Driver-Node based Security Analysis for Network Controllability

Shuo Zhang[1] Stephen D. Wolthusen [2]

***Abstract*— In the study of network controllability, because driver nodes are vulnerable to control hijack and removals, and harmfulness of removing a driver node is still unknown. Therefore, to defend against such attacks, we identify each vertex of all minimum sets of driver nodes firstly. Also, to know the harmfulness of removing a driver node, we classify those identified nodes by impacts of removing a driver node on the minimum set of driver nodes to control the residual network. By the minimum input theorem, given a digraph, these two issues are respectively solved by finding each vertex that is an unmatched node related to a maximum matching, and classifying it by the impact of its removal on the number of unmatched nodes of the residual digraph. As a result, our driver-node identification and classification are executed in more efficient polynomial time than related works.**

I. Introduction

According to the control theory [1], network controllability [2] is one of network properties to ensure that the state of each network vertex could be forced from any given state to a proposed one by inputs in limited steps, where network vertices directly forced by external inputs are called the driver nodes [2]. To get control into directed networks with LTI dynamics, based on the theory of strucutral controllability [3], [4] and a maximum matching of the given digraph, Liu $et\ al.$ [2] can effectively identify a minimum set of driver nodes in order to control the entire network.

Nevertheless, due to the importance of driver nodes in terms of obtaining control into the network with LTI dynamics, driver nodes are easily targeted by vertex removals [5], by which, attackers can disrupt current control and make the residual network out of control [6]. Besides, since each node of all minimum sets of driver nodes can be effectively found by the method of [7] in polynomial time, driver nodes might be also vulnerable to attackers, who can effect driver nodes and thus hijack the control into network [8], which is called the control hijack.

To defend against control hijack and vertex removal through driver nodes, and defend such attacks in advance, we firstly address the problem of more efficiently identifying every vertex of all minimum sets of driver nodes than that of [7]. Then, to further understand the harmfulness caused by removing a single driver node, we also solve the problem of classifying any driver node based on the impact of its removal on the minimum set of driver nodes to control the residual network. Particularly, our classification is different from the driver-node classification of [9], and other related works [6].

In accordance of the minimum input theorem [2], initially, we get one minimum set of driver nodes to control a given LTI-dynamic digraph by finding an arbitrary maximum matching of it. Then, the first problem is solved by finding every vertex, which must be an unmatched node related to a maximum matching of this given digraph. With these identified nodes of the given digraph, the second problem is solved by classifying each of them by the impact of its removal on the unmatched nodes of the residual digraph. We furhter conclude that removing a driver node either reduces the original cardinality of the minimum set of driver nodes by one, or not change, and those remaining driver nodes can still be involved into controlling the residual network. Besides, all operations are executed in a bipartite graph mapped by the given digraph. As a result, the worst-case execution time of our solution except for identifying a maximum matching of the given digraph is linear.

For our contribution, given any LTI dynamic digraph, to defend against control hijack and disruption in advance, vertices of all minimum sets of driver nodes are identified more efficient than the method of [7] in the worst case. And they are efficiently classified by impacts of any single driver-node removal on controlling residual network to enrich the understanding of the harmfulness caused by a single driver-node removal.

The remaining paper is arranged as follows: section II acquires network controllability; section III shows related work on driver nodes; section IV, V identifies and classifies driver nodes; section VI concludes this paper.

II. Acquire Network Controllability

Controllability of networks strictly obeys the control theory [10], [1], and it can be derived in different graph-theoretical methods. This section mainly introduces the maximum-matching based method.

First and the foremost, a linear time-invariant system is expressed by an equation:

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

where $x(t) = (x_1(t), x_2(t), \ldots, x_N(t))^T$, $x(t) \in \mathbb{R}^N$ is the system vector, capturing the state of each system vertex at time t. $u(t) = (u_1(t), u_2(t), \ldots, u_M(t))^T (M \leq N)$, $u(t) \in \mathbb{R}^M$ is the input vector, holding external inputs at time t. Matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ shows the interaction among N system vertices, while input matrix $\mathbf{B} \in \mathbb{R}^{N\times M}$ shows interactions

[1] Shuo Zhang is with School of Mathematics and Information Security, Royal Holloway, University of London, Egham TW20 0EX, UK MYVA375@live.rhul.ac.uk

[2] Stephen D. Wolthusen with School of Mathematics and Information Security, Royal Holloway, University of London, Egham TW20 0EX, UK stephen.wolthusen@rhul.ac.uk

among system vertices and inputs. A system described by equation 1 is completely controllable if and only if the matrix $\mathbf{C} = [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \ldots, \mathbf{A}^{N-1}\mathbf{B}]$ and $\mathbf{C} \in \mathbb{R}^{N \times NM}$, has full rank, noted by $rank(\mathbf{C}) = \mathbf{N}$ and called the controllability rank condition.

