10^{-4}	19.5	2	-3.4×10^{-1}	129.1	20
30	-1.5×10^{-4}	98.9	2	-8.2×10^{-2}	1383.4	24
40	-6.1×10^{-5}	406.0	2	-2.6×10^{-1}	3844.7	40
50	-2.3×10^{-5}	1736.7	2.30	-5.8×10^{-2}	28513.9	26

(static output feedback considers pure stabilization whereas dynamic output feedback with $\mathcal{H}_\infty$ control considers robustness) with complexity. The results shown here are meant to give an indication that when robustness is considered as a metric through DOFC and SA selection, the corresponding problem requires more computational time, the activation of more SA, yet returns a closed-loop system that is more robust to disturbances.

9. Summary, limitations, and future directions

In this paper, we propose computational methods to solve the simultaneous sensor and actuator selection problem (SSASP) through static output feedback control framework. Three different approaches to obtain the minimal selection of activated SA that yield stable closed-loop systems are proposed. The first approach utilizes disjunctive programming principles and linear algebra techniques to convert the mixed-integer nonconvex problem into a MI-SDP. The second approach uses a simple algorithm that is akin to the binary search algorithm. The third approach is a simple heuristic that constructs a dynamic data structure with infeasible combinations.

The first two approaches yield optimal solutions for the SSASP, while the third yields suboptimal results while resulting in an improved computational time. In particular, the first approach requires finding suitable constants, namely $L_{1,2,3}$, to obtain an optimal solution to the MI-SDP and hence the SSASP. This is for combinatorial optimization problems that use the Big-M method or the McCormick Relaxation. The second optimal approach requires efficient data structures to store and update the feasible combinations of sensors and actuators, without requiring any tuning parameters. The third approach is suitable for larger networks as it trades optimality with improved computational time. It is noteworthy to mention that the second and third algorithms only require an LMI solver making it easier to interface with without the need to install any additional optimization packages or tune any parameters.

The limitations of the proposed methods in this paper are listed as follows. First, we do not consider any robustness or energy metrics through SOFC and SSASP. For example, an interesting

extension can capture the minimal SA selection alongside designing energy-aware and robust output feedback control laws for the activated nodes. Second, our approach still requires solving LMIs. Albeit the LMIs solved in this paper are simple with few optimization variables and only one block, and while the selection/placement problem can be performed offline, the proposed approaches do *not* scale graciously when large-scale dynamic networks with tens of thousands of nodes are studied.

To that end, future work will focus on addressing the aforementioned paper's limitations by deriving SOFC energy and robustness metrics with the SSASP, and consequently examining the performance of the presented algorithms in this paper. Furthermore, and to address the computational burden of solving many LMIs, we plan to investigate algebraic conditions on the existence of SOFC given a fixed SA selection. The idea here is to avoid solving LMI feasibility problem at each iteration. Instead, we can learn the feasibility of specific SA selection using these algebraic conditions. Finally, and instead of solving the MI-SDP form of SSASP using the classical BnB algorithm, we plan to test the performance of the outer approximations (Lubin et al., 2016) and the branch-and-cut algorithm (Kobayashi & Takano, 2018)—as these approaches have shown significant savings for the computational time when compared with the classical BnB methods.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grants 1728629 and 1728605.

Appendix A. Proof of Lemma 1

Proof. Let $\boldsymbol{M}$, $\boldsymbol{P}$, and $\boldsymbol{B}$ satisfy $\boldsymbol{BM} = \boldsymbol{PB}$. Then, suppose that $\boldsymbol{P} \succ 0$ and $\text{Rank}(\boldsymbol{B}) = m$. Since $\boldsymbol{B}$ is full column rank, $\boldsymbol{B}^\top\boldsymbol{B}$ is nonsingular. Premultiplying both sides of $\boldsymbol{BM} = \boldsymbol{PB}$ with $\boldsymbol{B}^\top$ yields $\boldsymbol{B}^\top\boldsymbol{BM} = \boldsymbol{B}^\top\boldsymbol{PB}$, which is equivalent to $\boldsymbol{M} = (\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{PB}$. Since $\boldsymbol{P} \succ 0$ and $\text{Rank}(\boldsymbol{B}) = m$, by Horn and Johnson (2013, Observation 7.1.8), we have $\boldsymbol{B}^\top\boldsymbol{PB} \succ 0$. Thus $\boldsymbol{M}^{-1} = (\boldsymbol{B}^\top\boldsymbol{PB})^{-1}\boldsymbol{B}^\top\boldsymbol{B}$. ■

