Recovery of Structural Controllability for Control Systems

Cristina Alcaraz, Stephen Wolthusen

Chapter 4

RECOVERY OF STRUCTURAL CONTROLLABILITY FOR CONTROL SYSTEMS

Cristina Alcaraz and Stephen Wolthusen

Abstract Fundamental problems in control systems theory are controllability and observability, and designing control systems so that these properties are satisfied or approximated sufficiently. However, it is prudent to assume that an attacker will not only be able to subvert measurements but also control the system. Moreover, an advanced adversary with an understanding of the control system may seek to take over control of the entire system or parts thereof, or deny the legitimate operator this capability. The effectiveness of such attacks has been demonstrated in previous work. Indeed, these attacks cannot be ruled out given the likely existence of unknown vulnerabilities, increasing connectivity of nominally air-gapped systems and supply chain issues. The ability to rapidly recover control after an attack has been initiated and to detect an adversary's presence is, therefore, critical. This paper focuses on the problem of structural controllability, which has recently attracted substantial attention through the equivalent problem of the power dominating set introduced in the context of electrical power network control. However, these problems are known to be $\mathcal{NP}$-hard with poor approximability. Given their relevance to many networks, especially power networks, this paper studies strategies for the efficient restoration of controllability following attacks and attacker-defender interactions in power-law networks.

Keywords: Control systems, structural controllability, power domination, resilience

1. Introduction

Domination, a central topic in graph theory, is a relevant theme in the design and analysis of control systems because it is an equivalent problem to that of (Kalman) controllability. The motivation comes from the concept

of structural controllability introduced by Lin [15], which is based on control theory as defined by Kalman [13]:

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

In this formulation, $x(t)$ is a vector $(x_1(t), \ldots, x_n(t))^T$ representing the current state of a system with n nodes at time t; $\mathbf{A}$ is an $n \times n$ adjacency matrix specifying the network topology that identifies interaction between nodes; and $\mathbf{B}$ is an $n \times m$ input matrix, where $m \leq n$, identifies the set of nodes controlled by a time-dependent input vector $u(t) = (u_1(t), \ldots, u_m(t))$ that forces the system to a desired state in a finite number of steps. According to Kalman's rank criterion, the system in Equation (1) is controllable if and only if:

$$\text{rank}[\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{n-1}\mathbf{B}] = n. \tag{2}$$

However, this formulation is quite restrictive for large networks (e.g., power networks or similarly large control systems) where there exists an exponential growth of input values as a function of nodes. This is the main reason that our investigations concentrate on structural controllability, where matrix $\mathbf{A}$ in Equation (1) represents the network topology and matrix $\mathbf{B}$ contains the set of nodes with the capacity to drive control [16].

Lin [15] defines $\mathcal{G}(\mathbf{A}, \mathbf{B}) = (V, E)$ as a digraph where $V = V_{\mathbf{A}} \cup V_{\mathbf{B}}$ is the set of vertices and $E = E_{\mathbf{A}} \cup E_{\mathbf{B}}$ is the set of edges. In this representation, $V_{\mathbf{B}}$ comprises the nodes capable of injecting control signals into the entire network, also known as driver nodes (denoted as n_d) corresponding to input vector u in Equation (1). The identification of these nodes has so far been studied in relation to general networks. This paper concentrates on power-law networks, most pertinent to a number of large-scale infrastructure networks. To identify the minimum driver node subsets $\mathbf{N}_D$, we follow the approach based on the power dominating set (PDS) problem, which is described in more detail in [1, 2]. This interest is primarily because PDS-based networks have similar logical structures as real-world monitoring systems, where driver nodes can represent, for example, remote terminal units that control industrial sensors and actuators. In fact, the PDS problem was originally introduced as an extension of the dominating set (DS) by Haynes, *et al.* [12], mainly motivated by the structure of electric power networks and the need to efficiently monitor the networks.

Building on previous work [1, 2], this paper proposes several restoration strategies for controlling a network after $\mathbf{N}_D$ has been perturbed. Different attack patterns that compromise nodes and the effects of the attacks have been considered extensively in [1, 2], in particular, the analysis and evaluation of interactive and non-interactive attacks, including multiple rounds between attackers and defenders, respectively. However, it is clearly undesirable to restore overall controllability through complete re-computation if the PDS properties are only partially violated – where this is possible given the constraints imposed by compromised nodes – because the PDS problem is known to be $\mathcal{NP}$-

complete for general graphs as well as for bipartite and chordal graphs as shown by Haynes, *et al.* [12].

Subsequent research by Guo, *et al.* [11] extended $\mathcal{NP}$-completeness proofs to planar, circle and split graphs, with the exception of partial k-tree graphs with $k \geq 1$ and parameterized using $\mathbf{N}_D$, in which the DS and PDS problems can become tractable in linear-time, while the parameterized intractability can result in $W[2]$-hardness [8]. Pai, *et al.* [17] have provided results for grid graphs while Atkins, *et al.* [3] have studied block graphs. There are other approaches that address PDS for specific cases [5, 6], but none of them focus on efficient solutions for the restoration of the PDS problem following perturbations, i.e., where a PDS of the original graph $\mathcal{G}$ is known along with the changes induced on $\mathcal{G}$.

The restoration strategies defined in this paper center on general power-law and scale-free distributions by offering similar characteristics to real power networks. In particular, three strategies are defined to determine the complexity of restoration. To evaluate the complexity, this paper considers: (i) a strategy without any type of constraint for restoration; (ii) a strategy based on the graph diameter to minimize the intrinsic problem of the non-locality of PDS; and (iii) a strategy based on backup instances of driver nodes. The paper shows that this offers a gain in efficiency over re-computation while resulting in acceptable deviations from an optimal (i.e., minimal $|\mathbf{N}_D|$) PDS. Because many critical infrastructures require timely or even real-time bounded restoration to ensure resilience and continued operation, the ability to restore controllability rapidly is essential and, of course, highly desirable.

