sensors

MDPI

Article

Bi-Layer Shortest-Path Network Interdiction Game for Internet of Things

Jingwen Yan [1,*], Kaiming Xiao [1], Cheng Zhu [1], Jun Wu [2,3], Guoli Yang [1] and Weiming Zhang [1]

1 Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China; kmxiao@nudt.edu.cn (K.X.); zhucheng@nudt.edu.cn (C.Z.); yangguoli@nudt.edu.cn (G.Y.); wmzhang@nudt.edu.cn (W.Z.)
2 International Academic Center of Complex Systems, Beijing Normal University, Zhuhai 519087, China; junwu@nudt.edu.cn
3 Rural Vitalization Research Institute, Changsha University, Changsha 410073, China
* Correspondence: yanjingwen18@nudt.edu.cn

Received: 10 September 2020; Accepted: 19 October 2020; Published: 21 October 2020

Abstract: Network security is a crucial challenge facing Internet-of-Things (IoT) systems worldwide, which leads to serious safety alarms and great economic loss. This paper studies the problem of malicious interdicting network exploitation of IoT systems that are modeled as a bi-layer logical–physical network. In this problem, a virtual attack takes place at the logical layer (the layer of Things), while the physical layer (the layer of Internet) provides concrete support for the attack. In the interdiction problem, the attacker attempts to access a target node on the logical layer with minimal communication cost, but the defender can strategically interdict some key edges on the physical layer given a certain budget of interdiction resources. This setting generalizes the classic single-layer shortest-path network interdiction problem, but brings in nonlinear objective functions, which are notoriously challenging to optimize. We reformulate the model and apply Benders decomposition process to solve this problem. A layer-mapping module is introduced to improve the decomposition algorithm and a random-search process is proposed to accelerate the convergence. Extensive numerical experiments demonstrate the computational efficiency of our methods.

Keywords: network interdiction; Internet of Things; logical–physical networks; shortest path; Benders decomposition

1. Introduction

With the development of information and communication technologies, more and more functional systems have begun to be built based on information networks. Thanks to the high-speed and massive data transmission of the information network, the scale and efficiency of the functional system has been greatly increased. It makes the multi-layer network system represented by Internet of Things (IoT) widely used in various fields such as energy, health care, communication, transportation and manufacturing [1]. The combination of networks of different layers makes the whole system have far more powerful and intelligent functions than ever before. However, the high integration and connectivity of IoT make them more vulnerable to malicious attacks [2–4]. An error or failure of a certain layer may have serious effects on the entire system, and the malicious attacks on the system may be carried out from multiple layers, e.g., the cascading failures of the Italian smart grid on 28 September 2003 [5] and the Stuxnet worm against Iranian nuclear facilities [6]. The great significance and vulnerability of IoT has inspired researchers to pay attention to the security protection of them. Recent literature has carried out relevant research on the security of IoT and other layered network systems from different perspectives [7–14].

On some occasions, we need to interdict the layered network of functional systems to defend against malicious attacks. For example, criminals invade the IoT and occupy some important devices, attempting to issue malicious instructions to destroy key facilities. The network operators hope to delay the criminals' attack by interdicting some links to buy time for the deployment of heavyweight countermeasures [9]. Such malicious attacks usually spread through direct connections in the network, expand the scope of influence, and ultimately achieve functional damage. To characterize the propagation behavior of malicious attacks in layered network systems such as IoT and to study the defense strategy, a suitable network model is necessary. The logical–physical network model (Concept of logical–physical network. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSPHQG_7.2/concept/ha_concepts_physical_logical.html), which provides us a proper approach to modeling IoT and other real layered network systems, is used in this paper. IoT is a huge network which combines functional devices with the Internet [15]. In IoT, there are functional collaboration and dependency relationships among smart devices such as information-sensing devices, information-processing devices, information-fusion devices, and effectors. In the framework of logical–physical network models, the network formed by these devices and relationships describes how the system performs tasks and achieves functions, and we call it the logical-layer network. Before being functionally connected, these devices need to be physically connected through the Internet, i.e., they need to transfer information through the actual communication network, which we call the physical-layer network. The specific realization of system functions, on the logical layer, needs devices to cooperate according to certain organizational rules; on the physical layer, needs devices to transmit information through actual communication pathways. Figure 1 shows how the functional processes in the logical-layer network correspond to real physical communication paths, and the difference from the single-layer case.

A specific system function, which is determined by a starting node s_l and an ending node t_l, may have different implementations on the logical layer, and these logical implementations may correspond to different communication paths. In the sense of optimization, the communication paths are designed to be shortest (i.e., has minimum delay).