Appendix B. Proof of Proposition 3

Proof. Without loss of generality, $\boldsymbol{B\Pi}^*$ and $\boldsymbol{C}^\top\boldsymbol{\Gamma}^*$ can be expressed as $\boldsymbol{B\Pi}^* = \begin{bmatrix}\hat{\boldsymbol{B}} & \boldsymbol{O}\end{bmatrix}$ and $\boldsymbol{C}^\top\boldsymbol{\Gamma}^* = \begin{bmatrix}\hat{\boldsymbol{C}}^\top & \boldsymbol{O}\end{bmatrix}$. Then, $\boldsymbol{M}$ and $\boldsymbol{N}$ can be partitioned as

$$\boldsymbol{M} = \begin{bmatrix}\boldsymbol{M}_1 & \boldsymbol{M}_2\\ \boldsymbol{M}_3 & \boldsymbol{M}_4\end{bmatrix}, \quad \boldsymbol{N} = \begin{bmatrix}\boldsymbol{N}_1 & \boldsymbol{N}_2\\ \boldsymbol{N}_3 & \boldsymbol{N}_4\end{bmatrix},$$

where $\boldsymbol{M}_1 \in \mathbb{R}^{m\times m}$ and $\boldsymbol{N}_1 \in \mathbb{R}^{m\times r}$. By letting $\hat{\boldsymbol{N}} = \boldsymbol{N}_1$, (7b) can be expressed as

$$\begin{aligned}\boldsymbol{A}^\top\boldsymbol{P} + \boldsymbol{PA} + \begin{bmatrix}\hat{\boldsymbol{C}}^\top & \boldsymbol{O}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{N}}^\top & \boldsymbol{N}_3^\top\\ \boldsymbol{N}_2^\top & \boldsymbol{N}_4^\top\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{B}}^\top\\ \boldsymbol{O}\end{bmatrix}\\ + \begin{bmatrix}\hat{\boldsymbol{B}} & \boldsymbol{O}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{N}} & \boldsymbol{N}_2\\ \boldsymbol{N}_3 & \boldsymbol{N}_4\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{C}}\\ \boldsymbol{O}\end{bmatrix} \prec 0\\ \Leftrightarrow \boldsymbol{A}^\top\boldsymbol{P} + \boldsymbol{PA} + \hat{\boldsymbol{C}}^\top\hat{\boldsymbol{N}}^\top\hat{\boldsymbol{B}}^\top + \hat{\boldsymbol{B}}\hat{\boldsymbol{N}}\hat{\boldsymbol{C}} \prec 0. \qquad \text{(B.1)}\end{aligned}$$

Since (7b) is feasible for $\boldsymbol{P}$ and $\boldsymbol{N}$, then (B.1) is also feasible. Similarly, letting $\hat{\boldsymbol{M}} = \boldsymbol{M}_1$ allows (7c) to be

$$\begin{aligned}&\begin{bmatrix}\hat{\boldsymbol{B}} & \boldsymbol{O}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{M}} & \boldsymbol{M}_2\\ \boldsymbol{M}_3 & \boldsymbol{M}_4\end{bmatrix} = \boldsymbol{P}\begin{bmatrix}\hat{\boldsymbol{B}} & \boldsymbol{O}\end{bmatrix}\\ &\Leftrightarrow \begin{bmatrix}\hat{\boldsymbol{B}}\hat{\boldsymbol{M}} & \hat{\boldsymbol{B}}\boldsymbol{M}_2\end{bmatrix} = \begin{bmatrix}\boldsymbol{P}\hat{\boldsymbol{B}} & \boldsymbol{O}\end{bmatrix}. \qquad \text{(B.2)}\end{aligned}$$

Realize that (B.2) holds since we assume that (7c) holds. From (B.2), we have $\hat{\boldsymbol{B}}\hat{\boldsymbol{M}} = \boldsymbol{P}\hat{\boldsymbol{B}}$. Since $\hat{\boldsymbol{B}}$ is full column rank and $\boldsymbol{P} \succ 0$, by Lemma 1, $\hat{\boldsymbol{M}}$ is nonsingular. Finally, having $\boldsymbol{P} \succ 0$, $\hat{\boldsymbol{M}}$, and $\hat{\boldsymbol{N}}$ that satisfy (B.1) and $\hat{\boldsymbol{B}}\hat{\boldsymbol{M}} = \boldsymbol{P}\hat{\boldsymbol{B}}$, then according to Proposition 2, the closed loop system $\boldsymbol{A} + \hat{\boldsymbol{B}}\boldsymbol{F}\hat{\boldsymbol{C}}$ is stable with $\boldsymbol{F} = \hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{N}}$. ■