2. Conditions for the Analysis

This section discusses the initial assumptions and conditions used to restore the structural controllability when nodes are attacked from within a network. Let $G(V, E)$ be a directed acyclic graph (DAG) based on an arbitrary set of nodes V and a set of edges E, where each vertex $v_i \in V$ can be linked to other $v_j \in V$ such that $(v_i, v_j) \in E$, without producing loops or self-loops (i.e., $v_i \neq v_j$).

2.1 Assumptions for Perturbation

The first assumption we consider here is that one or several vertices can be targeted by one or several attackers, knowing the structure or probability distribution of edges of the graph, its topology, and the identities of the current driver nodes $\mathbf{N}_D$ (note that $\mathbf{N}_D$ is not necessarily unique). These driver nodes that also belong to V satisfy the two observation rules for controllability, which were simplified by Kneis, *et al.* [14] from the original formulation specified by Haynes, *et al.* [12]. The two rules and their algorithms are detailed in [1, 2] and below:

- **OR1:** A vertex in $\mathbf{N}_D$ observes itself and all its neighbors.

- **OR2:** If an observed vertex v of degree $d \geq 2$ is adjacent to $d-1$ observed vertices, the remaining unobserved vertex becomes observed as well.

Note that the omission of OR2 already results in the $\mathcal{NP}$-complete DS problem with a polynomial-time approximation factor of $\Theta(\log n)$ [9]. The following condition is that the construction of $\mathbf{N}_D$ is arbitrary and depends on the selection of vertices satisfying OR1, allowing the customizable selection of controllability generation strategies as specified in [1]. After $\mathbf{N}_D$ has been obtained, we evaluate two different scenarios concentrating on attacks against either node or edge (communications link) availability [1, 2]:

- **SCN-1:** Randomly remove some (not all) edges of one or several vertices, which may compromise the controllability of dependent nodes or disconnect parts of the control graph and underlying network.

- **SCN-2:** Randomly isolate one or several vertices from the network by intentionally deleting all their links (i.e., this attack may result in the complete isolation of nodes from the network).

As detailed in [1, 2], either attack scenario may result in a degradation of the control of a network and a significant reduction in observability (including partial observability). To address this aspect, we identify two classes of nodes:

- **U-1:** The node u is not observed by an n_d, but belongs to $\mathbf{N}_D$ and is part of the control node set.

- **U-2:** The node u is not observed by an n_d and does not belong to $\mathbf{N}_D$. This means that u is part of the set of observed nodes, denoted as O, such that $O = V - \mathbf{N}_D$.

When such a node is not being observed by a member of $\mathbf{N}_D$, the set of unobserved nodes U has to be updated so that each node $u \in U$ can be again observed by at least one member of $\mathbf{N}_D$.

2.2 Assumptions for Restoration

We assume that the restoration of structural controllability $\mathbf{N}_D$ is initially based on searching driver nodes in $\mathbf{N}_D$ that offer the coverage of unobserved nodes in U with dependence on attacked nodes in A. The term coverage refers to the ability of a new link to be established between the best candidate in $\mathbf{N}_D$ and an unobserved node in U such that the two observation rules OR1 and OR2 specified above are satisfied. For this, the candidates for restoring the controllability must have the following properties:

- Satisfy the conditions of OR1, i.e., select an $n_d \in \mathbf{N}_D$ capable of observing itself and an unobserved $u \in U$ through a new link $(n_d, u) \in E$ such that $\mid \mathbf{N}_D \mid \geq 1$.

- None of the new restoration links must violate the out-degree distribution of a power-law network and must not introduce cycles.

- Satisfy the conditions of OR2, i.e., verify that $\forall\ n_d \in \mathbf{N}_D$ of degree $d \geq 2$, OR2 is not infringed. This can involve the inclusion of one or several new members in $\mathbf{N}_D$ such that $\mid \mathbf{N}_D \mid \leq \mid\ V \mid$.

At the end of the algorithm, the restored set $\mathbf{N}_D$ can increase the initial number of driver nodes such that $U = \oslash$ (note that $\mid \mathbf{N}_D \mid = \mid V \mid$ in degenerate cases). However, and unfortunately, we must also consider the handicap of non-locality of PDS and the $\mathcal{NP}$-complete property demonstrated by Haynes, *et al.* [12].

Our heuristic approach is based on ensuring that the hard constraints, i.e., observation rules, are satisfied primarily and that, as a secondary constraint, the out-degree distribution property of the underlying power-law network remains unaltered. This strategy also depends on the approaches taken for each restoration strategy defined in the remainder of the paper. In this case, the study is based on two main approaches:

- **APPR-1:** Find an $n_d \in \mathbf{N}_D$ to re-link it to an unobserved node $u \in U$.

- **APPR-2:** Find an $n_{d_{bl}}$ belonging to a backup list of driver nodes such that there is an edge between $n_{d_{bl}}$ and unobserved node $u \in U$.

Likewise, each restoration strategy has to consider some of the following "restoration rules" (heuristics):

- **RR1:** If u is a U-1, then it is necessary to ensure that u still satisfies OR1.

- **RR2:** If u is a U-2, and the restoration strategy follows the APPR-1 approach, it is necessary to first find the driver node $n_d \in \mathbf{N}_D$ with out-degree equal to zero, or find a vertex $n_d \in \mathbf{N}_D$ of out-degree $d \geq 2$ such that $|\mathit{children}(n_d) - \mathbf{N}_D \mid \geq 1$ where $\mathit{children}(n_d)$ is a function that obtains the set of child nodes corresponding to the out-degree of n_d. In this way, the violation of OR2 after the link is avoided.

