Sparse Actuator Scheduling for Discrete-Time Linear Dynamical Systems

Kondapi, Krishna Praveen V.S.; Sriram, Chandrasekhar; Joseph, Geethu; Murthy, Chandra R.

DOI
10.1109/ICC64753.2024.10883726

Publication date
2024

Document Version
Final published version

Published in
2024 10th Indian Control Conference, ICC 2024 - Proceedings

2024 Tenth Indian Control Conference (ICC)
9 - 11 December, 2024
IISER Bhopal, India

Sparse Actuator Scheduling for Discrete-Time Linear Dynamical Systems

Krishna Praveen V. S. Kondapi[1], Chandrasekhar Sriram[2], Geethu Joseph[3] and Chandra R. Murthy[1]
Fellow, IEEE

***Abstract*— We consider the control of discrete-time linear dynamical systems using sparse inputs where we limit the number of active actuators at every time step. We develop an algorithm for determining a sparse actuator schedule that ensures the existence of a sparse control input sequence, following the schedule, that takes the system from any given initial state to any desired final state. Since such an actuator schedule is not unique, we look for a schedule that minimizes the energy of sparse inputs. For this, we optimize the trace of the inverse of the resulting controllability Gramian, which is an approximate measure of the average energy of the inputs. We present a greedy algorithm along with its theoretical guarantees. Finally, we empirically show that our greedy algorithm ensures the controllability of the linear system with a small number of active actuators per time step without a significant average energy expenditure compared to the fully actuated system.**

***Index Terms*— Greedy algorithm, controllability, time-varying schedule, sparse actuator selections, piecewise sparsity.**

I. Introduction

The field of networked control systems, in which the controllers, sensors, and actuators communicate over a band-limited network to achieve a given control objective, is a growing area of research [1]–[3]. Due to the bandwidth constraints, these systems demand communication-efficient control inputs. One way to reduce the communication cost is to use only a few available actuators at any given time instant, i.e., the number of nonzero entries of the control input is small compared to its length, leading to a *sparse control input* [3], [4]. The sparse vectors admit compact representation in a suitable basis [5], and thus, save bandwidth. Sparse control also finds applications in resource-constrained systems such as environmental control systems and network opinion dynamics [6], [7]. Motivated by this, we consider the problem of designing an actuator schedule for sparse control inputs to drive the system to any given desired state.

The controllability of a discrete-time linear dynamical system (LDS) using sparse inputs, along with its necessary and sufficient conditions, was defined and developed in [8]. Here, at each time step, the input can utilize at most s actuators, but the selected subset of s actuators can be time-varying. However, the work does not provide the actuator schedule or design the sparse inputs. Some researchers have considered the actuator scheduling for continuous-time systems by maximizing the trace of the controllability Gramian with a limit on the number of active actuators per time step [9]–[12]. This problem has been extended to cases where system dynamics are unknown, resulting in the development of an online, data-driven approach for learning the optimal actuator placement [11], [12]. Some works have considered a restrictive variant where the subset of chosen actuators remains fixed across time [13]–[15]. However, time-varying actuator scheduling can potentially ensure the controllability of systems that are uncontrollable systems when using inputs with fixed support, and could guide the LDS to the desired state faster than time-invariant actuator scheduling. A variant of time-varying sparse actuator scheduling problem considered in the literature only limits the *average* number of active actuators across all the time steps [16], [17]. They developed schedulers that yield a Gramian matrix that approximates the Gramian matrix of the fully actuated system. An alternative approach for the time-varying actuator scheduling problem is using a linear-quadratic regulator or linear-quadratic-Gaussian control [18], [19]. The authors in [20] also consider a time-varying sparse schedule with the goal of obtaining controllability with a symmetric transfer matrix $\boldsymbol{A}$. To the best of our knowledge, our problem of identifying a time-varying sparse actuator schedule that ensures controllability with minimal control energy has not been explored in the literature.

We formulate our actuator scheduling problem as minimizing the trace of the inverse of the controllability Gramian, a widely used metric for quantifying average control energy [16]. This optimization is subject to a constraint limiting the active actuators to at most s per time step. The specific contributions of this paper are as follows:

- We show that our objective function exhibits α-supermodularity, and the sparsity constraint is a matroid. This result leads to a greedy algorithm for finding the sparse actuator schedule with provable guarantees.
- Using our algorithm, we empirically study the tradeoff between the sparsity and average energy for time-invariant and time-varying schedules. We observe that, for large social networks, the increase in the average control energy compared to the average energy required by a fully actuated system is inversely proportional to the fraction of active actuators.

Overall, we present a systematic study of sparse time-varying actuator scheduling, develop a novel scheduling algorithm

[1]K. Kondapi and C. R. Murthy are with the Dept. of ECE at the Indian Institute of Science (IISc), Bangalore 560012, India. Emails: {praveenkvsk, cmurthy}@iisc.ac.in.
[2]C. Sriram is with Texas Instruments India Pvt. Ltd. He was at the Dept. of ECE, IISc, during the course of this work. Email: chandrasekhars@alum.iisc.ac.in
[3]G. Joseph is with the Faculty of Electrical Engineering, Mathematics, and Computer Science at the Delft University of Technology, Delft 2628 CD, Netherlands. Email: G.Joseph@tudelft.nl.

2024 Tenth Indian Control Conference (ICC) | 979-8-3315-1721-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICC64753.2024.10883726

with theoretical guarantees, and empirically elucidate its energy requirements.

