How to Control Target Nodes in Dynamic Networks

Zhaoquan Gu
Guangzhou University
Guangzhou, China
zqgu@gzhu.edu.cn

Yongcai Wang
Renming University
Beijing, China
ycw@ruc.edu.cn

Jing Qiu
Guangzhou University
Guangzhou, China
qiujing@gzhu.edu.cn

Lihua Yin
Guangzhou University
Guangzhou, China
yinlh@gzhu.edu.cn

Zhihong Tian
Guangzhou University
Guangzhou, China
tianzhihong@gzhu.edu.cn

***Abstract*—Controlling a dynamic network, i.e. driving the network from any initial state to any desired state, is important in practice. Examples include influencing people's sentiments in social networks and easing the pressure in traffic networks. Much work has been done to the problem of controlling the whole network, which is complex and might not be necessary as there are situations where we are concerned with only some of the nodes (called target nodes). In this paper, we study the target controlling (TC) problem in a directed dynamic network, which is to govern the target nodes by the minimum number of controllers. We first derive two important graphical properties of target controlling, which serve as a foundation for all subsequent derivations. Then we show that the TC problem is NP-hard and any traditional method has bad performance. Furthermore, we propose an optimal algorithm for the TC problem in directed tree networks and an efficient heuristic algorithm for general dynamic networks. Finally, we conduct extensive simulations on both synthetic and real networks to evaluate our algorithms, and the results show that our algorithms have better performance than the state-of-the-art results.**

I. Introduction

Dynamic networks have attracted much attention from academic researchers as well as in industry. In dynamic networks, the network state can be influenced and updated through the underlying topology. For example, in a social network [16], [17], an individual's sentiment towards a particular topic can be influenced by his neighbors and this process can be modeled as a dynamic system. In a public traffic network [1], the load of a road can be affected by nearby roads. In order to influence the public opinion about a topic or ease the traffic pressure, we need to solve the controllability problem: how to drive the network from any initial state to any desired state (by imposing some controllers). The problem was first proposed in [13].

There are many important works in handling the controllability problem [3], [11], [13], [18], [20], and they assume a dynamic network, which consists of N nodes, can be modeled by a linear time-invariant model:

$$\overrightarrow{x(t+1)} = A \cdot \overrightarrow{x(t)} + B \cdot \overrightarrow{u(t)} \tag{1}$$

where $\overrightarrow{x(t)} \in R^N$ represents network state (value) of the N nodes at time t, $\overrightarrow{u(t)} \in R^M$ represents the input signals of M controllers, $A \in R^{N\times N}$ represents the transmission matrix that captures the underlying influence relations of the network, and $B \in R^{N\times M}$ is the controlling matrix that represents the direct influences of the controllers on the network. It is proved in [8] that the network is *controllable* if and only if the $N \times NM$ matrix $D = (B, AB, A^2B, \dots, A^{N-1}B)$ has full rank, i.e. $rank(D) = N$.

Controlling the whole network is essential in some engineered environments [5], but it is unnecessary to control the full network in many practical scenarios, partly due to the massive network size and complexity. It is more realistic to control pre-defined *target nodes* for certain task. In this paper, we study the **target controlling (TC) problem**, aiming at controlling a chosen subset of nodes by imposing minimum number of controllers. Denote the set of target nodes as $T = \{t_1, t_2, \dots, t_S\}$, we define the output matrix $C = [I(t_1), I(t_2), \dots, I(t_S)] \in R^{S\times N}$ where $I(t_i)$ denotes the t_i-th ($i \in [1, S]$) row of the identify matrix I; then:

$$\overrightarrow{y(t)} = C\overrightarrow{x(t)}$$

and $\overrightarrow{y(t)}$ represents the state of the target nodes at time t. The goal is to control $\overrightarrow{y(t)}$ by imposing time-dependent input signals $\overrightarrow{u(t)}$ via the M controllers. It is proved in [15] that the target nodes are controllable if and only if the rank of the matrix $E = (CB, CAB, CA^2B, \dots, CA^{N-1}B)$ is S.

In our pursuit, we face three main challenges. First, although controlling the whole network by the minimum number of controllers can be solved through the maximum matching method [13], the number of controllers needed to control only a subset of nodes could be much smaller, and so the method is inapplicable for target control. Second, controlling the whole network has graphical properties that make determining the minimum number of *cactuses* (which will be discussed in Section II) spanning the network possible [2], [13], but target controlling cannot be described by such good graphical structures. Third, finding the minimum number of controllers for the TC problem is very hard; a greedy algorithm was proposed in [5], but it works badly even for a simple directed tree network (as we will see in Section V-B).

In this paper, we address these issues and present efficient algorithms for solving the TC problem. The main contributions are summarized as follows:

1) We introduce two important graphical properties for the TC problem. The first property concerns controlling multiple nodes by a controller if there exist paths of different length to the controlled node (i.e., the driver node; please refer to Section III-C). The second one reveals the condition that two nodes cannot be controlled

by a single controller if their paths to the driver node have the same length.

2) Based on the graphical properties, we show that the TC problem is NP-hard, by reducing the Maximum Set Packing problem to it. Moreover, we propose a directed tree example for which all cactus-based algorithms (including the only greedy algorithm [5] for the problem) have bad performance. Specifically, the number of needed controllers could be $\Omega(N)$ times the optimal solution.
3) We propose an optimal algorithm for the TC problem in directed tree networks, and an efficient heuristic algorithm for general dynamic networks based on the graphical properties.
4) We conduct extensive simulations on both synthetic and real networks, and the results show that our algorithms have good performance.

The remainder of the paper is organized as follows. In the next section, we introduce some related works about the TC problem. The system model and problem formulation are presented in Section III. The graphical properties are described in Section IV and we show the problem hardness in Section V. We propose the optimal algorithm for directed tree networks and a heuristic algorithm for general networks in Section VI. We report the results of extensive simulation in Section VII and conclude the paper in Section VIII.

II. Related Works

In this paper, we study the target controlling (TC) problem in dynamic networks. When all nodes are target nodes, this problem is equivalent to controlling the whole network. According to the control theory, a dynamic network is controllable if we can drive the network from any initial state to any desired state with some input signals [8], [10]. However, it is difficult to obtain the influence value that one node has on another in practice. Therefore, structural controllability has better significance since we do not need the exact values.

