Reconstruction of Structural Controllability over Erdős-Rényi Graphs via Power Dominating Sets

[Extended Abstract]

Bader Alwasel
School of Mathematics and Information Security
Royal Holloway, University of London
Egham TW20 0EX, United Kingdom
Bader.Alwasel.2012@live.rhul.ac.uk

Stephen D. Wolthusen*
School of Mathematics and Information Security
Royal Holloway, University of London
Egham TW20 0EX, United Kingdom

Norwegian Information Security Laboratory
Faculty of Computer Science
Gjøvik University College
Gjøvik, Norway
stephen.wolthusen@rhul.ac.uk

ABSTRACT

Controllability, or informally the ability to force a system into a desired state in a finite time or number of steps, is a fundamental problem studied extensively in control systems theory with *structural controllability* recently gaining renewed interest. In distributed control systems, possible control relations are limited by the underlying network (graph) transmitting the control signals from a single controller or set of controllers. Attackers may seek to disrupt these relations or compromise intermediate nodes, thereby gaining partial or total control.

For a defender to re-gain full or partial control, it is therefore critical to rapidly reconstruct the control graph as far as possible. Failing to achieve this may allow the attacker to cause further disruptions, and may — as in the case of electric power networks — also violate real-time constraints leading to catastrophic loss of control. However, as this problem is known to be computationally hard, approximations are required particularly for larger graphs. We therefore propose a reconstruction algorithm for (directed) control graphs of bounded tree width embedded in Erdős-Rényi random graphs based on recent work by Aazami and Stilp as well as Guo *et al.*

Categories and Subject Descriptors

C.4 [**Computer-Communication Networks**]: General—*Security and Protection*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph Algorithms*; F.2.3 [**Analysis of Algorithms and Problem Complexity**]: Non-Numerical Algorithms and Problems—*Tradeoffs between Complexity Measures*

*Contact author

General Terms

Robustness of Control Systems and Networks

Keywords

Structural Controllability, Power Dominating Sets, Recovery from Attacks

1. INTRODUCTION

The ability of an attacker to take over control of a distributed system or to deny the defender the same is a general problem, but of particular significance in cyber-physical systems where even temporary *loss of view* or *loss of control* can result in outright failure and severe cascading effects. Moreover, many such cyber-physical systems not only exhibit a safe *fail-stop* behaviour such that they can be brought to a halt in a safe state, but also have hard real-time requirements such as in the case of electrical power networks and their constituent elements.

This offers a strong motivation to study the ability of such systems to recover from deliberate attacks. Structural controllability, originally proposed in seminal work by Lin [10], offers a graph-theoretical interpretation for control systems as first described by Kalman [8]. Informally, controllability requires that a desired configuration can be forced from an arbitrary configuration in a finite number of steps; for a time-dependent linear dynamical system

$$\dot{x}(t) = \mathbf{A}x(t) + \mathbf{B}u(t), \qquad x(t_0) = x_0 \tag{1}$$

with $x(t) = (x_1(t), \ldots, x_n(t))^T$ representing the current state of a system with n nodes at time t, a $n \times n$ adjacency matrix $\mathbf{A}$ representing the network topology of interactions among nodes, and $\mathbf{B}$ the $n \times m$ *input* matrix ($m \leq n$), identifying the set of nodes controlled by a time-dependent *input vector* $u(t) = (u_1(t), \ldots, u_m(t))$ which forces the system to a desired state. The system in eq. 1 is *controllable* if and only if rank$[\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}] = n$ (*Kalman rank criterion*). The ability to efficiently identify *driver node* sets under certain assumptions as identified recently by Liu *et al.* [11] implicitly gives a criterion for both attackers and defenders for vertices and edges to target.

The approach by Liu *et al.* [11] is based on a non-rigorous variant of the maximum matching problem to identify a subset of driver nodes (i.e. vertices not sharing inputs); in this paper, however, we focus on the equivalent Power Dominating Set (PDS) problem originally proposed by Haynes *et al.* [7] as a refinement of Dominating Set, largely motivated by the structure of electric power networks and their efficient control. We consider the rapid re-construction of a PDS under attack as the more critical requirement relative to optimality of the resulting PDS and hence propose an approximation based on a dynamic programming approach for directed graphs where a tree of bounded width can be embedded in an Erdős-Rényi random graph.