II. Actuator Scheduling Problem

We consider a discrete-time LDS whose dynamics are governed by the following equation:

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

where $k = 0, 1, \ldots$ is the integer time index. Here, $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the system state and $\boldsymbol{u}(k) \in \mathbb{R}^m$ is the input to the system. The matrices $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ are the system transfer matrix and the input matrix, respectively. We consider the input $\boldsymbol{u}(k)$ to have at most s nonzero entries (s-sparse inputs) i.e., $\|\boldsymbol{u}(k)\|_0 \le s,\ \forall\, k = \{0, 1, \ldots, K-1\}$ where s is pre-specified to account for the communication bandwidth and data rate limitations which arise in networked control systems. If the support of $\boldsymbol{u}(k)$ is known, we can find the sparse inputs using the conventional methods. Our objective is to design a sparse actuator schedule (support of the sparse inputs) that can be used to drive the system to any given desired state in K time steps ($\boldsymbol{x}(K) = \boldsymbol{x}_f \in \mathbb{R}^n$).

We denote the sparse actuator schedule by $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{K-1})$, where, $\mathcal{S}_k \subseteq \{1, 2, \ldots, m\}$ and $|\mathcal{S}_k| \le s$ for all values of k to satisfy the sparsity constraint. It is known that for any s-sparse controllable LDS, we can find a finite K and sparse actuator schedule $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{K-1})$ with $|\mathcal{S}_k| \le s$ such that

$$\mathsf{Rank}\left\{\begin{bmatrix}\boldsymbol{A}^{K-1}\boldsymbol{B}_{\mathcal{S}_0} & \boldsymbol{A}^{K-2}\boldsymbol{B}_{\mathcal{S}_1} & \ldots & \boldsymbol{B}_{\mathcal{S}_{K-1}}\end{bmatrix}\right\} = n. \tag{2}$$

Sparse controllability is equivalent to two conditions being satisfied: the system is controllable and $s \ge n - \mathsf{Rank}\{\boldsymbol{A}\}$ [8]. So, in the sequel, to ensure that the problem is feasible, we assume that these two conditions hold.

Further, for an s-sparse controllable LDS, the minimum number of time steps K required to ensure controllability is bounded as [8]

$$\frac{n}{s} \le K \le \min\left(q\left\lceil\frac{\mathsf{Rank}\{\boldsymbol{B}\}}{s}\right\rceil, n-s+1\right) \le n. \tag{3}$$

where q is the degree of the minimum polynomial of $\boldsymbol{A}$. For simplicity, we let $K = n$. Our goal is to find an actuator schedule $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{n-1}) \in \Phi$ such that (2) holds, where the feasible set Φ for $\mathcal{S}$ is defined as

$$\Phi = \{(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{n-1})\colon \mathcal{S}_k \subseteq \{1, 2, \ldots, m\}, |\mathcal{S}_k| \le s, \forall k\}. \tag{4}$$

The rank of the controllability matrix can be analyzed using the Gramian matrix. For any actuator schedule $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{n-1}) \in \Phi$, the corresponding Gramian is

$$\boldsymbol{W}_{\mathcal{S}} = \sum_{k=1}^{n} \boldsymbol{A}^{k-1}\boldsymbol{B}_{\mathcal{S}_{n-k}}\boldsymbol{B}_{\mathcal{S}_{n-k}}^{\mathsf{T}}(\boldsymbol{A}^{k-1})^{\mathsf{T}}. \tag{5}$$

Our goal is to select the actuator schedule $\mathcal{S}$ so that $\mathsf{Rank}\{\boldsymbol{W}_{\mathcal{S}}\} = n$, which in turn ensures that the system is controllable. However, the solution does not need to be unique. Therefore, we look at minimizing $\mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}}^{-1}\right\}$ which is a popular measure of the "difficulty of controllability" [21]. It represents the average energy to drive the LDS from $\boldsymbol{x}(0) = \boldsymbol{0}$ to a uniformly random point on the unit sphere. We present a greedy algorithm to minimize $\mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}}^{-1}\right\}$ next. The algorithm and the results presented in the next section can be easily extended to other metrics such as $\det(\boldsymbol{W}_{\mathcal{S}})^{-\frac{1}{n}}$.

III. Greedy Scheduling Algorithm

First, we note that $\mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}}^{-1}\right\}$ is not well-defined if $\boldsymbol{W}_{\mathcal{S}}$ is rank-deficient. So, we use the lower bound $\mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}} + \epsilon\boldsymbol{I})^{-1}\right\} (\le\ \mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}})^{-1}\right\})$ as an alternative cost function, referred to as the ϵ-auxiliary energy, for some small $\epsilon > 0$. The term $\epsilon\boldsymbol{I}$ guarantees that the inverse exists and the cost function is well-defined for any schedule. Thus, our new optimization problem is

$$\underset{\mathcal{S}\in\Phi}{\arg\min}\ \mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}} + \epsilon\boldsymbol{I})^{-1}\right\}. \tag{6}$$

where Φ is defined in (4) and $\boldsymbol{W}_{\mathcal{S}}$ is given by (5). As ϵ decreases, (6) becomes a better approximation of the original objective of attaining full rank matrix $\boldsymbol{W}_{\mathcal{S}}$.

Next, we observe that any actuator schedule of length n can be represented using a subset of the set

$$\mathcal{V} = \{(k, j) | k = 0, 1, \ldots, n-1,\ j = 1, 2, \ldots, m\}. \tag{7}$$

Here, k represents the time index, and j represents the actuator index. Let $2^{\mathcal{V}}$ denote the power set of $\mathcal{V}$. It is clear that there is a natural bijection between any $\mathcal{T} \in 2^{\mathcal{V}}$ and the corresponding actuator schedule:

$$\mathcal{S}(\mathcal{T}) = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{n-1}) \ \text{ with } \ \mathcal{S}_k = \{j : (k, j) \in \mathcal{T}\}. \tag{8}$$