Appendix C. Proof of Theorem 1

Proof. Let $(\boldsymbol{\Pi N\Gamma})_{ij}$ be the (i,j) element of $\boldsymbol{\Pi N\Gamma}$ and $\{\pi_i,\ \gamma_j\}$ be the associated SA selection that corresponds to $(\boldsymbol{\Pi N\Gamma})_{ij}$. Then, there exists $\boldsymbol{\Theta} \in \mathbb{R}^{n_u\times n_y}$ such that $\boldsymbol{\Theta} = \boldsymbol{\Pi N\Gamma}$. This relation is established as follows. Realize that, as $\boldsymbol{\Pi}$ and $\boldsymbol{\Gamma}$ are symmetric diagonal matrices with binary values, we can write $(\boldsymbol{\Pi N\Gamma})_{ij}$ as

$$(\boldsymbol{\Pi N\Gamma})_{ij} = \begin{cases}\boldsymbol{N}_{ij}, & \text{if} \quad \pi_i \wedge \gamma_j = 1\\ 0, & \text{if} \quad \pi_i \wedge \gamma_j = 0.\end{cases}$$

for $i = 1,\dots,n_u$ and $j = 1,\dots,n_y$. That is, if $\{\pi_i,\ \gamma_j\} = \{1,\ 1\}$, then $\boldsymbol{N}_{ij} = \boldsymbol{\Theta}_{ij}$. Otherwise, $\boldsymbol{N}_{ij} \in \mathbb{R}$ and $\boldsymbol{\Theta}_{ij} = 0$. For an appropriate large constant L_1, this is equivalent to

$$\begin{aligned}&|\boldsymbol{\Theta}_{ij}| \le L_1\pi_i,\ |\boldsymbol{\Theta}_{ij}| \le L_1\gamma_j\\ &|\boldsymbol{\Theta}_{ij} - \boldsymbol{N}_{ij}| \le L_1(2 - \pi_i - \gamma_j),\end{aligned}$$

where $|\cdot|$ denotes the absolute value function. The above equations can be represented as $\boldsymbol{\Psi}_1(\boldsymbol{N},\boldsymbol{\Theta}) \le L_1\boldsymbol{\Delta}_1(\boldsymbol{\Gamma},\boldsymbol{\Pi})$ where $\boldsymbol{\Psi}_1(\boldsymbol{N},\boldsymbol{\Theta})$ and $\boldsymbol{\Delta}_1(\boldsymbol{\Gamma},\boldsymbol{\Pi})$ are given in (9a) and (9b). This establishes (8c). Consequently, $\boldsymbol{\Pi N\Gamma}$ in (7b) can be replaced with $\boldsymbol{\Theta}$, giving (8b). Now, pre-multiplying both sides of (7c) with $\boldsymbol{BB}^\dagger$ ($\boldsymbol{B}^\dagger$ denotes the pseudoinverse of $\boldsymbol{B}$) and, since $\boldsymbol{BB}^\dagger\boldsymbol{B} = \boldsymbol{B}$, we obtain

$$\boldsymbol{BB}^\dagger\boldsymbol{B\Pi M} = \boldsymbol{BB}^\dagger\boldsymbol{PB\Pi} \qquad \text{(C.1a)}$$

$$\Leftrightarrow \boldsymbol{B\Pi M} = \boldsymbol{BB}^\dagger\boldsymbol{PB\Pi}. \qquad \text{(C.1b)}$$

Since $\boldsymbol{B}$ is full column rank, $\boldsymbol{B}^\dagger$ can be expressed as $\boldsymbol{B}^\dagger = (\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top$. Then by the same reason, (C.1b) implies

$$\boldsymbol{\Pi M} = \boldsymbol{B}^\dagger\boldsymbol{PB\Pi} \Leftrightarrow \boldsymbol{\Pi M} = (\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{PB\Pi}. \qquad \text{(C.2)}$$