However, exact value of non-zero entries of $\mathbf{A}$, $\mathbf{B}$ is difficult to measure [3], and calculating the rank of $\mathbf{C}$ is computationally massive [2]. To avoid these two problems and still design completely controllable system, strucutral controllability of LTI systems [3], [4] was raised later:

Definition 1 (**Structural Controllability** [3]). *A system described by equation 1 is structurally controllable if and only if there is at least one completely controllable system with the same structure as it.*

Here, for a structural controllable system, and a completely controllable system, the same structure between them means that each position of non-zero entries of the input and state matrices are same. Particularly, it turns out that almost all combinations among values of non-zero entries of $\mathbf{A}$ and $\mathbf{B}$ satisfy the controllability rank condition [11], so that a structurally controllable system could be completely controllable with high probability. In the most case, verifying or constructing a structurally controllable system, it just requires a set of disjoint cacti of definition 6 within the digraph of definition 2.

More specifically, such condition of a structurally controllable system are generalized by theorem 1 with following definitions:

Definition 2 (**Digraph G(A, B)**). *Given matrix* $\boldsymbol{A}$, $\boldsymbol{B}$ *of equation 1, let* $G(\boldsymbol{A}, \boldsymbol{B}) = (V_1 \cup V_2, E_1 \cup E_2)$ *be a non-empty system network, and* $\alpha : \{\boldsymbol{A}, \boldsymbol{B}\} \rightarrow G(\boldsymbol{A}, \boldsymbol{B})$ *be a bijection. For each non-zero* $a_{ij} \in \boldsymbol{A}$, $b_{pq} \in \boldsymbol{B}$, *there are* $\alpha : a_{ij} \rightarrow \overrightarrow{\langle v_j, v_i \rangle}$, $\alpha : b_{pq} \rightarrow \overrightarrow{\langle u_q, v_p \rangle}$, *where* $\overrightarrow{\langle v_j, v_i \rangle} \in E_1$, $\overrightarrow{\langle u_q, v_p \rangle} \in E_2$, $\{v_i, v_j, v_p\} \in V_1$ *and* $u_q \in V_2$.

Definition 3 (**Stem & Bud**[3]). *By definition 2, a stem in* $G(\boldsymbol{A}, \boldsymbol{B})$ *is a directed path starting from a node of* V_2. *A bud is a directed cycle plus an arc, whose head is shared with this cycle, and this arc is called the distinguished edge.*

Definition 4 (**Dilation**[3]). *In* $G(\boldsymbol{A}, \boldsymbol{B}) = (V_1 \cup V_2, E_1 \cup E_2)$ *of definition 2,* $S \subseteq V_1$ *is a set of nodes,* $T(S) \subseteq V_1 \cup V_2$ *is set of vertices as tails of the arcs whose heads are in* S. *When* $G(\boldsymbol{A}, \boldsymbol{B})$ *contains a dilation, if and only if* $|S| > |T(S)|$.

Definition 5 (**Inaccessibility** [3]). *In* $G(\boldsymbol{A}, \boldsymbol{B}) = (V_1 \cup V_2, E_1 \cup E_2)$ *of definition 2, a node of* V_1 *that can not be visited through directed paths starting from any node of* V_2 *is inaccessible.*

Definition 6 (**Cactus**[3]). *Let* $B_1, B_2, \ldots, B_l$ *be a set of buds, and let* S_1 *be a stem,* $S_1 \cup B_1 \cup B_2, \ldots, \cup B_l$ *is a cactus if and only if the tail of distinguished edge of* $B_i (1 \leq i \leq l)$ *is not the top node of* S_1 *but the only common node of* $S_1 \cup B_1 \cup B_2, \ldots \cup B_{i-1}$. *Besides, a stem of definition 3 is also a cactus.*

Theorem 1 (**Structural Controllability Theorem** [3], [12]). *Following statements are equivalent:*

1) *System described by equation 1 is structurally controllable.*
2) $G(\boldsymbol{A}, \boldsymbol{B})$ *of definition 2 contains neither inaccessible nodes nor a dilation.*
3) $G(\boldsymbol{A}, \boldsymbol{B})$ *is spanned by a set of disjoint cacti.*

In particular, according to [3], [12], when statement one is satisfied, statement two implied by it can thus imply statement three, while statement three can always imply both statement one and two. It means that the second statement can not be independently used to verify if a system is structurally controllable or not.

To derive the structural control into $G(\mathbf{A}) = (V_1, E_1)$ of definition 2 with a minimum set of driver nodes, Liu $et\ al.$ [2] raised a maximum-matching based method so that complete control into this network with a minimum set of driver nodes can be obtained as well, which is summarized by theorem 3, and definition 7:

Definition 7 (**Driver Node** [2]). *Given matrix* $\boldsymbol{A}$, $\boldsymbol{B}$ *of equation 1, network* $G(\boldsymbol{A}, \boldsymbol{B}) = (V_1 \cup V_2, E_1 \cup E_2)$ *of definition 2. Then, for* $\overrightarrow{\langle u_q, v_p \rangle} \in E_2$, *where* $u_q \in V_2$ *and* $v_p \in V_1$, *a single driver node is* v_p.