How to deploy an effective interdiction strategy when the logical–physical network suffers malicious intrusion is worthy of attention. Appropriate interdiction strategies can earn sufficient time for network operators, and the optimization of interdiction strategies, which we call the logical–physical network shortest-path interdiction (LPNSPI) problem, is exactly what we want to solve in this paper. LPNSPI is a zero-sum game involving two players: an attacker and a defender. The attacker's goal is to find a path which has minimum delay between a particular origin–destination pair in the logical-layer network, and the defender aims to delay the path as much as possible by interdicting connections in the physical-layer network. LPNSPI is an expansion and supplement of the shortest-path network interdiction (SPNI), which is a derivative of the network interdiction problem (NIP), on logical–physical networks. As an important issue in network attack and defense, NIP has applications in various fields, such as the supply-chain network [16], infrastructure protection [17], interdicting nuclear proliferation [18], etc.

NIP is also a useful model for researching the protection of IoT, e.g., the defensive resource allocation [15] and adversarial outbreak detection [19]. Numerous variations of NIP have been proposed and studied to meet different hypotheses under particular scenarios [16,17,20–24]. Among the many types of NIP, SPNI is a classic and important branch. Generally, SPNI can be described as *maximizing the shortest path* and may be formulated as a bi-level mixed-integer program. Researchers have applied the SPNI model to transportation [25,26], port security, border patrol and other aspects [27]. However, most of the literature about SPNI focuses on the single-layer network where interdiction and pathfinding happen in the same layer [28,29]. In LPNSPI, interdiction is applied on the physical layer, but the path that the attacker wants to minimize is on the logical layer. With the physical-layer network alone, it is impossible to judge whether a path meets the functional requirements; with the logical-layer network alone, it is impossible to confirm the specific delay of

links in the network and the effect of interdiction. The separation of these two layers brings about the non-linearity of the objective function (In Section 2, we present the non-linearity in LPNSPI. The two-layer structure leads to the product term of decision variables in the objective function (Equation (7))), and also make the traditional single-layer NIP solving methods unable to be directly applied to LPNSPI. Recently the layered network interdiction problem has attracted more attention. Kennedy [30] studied the maximum flow interdiction problem in a kind of multi-layer network where different layers are connected by sharing some common elements (nodes or edges). Wei et al. [31] studied the shortest-path interdiction problem in a kind of bi-layer network where the interdiction effects on one layer can be determined by the interdiction on the other layer through logical operations. Baycik et al. [32] studied the interdiction problem in layered physical and information flow networks. In this problem, a physical node can only be used when the flow passing through its corresponding information-layer node is more than a particular demand. Compared to this research, LPNSPI has a completely different setting on the inter-layer relationships of logical–physical networks, which pays more attention to the relationships between logical functions and physical communication paths. Table 1 briefly compares some of the characteristics of LPNSPI and layered network interdiction problems mentioned above.

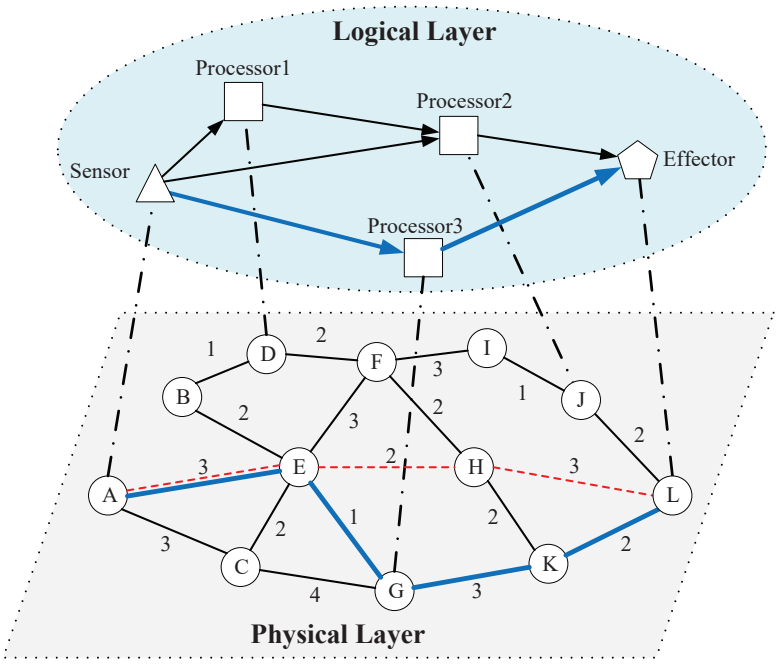


Figure 1. A simple example of the logical–physical network in IoT. The logical layer contains a sensor, an effector and three different processors. The physical layer is the communication network, where the time delay of each link is given by the number beside it. Each dotted line between the two layers connects the functional part and the communicating part of the same entity. The sensor collects information and sends it to either of the processors. The processor analyzes the information and then sends order to the effector. As the figure shows, the shortest logical flow *Sensor* $\rightarrow$ *Processor3* $\rightarrow$ *Effector* corresponds to a physical path *A(Sensor)* $\rightarrow$ *E* $\rightarrow$ *G(Processor 3)* $\rightarrow$ *K* $\rightarrow$ *L(Effector)*, which weighs 9. Although the shortest *A*–*L* path on the physical layer is *A(Sensor)* $\rightarrow$ *E* $\rightarrow$ *H* $\rightarrow$ *L(Effector)* with total weight 8, it is not functionally feasible because no processor is on this path and thus no effective order can be sent to Effector.