- **RR3:** If u is a U-2 and the restoration strategy follows the APPR-2 approach, it is necessary to first find the driver node $n_{d_{bl}}$ of a given backup list with out-degree equal to one (pointing out to itself), or find an $n_{d_{bl}}$ in the backup list of out-degree $d \geq 2$ such that $|\mathit{children}(n_{d_{bl}}) - \mathbf{N}_D \mid > 1$ to avoid violating OR2.

3. Restoration of Structural Controllability

The three restoration rules given in Section 2.1 are the basic constraints to address the following three restoration strategies:

Algorithm 1 : Basic_Re-Link($\mathcal{G}(V,E), \mathbf{N}_D, U, A$).

output $\mathbf{N}_D$
local $S_1, u, n_d, or2$;
$or2 \leftarrow false$;
while $U \neq \oslash$ **do**
  $(\star)$ *Randomly choose a vertex* $u \in U$;
  if $u \notin \mathbf{N}_D$ **then**
    $(\star)$ $(S_1 \leftarrow \forall\, n_d \in \mathbf{N}_D - A$ *with maximum in_degree*$)$
    and $((n_d, u) \in E$ *is DAG*$)$;
    $(\star)$ $Common\_Relink(\mathcal{G}(V,E), \mathbf{N}_D, S_1, u, U, A, or2)$;
  end if
  $U \leftarrow U \setminus \{u\}$;
end while
return $(\star)$ $verifyOR2(\mathcal{G}(V,E), \mathbf{N}_D, A, or2)$;

- **STG-1:** No constraints through APPR-1.
- **STG-2:** Parameterization using the network diameter and APPR-1.
- **STG-3:** A backup list of driver nodes through APPR-2.

This section develops and analyzes the three associated algorithms, considering in addition the parameters and functions described above.

3.1 STG-1: Restoration Algorithm and Analysis

For any attack scenario (SCN-1 and SCN-2), the approach involves finding the candidates in $\mathbf{N}_D$ that can provide coverage to each vertex contained in U through a new edge. This approach is specified by Algorithm 1, where the symbol $(\star)$ is an indication for the complexity analysis given in Section 4.

We briefly outline the semantics of Algorithm 1. For each unobserved node, the first step is to verify that it is part of $\mathbf{N}_D$. If a vertex $u \in U$ is an n_d by itself (U-1), then it is not necessary to find a member of $\mathbf{N}_D$ to establish the link because such a node observes itself, satisfying the first restoration condition (RR1) given in Section 2.2. Otherwise, a non-attacked n_d is randomly chosen to proceed with link restoration. However, because this new link (generated by Algorithm 2) can change the power-law distribution given in $\mathcal{G}(V,E)$, only the candidates with the highest in-degree (≥ 0) are chosen so as not to skew the degree distribution, effectively obtaining a preferential attachment process [4]. From these candidates, those that do not produce cycles after the attachment are selected in order to comply with the second assumption given in Section 2.2.

Regardless of the type of restoration strategy (STG-1, STG-2, STG-3) and the state of U, the verification of the existence of nodes in $\mathcal{G}(V,E)$ that violate OR2 after a perturbation is always required. To be more concise, the analysis is reduced to a subset of nodes instead of the entire graph; this subset contains the nodes related to the set of A. Moreover, depending on the target (TG) that is attacked, the analysis can vary:

Algorithm 2 : Common_Re-Link($\mathcal{G}(V,E), \mathbf{N}_D, S_1, u, U, A, or2$).

output $\mathcal{G}(V,E), \mathbf{N}_D, or2$
local S_2, n_d;
if $S_1 \neq \oslash$ **then**
  $(\star)$ $S_2 \leftarrow \forall\ n_d\ \in\ S_1 /\ (out\_degree\ ==\ 0)$ *or* $(\mid children(n_d) - \mathbf{N}_D \mid \geq\ 1)$;
  if $S_2 \neq \oslash$ **then**
    Randomly choose a $n_d\ \in\ S_2$;
  else
    $(\star)$ *Randomly choose a* $n_d\ \in\ S_1$;
    $(\star)$ $\mathbf{N}_D \leftarrow \mathbf{N}_D\ \cup\ \{n_d\}$; $(\star)$ $or2 \leftarrow true$;
  end if
  Establish a link between n_d *and* u, *such that* $(n_d, u)\ \in\ E$;
else
  $\mathbf{N}_D \leftarrow \mathbf{N}_D\ \cup\ \{u\}$; $or2 \leftarrow true$;
end if

- **TG-1:** An n_d has been attacked, so OR2 is performed for each n_d in A. However, the verification process is only effective for scenarios SCN-1 in which the state of each child of an affected node has to be evaluated. This process is disrupted when there is an $n_d \in A$ of degree ≥ 2 with $\mid children(n_d) - \mathbf{N}_D \mid = 1$.

- **TG-2:** A node $v \in O$ has been attacked, so OR2 is applied for each $v \in A$. However, the analysis is only effective for scenarios SCN-1 where the algorithm is applied to the father nodes $n_{d_{fv}}$ related to v and $n_{d_{fv}} \in \mathbf{N}_D$. Because of TG-1, the proof may be interrupted when there is an $n_{d_{fv}}$ of degree ≥ 2 with $\mid children(n_{d_{fv}}) - \mathbf{N}_D \mid = 1$. The set of driver fathers is obtained through the function *fathers(v)* corresponding to the in-degree of v.

As stated in Algorithm 3, the breach of OR2 involves an update of $\mathbf{N}_D$ and the execution of Algorithm OR2, which is detailed in [1]. On the other hand, the correctness proof of STG-1 involves induction:

- **Precondition:** $A \neq \oslash$ such that $\mid \mathbf{N}_D - A \mid \geq 1$.

- **Postcondition:** $U = \oslash$, and OR1 and OR2 are fulfilled.