Therefore, the problem in (6) can be written using $\mathcal{V}$ as

$$\min_{\mathcal{T}\in 2^{\mathcal{V}}}\ \mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}(\mathcal{T})} + \epsilon\boldsymbol{I})^{-1}\right\} \text{ s.t. } \mathcal{S}(\mathcal{T}) \in \Phi. \tag{9}$$

Using the new formulation, the greedy algorithm starts with $\mathcal{T}$ being the empty set and finds the element from $\mathcal{V}$ which when added to $\mathcal{T}$ minimizes the cost function. Specifically, in the rth iteration of the algorithm, let $\mathcal{T}^{(r)}$ set of indices collected up to the previous iteration. Then, we find

$$(k^*, j^*) \ = \ \underset{(k,j)\in\mathcal{V}^{(r)}}{\arg\min}\ \mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}(\mathcal{T}^{(r)}\cup\{(k,j)\})} + \epsilon\boldsymbol{I})^{-1}\right\}, \tag{10}$$

where $\mathcal{V}^{(r)} \subseteq \mathcal{V}\setminus\mathcal{T}^{(r)}$ is obtained by removing all infeasible index pairs, i.e.,

$$\mathcal{V}^{(r)} = \left\{(i, j) \in \mathcal{V}\setminus\mathcal{T}^{(r)} : \mathcal{S}(\mathcal{T}^{(r)} \cup (i, j)) \in \Phi\right\}. \tag{11}$$

Finally, to complete the algorithm's description, we must choose ϵ. We start with some $\epsilon_0 > 0$ and repeat the greedy algorithm (outer loop) for decreasing values of ϵ until we obtain a full-rank Gramian matrix. The overall procedure is outlined in Algorithm 1.

The intuition behind the greedy algorithm is as follows. For a given value of ϵ, let $\mathcal{S}^{(r)}$ be the actuator schedule obtained in the rth iteration of the greedy algorithm. Suppose that $\mathsf{Rank}\{\boldsymbol{W}_{\mathcal{S}^{(r)}}\} = R$, which implies it has R nonzero

Algorithm 1 Greedy actuator scheduling

Input: System matrices: $\boldsymbol{A}, \boldsymbol{B}$; sparsity s
Parameters: $\epsilon_0 > 0$, $c > 1$
Initialize outer loop index $t = 0$
do
  Initialize $r = 1$, $\mathcal{T}^{(r)} = \emptyset$, $\mathcal{V}^{(r)} = \mathcal{V}$
  while $\mathcal{V}^{(r)} \neq \emptyset$ **do**
    (k^*, j^*) using (10) with $\epsilon = \epsilon_t$
    $\mathcal{T}^{(r+1)} = \mathcal{T}^{(r)} \cup \{(k^*, j^*)\}$
    $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{n-1}) = \mathcal{S}(\mathcal{T}^{(r+1)})$
    if $|\mathcal{S}_{k^*}| = s$ **then**
      $\mathcal{V}^{(r+1)} = \mathcal{V}^{(r)} \setminus \{(k^*, j),\ j = 1, 2, \ldots, m\}$
    else
      $\mathcal{V}^{(r+1)} = \mathcal{V}^{(r)} \setminus \{(k^*, j^*)\}$
    end if
    $r \leftarrow r + 1$
  end while
  $t \leftarrow t + 1,\ \epsilon_{t+1} = \epsilon_t / c$
while $\mathsf{Rank}\{\boldsymbol{W}_{\mathcal{S}}\} < n$
Output: Actuator schedule $\mathcal{S}$

eigenvalues $\{\lambda_i\}_{i=1}^R$. Then, for a small ϵ, the objective function is

$$\mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}^{(r)}} + \epsilon \boldsymbol{I})^{-1}\right\} = \sum_{i=1}^{R} \frac{1}{\lambda_i + \epsilon} + \frac{n - R}{\epsilon}. \tag{12}$$

The term $(n-R)/\epsilon$ acts as the penalty added to the objective function when $\boldsymbol{W}_{\mathcal{S}^{(r)}}$ is rank deficient. Our next result shows that the greedy algorithm improves the rank of the resulting Grammian until the rank becomes n under the availability of such actuators for sufficiently small ϵ.

Proposition 1: For a given outer loop iteration index t of Algorithm 1, let the rth (inner) iteration start with $\mathcal{T}^{(r)}$ and its search space be $\mathcal{V}^{(r)}$ be as defined in (11). Suppose that the following set,

$$\left\{v \in \mathcal{V}^{(r)} : \mathsf{Rank}\left\{\boldsymbol{W}_{\mathcal{S}(\mathcal{T}^{(r)} \cup \{v\})}\right\} > \mathsf{Rank}\left\{\boldsymbol{W}_{\mathcal{S}(\mathcal{T}^{(r)})}\right\}\right\}, \tag{13}$$

is nonempty. Then, there exists $\epsilon^* > 0$ such that if $\epsilon_t < \epsilon^*$, the next iteration of the greedy algorithm satisfies

$$\mathsf{Rank}\left\{\boldsymbol{W}_{\mathcal{S}(\mathcal{T}^{(r+1)})}\right\} = \mathsf{Rank}\left\{\boldsymbol{W}_{\mathcal{S}(\mathcal{T}^{(r)})}\right\} + 1. \tag{14}$$

Here, the schedule $\mathcal{S}(\cdot)$ is defined in (8) and the resulting Gramian is defined in (5).

Proof: See Appendix I. ∎

The above result shows that the greedy algorithm favors adding an element that increases the rank of the Gramian over an element that only increases the eigenvalues of $\boldsymbol{W}_{\mathcal{S}^{(r+1)}}$ without concurrently increasing its rank. The inner iteration of Algorithm 1 performs a greedy optimization of (9). When the stopping criterion of Algorithm 1 is not satisfied, ϵ is reduced by a factor of c. From Proposition 1, reducing ϵ in the outer iteration eventually forces the algorithm to choose an actuator that improves the rank in the next inner iteration. Thus, when ϵ becomes small enough, the approach greedily chooses actuators to minimize the average control energy while improving controllability.