Theorem 2 (**Minimal Input Theorem** [2]). *Given a network* $G(\boldsymbol{A}) = (V_1, E_1)$ *of definition 2, the minimum number of driver nodes to fully control* $G(\boldsymbol{A})$ *is one, where any vertex can be a driver node, if it has a perfect matching. Otherwise, it equals the number of unmatched nodes related to a maximum matching, which are directly adjacent to the same number of inputs.*

where, a maximum matching of a digraph is a maximum set of arcs without common tails and heads [13], [14]. The head of an arc involved into a maximum matching is a matched node related to this maximum matching, otherwise, it is unmathed. When all nodes of a digraph are matched, it contains a perfect matching. In bipartite graphs, a maximum matching is a maximum set of disjoint edges, and a node incident to any edge of this maximum matching is matched, otherwise, it is unmatched related to this maximum matching. In general, there are multiple maximum matchings in a same graph. By the algorithm of [15], a maximum matching of a bipartite graph can be identified in $O(\sqrt{N} \cdot L)$ time, where N and L represent the number of vertices and edges of a given bipartite graph, which also ensures that the network controllability via constructing a network with structural controllability can be effectively acquired. And problems about structural controllability can be solved by a graph-theoretical problems related to the maximum matching.

A. Problem formulation

Based on the minimum input theorem above, we assume that each minimum set of driver nodes to control a network with LTI dynamics is derived by identifying a maximum matching. Then, the research problem of this paper is:

Research Question: Given a digraph with LTI dynamics, efficiently identify each single vertex that could be included by a minimum set of driver nodes, and also classify those nodes based on how any single driver node maintains the current minimum set of driver nodes.

III. Related Works

Previous works on driver nodes are wide. Chronologically, based on the minimum input theorem [2], Jia *et al.* [7] firstly classified a vertex into critical, redundant or intermittent categories, if it is always, never or sometimes, included by a mininum set of driver nodes to control a given network. Based on this network-vertex classification, all vertices able to be involved into a minimum set of driver nodes are identified with time complexity of $O(N{\cdot}L)$ in the worst case, where N, L are the number of nodes and arcs of a given digraph. Obviously, critical nodes are the most vulnerable vertices to control hijack and disruption, whereas they can be found by finding vertices without indegrees in linear time [7]. Even though, critical nodes might be protected in advance, while intermittent nodes may thus become new targets of attackers. Therefore, we propose to more efficiently identify each node of all minimum sets of driver nodes than the method of [7], and time complexity of our method is $O(\sqrt{N}\cdot L)$.

Later, control capacity [16] and control backbone [6] were raised to quantify the amount of each single node being the driver node. Because of the huge number of various mimimum sets of driver nodes, and the complex combinations to generate different maximum matchings [17], [18], the exact value for each vertex is almostly impossible to calculate. Thus, they created sampling algorithms respectively. Meanwhile, Ruths *et al.* [9] concluded that any driver node set must contain the nodes without either indegree or out degree. And nodes having both indegree and outdegree can also be a driver node. But there was not a method to identify all nodes able to be a driver node. Recently, Peter, and Cohen *et al.* [19] analysed driver nodes among randomized networks. They find that some nodes are always the driver nodes during the given network under randomization, which does not change the initial graph degree distribution. In [20], authors claimed that they can find all nodes able to be driver nodes in linear time. Nonetheless, they did not prove that the number of those found vertices are maximum. Similar to the classification of Jia *et al.* [7], Commault *et al.* [21] classified nodes for network structural controllability, while each external input is stimulated to be dedicated, which means that each input can be adjacent to only one node. Additionally, since single-vertex removals can dramatically increase the minimum set of driver nodes in cardianlity [5] to control the residual network, and those related works above are all unable to show possible impacts of removing any single driver node on controlling the residual network with a minimum set of driver nodes. Therefore, after finding each node involved into all minimum sets of driver nodes, we are also motivated to classify them by impacts of a single driver-node removal on the minimum set of driver nodes to control the residual network.

