3-8-2019

# Identifying and Using Driver Nodes in Temporal Networks

Babak Ravandi
*Purdue University*, bravandi@purdue.edu

Fatma Mili
*University of North Carolina at Charlotte*

John Springer
*Purdue University*

**Recommended citation:**
Ravandi, B., Mili, F., & Springer, J. A. (2019).
Identifying and using driver nodes in temporal networks.
Journal of Complex Networks.
https://academic.oup.com/comnet/advance-article-abstract/doi/10.1093/comnet/cnz004/5372353

*IMA Journal of Complex Networks* (2019) Page 1 of 29
doi:10.1093/comnet/xxx000

# Identifying and Using Driver Nodes in Temporal Networks

BABAK RAVANDI*
*Department of Computer and Information Technology, Purdue University, IN, USA*
*Corresponding author: bravandi@purdue.edu

FATMA MILI
*College of Computing and Informatics, University of North Carolina at Charlotte, NC, USA*
fatma.mili@uncc.edu

AND

JOHN A. SPRINGER
*Department of Computer and Information Technology, Purdue University, IN, USA*
jaspring@purdue.edu

[Received on 13 November 2018]

In many approaches developed for defining complex networks, the main assumption is that the network is in a relatively stable state that can be approximated with a fixed topology. However, in several applications, this approximation is not adequate because a) the system modeled is dynamic by nature, and b) the changes are an essential characteristic that cannot be approximated. Temporal networks capture changes in the topology of networks by including the temporal information associated with their structural connections, i.e., links or edges. We focus here on controllability of temporal networks, that is, the study of steering the state of a network to any desired state at deadline $t_f$ within $\Delta t = t_f - t_0$ steps through stimulating key nodes called driver nodes. Recent studies provided analytical approaches to find a maximum controllable subspace for an arbitrary set of driver nodes. However, finding the minimum number of driver nodes $N_c$ required to reach full control is computationally prohibitive.
In this work, we propose a heuristic algorithm that quickly finds a suboptimal set of driver nodes with size $N_s \geqslant N_c$. We conduct experiments on synthetic and real-world temporal networks induced from ant colonies and e-mail communications of a manufacturing company. The empirical results in both cases show the heuristic algorithm efficiently identifies a small set of driver nodes that can fully control the networks. Also, as shown in the case of ants' interactions networks, the driver nodes tend to have a large degree in temporal networks. Furthermore, we analyze the behavior of driver nodes within the context of their datasets, through which, we observe that queen ants tend to avoid becoming a driver node.

*Keywords*: temporal networks; complex networks; driver nodes; controllability; heuristic.

## 1. Introduction

A complex network is a network of often numerous interconnected components with non-trivial topological/structural characteristics that do not occur in simple networks, such as lattices or random graphs, but often occur in networks modeling real-world systems. The application of such networks spans over many scientific disciplines including abstract and applied physics [1], biology [2, 3], chemistry [4], and sustainability in ecosystem management [5]. Understanding the structure of these complex networks is necessary to understand and predict their behavioral characteristics.

Controlling such systems motivated the scientific community for decades and resulted in a tremendous

number of studies on controlling networks mostly focused on fixed topology [6, 7]. Given the dynamic nature of complex systems, it is necessary to study how the changes in structure of the networks influence our ability to control them. In this work, we focus on the controllability of networks with changing structure over time, also called temporal networks or dynamic networks [8]. When non-trivial temporal correlations govern a system, it is necessary to include the order of interactions in the structure. Such complex systems include transportation, communication, biological, and neural networks [9–13]. Furthermore, temporal correlations in pathways govern the spreading process [10]. Static networks capture the existence of connections between nodes during an observation period, ignoring the temporal correlations. To capture the dynamic aspects, the temporality of links must be explicit. Fig. 1 presents a simple temporal network in the observation period $1 \leqslant t \leqslant 2$ with three time-respecting paths, marked with colors red, blue, and green. If a path follows the temporal ordering of connections, it is denoted as a time-respecting path. The static representation of a temporal network is is known as time-aggregated network where the weights of edges represent the observation times, in which interactions between nodes were observed. Consider the time-aggregated network presented in Fig. 1 (a) with timestamped edges. Without considering the temporal correlations, information can spread from *node 4* to *node 1*. However, this is not possible if considering the temporal order of interactions. To clearly show the temporal correlations, we use a time-layered network, i.e., the network representation created by making a copy of nodes for each observation period as a time-layer and connecting the nodes between the layers. In other words, the time-layers represent the intervals in which the interactions between nodes are observed. Fig. 1 (b) presents the time-layered visualization of the network; it unpacks the temporal correlations with layers of time and presents the interactions between them.

The study of controllability in complex networks addresses the existence of ways to influence a network within a finite time to reach a targeted state. Studies on structural controllability of graphs showed it is possible to answer control-related questions by only leveraging the underlying structure of graphs [14, 15]. Compared to static networks, temporal networks have fundamental advantages in terms of time and energy required to reach controllability. This means the changing topology of complex system can be utilized to increase the efficiency of control [16]. Pósfai and Hövel extended the structural controllability theorem to temporal networks [17] and provided the necessary tools to study controllability for this class of networks. Moreover, they investigated the Maximum Controllable Subspace (MCS) for a single driver node. We extend these results by focusing on finding a minimum number of nodes required to reach complete control in temporal networks.

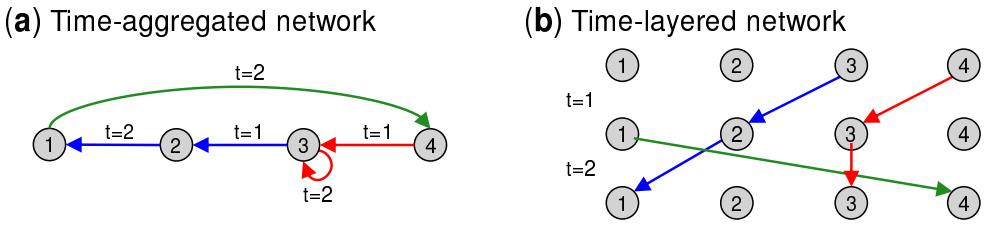


FIG. 1. Temporal correlations in pathways. Three time-respecting paths that follow the temporal correlations are marked with colors red, blue, and green. (a) A time-aggregated temporal network within the observation period $1 \leqslant t \leqslant 2$. Considering the temporal correlations, the path $4 \to 3 \to 2 \to 1$ physically does not exists. (b) Visualization of the temporal correlations using the time-layered network presentation. The self-loop at interval $t = 2$ allows *node 3* to retain the influence it received from *node 4* at interval $t = 1$.

### 1.1 *Innovation*

Networks are controlled by stimulating a subset of nodes known as driver nodes. Controllable networks can have multiple sets of driver nodes. The problem of control is twofold: determining whether an actual set of driver nodes exists and if so, identifying a minimum set of driver nodes with size $N_c$ [15, 17, 18]. Because the control of temporal networks is still emerging, little attention has been devoted to the latter of these, i.e., to identify the minimum number of driver nodes $N_c$ required to reach complete control in temporal networks. Finding $N_c$ is a computationally prohibitive task for both static and temporal networks. For static networks, the most efficient approach reduces the problem to the maximum matching problem that finds a Minimum Driver node Set (MDS) with size $N_c$ [15]. However, the analytical approach for temporal networks requires validating an order of $O(2^{N\Delta t})$ configurations where $N$ is the number of nodes and $\Delta t$ is the desired number of steps to reach control. In addition, the validation itself requires calculating the rank of the corresponding temporal controllability matrix where building this matrix involves an exponential number of matrix multiplication. This validation is $O(A(t)^{\Delta t})$ where $A(t)$ is the adjacency matrix of the temporal network at interval $t$ [17]. This motivates the desire for finding more efficient approaches. We introduce a heuristic algorithm with complexity $O(N^3+N^2E+N\Delta t)$ that finds multiple Suboptimal Minimum Driver node Sets (SMDS) with size $N_s \geqslant N_c$ to efficiently control a temporal network where $E$ is the set of timestamped edges.

## 2. Complex Networks Controllability

In many approaches developed for defining complex networks, the main assumption is that the network is in a static or quasi-static state with a static topology [19]. Static construction is typically used to build networks from a snapshot or aggregated snapshots of a complex networked system. However, many real-world networks exhibit a significant level of dynamic behavior in both spatial and temporal domains [8]. For instance, network-analytic methods make the fundamental assumption that paths are transitive (i.e., the existence of path $a \to b$ and $b \to c$ implies a transitive path $a \to b \to c$). However, as shown in [20] and Fig. 1, temporal correlations in pathways can invalidate this assumption and accordingly invalidate the approaches based on them.

Another important aspect of complex systems traditionally not captured in static networks is the effect of self-interactions [21, 22]. Most natural systems enjoy passive stability, i.e., the ability to retain information over time [23]. The framework provided by Pósfai and Hövel allows using self-loops to model the state retention aspect [17]. A simple example of self-loop is presented in Fig. 1 enabling the influence of *node 4* on *node 3* at interval $t = 1$ to persist to interval $t = 2$.

### 2.1 *Controllability of Static Networks*

Static networks assume the topology of network is fixed over time. Controllability of static networks is interested in modeling systems that can be presented by a set of nodes with state variables. The relationships between the nodes are captured with directed weighted edges and a protocol that governs the dynamics by which the network changes the state variables over time. These systems interact with an environment from which they receive stimuli (inputs), denoted by $u(t)$, and to which they produce effects (outputs). In other words, the controllability of complex networks is the study of the relationship between the external stimuli and the graphs' behaviors and outputs.

Although most physical systems have intrinsic nonlinear dynamic behavior, their controllability in many aspects can be structurally considered to be linear [24]. Recent literature mostly focused on the linear time-invariant dynamics (LTI) [25–28]. Equation 2.1 presents the dynamics of network in time.

$$x(t+1) = A^\top x(t) + Bu(t) \tag{2.1}$$

The adjacency matrix $A \in \mathbb{R}^{N\times N}$ describes the "wiring" of the system and its interaction strengths. The time varying vector $x(t) \in \mathbb{R}^{N\times 1}$ describes the state of the system at time $t$. The second term $Bu(t)$ models the effect of the external stimuli, in which $B \in \mathbb{R}^{N\times M}$ captures the connections with a controller and $M$ driver nodes. Also, the strength of the stimuli imposed by the controller is captured by the time-varying input vector $u(t) \in \mathbb{R}^{M\times 1}$. To reach controllability, first we need to identify a set of nodes that can provide full control over the network if driven by different external stimuli; as indicated earlier, such nodes are called driver nodes, and most control problems are interested in identifying the minimum number of driver nodes, denoted by $N_c$, to fully control a system.

The system described above is controllable if we can steer its state from any initial state to a desired state within a finite time. This is only possible iff the controllability matrix $C \in \mathbb{R}^{N\times NM}$ (defined in Equation 2.2) has full rank (i.e., $rank(C) = N$). This represents the mathematical controllability condition called Kalman's rank condition [29].

$$C = (B, AB, A^2B, \ldots, A^{N-1}B) \tag{2.2}$$

Applying the Kalman's rank condition to Equation 2.2 requires knowing the weights of all links in the entire network that for most physical systems it is not practical. The structural controllability theorem enables one to bypass the problem of *a priori* having the exact weight of the links. This means, the system $\langle A, B\rangle$ is "structurally controllable" if it is possible to place nonzero weights in $A$ and $B$ and assuming Equation 2.1 governs the system's dynamics. In [14], Lin shows that a structurally controllable system is controllable for almost any weight configuration. However, $O(2^N)$ combinations of driver nodes are required to test the Kalman's rank condition to find an MDS. For static networks, extensive research has been done to efficiently test structural controllability and determine the location of the minimum number of drivers required. One approach to identify an MDS is by finding a maximum matching of the network. Murota [15] represents a comprehensive view of the conducted research on controllability of dynamical systems and the maximum matching approach; this is on the order of $O(\sqrt{N}E)$ using Hopcroft-Karp bipartite matching algorithm [30] where $N$ is the number of nodes and $E$ is the number of edges.

### 2.2 *Controllability of Temporal Networks*

Temporal networks allow modeling a system with a topology that changes over time. We can utilize these changes in network's wiring to increase efficiency of control in terms of the time and energy required to gain control compared to the static networks [16]. Also, including time in the definition of controllability allows to explicitly study the time necessary to gain control; in contrast, this is an area that only receives implicit treatment in static networks [31]. These motivate the need to study the controllability of temporal networks. In this section, we first introduce the definition of temporal networks and then cover the definition of structural controllability for temporal networks.

A directed temporal network $T(V,E)$, consists of a set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and a set of timestamped directed edges $E = \{e_1, e_2, \ldots, e_m\}$. Each temporal link $e = (v_i, v_j, w_{ij}, t) \in E$ connects two vertices $v_i \to v_j$ at an observation interval $t$, and the weight of their connection is captured by $w_{ij}$. The weights in static networks often present aggregated information about interactions between two vertices (i.e., the frequency of interactions). In temporal networks, the weights present the strength of a connection at a point in time. For example, in social networks weights could present the strength of trust between people during a time period.