2. PROBLEM AND RELATED WORK

Non-trivial control systems and controlled networks are necessarily sparse, and direct control of all nodes in such a network is not feasible as direct edges to these would typically result in too high costs as well as an out-degree of the controller node that would be difficult to realise in larger networks. Instead, the general case to be considered is for control to be indirect. However, as control systems will seek to minimise parameters such as latency, the formulation of PDS by Haynes *et al.* [7] extended the classic Dominating Set (DS) problem to seek a minimal Power Dominating Set where the covering rule is the same as in DS with a propagation rule. Here we consider a straightforward extension to directed graphs in definition 1.

Definition 1. (**Directed PDS**)
Let G be a directed graph. Given a set of vertices $S \subseteq V(G)$, the set of vertices that are power dominated by S, denoted by $P(S)$, is obtained as follows:

D1 *If a vertex v is in S, then v and all of its out-neighbors are in $P(S)$;*

D2 *(Propagation) if a vertex v is in $P(S)$, one of its out-neighbors w is not in $P(S)$, and all other out-neighbors of v are in $P(S)$, then w is inserted into $P(S)$.*

The robustness of such networks has been studied by Pu *et al.* [12] *inter alia*, subsequent work by both Liu *et al.* [11] and Wang *et al.* [13] has invesigated the effects of attacks including edge and vertex removal on the network and the subgraph representing the controlling PDS structures, clearly identifying the effect that network topology has on the impact achievable by such removal attacks. Further studies by Alcaraz *et al.* for both single-round and multi-round vertex removal attacks for different types of sparse graphs [3, 2] have identified optimised attack strategies.

One problem immediately arising from vertex removal from a minimal power dominating set is the *reconstruction* and recovery of control. However, even the basic DS problem is known to be $\mathcal{NP}$-complete with a polynomial-time approximation factor of $\Theta(\log n)$ as shown by Feige [5]. The approach by Feige gives a polynomial-time solution for graphs with maximum out-degree 2, otherwise the best currently known approach gives exponential time in $\mathcal{PSPACE}$ as shown recently for cubic graphs by Binkele-Raible and Fernau [4], requiring a trade-off in the complexity against the achievable approximation factor.

Given a directed graph $G = (V, E)$, constructed as $ER(n, p)$ with a set of vertices n, where each edge included in the graph G is determined independently with the edge probability p, we consider the directed graphs where the underlying undirected graph has bounded tree-width and have no self-loops nor parallel edges, but may have two edges with different directions on the same two end vertices (we call such edges antiparallel), and may have directed cycles. We assume that a PDS instance is given, where a set of vertices $S \subseteq V(G)$, denoted by $P(S)$, power dominates G if $P(S) = V(G)$. The resulting graph G is partitioned into k subgraphs $H_1, H_2, \ldots H_k$, where $1 \leq i \leq k$, such that the partition of G such that $\forall H_i : V_i \neq \emptyset$ and $\forall (i, j) \in \{1, \ldots k\}, i \neq j, V_i \cap V_j = \emptyset$. Each subgraph $H_i = (V_i, E_i)$ is weakly connected (i.e. the underlying undirected subgraph is connected). All subgraphs H_i are assumed to contain a tree-embedding satisfying a nice tree decomposition (where a tree decomposition of G can be transformed into a nice tree decomposition).

3. PDS TREE DECOMPOSITION

Guo *et al.* [6] developed a linear-time dynamic programming algorithm based on valid orientations for optimally solving PDS on graphs of bounded tree-width, introducing the notion of valid orientations for a new formulation of PDS (over undirected graphs). Computational complexity is dominated by determination of the mapping A_i for a join node, where for each bag state s we need to consider all pairs of compatible bag states of its two children. Therefore, the running time of Guo *et al.*'s algorithm is $O(nc^{k^2})$ where n denotes a set of tree nodes and c is a constant. Based on this orientation by Guo *et al.*, Azami and Stilp [1] reformulated *Directed PDS* in terms of valid colourings of the edges in order to develop an algorithm based on dynamic programming for Directed PDS. This algorithm is applied to directed graphs such that the underlying undirected graph has bounded tree-width. The aim of the valid colouring is to model the application of rules (D1) and (D2) of Directed PDS. Both algorthims are based on tree decompositions of graphs and their use with respect to dynamic programming. We now introduce some basic definitions:

Definition 2. (**Tree Decomposition**)
For $G = (V, E)$, tree decomposition of G is a pair $\langle X_i | i \in I, T \rangle$, where each X_i is a subset of V, called a bag, and T is a tree with the elements of I as nodes satisfying

1. $\bigcup_{i \in I} X_i = V$
2. *Every edge $u, v \in E$ has both ends in some X_i such that $u, v \subseteq X_i$*
3. *$\forall i, j, k \in I$: If j is on the unique path from i to k in T, then $X_i \cap X_k \subseteq X_j$*

Definition 3. (**Tree Width**)
The width of a tree decomposition $\langle X_i | i \in I, T \rangle$ is defined as $\max_{i \in I} |X_i| - 1$. The tree width of G, denoted by $tw(G)$, is defined as the minimum width k over all tree decompositions such that G has a tree decomposition of width k. The nodes of T are called T-nodes and X_i are called bags.

We subsequently rely on the special case of nice tree decompositions to simplify the design of the dynamic programming algorithm.

Definition 4. (**Nice Tree Decomposition**)
A nice tree decomposition $\langle X_i | i \in I, T \rangle$ for a graph $G = (V, E)$, where T is a rooted tree, is a tree decomposition for G if the following conditions are satisfied:

1. *Every node of the tree T has at most 2 children, which means that T is a binary tree*
2. *The nodes i of T are one of four node types:*

(a) ***Leaf*** *nodes without children and corresponding leaf bags X_i have $|X_i| = 1$.*
(b) ***Forget*** *nodes have one child j with $X_j = X_i \cup \{v\}$*
(c) ***Introduce*** *nodes with one child j where $X_i = X_j \cup \{v\}$*
(d) ***Join*** *nodes i have two children $j, k \in I$ with $X_i = X_j = X_k$*

For a graph of width k the following lemma due to Kloks [9] can be drawn on:

Lemma 1. **Nice Tree Decomposition**
Given a tree decomposition of a graph $G = (V, E)$ of width k and $\mathcal{O}(n)$ nodes, where n is the number of nodes in G, one can find a nice tree decomposition of G that has $\mathcal{O}(n)$ nodes and the same width k in time $\mathcal{O}(n)$.

Based on the valid orientation of undirected graphs, Aazami [1] introduced the reformulation of Directed PDS in terms of *valid colourings* of the edges. The key idea is to reformulate Directed PDS in order to develop an algorithm based on dynamic-programming for Directed PDS; our approach will also rely on this colouring approach to approximate PDS for the partition elements.

Definition 5. (**Valid Colouring**)
A colouring of a directed graph $G = (V, E)$ is a partition of the edges in G into red and blue edges. We denote a colouring by $C = (V, E_r \cup E_b)$ where E_r is the set of red edges and E_b is the set of blue edges. A ***valid*** *colouring $C = (V, E_r \cup E_b)$ of a digraph $G = (V, E)$ is a colouring of G with the following properties:*

1. *No two antiparallel edges can be coloured red.*
2. *For all other vertices covered by the red edges in $G_r = (V, E_r)$, $\forall v \epsilon G : d^-_{G_r}(v) \leq 1$ and $\forall v \epsilon G : d^-_{G_r}(v) = 1 \Longrightarrow d^+_{G_r}(v) \leq 1$ holds.*
3. *G has no dependency cycle.*
4. *A vertex with $d^-_{G_r}(v) = 0$ in $G_r = (V, E_r)$ is an origin of C*

The DP for Directed PDS is based on Lemma 2 due to Aazami with valid orientations are replaced by viewing blue edges as unoriented edges, and red edges as oriented edges:

Lemma 2. **Valid Colouring**
Given a directed graph G and $S \subseteq V(G)$, S power dominates G if and only if there is a valid colouring of G with S as the set of origins.

We now reformulate PDS of the partition elements of directed graph in terms of valid colourings of (similar to the formulation by [6], where blue and red edges play the same role as unoriented and oriented edges, respectively) to obtain a formulation for directed graphs; this allows the use of related results by Aazami for optimally solving PDS on graphs of bounded tree-width. Our approach applies to the partition elements of a directed graph such that the underlying undirected graph has bounded tree-width.