IV. Identification of Drive Nodes

By the minimum input theorem [2], the first issue of finding each vertex contained by all minimum sets of drive nodes can be solved via identifying the single nodes, each of which is an unmatched node with respect to a maximum matching of a digraph. Then, we execute this identification process with following graphs:

Definition 8 (**Input Network**). *Let $D = (V, E)$ be a finite digraph, excluding self loops, parallel arcs and isolated nodes, where $V \neq \emptyset$, $V = \{v_i | 1 \leq i \leq N\}(N > 2)$, and $E \neq \emptyset$, $|E| > 2$, $E = \{\overrightarrow{\langle v_i, v_j\rangle} | i \neq j, v_i, v_j \in V\}$.*

Definition 9 (**Bipartite graph B**). *Given $D = (V, E)$, let $B = (V_B, E_B)$ be a bipartite graph, V_B^+ and V_B^- be two disjoint and independent sets of V_B, where $|E_B| = |E|$, $|V_B| = 2|V|$, $|V_B^-| = |V_B^+|$, and $V_B = \{\{v_i^-, v_i^+\} | v_i^- \in V_B^-, v_i^+ \in V_B^+\}$. Besides, let $\beta : V \to V_B$, and $\gamma : E \to E_B$ be two different bijections. For any $v_i \in V$, $\beta : v_i \to \{v_i^+, v_i^-\}$, $v_i^- \in V_B^-, v_i^+ \in V_B^+$; for any $\overrightarrow{\langle v_i, v_j\rangle} \in E$, $\gamma : \overrightarrow{\langle v_i, v_j\rangle} \to (v_i^+, v_j^-)$, where $(v_i^+, v_j^-) \in E_B$, $v_i^- \in V_B^-$ and $v_j^+ \in V_B^+$(See an example of figure 1).*

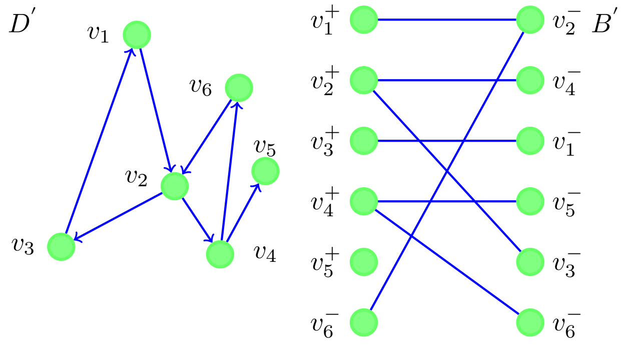

Fig 1. A digraph D' is mapped into a bipartite graph B' by definition 9.

Then, lemma 1 uses B to identify unmatched nodes of D:

Lemma 1. *Given $D = (V, E)$ and $B = (V_B, E_B)$, let $\{v^-, v^+\} \subseteq V_B$ be mapped from $v \in V$. Also, let M_D be a maximum matching of D, M_B be a maximum matching of B, which is mapped from M_D by definition 9. Then, $v \in V$ in D is an unmatched node with respect to M_D, if and only if v^- is an unmatched node in V_B^- with respect to M_B.*

Proof: Sufficiency: If $v \in V$ in D is an unmatched node with respect to M_D, and M_B is mapped by M_D. By definition 9, $\{v^-, v^+\} \subseteq V_B$ is mapped from v, and $v^- \in V_B^-$ can not be incident to any edge of M_B. Thus, v^- is unmatched related to M_B in V_B^-.

Necessity: If v^- is an unmatched node in V_B^- related to M_B. By definition 9, v^- can be only incident to edges out of M_B. Let $(v_i^+, v^-) \notin M_B$ be such an edge. Then, there is $\gamma^{-1} : (v_i^+, v^-) \to \overrightarrow{\langle v_i, v\rangle}$, and $\overrightarrow{\langle v_i, v\rangle} \notin M_D$. Thus, v is unmatched related to M_D in D. □

By lemma 1, finding every unmatched node related to a maximum matching of D can be solved by finding each node of V_B^-, which is unmatched related to a maximum matching of B.

In the following paper, we define $\oplus$ as a symmetric difference between two sets. For instance, let S_1, S_2 be two edge sets, then $S_1 \oplus S_2 = \{S_1 \backslash \{S_1 \cap S_2\}\} \cup \{S_2 \backslash \{S_1 \cap S_2\}\}$. Next, lemma 2 identifies a vertex of V_B^-, which is a matched node with respect to the given maximum matching but is an unmatched node related to another different maximum matching:

Lemma 2. *In $B = (V_B, E_B)$ of definition 9, let M_i be a maximum matching of B. With respect to M_i, let $v_i^- \in V_B^-$ be a matched node, and $v_j^- \in V_B^-$ be an unmatched node. Then, with respect to a maximum matching different from M_i, v_j^- is matched, while v_i^- is unmatched, if and only if v_i^- and v_j^- are connected by an existing path of B, which alternatively involves edges of M_i and $E_B \setminus M_i$.*

Proof: Let M_j be a different maximum matching from M_i and involving an edge incident to v_j^-.

Sufficiency: Related to M_j, if v_i^- is an unmatched node. Because $M_i \oplus M_j$ contains vertex-disjoint paths, or cycles alternatively involving edges of M_i and M_j with even length, v_i^- and v_j^- can be only involved into a path. Otherwise, either v_j^- is matched related to M_i, or v_i^- is matched related to M_j, which is a contradiction. Hence, v_i^- and v_j^- are connected by a path alternatively involving edges of M_i and M_j in B.