Many physical systems are driven by nonlinear processes. However, the controllability of nonlinear systems in many aspects is structurally similar to linear systems [24]. In this work, we consider discrete time-varying linear dynamics govern the interactions between nodes as shown in Equation 2.3 [17, 32].

$$x(t+1) = A(t)^{\top} x(t) + B(t)u(t) \tag{2.3}$$

The state of network is captured by the time-varying vector $x(t) \in \mathbb{R}^{N\times 1}$ where $N$ is the number of nodes. Equation 2.3 defines the state $x$ at time $t+1$ as a sum of two terms: the resultant of the internal system dynamics applied to the state at interval $t$ and the impact of external stimuli. The matrix $A(t) \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix of the network at observation interval $t$. Control is applied by stimulating driver nodes $d_i \in V$ at interval $t$ that propagates the stimulated value to the network at interval $t+1$. Through this article, stimulating a driver node is referred as an *intervention*, and the pair $(d_i, t)$ is called an *intervention point*. The matrix $B(t) \in \mathbb{R}^{N\times N_I(t)}$ identifies the intervention points where a) $N_I(t)$ is the number of intervention points at interval $t$, and b) $u(t) \in \mathbb{R}^{N_I(t)}$ denotes the strength of stimuli to enforce control.

### 2.2.1 *Independent Path Theorem*

Pósfai and Hövel extended the standard definition of structural controllability to time-varying systems modeled with directed temporal networks [17]. We note that a structurally controllable system is controllable for almost any weight configuration [14]. According to the independent path theorem [17], a temporal network is structurally controllable at target interval $t_f$ in the desired $\Delta t$ steps if all nodes $v_i \in V$ at deadline $t_f$ are connected to the intervention points through a set of independent time-respecting paths originated within $(t_f - \Delta t, t_f]$; this is under the assumption that Equation 2.3 governs the system's dynamics. A set of time-respecting paths are independent if they do not pass the same node at a same time. Fig. 2 illustrates two scenarios for controlling the simple temporal network introduced in Fig. 1. The intervention points are marked with the red nodes, and the independent time-respecting paths originating from them are marked with the solid red arrows. In Fig. 2 (a), we only illustrate the first interval of the network since the target deadline to reach control is $t_f = 1$. The figure shows at least three driver nodes are required to reach control. However, for deadline $t_f = 2$ in Fig. 2 (b), at least two driver nodes are needed to fully control the network (*node 2* is stimulated twice) since the self-loop (on *node 3* at interval $t = 2$) enables preserving the influence of *node 4* on *node 3* over the interval $t = 2$.

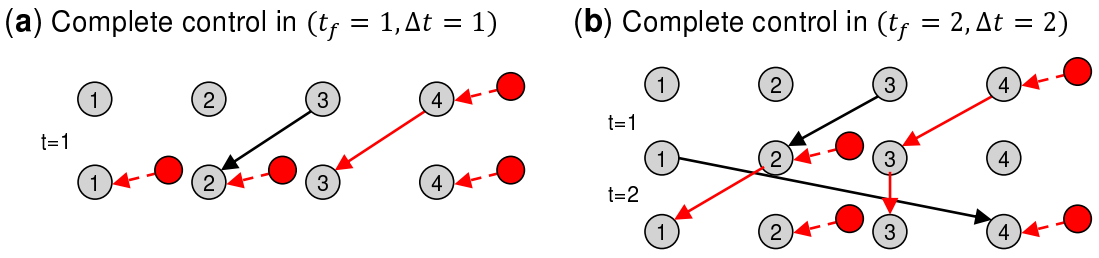


FIG. 2. Controlling the simple temporal network illustrated in Fig. 1. The driver nodes and independent time-respecting paths are marked with the red colors. (a) Three driver nodes are required to reach full control over interval $t = 1$. (b) Two driver nodes are needed to reach control over interval $t = 2$ as the self-loop of *node 3* at the interval $t = 2$ enables controlling *node 3* through *node 4*.

2.2.2 *MCS for a Set of Driver Nodes* An Maximum Controllable Subspace (MCS), namely $C \subseteq V$, at any deadline $t_f$ for a set of driver nodes $d_i \in V$ can be efficiently identified using Ford-Fulkerson algorithm [17]. The problem is equivalent to obtaining a maximum flow of the time-layered network by connecting the source node to all driver nodes in all of the time-layers, next connecting all the nodes at the deadline time-layer to the sink node, and then limiting the maximum capacity that can pass through each node to one (see Appendix A for details). Fig. 3 illustrates the identification of an MCS for a temporal network by using the maximum flow approach. To improve visualization, we selected distinct colors for the independent time-respecting paths in Fig. 3 (c) and (d). Also, we grouped the connections from the source node to the driver nodes. The colored dotted edges from the source node represent the selected intervention points by the maximum flow algorithm. The temporal network in Fig. 3 requires at least two driver nodes to reach control at deadline $t = 3$ given the possibilities of independent time-respecting paths. In many control problems, the goal is to identify a minimally sized set of nodes that can control a network.

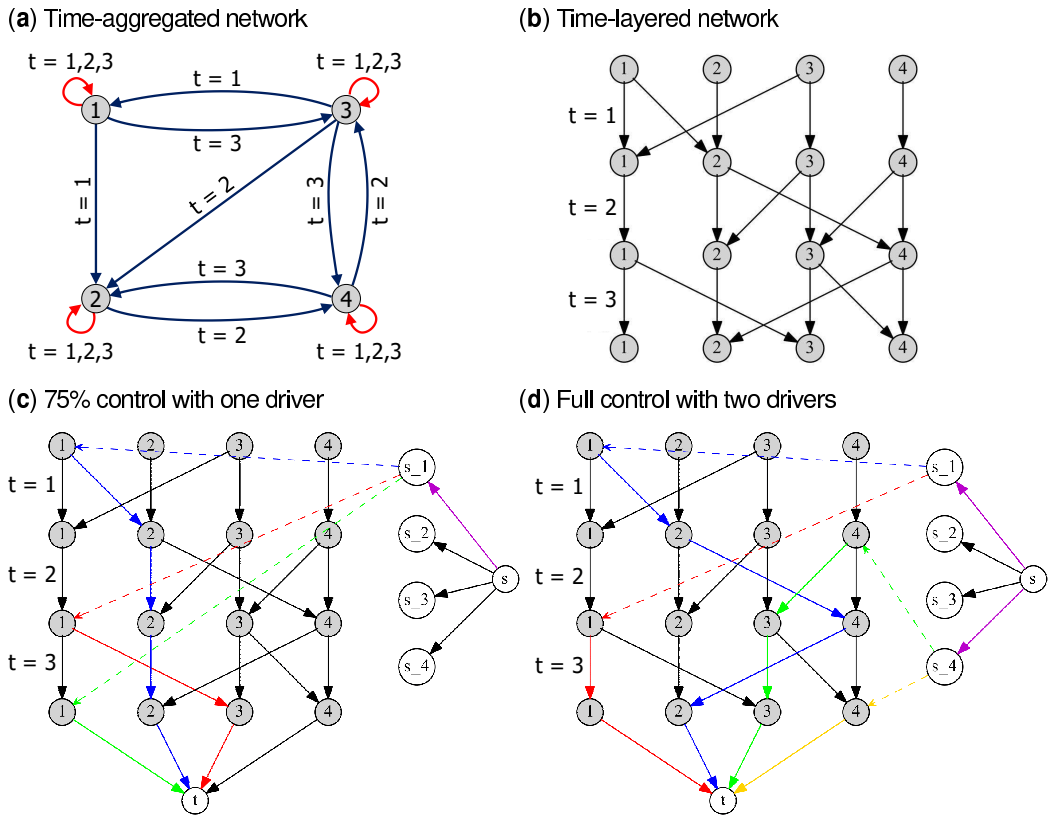


FIG. 3. Identifying an MCS using the maximum flow algorithm. The independent time-respecting paths are marked with distinct colors. The intervention points connected to driver nodes are marked with dotted lines. (a) Time-aggregated network with timestamped edges and observation period $1 \leqslant t \leqslant 3$. (b) The temporal pathways are visualized with the time-layered network. (c) *Node 1* as a single driver node at most can control 75% of the network (three nodes). (d) *Node 1* and *Node 4* together can fully control the network.

## 3. Problem

Finding an MDS is computationally prohibitive due to two difficulties: 1) identifying the minimum number of driver nodes and their intervention points requires testing $O(2^{N\Delta t})$ configurations where $\Delta t = t_f - t_0$ is the desired number of steps to reach control at deadline $t_f$, and 2) testing a configuration requires computing rank of the temporal controllability matrix $C\left(t_0, t_f\right)$ as found in Equation 3.1; this latter step involves matrix multiplications on the order of $O(A(t)^{\Delta t})$ [17]. Without loss of generality, we assume the start interval is $t_0 = 0$ within this article. There is no loss of generality as control can be applied within any period $(t_f - \Delta t, t_f]$.

$$C(t_0, t_f) = \Big[A(t_f - 1)A(t_f - 2)\cdots A(t_0 + 1)B(t_0); \cdots ; A(t_f - 1)B(t_f - 2); B(t_f - 1)\Big] \tag{3.1}$$

The operator $[X;Y]$ represents the concatenation of two matrices and hence $C(t_0, t_f) \in \mathbb{R}^{N \times N_I}$ where $N_I = \sum_t N_I(t)$ is the total number of interventions. As the linear rank of $C(t_0, t_f)$ is the number of variables that can be controlled independently, the system $\langle A(t), B(t)\rangle$ (governed by Equation 2.3) becomes controllable if $rank(C(t_0, t_f)) = N$. This can be done by a proper selection of intervention points via the driver nodes.

Constructing an optimal driver node set (i.e., MDS) that satisfies the aforementioned rank condition is computationally prohibitive, and thus, we propose a heuristic algorithm to efficiently control the temporal networks.

## 4. Heuristic for Efficiently Control Temporal Networks

To start introducing the heuristic approach, we first need to introduce two properties that correlate the MCS of individual nodes to the concepts of MDS (optimal sets in terms of size) and SMDS (i.e., suboptimal driver sets that can fully control a temporal network). As discussed in Section 3, finding an MDS with size $N_c$ is computationally prohibitive. In this work, our goal is to efficiently create an SMDS with size $N_s \geqslant N_c$. According to the independent path theorem in [17], a subset of network variables (nodes) $C \subseteq V$ is controllable at deadline $t_f$ in a desired $\Delta t$ steps iff $|C|$ number of independent time-respecting paths exists that a) start from intervention points within $(t_f - \Delta t, t_f]$ period, and b) end on the nodes $v_k \in C$ at deadline $t_f$. Moreover, a structurally controllable temporal network has at least $|V|$ number of independent time-respecting paths originating from the driver nodes within $(t_f - \Delta t, t_f]$ and end on all nodes $v \in V$ at deadline $t_f$.

Here we focus on the problem of identifying an SMDS, namely $D$, with its MCS denoted by $M_C(D, t_f, \Delta t) = V$ where function $M_C \subseteq V$ finds a maximum subset of nodes that can be controlled by applying interventions at points $(d \in D, t)$ for any interval in $t_f - \Delta t < t \leqslant t_f$. This leads to the formation of the following properties.

Property 1 $\sum_{d \in L} |M_C(\{d\}, t_f, \Delta t)| \geqslant |V|$ for any MDS, namely $L$, where function $M_C(X, t_f, \Delta t) \subseteq V$ finds an MCS for any driver node set $X \subseteq V$ by applying interventions at points $(d \in X, t)$ within any interval in $t_f - \Delta t < t \leqslant t_f$, and $V$ is the set of nodes in the temporal network.

*Proof.* Recall that function $M_C(X, t_f, \Delta t)$ finds an MCS for any driver node set $X$ based on a maximum flow of $X$, denoted by $f(X)$, in the time-layered network constructed for the interval $(t_f - \Delta t, t_f]$ (Section 2.2.2). Therefore, $|M_C(\{d\}, t_f, \Delta t)| = f(\{d\})$ for any node $d \in X$ and hence $\sum_{d \in X} |M_C(\{d\}, t_f, \Delta t)| = \sum_{d \in X} f(\{d\})$ for any driver node set $X$. Now, suppose there exists an MDS, namely $L$, that can fully control the network, i.e., $f(L) = |V|$. It is enough to show that $\sum_{d \in L} f(\{d\}) \geqslant |V|$. Assume otherwise, i.e., $\sum_{d \in L} f(\{d\}) < |V|$. Notice that by structure of the time-layered network, and the fact that the

maximum capacity that can pass through each node is one $f(L) \leqslant \sum_{d\in L} f(\{d\})$ holds. Thus, combined with the earlier assumption we get $f(L) \leqslant \sum_{d\in L} f(\{d\}) < |V|$, and thus $f(L) < |V|$ would hold. This contradicts the independent path theorem; that is, to gain full control all nodes $v_i \in V$ at deadline $t_f$ must be connected through a set of independent time-respecting paths originated from intervention points $(d \in L, t)$ within time-layers in interval $t_f - \Delta t < t \leqslant t_f$. This in turn would mean that $L$ cannot be an MDS, and as this contradicts our original supposition regarding $L$, $\sum_{d\in L} f(\{d\}) < |V|$ does not hold and accordingly $\sum_{d\in L} f(\{d\}) \geqslant |V|$. □

Based on Property 1, the sum of the flow of individual driver nodes for any driver node set that can reach full control must be equal to or bigger than $|V|$. This leads to the formation of Property 2 that belongs to all MDSs and SMDSs.

Property 2 If $\sum_{d\in D'} |M_C(\{d\}, t_f, \Delta t)| \geqslant |V|$, then the driver set $D' \subseteq V$ is a possible SMDS within interval $(t_f - \Delta t, t_f]$.