In this paper, we propose the LPNSPI game and model it as a bi-level integer program. We reformulate the problem and develop a Benders decomposition algorithm framework to solve LPNSPI. There are two major approaches to solve SPNI problems: the decomposition method and the

dual method. The advantages of the decomposition approach have been mentioned in [33], and as shown in Table 1, the dual method is not suitable for LPNSPI. Then we propose a Layer-Mapping module to reduce unnecessary calculations of the decomposition algorithm. Also, a Random-Search module is developed to accelerate the convergence of the algorithm with a given approximate ratio. Simulation experiments are designed, and the computational results prove the significant efficiency of Layer-Mapping and Random-Search. Finally, we test our algorithms in a real bi-layer IoT network, and our improving methods perform well in both solving time and interdiction effects.

Table 1. Comparisons with LPNSPI and other layered network interdiction problems.

	Kennedy [30]	**Wei et al. [31]**	**Baycik et al. [32]**	**LPNSPI**
objective function	Minimize the maximum flow	Maximizing the shortest path	Minimize the maximum flow	Maximizing the shortest path
inter-layer relationship	geographical co-located relations	logic constraints	flow requirement constraints	path constraints
duality	Yes	Yes	Yes	No

The paper is organized as follows. In Section 2, the LPNSPI model is defined and formulated. In Section 3, a basic decomposition algorithm is developed after reformulation. Layer-Mapping is introduced in Section 4, and Random-Search is proposed in Section 5. Section 6 provides the experimental results. Conclusions are in Section 7.

2. Shortest Path Interdiction Problem in Logical–Physical Networks

The LPNSPI problem involves two different networks: the logical-layer network which represents the flow of information between logical entities, and the physical-layer network which represents the actual transmission path of information in a physical environment. Throughout the present work, we use normal symbols to represent scalars, and bold symbols for vectors/matrices. The logical-layer network is defined as a directed graph $G_l = (N_l, A_l)$, where N_l represents the set of logical nodes and A_l represents the set of logical arcs. Each logical node corresponds to a logical entity such as a person, a unit, an organization, etc. Each logical arc $e_l = (i_l, j_l)$ represents an allowed information transfer direction between logical nodes. The direction of a logical link is generally defined by artificial rules (such as process rules, hierarchy rules, etc.) rather than natural ones. For instance, in the logical network, two nodes are adjacent because they may have a direct functional dependency (the proper functioning of node i depends on the processed information provided by node j) rather than because they have a direct physical connection. Unlike a logical-layer network, a physical-layer network is defined as a bidirected graph $G_p = (N_p, A_p)$, where N_p represents the set of physical nodes and A_p represents the set of physical arcs. Physical arcs are bidirected because they correspond to connections in the actual physical environment, such as road connections, routing connection, and so on. These connections are not directional in themselves and the flow of information on them can be two-way. Each logical node i_l has a corresponding physical node i_p, but not necessarily the other way around (many nodes in the physical network serve only as information transfer nodes and are not necessary to perform system functions). Each physical arc $e_p = (i_p, j_p) \in A_p$ has a cost of communication c_{e_p} for the attacker, which will be increased to $c_{e_p} + d_{e_p}$ if the arc is interdicted by the defender. In addition, for the defender, the corresponding resource consumption of interdicting e_p is denoted as r_{e_p}. The total interdiction resource for the defender is R. The communication cost of a logical arc w_{e_l} is the total communication cost of a path that the attacker choose to travel in the physical-layer network, the corresponding node of endpoints of the logical arc being the start node and end node of the physical path.