Necessity: If v_i^- and v_j^- are connected by a path alternatively involving edges of M_i and $E_B \setminus M_i$. Let P be this path, and let its two terminal nodes be v_i^- and v_j^-. Then, P must contain the same number of edges of M_i and $E \setminus M_i$. Otherwise, v_i^- and v_j^- can not be connected. Thus, a different maximum matching of B can be obtained by: $M_i \oplus P$, and with respect to $M_i \oplus P$, v_i^- is an unmatched node and v_j^- is matched. □

According to lemma 2, let $v_i^+ \in V_B^+$ be a matched node related to a maximum matching, noted by M_i, we can also know how v_i^+ is an unmatched node related to a different maximum matching. Besides, in the following paper, we call matched vertices of $V_B^-(V_B^+)$ related to M_i that are unmatched nodes related to other different maximum matchings, the *extra-unmatched nodes of $V_B^-(V_B^+)$ via M_i*. For example, v_i^- of lemma 2 is an *extra-unmatched node of V_B^- via M_i*.

However, when the number of all *extra-unmatched nodes of V_B^- via M_i* is less than $|M_i|$, for any node of V_B^- that is not an *extra-unmatched node of V_B^- via M_i*, it is indispensable to clarify whether it is an extra-unmatched node of V_B^- via a different maximum matching from M_i according to lemma 2. Otherwise, it is still unknown if all extra-unmatched nodes of V_B^- via M_i and all unmatched nodes of V_B^- with respect to M_i are all single vertices of V_B^-, each of which is an unmatched node related to a maximum matching of B. Hence, we deduce theorem 3 to make a clarification:

Theorem 3. *In $B = (V_B, E_B)$, let M_i, M_j be two different maximum matchings of B, and v_i^- be a matched node related to both M_i and M_j. Also, assume that v_i^- is not an extra-unmatched node of V_B^- via M_i. Then, v_i^- is still not an extra-unmatched node of V_B^- via M_j.*

Proof: We assume that v_i^- is an extra-unmatched node of V_B^- via M_j, and thus define P_j as an existing path, which alternatively involves edges of M_j and $E_B \backslash M_j$ and connects v_i^- with an unmatched node of V_B^- related to M_j. By proving that P_j can not exist, v_i^- can thus not be an extra-unmatched node of V_B^- via M_j, and theorem 3 is correct.

Further, we let $h(P_j)$ be a subset of P_j, where $|P_j| = 2|h(P_j)|$ and $h(P_j) \subseteq M_j$. Then, an edge of $h(P_j)$ is therefore incident to v_i^-. Also, because any edge of M_j can not be disjoint with edges of M_i, otherwise, maximality of M_i is contradicted. Thus, some edges of M_j either construct disjoint paths or cycles with the same number of edges of M_i, while other edges of M_j can be also shared with M_i. According to this fact, we investigate whether $h(P_j)$ exists or not in following five cases.

1) If $h(P_j) \subseteq \{M_i \cap M_j\}$. Then, P_j that contains an edge incident to v_i^-, can alternatively contain edges of M_i and $E_B \setminus M_i$, and v_i^- is thus an extra-unmatched node of V_B^- via M_i by lemma 2, which contradicts to that v_i^- is not an extra-unmatched node of V_B^- via M_i. Therefore, this case is invalid.
2) If $h(P_j) \subseteq \{M_j \setminus \{M_j \cap M_i\}\}$ and $h(P_j)$ only constructs a path with edges of M_i. Obviously, this path can not connect v_i^- with any unmatched node of V_B^- related to M_j. Otherwise, v_i^- is thus an extra-unmatched node of V_B^- via M_i. Let $e_i = (v_k^+, v_i^-)$ be an edge of M_j and $e_i \notin M_i$. Then, this path must alternatively involve edges of M_i and M_j, and connect v_k^+ with an unmatched node of V_B^+ related to M_j. We denote this path as P_i. Also, since $h(P_j) \subseteq P_j$, vertices of P_j are shared with P_i, P_j and P_i can construct an augmenting path [15] with respect to M_j, so that the cardinality of M_j is augmented, which contradicts the maximality of M_j. Thus, P_i can not exist and this case is invalid.
3) If $h(P_j) \subseteq \{M_j \setminus \{M_j \cap M_i\}\}$ and $h(P_j)$ only constructs a cycle with edges of M_i. Because all vertices of this cycle are connected, nodes of this cycle are shared by both P_j and M_i. Thus, there is a path alternatively involving edges of $E_B \setminus M_i$ and M_i, so that v_i^- is an extra-unmatched node of V_B^- via M_i, which is a contradiction, and this case is invalid.
4) If $h(P_j) \subseteq \{M_j \setminus \{M_j \cap M_i\}\}$ and $h(P_j)$ constructs both a path and a cycle with edges of M_i. Referring to the contradiction occurring in cases two or three, this case can not be valid.
5) If $h(P_j)$ contains edges of both $M_i \cap M_j$ and $M_j \setminus \{M_i \cap M_j\}$. For the same reason of impossibility of case four above, or the contradiction of case three, or both of them, this case is also impossible.