*Proof.* By definition, an MDS, namely $L$, is a driver node set with the minimum number of driver nodes $N_c$ that can fully control the network, i.e., $M_C(L, t_f, \Delta t) = V$. Also, an SMDS, namely $D'$, is a driver node set with size $N_s \geqslant N_c$ that can fully control the network, i.e., $M_C(D', t_f, \Delta t) = V$. Assume there exists a possible SMDS, namely $D'$, where $\sum_{d\in D'} |M_C(\{d\}, t_f, \Delta t)| < |V|$. Since $D'$ could be an SMDS only if it can gain full control, then the proof of this property follows the same logic as the property 1. □

The heuristic identifies a possible SMDS that can fully control a temporal network by applying the Property 2 to a list of MCSs denoted by $\mathbb{C}$. We say the set $S_{v_k}$ is an MCS of node $v_k$ such that $S_{v_k} = M_C(\{v_k\}, t_f, \Delta t) \subseteq V$. We propose to construct a possible SMDS, namely $D'$, from an MCS list denoted by $\mathbb{C}$ in the following way:

1. Choose all MCSs of nodes to include in the MCS list $\mathbb{C} = [S_{v_1}, S_{v_2}, \ldots, S_{v_{k=N}}]$. For each node added, we keep record of the set of nodes it controls.

2. Add MCS of nodes to the list $\mathbb{C}$ in decreasing order of the size of subspaces such that the sets $S_{v_k} \in \mathbb{C}$ follow the order $|S^1| \geqslant |S^1| \geqslant \cdots \geqslant |S^N|$ where the superscripts denote the position of sets in $\mathbb{C}$. In other words, the sets in $\mathbb{C}$ are sorted in decreasing order of their respective cardinalities.

3. Starting from the first set $S^{i=1}$, subtract its controllable subspace from the controllable subspace of sets located after it (i.e., $S^{j>i} \in \mathbb{C}$). In other words, $S^j \leftarrow S^j \setminus S^i$ where $1 \leqslant i < j \leqslant N$ denotes the position of members in $\mathbb{C}$ which themselves are sets.

4. Stop when all sets in $\mathbb{C}$ are processed. Nodes with non-empty controllable subspace are driver nodes. That is, a possible SMDS denoted by $D'$ is derived from $\mathbb{C}$ where $P = \{v_k | S_{v_k} \in \mathbb{C}, S_{v_k} \neq \emptyset\}$.

5. If $M_C(D', t_f, \Delta t) = V$, then $D'$ is an actual SMDS, namely $D$, that can fully control the temporal network. Otherwise, $D'$ partially controls the network.

In summary, the proposed heuristic algorithm converts an MCS list $\mathbb{C}$ to a possible SMDS by enforcing Property 2. Therefore, the heuristic ensures any MCS list $\mathbb{C}$ satisfies the following: a) $\bigcup_{S_{v_k}\in\mathbb{C}} S_{v_k} = V$, b) $\bigcap_{S_{v_k}\in\mathbb{C}} S_{v_k} = \{\emptyset\}$, and c) $\sum_{S_{v_k}\in\mathbb{C}} |S_{v_k}| = |V|$; these increase the likelihood of deriving a possible SMDS that is an actual SMDS (i.e., can fully control the network) or control a significant percentage of the network.

In Fig. 4, we illustrate an execution of the heuristic approach on a simple temporal network. Fig. 4 (a) visualizes a step by step execution of the heuristic on the indicated MCS list $\mathbb{C}$, which is induced

from the temporal network in Fig. 4 (b). The blue arrows indicate the MCS of a node that is being processed in each iteration of the heuristic. The heuristic starts with processing $S^1_{v_2}$, i.e., the first MCS in $\mathbb{C}$ where the superscript indicates the position of the sets in $\mathbb{C}$. In *Iteration 1*, the common vertices within the controllable subspace of $S^1_{v_2}$ and its following subspaces are marked red. Therefore, $S^3_{v_1}$ and $S^4_{v_3}$ are adjusted based on the assumption that $S^1_{v_2}$ can control $v_1$ and $v_3$. These iterations continue until all $S_{v_i} \in \mathbb{C}$ are processed. Thereby Property 2 will be enforced on $\mathbb{C}$. In this example, the derived driver set is both an SMDS and MDS. In Fig. 4 (b), we use the identified SMDS to control the network on deadline $t_f = 3$ and in $\Delta t = 2$ steps. The independent time-respecting paths and the intervention points required to control the network are marked with the red nodes and links.

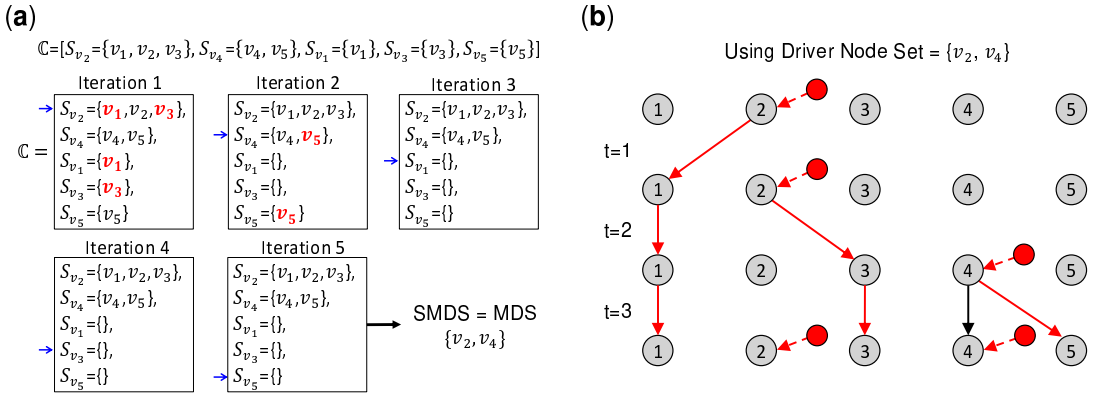


FIG. 4. Execution of the heuristic approach on a simple temporal network. (a) Execution of the heuristic for the MCS list $\mathbb{C}$. The heuristic identified one SMDS that is also the optimal solution (MDS). Blue arrows indicate the controllable subspace of a node that is being processed by the heuristic. An MCS for each node is denoted by $S_{v_k} = M_C(\{v_k\}, t_f = 3, \Delta t = 3)$. (b) Controlling the temporal network with the identified SMDS. The intervention points and their independent time-respecting paths are marked with the red nodes and links.

### 4.1 *Multiple SMDSs and Non-unique Controllable Subspace*

Having multiple driver nodes simultaneously could affect the number of independent time-respecting paths in a network. This means the MCS of multiple driver nodes is not always equal to the union of their MCSs; i.e., $M_C(\{v_1, v_2\}, t_f, \Delta t) \neq M_C(\{v_1\}, t_f, \Delta t) \cup M_C(\{v_2\}, t_f, \Delta t)$ where the function $M_C$ finds an MCS for a set of nodes. Furthermore, the MCS of a node often may have several possible solutions. For example, Fig. 5 illustrates an example by adding the temporal edge $(v_1 \to v_4, t = 2)$ to the network of Fig. 4. With the presence of the new edge, now there are three sets with size 3 that could be selected as an MCS for node $v_2$ by the maximum flow algorithm, and there are two possibilities for node $v_1$. For instance, in Fig. 5 function $M_C(v_1, t_f = 3, \Delta r = 3)$ could return one of the sets in $\{\{v_1, v_4\}, \{v_1, v_5\}\}$ as an MCS for node $v_1$. Suppose function $M_C$ selects $S_{v_1} = \{v_1, v_4\}$ (i.e., an MCS of node $v_1$) as well as $S_{v_2} = \{v_2, v_3, v_5\}$ (i.e., an MCS of node $v_2$). Based on all possible independent time-respecting paths $M_C(\{v_1, v_2\}, t_f = 3, \Delta t = 3) \in Z = \{\{v_1, v_2, v_3, v_4\}, \{v_1, v_2, v_3, v_5\}\}$. We can see that for the selected MCSs $S_{v_1} \cup S_{v_2} \notin Z$. On the other hand, if $M_C$ selects $S'_{v_2} = \{v_1, v_2, v_3\}$ then in this case $S_{v_1} \cup S'_{v_2} \in Z$. In the network of Fig. 5, the edge $(v_1 \to v_2, t = 2)$ enables the possibilities for node $v_1$ to either control

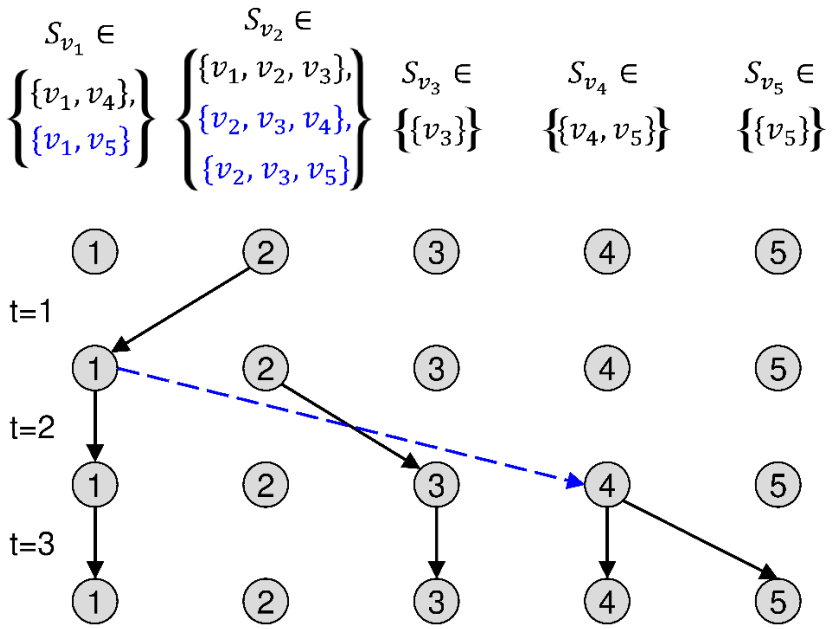


FIG. 5. Non-unique MCSs. (a) Identical network as Fig. 4 (b) except the new temporal edge from $(v_1 \to v_4, t = 2)$. The new edge extends the MCS possibilities for $S_{v_1}$ and $S_{v_2}$. The blue sets mark the new MCS possibilities compared to the network in Fig. 4 (b).

node $v_4$ or $v_5$. However, this new edge also increases the possible MCS choices for nodes $v_1$ and $v_2$. In turn, the size of an SMDS created by the heuristic approach varies depending on which MCS is selected by the maximum flow algorithm.

To address the aforementioned problem, we introduce a complementary step to our heuristic that enables the creation of multiple SMDSs on the basis of the following three considerations: 1) the number of SMDSs discovered could be used to characterize the control behavior of the network and make approximations (see Section 6.1), 2) creating multiple SMDSs could increase the chances of finding a smaller set of driver nodes, and 3) creating multiple SMDSs would increase the chance of finding at least one SMDS that can fully control the network. As discussed later, our empirical results show all of the derived possible SMDSs identified by the heuristic approach are an actual SMDS that can reach full control.

We introduce a branching process to create multiple MCS lists stemmed from the initial list $\mathbb{C}$. We refer to a stemmed MCS list by $\mathbb{C}'$. The order of sets in $\mathbb{C}$ plays a crucial role in our heuristic. Therefore, the branching process creates new stems by switching the position of sets in $\mathbb{C}$. The branching process is in the following way: 1) in execution of the heuristic, while iterating for each set $S^i \in \mathbb{C}$ to subtract $S^{j>i}$ from the controllable subspace $S^i$; before subtracting any two controllable subspaces, create a stem $\mathbb{C}' \leftarrow \mathbb{C}$ if the intersection of those two subspaces is not empty (i.e., if $S^i \cap S^j \neq \emptyset$), 2) switch the location of those two sets in $\mathbb{C}'$, 3) run the heuristic on each created $\mathbb{C}'$ without branching (only one level of branching is allowed). Fig. 6 illustrates an example of the aforementioned branching process. Each row presents an execution of the heuristic for the MCS list $\mathbb{C}$ and its stems $\mathbb{C}'$. Overall, the branching process creates four stems denoted by $\mathbb{C}'$. Running the heuristic on $\mathbb{C}$ and its stems results in four unique SMDSs with their sizes ranging between 2 and 4 (Fig. A.12 provides the execution of the heuristic for each stem).

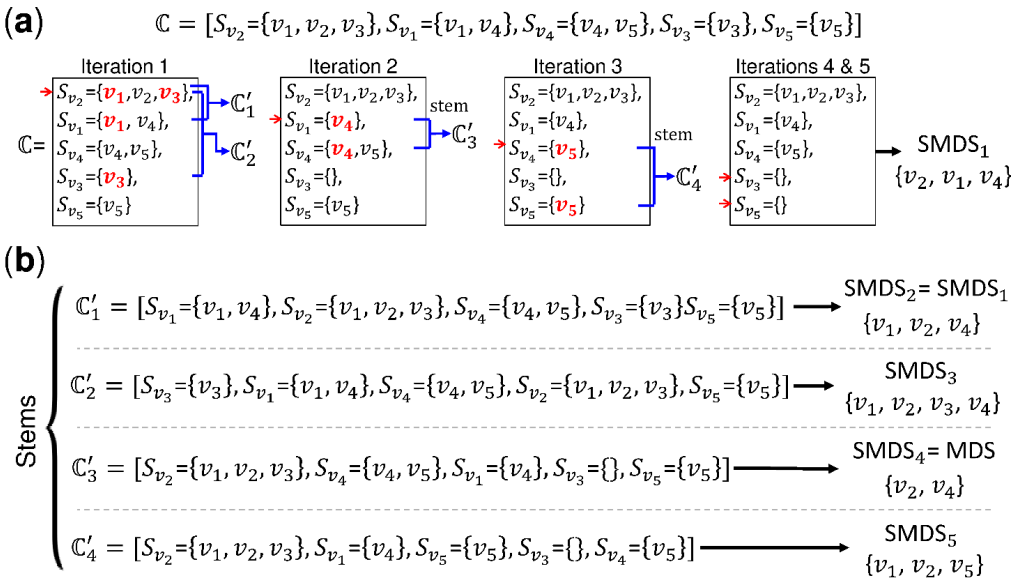


FIG. 6. Illustration of the branching process for the temporal network in Fig. 5. (a) Execution of the heuristic approach with branching enabled. Each iteration focuses on processing the controllable subspace of a node based on the order of nodes in $\mathbb{C}$. The red arrows mark the controllable subspace set of a node being focused in an iteration. On each iteration, the heuristic creates stems denoted by $\mathbb{C}'$. (b) The heuristic identified four unique possible SMDSs that are actual SMDSs with their sizes ranging from 2 and 4. Execution of the heuristic for each stem is illustrated in Appendix B.