- **Case 1:** $U = \oslash$ after perturbation (SCN-1 or SCN-2). Although the `while` loop in Algorithm 1 is not processed, Algorithm 3 must be executed to verify the fulfillment of OR2. Depending on the attack scenario, the resolution of Algorithm 3 changes. For scenarios SCN-1, the loops for the sets $attacked\_N_D$ and $attacked\_O$ must be launched to detect the existence of one driver node ($\in attacked\_N_D$) or parent drivers ($\in attacked\_O$) that violate the second controllability rule. In contrast, such sets are not considered for SCN-2 scenarios because the affected nodes are completely isolated, without children and parent vertices, satisfying OR2 through $out\_degree = 0$.

Algorithm 3 : Common_VerifyOR2($\mathcal{G}(V,E), \mathbf{N}_D, A, or2$).

output $\mathbf{N}_D$
local $n_d, attacked\_N_D, attacked\_O, or2, i$;
$attacked\_N_D \leftarrow N_D \cap A$; $attacked\_O \leftarrow A - N_D$;
if $(attacked\_N_D \notin \oslash)$ *and* $(or2 == false)$ **then**
  if $\exists$ *a* $n_d \in attacked\_N_D$ *that breaks* OR2 **then**
    $\mathbf{N}_D \leftarrow \mathbf{N}_D \cup \{children(n_d) - \mathbf{N}_D\}$; $or2 \leftarrow true$;
  end if
end if
if $(attacked\_O \notin \oslash)$ *and* $(or2 == false)$ **then**
  $i \leftarrow 1$;
  while $(i \leq \mid attacked\_O \mid)$ *and* $(or2 == false)$ **do**
    $S_1 \leftarrow fathers(attacked\_O[i])$;
    if $\exists$ *a* $s_1 \in S_1$ *that breaks* OR2 **then**
      $\mathbf{N}_D \leftarrow \mathbf{N}_D \cup \{children(s_1) - \mathbf{N}_D\}$; $or2 \leftarrow true$;
    end if
    $i \leftarrow i + 1$;
  end while
end if
if $or2$ **then**
  Execute OR2 *defined in* [1];
end if
return $(\star)$ $\mathbf{N}_D$;

- **Case 2:** $U \neq \oslash$ after perturbation, being $\mid U \mid = 1$. In these circumstances, two cases must be distinguished:

  (i) If u is U-1, the condition RR is met.

  (ii) If u is U-2, it is necessary to explore the existence of one or several candidates $\{n_{d_1}, \ldots, n_{d_n}\}$ with maximum in-degree (≥ 0) and with the capability to cover u without producing cycles and complying with RR2. If this is the case, we ensure that $\exists$ an $n_d \in \mathbf{N}_D$ for coverage, and u therefore becomes part of O, guaranteeing that U is null for next iteration. If not, $\mathbf{N}_D$ is updated with $\mathbf{N}_D \cup \{u\}$ to be observed at least by itself, where $U = \oslash$ in the next iteration. However, this updating involves performing a verification process of OR2 [1] to determine the observation degree of the entire network after the loop of Algorithm 1.

- **Induction:** Assuming that we are in step k ($k > 1$) with $U \neq \oslash$, $k =\mid U \mid$ and $\mid \mathbf{N}_D \mid \geq 1$, we randomly select a node $u \in U$ in each iteration of the `while` loop. When selecting a node, two cases can arise depending on u (U-1 or U-2), which pursue the same goals as Case 2 (but with $\mid U \mid > 1$). At the end of Algorithm 1, the set U and k are always updated through $U = U \setminus \{u\}$ (see Case 2). In the next state, with $k - 1$, the procedure adopted is still valid, which means that the postcondition $U = \oslash$ is not met and the loop must be run again for the next state k until $k = 0$.

Algorithm 4 : Diameter-Based Relinking($\mathcal{G}(V,E), \mathbf{N}_D, U, A$).

output $\mathbf{N}_D$
local $S_{d_1}, S_{d_2}, S_1, u, n_d, or2$;
$or2 \leftarrow false$;
while $U \neq \emptyset$ **do**
  $(\star)$ *Randomly choose a vertex* $u \in U$;
  if $u \notin \mathbf{N}_D$ **then**
    $(\star)$ $S_{d_1} \leftarrow BFS(\mathcal{G}(V,E))$;
    $(\star)$ $S_{d_2} \leftarrow \forall\, n_d \in \mathbf{N}_D - A$ *with minimum diameter* $\in S_{d_1}$ *and* $(n_d, u) \in E$ *is DAG*;
    if $S_{d_2} \neq \emptyset$ **then**
      $(\star)$ $S_1 \leftarrow \forall\, n_d \in S_{d_2}$ *with maximum in_degree*(n_d);
      $(\star)$ *Common_Relink*$(\mathcal{G}(V,E), \mathbf{N}_D, S_1, u, U, A, or2)$;
    else
      $\mathbf{N}_D \leftarrow \mathbf{N}_D \cup \{u\}$; $or2 \leftarrow true$;
    end if
  end if
  $U \leftarrow U \setminus \{u\}$;
end while
return $(\star)$ $verifyOR2(\mathcal{G}(V,E), \mathbf{N}_D, A, or2)$;

When $k = 0$, Case 1 occurs, and therefore the postcondition is true and Algorithm 1 terminates.

3.2 STG-2: Restoration Algorithm and Analysis

One extension of STG-1 is to consider the network diameter as described in Algorithm 4. By induction, the proof of STG-1 can be expanded by taking into account the initial and final conditions and base cases. For each iteration k ($k > 1$) with $U \neq \emptyset$ and $\mid \mathbf{N}_D \mid \geq 1$, a node $u \in U$ is selected randomly. As in the previous proof, we distinguish two types of affected nodes. If the node is U-1, then RR1 is still satisfied. However, if the node is U-2, then $\forall$ $n_d \in \mathbf{N}_D - A$ nodes with the minimum diameter are selected to ensure acyclic graphs after repair. Because the graph is unweighted, breadth-first search is used to obtain a list of nodes together with their diameters, and through this list the n_d with minimum diameter (≥ 0) with respect to the entire graph are obtained.