Above all, because $h(P_j)$ can not exist, P_j can not exist, and v_i^- is also not an extra-unmatched node of V_B^- based on lemma 2. □

By theorem 3 and lemma 2, algorithm 1 finds all nodes of V_B^-, each of which is unmatched related to a maximum matching of B. In this algorithm, let M_B be a maximum matching of B.

Algorithm 1: Find extra-unmatched nodes via M_B

```
**Input:** $B = (V_B, E_B)$ of definition 9
**Output:** Single vertices of $V_B^-$
1 Identify $M_B$ of $B$ by Hopctoft-Karp algorithm [15];
2 Set the direction of each edge of $M_B$ from $V_B^+$ to
    $V_B^-$; Set the direction of each edge of $E_B \setminus M_B$
    from $V_B^-$ to $V_B^+$;
3 Identify unmatched nodes of $V_B^-$ related to $M_B$;
4 for each unmatched node of $V_B^-$ related to $M_B$ do
5     Run Breath-First search algorithm [22] from it to
        visit matched nodes of $V_B^-$ related to $M_B$ ;
6 return unmatched nodes of $V_B^-$ related to $M_B$; each
    visited node of $V_B^-$;
```

Proof: Initially, identifying M_B of B costs $O(\sqrt{|V_B} \cdot |E_B|)$ steps at most by Hopctoft-Karp algorithm [15]. Then, setting directions of edges of E_B in $O(|E_B|)$ time ensures that matched nodes related to M_B can be visited by unmatched nodes via paths alternatively involving edges of M_B and $E_B \setminus M_B$. Obviously, the visited node of V_B^- is an *extra-unmatched node of V_B^- via M_B* by lemma 2. By running breath-first search algorithm, those nodes can be visited in $O(|E_B| + |V_B|)$ time. Finally, time complexity of finding each unmatched node of V_B^- related to a maximum matching of B is $O(\sqrt{|V_B} \cdot |E_B|)$ in the worst case. □

By the minimum input theorem [2] and this algorithm, because mapping $D = (V, E)$ of definition 8 into $B = (V_B, E_B)$ costs $O(|V| + |E|)$ time, identifying nodes of all minimum sets of driver nodes of $D = (V, E)$ is therefore more efficient than the method of by [7] in the worst-case execution time.

V. Security-aware analysis of Driver nodes

For the second issue of the research question, given a single node of a minimum set of driver nodes, when it is lost due to attack or failure, we analyse how the minimum set of driver nodes is harmed to control the residual network. By the minimum input theorem [2], given an unmatched node related to a maximum matching of D of definition 8, the harmfulness is represented by all impacts of its removal on unmatched nodes with respect to a maximum matching of residual D. Given $B = (V_B, E_B)$ of definition 9, in the following paper, by lemma 1, let $\{v^-, v^+\} \subseteq V_B$ be mapped by v of D, and v^- be unmatched related to a maximum matching of B. Then, all impacts of removing v on unmatched nodes of $D \setminus v$ depend on all impacts of removing $\{v^-, v^+\}$ on the unmatched nodes of V_B^- with respect to a maximum matching of $B \setminus \{v^-, v^+\}$, which is concluded by lemma 3:

Lemma 3. *In B, let M_i be a maximum matching of B, and S be a set of unmatched nodes of V_B^- related to M_i, where $v^- \in S$. Also, when v^+ is matched related to M_i, let $e = (v^+, v_i^-)$ be an edge of M_i, and P_{v^+} be a path to make v^+ be unmatched related to $M_i \oplus P_{v^+}$ by lemma 2. Then, in $B \setminus \{v^-, v^+\}$, one of following cases occurs:*

1) *if v^+ is a matched node related to M_i and $P_{v^+} = \emptyset$. Then, $\{S \setminus v^-\} \cup v_i^-$ is a set of unmatched nodes with respect to the maximum matching $M_i \setminus e$;*
2) *if v^+ is either an unmatched node related to M_i, or v^+ is matched and $P_{v^+} \neq \emptyset$. Then, $S \setminus v^-$ is a set of unmatched nodes with respect to M_i or $M_i \oplus P_{v^+}$.*