There are many existing important works about controlling the full network [2], [3], [11], [13], [20]. A pioneering idea was proposed in [11] which transforms structural controllability into graphical properties. It introduces three important graphical structures: *stem, bud*, and *cactus*. A stem is a directed path of which the first node is a controller, while a bud consists of a cycle and a controller connected to any node in the cycle. On the basis of stem and bud, a cactus is constructed with one stem and multiple buds. As depicted in Fig. 1, c_1 is the controller, $\{c_1, v_1, v_2, v_3, v_4, v_5\}$ is a stem, $\{v_2, v_{12}, v_{13}, v_{14}, v_{15}\}$, $\{v_3, v_{16}, v_{17}, v_{18}, v_{19}\}$, $\{v_4, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$ are three buds, and all these nodes comprise a cactus.

Based on the graphical structures, it is proved that a network is structural controllable if and only if there exists a node disjoint union of cactuses that span the network [3]. In order to find such minimum number of cactuses, a breakthrough result was proposed in [13], where a polynomial time algorithm is presented by transforming the problem to maximum matching. Following that result, massive research has been conducted. For example, Jia et al. classified the nodes by their roles in the

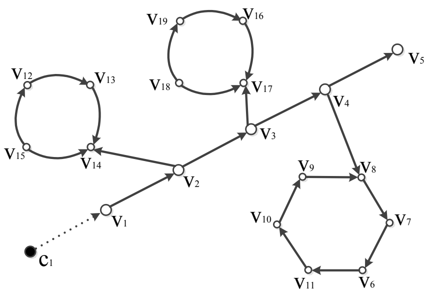

Fig. 1. An example of a cactus

control configuration [9]; a control profile was used to classify the nodes through the underlying correlations of the network in [18]; a mathematic method to control the network with respect to a given exact transmission matrix was proposed in [20]; and several constant approximation algorithms to minimize the control latency, i.e. the number of time slots to reach the final desired state, were proposed in [2].

However, relatively little has been done about the TC problem. A theoretical result by computing the rank of matrix $E = (CB, CAB, CA^2B, \dots, CA^{N-1}B)$ was proposed in [15], but it is too complex and unrealistic for finding the minimum number of controllers. Note that a novel idea of controlling the target nodes would be to find the minimum number of cactuses that contain all the target nodes, but this problem is also very hard. A greedy algorithm was presented in [5] to find these cactuses, but we will show that this cactus based method has bad performance (in Section V).

III. Preliminaries

A. System Model

Suppose a directed network $G(A)$ consists of N nodes $V = \{v_1, v_2, \dots, v_N\}$. Denote the column vector $\overrightarrow{x(t)} \in R^N$ as the states (i.e. values) of all the nodes at time t, and $A \in R^{N\times N}$ as the transmission matrix that reveals the influence values between the nodes. Specifically, if and only if there is a directed edge connecting node v_j to node v_i, the influence value $a_{ij} \in A \neq 0$. The transmission matrix is different from the network's adjacency matrix, but it is obvious that A is the transpose of the adjacency matrix if all influence values are 1.

In this paper, we focus on controlling a subset of nodes $T = \{t_1, t_2, \dots, t_S\} \subseteq V$ (i.e. target nodes) by imposing M input signals (via the controllers) $C = \{c_1, c_2, \dots, c_M\}$ on the network, and we call $G(A, B)$ the controlled network where $B \in R^{N\times M}$ is the controlling matrix that reveals the connection pattern between the controllers and the nodes in the network. Suppose time is divided into slots of equal length and the controllers can input different signals in each slot. Assuming that the network $G(A, B)$ can be described by a linear time-invariant model in a discrete-time manner, we formulate the states of the chosen target nodes as:

$$\begin{cases} \overrightarrow{x(t+1)} &= A \cdot \overrightarrow{x(t)} + B \cdot \overrightarrow{u(t)} \\ \overrightarrow{y(t)} &= C \cdot \overrightarrow{x(t)} \end{cases} \tag{2}$$

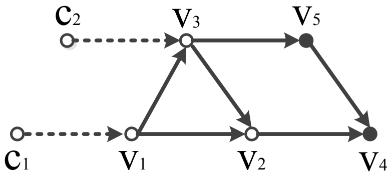

Fig. 2. An example of a controlled network

where $\overrightarrow{u(t)} \in R^M$ represents the input signals of M controllers, $C = [I(t_1), I(t_2), \ldots, I(t_S)] \in R^{S\times N}$ represents the output matrix where $I(t_i)$ is the t_i-th ($i \in [1, S]$) row of the identity matrix I, and $\overrightarrow{y(t)} \in R^S$ represents the states of the target nodes.

For example, given 5 nodes in the network $V = \{v_1, v_2, v_3, v_4, v_5\}$ as depicted in Fig. 2, the matrix A can be expressed as:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ a_{21} & 0 & a_{23} & 0 & 0 \\ a_{31} & 0 & 0 & 0 & 0 \\ 0 & a_{42} & 0 & 0 & a_{45} \\ 0 & 0 & a_{53} & 0 & 0 \end{pmatrix}$$

where $a_{21}, a_{23}, a_{31}, a_{42}, a_{45}, a_{53} \neq 0$ and they represent the influence values between two connecting nodes. Two controllers $\{c_1, c_2\}$ are imposed upon the network with two directed edges (dotted edges in the figure), and assuming that the target nodes are $T = \{v_4, v_5\}$ (solid nodes in the figure), the controlling matrix B and the output matrix C can be expressed as:

$$B = \begin{pmatrix} b_{11} & 0 \\ 0 & 0 \\ 0 & b_{32} \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad C = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

where $b_{11}, b_{32} \neq 0$ represents the controlling strength.

B. Structural Controllability

We call the matrices (A, B, C) controllable if we can drive the target nodes from any initial state $\overrightarrow{y_0}$ to a desired state $\overrightarrow{y_d}$ in finite time by injecting time-dependent input signals. It is proved in [15] that (A, B, C) is controllable if and only if the rank of the $S \times NM$ matrix

$$E = (CB, CAB, CA^2B, \ldots, CA^{N-1}B)$$

is S, i.e. $rank(E) = |T| = S$. But it is difficult to obtain the exact values of the entries of the transmission matrix in practice. Therefore, we focus on *structural controllability* where each entry of the matrices (A, B, C) is either zero or non-zero, but without the exact values. We say (A, B, C) are structural controllable if there exist (A', B', C') having the same structure that is controllable, where (A, B, C) and (A', B', C') are said to have the same structure if they have the same zero and non-zero entries. For example, the following matrices (A, B, C) and (A', B', C') have the same structure.

$$A = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$A' = \begin{pmatrix} 0 & 0 & 3 \\ 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix} \quad B' = \begin{pmatrix} 2 & 0 \\ 0 & 4 \\ 3 & 3 \end{pmatrix} \quad C' = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

Structural controllability can help overcome the incomplete knowledge of the influence values between the nodes, but an (A, B, C) that is structural controllable may not be controllable. In [11], [19], it is proved that a structurally controllable system is also controllable under almost all combinations of the elements, except for few pathological situations where the matrices A, B, C satisfy some accident constraints. Therefore, we focus on the structural controllability of dynamic networks in this paper.