In the case where there does not exist a candidate node that satisfies all the constraints, the unobserved node becomes part of the $\mathbf{N}_D$ to guarantee at least OR1; otherwise, the inductive proof of STG-1 can be employed. At the end of the loop, the precondition $\mid U \mid = k$ is updated in each stage k by computing $U = U \setminus \{u\}$ until $k = 0$. As a result of the proof of STG-1, the postcondition is true and Algorithm 4 terminates when Case 1 as defined in STG-1 is finally reached.

3.3 STG-3: Restoration Algorithm and Analysis

This strategy requires initial pre-processing before the generation of backup instances composed of driver nodes in $\mathcal{G}(V, E)$. These instances have to be organized into a tree-like structure based on the concept of "nice tree decomposition." To do this, a previous construction of a tree decomposition must be built, taking into account the network diameter for later transformation into a nice tree decomposition. A tree decomposition is a tree T of $\mathcal{G}(V, E)$ with I nodes, where each node in T is a bag containing a set of $n_d \subseteq \mathbf{N}_D$ satisfying the following properties [11]:

- **Property 1:** $\bigcup_{i \in T} bag_i = \mathbf{N}_D$.
- **Property 2:** $\forall\ (n_{d_{bl_w}}, n_{d_{bl_z}}) \in E$ with diameter ≥ 0, there exists a bag_i in T such that $(n_{d_{bl_w}}, n_{d_{bl_z}}) \subseteq bag_i$.
- **Property 3:** $\forall\ bag_i, bag_j, bag_z \in T$, if bag_j is on the path from bag_i to bag_z in T, then $bag_i \cap bag_z \subseteq bag_j$.

The tree width corresponds to the minimum width w over all tree decompositions of $\mathcal{G}(V, E)$, where $w = max_{i \in I}(\mid bag_i \mid \in T) - 1$ and $w \geq 1$. This means that a tree decomposition T of width w with $\mid \mathbf{N}_D \mid$ driver nodes can be turned into a nice tree decomposition of width w, but subject to the diameter associated with each driver node within the network [7]. In this way, bags containing driver nodes with smaller diameters are the leaves of T while driver nodes with higher diameters are located closer to the root.

For transformation to a nice tree decomposition, each node i in the tree T has at most two children (j, z) complying with two additional conditions: (i) nodes with two children bag_j and bag_z, $bag_i = bag_j = bag_z$ (bag_i as a join node); and (ii) nodes with a single child bag_j such that $bag_i = bag_j \cup \{n_d\}$ (bag_i as an introduce node) or $bag_i = bag_j - \{n_d\}$ (bag_i as a forget node). In practice, these trees are constructed using tables with at least three columns (i, j, z), where each entry i contains those subsets of n_d in relation to i. However, this data structure also takes into account the maximum diameter associated with each bag because the approach does not focus on re-linking (APPR-1), the value of which remains constant throughout the restoration process. Therefore, the spatial overhead for such a table may become $3 \times 2^{w+1} = O(2^{w+1})$ entries [10].

Algorithm 5 describes the behavior of the restoration strategy with one or several nice tree decompositions T_{bk} as the main input parameter with a storage cost of $O(\sum_{bk=1}^{M} 2^{w+1})$. The idea is to process this parameter in a bottom-up fashion to find the driver nodes with minimum diameter that ensure the fulfillment of RR3 specified in Section 2.2. The inductive proof begins by defining the initial and final conditions, and the base cases:

- **Precondition:** $A \neq \oslash$ with at least one T_{bk_j} with $M \geq j \geq 1$.
- **Postcondition:** $U = \oslash$, and OR1 and OR2 are fulfilled.

Algorithm 5 : Backup Instance-Based Scheme($\mathcal{G}(V,E), N_D, A, U, T_{bk}, M$).

output $\mathbf{N}_D$
local $current\_diam, S_1, S_2, u, n_d, or2, bk$;
$or2 \leftarrow false$;
while $U \neq \emptyset$ **do**
  Randomly choose a vertex $u \in U$;
  if $u \notin N_D$ **then**
    for $bk \leftarrow 1$ *to* M **do**
      $current\_diam \leftarrow \infty$;
      while ($maximun(diameter\ in\ bag_i) \leq current\_diam$) *and*
        (! *visited the whole* T_{bk}) **do**
        if ($\exists$ *a* $n_{d_{bl}} \in (bag_i - A)$ *such that* $(n_{d_{bl}}, u) \in E$) *and*
          (($out\_degree == 1$) *or* ($|\ children(n_{d_{bl}}) - \mathbf{N}_D\ | > 1$)) **then**
          ($\star$) $current\_diam \leftarrow maximum(diameter\ in\ bag_i)$;
          if $n_{d_{bl}} \in \mathbf{N}_D$ **then**
            $S_1 \leftarrow S_1 \cup \{n_{d_{bl}}\}$;
          else
            ($\star$) $S_2 \leftarrow S_2 \cup \{n_{d_{bl}}\}$;
          end if
        end if
      end while
    end for
    if $S_1 = \emptyset$ **then**
      if $S_2 \neq \emptyset$ **then**
        ($\star$) *Randomly choose a vertex* $s_i \in S_2$; ($\star$) $N_D \leftarrow N_D \cup \{s_i\}$;
      else
        ($\star$) $N_D \leftarrow N_D \cup \{u\}$; ($\star$) $or2 \leftarrow true$;
      end if
    end if
  end if
  $U \leftarrow U \setminus \{u\}$;
end while
return ($\star$) $verifyOR2(\mathcal{G}(V,E), \mathbf{N}_D, A, or2)$;