Next, in order to characterize the near-optimality of Algorithm 1, we introduce the notions of supermodularity and matroid constraints:

Definition 1 (α-Supermodularity): The set function $f : \mathcal{U} \to \mathbb{R}$ is said to be α-submodular if $\alpha \in \mathbb{R}_+$ is the largest number for which the following holds $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$ and $\forall e \in \mathcal{U} \setminus \mathcal{B}$,

$$f(\mathcal{A} \cup \{e\}) - f(\mathcal{A}) \geq \alpha[f(\mathcal{B} \cup \{e\}) - f(\mathcal{B})]. \tag{15}$$

Also, the function $-f(\cdot)$ is said to be α-supermodular.

Definition 2 (Matroid): A matroid is a pair $(\mathcal{U}, \mathcal{I})$ where $\mathcal{U}$ is a finite set and $\mathcal{I} \subseteq 2^{\mathcal{U}}$ satisfies the following three properties: (i) $\emptyset \in \mathcal{I}$; (ii) For any two sets, $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{U}$, if $\mathcal{B} \in \mathcal{I}$, then $\mathcal{A} \in \mathcal{I}$; (iii) For any two sets, $\mathcal{A}, \mathcal{B} \in \mathcal{I}$, if $|\mathcal{B}| > |\mathcal{A}|$, then there exists $e \in \mathcal{B} \setminus \mathcal{A}$ such that $\mathcal{A} \cup \{e\} \in \mathcal{I}$.

Our next result establishes that the cost function of the optimization problem in (9) is supermodular and that its constraint set is a matroid.

Proposition 2: The objective function of (9), $E(\mathcal{T}, \epsilon) \triangleq \mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}(\mathcal{T})} + \epsilon \boldsymbol{I})^{-1}\right\}$ is an α-supermodular function with α satisfying

$$\alpha(\epsilon) \geq \frac{\epsilon}{\lambda_{\max}(\epsilon \boldsymbol{I} + \boldsymbol{W})}, \tag{16}$$

where $\boldsymbol{W} \triangleq \sum_{k=1}^{n} \boldsymbol{A}^{k-1} \boldsymbol{B} \boldsymbol{B}^{\mathsf{T}} (\boldsymbol{A}^{k-1})^{\mathsf{T}}$ and $\lambda_{\max}(\cdot)$ is the largest magnitude eigenvalue. Also, $(\mathcal{V}, \{\mathcal{T} :\ \mathcal{S}(\mathcal{T}) \in \Phi\})$ is a matroid.

Proof: See Appendix II. ∎

In (16), we explicitly retain the dependence of α on ϵ, as the value of ϵ varies across the iterations of Algorithm 1. From Proposition 2, we have the following guarantee for the inner loop of Algorithm 1.

Theorem 1: Let $(\boldsymbol{A} \in \mathbb{R}^{n \times m}, \boldsymbol{B} \in \mathbb{R}^{n \times m})$ be an s-sparse controllable LDS. Let $\beta(\epsilon) = \min\left(\frac{\alpha(\epsilon)}{2}, \frac{\alpha(\epsilon)}{1+\alpha(\epsilon)}\right)$ where $\alpha(\cdot)$ is the submodularity constant of the cost function of (9). The cost function corresponding to the set $\mathcal{S}$ returned by the inner loop of Algorithm 1 run with a given value of ϵ satisfies

$$\mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}} + \epsilon \boldsymbol{I})^{-1}\right\} < [1 - \beta(\epsilon)] \frac{n}{\epsilon} + \beta(\epsilon) E^*, \tag{17}$$

where E^* is the optimal value of the objective function of the optimization problem in (9) with $\epsilon = 0$.

Proof: See Appendix III. ∎

Theorem 1 shows that the cost function evaluated at the solution returned by Algorithm 1 for a given value of ϵ is an additive term away from the optimal cost. Admittedly, the n/ϵ dependence in the additive term on the right hand side makes the bound loose. In practice, Algorithm 1 returns a schedule that far outperforms, for example, a random schedule. However, besides the above result, it is hard to quantify the optimality gap even numerically, due to the combinatorial nature of the problem.

When Algorithm 1 terminates, the resulting actuator schedule $\mathcal{S}$ satisfies $\mathsf{Rank}\{\boldsymbol{W}_{\mathcal{S}}\} = n$. Note that, due to the

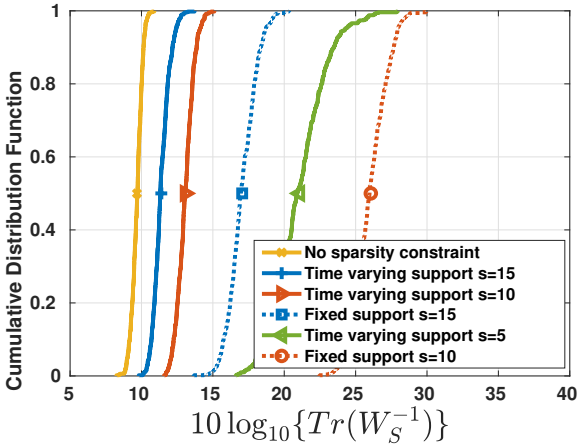

Fig. 1: CDF of $10\log_{10}\left(\mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}})^{-1}\right\}\right)$ for varying sparsity levels with $m = n = 20$.