C. Problem Formulation

We define the Target Controlling (**TC**) problem as follows:

Definition 3.1: For a dynamic network $G(A)$ consisting of N nodes $V = \{v_1, v_2, \ldots, v_N\}$, and a set of target nodes $T = \{t_1, t_2, \ldots, t_S\} \subseteq V$, find the minimum number of controllers M (and the controlling matrix B) such that the target nodes are structural controllable, i.e. the states of the target nodes can be driven from any initial state to any desired state.

In our approach, we find M nodes in the network and add a controller to connect to each of them. We call these nodes in the network *driver nodes*: $\{d_1, d_2, \ldots, d_M\} \subseteq V$ and the controlling matrix $B = [I'(d_1)\ I'(d_2)\ \ldots\ I'(d_M)]$ where $I'(d_i)$ is the d_i-th ($i \in [1, M]$) column of the identify matrix I times a constant $b_{d_i} \neq 0$.

IV. Graphical Properties of Target Controlling

Graphical properties of structural controllability were proposed in [11], and they can be used to identify the minimum number of controllers to control the whole network through a maximum matching method [13]. However, to the best of our knowledge, there have been no proposed graphical properties for the target controlling (TC) problem. In this section, we pioneer to introduce two such properties.

Property 4.1: For any three nodes $u, v, r \in V$, if there exist two paths of different lengths from r to u, v, i.e. $\exists P_1 : r \rightsquigarrow v$, $P_2 : r \rightsquigarrow u$, $|P_1| \neq |P_2|$, then r is a driver node and we can control nodes u, v by adding a controller at node r.

To show the correctness of the property, assume a controller is added at node r and the controlling matrix is $B = (0, \ldots, 0, b_r, 0, \ldots, 0)^T$, where the r-th element $b_r \neq 0$. Without loss of generality, suppose $u < v$, $|P_1| = n_1 < n_2 = |P_2|$, and we get the recursion equations through Eqn. (2) as:

$$\begin{aligned} \overrightarrow{x(1)} &= A \cdot \overrightarrow{x(0)} + B \cdot u(0) \\ \overrightarrow{x(2)} &= A \cdot \overrightarrow{x(1)} + B \cdot u(1) \\ &\vdots \\ \overrightarrow{x(n_2+1)} &= A \cdot \overrightarrow{x(n_2)} + B \cdot u(n_2) \end{aligned}$$

Combining these equations to get:

$$\overrightarrow{x(n_2+1)} - A^{n_2+1}\overrightarrow{x_0} = Bu(n_2) + ABu(n_2-1) + \cdots + A^{n_2}Bu(0)$$

In order to control the states of nodes u, v, output matrix $C = [I(u), I(v)] \in R^{2\times N}$ and we multiply both sides by C; then the left side is:

$$\overrightarrow{y(n_2+1)} - CA^{n_2+1}\overrightarrow{x_0} = \begin{pmatrix} x'_u \\ x'_v \end{pmatrix}$$

and the values x'_u, x'_v are constants since $C, A, \overrightarrow{x_0}$ and the desired state $\overrightarrow{y(n_2+1)}$ are fixed. Therefore, we can derive:

$$\begin{pmatrix} x'_u \\ x'_v \end{pmatrix} = (CB, CAB, \cdots, CA^{n_2}B) \begin{pmatrix} u(n_2) \\ u(n_2-1) \\ \vdots \\ u(0) \end{pmatrix} \quad (3)$$

It is easy to know that if there is a directed path of length n_1 from node r to node u, the corresponding entry $a'_{ur} \in A^{n_1} \neq 0$. Similarly, $a'_{vr} \in A^{n_2} \neq 0$. Therefore, the first element of $CA^{n_1}B$ is non-zero and the second element of $CA^{n_2}B$ is non-zero. By assigning appropriate values of $a_{ij} \in A$, it is easy to derive that $rank(CA^{n_1}B, CA^{n_2}B) = 2$, and thus $rank(CB, CAB, \cdots, CA^{n_2}B) = 2$ and the input signals $u_0, u_1, \ldots, u_{n_2}$ can be computed by Eqn. (3).

Notice that, the path between two nodes does not need to be the shortest path. We can also extend the property to multiple nodes scenario, as indicated by the following corollary.

Corollary 1: Considering a node r and a set of nodes $T' = \{t_1, t_2, \ldots, t_k\}$, if there exist paths of different lengths from r to node $t_i \in T'$, i.e. $\exists P_i : r \rightsquigarrow t_i$ such that $\forall 1 \leq i, j \leq k, i \neq j$, $|P_i| \neq |P_j|$, we can control the set of nodes T' by adding a controller at node r.

The corollary can be proved similarly.

Property 4.2: For any three nodes $u, v, r \in V$, if there is only one path P_1 from r to u, one path P_2 from r to v, and P_1, P_2 have the same length, we cannot control nodes u, v by adding a controller at node r.

Property 4.2 can be checked in the same fashion as Property 4.1, since the rank of the matrix $(CB, CAB, \cdots, CA^N B)$ is 1 when two paths have the same length. Thus, adding a controller r is inadequate. For example, the paths from node v_1 to the target nodes $\{v_7, v_8\}$ have the same length 3 as shown in Fig. 3(b), and thus they cannot be controlled by adding c_1 at v_1. We call the network a directed tree if there exists only one directed path from the root to any node (and thus there exists no cycle). However, in Fig. 3(a), there exist paths $v_1 \to v_2 \to v_4 \to v_5 \to v_7$ and $v_1 \to v_3 \to v_6 \to v_5 \to v_7$ of length 4, and the path length from v_1 to v_8 is 3, thus they can be controlled by c_1 by Property 4.1.

The two properties are important in designing efficient algorithms for the TC problem. The first property reveals the condition that multiple nodes can be governed by a controller, while the second one shows the situation when two nodes cannot be controlled by a single controller.