Proof: Since v^- is unmatched related to M_i, its removal does not influence M_i. Besides, if v^+ is not an extra unmatched node related to M_i, v^+ is also not an extra-unmatched node related to all maximum matchings of B by theorem 3, and its removal reduces the cardinality of the maximum matching of $B \setminus \{v^-, v^+\}$ by one, and case one holds. Otherwise, if v^+ is either unmatched related to M_i, or an extra-unmatched node of V_B^+ via M_i. Its removal does not influence either M_i or $M_i \oplus P_{v^+}$ by lemma 2, and $S \setminus v^-$ is a set of unmatched nodes related to them. □

Based on lemma 3, we call v of D either *ordinary node* by case one, or the *spare node* by case two. Then, with the minimum input theorem [2], removing any vertex of a minimum set of driver nodes has no need for more number of driver nodes to control the residual network, and those remaining nodes used to control the original network can still be involved into current minimum set of driver nodes. Accordingly, we classify each node unmatched related to a maximum matching of D. Further, in B, once v^- is known as an unmatched node, or an extra-unmatched node of V_B^- via a maximum matching, category of v only depends on $v^+ \in V_B^+$.

Nevertheless, for any pair vertices v^+ and v^- of bipartite graph B that are mapped by v of $D = (V, E)$ of definition 8, when they are two *extra-unmatched nodes* of V_B^- and V_B^+ via the same maximum matching, it is essential to know once v^- is an unmatched node related to a maximum matching, if v^+ is an *extra-unmatched node* of V_B^+ via this maximum matching. Otherwise, there needs one more time to confirm if v^+ is an extra unmatched node of V_B^+ via this maximum matching. For this purpose, lemma 4 is deduced:

Lemma 4. *In $B = (V_B, E_B)$, let M_i be a maximum matching of B, $P_{v_i^-}$, $P_{v_j^+}$ be two paths, and $v_i^- \in V_B^-$, $v_j^+ \in V_B^+$ be two extra-unmatched nodes via M_i due to existence of $P_{v_i^-}$ and $P_{v_j^+}$, respectively. Then, $P_{v_j^+}$ still exists in $M_i \oplus P_{v_i^-}$, and v_j^+ is also an extra-unmatched node of V_B^+ via $M_i \oplus P_{v_i^-}$.*

Proof: Assume that $P_{v_j^+}$ does not exist in $M_i \oplus P_{v_i^-}$. Then, it might be that one or more edges of $P_{v_i^-}$ and $P_{v_j^+}$ are shared, so that $M_i \oplus P_{v_i^-} \oplus P_{v_j^+}$ can not be a maximum matching of B. However, because $(P_{v_j^+} \oplus M_i) \oplus (M_i \oplus P_{v_i^-}) = P_{v_j^+} \oplus P_{v_i^-}$, and $(P_{v_j^+} \oplus M_i) \oplus (M_i \oplus P_{v_i^-})$ results in vertex-disjoint paths or cycles with even length, and shared edges of them can be only contained by M_i. Thus, there can not be common edges for $P_{v_j^+}$ and $P_{v_i^-}$. Otherwise, paths with odd length would exist. Thus, edges of $M_i \cap P_{v_j^+}$ still exist in $M_i \oplus P_{v_i^-}$, and $P_{v_j^+}$ still exist in $M_i \oplus P_{v_i^-}$. □

To execute classification of each unmatched node related to a maximum matching of $D = (V, E)$ of definition 8, by lemma 4, we just need to identify all extra-unmatched nodes of both V_B^- and V_B^+ via a same maximum matching of B once only. Obviously, algorithm 1 can also return all extra-unmatched nodes of V_B^+ via M_B. Then, algorithm 2 executes nodal classification. Here, let $S^- \subseteq V_B^-$, $S^+ \subseteq V_B^+$ be two nodal sets, which involves unmatched nodes related to M_B and extra-unmatched nodes of V_B^- and V_B^+ via M_B returned by algorithm 1, respectively. And let v^- be a vertex of S^-, $v \in V$ be a vertex of D that maps into $\{v^+, v^-\}$ of V_B.

Algorithm 2: Classify unmatched nodes of D

```
Input: $D = (V, E)$ of definition 8, $S^-$ and $S^+$
       returned by algorithm 1
Output: Single vertices of $D$
1 Colour each node of $S^+$ into red;
2 while $S^- \neq \emptyset$ and each $v^- \in S^-$ do
3     Remove $v^-$ from $S^-$;
4     if $v^+$ is red then
5         return $v$ of $D$ is a spare node;
6         else if then
7             return $v$ of $D$ is an ordinary node;
```

Proof: By lemma 1 and 2, each $v^- \in S^-$ is mapped by an unmatched node with respect to a maximum matching of D. Then, any vertex of S^+ is either unmatched related to M_B, or an extra-unmatched node of V_B^+ via M_B, checking if v^+ is red or not can confirm the category of $v \in V$ of D by lemma 4. Thus, this algorithm is correct for each node of S^-. Also, because each chosen node of S^- is removed in line 3, this procedure terminates when $S^- = \emptyset$. For the running time, excluding running algorithm 1 twice to obtain S^- and S^+, colouring nodes of S^- costs $O(|V_B|)$ steps, and each classification for a node of S^- is $O(1)$. The worst-case time complexity of this algorithm is $O(|V_B|)$ for $B = (V_B, E_B)$. □

In summary, given $D = (V, E)$ of definition 8, due to $2|V| = |V_B|$ by definition 9, our second problem of classifying each single vertex that is able to be a driver node by various impacts of its removal on the minimum number of driver nodes to control the residual network can be solved with the time complexity $O(|V| + |E|) + O(\sqrt{|V|} \cdot |E|)$, by running algorithm 1 and 2.