The problem of finding independent time-respecting paths to satisfy the independent path theorem (see Section 2.2.1) resembles the Maximum Node Disjoint Paths (MNDP) problem [33]. The objective of MNDP problem is to find the maximum cardinality of source-sink pairs $M' \subseteq \{(s_1,t_1),\ldots,(s_k,t_k)\} \subseteq V \times V$ that are connected via node disjoint paths. The focus of this article differs from MNDP in two aspects. First, the objective of MNDP is finding a maximum cardinality, whereas the focus of this article is finding a minimum cardinality. Precisely, the goal of this article is finding a minimum number of driver nodes that can fully control a temporal network. To do this, we reduced the problem to finding node disjoint paths in the time-layered network in which each time-layer represents a copy of all nodes and the aim is to minimize the size of union of selected driver nodes within all time-layers. Second, to reach full control, exactly $|V|$ time-respecting paths (i.e., node disjoint paths in the time-layered network) with specific sinks are required; however, the objective of MNDP problem is finding the maximum number of node disjoint paths between all source-sink pairs.

Moreover, MNDP is an optimization version of the classical Node Disjoint Paths (NDP); NDP is a decision problem with the objective to verify whether all pairs of source and sink nodes in a graph can be connected via node disjoint paths. Karp shows when the number of source and sink pairs is an input, the NDP problem is NP-complete [34]. Furthermore, NDP and MNDP problems can be reduced to Edge Disjoint Paths (EDP) and Maximum Edge Disjoint Paths (MEDP) problems. In fact, MEDP is also NP-hard [33]. Approximation algorithms for MNDP and MEDP are extensively discussed in the area of graph theory with important applications in computer-communication networks [35–37]. However, adapting such approximation algorithms to satisfy the independent path theorem requirements is non-trivial.

### 4.2 *Algorithms and Complexity of the Heuristic*

For the sake of clarity and simplicity, the proposed heuristic approach is presented in two Algorithms. Algorithm 1 manages the necessary tasks to create an SMDS for a temporal network within three phases; namely *initialization*, *creation*, and *selection* phases. A brief description of the phases follows: 1) the *initialization* phase initializes the necessary variables to execute our proposed heuristic, 2) the *creation* phase employs Algorithm 2 to convert an input MCS list $\mathbb{C}$ to a possible SMDS, namely $D'$, and create the stems of $\mathbb{C}$, and 3) the *selection* phase validates the derived driver node sets from $\mathbb{C}$ and its stems $\mathbb{C}_{stem}$ to ultimately return an actual SMDS, namely $D$, with the smallest number of driver nodes. The total complexity of our heuristic is equal to the sum of the complexity of each phase. This section analyzes the three phases individually and derives the complexity class of the heuristic as follows:

(1) Initialization phase (Lines 2 - 4 of Algorithm 1): This phase starts with computing a maximum flow for each node on the time-layered network. Building the time-layered network requires $O(N\Delta t + E)$ computation where $\Delta t = t_f - t_0$ is the desired number of steps to reach control and $E$ is the number of edges in the temporal network. Since the capacity of edges in the time-layered network is always equal to one and the value of maximum flow is $N$, employing the Ford-Fulkerson algorithm requires $O(NE)$ computation[1] [30, 38]. We need to compute an MCS for each node in the network. As a result, computing the MCSs for all of the network nodes is on the order of $O(N^2E)$ (Line 3 of Algorithm 1). We then sort the created list $\mathbb{C}$ in the descending order of the size of its members (which themselves are sets); this requires $O(N\log N)$ computations. Accordingly, the total complexity of *Initialization* phase is $O(N^2E + N\Delta t)$.

(2) Creation phase (Lines 5 - 11 of Algorithm 1): The *creation* phase uses Algorithm 2 for two purposes: a) Creating a possible SMDS from an MCS list takes $O(N^2)$ computations because of the nested loops in Lines 3 and 4 of Algorithm 2 and b) Employing the branching process to create stems of the MCS list $\mathbb{C}$ denoted by $\mathbb{C}_{stem}$ in Algorithm 1. The maximum number of stems from an MCS list $\mathbb{C}$ is on the order of $\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} 1 = O(N^2)$; that is, in each iteration of the Algorithm 2, all of the subspace sets $S^{j>i}$ must have a common node with $S^i$ where $1 \leqslant i < j \leqslant N$ denotes the position of members in $\mathbb{C}$ which themselves are sets (Fig. 6 illustrates an example). Hence, the *creation* phase is $O(N^4)$ since we need to run Algorithm 2 for each stem.

(3) Selection phase (Lines 12 - 18 of Algorithm 1): Finally, in the *selection* phase, we sort the created possible SMDSs in an ascending order (Line 13 of Algorithm 1). Next, starting from the possible SMDSs (stored in set $P$) with the smallest size, we look for an actual SMDS that can fully control the network. This requires testing the created set of possible SMDSs (denoted as $P$ in Algorithm 1) by computing the maximum flow of the possible SMDSs. Since the maximum number of stems is expected to be linear (explained below), at worse the validation requires a call to Ford-Fulkerson for all of the stems which results in overall computation of $O(N^2E)$.

Altogether, computational complexity of the proposed heuristic is $O(N^4)$. It is important to note that the derived complexity of our proposed heuristic is a factor of $N$ lower in practice. For instance, even though the maximum number of stems is $O(N^2)$, as suggested by empirical results in Section 5.1 it is expected to be $O(N)$. Thereby, in practice the expected computational cost is closer to $O(N^3 + N^2E + N\Delta t)$.

[1]The time complexity of Ford-Fulkerson algorithm is $O(Ef^*)$ where $f^*$ is the value of maximum flow.

**Algorithm 1** Create a single SMDS

**Inputs:** (1) $V$: Set of nodes, (2) $t_f$: Deadline layer, (3) $\Delta t$: Number of steps.
**Output:** One SMDS, denoted by $D$, that can fully control the temporal network.

**procedure** FIND_SMDS($V, t_f, \Delta t$)
  //Initialization phase
  $\mathbb{C} \leftarrow [S_{v_k} = M_C(\{v_k\}, t_f, \Delta t) | v_k \in V]$
  **Sort** $\mathbb{C}$ with descending order of its elements size (flows)
  //Creation phase
  $D', \mathbb{C}_{stem} \leftarrow$ Driver_Selection($\mathbb{C}$, True)
  $P \leftarrow D'$ // Store all possible SMDSs
  **for each** $\mathbb{C}'$ **in** $\mathbb{C}_{stem}$ **do**
    $D' \leftarrow$ Driver_Selection($\mathbb{C}'$, False)
    $P \leftarrow P \cup D'$
  **end for**
  //Selection and Validation phase
  **Sort** $P$ in ascending order of its elements size
  **for each** $D'$ **in** $P$ **do**
    **if** $M_C(D', t_f, \Delta t) == V$ **then**
      **return** $D \leftarrow D'$ //therefore, $D'$ is an actual SMDS
    **end if**
  **end for**
**end procedure**

**Algorithm 2** Heuristic for Driver Node Selection

**Inputs:** (1) $\mathbb{C}$: a maximum controllable subspace (MCS) list,
(2) *stem*: a flag to allow creating stems of $\mathbb{C}$ denoted by $\mathbb{C}'$,
**Outputs:** (1) $D'$: a possible SMDS derived from $\mathbb{C}$.
(2) $C_{stem}$: a set of stems created from $\mathbb{C}$.

**procedure** DRIVER_SELECTION($\mathbb{C}, stem$)
  $\mathbb{C}_{stem} \leftarrow \{\}$
  **for each** $S^i$ **in** $\mathbb{C}$ **;** $i = 1, 2, \ldots, |\mathbb{C}|$ **do**
    **for each** $S^j$ **in** $\mathbb{C}$ **;** $j = i+1, i+2, \ldots, |\mathbb{C}|$ **do**
      **if** *stem* **is** True **and** $|S^i \cap S^j| > 0$ **then**
        $\mathbb{C}' \leftarrow \mathbb{C}$
        $\mathbb{C}' \leftarrow$ Switch the position of $S^i$ and $S^j$ in $\mathbb{C}'$
        $\mathbb{C}_{stem} \leftarrow (\mathbb{C}_{stem} \cup \mathbb{C}')$
      **end if**
      $S^j \leftarrow S^j \setminus S^i$
    **end for**
  **end for**
  $D' \leftarrow \{v_k | S_{v_k} \in \mathbb{C}, S_{v_k} \neq \{\emptyset\}\}$
  **return** $D', \mathbb{C}_{stem}$
**end procedure**

### 4.3 *Discussion on the Algorithms*

The key idea of our proposed heuristic algorithms is to find a single or multiple SMDSs by strategically selecting driver nodes using a single MCS list, denoted by $\mathbb{C}$ (which is sorted in a descending order of the size of its elements $S_{v_k} \in \mathbb{C}$). The goal is to find a small number of driver nodes to obtain a maximum flow in the time-layered network. As illustrated in Fig. 6, with the branching technique our proposed heuristic generates possible SMDSs with various sizes. In the worst case if all possible SMDSs with the smallest size $s$ fail to reach full control, by strategically increasing the number of driver nodes to $s+1$ (and beyond if necessary), we increase the probability of finding at least one SMDS. However, our empirical results on the real-world and synthetic networks show that all possible SMDSs with sizes $s$ and $s+1$ passed the verification test (i.e., they are actual SMDSs).

Moreover, each driver node increases the number of possible intervention points by the count of time-layers. That is, each driver node adds at least $\Delta t$ possibilities to find $|V|$ independent time-respecting paths. The process of testing multiple SMDSs is implemented in the *selection* phase (Lines 12 - 18 in Algorithm 1). The algorithm begins by testing the smallest possible SMDSs (i.e., size $s$), and it continues testing the remaining possibilities with bigger sizes (i.e., $s+1$ and beyond if necessary).

Furthermore, there is a population-based heuristic [39] extension of our algorithm to create more than one generation of possible SMDSs by varying the order of elements in an MCS list. However, the empirical results support that one generation is enough to find SMDSs with size $N_s$ close to $N_c$ (the minimum number of driver nodes).

## 5. Results

We evaluated our proposed heuristic algorithm on synthetic temporal networks and two real-world temporal networks induced by a) interactions between ants in six colonies, and b) e-mail communications between employees of a mid-sized manufacturing company [40–42]. We specifically picked these real-world datasets because they exhibit different behaviors of complex systems. For example, the ant colonies self-organize whereas, in contrast, hierarchical organizations rely on key employees for management. Therefore, we expect their respective temporal networks to exhibit dissimilar behaviors. We provide a showcase that the proposed heuristic can identify those expected behaviors.

All evaluations assume nodes have self-loops within all intervals to enable the state retention for all nodes. There is no loss of generality since by modifying the self-loops, analysis without state retention or with conditions to allow state retentions can be conducted. For the ants datasets, we studied the controllability within the entire observation periods of datasets. In other words, $\Delta t = t_f - t_0$ is the number of intervals between observations, and deadline $t_f$ is the last observation. For the e-mails dataset, we set the resolution to one-hour intervals between observation period of 2010-01-05 06:00 AM to 2010-09-29 11:00 AM. More details on the datasets' observation periods and frequency of interactions are provided in Appendix C. We computed the optimal solutions (MDSs) for the ant colony datasets using a branch-and-bound brute force approach based on Property 1. However, for the e-mails dataset, we could not find an optimal solution due to the factorial growth of combinations. Fig. 7 shows the minimum number of driver nodes required for the heuristic ($N_s$) and the optimal ($N_c$) solutions. The heuristic approach was able to fully control the networks with a few number of driver nodes. Also, for the ant colonies 3-1, 3-2, and 6-1 the heuristic approach found optimal solutions.

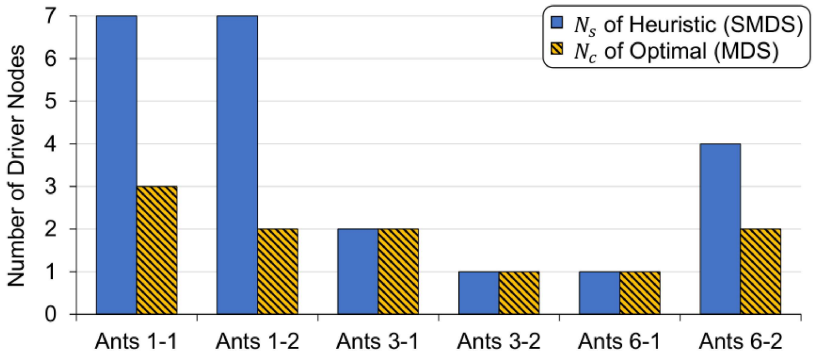


FIG. 7. The minimum number of driver nodes for the optimal ($N_c$) and heuristic ($N_s$) solutions within ants' interactions datasets.

### 5.1 *Analyzing Multiple Driver Node Sets*

Most networks have multiple MDSs with size $N_c$ that increases the domain of driver nodes. Moreover, this led to categorizing nodes in three groups with respect to their contribution within all MDSs. These groups are called *critical*, *intermittent*, and *redundant* [43–45]. A *critical* node exists in all MDSs, i.e., the *critical* nodes are always driver nodes. An *intermittent* node exists in at least one MDS. Lastly, the *redundant* nodes do not belong to any MDS (never employed as a driver node).