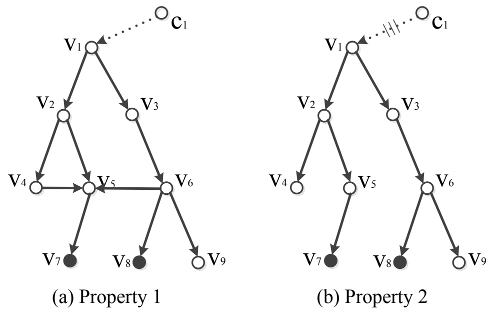

Fig. 3. Illustrations of Properties 1 and 2

V. Hardness of Target Controlling Problem

Based on the proposed graphical properties, we show that the target controlling (TC) problem is NP-hard by reducing the Maximum Set Packing problem to it. Furthermore, we propose a simple example using a directed tree that any cactus based algorithm (including the only algorithm for the problem [5]) has very bad performance.

A. NP-Hardness

We show that the TC problem is NP-hard as the Maximum Set Packing problem, which is NP-hard [6], is reducible to it in polynomial time. The Maximum Set Packing problem is: given a set U and and a family S of subsets of U, find the maximum cardinality of subsets such that they are pairwise disjoint.

Theorem 1: The target controlling (TC) problem is NP-hard.

Proof: For any instance of the Maximum Set Packing problem: $U = \{u_1, u_2, \ldots, u_n\}$ and a family $S = \{s_1, s_2, \ldots, s_m\}$ of subsets of U, i.e. $s_i \subseteq U, \forall 1 \leq i \leq m$. First order all the elements in $U = \{u'_1, u'_2, \ldots, u'_n\}$ such that $u'_1 < u'_2 < \ldots < u'_n$ and construct a new set $U' = \{1, 2, \ldots, n\}$. Then we rewrite each set $s_i \in S$ as $s_i = \{u'_{i_1}, u'_{i_2}, \ldots, u'_{i_k}\}$ and construct the corresponding set $s'_i = \{i_1, i_2, \ldots, i_k\}$. After the construction, all elements in each set s'_i are positive and they have the range $[1, n]$. Now we construct a directed tree as follows:

1. Construct a root node r and mark it as a target node;
2. For each set $s'_i = \{i_1, i_2, \ldots, i_k\}$, $1 \leq i \leq m$, construct a directed path starting from the root r, and it consists of $d_i = \max_{1 \leq j \leq k} i_j$ nodes $\{v(i,1), v(i,2), \ldots v(i, d_i)\}$ such that the path length from r to $d(i,j)$ is j. Mark the nodes $v(i, i_j)\ \forall 1 \leq j \leq k$ as target nodes.

After the two steps, we construct a directed tree with m directed paths from the root node r. For example, we construct 9 nodes in the first path and mark nodes $\{v(1,1), v(1,9)\}$ as target nodes as shown in Fig. 4 if $s'_1 = \{1, 9\}$, and 3 nodes in the second path and mark nodes $\{v(2,2), v(2,3)\}$ if $s'_2 = \{2, 3\}$. Then, we show that an optimal solution of the maximum set packing instance corresponds to an optimal solution of the TC problem.

Suppose there is an optimal solution of the maximum set packing problem $S^* = \{s_{i_1}, s_{i_2}, \ldots, s_{i_k}\}$ of cardinality k, such that $\forall 1 \leq j, l \leq k, j \neq l$, $s_{i_j} \bigcap s_{i_l} = \emptyset$. We construct the corresponding solution of the TC problem as follows: we place

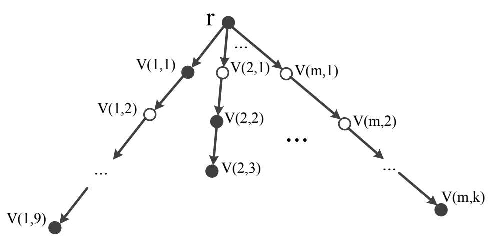

Fig. 4. An example of the directed tree construction

a controller at the root node r and add a controller at node $v(j,1)$, $\forall j \in Z_m^+ \setminus \{i_1, i_2, \ldots, i_k\}$ where $Z_m^+ = \{1, 2, \ldots, m\}$. Thus, the number of added controllers is $1 + m - k$. First, we can check that the solution is feasible. According to Property 4.1, the controller added at the root node r can control the root node and the target nodes of the i_j-th path, where $1 \leq j \leq k$. This is because each target node in the i_j-th path has distinct distance from the root node after the tree construction as any two sets of S^* are disjoint. Then each controller at node $v(j,1)$, $\forall j \in Z_m^+ \setminus \{i_1, i_2, \ldots, i_k\}$, can control the target nodes in the j-th path. Next, we show that the solution of the TC problem is optimal.

Suppose there exists another solution with less than $m-k+1$ controllers. First, we must add a controller at the root node r since it is a target node itself. Then, there are less than $m-k$ controllers added at the m paths. Pick the paths that have no controller (except the root) as P' and the number of such paths is larger than k. Since the controller at the p_i-th path cannot control the target nodes of the p_j-th path, $\forall i \neq j$, the root has to control all the target nodes of the paths in P', which implies that the corresponding sets of the set packing instance are pairwise disjoint according to Property 4.2 and the construction steps. Hence, there exists a solution of cardinality larger than k for the set packing problem, which is a contradiction. Therefore, the solution of the TC problem is also optimal.

Similarly, if there exists an optimal solution of the TC problem, we can pick the paths that have no controllers as P'; the corresponding sets comprise the maximum set packing, which is also optimal. Therefore, the Maximum Set Packing problem is reducible to the TC problem and hence the TC problem is NP-hard. ■

B. Inefficiency of Cactus-Based Algorithm

As discussed in Section II, a novel idea of tackling the TC problem is to find the minimum number of cactuses that contain all the target nodes. A cactus-based greedy algorithm was proposed in [5] (the only existing algorithm), but we show below that all cactus-based algorithms have bad performance.

Theorem 2: Any cactus-based algorithm for the target controlling (TC) problem could have an approximation ratio as bad as $\Omega(N)$ when compared to the optimal solution.