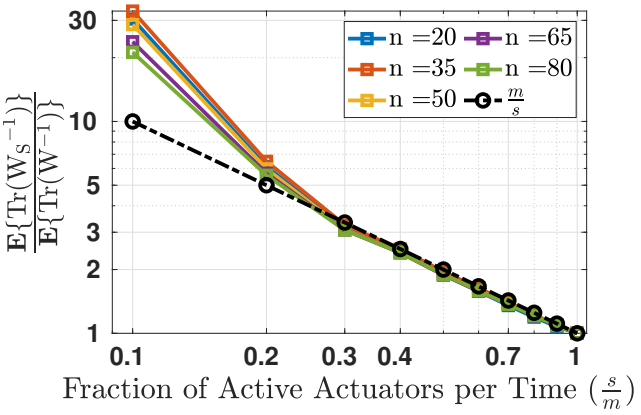

Fig. 2: $\frac{\mathbb{E}_{A,B}\mathsf{Tr}\{\boldsymbol{W}_{\mathcal{S}}^{-1}\}}{\mathbb{E}_{A,B}\mathsf{Tr}\{\boldsymbol{W}^{-1}\}}$ vs. the fraction of active actuators $(\frac{s}{m})$, averaged over 100 independent trials with $m = 50$.

greedy nature of the algorithm, $\mathcal{V}^{(r)}$ could become an empty set before $\boldsymbol{W}_{\mathcal{S}}$ attains full rank, and the algorithm may not terminate. However, we have never found this to be an issue in any of our experiments.

Finally, we note that our approach can also be used to find a time-invariant sparse actuator schedule. For this, we change Φ, the feasible set of (6), defined in (4), to $\{(\mathcal{S}_0, \mathcal{S}_0, \ldots, \mathcal{S}_0)\colon \mathcal{S}_0 \subseteq \{1, 2, \ldots, m\},\ |\mathcal{S}_0| \leq s\}$, i.e., $\mathcal{S}_k = \mathcal{S}_0$. We can drop the index t to replace $\mathcal{V}$ with $\{1, 2, \ldots, m\}$ and define the schedule obtained from $\mathcal{T}$ as $(\mathcal{T}, \mathcal{T}, \ldots, \mathcal{T})$ instead of (8) (Step 6 of Algorithm 1). Also, in this case, $\mathcal{V}^{(r)} = \{1, 2, \ldots, m\} \setminus \mathcal{T}^{(r)}$. Similar to Proposition 2, we can prove that this new problem optimizes an α-supermodular function subject to a matroid constraint, and like Theorem 1, the modified greedy strategy possesses similar guarantees. We omit the details to avoid repetition.

IV. Simulation Results

In this section, we illustrate the performance of the time-varying and sparse actuator scheduling algorithm developed in this work by applying it to an LDS whose transfer matrix $\boldsymbol{A}$ is generated from an Erdős-Renyi random graph. An Erdős-Renyi random graph consists of n vertices. The pairs of vertices are independently connected by an edge with probability $p = \frac{2\log n}{n}$, a setting used to model real-time networked systems [14]. The transfer matrix of the LDS is then generated from the graph as $\boldsymbol{A} = \boldsymbol{I} - \frac{1}{n}\boldsymbol{L}$, where $\boldsymbol{L}$ is the graph Laplacian.

We first illustrate the dependence of the average control energy, $\mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}})^{-1}\right\}$, on the sparsity level s of the control inputs. To this end, we generate the transfer matrices $\boldsymbol{A}$ from 500 Erdős-Renyi random graphs drawn independently. We use Algorithm 1 to determine the sparse actuator schedule, and compute $\mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}})^{-1}\right\}$ corresponding to the schedule $\mathcal{S}$ returned by the algorithm. For this experiment, we set $n = 20$, $m = n$, and $\boldsymbol{B} = \boldsymbol{I}$. In Fig. 1, we show the empirical cumulative distribution function (CDF) of $\mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}})^{-1}\right\}$ in two settings: the time-varying support case with sparsity levels $s = 5, 10, 15$, and the fixed support case with $s = 10, 15$. We also plot the CDF for the case with no sparsity constraint. The curves shift to the right as s is decreased as expected since the system becomes more constrained, thereby incurs a higher average control energy. Similarly, using a fixed support makes the system more constrained and hence requires higher average control energy compared to the time-varying support case. Further, the control energy with time-varying support and $s = 15, s = 10$ is close to the control energy without sparsity constraints. Therefore, time-varying sparsity constraints do not impose a significant energy burden to control the LDS at moderate sparsity levels; we elaborate on this in our next experiment.

Next, we illustrate the relative cost of imposing the sparsity constraints compared to the non-sparse case. Here, the entries of the input matrix $\boldsymbol{B}$ are i.i.d. and uniformly distributed over $[0, 1]$. We plot $\rho \triangleq \mathbb{E}_{A,B}\mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}}^{-1}\right\}/\mathbb{E}_{A,B}\mathsf{Tr}\left\{\boldsymbol{W}^{-1}\right\}$ as a function of the fraction of active actuators at each time instant (s/m). The quantity ρ represents the ratio of the average control energy with the sparsity constraint to the average control energy in the unconstrained case. We show the behavior in Fig. 2, with m fixed at 50 and for various values of n from 20 to 80. We observe that there is nearly an inverse-linear relationship between s/m and ρ. This shows that using time-varying s-sparse control inputs increases the average energy to control the system by a factor proportional to the reciprocal of the fraction of active actuators. We have also observed this trend when $\boldsymbol{B}$ is a random matrix with i.i.d. entries drawn from a Gaussian distribution and when $\boldsymbol{B} = \boldsymbol{I}$ (for $n = m$). With reasonable approximations, it is possible to mathematically derive that this behavior holds when $\boldsymbol{B}$ has i.i.d. entries; we omit the details due to lack of space.