In this problem, we assume both the attacker and the defender have complete information about the network. The defender pre-deploys the defense strategy according to the attacker's source node

s_l and target node t_l, and blocks some edges in the network. Subsequently, the attacker develops an optimal attack plan to minimize the communication cost from the starting node to the target node. Let x_{e_p} (the vector form is denoted by bold $\boldsymbol{x}$) denote the defender's interdiction strategy on e_p, and let y_{e_l} (the vector form is denoted by bold $\boldsymbol{y}$) denote the attacker's pathfinding variable on the logical layer. Then the attacker's problem can be formulated as follows:

$$\min_{\boldsymbol{y}} \quad \sum_{e_l \in A_l} w_{e_l} y_{e_l} \tag{1}$$

$$\text{s.t.} \quad \sum_{e_l \in FS(v_l)} y_{e_l} - \sum_{e_l \in RS(v_l)} y_{e_l} = \begin{cases} 1 & if\ v_l = s_l \\ -1 & if\ v_l = t_l \\ 0 & else \end{cases}, \forall v_l \in N_l \tag{2}$$

$$y_{e_l} \in \{0,1\},\ \forall e_l \in A_l \tag{3}$$

where w_{e_l} is the weight of the logical link $e_l = (i_l, j_l)$, i.e., the minimum total cost of its corresponding paths on the physical layer:

$$w_{e_l} = w_{(i_l,j_l)} = \quad \min_{\boldsymbol{k}} \quad \sum_{e_p \in A_p} (c_{e_p} + x_{e_p} d_{e_p}) k_{e_l e_p} \tag{4}$$

$$\text{s.t.} \quad \sum_{e_p \in FS(v_p)} k_{e_l e_p} - \sum_{e_p \in RS(v_p)} k_{e_l e_p} = \begin{cases} 1 & if\ v_p = i_p \\ -1 & if\ v_p = j_p \\ 0 & else \end{cases}, \ \forall v_p \in N_p \tag{5}$$

$$k_{e_l e_p} \in \{0,1\},\ \forall e_p \in A_p \tag{6}$$

where $k_{e_l e_p}$ (Let $\boldsymbol{K}$ denotes the matrix form) is the pathfinding variable which indicates whether e_p is chosen in the corresponding physical path of e_l. $x_{e_p} = 1$ when the physical arc e_p is interdicted and $x_{e_p} = 0$ otherwise. $k_{e_l e_p} = 1$ indicates that e_p is in the physical shortest path corresponding to e_l. $FS(v_p)$ and $RS(v_p)$ represent respectively the arc set directed out of and into node v_p. i_p and j_p are respectively the corresponding physical nodes of i_l and j_l. Constraint (2) and (5) are the flow-balance constraints. In practice, w_{e_l} can be calculated by using common shortest-path algorithms such as the Dijkstra algorithm.

We define $V_{s_l t_l}$ as the total communication cost from s_l to t_l. Then the defender's problem of maximizing $V_{s_l t_l}$, which is exactly the LPNSPI problem, can be formulated as follows

$$\text{[LPNSPI]} \quad V^*_{s_l t_l} = \max_{\boldsymbol{x}} \min_{\boldsymbol{y}} V_{s_l t_l} = \max_{\boldsymbol{x}} \min_{\boldsymbol{y}} \sum_{e_l \in A_l} w_{e_l}(\boldsymbol{x}) y_{e_l}$$

$$= \max_{\boldsymbol{x}} \min_{\boldsymbol{y},\boldsymbol{k}} \sum_{e_l \in A_l} \left[\sum_{e_p \in A_p} (c_{e_p} + x_{e_p} d_{e_p}) k_{e_l e_p} \right] y_{e_l} \tag{7}$$

$$\text{s.t.} \quad \sum_{e_l \in FS(v_l)} y_{e_l} - \sum_{e_l \in RS(v_l)} y_{e_l} = \begin{cases} 1 & if\ v_l = s_l \\ -1 & if\ v_l = t_l \\ 0 & else \end{cases}, \forall v_l \in N_l \tag{8}$$

$$\sum_{e_p \in FS(v_p)} k_{e_l e_p} - \sum_{e_p \in RS(v_p)} k_{e_l e_p} = \begin{cases} 1 & if\ v_p = i_p \\ -1 & if\ v_p = j_p \\ 0 & else \end{cases}, \ \forall v_p \in N_p, \forall e_l = (i_l, j_l) \in A_l \tag{9}$$

$$x_{e_p} = x_{\overleftarrow{e_p}},\ \forall e_p \in A_p \tag{10}$$

$$\sum_{e_p \in A_p} x_{e_p} r_{e_p} \leq 2R \tag{11}$$

$$x_{e_p}, y_{e_l}, k_{e_l e_p} \in \{0,1\},\ \forall e_p \in A_p,\ \forall e_l \in A_l \tag{12}$$

where $\overleftarrow{e_p}$ is the reverse arc of e_p and constraint (10) indicates that the interdiction of an arc is effective for both directions. Constraint (11) is the resource constraint for the defender, where we use $2R$ as the resource limit because of the counting for both directions. It is noted that the objective function (7) is nonlinear, which results from the mapping relationship between the physical-layer network and the logical-layer network. A Benders decomposition algorithm framework and related improvement methods are proposed in this paper.