By using the definition of $\boldsymbol{\Omega}(\boldsymbol{P})$ given in (10a), allows (7c) to be replaced by $\boldsymbol{\Pi M} = \boldsymbol{\Omega}(\boldsymbol{P})\boldsymbol{\Pi}$. Next, consider $\boldsymbol{\Pi}_i\boldsymbol{M}_{ij} = \boldsymbol{\Omega}(\boldsymbol{P})_{ij}\boldsymbol{\Pi}_j$ as the (i,j) element of $\boldsymbol{\Pi M} = \boldsymbol{\Omega}(\boldsymbol{P})\boldsymbol{\Pi}$. Then, we have

$$\boldsymbol{\Pi}_i\boldsymbol{M}_{ij} = \boldsymbol{\Omega}(\boldsymbol{P})_{ij}\boldsymbol{\Pi}_j \Leftrightarrow \boldsymbol{\Pi}_i\boldsymbol{M}_{ij} - \boldsymbol{\Pi}_j\boldsymbol{\Omega}(\boldsymbol{P})_{ij} = 0, \qquad \text{(C.3)}$$

such that

$$\text{(C.3)} \Leftrightarrow \begin{cases}\boldsymbol{M}_{ij} = \boldsymbol{\Omega}(\boldsymbol{P})_{ij}, & \text{if} \quad \pi_i = 1,\ \pi_j = 1\\ \boldsymbol{M}_{ij} = 0,\ \boldsymbol{\Omega}(\boldsymbol{P})_{ij} \in \mathbb{R}, & \text{if} \quad \pi_i = 1,\ \pi_j = 0\\ \boldsymbol{M}_{ij} \in \mathbb{R},\ \boldsymbol{\Omega}(\boldsymbol{P})_{ij} = 0, & \text{if} \quad \pi_i = 0,\ \pi_j = 1\\ \boldsymbol{M}_{ij},\ \boldsymbol{\Omega}(\boldsymbol{P})_{ij} \in \mathbb{R}, & \text{if} \quad \pi_i = 0,\ \pi_j = 0.\end{cases}$$

for all $i,j = 1,\dots,n_u$. For an appropriate large constant L_2, the above is equivalent to

$$\begin{aligned}&|\boldsymbol{M}_{ij}| \le L_2(1 - \pi_i + \pi_j), \quad |\boldsymbol{\Omega}(\boldsymbol{P})_{ij}| \le L_2(1 + \pi_i - \pi_j)\\ &|\boldsymbol{M}_{ij} - \boldsymbol{\Omega}(\boldsymbol{P})_{ij}| \le L_2(2 - \pi_i - \pi_j),\end{aligned}$$

that can be expressed as $\boldsymbol{\Psi}_2(\boldsymbol{M},\boldsymbol{\Omega}(\boldsymbol{P})) \le L_2\boldsymbol{\Delta}_2(\boldsymbol{\Pi})$ in which $\boldsymbol{\Psi}_2(\boldsymbol{M},\boldsymbol{\Omega}(\boldsymbol{P}))$ and $\boldsymbol{\Delta}_2(\boldsymbol{\Pi})$ are given in (9c) and (9d). This establishes (8d). Finally, since the left-hand side of (7c) and (C.1b) are equal, then (Skelton, Iwasaki, & Grigoriadis, 2013, Theorem 2.3.1)

$$\begin{aligned}&\boldsymbol{PB\Pi} = \boldsymbol{BB}^\dagger\boldsymbol{PB\Pi} \Leftrightarrow \boldsymbol{PB\Pi} = \boldsymbol{B}(\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{PB\Pi}\\ &\Leftrightarrow \boldsymbol{O} = (\boldsymbol{I} - \boldsymbol{B}(\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top)\boldsymbol{PB\Pi}. \qquad \text{(C.4)}\end{aligned}$$