Definition 6. (**Digraph Partition Element Colouring**)
A colouring of the partition elements $H_i = (V, E)$ of a directed graph G is a partition of the edges in H_i into red and blue edges. We denote a colouring by $C = (V, E_r \cup E_b)$ where E_r is the set of red edges and E_b is the set of blue edges.

Definition 7. (**Origin of Valid Colouring**)
A vertex is an origin of the colouring of the partition elements $H_i = (V, E)$ of a directed graph G if it either has no in-degree in H_i, where $v \in H_i : d^-(v) = 0$, or if it has no in-coming red edges in H^r_i, where $v \in H^r_i : d^-(v) = 0$.

We can now define colourings in dependency paths:

Definition 8. (**Valid Colouring in Dependency Path**)
To detect dependency cycles, we define seven colours for every vertex $v \in H^{r/b}_i$ in a dependency path depending on the directions of in/out blue edges and red edges, where a colouring of vertices in a dependency path of the partition elements $H^{r/b}_i = (V, E_r \cup E_b)$ is assigning one of the colours $\{\mathcal{F}, \mathcal{M}, \mathcal{E}\}$ to each vertex with in/out red edges and $\{\mathcal{W}, \mathcal{Z}, \mathcal{Q}\}$ to each vertex with in/out blue edges and $\{\mathcal{L}\}$ to each vertex with in/out blue and edge edges (or in reverse) such that $\mathcal{F}$ is assigned to a vertex if there exists a vertex with no in-degree and at least one out-going red edge ($\exists v \in H^{r/b}_i : ((d^-_{H_i}(v) = 0) \wedge (d^+_{H^r_i}(v) = 1)) \Longrightarrow \mathcal{F}$), $\mathcal{M}$ if there exists a vertex with one in-coming red edge and no out-degree ($\exists v \in H^{r/b}_i : ((d^-_{H^r_i}(v) = 1) \wedge (d^+_{H_i}(v) = 0)) \Longrightarrow \mathcal{M}$), $\mathcal{E}$ if there exists a vertex with one in-coming red edge and one out-going red edge ($\exists v \in H^{r/b}_i : ((d^-_{H^r_i}(v) = 1) \wedge (d^-_{H^r_i}(v) = 1)) \Longrightarrow \mathcal{E}$), $\mathcal{W}$ if there exists a vertex with no in-degree and one out-going blue edge ($\exists v \in H^{r/b}_i : ((d^-_{H_i}(v) = 0) \wedge (d^+_{H^b_i}(v) = 1)) \Longrightarrow \mathcal{W}$), $\mathcal{Z}$ if there exists a vertex with one in-coming blue edge and no out-degree ($\exists v \in H^{r/b}_i : ((d^-_{H^b_i}(v) = 1) \wedge (d^+_{H_i}(v) = 0)) \Longrightarrow \mathcal{Z}$), $\mathcal{Q}$ if there exists a vertex with one in-coming blue edge and one out-going blue edge ($\exists v \in H^{r/b}_i : ((d^-_{H^b_i}(v) = 1) \wedge (d^-_{H^b_i}(v) = 1)) \Longrightarrow \mathcal{Q}$), and $\mathcal{L}$ if there exists a vertex with one in-coming red edge and one out-going blue edge or with one in-coming blue edge and one out-going red edge: ($\exists v \in H^{r/b}_i : ((d^-_{H^r_i}(v) = 1) \wedge (d^+_{H^b_i}(v) = 1)) \vee ((d^-_{H^b_i}(v) = 1) \wedge (d^+_{H^r_i}(v) = 1)) \Longrightarrow \mathcal{L}$).

Definition 9. (**Dependency Path**)
A dependency path in a valid colouring of the H_i, $P = v_1, e_1, v_2, e_2, \ldots, e_{i-1}, v_i$ is a sequence of vertices with colours $\mathcal{F}$, $\mathcal{M}$ and $\mathcal{E}$ (i.e. a sequence of red edges) such that P has no a vertex with the colour $\mathcal{Q}$ (i.e. no two consecutive blue edges) and P starts from a vertex (v_{i-1}) with the colour $\mathcal{F}$ and ends of a vertex (v_i) with the colour $\mathcal{E}$ (i.e. all red edges are directed away from a vertex with the colour $\mathcal{F}$ of P). The length of a dependency path is defined as the number of red edges in the path. A dependency cycle in the partition elements of a directed graph is a sequence of directed edges whose underlying undirected graph forms a cycle such that a cycle has vertices with the colours $\mathcal{F}$, $\mathcal{E}$ and $\mathcal{M}$ (i.e. all red edges in one direction), and two vertices with the colour $\mathcal{L}$ (i.e. all blue edges in the other direction), and there is no a vertex with colour $\mathcal{Q}$ (i.e. no two consecutive blue edges).