Proof: We construct a directed tree as in Fig. 5, where v_1 is the root node and the set of target nodes is $T = \{v_1, v_{\frac{N}{2}+1}, v_{\frac{N}{2}+2}, \ldots, v_N\}$. Obviously, any cactus-based algorithm can only find N node disjoint cactuses to cover all the target nodes: $\{v_1, v_{\frac{N}{2}+1}\}, \{v_2, v_{\frac{N}{2}+2}\}, \ldots, \{v_{\frac{N}{2}}, v_N\}$. Thus

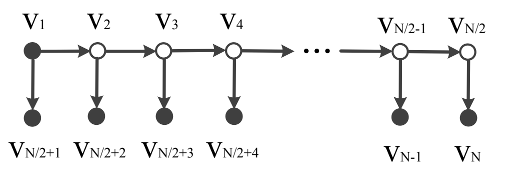

Fig. 5. An example of the tree-like network in Theorem 2

any cactus-based algorithm needs at least $N/2$ controllers to control the cactuses.

However, according to Property 4.1 and Corollary 1, we can only add one controller at node v_1 since the length of the paths from v_1 to the set of target nodes is $\{0, 1, 2, \ldots, \frac{N}{2}\}$. Thus, any cactus-based algorithm has a bad approximation ratio of $\Omega(N)$ when compared to the optimal solution. ■

VI. Target Controlling Algorithms

In this section, we propose efficient algorithms for the target controlling (TC) problem. As shown in Section V-B, any cactus-based algorithms cannot guarantee good performance even for a simple directed tree. We therefore first propose an optimal algorithm for the directed tree networks which outputs the minimum number of controllers. Then, we propose an efficient heuristic algorithm for any general dynamic network.

A. Optimal Algorithm for Directed Tree Networks

The cactus-based algorithms work badly for the directed tree network, because one cactus contains only one path and multiple cycles [2], [11], but there exists no cycle in the directed tree, so the number of node disjoint cactuses is as large as the number of directed paths. In our work, we introduce two important properties in Section IV that are totally different from the structural controllability of controlling the whole network. Therefore, we design an optimal algorithm for the directed tree, Alg. 1, based on the properties. We assume the input of the algorithm are a directed tree consisting of N nodes $V = \{v_1, v_2, \ldots, v_N\}$, plus a set of target nodes $T = \{t_1, t_2, \ldots, t_S\} \subseteq V$.

We denote the root of the tree as r. Then, each node u computes the distance from root r to u as d_u, i.e. the path length from r to u. In addition, each node maintains two variables: L_u records the distances of controlled target nodes and $AddController$ represents whether we should add a controller at node u. After the initialization, each node invokes the *controller searching* step, which works as follows. Node u first decides whether it is a leaf node by checking the children set N_u. If u is a leaf, it adds a controller if $u \in T$. Otherwise, node u waits until all children have finished the controller searching step. Then it computes the set S' of maximum cardinality such that any two elements do not intersect. After that, node u adds a controller at itself and removes some controllers that have been chosen, as in Lines 10-11. Finally, node u updates the list L_u if it is a target node.

As illustrated in Fig. 6(a), there are 15 nodes in the tree and 8 target nodes that are marked black (solid nodes). Fig. 6(b) shows the process that five controllers are added (the dotted arrows) and these nodes update the list respectively. Fig. 6(c)

Algorithm 1 Optimal Algorithm for Directed Tree Networks

```
1: Denote the root of the tree as $r \in V$;
2: for each node $u \in V$ do
3:     Denote the distance from $r$ to $u$ as $d_u \geq 0$;
4:     $AddController_u = false$, $L_u = \emptyset$;
5:     Invoke Controller Searching on node $u$;
6: end for
Controller Searching on $u$
1: Denote node $u$'s children as $N_u = \{u_1, u_2, \ldots, u_k\}$;
2: if $N_u = \emptyset$ then
3:     if Node $u$ is a target node, i.e. $u \in T$ then
4:         $AddController_u = true$, $L_u = \{d_u\}$;
5:     end if
6: else
7:     Wait until all children have finished the controller
       searching step;
8:     Denote set $S = \{L_{u_1}, L_{u_2}, \ldots, L_{u_k}\}$;
9:     Find set $S' \subseteq S$ of maximum cardinality such that any
       two elements of $S'$ do not intersect;
10:    $AddController_u = true$, $L_u = L_u \bigcup_{L_{u_i} \in S'} L_{u_i}$;
11:    For each $L_{u_i} \in S'$, $AddController_{u_i} = false$;
12:    if Node $u$ is a target node, i.e. $u \in T$ then
13:        $L_u = L_u \bigcup \{d_u\}$;
14:    end if
15: end if
```

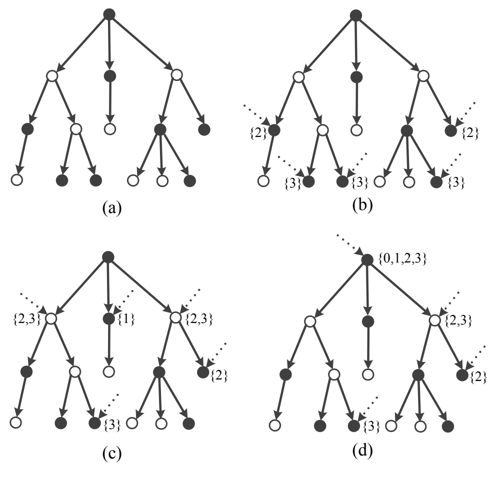

Fig. 6. An example of Alg. 1

shows that the nodes with distance 1 from the root finish the controller searching step and Fig. 6(d) depicts the final output of the algorithm: only four controllers are needed to control all the target nodes. We show the correctness of Alg. 1 rigorously.

Theorem 3: Alg. 1 computes the minimum number of controllers to govern all target nodes in a directed tree network.

Proof: We prove the theorem through induction. Denote the maximum distance from the root node r as $\Delta = \max_{u \in V}\{d_u\}$, and we show that for any $0 \leq k \leq \Delta$, the target nodes with distance no less than k can be controlled optimally by Alg. 1.

- If $k = \Delta$, we only consider the leaf nodes that are also target nodes. By Line 4 of the controller searching step, we add a controller at each target node and thus it is optimal;
- Suppose when $k = k^* > 0$, all target nodes $u \in V$ that have distance $d_u \geq k^*$ can be controlled by the minimum number of controllers. When $k = k^* - 1$, denote the set of nodes $u \in V$ with distance $d_u = k^* - 1$ as $D_{k^*-1} = \{u_1, u_2, \ldots, u_s\}$. For any node $u_i \in D_{k^*-1}$, we first find the set of maximum cardinality such that the children's lists do not intersect as in Lines 8-9, and then we remove these corresponding controllers and add one controller at u_i, which is obviously optimal from the proof of Theorem 1. Moreover, the list records the distance updates when the node itself is a target, which makes the algorithm correct. Therefore, all nodes with distance no larger than k from the root node r can be controlled optimally.