3. Basic Decomposition Algorithm for LPNSPI

The problem of shortest-path network interdiction can be naturally divided into two processes: blocking resource deployment process and pathfinding process. These two processes respectively correspond to the max operation and min operation in (7), and correspond to the master problem and subproblem of the Benders decomposition algorithm. Let $\boldsymbol{d}$ denote the vector of d_{e_p} and $\boldsymbol{D} = diag(\boldsymbol{d})$. $\hat{\boldsymbol{z}}$ denotes an s_p–t_p path on the physical layer and $\hat{Z}$ denotes a collection of s_p–t_p paths. The master problem and the subproblem of LPNSPI are defined as follows:

$$[\text{Master}(\hat{Z})] \qquad V_{\hat{Z}} = \max_{\boldsymbol{x}} V \tag{13}$$

$$\text{s.t.} \quad V \leq \boldsymbol{c}^T \hat{\boldsymbol{z}} + \boldsymbol{x}^T \boldsymbol{D} \hat{\boldsymbol{z}}, \ \ \forall \hat{\boldsymbol{z}} \in \hat{Z} \tag{14}$$

$$\text{Constraint (10) and Constraint (11)}$$

$$x_{e_p} \in \{0,1\}, \ \ \forall e_p \in A_p \tag{15}$$

$$[\text{Sub}(\hat{\boldsymbol{x}})] \qquad V_{\hat{\boldsymbol{x}}} = \min_{\boldsymbol{z}} \sum_{e_p \in A_p} (c_{e_p} + \hat{x}_{e_p} d_{e_p}) z_{e_p} \tag{16}$$

$$\text{s.t.} \quad \text{Constraint (8) and Constraint (9)}$$

$$z_{e_p} = \sum_{e_l \in A_l} y_{e_l} k_{e_l e_p}, \ \ \forall e_l \in A_l \tag{17}$$

$$y_{e_l}, k_{e_l e_p} \in \{0,1\}, \ \ \forall e_p \in A_p, \ \forall e_l \in A_l \tag{18}$$

In contrast to the case in a single-layer network, $\hat{\boldsymbol{z}}$ does not necessarily represent a simple path. For the attacker, searching a path with minimum communication cost will lead to the shortest path in the logical layer, which is definitely a simple path. Although the corresponding path in physical and may have repeated arcs, the attacker cannot avoid going through them because the topology of logical layer specifies the process that the attacker must follow to achieve his goal.

Let Z denote the set of shortest physical layer paths that all simple s_l-t_l paths in logical layer correspond. Notice that $\hat{\boldsymbol{z}} \in Z$ is always established, then [Master($\hat{Z}$)] is an equivalent formulation of [LPNSPI] when $\hat{Z} = Z$. Benders decomposition algorithm fixes $\hat{Z}$ and $\hat{\boldsymbol{x}}$ in turn, and iteratively solves the master problem and subproblem in turn. [Master($\hat{Z}$)] fixes the set of feasible paths $\hat{Z}$ and solves an optimal interdiction strategy from the aspect of the defender, while [Sub($\hat{\boldsymbol{x}}$)] gives an optimal path selection with fixed logical-layer network status, standing at the angle of the attacker.

Israeli and Wood [33] proposed two types of "supervalid inequalities" (SVI) constraints to strengthen the LP relation of the master problem of Benders decomposition for the shortest-path interdiction problem of single-layer networks. These inequalities are constructed after the subproblem gives a currently optimal path and added to the master problem as constraints. SVIs are based on the following idea: they may make some solutions infeasible but are guaranteed not to eliminate any optimal solutions unless the incumbent is itself optimal; by reducing the size of the feasible region, SVIs accelerate the master problem. These inequalities can be extended to the logical–physical networks, and the proofs are basically the same with the single-layer case. The master problem containing SVIs is as follows:

$$\text{[Master}(\hat{Z})\text{-SVI]} \qquad V_{\hat{Z}} = \max_{\boldsymbol{x}} V \tag{19}$$

$$\text{s.t.} \qquad V \le \boldsymbol{c}^T\hat{\boldsymbol{z}} + \boldsymbol{x}^T\boldsymbol{D}\hat{\boldsymbol{z}}, \ \ \forall \hat{\boldsymbol{z}} \in \hat{Z} \tag{20}$$

$$\hat{\boldsymbol{z}}^T\boldsymbol{x} \ge p+1, \ \ \forall \hat{\boldsymbol{z}} \in \hat{Z} \tag{21}$$

$$\tilde{\boldsymbol{z}}^T\boldsymbol{x} \ge 1, \ \ \forall \hat{\boldsymbol{z}} \in \hat{Z} \tag{22}$$