- **Case 1:** Analogous to Case 1 of the STG-1 proof in Section 3.1.
- **Case 2:** $U \neq \emptyset$ after perturbation, being $|\ U\ | = 1$. As in Case 2 of the STG-1 proof, two sub-cases must be distinguished:
  (i) If u is **U-1**, then the condition RR1 is satisfied.
  (ii) If u is **U-2**, then Algorithm 5 needs to traverse all trees T_{bk_j} from the bottom to locate the bags in T_{bk_j} that contain the best driver candidates to cover u. This process involves the verification of the existence of an $n_{d_{bl}} \in bags_i - A$ such that $(n_{d_{bl}}, u) \in E$ with minimum diameter in which RR3 is fulfilled. During this process, we also explore if such an $n_{d_{bl}}$ belongs to $\mathbf{N}_D$ to avoid increasing $\mathbf{N}_D$. If so, the set S_1 is updated through $S_1 \cup \{n_{d_{bl}}\}$; otherwise, S_2 is updated. In the case where $S_1 \neq \emptyset$, we ensure that u is covered by

at least one member in $\mathbf{N}_D$ and the set O is updated, guaranteeing that U is empty in the next iteration. In contrast, if there is no perfect candidate (as above) in $\mathbf{N}_D$ and $S2 \neq \oslash$, we also guarantee the existence of an $n_{d_{bl}} \in T_{bk_j}$ with the ability to cover u, and hence $O = O \cup \{u\}$ and $U = \oslash$ for the next iteration. However, $\mathbf{N}_D$ must be updated with $\mathbf{N}_D \cup \{n_{d_{bl}}\}$, requiring Algorithm 5 to verify the observation degree of the entire network when the loop finishes.

This verification process, described in detail in Section 3.1, may also be performed when there is no perfect candidate ($S_1 = \oslash$ and $S_2 = \oslash$) to cover u. In this case, u becomes part of $\mathbf{N}_D$ to comply with OR1, and hence $U = \oslash$ in the next iteration. When $U = \oslash$ and the rule OR2 is satisfied, the postcondition is true.

- **Induction:** In step k ($k > 1$) with $U \neq \oslash$, $k =\mid U \mid$ and $\mid \mathbf{N}_D \mid \geq 1$, we randomly select a node $u \in U$ in each iteration of the loop. When selecting a node, two situations can occur depending on u: U-1 or U-2, and both following the same goals set out for Case 2 (of this proof), but with $\mid U \mid > 1$. At the end of the algorithm, the set U and k are always updated through $U = U \setminus \{u\}$. In the next state, with $k - 1$, the procedure adopted is still valid, which means that the postcondition $U = \oslash$ is not met and the loop must be run up again for the next state k until $k = 0$. When this happens, Case 1 of the STG-1 proof must be verified to conclude that the postcondition is true, and therefore Algorithm 4 terminates.

4. Complexity Analysis and Discussion

This section analyzes the computational complexity of the three restoration algorithms, Algorithm 1, Algorithm 4 and Algorithm 5. In the case of STG-1, it is required to process the entire U set k times where $k =\mid U \mid$. For these k iterations, the algorithm must also find the best candidates in $\mathbf{N}_D - A$ with the highest in-degree to ensure the fulfillment of RR2 in the best scenario, or increase $\mathbf{N}_D$, at least, by one unit in the worst case.

For simplicity, we denote $\mid V \mid = n$, $\mid E \mid = e$, $\mid A \mid = a\ (= 1)$, $\mid \mathbf{N}_D \mid = nd$ and $f = fathers(n_d)$; and we study the upper bounds for SCN-1 and SCN-2. To evaluate the worst scenario of each SCN-x ($x = \{1, 2\}$), we assume that $nd \approx n$ and the adversarial scenario is non-interactive (a single target TG-x ($x = \{1, 2\}$)), so that if $A \subseteq \mathbf{N}_D$, then $nd - a \approx n$ as well. In addition, we must also select the longest trace of Algorithm 1 that includes Algorithm 3, following the indication given by $\star$ – note that both the assignment and `if` instructions have constant complexity $O(i)$ and can be neglected. To address this aspect, we first evaluate the upper bound needed to find the non-attacked driver nodes ($\mathbf{N}_D - A$) with maximum in-degree (≥ 0) that satisfy the directed acyclic test after repair and RR2. The computation time of this entire process may become $O(kn^2)$ if $nd \approx n$.

Depending on SCN-x and the targeted node TG-x, the computational complexity of Algorithm 3 can become variable as described in Section 3.1:

- **TG-1 in SCN-1:** In this case, it is necessary to verify OR2 for each attacked driver node in $attacked\_N_D$ with a cost of $O(a+e)$. As we are evaluating the worst scenario, we must observe that after computing all the nodes in $attacked\_N_D$, there exists an n_d that infringes OR2, which forces Algorithm 3 to compute Algorithm OR2 given in [1] with an overhead of $O(nd(nd+e)) = O(n^2)$. Therefore, the total complexity for this scenario is $O(kn^2 + ((a+e)+n^2)) = O(kn^2)$.
- **TG-1 in SCN-2:** The verification of OR2 is not possible because of the complete isolation of the nodes in $attacked\_N_D$; hence $O(kn^2+(a+e)) = O(kn^2)$.
- **TG-2** in **SCN-1**: This attack scenario requires Algorithm 3 to explore the existence of a parent $n_{d_{fv}}$ related to $v \in attacked\_O$ that does not comply with OR2. This entails an upper bound of $O(kn^2 + (a(f+e) + n^2)) = O(kn^2)$.
- **TG-2 in SCN-2:** This case is similar to TG-1 in the SCN-2 proof.

The extension of $\mathbf{N}_D$ can be influenced according to:

- **TG-x in SCN-1:** An increase of at least two new n_d in $\mathbf{N}_D$.
- **TG-x in SCN-2:** An increase of one unit in $\mathbf{N}_D$ in the worst case.