Combining the two aspects, Alg. 1 can compute the optimal solution for the TC problem. ■

Notice that, we use the brute-force method to find the optimal solution (maximum subset of maximum cardinality) in Line 9 since finding the maximum sets such that they are pairwise disjoint is NP-hard. If the maximum degree of each node can be bounded by $O(\log N)$, the method runs in polynomial time, which is acceptable. Considering more general situations, we can use the approximation algorithm in [7] to solve the maximum set packing problem, which yields $O(\sqrt{|U|})$ approximation ratio where U is the universe set. Then, we can derive the following lemma:

Lemma 6.1: Alg. 1 can achieve $O(\sqrt{h})$ approximation ratio if the algorithm is bounded in polynomial time, where h is the height of the tree.

Proof: Considering any node $v \in V$, denote the height from v to the root r as $d_v \leq h$. Suppose the children nodes of v are $C_v = \{v_1, v_2, \ldots, v_k\}$, and the corresponding variables are $L_{v_1}, L_{v_2}, \ldots, L_{v_k}$ in Line 8. Obviously, $L_{v_i} \subseteq Z_h \setminus Z_{d_v}$, and when we adopt the approximation algorithm in [7], the generated maximum set has cardinality $OPT_v / O(\sqrt{h - d_v})$, where OPT_v denotes the optimal solution of maximum set packing. Therefore, $O(\sqrt{h - d_v}) \times OPT'_v \leq O(\sqrt{h} \times OPT'_v)$ controllers are can be used to control all target nodes that rooted at node v, where OPT'_v denotes the optimal solution of controlling these target nodes. Deducing the process from height $h_v = h$ to $h_v = 0$, it is easy to check that the approximation ratio of the algorithm is only $O(\sqrt{h})$ and the algorithm runs in polynomial time. ■

Remark 6.1: Finding the maximum sets such that they are pairwise disjoint in Line 9 is NP-hard. However, if the maximum degree of each node is bounded by $O(\log N)$, we can use a brute-force method to find the optimal solution, which produces a polynomial algorithm. Moreover, we can adopt the approximation algorithm [7] if the maximum degree is large; we omit the detailed analysis of the approximation ratio.

B. Heuristic Algorithm for General Networks

Alg. 1 can find the minimum number of controllers to govern the target nodes in a directed tree network, but it is inapplicable to general networks because there may exist multiple paths between two nodes. Therefore, we put forward an efficient heuristic algorithm for any general network. Supposing the network consists of nodes $V = \{v_1, v_2, \dots, v_N\}$ and the set of target nodes is $T = \{t_1, t_2, \dots, t_S\}$, we describe the method in Alg. 2.

Algorithm 2 Heuristic Algorithm for General Networks

```
1:  Reverse the orientation of each edge in the network;
2:  for each target node $t_i \in T$ do
3:      Use depth-first-searching (DFS) method to compute the
        minimum distance from $t_i$ to node $v_j \in V$ as $d$;
4:      For any node $v_j \in V$, let $L_{v_j} = \{(t_i, d)\} \bigcup L_{v_j}$;
5:  end for
6:  while $T \neq \emptyset$ do
7:      for each node $v_j \in V$ do
8:          $D_{v_j} = \{d \mid \exists t_i \in T, s.t.\ (t_i, d) \in L_{v_j}\}$;
9:      end for
10:     Find node $u \in V$ such that $|D_u|$ is maximum;
11:     Add a controller at node $u$;
12:     Denote $D_u = \{d_1, d_2, \dots, d_l\}$;
13:     for each $d_k \in D_u$ do
14:         Find a target node $t_i \in T$ such that $(t_i, d_k) \in L_u$;
15:         For each node $v_j \in V$, delete $(t_i, d)$ from set $L_{v_j}$;
16:         $T = T \setminus \{t_i\}$;
17:     end for
18:     $D_u = \emptyset$;
19: end while
```

The algorithm consists of three steps. The first step is to compute the distances from node $v_j \in V$ to all the target nodes $t_i \in T$, as in Lines 1-5. For simplicity, we first reverse the orientation of each edge, and then we only need to compute the distance from any target node t_i to node v_j via depth-first search. Then, v_j keeps a list L_{v_j} that contains all the pairs $(t_i, d), \forall t_i \in T$ (d is the minimum distance from t_i to v_j).

The second step is to find an appropriate node as a driver node and add a controller at the node, as in Lines 6–11. First, each node v_j computes the set D_{v_j} which only records the distance values. Then, we find node u such that D_u is of maximum cardinality among all the nodes. Finally, we choose u as a driver node and add a controller at it.

The third step is to update the distance values of each node $v_j \in V$ as in Lines 12–18. Denote the distance values of node u as $D_u = \{d_1, d_2, \dots, d_l\}$ and we find a target node $t_i \in T$ such that $(t_i, d_k) \in L_u$ for any distance $d_k \in D_u$. Then, we delete the target node t_i and the computed distances from any node $v_j \in V$ to t_i. In addition, reset $D_u = \emptyset$. We repeat the final two steps until the set of target nodes T becomes empty.

As shown in Fig. 7(a), the network there consists of 7 nodes, and 5 nodes are marked as target nodes. We first reverse the orientation of each directed edge and compute the list of distance values for each node as in Fig. 7(b). Then,

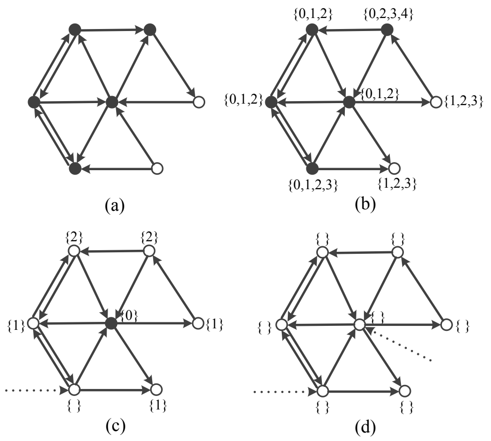

Fig. 7. An example of Alg. 2

as illustrated in Fig. 7(c), we choose the node that has the maximum cardinality of distance values as a driver node and we add a controller at it (the dotted arrow), and then we update the distance value list of each node. Since there is one target node left, Fig. 7(d) shows the second iteration of adding another controller. We derive the time complexity of the algorithm as follows.