$$\text{Constraint (10) and Constraint (11)}$$

$$x_{e_p} \in \{0,1\}, \ \ \forall e_p \in A_p \tag{23}$$

where $\boldsymbol{c}^T\hat{\boldsymbol{z}} + \sum_{h=1}^{p+1} d_{m_h}\hat{z}_{m_h} > \underline{V} \ge \boldsymbol{c}^T\hat{\boldsymbol{z}} + \sum_{h=1}^{p} d_{m_h}\hat{z}_{m_h}$ for $p \le H$ and the sequence $\{d_{m_h}\hat{z}_{m_h}\}$. $\{d_{m_h}\hat{z}_{m_h}\}$ is the descending order of the sequence $\{d_h\hat{z}_h\}$ (the number of terms is H). $\tilde{\boldsymbol{z}} = (diag(\boldsymbol{1} - \hat{\boldsymbol{x}}))\hat{\boldsymbol{z}}$. Constraint (21) is the Type-I SVI and Constraint (22) is the Type-II SVI. For detailed introduction and related proofs of SVI, please refer to [33].

Because of the logical–physical structure, [Sub($\hat{\boldsymbol{x}}$)] contains nonlinear terms and cannot be solved directly. However, we can divide the solution process into two steps: first, calculate the current communication cost $\hat{\boldsymbol{w}}$ of the logical layer links with the current interdiction strategy $\hat{\boldsymbol{x}}$, and then calculate the shortest path of the logical layer. Here we rewrite the subproblem as follows:

$$\text{[Sub}(\hat{\boldsymbol{w}})\text{-LM]} \qquad V_{\hat{\boldsymbol{w}}} = \min_{\boldsymbol{y}} \sum_{e_l \in A_l} \hat{w}_{e_l} y_{e_l} \tag{24}$$

$$\text{s.t.} \quad \text{Constraint (2) and Constraint (3)}$$

We denote the matrix $\boldsymbol{K}$ obtained when calculating $\hat{\boldsymbol{w}}$ as $\hat{\boldsymbol{K}}$, and denote the current logical path given by [Sub($\hat{\boldsymbol{w}}$)-LM] as $\hat{\boldsymbol{y}}$, then we have $\hat{\boldsymbol{z}} = \hat{\boldsymbol{K}}^T\hat{\boldsymbol{y}}$. [Sub($\hat{\boldsymbol{w}}$)-LM] is also a necessary reformulation of the subproblem in order to use Layer-Mapping, which we will introduce in the next section. Now we give the basic decomposition algorithm for LPNSPI:

Algorithm 1 Basic Benders decomposition algorithm for LPNSPI

Input : An instance of LPNSPI
Output: An optimal interdiction plan $\boldsymbol{x}^*$

1: $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{0}; \hat{Z} \leftarrow \emptyset; \underline{V} \leftarrow -\infty; \overline{V} \leftarrow \infty$
2: **while** $\overline{V} - \underline{V} > 0$ **do**
3: Calculate $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$ according to $\hat{\boldsymbol{x}}$ using (4)-(6)
4: Solve [Sub($\hat{\boldsymbol{w}}$)-LM] for $\hat{\boldsymbol{y}}$ and the objective value $V_{\hat{\boldsymbol{w}}}$
5: $\hat{\boldsymbol{z}} = \hat{\boldsymbol{K}}^T\hat{\boldsymbol{y}}; \hat{Z} \leftarrow \hat{Z} \cup \hat{\boldsymbol{z}}$;
6: **if** $\underline{V} < V_{\hat{\boldsymbol{w}}}$ **then**
7: $\boldsymbol{x}' \leftarrow \hat{\boldsymbol{x}}; \underline{V} \leftarrow V_{\hat{\boldsymbol{w}}}$;
8: **end if**
9: **if** $\overline{V} - \underline{V} \le 0$ **then**
10: **break**;
11: **end if**
12: Solve [Master($\hat{Z}$)-SVI] for $\hat{\boldsymbol{x}}$ and the objective value $V_{\hat{Z}}$;
13: $\overline{V} \leftarrow V_{\hat{Z}}$;
14: **end while**
15: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}'$
16: **return** $\boldsymbol{x}^*$

The correctness of Algorithm 1 is based on the following facts: $V_{\hat{\boldsymbol{w}}}$ gives a lower bound on the attacker's optimal objective value and $V_{\hat{Z}}$ gives an upper bound on the defender's optimal objective value. Although the actual path in the physical layer $\boldsymbol{z}$ may no longer be a simple path in the case of

layered network, the corresponding path in logical layer $\boldsymbol{y}$ is certainly simple, whose number is finite. The number of possible interdiction plans $\boldsymbol{x}$ is also finite. In addition, once $\boldsymbol{y}$ and $\boldsymbol{x}$ are fixed, $\boldsymbol{z}$ is fixed, which means that the algorithm converges in a finite number of iterations.