With a minor modification, our proposed heuristic approach can generate multiple SMDSs with a size ranging from $N_s$ and larger. To create multiple SMDSs, instead of returning a single SMDS at Step 16 of Algorithm 1, one can keep record of the validated SMDSs and return them when validation is done. To further explore the behavior of driver nodes in control, we only study the intersections of nodes within SMDSs of size $N_s$ and $N_s+1$. We note that the purpose of this study is not establishing a definition for characterizing driver nodes for the temporal network, and instead, our focus is on analyzing the behavior of the heuristic approach and the proposed network model.

Details of the results provided in Fig. 7 are presented in Table 1 for both multiple MDSs and SMDSs of size $N_s$ and $N_s+1$. The column *count MDSs* presents the number of discovered optimal solutions using brute force. However, we could not find an optimal solution for the e-mails dataset because it requires a larger number of driver nodes; e.g., if $N_c \approxeq 50$ then $\binom{109}{50} = 109!/(50!59!)$ number of driver sets needs to be validated. Also, the column Count SMDSs presents the number of SMDSs found with size $N_s$ and $N_s+1$. To analyze the efficiency of the branching process, column $|P|$ presents the total number of tested driver sets (the initial MCS list $\mathbb{C}$ plus the number of its stems with size $N_s$). Lastly, Table 1 shows the fraction of *intermittent*, *redundant*, and *critical* nodes denoted by columns $f_i'$, $f_r'$, and $f_c'$ for the heuristic approach and $f_i$, $f_r$, and $f_c$ for the optimal solutions.

Also, in Section 4.2 we mentioned that the empirical results suggest the maximum number of stems is expected to be in the order of $O(N)$ instead of $O(N^2)$. This is illustrated with the column *Count Stems* in Table 1 indicating the number of stems linearly grows with respect to N. For the ant colonies 3-1, 3-2, and 6-1, the heuristic approach finds optimal solutions since $N_s = N_c$ (marked with a star). This can be explained by the degree of uniqueness within the discovered MCSs of nodes in a dataset. That is, the intersection between the discovered single MCSs for each node is empty or small. In contrast, within the ant colonies 1-1, 1-2, and 6-2, the discovered MCSs of nodes have a large intersection that intuitively increases $N_s$. One can conduct a more precise analysis on the intersections of MCSs by computing all of the MCSs of each individual node. However, computing all MCSs of nodes is computationally

prohibitive. Overall, the proposed heuristic approach fully controls the ants' interactions networks with few extra driver nodes compared to the optimal solution. Also, the experiments show all of the created possible SMDSs with size $N_s$ and $N_s + 1$ are actual SMDSs (i.e., in Table 1, the values of columns *Count SMDSs* and $|P|$ are equal). However, as explained in Section 4.1, this is not a guaranteed behavior, and for that reason, we included the validation process in Step 15 of Algorithm 1.

Table 1. Comparison between optimal and suboptimal solutions. The column $|P|$ presents the total number of possible SMDSs with size $N_s$ by the heuristic approach.

| Dataset | | | Optimal (MDS) | | | | | Suboptimal (SMDS) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name | $N$ | $E$ | $N_c$ | Count MDSs | $f_i$ | $f_r$ | $f_c$ | Count Stems | $N_s$ | Count SMDSs | $\|P\|$ | $f_i'$ | $f_r'$ | $f_c'$ |
| Ants 1-1 | 89 | 1911 | 3 | 153 | 0.59 | 0.41 | 0 | 231 | 7 | 2 | 2 | 0.02 | 0.91 | 0.07 |
| | | | | | | | | | 8 | 11 | 11 | 0.13 | 0.81 | 0.06 |
| | | | | | | | | | | | Average: | 0.07 | 0.86 | 0.06 |
| Ants 1-2 | 72 | 1820 | 2 | 21 | 0.20 | 0.80 | 0 | 198 | 7 | 6 | 6 | 0.15 | 0.82 | 0.03 |
| | | | | | | | | | 8 | 19 | 19 | 0.26 | 0.72 | 0.02 |
| | | | | | | | | | | | Average: | 0.2 | 0.77 | 0.02 |
| Ants 3-1 | 11 | 78 | **2** | 7 | 0.54 | 0.46 | 0 | 17 | **2*** | 1 | 1 | 0 | 0.82 | 0.18 |
| | | | | | | | | | 3 | 6 | 6 | 0.64 | 0.27 | 0.09 |
| | | | | | | | | | | | Average: | 0.32 | 0.54 | 0.13 |
| Ants 3-2 | 6 | 104 | **1** | 5 | 0.83 | 0.17 | 0 | 11 | **1*** | 5 | 5 | 0 | 0.83 | 0.17 |
| | | | | | | | | | 2 | 5 | 5 | 1.0 | 0 | 0 |
| | | | | | | | | | | | Average: | 0.5 | 0.41 | 0.08 |
| Ants 6-1 | 33 | 652 | **1** | 3 | 0.09 | 0.91 | 0 | 50 | **1*** | 3 | 3 | 0.09 | 0.91 | 0 |
| | | | | | | | | | 2 | 19 | 19 | 0.54 | 0.46 | 0 |
| | | | | | | | | | | | Average: | 0.31 | 0.68 | 0 |
| Ants 6-2 | 32 | 367 | 2 | 10 | 0.28 | 0.72 | 0 | 44 | 4 | 5 | 5 | 0.22 | 0.78 | 0 |
| | | | | | | | | | 5 | 14 | 14 | 0.53 | 0.47 | 0 |
| | | | | | | | | | | | Average: | 0.37 | 0.62 | 0 |
| E-mails 1h | 109 | 250 | Too big to compute | | | | | 56 | 51 | 6 | 6 | 0.08 | 0.50 | 0.42 |
| | | | | | | | | | 52 | 17 | 17 | 0.25 | 0.39 | 0.36 |
| | | | | | | | | | | | Average: | 0.16 | 0.44 | 0.39 |

* indicates the heuristic approach found an optimal solution.

## 5.2 *Datasets Analysis*

An interesting observation from the ant colonies dataset is that the queen ants tend to avoid becoming a driver node within all MDSs and even SMDSs. In ant societies, the queen ant's primary responsibility is reproduction and she is not directly related to the management of the colony. The fraction of queen ants within MDSs and SMDSs along with the number of interactions with queens are presented in Table 2. Although many interactions with and by the queens exist (in-edges and out-edges columns), the queen ants rarely participated as a driver node (note that colony 6-1 has two queens).

Table 2. Queen ants' interactions. Almost in all colonies queen ants are not a driver node.

| Colony | Ants (N) | Queen(s) | in-edges | out-edges | Number of ants contacted | Fraction within all MDSs | Fraction within all SMDSs |
|---|---|---|---|---|---|---|---|
| Ants 1-1 | 89 | Q1 | 42 | 30 | 23 | 4% | 0% |
| Ants 1-2 | 72 | Q1 | 17 | 23 | 21 | 0% | 30% |
| Ants 3-1 | 11 | Q1 | 3 | 4 | 3 | 0% | 0% |
| Ants 3-2 | 6 | Q1 | 10 | 7 | 4 | 0% | 10% |
| Ants 6-1 | 33 | Q1 | 23 | 0 | 11 | 0% | 0% |
| | | Q2 | 14 | 26 | 15 | 0% | 0% |
| Ants 6-2 | 32 | Q1 | 10 | 6 | 9 | 0% | 0% |

Moreover, to illustrate the frequency of interactions with driver nodes, we compare the average in-out-total degree distributions of the driver nodes in the time-aggregated networks of ants' interactions. Fig. 8 illustrates the average degrees for all driver nodes in MDSs and SMDSs as well as the networks themselves. Overall in the ants' interaction networks, the driver nodes tend to have a large degree, and on average their out-degree is slightly higher than their in-degree. These observations contradict with what is seen in static directed networks where driver nodes tend to avoid hubs and have a smaller degree than the average degree of network [46]. The average degree of SMDSs follows the same trend as MDSs.

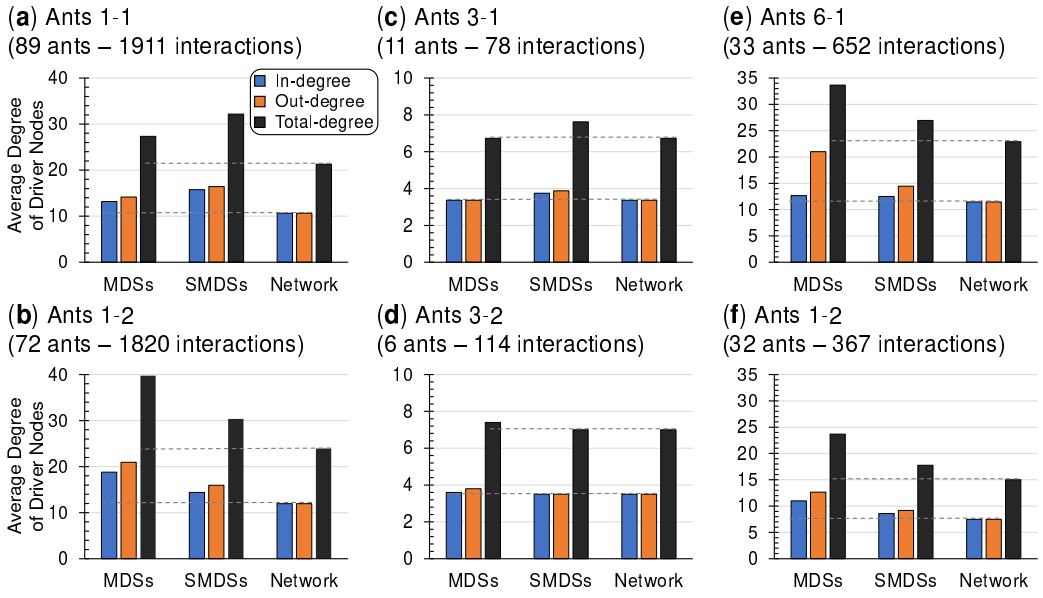


FIG. 8. Average in-out-total degree distributions of driver nodes in the time-aggregated networks of the ants' interactions datasets. The average degree of SMDSs follows the same trend as MDSs.

5.2.1 *Randomizations Analysis*

We use five different randomization techniques to assess which network or temporal characteristics influence the controllability of the ants and e-mails networks. Holme and Saramäki provide a comprehensive explanation of randomizing temporal networks in [8]. Also, we provide a brief description of the employed randomization techniques in Appendix D and, we compare their effects in Table A.3.

We present the effect of randomizing networks on $N_s$ (the smallest number of driver nodes identified by the heuristic) and $N_c$ (the minimum number of driver nodes – optimal). Fig. 9 compares $N_s$ of the randomized networks versus $N_s$ of the original networks, also the figure compares $N_c$ in the similar way. All results are averaged within 10 randomized networks for each randomization technique. Overall, the results show degree distribution has the most influence on the controllability of ants' networks. In Fig. 9 (d), DPN (Degree Preserved Network) has the smallest difference between $N_c$ of the randomized and original networks. Except for Ants 1-1 network, DPN behave similarly against our heuristic approach (comparing $N_s$ of the randomized and original networks). DPN preserves the degree distributions of temporal networks in each time-layer as well as the degree distributions of their time-aggregated network, but it eliminates the structure of network.

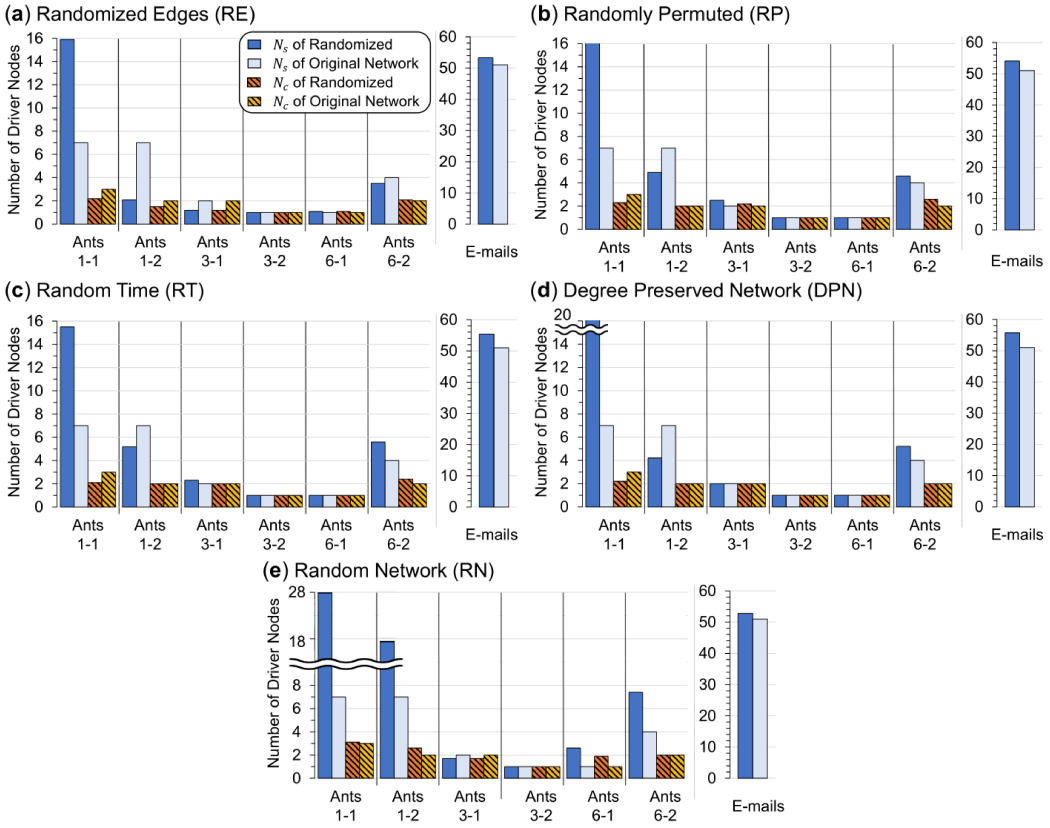


FIG. 9. Randomization of the ants and e-mails networks. The smallest number of driver nodes identified by the heuristic approach is denoted by $N_s$ and the minimum number of driver nodes is denoted by $N_c$. The results are averaged over 10 realizations for each randomization technique.