By using the definition of $\boldsymbol{\Xi}(\boldsymbol{P})$ as in (10b), we get $\boldsymbol{\Xi}(\boldsymbol{P})\boldsymbol{\Pi} = \boldsymbol{O}$. Let $\boldsymbol{\Xi}(\boldsymbol{P})_{ij}\boldsymbol{\Pi}_j$ be the (i,j) element of $\boldsymbol{\Xi}(\boldsymbol{P})\boldsymbol{\Pi}$. Then, this constraint is equivalent to

$$\boldsymbol{\Xi}(\boldsymbol{P})_{ij}\boldsymbol{\Pi}_j = 0 \Leftrightarrow \begin{cases}\boldsymbol{\Xi}(\boldsymbol{P})_{ij} = 0, & \text{if} \quad \pi_j = 1\\ \boldsymbol{\Xi}(\boldsymbol{P})_{ij} \in \mathbb{R}, & \text{if} \quad \pi_j = 0,\end{cases}$$

for $i = 1, \ldots, n_x$ and $j = 1, \ldots, n_u$. For an appropriate large constant L_3, the above is equivalent to $|\Xi(\boldsymbol{P})_{ij}| \leq L_3(1 - \pi_j)$ such that we obtain $\Psi_3(\Xi(\boldsymbol{P})) \leq L_3\Delta_3(\Pi)$, where $\Psi_3(\Xi(\boldsymbol{P}))$ and $\Delta_3(\Pi)$ are given in (9e) and (9f). This establishes (8e).

The equivalence between (7) and (8) is now summarized. For any feasible $\{\boldsymbol{\pi}, \boldsymbol{\gamma}, \boldsymbol{N}, \boldsymbol{M}, \boldsymbol{P}\}$ that satisfies (7), by constructing Θ such that $\Psi_1(\boldsymbol{N}, \Theta) \leq L_1\Delta_1(\Gamma, \Pi)$ for a sufficiently large L_1, we get $\Pi\boldsymbol{N}\Gamma = \Theta$. Substituting $\Pi\boldsymbol{N}\Gamma = \Theta$ into (7b) yields (8b). Next, since $\boldsymbol{B}$ is full column rank, $\boldsymbol{B}^\top\boldsymbol{B}$ is nonsingular. By using the Moore–Penrose pseudoinverse of $\boldsymbol{B}$ given as $\boldsymbol{B}^\dagger = (\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top$, pre-multiplying both sides of (7c) with $\boldsymbol{B}\boldsymbol{B}^\dagger$ yields (C.2). By computing $\Omega(\boldsymbol{P})$ using (10a) such that $\Pi\boldsymbol{M} = \Omega(\boldsymbol{P})\Pi$, we get $\boldsymbol{M}$ and $\Omega(\boldsymbol{P})$ satisfy $\Psi_2(\boldsymbol{M}, \Omega(\boldsymbol{P})) \leq L_2\Delta_2(\Pi)$ for a sufficiently large L_2. Then, $\Xi(\boldsymbol{P})$ can be computed as (10b). Since we have (C.4), $\Xi(\boldsymbol{P})$ must satisfy $\Psi_3(\Xi) \leq L_3\Delta_3(\Pi)$ for a large constant L_3. Therefore, $\{\boldsymbol{\pi}, \boldsymbol{\gamma}, \boldsymbol{N}, \boldsymbol{M}, \boldsymbol{P}, \Theta, \Omega(\boldsymbol{P}), \Xi(\boldsymbol{P})\}$ is feasible for (8). Conversely, given sufficiently large constants L_1, L_2, and L_3, we always have $\Theta = \Pi N\Gamma$, $\Pi\boldsymbol{M} = \Omega(\boldsymbol{P})\Pi$ with $\Omega(\boldsymbol{P})$ satisfying (10a), and $\Xi(\boldsymbol{P})\Pi = \boldsymbol{O}$ with $\Xi(\boldsymbol{P})$ satisfying (10b) for any feasible $\{\boldsymbol{\pi}, \boldsymbol{\gamma}, \boldsymbol{N}, \boldsymbol{M}, \boldsymbol{P}, \Theta, \Omega(\boldsymbol{P}), \Xi(\boldsymbol{P})\}$ that satisfies (8). Substituting $\Theta = \Pi N\Gamma$ into (8b) and (10a) into $\Pi\boldsymbol{M} = \Omega(\boldsymbol{P})\Pi$ yield (7b) and $\Pi\boldsymbol{M} = (\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top\boldsymbol{P}\boldsymbol{B}\Pi$. The fact that $\boldsymbol{B}$ being full column rank implies $(\boldsymbol{B}^\top\boldsymbol{B})^{-1}\boldsymbol{B}^\top = \boldsymbol{B}^\dagger$ so that we have $\Pi\boldsymbol{M} = \boldsymbol{B}^\dagger\boldsymbol{P}\boldsymbol{B}\Pi$. Then, pre-multiplying both sides of $\Pi\boldsymbol{M} = \boldsymbol{B}^\dagger\boldsymbol{P}\boldsymbol{B}\Pi$ with $\boldsymbol{B}$ yields $\boldsymbol{B}\Pi\boldsymbol{M} = \boldsymbol{B}\boldsymbol{B}^\dagger\boldsymbol{P}\boldsymbol{B}\Pi$. Then, substituting (10b) into $\Xi(\boldsymbol{P})\Pi = \boldsymbol{O}$ yields (C.4), which implies that $\boldsymbol{B}\boldsymbol{B}^\dagger\boldsymbol{P}\boldsymbol{B}\Pi = \boldsymbol{P}\boldsymbol{B}\Pi$ and finally $\boldsymbol{B}\Pi\boldsymbol{M} = \boldsymbol{P}\boldsymbol{B}\Pi$, which is (7c). Hence, $\{\boldsymbol{\pi}, \boldsymbol{\gamma}, \boldsymbol{N}, \boldsymbol{M}, \boldsymbol{P}\}$ is feasible for (7). This completes the proof. ■