Proof of the following theorem is by contradiction, omitted in the extended abstract:

Theorem 1. **7-Colouring of Dependency Path**
Given a dependency path p, one can decide if p is a path or has a cycle, by colouring the vertices in the path with only seven colours.

We now define valid colourings for partition elements:

Definition 10. (**Valid Colouring of Partition Elements**)
A valid colouring $C = (V, E_r \cup E_b)$ of $H_i = (V, E)$ of the digraph G is a colouring of H_i satisfying:

1. *$\exists v \in V(H_i) : ((d^-_{H_i}(v) = 0) \wedge (d^+_{H_i}(v) \geq 2)) \Longrightarrow d^+_{H^r_i}(v) \geq 2$*

2. $\exists v \in V(H_i) : ((d^-_{H_i}(v) = 0)) \vee ((d^-_{H_i^b}(v) \geq 1) \wedge (d^+_{H_i}(v) \leq 1)) \Longrightarrow d^+_{H_i^r}(v) \leq 1$

3. Other vertices in $H_i \setminus (V_D \cup V_S)$ that are covered by the red edges in $H_i^r = (V, E_r)$, satisfy (I) $\forall v \in H_i : d^-_{H_i^r}(v) \leq 1$, and (II) $\forall v \in H_i : d^-_{H_i^r}(v) = 1 \Longrightarrow d^+_{H_i^r}(v) \leq 1$.

4. No two antiparallel edges can be coloured red.

5. H has no dependency cycle.

A vertex with $d^-_{H_i^r}(v) = 0 \in H_i^r = (V, E_r)$ is an origin of C.

This allows us to extend Aazami and Stilps result (proof omitted in the extended abstract) leading to the formulation of a dynamic programming algorithm:

Theorem 2. **Partition Colouring**
Given the partition elements $H_i = (V, E)$ of a directed graph G and $S \subseteq V(H_i)$, S power dominates H_i if and only if there is a valid colouring of H_i with S as the set of origins.

4. COLOURING AND REPAIR ALGORITHMS

We omit the formal definition of the algorithm and proof of correctness and only sketch the algorithms. We rely on the nice tree decomposition of Lemma 2 in linear time as we assume bounded tree width embeddings.

We determine the initial states by colouring edges in subgraphs induced by the partition, retaining sufficient information and then identify vertex degree states $s^d(v)$ in a bag for incoming and outgoing red edges; the seven vertex states for each bag are then sufficient to detect dependency cycles, which we can then concatenate in a bottom-up DP strategy. For this we must satisfy degree constraints to satisfy Definition 10 together with the colour constraints. Our algorithm is initialised by setting a mapping A_i for each leaf node i of T and for each bag state $s_i \in \Delta_i$ as $A_i(s_i) := \{+\infty\}$ if $(\exists v \in X_i : s_i^{d^-}(v) + s_i^{d^+}(v) \neq 0) \vee (\exists uv : (u \in X_i) \wedge (v \in X_i) \wedge (s_i(u,v) = \emptyset)) \vee ((\top[X_i], s_i) = \perp)$, otherwise we define $A_i(s_i)$ as the origin of vertices with $s_i^{d^-}(v) = 0$ in the colouring given by s_i: $A_i(s_i) := \{vu \in X_i : \exists e = \{v,u\} \in E(H_i[X_i]) \Longrightarrow s_i(e) = vu\}$. Hence only those bag states are taken into consideration where the edge states form a valid colouring. We then perform a bottom-up step visitin bags of the decomposition changing bag states for combining colourings (treating **Forget**, **Introduce**, **Join** nodes, and ultimately the **Root** node separately). The resulting algorithm algorithm will have a best-case complexity (where uncovered vertices W of H_i and where W are not **join** nodes) of $\mathcal{O}(nc^k)$, a worst-case complexity of $\mathcal{O}(nc^{k^2})$, and an average complexity of $\mathcal{O}(\log nc^{k^2})$ (proof omitted):

Lemma 3. **Colouring Algorithm Complexity**
Given partition elements H_i of a digraph G and a tree decomposition of width k, one can solve PDS in $\mathcal{O}(n\ c^{k^2})$ time.