4. Layer-Mapping Module

As shown in Algorithm 1, during the iteration of the decomposition algorithm, the current interdiction strategy $\hat{\boldsymbol{x}}$ changes when the master problem finds a better solution. In addition, the change of the interdiction strategy results in the change of the network status which can be represented by $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$: $\hat{\boldsymbol{w}}$ indicates the weights of logical links and $\hat{\boldsymbol{K}}$ indicates the corresponding relationships between logical links and physical paths. Once the current network status ($\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$) is determined, we can solve a relatively simple subproblem [Sub($\hat{\boldsymbol{w}}$)-LM] ([Sub($\hat{\boldsymbol{w}}$)-LM] shares the same representation form with the subproblem of the single-layer case) and give the currently optimal logical path $\hat{\boldsymbol{y}}$. However, this process (line 3 in Algorithm 1) contains a lot of double counting during the iteration because we recalculate $\hat{w}_{e_l}$ and $\hat{k}_{e_l e_p}$ for every logical link and physical arc in each iteration. It inspires us to add a Layer-Mapping module between the master problem and the subproblem to find the specific changes of the network status when $\hat{\boldsymbol{x}}$ changes.

Since the interdiction impact d_{e_p} is positive, interdicting a physical arc will not shorten any physical shortest path. So, the change of $\hat{\boldsymbol{x}}$ has no effect on a physical shortest path (that means this path is still shortest) if no arc of it is interdicted. Let $\boldsymbol{x^0}$ denote an initial interdiction strategy. Let $\boldsymbol{w^0}$ denote the corresponding weights (communication costs) of logical links and let $\boldsymbol{K^0}$ denote the corresponding path-mapping matrix $\boldsymbol{K}$. When the interdiction strategy changes to $\hat{\boldsymbol{x}}$ which satisfies $\{e_p|x^0_{e_p}=1\} \subseteq \{e_p|\hat{x}_{e_p}=1\}$, we need only to recalculate $\hat{w}_{e_l}$ and $\hat{k}_{e_l e_p}$ for logical link e_l if there exists a physical arc $e_p \in \left\{e_p \in A_p | x^0_{e_p} \neq \hat{x}_{e_p}\right\}$ which makes $K^0_{e_l e_p}=1$. In practice, we set $\boldsymbol{x^0}$ to $\mathbf{0}$, and thus $\boldsymbol{K^0}$ indicates the initial network status when there is no interdiction. When the resource limit makes the number of physical arcs that can be interdicted simultaneously much smaller than the edge number of the physical network, layer-mapping will significantly reduce the weight calculations for logical links.

Algorithm 2 Layer-Mapping

Input: Initial interdiction strategy $\boldsymbol{x^0}$; initial network status $\boldsymbol{w^0}$ and $\boldsymbol{K^0}$; the new interdiction strategy $\hat{\boldsymbol{x}}$
Output: The new network status $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$;

1: Initialize $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$
2: $\hat{Diff} = \left\{e_p \in A_p | {x^0}_{e_p} \neq \hat{x}_{e_p}\right\}$
3: **for** $e_l \in A_l$ **do**
4: flag $\leftarrow 0$;
5: **for** $e_p \in \hat{Diff}$ **do**
6: **if** $K^0_{e_l e_p}=1$ **then**
7: Calculate $\hat{w}_{e_l}$ and $\hat{\boldsymbol{K}}_{e_l}$ using (4)-(6);
8: flag $\leftarrow 1$;
9: **break**;
10: **end if**
11: **end for**
12: **if** flag = 0 **then**
13: $\hat{w}_{e_l} \leftarrow w0_{e_l}, \hat{\boldsymbol{K}}_{e_l} \leftarrow {\boldsymbol{K^0}}_{e_l}$
14: **end if**
15: **end for**
16: **return** $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$

$\hat{\boldsymbol{K}}_{e_l}$ in Layer-Mapping is the vector of $\hat{k}_{e_l e_p}$ when e_l is fixed (it can also be explained as the e_lth row vector of matrix $\hat{\boldsymbol{K}}$), and ${\boldsymbol{K^0}}_{e_l}$ shares the same representation. Layer-Mapping intuitively

shows the change of the mapping status of the logical–physical network when interdiction happens, which not only avoids the non-linearity of the objective function from explicitly appearing in the decomposition algorithm, but also speeds up the solution of the subproblem. For simplicity, we call this module Layer-Mapping($\boldsymbol{x^0}$,$\hat{\boldsymbol{x}}$), where $\boldsymbol{x^0}$ is the initial interdiction strategy and $\hat{\boldsymbol{x}}$ is the new one. Applying Layer-Mapping to the basic decomposition algorithm, we get Algorithm 3.