VI. Conclusion

To protect control into networks with LTI dynamics against control hijack and disruption through driver nodes, and further understand the harmfulness of single driver-node removals, we efficiently identify each vertex involved into all minimum sets of driver nodes, and classify them by impacts of its removal on the minimum set of driver nodes to control the residual network. Based on the minimum input theorem [2], by identifying each node that is an unmatched node related to a maximum matching of the given network, and classifying them by impacts of a single removal on the unmatched nodes related to the residual digraph's maximum matching. These problems are solved with the time complexity of finding a maximum matching of a digraph totally. For our future work, we extend this work to classify all network vertices by impacts of any single-node removal on the control into the residual network efficiently.

References

[1] R. Kalman, "On the general theory of control systems," *Automatic Control, IRE Transactions on*, vol. 4, no. 3, pp. 110–110, 1959.

[2] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, no. 7346, pp. 167–173, 2011.

[3] C. T. Lin, "Structural controllability," *Automatic Control, IEEE Transactions on*, vol. 19, no. 3, pp. 201–208, 1974.

[4] R. Shields and J. Pearson, "Structural controllability of multiinput linear systems," *IEEE Transactions on Automatic control*, vol. 21, no. 2, pp. 203–212, 1976.

[5] C.-L. Pu, W.-J. Pei, and A. Michaelson, "Robustness analysis of network controllability," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 18, pp. 4420–4425, 2012.

[6] J. Ding and Y.-Z. Lu, "Control backbone: An index for quantifying a node s importance for the network controllability," *Neurocomputing*, no. 153, pp. 309–318, 2015.

[7] T. Jia, Y.-Y. Liu, E. Csóka, M. Pósfai, J.-J. Slotine, and A.-L. Barabási, "Emergence of bimodality in controlling complex networks," *Nature communications*, vol. 4, 2013.

[8] A. Chapman and M. Mesbahi, "Security and infiltration of networks: A structural controllability and observability perspective," in *Control of Cyber-Physical Systems*. Springer, 2013, pp. 143–160.

[9] J. Ruths and D. Ruths, "Control profiles of complex networks," *Science*, vol. 343, no. 6177, pp. 1373–1376, 2014.

[10] R. E. Kalman, "Mathematical description of linear dynamical systems," *Journal of the Society for Industrial and Applied Mathematics, Series A: Control*, vol. 1, no. 2, pp. 152–192, 1963.

[11] J.-M. Dion, C. Commault, and J. Van Der Woude, "Generic properties and control of linear structured systems: a survey," *Automatica*, vol. 39, no. 7, pp. 1125–1144, 2003.

[12] H. Mayeda, "On structural controllability theorem," *IEEE Transactions on Automatic Control*, vol. 26, no. 3, pp. 795–798, 1981.

[13] G. Chartrand, L. Lesniak, and P. Zhang, *Graphs & digraphs*. CRC Press, 2010.

[14] J. Bang-Jensen and G. Z. Gutin, *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[15] J. E. Hopcroft and R. M. Karp, "An n^5/2 algorithm for maximum matchings in bipartite graphs," *SIAM Journal on computing*, vol. 2, no. 4, pp. 225–231, 1973.

[16] T. Jia and A.-L. Barabási, "Control capacity and a random sampling method in exploring controllability of complex networks," *Scientific reports*, vol. 3, 2013.

[17] L. Zdeborová and M. Mézard, "The number of matchings in random graphs," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2006, no. 05, p. P05003, 2006.

[18] T. Uno, "Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs," in *International Symposium on Algorithms and Computation*. Springer, 1997, pp. 92–101.

[19] S. P. Chin, J. Cohen, A. Albin, M. Hayvanovych, E. Reilly, G. Brown, and J. Harer, "A mathematical analysis of network controllability through driver nodes," *IEEE Transactions on Computational Social Systems*, vol. 4, no. 2, pp. 40–51, 2017.

[20] X. Zhang, J. Han, and W. Zhang, "An efficient algorithm for finding all possible input nodes for controlling complex networks," *Scientific Reports*, vol. 7, no. 1, p. 10677, 2017.

[21] C. Commault and J. Van Der Woude, "A classification of nodes for structural controllability," 2016.

[22] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.