Appendix D. Proof of Theorem 2

Proof. Assume that $\boldsymbol{\mathcal{V}} \neq \emptyset$ (that is, SSASP has a solution) and $\boldsymbol{\mathcal{S}}_p$ be the candidate set at iteration p and $\mathcal{S}_q \in \boldsymbol{\mathcal{S}}_p$ with $q = \lceil \sigma/2 \rceil$ and $\sigma = |\boldsymbol{\mathcal{S}}_p|$. Also, let $\mathcal{S}_p^*$ be the best known solution at iteration p. If $\mathcal{S}_q$ is infeasible for (5), then $\mathcal{S}_p^* = \mathcal{S}_{p-1}^*$ and, considering that for practical systems, if a set of selected sensors or actuators renders (5) infeasible, then a subset thereof should also render (5) infeasible, the candidate set is updated such that $\boldsymbol{\mathcal{S}}_{p+1} = \boldsymbol{\mathcal{S}}_p \setminus \{\mathcal{S} \in \boldsymbol{\mathcal{S}}_p \mid \mathcal{S}_q \vee \mathcal{S} = \mathcal{S}_q\}$. However, assume that $\mathcal{S}_q$ is feasible for (5), then according to Algorithm 1, $\mathcal{S}_p^* = \mathcal{S}_q$. In this case, for all $\mathcal{S} \in \boldsymbol{\mathcal{U}}_p$ where $\boldsymbol{\mathcal{U}}_p \triangleq \{\mathcal{S} \in \boldsymbol{\mathcal{S}}_p \mid \mathcal{H}(\mathcal{S}) \geq \mathcal{H}(\mathcal{S}_q)\}$, we have $\mathcal{H}(\mathcal{S}_p^*) \leq \mathcal{H}(\mathcal{S})$. However, since $\boldsymbol{\mathcal{V}}_p \subseteq \boldsymbol{\mathcal{U}}_p$ where $\boldsymbol{\mathcal{V}}_p \triangleq \{\mathcal{S} \in \boldsymbol{\mathcal{U}}_p \mid \mathcal{S} \textit{ is feasible for } (5)\}$, we have $\mathcal{H}(\mathcal{S}_p^*) \leq \mathcal{H}(\mathcal{S})$ for all $\mathcal{S} \in \boldsymbol{\mathcal{V}}_p$. Then, the candidate set is updated such that $\boldsymbol{\mathcal{S}}_{p+1} = \boldsymbol{\mathcal{S}}_p \setminus \boldsymbol{\mathcal{U}}_p$ and the algorithm proceeds. Accordingly, σ and q are updated such that $\sigma = |\boldsymbol{\mathcal{S}}_{p+1}|$ and $q = \lceil \sigma/2 \rceil$. If $\mathcal{S}_q$ is infeasible for (5), where $\mathcal{S}_q \in \boldsymbol{\mathcal{S}}_{p+1}$, then $\mathcal{S}_{p+1}^* = \mathcal{S}_p^*$. Nonetheless, if $\mathcal{S}_q$ is feasible for (5), then according to Algorithm 1, $\mathcal{S}_{p+1}^* = \mathcal{S}_q$.