Next, we compare Algorithm 1 with schedulers that are adapted from [16], the closest to our work. The algorithms in [16] are developed under an *average* sparsity constraint, i.e., the individual inputs need not be sparse, but the overall sequence of inputs need to satisfy an average

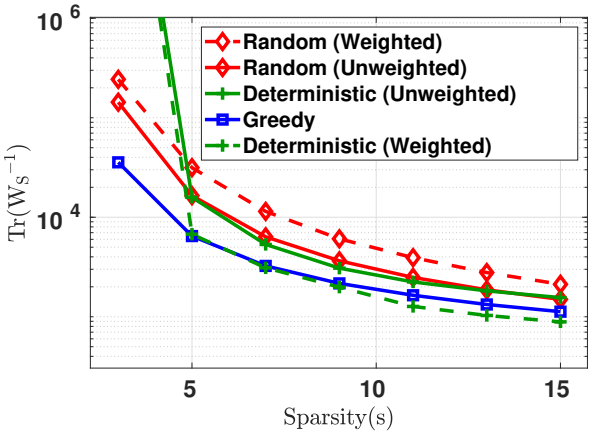

Fig. 3: $\mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}}^{-1}\right\}$ as a function of sparsity (s), averaged over 100 independent trials with $n = 100, m = 50$.

sparsity constraint. We modify the algorithms in [16] by adding a piecewise sparsity constraint. In Fig. 3, we plot $\mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}}^{-1}\right\}$ as a function of s for four algorithms adapted from [16], namely, random-weighted, random-unweighted, deterministic-unweighted, and deterministic-weighted. The random scheduler samples an actuator from the probability distribution given in [16, Algorithm 6] and adds it to the schedule provided adding it still satisfies the sparsity constraint. The deterministic scheduler picks a actuator that greedily optimizes the objective function used in [16, Algorithm 1]. The weighted schedulers have an additional weight (amplification) associated with the selected actuator, while the weights are set to 1 for the unweighted schedule (see [16, Algorithms 2, 6, 7].) For the unweighted random scheduler, the actuators are sampled without replacement. The plot shows that Algorithm 1 generally outperforms existing algorithms modified to satisfy our sparsity constraint, especially at lower sparsity levels.

V. Conclusion

We studied the problem of finding an energy-efficient actuator schedule for controlling an LDS in a finite number of steps, with at most s active inputs per time step. We presented a greedy scheduling algorithm by considering the minimization of $\mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}}^{-1}\right\}$, which models the average control energy. We presented several interesting theoretical guarantees for the algorithm. Through simulations, we showed that the algorithm returns a feasible schedule, with the average energy increasing (compared to the case without sparsity constraints) by a factor proportional to the inverse of the fraction of active inputs, for randomly drawn LDSs. Overall, we conclude that sparse inputs can drive an LDS to a desired state without a significant increase in energy requirements, at moderate sparsity levels.

Appendix I
Proof of Proposition 1

Let $\mathcal{S}$ be the actuator schedule at some iteration of the greedy algorithm such that $\mathsf{Rank}\{\boldsymbol{W}_{\mathcal{S}}\} = R < n$. By (13), there exists a candidate feasible schedule $\hat{\mathcal{S}}$ for the next iteration such that $\mathsf{Rank}\left\{\boldsymbol{W}_{\hat{\mathcal{S}}}\right\} = R+1$. Suppose there is an alternative feasible schedule $\tilde{\mathcal{S}}$ such that $\mathsf{Rank}\left\{\boldsymbol{W}_{\tilde{\mathcal{S}}}\right\} = R$. The greedy algorithm chooses the schedule $\hat{\mathcal{S}}$ if

$$\mathsf{Tr}\left\{(\boldsymbol{W}_{\tilde{\mathcal{S}}} + \epsilon \boldsymbol{I})^{-1}\right\} > \mathsf{Tr}\left\{(\boldsymbol{W}_{\hat{\mathcal{S}}} + \epsilon \boldsymbol{I})^{-1}\right\}.$$

Let $\lambda_i, i = 1, 2, \ldots n$, with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ denote the ordered eigenvalues of $\boldsymbol{W}_{\mathcal{S}}$ (similarly, $\hat{\lambda}_i$ and $\tilde{\lambda}_i$ for $\boldsymbol{W}_{\hat{\mathcal{S}}}$ and $\boldsymbol{W}_{\tilde{\mathcal{S}}}$, respectively). Then, we can write

$$\mathsf{Tr}\left\{(\boldsymbol{W}_{\tilde{\mathcal{S}}} + \epsilon \boldsymbol{I})^{-1}\right\} \geq \frac{R}{\tilde{\lambda}_1 + \epsilon} + \frac{n-R}{\epsilon}, \tag{18}$$

$$\mathsf{Tr}\left\{(\boldsymbol{W}_{\hat{\mathcal{S}}} + \epsilon \boldsymbol{I})^{-1}\right\} \leq \frac{R+1}{\hat{\lambda}_{R+1} + \epsilon} + \frac{n-R-1}{\epsilon}. \tag{19}$$

The following inequality is a sufficient condition for the greedy algorithm to choose $\hat{\mathcal{S}}$:

$$\frac{R}{\tilde{\lambda}_1 + \epsilon} + \frac{n-R}{\epsilon} > \frac{R+1}{\hat{\lambda}_{R+1} + \epsilon} + \frac{n-R-1}{\epsilon}, \tag{20}$$

Simplifying the above, we get

$$\tilde{\lambda}_1 \hat{\lambda}_{R+1} > \epsilon \left[R\tilde{\lambda}_1 - (R+1)\hat{\lambda}_{R+1}\right], \tag{21}$$

The inequality holds trivially for $R\tilde{\lambda}_1 - (R+1)\hat{\lambda}_{R+1} \leq 0$. For $R\tilde{\lambda}_1 - (R+1)\hat{\lambda}_{R+1} > 0$, we obtain

$$\epsilon < \frac{\tilde{\lambda}_1 \hat{\lambda}_{R+1}}{R\tilde{\lambda}_1 - (R+1)\hat{\lambda}_{R+1}}. \tag{22}$$