Moreover, in Fig. 9 (b-c), $N_s$ has the smallest gap against RP (Randomly Permuted) and RT (Random Time) that emphasizes the influence of both the degree distribution and the structure of networks on the heuristic's performance. Against RE (Randomized Edges), in Fig. 9 (a), the fluctuations on $N_s$ and $N_c$ show the effect of network topology. In Fig. 9 (e), RN (Random Network) shows the largest differences on both $N_s$ and $N_c$ as RN eliminates both the degree distributions of the time-aggregated network and the networks between each time-layer. Lastly, the e-mails dataset behave similarly to all randomization techniques. The reason could be the small number of time-layers (224) compared to the number of nodes (109) in the e-mails network, which is created with one-hour resolution (Fig. A.14 (g)).

Furthermore, the reason for observing small differences in the randomization effect on $N_s$ and $N_c$ in Fig. 9 could be the combination of a) the strong state retention assumption (which assumes all nodes have a self-loop) and b) the temporal characteristics of datasets. Appendix C provides an overview of the datasets' temporal characteristics. Having self-loops within all time-layers increases the controllability of temporal networks since there will be more possible combinations (i.e., $|V|$ independent time-respecting paths) to satisfy the independent path theorem (Section 2.2.1). Also, this is intensified by the rate of interactions in the ants network (small average inter-event gap) as illustrated in Fig. A.13. Plus, the ants' networks have many more time-layers (timestamps) compared to their number of nodes (Fig. A.14).

### 5.3 *Evaluating Synthetic Temporal Networks*

We evaluate the performance of our heuristic approach on Erdős-Reńyi and Barabási–Albert synthetic temporal networks with various sizes [19]. To build the synthetic networks, we generated a single Erdős-Reńyi or Barabási–Albert network for each time-layer. All temporal networks have 15 nodes and 50 time-layers; we chose 15 nodes due to the exponential cost of identifying $N_c$ with brute force. Fig. 10 presents the minimum number of driver nodes $N_c$ and the size of smallest SMDS, $N_s$, versus average degree $\langle k \rangle$ of the time-aggregated networks. All results are averaged within 10 synthetic networks for each $\langle k \rangle$. The heuristic results converge with the optimal solution as the size of networks grows (the convergence points are marked with an arrow). Overall, compared to Erdős-Reńyi, Barabási-Albert model requires requires more driver nodes and $N_s$ converges with $N_c$ within a slightly smaller average degree $\langle k \rangle$.

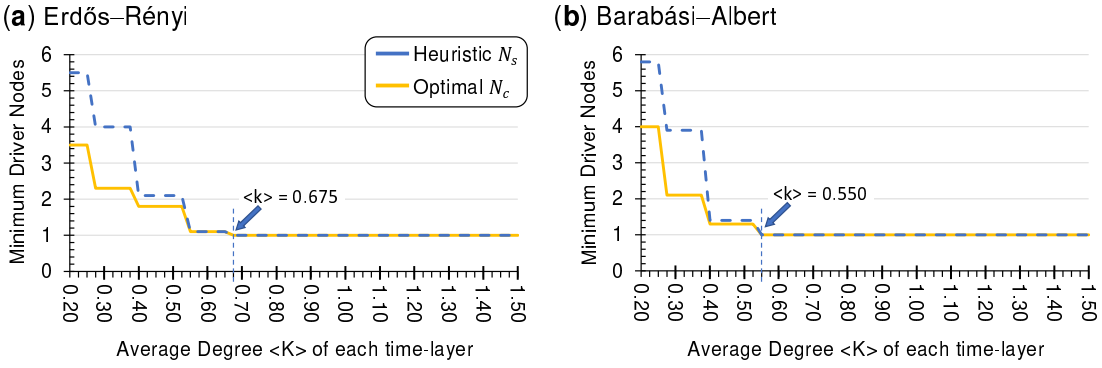


FIG. 10. Synthetic temporal networks. All networks have 15 nodes and 50 time-layers and the results are averaged within 10 networks for each average degree $\langle k \rangle$. (a) Each time-layer is connected by an Erdős-Reńyi network. (b) Time-layers are connected by generating Barabási–Albert networks.

## 6. Discussion

We conducted experiments assuming all nodes have strong state retention (by adding self-loops to all the nodes). Consequently, this allows nodes to carry their previous state in time in the absence of edges between time-layers (or lack of observations). In other words, lift the modeling constraint that the flow of influence or information depends on the existence of observations' chains.

Within the experimented datasets, the gap between interactions is relatively short compared to their observation periods (Appendix C). Therefore, we allowed strong state retention. There is no loss of generality since by removing the self-loops, analysis without state retention or with conditions to allow state retentions can be conducted with the proposed framework. However, including self-loops in the structure of networks is in line with the behavior of physical systems [22]. For example, in the e-mails dataset, lack of interactions during weekends and holidays does not convey that information from previous e-mails is forgotten.

Furthermore, the empirical results are aligned with the behavior of complex systems that shows the effectiveness of the proposed heuristic approach and network modeling. For instance, our analysis on ant networks shows queen ants tend to avoid being a driver node. This is in line with the primary responsibility of the queen that is reproduction rather than managing the colony (e.g., task management or temperature maintenance). Ant societies optimize their workforce between maintaining exploration (e.g., to forage) and exploitation (e.g., to transfer discovered food) [47]. Achieving the mentioned optimization requires an architecture that is in a highly distributed control regime. In such a society, relatively large groups of ants with few members have the ability to diffuse information fast enough within the scope of their colony.

In contrast, the hierarchical organizations depend on certain employees (e.g., managers) to make an influence and spread information. Our empirical results from the e-mails of a manufacturing company reflect that behavior. The analysis shows a few groups of employees with a large number of members are required to achieve full control. Furthermore, there is a large intersection between those groups indicating the existence of critical employees (e.g., managers) for spreading information; showing the system is in a centralized control regime as expected in the hierarchical organizations.

### 6.1 *Distributed and Centralized Control Regimes*

The aforementioned categories of driver nodes (*critical*, *intermittent*, and *redundant*) form two control regimes, namely *centralized* and *distributed* [43, 44]. A *centralized* control regime indicates *redundant* nodes are dominant in a network. In contrast, a *distributed* regime indicates *intermittent* and *critical* nodes are dominant. However, defining a universal definition for the distributed and centralized control regime is challenging because a definition needs to be relative to the size and state of a system. For instance, consider the ant colonies datasets that lack information about in what state a colony was during the data collection (e.g., foraging to find food or exploiting discovered food resources, etc.). Plus, the individual dynamics of the ants (nodal dynamics) such as their state retention strength/existence modeled by self-loops are unknown. Moreover, in the ant datasets, the observation period is on average 1800 seconds for all colonies (Appendix C).

Prior work classified the control regime only based on the fraction of redundant nodes $f_r$ [43, 44]. A large $f_r$ would indicate the centralized regime since a smaller number of nodes could be a driver node. Based on that definition, the ant colony 1-2, presented by Table 1, is in a centralized control regime since $f_r(0.8) > f_i(0.2)$. However, we argue the ant colony 1-2 is in a distributed control regime since there are 21 ($N_{MDS}$) unique groups of ants with only 2 members ($N_s$) that can fully control their colony. Plus, consider that only 15 ants ($= \lceil f_i * |V| \rceil$) are the possible minimum drivers (i.e., intermittent nodes). If we

consider some ants permanently stay inside their colony (i.e., they are not exposed to the external factors directly), 15 ants are still roughly 20% of the colony (with 72 ants) who have the capacity to reach a large percentage of control (if not full control) with groups of 3 ants ($N_c = 3$). Therefore, we propose Statement 1 in an attempt to formalize the control regime of a network.

Statement 1 The distributed regime refers to the ability to control a network with a relatively large number of unique groups of nodes that have a small number of members, that is, large $N_{MDS}$ and small $N_c$ relative to size of the network.

According to this statement, except the colony 6-1, all ant colonies are in a distributed control regime since relative to their number of ants they have a large number of MDSs with a small size ($N_c$). In the colony 6-1, a single ant is capable of controlling the entire colony since the minimum number of driver nodes is equal to one ($N_c = 1$). However, only three ants ($N_{MDS} = 3$) out of 33 have the capability to solely control the colony. In contrast, the e-mails dataset is in a centralized control regime since a large number of key employees (39% critical nodes) must be stimulated to reach controllability (Table 1).

Next, we analyzed the behavior of our heuristic approach toward identifying the control regime of a network. Table 1 provides comparisons between the heuristic's suboptimal and the optimal (MDS) solutions for the ants datasets. To increase the accuracy of heuristic, we provide the number of SMDSs discovered with size $N_s$ and $N_s + 1$, and we consider their average values in our analysis. Overall, according to Statement 1 the heuristic identifies that all colonies are in a distributed control regime with the exception of the colony 1-1. In this particular colony, the brute force approach identified 153 MDSs with optimal size $N_c = 3$, and naturally this indicates a highly distributed control regime. However, the heuristic in total identified $2 + 11 = 13$ SMDSs with size $N_s = 7$ and 8, and based on Statement 1, we would consider this to be a centralized control regime.

In contrast, based on the optimal solutions, the colony 6-1 is in a centralized regime since with $N_c = 1$ only three individuals are capable of solely controlling the network. But, the heuristic found 19 SMDSs with size $N_s = 2$ that is considered as the distributed control regime. Therefore, applying Statement 1 to the heuristic and optimal solutions must be done with cautions. For instance, by running the heuristic with different initial ordering, we can increase the number of SMDSs and hence increase the chance of finding optimal solutions. Currently, the initial ordering is a descending sort based on the cardinalities of the MCSs in the MCS list $\mathbb{C}$ (Step 4 of Algorithm 1). Instead, we could try an ascending sort or randomly positioning of the MCSs in $\mathbb{C}$.

Moreover, compared to the optimal solutions in Table 1, the heuristic's behavior regarding the number of driver node sets (the *count SMDSs* column) and the fractions of control categories (columns $f_i'$, $f_r'$, and $f_c'$) varies. For instance, in the ant colony 1-1 the optimal solutions indicate $f_i(0.59) > f_r(0.41)$ with 153 MDS of size $N_c = 3$ and without any critical node ($f_c = 0$). However, on average the heuristic shows 6% of the nodes are critical ($f_c' = 0.06$) and $f_i'(0.07) \ll f_r'(0.86)$ within total of 13 SMDSs with their size $N_s = 7$ and $N_s = 8$. In the ant colony 3-1, we see an interesting behavior, the fractions of intermittent and redundant nodes are switched between the optimal and heuristic approach solutions. The optimal solutions indicate $f_i(0.54) > f_r(0.46)$ with 7 MDSs of size $N_c = 2$, but on average the heuristic shows $f_i'(0.32) + f_c'(0.13) < f_r'(0.54)$ with $1 + 6 = 7$ SMDSs of size $N_s = 1$ and $N_s = 2$. For the rest of ant colonies, the optimal and heuristic solutions behave similarly regarding the inequalities between their corresponding fractions of control categories. Furthermore, due to the lack of information about the state of colonies during the observation periods, it is challenging to confirm whether utilizing MDS or SMDS is more effective in identifying the control regimes of the ant networks.

## 7. Conclusion

Temporal networks capture the time dimension in the network's structure and thus permit the study of systems with changing topologies. We focus on the controllability of this class of networks, that is, the study of steering the state of a network within a desired $\Delta t$ steps to a targeted state at deadline $t_f$. Controlling a network is done by stimulating key nodes called driver nodes. Finding the minimum number of driver nodes, $N_c$, requires testing $O(2^{N\Delta t})$ configurations that is computationally infeasible. Therefore, we propose a heuristic algorithm to find a set of driver nodes with suboptimal size $N_s \geqslant N_c$.

We evaluated our proposed heuristic algorithm on Erdős–Rényi and Barabási–Albert synthetic temporal networks as well as real-world temporal networks created from six ant colonies and e-mail communications in a manufacturing company. The heuristic was able to identify multiple driver node sets that can fully control the networks on $t_f$ in $\Delta t$ steps. For the ants' interactions datasets, we computed $N_c$ by brute force. The results show the heuristic requires few more driver nodes than the optimal in the ant colonies networks. Moreover, the driver nodes in all MDSs and SMDSs tend to have a larger in-out-total degree than the average degrees of these networks. Also, the results are in line with key behaviors of the complex systems modeled by the datasets. We show that the ants' networks are in a distributed control regime; that is, a large number of unique groups of ants with a few members can fully control the colony. Also, the queen ants tend to avoid becoming a driver node. In contrast, we show the e-mails network is in a centralized control regime; that is, a small number of unique groups of employees with large members can fully control the network. Those groups of employees (SMDSs) that can fully control the e-mails network have a large intersection with each other. This intersection could represent the employees who manage the company.

## 8. Acknowledgements

The authors acknowledge Mr. Javad Darivandpour, Ph.D. candidate in the Department of Computer Science at Purdue University, West Lafayette for his help and guidance on this article.