Theorem 4: Alg. 2 can generate a set of controllers to govern the target nodes in $O(|T|(|V|^2 + |E|))$ time, where V, E are the set of nodes and edges of the network respectively, and T is the set of target nodes.

Proof: The first step to compute the distance values from any node $v_j \in V$ to all target nodes $t_i \in T$ can be completed in $|T| \cdot (|V| + |E|)$ time, since DFS from any target node needs $O(|V| + |E|)$ time. The second step of each iteration can finish in $O(|V|)$ time, by constructing the distance list D and finding the maximum cardinality. After finding the driver node u in each iteration, $|D_u| \cdot O(|V|) = O(|V|^2)$ is needed to update the list of each node. As there are at most $|T|$ iterations, the time complexity is $O(|T|(|V| + |E|)) + O(|T|(|V| + |V|^2)) = O(|T|(|V|^2 + |E|))$. ■

Remark 6.2: Alg. 2 is a heuristic algorithm to identify the number of controllers, which runs in a polynomial time. The problem of designing efficient approximation algorithms with guaranteed performance is still open.

VII. Simulation Results

We ran simulations to evaluate our proposed algorithms and compare them with the state-of-the-art result. Since Alg. 1 is an optimal solution for the directed tree networks, and Alg. 2 is applicable to all dynamic networks, we conduct simulations on both directed tree networks and general networks. To the best of our knowledge, there exists only one algorithm [5], a greedy one, for the target controlling problem and therefore we compare our algorithms with it.

In our simulations, we generate random directed tree networks based on two parameters: tree depth d is the largest distance from the root to any node, and Δ is the maximum (out) degree of each node. Once we fix a given depth d, a directed tree can be generated, where each non-leaf node has

TABLE I
THE NUMBER OF CONTROLLERS GENERATED BY DIFFERENT ALGORITHMS

Tree Depth d \ δ_T		0.1	0.3	0.5	0.7	0.9
3	Greedy Alg. [5]	2	5	10	11	16
	Alg. 2	2	5	10	11	16
	Alg. 1	2	5	10	11	16
5	Greedy Alg.	17	56	82	124	157
	Alg. 2	16	54	79	117	151
	Alg. 1	16	54	78	117	151
7	Greedy Alg.	157	455	727	1003	1284
	Alg. 2	151	435	705	988	1271
	Alg. 1	149	429	701	978	1265
8	Greedy Alg.	641	1819	2978	4075	5186
	Alg. 2	611	1752	2930	4035	5149
	Alg. 1	602	1742	2894	4013	5135

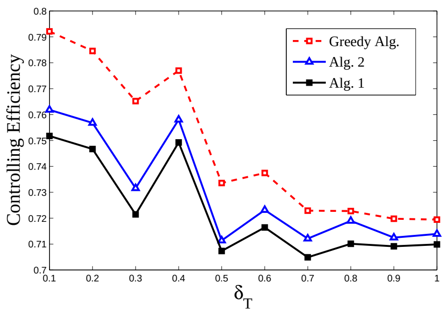

Fig. 8. Controlling efficiency comparison when $d = 7$, $\delta_T \in [0.1, 1]$

$k \in [1, \Delta]$ random children. We define $\delta_T = \frac{|T|}{|N|}$ as the ratio of the target nodes T over all nodes V in the network, and the controlling efficiency as $\frac{M}{|T|}$ where M is the number of generated controllers.

As shown in Table I, we conduct extensive simulations such that the depth of the directed tree increases from 3 to 8 (due to the page limit, we only list a subset of the results), and the target ratio δ_T increases from 0.1 to 1, under the setting $\Delta = 5$. We run each simulation 1000 times and the results are the means of them. When d increases from 3 to 8, the numbers of nodes in the networks are about $23, 75, 233, 742, 1982, 8007$ respectively. From the table, when we fix d (such as $d = 7$), the number of controllers increases as the target ratio increases for all three algorithms; this is because when there are more target nodes, we have to add more controllers, which is reasonable. We also show the controlling efficiency of the three algorithms when $d = 7$ in Fig. 8, and the results show that the optimal algorithm for the directed tree (Alg. 1) works the best, while the greedy algorithm [5] is the worst because the number of cactuses to cover all the target nodes is large. Similarly, for $\delta_T = 0.1$, we compare the controlling efficiencies when the tree depth d increases from 3 to 8, and the results also show that Alg. 1 works the best.

In order to evaluate our proposed algorithm for general networks, we use both synthetic networks (Erdös-Rényi (ER) model [4] is an example) and real networks.

In the ER model, each directed edge from u to v exists with

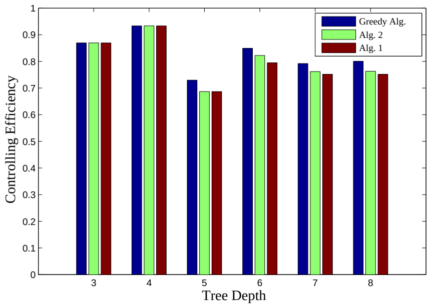

Fig. 9. Controlling efficiency comparison when $\delta_T = 0.1$, $d \in [3, 8]$

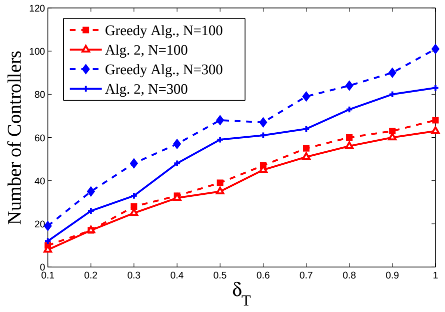

Fig. 10. Comparison of controllers in the generated random network: $p = 0.005$, $N = 100, 300$ respectively, and δ_T increases from 0.1 to 1

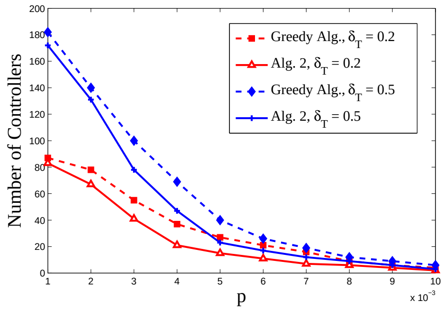

Fig. 11. Comparison of controllers in the generated random network: $N = 500$, $\delta_T = 0.2, 0.5$ respectively, and p increases from 0.001 to 0.01

probability p, and thus there are $n(n-1)p$ expected edges, where N is the number of nodes in the network. As illustrated in Fig. 10, we fix $p = 0.005$ and δ_T increases from 0.1 to 1. From the figure, when N is larger ($N = 300$), more controllers are needed; compared to the greedy algorithm, our proposed algorithm has better performance for both the $N = 100$ and $N = 300$ scenarios. When we fix $N = 500$ and p increases from 0.001 to 0.01, Fig. 11 shows the comparison of the algorithms for different target ratios $\delta_T = 0.2, 0.5$. From the figure, fewer controllers are needed when p increases, which is because larger p means more edges are added and it is easier to control the target nodes. Moreover, the heuristic algorithm performs better for both situations.