In the latter case, we have the fact that $\mathcal{H}(\mathcal{S}_{p+1}^*) < \mathcal{H}(\mathcal{S}_p^*)$ since $\mathcal{H}(\mathcal{S}) < \mathcal{H}(\hat{\mathcal{S}})$ for all $\mathcal{S} \in \boldsymbol{\mathcal{S}}_{p+1}$ and $\hat{\mathcal{S}} \in \boldsymbol{\mathcal{S}}_p$. This shows that at any iteration we have $\mathcal{H}(\mathcal{S}_{p+1}^*) \leq \mathcal{H}(\mathcal{S}_p^*)$. Let l denote the index of last iteration. This implies that the sequence $(\mathcal{H}(\mathcal{S}_p^*))_{p=1}^{l}$ is decreasing. Therefore, when $\boldsymbol{\mathcal{S}}_l = \emptyset$, $\mathcal{H}(\mathcal{S}_l^*) \leq \mathcal{H}(\mathcal{S})$ for all $\mathcal{S} \in \boldsymbol{\mathcal{V}}$. This shows that Algorithm 1 computes an optimal solution of SSASP. ■

References

Andersen, E. D., & Andersen, K. D. (2000). The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization* (pp. 197–232). Springer.

Argha, A., Su, S. W., Savkin, A., & Celler, B. (2017). A framework for optimal actuator/sensor selection in a control system. *International Journal of Control*, 1–19. http://dx.doi.org/10.1080/00207179.2017.1350755.

Astolfi, A., & Colaneri, P. (2000). Static output feedback stabilization of linear and nonlinear systems. In *Proceedings of the 39th IEEE conference on decision and control (Cat. No.00CH37187), vol. 3* (pp. 2920–2925 vol.3). http://dx.doi.org/10.1109/CDC.2000.914257.

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

Crusius, C. A. R., & Trofino, A. (1999). Sufficient LMI conditions for output feedback control problems. *IEEE Transactions on Automatic Control*, *44*(5), 1053–1057. http://dx.doi.org/10.1109/9.763227.

De Oliveira, M. C., & Geromei, J. (2000). Linear output feedback controller design with joint selection of sensors and actuators. *IEEE Transactions on Automatic Control*, *45*(12), 2412–2419.

Gally, T., Pfetsch, M. E., & Ulbrich, S. (2017). A framework for solving mixed-integer semidefinite programs. *Optimization Methods & Software*, 1–39.

Gamrath, G., Fischer, T., Gally, T., Gleixner, A. M., Hendel, G., Koch, T., et al. (2016). The scip optimization suite 3.2, ZIB Report,

Horn, R. A., & Johnson, C. R. (2013). *Matrix Analysis* (2nd ed.). New York, NY: Cambridge University Press.

Integer Programming, YALMIP Wiki, (2015). http://users.isy.liu.se/johanl/yalmip/pmwiki.php?n=Tutorials.IntegerProgramming.

Jovanović, M. Network with 100 unstable nodes, http://people.ece.umn.edu/~mihailo/software/lqrsp/dist_sys.html.

Kobayashi, K., & Takano, Y. A branch-and-cut algorithm for solving mixed-integer semidefinite optimization problems, http://www.optimization-online.org/DB_FILE/2018/09/6808.pdf.

Kučera, V., & Souza, C. D. (1995). A necessary and sufficient condition for output feedback stabilizability. *Automatica*, *31*(9), 1357–1359. http://dx.doi.org/10.1016/0005-1098(95)00048-2, URL http://www.sciencedirect.com/science/article/pii/0005109895000482.

Lin, F., Fardad, M., & Jovanović, M. R. (2013). Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, *58*(9), 2426–2431.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proc. IEEE int. symp. computer aided control systems design* (pp. 284–289). IEEE.

Lubin, M., Yamangil, E., Bent, R., & Vielma, J. P. (2016). *Extended Formulations in Mixed-Integer Convex Programming* (pp. 102–113). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-33461-5_9,

Motee, N., & Jadbabaie, A. (2008). Optimal control of spatially distributed systems. *IEEE Transactions on Automatic Control*, *53*(7), 1616–1629. http://dx.doi.org/10.1109/TAC.2008.929366.

Nugroho, S. A., Taha, A. F., Gatsis, N., Summers, T. H., & Krishnan, R. Algorithms for joint sensor and control nodes selection in dynamic networks, arXiv preprint arXiv:1811.11792.

Nugroho, S., Taha, A. F., Summers, T., & Gatsis, N. (2018). Simultaneous sensor and actuator selection/placement through output feedback control. In *2018 annual American control conference (ACC)* (pp. 4159–4164). http://dx.doi.org/10.23919/ACC.2018.8431548.