The computational cost of Algorithm 4 is analogous to that of restoration strategy STG-1 (Algorithm 1), but this time it is necessary to consider the overhead involved in the best-first search ($O(n+e)$) to compute the diameter of the entire network. After the list with diameter values is obtained, the driver nodes related to $\mathbf{N}_D - A$ are extracted in order to validate them with an acyclicity test ($O((nd-a)(n+e)) = O(n^2)$) and to subsequently obtain the driver nodes with the highest in-degree ($O(nd-a) = O(n)$) that comply with RR2 ($O((nd-a)+e) = O(n+e)$). The overhead of the first part is so far $O(k((n+e)+n^2+n+(n+e))) = O(kn^2)$ if $nd \approx n$. The rest of the analysis follows the same steps as for SCN-x and TG-x stated above, with the results summarized in Table 1.

With regard to strategy STG-3, we simplify the study considering $b = w+1$ (largest bag in T_{bk_j}) and the worst case with $n_d \approx n$. To compute a bag bag_i of T_{bk_j} with $j \leq M$, Algorithm 5 must identify the existence of an $n_{d_{bl}}$ that complies with RR3, which yields a cost of $O(b+e)$. To obtain the best candidates of each backup instance T_{bk_j} stored in memory, the algorithm needs to process each tree with a computational cost of $O(\sum_{bk=1}^{M} 2^{w+1}(b+e))$. The second part of the approach follows the same approach described above for Algorithm 3, which is summarized below and in Table 1:

Table 1. Complexity of the three restoration strategies.

	Threat Scenarios			
	SCN-1 – TG-x		**SCN-2 – TG-x**	
	Time	$\mathbf{N}_D$	**Time**	$\mathbf{N}_D$
STG-1	$O(kn^2)$	$nd+2$	$O(kn^2)$	$nd+1$
STG-2	$O(kn^2)$	$nd+2$	$O(kn^2)$	$nd+1$
STG-3	$O(k(\sum_{bk=1}^{M}(2^{w+1}(b+e))))$	$nd+2$	$O(k(\sum_{bk=1}^{M}(2^{w+1}(b+e))))$	$nd+1$

- **TG-x in SCN-1:** $O(k(\sum_{bk=1}^{M}(2^{w+1}(b+e)))) + O((a+e)+n^2)$, resulting in $O(k(\sum_{bk=1}^{M}(2^{w+1}(b+e))))$, where $\mathbf{N}_D$ increases its value at least in two nodes in the worst case.
- **TG-x in SCN-2:** $O(k(\sum_{bk=1}^{M}(2^{w+1}(b+e)))) + O(n^2) = O(k(\sum_{bk=1}^{M}(2^{w+1}(b+e))))$, where $\mathbf{N}_D$ increases its value at least in one node.

Therefore, the computational cost of STG-3 depends on the width $w+1$ of T_{bk_j}, whose cost can become undesirable for critical scenarios where the control has to be resolved in linear time. However, this study concentrates on the worst cases where $nd \approx n$, without considering the ability of the approach to prepare each backup instance using the diameter in the best cases. Similarly, STG-1 can also be an inadequate strategy with respect to STG-2 because the diameter computed in STG-2 benefits the fulfillment of RR2 ($out\_degree = 0$), reducing the computational costs and the expansion of $\mathbf{N}_D$ in each iteration. On the other hand, STG-1 and STG-2 have to transverse the entire network to search for the best candidates that satisfy conditions RR1 and RR2 (explicitly taking into account non-locality); while STG-3 must go over each backup instance to obtain the best candidates that satisfy condition RR3. Nevertheless, the dynamic computation of the diameter in STG-2 again highlights the benefit of the strategy to mitigate the non-locality problem of PDS inherent in the strategies by pre-computation.

On the other hand, we have implemented the three strategies over a power-law distribution called PLOD [18], which is analyzed in [1]. The developments are based on Matlab with a low connectivity probability to produce a more realistic critical scenario with sparse distributions, using y^{α} with $\alpha = 0.2$ and networks with medium ($\leq$ 1,000) and large ($\leq$ 3,100) numbers of nodes. For each network produced, we have analyzed the resulting effect that can cause an attack of the types SCN-1 and SCN-2 in one arbitrary node (either a TG-1 or a TG-2) or in a subset of "nodes/2" arbitrary nodes that are either TG-1 or TG-2. The results of the simulations are shown in Table 2, which depicts

Table 2. Changes in $\mathbf{N}_D$ when one or $n/2$ random nodes are targeted.

SCN-1: One Target				
n	$\mathbf{N}_D^{bef}$	$\mathbf{N}_D^{STG-1}$	$\mathbf{N}_D^{STG-2}$	$\mathbf{N}_D^{STG-3}$
100	**92**	=	=	=
1,100	**1,073**	=	=	=
2,100	**2,036**	=	=	=
3,100	**3,000**	=	=	=
SCN-1: n/2 Random Targets				
n	$\mathbf{N}_D^{bef}$	$\mathbf{N}_D^{STG-1}$	$\mathbf{N}_D^{STG-2}$	$\mathbf{N}_D^{STG-3}$
100	**95**	=	=	=
1,100	**1,072**	1,074	=	1,073
2,100	**2,029**	2,034	2,030	2,030
3,100	**3,022**	3,026	=	3,030
SCN-2: One Target				
n	$\mathbf{N}_D^{bef}$	$\mathbf{N}_D^{STG-1}$	$\mathbf{N}_D^{STG-2}$	$\mathbf{N}_D^{STG-3}$
100	**94**	=	=	=
1,100	**1,066**	=	=	=
2,100	**2,010**	=	=	=
3,100	**3,000**	=	=	=
SCN-2: n/2 Random Targets				
n	$\mathbf{N}_D^{bef}$	$\mathbf{N}_D^{STG-1}$	$\mathbf{N}_D^{STG-2}$	$\mathbf{N}_D^{STG-3}$
100	**99**	100	=	100
1,100	**1,049**	1,052	1,051	1,052
2,100	**2,053**	2,059	=	2,056
3,100	**3,013**	3,019	3,014	3,036

the efficiency of the three strategies with regard to the changes caused on the size of $\mathbf{N}_D$ after perturbation. It can be deduced from the table that the variation of the set of driver nodes does not become significant with respect to the number of attacked nodes. In addition, it is important to note that 99% of the observation rate (for U-1 and U-2 nodes) were completely lost for all cases after perturbation. Despite this, we also observed that the networks were equally able to retake 100% of the control after recovery without significant changes in the majority of the cases, and especially for STG-2 partly due to the use of the network diameter.