## References

**1.** Steven Strogatz. *Sync: The emerging science of spontaneous order*. Penguin, London, 2004.
**2.** Mikail Rubinov and Olaf Sporns. Complex network measures of brain connectivity: uses and interpretations. *Neuroimage*, 52(3):1059–1069, 2010.
**3.** Danuta Makowiec. The heart pacemaker by cellular automata on complex networks. In Hiroshi Umeo, Shin Morishita, Katsuhiro Nishinari, Toshihiko Komatsuzaki, and Stefania Bandini, editors, *Cellular Automata*, volume 5191, pages 291–298, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
**4.** Lucia Carlucci, Gianfranco Ciani, and Davide M Proserpio. Polycatenation, polythreading and polyknotting in coordination network chemistry. *Coordination Chemistry Reviews*, 246(1-2):247–289, 2003.
**5.** Pahola T. Benavides, Urmila Diwekar, and Heriberto Cabezas. Controllability of complex networks for sustainable system dynamics. *Journal of Complex Networks*, 3(4):566–583, 2015.
**6.** Zhengzhong Yuan, Chen Zhao, Zengru Di, Wen-Xu Wang, and Ying-Cheng Lai. Exact controllability of complex networks. *Nature Communications*, 4:2447, Sep 2013. Article.
**7.** Jose C. Nacher and Tatsuya Akutsu. Analysis of critical and redundant nodes in controlling directed and undirected complex networks using dominating sets. *Journal of Complex Networks*, 2(4):394–412, 2014.
**8.** Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97 – 125, 2012. Temporal Networks.
**9.** Raj Kumar Pan and Jari Saramäki. Path lengths, correlations, and centrality in temporal networks. *Phys. Rev. E*, 84:016105, Jul 2011.

**10.** Ingo Scholtes, Nicolas Wider, René Pfitzner, Antonios Garas, Claudio J. Tessone, and Frank Schweitzer. Causality-driven slow-down and speed-up of diffusion in non-markovian temporal networks. *Nature Communications*, 5:5024 EP –, Sep 2014. Article.

**11.** Teresa M. Przytycka, Mona Singh, and Donna K. Slonim. Toward the dynamic interactome: it's about time. *Briefings in Bioinformatics*, 11(1):15–29, 2010.

**12.** Stavros I. Dimitriadis, Nikolaos A. Laskaris, Vasso Tsirka, Michael Vourkas, Sifis Micheloyannis, and Spiros Fotopoulos. Tracking brain dynamics via time-dependent network analysis. *Journal of Neuroscience Methods*, 193(1):145 – 155, 2010.

**13.** M. Karsai, M. Kivelä, R. K. Pan, K. Kaski, J. Kertész, A.-L. Barabási, and J. Saramäki. Small but slow world: How network topology and burstiness slow down spreading. *Phys. Rev. E*, 83:025102, Feb 2011.

**14.** Ching-Tai Lin. Structural controllability. *IEEE Transactions on Automatic Control*, 19(3):201–208, 1974.

**15.** Kazuo Murota. *Matrices and Matroids for Systems Analysis*, volume 20. Springer, New York, 2009.

**16.** A. Li, S. P. Cornelius, Y.-Y. Liu, L. Wang, and A.-L. Barabási. The fundamental advantages of temporal networks. *Science*, 358(6366):1042–1046, 2017.

**17.** Márton Pósfai and Philipp Hövel. Structural controllability of temporal networks. *New Journal of Physics*, 16(12):123055, 2014.

**18.** Yan Zhang, Antonios Garas, and Ingo Scholtes. Controllability of temporal networks: An analysis using higher-order networks. *arXiv preprint arXiv:1701.06331*, 2017.

**19.** Mark Newman. *Networks: an introduction*. Oxford university press, 2010.

**20.** Ingo Scholtes. When is a network a network?: Multi-order graphical model selection in pathways and temporal networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1037–1046, New York, NY, USA, 2017. ACM.

**21.** Chen Zhao, Wen-Xu Wang, Yang-Yu Liu, and Jean-Jacques Slotine. Intrinsic dynamics induce global symmetry in network controllability. *Scientific Reports*, 5:8422 EP –, Feb 2015. Article.

**22.** Noah J. Cowan, Erick J. Chastain, Daril A. Vilhena, James S. Freudenberg, and Carl T. Bergstrom. Nodal dynamics, not degree distributions, determine the structural controllability of complex networks. *PLOS ONE*, 7(6):1–5, 06 2012.

**23.** Martin Rosvall, Alcides V. Esquivel, Andrea Lancichinetti, Jevin D. West, and Renaud Lambiotte. Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications*, 5:4630 EP –, Aug 2014. Article.

**24.** Jean-Jacques E Slotine, Weiping Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

**25.** Fabio Pasqualetti, Sandro Zampieri, and Francesco Bullo. Controllability metrics, limitations and algorithms for complex networks. *IEEE Transactions on Control of Network Systems*, 1(1):40–52, March 2014.

**26.** Tao Jia and Márton Pósfai. Connecting core percolation and controllability of complex networks. *Scientific Reports*, 4:5379, Jun 2014. Article.

**27.** Airlie Chapman, Marzieh Nabi-Abdolyousefi, and Mehran Mesbahi. Controllability and observability of network-of-networks via cartesian products. *IEEE Transactions on Automatic Control*, 59(10):2668–2679, Oct 2014.

**28.** Mohsen Zamani and Hai Lin. Structural controllability of multi-agent systems. In *2009 American Control Conference*, pages 5743–5748, June 2009.

**29.** R. Kalman. Mathematical description of linear dynamical systems. *Journal of the Society for Industrial and Applied Mathematics Series A Control*, 1(2):152–192, 1963.

**30.** Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

**31.** Sérgio Pequito, Victor M. Preciado, Albert-László Barabási, and George J. Pappas. Trade-offs between driving nodes and time-to-control in complex networks. *Scientific Reports*, 7:39978, jan 2017.

**32.** Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 1. Wiley-interscience New York, 1972.

**33.** Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-*

*Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
**34.** Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.
**35.** Chandra Chekuri and Alina Ene. *Poly-logarithmic Approximation for Maximum Node Disjoint Paths with Constant Congestion*, pages 326–341.
**36.** Chandra Chekuri and Sanjeev Khanna. Edge disjoint paths revisited. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 628–637, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.
**37.** Anand Srinivas and Eytan Modiano. Minimum energy disjoint path routing in wireless ad-hoc networks. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom '03, pages 122–133, New York, NY, USA, 2003. ACM.
**38.** Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.
**39.** Saïd Salhi. *Population-Based Heuristics*, pages 77–128. Springer International Publishing, Cham, 2017.
**40.** Benjamin Blonder and Anna Dornhaus. Time-ordered networks reveal limitations to information flow in ant colonies. *PLOS ONE*, 6(5):1–8, 05 2011.
**41.** Radosław Michalski, Sebastian Palus, and Przemysław Kazienko. Matching organizational structure and social network extracted from email communication. In Witold Abramowicz, editor, *Business Information Systems*, volume 87, pages 197–206. Springer Berlin Heidelberg, 2011.
**42.** Manufacturing emails network dataset. `http://konect.uni-koblenz.de/networks/radoslaw_email`. Accessed: 2018-06-03.
**43.** Tao Jia, Yang-Yu Liu, Endre Csóka, Márton Pósfai, Jean-Jacques Slotine, and Albert-László Barabási. Emergence of bimodality in controlling complex networks. *Nature Communications*, 4, jun 2013.
**44.** Xizhe Zhang and Qian Li. Altering control modes of complex networks based on edge removal. *Physica A: Statistical Mechanics and its Applications*, 516:185 – 193, 2019.
**45.** Xizhe Zhang, Tianyang Lv, and Yuanyuan Pu. Input graph: the hidden geometry in controlling complex networks. *Scientific Reports*, 6(1), nov 2016.
**46.** Yang-Yu Liu, Jean-Jacques Slotine, and Albert-László Barabási. Controllability of complex networks. *nature*, 473(7346):167, 2011.
**47.** Melanie Mitchell. *Complexity: A guided tour*. Oxford University Press, 2009.

## A. Maximum Flow Reduction

The problem of finding an MCS, denoted by $M_C(D,t_f,\Delta t)$ for an arbitrary set of driver nodes $D$ can be reduced to the maximum flow problem where the target is to reach control at deadline $t_f$ within $\Delta t = t_f - t_0$ steps. In other words, we need to find $|V|$ independent time-respecting paths (vertex disjoint paths) that a) start from intervention points $(d \in D,t)$ for any interval within $(t_f - \Delta t < t \leqslant t_f]$, and b) end at all nodes in deadline layer $t_f$.

To prepare the time-layered network for this reduction, we begin by connecting the source node $s$ to the nodes in all time-layers in interval $(t_0,t_f]$ and connecting all the nodes in the $t_f$ layer to the sink node $t$, and then, to ensure finding vertex disjoint paths, we establish that the maximum flow that can pass through each node must be one. As a result, we introduce an augmented time-layered network by adding regulatory nodes between all time-layers as more formally stated in the following:

1. For each node $v_i^t$ at layer $t$ add a new node $v_i'$ and add all out-links $v_i^t \to u_j^{t+1}$ to $v_i'$ such that $v_i' \to u_j^{t+1}$
2. Connect $v_i^t$ to $v_i'$ such that $v_i \to v_i'$ and remove all out-links $v_i^t \to u_j^{t+1}$
3. Repeat Steps 1-2 for all time-layers

4. Set the maximum capacity of all edges in the network to one

The above process is illustrated in Fig. A.11 (c) with the regulatory nodes marked by color green. In Fig. A.11, we run the maximum flow algorithm on the augmented time-layered network to find an MCS $M_C(D,t_f,\Delta t)$ for two driver sets $D_1 = \{v_1\}$ and $D_2 = \{v_1, v_4\}$, deadline $t_f = 3$, and $\Delta t = 3$ steps. Fig. A.11 (d-e) shows a maximum controllable space for $D_1$ and $D_2$ respectively with their independent time-respecting paths distinguished with distinct colors (that is, disjoint paths with flow one). For better visualization, the regulatory nodes are not shown in Figure A.11 (d-e).

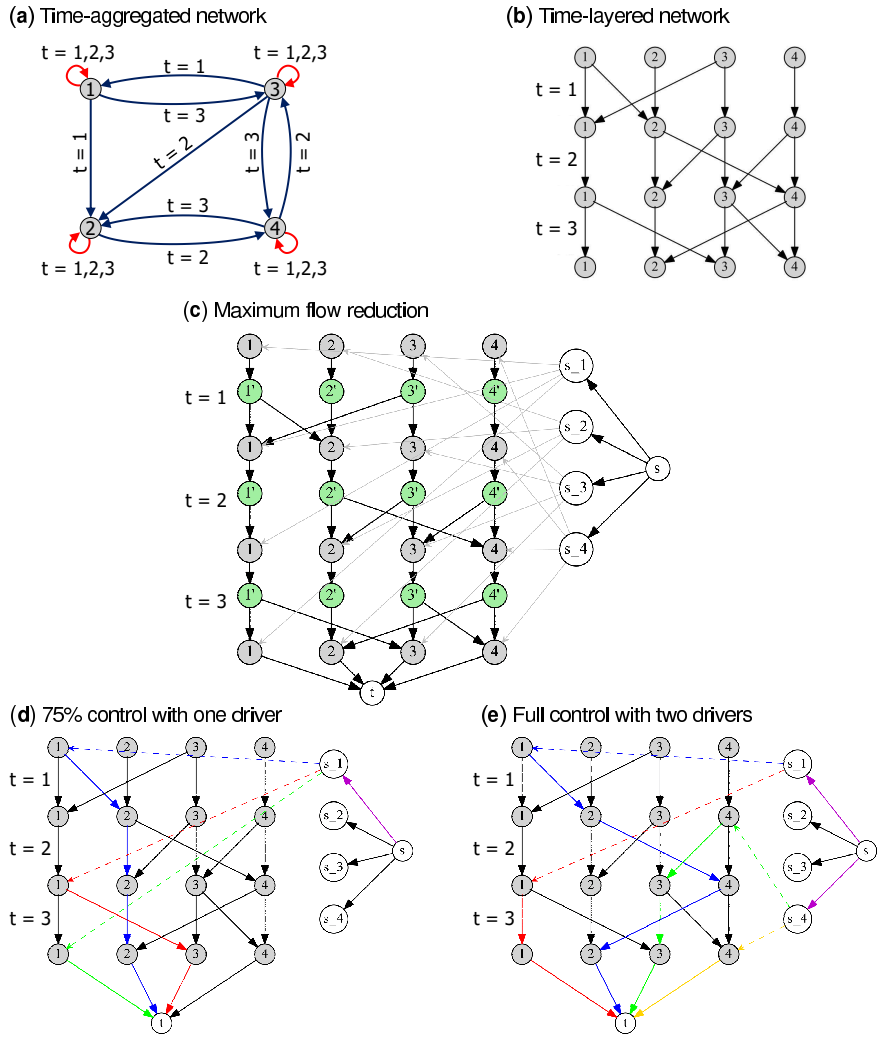


FIG. A.11. Maximum flow reduction. Identifying an MCS for a set of driver nodes on deadline $t_f = 3$ and within $\Delta t = 3$ steps. (a) Time-aggregated network with timestamped edges and observation period $1 \leqslant t \leqslant 3$. (b) Time-layered network visualizing the temporal pathways. (c) Augmented time-layered network with the capacity-regulator nodes marked by color green. (d) Driver set $D_1 = \{v_1\}$ at most can control 75% of the network ($v_4$ is not controlled). (e) Driver set $D_2 = \{v_1, v_4\}$ fully control the network.