The right hand side of above equation is finite and positive, and ϵ is decreased by factor $c > 1$ in each iteration of the Algorithm 1. Hence, there will always exists an ϵ for which (22) is satisfied. Hence, for a sufficiently small ϵ, (22) holds for any $\tilde{\mathcal{S}}$ given an $\hat{\mathcal{S}}$. Thus, the greedy algorithm chooses an actuator that improves the rank. ■

Appendix II
Proof of Proposition 2

From [22, Theorem 2], we know that the real-valued function f of the form

$$f(\mathcal{A}) = \mathsf{Tr}\left\{\left(\boldsymbol{M}_{\emptyset} + \sum_{i \in \mathcal{A}} \boldsymbol{M}_i\right)^{-1}\right\}, \tag{23}$$

where $\mathcal{A}$ is an index set and $\boldsymbol{M}_{\emptyset} \succ 0$ and $\boldsymbol{M}_i \succeq 0$, is monotonically decreasing and α-supermodular with $\alpha \geq \frac{\mu_{\min}}{\mu_{\max}} > 0$, where

$$0 < \mu_{\min} \leq \lambda_{\min}(\boldsymbol{M}_{\emptyset}) \leq \lambda_{\max}(\boldsymbol{M}_{\emptyset} + \sum_{i \in \mathcal{A}} \boldsymbol{M}_i) \leq \mu_{\max}. \tag{24}$$

Here, $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ are the largest and smallest eigenvalues in magnitude, respectively. Our cost function $E(\mathcal{T}, \epsilon)$ takes the same form as $f(\mathcal{A})$, with $\boldsymbol{M}_{\emptyset} = \epsilon \boldsymbol{I} \succ 0$ and the matrices $\boldsymbol{M}_i$ being constructed as the submatrices of the controllability gramian $\boldsymbol{W}$ (see (5)). Further, from the definitions, we can set $\mu_{\min} = \epsilon$ and $\mu_{\max} = \lambda_{\max}(\epsilon \boldsymbol{I} + \boldsymbol{W})$. Hence, the function $E(\mathcal{T}, \epsilon)$ is α-supermodular with α satisfying (16).

We next complete the proof by showing that the sparsity constraint on the actuator set is a matroid constraint. For this proof, we consider the equivalent sparsity constraint on $\mathcal{T} \in 2^{\mathcal{V}}$ in (9). The constraint is given by

$$|\{j : (k,j) \in \mathcal{T}\}| \leq s, \text{ for } k = 0, 1, \ldots, n-1 \tag{25}$$

To prove that the above constraint is a matroid, we verify the three conditions of Definition 2. The first condition holds trivially. If we have a set $\mathcal{T}$ that satisfies the sparsity constraint in (25), then any subset $\mathcal{T}' \subseteq \mathcal{T}$ also satisfies the sparsity constraint. Hence, the second condition holds. For the third condition, suppose there exist sets $\mathcal{T}, \mathcal{T}' \in 2^{\mathcal{V}}$ satisfying the sparsity constraint in (25) and $|\mathcal{T}| > |\mathcal{T}'|$. Then, there exists at least one time index k such that

$$|\{j : (k,j) \in \mathcal{T}\}| > |\{j : (k,j) \in \mathcal{T}'\}|. \tag{26}$$

If we take an element $\tilde{j} \in \{j : (k,j) \in \mathcal{T} \setminus \mathcal{T}'\}$, the new set $\mathcal{T}' \cup \{(k,\tilde{j})\}$ satisfies

$$\begin{aligned}\left|\{j : (k,j) \in \mathcal{T}'\} \cup \{(k,\tilde{j})\}\right| &= |\{j : (k,j) \in \mathcal{T}'\}| + 1 \\ &\leq |\{j : (k,j) \in \mathcal{T}\}| \leq s.\end{aligned}$$

So, the new set $\mathcal{T}' \cup \{(k,\tilde{j})\}$ also satisfies (25), verifying the third condition and completing the proof. ■

APPENDIX III
PROOF OF THEOREM 1

Consider the problem of minimizing an α-supermodular function $f : 2^{\mathcal{V}} \to \mathbb{R}$ subject a matroid constraint $(\mathcal{V}, \mathcal{I})$ i.e.,

$$\min_{\mathcal{A} \in 2^{\mathcal{V}}} f(\mathcal{A}) \text{ s.t. } \mathcal{A} \in \mathcal{I}. \tag{27}$$

Suppose f^* is the optimal solution of (27) and $\tilde{f}$ is the solution returned by a greedy algorithm that starts with $\mathcal{A} = \emptyset$ and adds elements from $\mathcal{V}$ one-by-one, so that at step t, it updates $\mathcal{A}_t$ as $\mathcal{A}_{t+1} = \mathcal{A}_t \cup e^*$, where $e^* = \text{argmin}_{e \in \mathcal{V} \setminus \mathcal{A}_t, \mathcal{A}_t \cup e \in \mathcal{I}} f(\mathcal{A}_t \cup e)$. Then, from [18, Theorem 1] we have,

$$\frac{f^* - \tilde{f}}{f^* - f(\emptyset)} \leq 1 - \min\left(\frac{\alpha}{2}, \frac{\alpha}{1+\alpha}\right), \tag{28}$$

where α is the submodularity constant of f.