5. Conclusions

Structural controllability offers a powerful abstraction for understanding the properties of critical nodes in a control network, which is vital to restoring control following node or link failures and, in particular, deliberate attacks.

This helps minimize the period during which a control system is held by an adversary. Also, it helps minimize the period during which the system may reach undesirable states – in the case of electrical power systems and networks, this period can be in the order of seconds or less before severe effects occur.

The main contributions of this paper are the three repair strategies for controllability in control graphs using the structural controllability abstraction, and relying on the PDS formulation to gain a clearer understanding of the effects of topology constraints on the repair strategies. These include re-linking without restrictions, re-linking with constrained network diameter and the use of pre-computed instances of driver nodes. In this way, controllability power-law networks can be restored more efficiently than by re-computing the controlling nodes when their links have been perturbed by attacks on availability. The three strategies have been analyzed formally and subjected to a complexity analysis. The results highlight that the use of a network diameter can be a suitable option to establish control with low computational and storage costs.

Our future work will focus on extending the analysis to explore the possibility of restoring control subgraphs instead of the entire network while retaining acceptable control graph parameters (primarily the number of nodes, maximum out-degree and diameter), thereby improving the respective approaches and their complexity. Another topic involves the renewed study of power-law networks and optimizing approximation mechanisms for controllability that give satisfactory average-time complexity. Finally, our research will also investigate new attack models, especially those involving interactions between attackers and defenders.

Acknowledgements

This research was partially funded by the Marie Curie COFUND Programme U-Mobility supported by the University of Malaga, by the EC FP7 Project under GA No. 246550, and by the Ministerio de Economia y Competitividad (COFUND2013-40259). This research was also partially funded by the EU FP7 ARTEMIS Project under GA No. 269374 and by the Spanish Ministry of Science and Innovation under the ARES Project (CSD2007-00004).

References

[1] C. Alcaraz, E. Miciolino and S. Wolthusen, Structural controllability of networks for non-interactive adversarial vertex removal, *Proceedings of the Eighth International Conference on Critical Information Infrastructures Security*, pp. 129–132, 2013.

[2] C. Alcaraz, E. Miciolino and S. Wolthusen, Multi-round attacks on structural controllability properties for non-complete random graphs, *Proceedings of the Sixteenth Information Security Conference*, 2014.

[3] D. Atkins, T. Haynes and M. Henning, Placing monitoring devices in electric power networks modeled by block graphs, *Ars Combinatorica*, vol. 79, 2006.

[4] A. Barabasi, R. Albert and H. Jeong, Scale-free characteristics of random networks: The topology of the world-wide web, *Physica A: Statistical Mechanics and its Applications*, vol. 281(1-4), pp. 60–77, 2000.

[5] J. Brueni and L. Heath, The PMU placement problem, *SIAM Journal on Discrete Mathematics*, vol. 19(3), pp. 744–761, 2005.

[6] M. Dorfling and M. Henning, A note on power domination in grid graphs, *Discrete Applied Mathematics*, vol. 154(6), pp. 1023–1027, 2006.

[7] Y. Dourisboure and C. Gavoille, Tree-decompositions with bags of small diameter, *Discrete Mathematics*, vol. 307(16), pp. 2008–2029, 2008.

[8] R. Downey and M. Fellows, *Parameterized Complexity*, Springer-Verlag, New York, 1999.

[9] U. Feige, A threshold of $\ln n$ for approximating set cover, *Journal of the ACM*, vol. 45(4), pp. 634–652, 1998.

[10] J. Guo and R. Niedermeier, Exact algorithms and applications for tree-like weighted set cover, *Journal of Discrete Mathematics*, vol. 4(4), pp. 608–622, 2006.

[11] J. Guo, R. Niedermeier and D. Raible, Improved algorithms and complexity results for power domination in graphs, *Algorithmica*, vol. 52(2), pp. 177–202, 2008.

[12] T. Haynes, S. Hedetniemi, S. Hedetniemi and M. Henning, Domination in graphs applied to electric power networks, *SIAM Journal on Discrete Mathematics*, vol. 15(4), pp. 519–529, 2002.

[13] R. Kalman, Mathematical description of linear dynamical systems, *Journal of the Society of Industrial and Applied Mathematics, Series A, Control*, vol. 1(2), pp. 152–192, 1963.

[14] J. Kneis, D. Molle, S. Richter and P. Rossmanith, Parameterized power domination complexity, *Information Processing Letters*, vol. 98(4), pp. 145–149, 2006.

[15] C. Lin, Structual controllability, *IEEE Transactions on Automatic Control*, vol. 19(3), pp. 201–208, 1974.

[16] H. Mayeda, On structural controllability theorem, *IEEE Transactions on Automatic Control*, vol. 26(3), pp. 795–798, 1981.

[17] K. Pai, J. Chang and Y. Wang, Restricted power domination and fault-tolerant power domination on grids, *Discrete Applied Mathematics*, vol. 158(10), pp. 1079–1089, 2010.

[18] C. Palmer and J. Steffan, Generating network topologies that obey power laws, *Proceedings of the IEEE Global Telecommunications Conference*, vol. 1, pp. 434–438, 2000.