Pequito, S. D., Kar, S., & Pappas, G. J. (2015). Minimum cost constrained input-output and control configuration co-design problem: A structural systems approach. In *2015 American control conference (ACC)* (pp. 4099–4105).

Peretz, Y. J. (2016). A randomized approximation algorithm for the minimal-norm static-output-feedback problem. *Automatica*, *63*, 221–234. http://dx.doi.org/10.1016/j.automatica.2015.10.001, URL http://www.sciencedirect.com/science/article/pii/S000510981500401X.

Singh, T., Swevers, J., & Pipeleers, G. (2018). Concurrent H_2/H_∞ feedback control design with optimal sensor and actuator selection. In *2018 IEEE 15th international workshop on advanced motion control (AMC)* (pp. 223–228). http://dx.doi.org/10.1109/AMC.2019.8371092.

Skelton, R., Iwasaki, T., & Grigoriadis, K. M. (2013). A unified algebraic approach to linear control design, https://www.researchgate.net/publication/225075682.

Syrmos, V. L., Abdallah, C. T., Dorato, P., & Grigoriadis, K. (1997). Static output feedbacka survey. *Automatica*, *33*(2), 125–137.

Taha, A., Gatsis, N., Summers, T. H., & Nugroho, S. A. (2018). Time-varying sensor and actuator selection for uncertain cyber-physical systems. *IEEE Transactions on Control of Network Systems*, http://dx.doi.org/10.1109/TCNS.2018.2873229, 1–1.

Sebastian A. Nugroho was born in Yogyakarta, Indonesia in 1990. He received the B.S. and M.S. degrees in Electrical Engineering from Institut Teknologi Bandung (ITB), Indonesia in 2012 and 2014. He is currently a graduate research assistant and a Ph.D. candidate with the ECE department at the University of Texas, San Antonio. His research interests are in control theory and engineering optimization with applications to cyber–physical systems.

Ahmad F. Taha is an assistant professor with the Department of Electrical and Computer Engineering at the University of Texas, San Antonio. He received the B.E. and Ph.D. degrees in Electrical and Computer Engineering from the American University of Beirut, Lebanon in 2011 and Purdue University, West Lafayette, Indiana in 2015. Dr. Taha is interested in understanding how complex cyber–physical systems (CPS) operate, behave, and *misbehave*. His research focus includes optimization, control, and state estimation in CPSs with applications to power, water, and traffic networks. Dr. Taha is an editor of IEEE Transactions on Smart Grid and the Control Systems Society Electronic Publications chair.

Nikolaos Gatsis received the Diploma degree in Electrical and Computer Engineering from the University of Patras, Greece, in 2005 with honors. He completed his graduate studies at the University of Minnesota, where he received the M.Sc. degree in Electrical Engineering in 2010, and the Ph.D. degree in Electrical Engineering with minor in Mathematics in 2012. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering at the University of Texas at San Antonio. His research focuses on optimal and secure operation of smart power grids and other critical infrastructures, including water distribution networks and the Global Positioning System. Dr. Gatsis is a recipient of the NSF CAREER award. He has co-organized symposia in the area of smart grids in IEEE GlobalSIP 2015 and IEEE GlobalSIP 2016. He has also served as a co-guest editor for a special issue of the IEEE Journal on Selected Topics in Signal Processing on Critical Infrastructures.

Tyler H. Summers is an Assistant Professor of Mechanical Engineering at the University of Texas at Dallas. Prior to joining UT Dallas, he was an ETH Postdoctoral Fellow at the Automatic Control Laboratory at ETH Zurich from 2011 to 2015. He received a B.S. degree in Mechanical Engineering from Texas Christian University in 2004 and an M.S. and PhD degree in Aerospace Engineering with emphasis on feedback control theory at the University of Texas at Austin in 2007 and 2010, respectively. He was a Fulbright Postgraduate Scholar at the Australian National University in Canberra, Australia in 2007–2008. His research interests are in feedback control, optimization, and learning in complex dynamical networks, with applications to electric power networks and distributed robotics.

Ram Krishnan is an Associate Professor of Electrical and Computer Engineering at the University of Texas, San Antonio where he holds Microsoft President's Endowed Professorship. His current research focuses on security and privacy issues of machine learning and its application to various aspects of cybersecurity such as malware detection