From Proposition 2, our cost function is α-supermodular and the constraint set is a matroid. Further, Algorithm 1 is a greedy algorithm of the form described above. For a fixed ϵ, let $E^*(\epsilon)$ be the optimal energy obtained by solving (9) exactly and $\tilde{E}(\epsilon) = \mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}} + \epsilon \boldsymbol{I})^{-1}\right\}$ be the objective function returned by Algorithm 1 with parameter ϵ. Then, using (28), we have

$$\frac{\tilde{E}(\epsilon) - E^*(\epsilon)}{n/\epsilon - E^*(\epsilon)} \leq 1 - \beta(\epsilon), \tag{29}$$

where $\beta(\epsilon) > 0$ is defined in Theorem 1. Let, $E^* = \mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}^*}^{-1}\right\}$ where $\mathcal{S}^*$ be the corresponding optimal actuator schedule obtained by solving (9) with $\epsilon = 0$. We obtain the following inequality,

$$E^* = \mathsf{Tr}\left\{\boldsymbol{W}_{\mathcal{S}^*}^{-1}\right\} > \mathsf{Tr}\left\{(\boldsymbol{W}_{\mathcal{S}^*} + \epsilon \boldsymbol{I})^{-1}\right\} \geq E^*(\epsilon). \tag{30}$$

Plugging in this inequality into (29) leads to

$$\tilde{E}(\epsilon) \leq \frac{n(1-\beta(\epsilon))}{\epsilon} + \beta(\epsilon)E^*(\epsilon) < \frac{n(1-\beta(\epsilon))}{\epsilon} + \beta(\epsilon)E^*. \tag{31}$$

Hence, the proof is complete. ■

REFERENCES

[1] S. C. Tatikonda, "Control under communication constraints," Ph.D. dissertation, Dept. Elect. Comput. Sci., MIT, Cambridge, USA, 2000.

[2] A. Jadbabaie, A. Olshevsky, G. J. Pappas, and V. Tzoumas, "Minimal reachability is hard to approximate," *IEEE Trans. Autom. Control*, vol. 64, no. 2, pp. 783–789, 2019.

[3] M. Nagahara and D. E. Quevedo, "Sparse representations for packetized predictive networked control," *IFAC Proc. Volumes*, vol. 44, no. 1, pp. 84 – 89, 2011.

[4] Z. Li, Y. Xu, H. Huang, and S. Misra, "Sparse control and compressed sensing in networked switched systems," *IET Control Theory Appl.*, vol. 10, no. 9, pp. 1078–1087, 2016.

[5] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*. Birkhäuser, 2013.

[6] G. Joseph, B. Nettasinghe, V. Krishnamurthy, and P. K. Varshney, "Controllability of network opinion in Erdos-Renyi graphs using sparse control inputs," *SIAM J. Control Optim.*, vol. 59, no. 3, pp. 2321–2345, 2021.

[7] N. Wendt, C. Dhal, and S. Roy, "Control of network opinion dynamics by a selfish agent with limited visibility," *IFAC-PapersOnLine*, vol. 52, no. 3, pp. 37 – 42, 2019, iFAC Symp. Large Scale Complex Syst.

[8] G. Joseph and C. R. Murthy, "Controllability of linear dynamical systems under input sparsity constraints," *IEEE Trans. Autom. Control*, vol. 66, no. 2, pp. 924–931, 2021.

[9] A. Olshevsky, "On a relaxation of time-varying actuator placement," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 656–661, 2020.

[10] T. Ikeda and K. Kashima, "Sparsity-constrained controllability maximization with application to time-varying control node selection," *IEEE Control Syst. Lett.*, vol. 2, no. 3, pp. 321–326, 2018.

[11] F. Fotiadis and K. G. Vamvoudakis, "Learning-based actuator placement for uncertain systems," in *Proc. CDC*, 2021, pp. 90–95.

[12] F. Fotiadis, K. G. Vamvoudakis, and Z.-P. Jiang, "Data-based actuator selection for optimal control allocation," in *Proc. CDC*, 2022, pp. 4674–4679.

[13] A. Olshevsky, "Minimal controllability problems," *IEEE Trans. Control Netw. Syst.*, vol. 1, no. 3, pp. 249–258, 2014.

[14] V. Tzoumas, M. A. Rahimian, G. J. Pappas, and A. Jadbabaie, "Minimal actuator placement with bounds on control effort," *IEEE Trans. Control Netw. Syst.*, vol. 3, no. 1, pp. 67–78, 2016.

[15] A. S. A. Dilip, "The controllability gramian, the hadamard product, and the optimal actuator/leader and sensor selection problem," *IEEE Control Syst. Lett.*, vol. 3, no. 4, pp. 883–888, 2019.

[16] M. Siami, A. Olshevsky, and A. Jadbabaie, "Deterministic and randomized actuator scheduling with guaranteed performance bounds," *IEEE Trans. Autom. Control*, vol. 66, no. 4, pp. 1686–1701, 2020.

[17] M. Siami and A. Jadbabaie, "A separation theorem for joint sensor and actuator scheduling with guaranteed performance bounds," *Automatica*, vol. 119, p. 109054, 2020.

[18] L. F. O. Chamon, A. Amice, and A. Ribeiro, "Matroid-constrained approximately supermodular optimization for near-optimal actuator scheduling," in *Proc. CDC*, 2019, pp. 3391–3398.

[19] J. Jiao, D. Maity, J. S. Baras, and S. Hirche, "Actuator scheduling for linear systems: A convex relaxation approach," *IEEE Control Syst. Lett.*, vol. 7, pp. 7–12, 2023.

[20] Y. Zhao, F. Pasqualetti, and J. Cortés, "Scheduling of control nodes for improved network controllability," in *Proc. CDC*, 2016, pp. 1859–1864.

[21] A. Olshevsky, "On (non)supermodularity of average control energy," *IEEE Tran. Control Netw. Syst.*, vol. 5, no. 3, pp. 1177–1181, 2018.

[22] L. F. O. Chamon, G. J. Pappas, and A. Ribeiro, "The mean square error in Kalman filtering sensor selection is approximately supermodular," in *Proc. CDC*, 2017, pp. 343–350.