A second *repair* algorithm can now also be formulated based on a nice tree decomposition $\langle X_i | i \in I, T\rangle$, of the partition elements $H_i = (V, E)$ of a digraph G and PDS of H_i, giving the respective PDS of H_i with equivalent computational complexity.

5. CONCLUSIONS

Our result reduces the complexity compared to Aazami and Stilp by reducing c, also reducing the space complexity owing to reducing the number of colours required; checking existence of a dependency cycle is possible in $\mathcal{O}(k)$. This allows faster (re-)construction of PDS, and ultimately the re-gaining of control for operators of control systems.

On-going work seeks to reduce the average-case complexity through partial re-use of PDS fragments and to identify criteria for the efficient addition of vertices to the sparse underlying graph. Future work will re-merge partitioned control graphs to permit the eventual full recovery of control with particular emphasis on approximation characteristics and minimising the recovered control graph diameter.

6. REFERENCES

[1] Aazami, A., and Stilp, K. Approximation Algorithms and Hardness for Domination with Propagation. *SIAM Journal on Discrete Mathematics 23*, 3 (Sep. 2009), 1382–1399.

[2] Alcaraz, C., Etchevés Miciolino, E., and Wolthusen, S. D. Multi-Round Attacks on Structural Controllability Properties for Non-Complete Random Graphs. In *Proceedings of the 16th Information Security Conference (ISC 2013)* (Dallas, TX, USA, Nov. 2013), Y. Desmedt, Ed., Lecture Notes in Computer Science, Springer-Verlag. *(in press)*.

[3] Alcaraz, C., Etchevés Miciolino, E., and Wolthusen, S. D. Structural Controllability of Networks for Non-interactive Adversarial Vertex Removal. In *Proceedings of the 8th International Workshop on Critical Information Infrastructures Security (CRITIS 2013)* (Amsterdam, The Netherlands, Sept. 2013), E. A. M. Luiijf and P. H. Hartel, Eds., vol. 8328 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 120–132.

[4] Binkele-Raible, D., and Fernau, H. An Exact Exponential Time Algorithm for Power Dominating Set. *Algorithmica 63*, 1–2 (June 2012), 323–346.

[5] Feige, U. A Threshold of $\ln n$ for Approximating Set Cover. *Journal of the ACM 45*, 4 (July 1998), 634–652.

[6] Guo, J., Niedermeier, R., and Raible, D. Improved Algorithms and Complexity Results for Power Domination in Graphs. *Algorithmica 52*, 2 (Oct. 2008), 177–202.

[7] Haynes, T. W., Hedetniemi, S. M., Hedetniemi, S. T., and Henning, M. A. Domination in Graphs Applied to Electric Power Networks. *SIAM Journal on Discrete Mathematics 15*, 4 (Aug. 2002), 519–529.

[8] Kalman, R. E. Mathematical description of linear dynamical systems. *Journal of the Society of Industrial and Applied Mathematics Control Series A 1* (1963), 152–192.

[9] Kloks, T. *Treewidth: Computations and Approximations*, vol. 842 of *Lecture Notes in Computer Science*. Springer-Verlag, Heidelberg, Germany, 1994.

[10] Lin, C.-T. Structual Controllability. *IEEE Transactions on Automatic Control 19*, 3 (June 1974), 201–208.

[11] Liu, Y.-Y., Slotine, J.-J., and Barabási, A.-L. Controllability of Complex Networks. *Nature 473* (May 2011), 167–173.

[12] Pu, C.-L., Pei, W.-J., and Michaelson, A. Robustness analysis of network controllability. *Physica A: Statistical Mechanics and its Applications 391*, 18 (Sep. 2012), 4420–4425.

[13] Wang, W.-X., Ni, X., Lai, Y.-C., and Grebogi, C. Optimizing controllability of complex networks by minimum structural perturbations. *Physical Review E 85*, 2 (Feb. 2012), 026115.