## B. Branching Process

Fig. A.12 illustrates the execution of the proposed heuristic on all of the stems in Fig. 6. All derived SMDSs can fully control the network of Fig. 5.

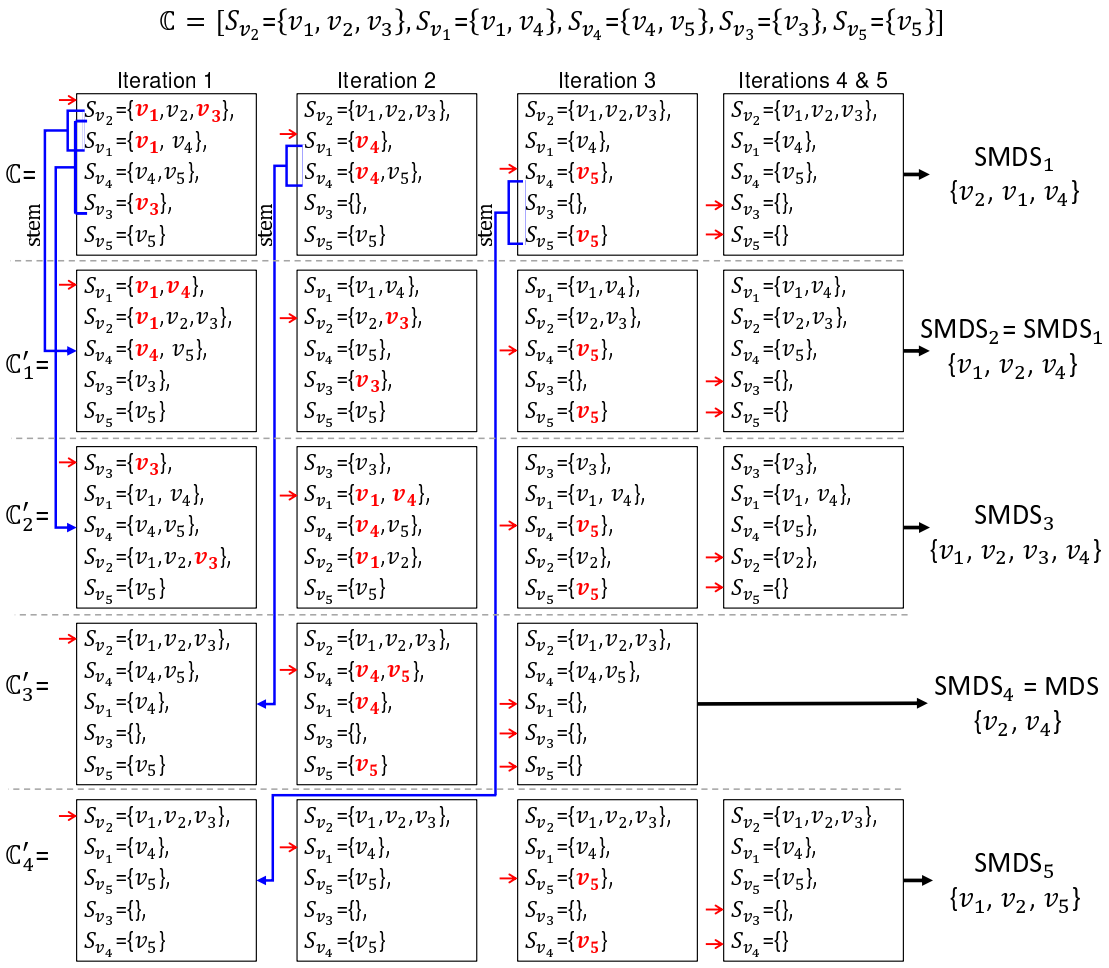


FIG. A.12. Illustration of the branching process plus the executions of the proposed heuristic for $\mathbb{C}$ and its stems $\mathbb{C}'$. The MCS list $\mathbb{C}$ is constructed based on the MCSs of nodes in the network of Fig. 5. Each row presents an execution of the heuristic that derives an SMDS. The red arrows indicate the controllable subspace set being processed by the heuristic in its iterations. Each stem (marked with blue arrows) initiates from switching the position of two controllable subspace sets that have non-empty intersection with the sets that are located after the red arrows. In total, the process creates four unique possibles SMDSs and all of them are actual SMDSs that can fully control the network in Fig. 5.

## C. Datasets Temporal Characteristics

Figure A.13 illustrates the observation periods and frequency of interactions for Ants and Emails datasets.

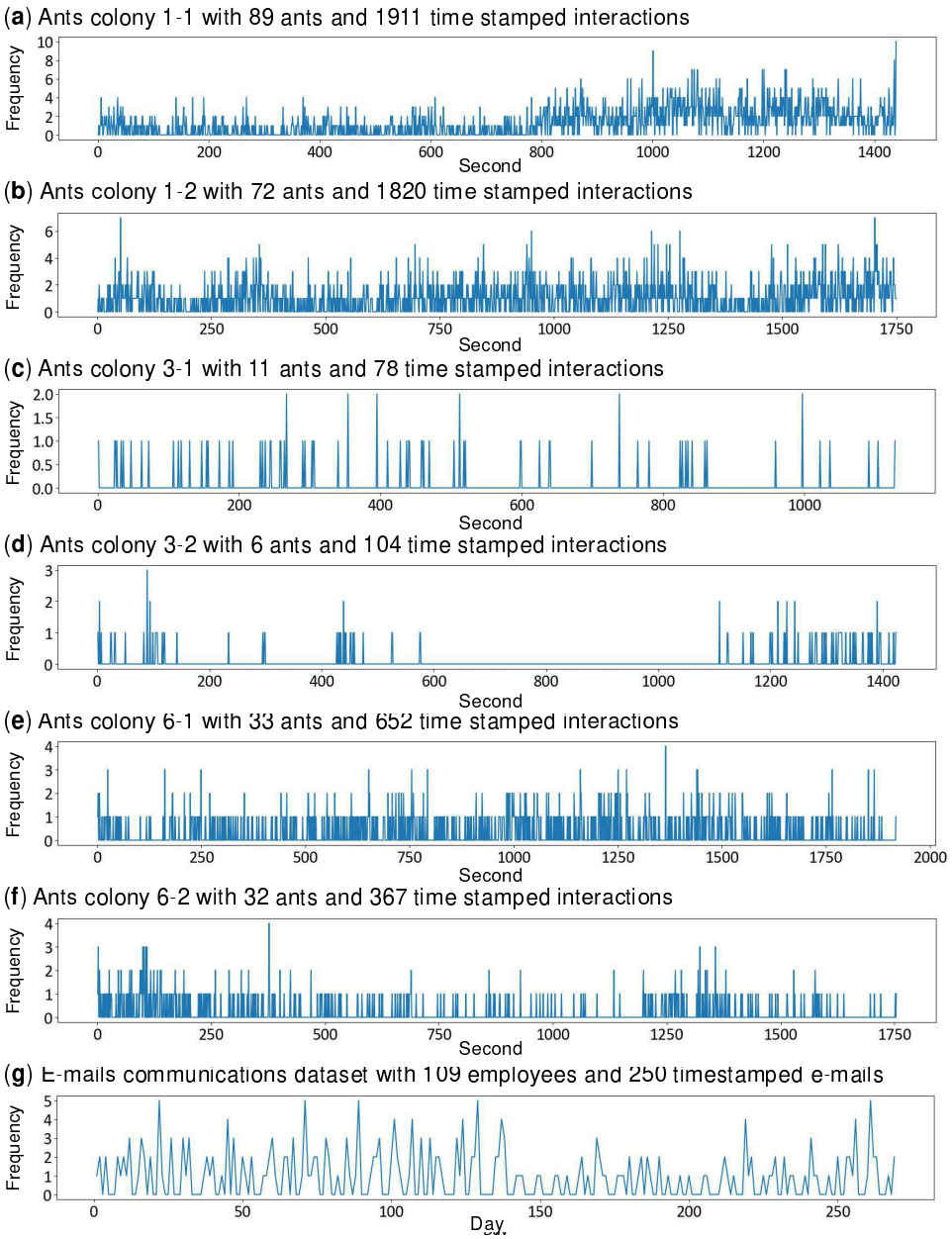


FIG. A.13. Frequency of interactions versus observation periods. (a-f) Ants' interactions. (g) Daily e-mail communications with frequency in 1h resolution between 2010-01-05 06:00 AM and 2010-09-29 11:00 AM.

Fig. A.14 illustrates the temporal characteristics of dataset including the number of time-layers (i.e., the number of timestamps) and inter-event gaps.

| (**a**) **Ants 1-1** | | (**b**) **Ants 1-2** | |
|---|---|---|---|
| Nodes: | 89 | Nodes: | 72 |
| Time-stamped links: | 1911 | Time-stamped links: | 1820 |
| Links ÷ Nodes: | 21.47 | Links ÷ Nodes: | 25.28 |
| Time-layers: | 883 | Time-layers: | 1048 |
| Observation period: | [0 s, 1438 s] | Observation period: | [0 s, 1749 s] |
| Observation length: | 1438 s | Observation length: | 1749 s |
| Avg. inter-event*dt: | 1.63 s | Avg. inter-event dt: | 1.67 s |
| Min/Max inter-event dt: | 1/15 s | Min/Max inter-event dt: | 1/12 s |
| (**c**) **Ants 3-1** | | (**d**) **Ants 3-2** | |
| Nodes: | 11 | Nodes: | 6 |
| Time-stamped links: | 78 | Time-stamped links: | 104 |
| Links ÷ Nodes: | 7.09 | Links ÷ Nodes: | 17.33 |
| Time-layers: | 72 | Time-layers: | 92 |
| Observation period: | [12 s, 1139 s] | Observation period: | [2 s, 1425 s] |
| Observation length: | 1127 s | Observation length: | 1423 s |
| Avg. inter-event dt: | 15.87 s | Avg. inter-event dt: | 15.64 s |
| Min/Max inter-event dt: | 1/97 s | Min/Max inter-event dt: | 1/533 s |
| (**e**) **Ants 6-1** | | (**f**) **Ants 6-2** | |
| Nodes: | 33 | Nodes: | 32 |
| Time-stamped links: | 652 | Time-stamped links: | 367 |
| Links ÷ Nodes: | 19.76 | Links ÷ Nodes: | 11.47 |
| Time-layers: | 537 | Time-layers: | 312 |
| Observation period: | [1 s, 1918 s] | Observation period: | [2 s, 1755 s] |
| Observation length: | 1917 s | Observation length: | 1753 s |
| Avg. inter-event dt: | 3.58 s | Avg. inter-event dt: | 5.64 s |
| Min/Max inter-event dt: | 1/34 s | Min/Max inter-event dt: | 1/58 s |

| (**g**) **E-mail Communications of a midsized manufacturing company** | |
|---|---|
| Nodes: | 109 |
| Time-stamped links: | 250 |
| Links ÷ Nodes: | 2.30 |
| Time-layers: | 224 |
| Observation period: | [1262671200 s, 1285758000 s] |
| Observation length: | 23086800 s |
| Avg. inter-event dt: | 103528.25 s |
| Min/Max inter-event dt: | 3600/500400 s |

***** inter-event: All time differences between any two-consecutive time-stamped links (involving any node).

FIG. A.14. Temporal characteristics of datasets including the number of time-layers (i.e., number of timestamps) and inter-event gaps.

## D. Randomization Procedures

We used the below randomization techniques to randomize the ants and e-mail datasets. For a comprehensive explanation of randomizing temporal networks refer to [8]. Also, Table A.3 provides a comparison between these randomization techniques.

*Randomized Edges (RE)*: This randomization can be used to study the effect of the network topology. RE changes who contacts whom and it assumes the edges govern the time of contacts rather than the vertices. This randomization changes the number of contacts and the timing of contacts for each vertex. However, RE preserves the degree distribution of the time-aggregated network. Since the timestamps of edges are not changed (contact sequence is preserved), all temporal correlations associated with edges such as the average degree fluctuations are preserved. For the algorithm to implement RE refer to [8].

*Randomly Permuted (RP)*: For this randomization, we shuffle the timestamps of temporal edges. RP retains the time-aggregated network structure and the number of contacts for each edge. However, this null model eliminates the temporal correlations such as the causal events, and it alters the degree distribution of each time-layer.

*Random Time (RT)*: In this randomization, we randomly assign a time to each temporal link. Therefore, RT removes all temporal interactions, both the local correlations such as simultaneous events, and the global correlations such as overall fluctuations in the degree distributions. However, RT does not change the structure of time-aggregated network.

*Degree Preserved Network (DPN)*: To apply this randomization, we randomly rewire the networks between each time-layer using the original degree sequence of the time-layers (the timestamps of interaction are not changed). Hence, this randomization only preserves the degree distribution in each time-layer. However, all other temporal and structural correlations are eliminated.

*Random Network (RN)*: For this randomization, we replace the network of each time-layer with an Erdős-Reńyi network with the same number of links as the original network between each time-layer. Therefore, RN removes all temporal correlations and the structure of time-aggregated network including heterogeneity of in-out-total degree distributions. Similar to DPN, we do not change the timestamps of interactions.

Table A.3. Randomization techniques. Green color indicated preservation and red color indicates elimination of network's characteristics.

| Randomization | Aggregated network structure[a] | Aggregated network degree distribution | Local degree distributions[b] | Overall rate of events[c] | Average degree fluctuations | Causal chain of events |
|---|---|---|---|---|---|---|
| RE | | | | | | |
| RP | | | | | | |
| RT | | | | | | |
| DPN | | | | | | |
| RN | | | | | | |

[a] Preserves who contacts whom.
[b] Degree distribution of each time-layer.
[c] For example, the number of simultaneous interactions in a time-layer such as patterns in the rate of daily or weekly interactions in communication networks.