We choose two real networks to evaluate the heuristic algorithm (Alg. 2) and the greedy algorithm, of which the Facebook-like network [17] is an online social network containing 899 nodes and 7089 directed edges; and the UCIonline network [16] is an online message network of students at UC, Irvine, and it contains 1899 nodes and 20296 edges. As

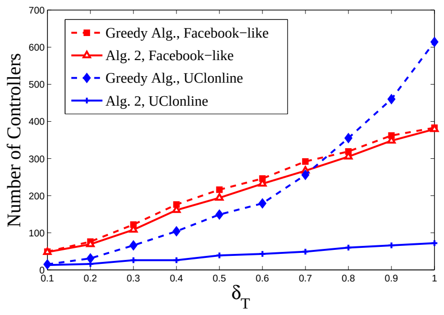

Fig. 12. Comparison of controllers in the Facebook-like network [17] and the UClonline network [16]

depicted in Fig. 12, when the target ratio δ_T increases from 0.1 to 1, both the Facebook-like network and the UCIonline network reveal that our heuristic algorithm works better than the greedy algorithm; in particular, the UCIonline network shows that our algorithm uses only $\frac{1}{6} \sim \frac{1}{5}$ nodes to control the target nodes.

Through the simulation results, clearly, our proposed optimal algorithm for the directed tree networks can find the minimum number of controllers to govern the target nodes, while the heuristic algorithm also has good performance as compared to the greedy algorithm. For general networks, whether it is synthetic networks or real networks, the heuristic algorithm has better performance than the greedy algorithm.

VIII. Conclusion

Controlling dynamic networks has many important applications, and target controlling in particular can be applied in a broad scale. In this paper, we study the target controlling (TC) problem and derive two important graphical properties for it. With these graphical properties, we prove that the TC problem is NP-hard, and propose a simple directed tree example to show the inefficiency of cactus-based algorithms. Furthermore, we propose the first optimal algorithm for directed tree networks and then an efficient heuristic algorithm for general networks. Through extensive simulations on both synthetic and real networks, our algorithms are shown to have better performance when compared to the state-of-the-art result.

In the future, we will focus on designing efficient distributed algorithms for the TC problem and designing exact controlling patterns when the transmission matrix contains exact values.

References

[1] V. Colizza, R. P. Satorras, and A. Vespignani. Reaction-Diffusion Process and Metapopulation Models in Heterogeneous Networks. *Nat. Phys.*, 3, 276-282, 2007.

[2] W. Dai, Z. Gu, X. Lin, Q.-S. Hua, and F. C.M. Lau. Minimum Control Latency of Dynamic Networks. In *INFOCOM*, 2015.

[3] J. M. Dion, C. Commault, and J. V. D. Woude. Generic Properties and Control of Linear Structured Systems: A Survey. *Automation*, 39, 1125-1144, 2003.

[4] P. Erdos, and A. Renyi. On the Evolution of Random Graphs. *Mathematical Institute of the Hungarian Academy of Sciences*, 5, 17-61, 1960.

[5] J. Gao, Y.-Y. Liu, R. M. D'Souza, and A.-L. Barabasi. Target Control of Complex Networks. In *Nature Communications*, 2014.

[6] M. R. Garey, and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Company, New York, USA, 1990.

[7] M. M. Halldorsson, J. Kratochcil, and J. A. Telle. Independent Sets with Domination Constraints. In *ICALP*, 1998.

[8] J. P. Hespanha. Linear Systems Theory. Princeton, New Jersey, 2009.

[9] T. Jia, Y.-Y. Liu, E. Csoka, M. Posfai, J.-J. Slotine, and A.-L. Barabasi. Emergence of Bimodality in Controlling Complex Networks. *Nature Communications*, 4, 1-6, 2013.

[10] R. E. Kalman. Mathmatical Description of Linear Dynamical Systems. *J. Soc. Indus. Appl. Math. Ser.*, A(1), 152-192, 1963.

[11] C.-T. Lin. Structural Controllability. *IEEE Trans. on Automatic Contr.*, 19(3), 201-208, 1974.

[12] R. Liu. The Dracula Dynamic Traffic Network Microsimulation Model. *Simulation Approaches in Transportation Analysis: Recent Advanced and Challenges*, 23-56, 2005.

[13] Y.-Y. Liu, J.-J. Slotine, and A.-L. barabasi. Controllability of Complex Networks. *Nature*, 473(7346): 167-173, 2011.

[14] A. Lombardi, and M. Hornquist. Controllability Analysis of Networks. In *Phys. Rev. E*, 2007.

[15] K. Murota, S. Poljak. Note on a graph-theoretic criterion for structural output controllability. In *IEEE Transaction on Automat. Contr.*, 35, 939-942, 1990.

[16] T. Opsahl, and P. Panzarasa. Clustering in Weighted Networks. *Soc. Netw.*, 31, 155-163, 2009.

[17] T. Opsahl. Triadic Closure in Two-Mode Networks: Redefining the Global and Local Clustering Coefficients. *Soc. Netw.* 35, 159-167, 2013.

[18] J. Ruths, and D. Ruths. Control Profiles of Complex Networks. *Science*, 343(6177), 1373-1376, 2014.

[19] R. W. Shields, and J. B. Pearson. Structural Controllability of Multi-input Linear Systems. *IEEE Trans. Automat Contr.*, 21, 203-212, 1976.

[20] Z. Yuan, C. Zhao, Z. Di, W.-X. Wang, and Y.-C. Lai. Exact Controllability of Complex Networks. *Nature Communications* 4, 2447, 2013.