Algorithm 3 Improved decomposition algorithm with Layer-Mapping

Input: An instance of LPNSPI
Output: An optimal interdiction plan $\boldsymbol{x}^*$

$\hat{\boldsymbol{x}} \leftarrow \mathbf{0}$; $\hat{Z} \leftarrow \emptyset$; $\underline{V} \leftarrow -\infty$; $\overline{V} \leftarrow \infty$
while $\overline{V} - \underline{V} > 0$ **do**
  if $\hat{\boldsymbol{x}} = \mathbf{0}$ **then**
    $\boldsymbol{x^0} \leftarrow \hat{\boldsymbol{x}}$; Calculate $\boldsymbol{w^0}$ and $\boldsymbol{K^0}$ using (4)-(6);
    $\hat{\boldsymbol{w}} \leftarrow \boldsymbol{w^0}$;
  else
    Solve **Layer-Mapping($\boldsymbol{x^0}$,$\hat{\boldsymbol{x}}$)** for $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$;
  end if
  Solve [Sub($\hat{\boldsymbol{w}}$)-LM] for $\hat{\boldsymbol{y}}$ and the objective value $V_{\hat{\boldsymbol{w}}}$
  $\hat{\boldsymbol{z}} = \hat{K}^T\hat{\boldsymbol{y}}$; $\hat{Z} \leftarrow \hat{Z} \cup \hat{\boldsymbol{z}}$;
  if $\underline{V} < V_{\hat{\boldsymbol{w}}}$ **then**
    $\boldsymbol{x}' \leftarrow \hat{\boldsymbol{x}}$; $\underline{V} \leftarrow V_{\hat{\boldsymbol{w}}}$;
  end if
  if $\overline{V} - \underline{V} \leq 0$ **then**
    break;
  end if
  if [Master($\hat{Z}$)-SVI] is feasible **then**
    Solve [Master($\hat{Z}$)-SVI] for $\hat{\boldsymbol{x}}$ and the objective value $V_{\hat{Z}}$;
    $\overline{V} \leftarrow V_{\hat{Z}}$;
  else
    break;
  end if
end while
$\boldsymbol{x}^* \leftarrow \boldsymbol{x}'$
return $\boldsymbol{x}^*$

5. A Random-Search Method for Accelerating Convergence

To accelerate the convergence speed of the decomposition algorithm, we try to use the information of $\hat{\boldsymbol{z}}$ obtained by the subproblem as much as possible in each iteration to limit the feasible domain of the master problem. We propose a Random-Search procedure to increase the number of paths added to $\hat{Z}$ for each iteration, which shares the basic idea of Local-Search proposed by Wood [33] in the NIP of a single-layer network. We hope to find more near-optimal paths in one iteration. However, the existence of inter-layer relationships in multi-layer networks makes the search for near-optimal paths complicated, and the time cost of finding all near-optimal paths in each iteration, which is what Local-Search does, is not small in the face of the large-scale layered network. The process of Random-Search is described as follows. The total communication consumption of these paths is limited by a set constant λ and the current lower bound $\underline{V}$. Let $\hat{\boldsymbol{z}}_\lambda$ denote an near-optimal path found by Random-Search and let $\hat{V}_\lambda$ denote the communication cost of this path with an interdiction plan $\hat{\boldsymbol{x}}_\lambda$, of which the initial value is set to $\hat{\boldsymbol{x}}$. For a path $\hat{\boldsymbol{z}}$ found by the subproblem, Random-Search first selects one of its edges (denoted as $e_{p_{a1}}$) at random and interdict it, i.e., set $\hat{x}_{\lambda_{e_{p_{a1}}}} = 1$ and provisionally set $d_{e_{p_{a1}}} = \infty$. Then, for each edge e_l in logical layer, recalculate the weight $\hat{w}_{e_l}$ of it if $\hat{k}_{e_l e_{p_{a1}}} = \hat{K}_{e_l,e_{p_{a1}}} = 1$.

As for the edges of the logical layer that satisfy $\hat{K}_{e_l,e_{p_{a1}}} = 0$, their weights will not change. $e_{p_{a1}}$ does not appear in the physical paths they correspond, so the blocking of $e_{p_{a1}}$ will not change the mapping relationship between them and these paths. Please note that the process of recalculating $\hat{w}_{e_l}$ can be represented by Layer-Mapping($\hat{\boldsymbol{x}}$,$\hat{\boldsymbol{x}}_\lambda$). The shortest path calculated based on the updated edge weights is a near-optimal path. If the total weight of the path newly found is no more than $\lambda\underline{V}$, the path will be added into $\hat{Z}$ as $\hat{\boldsymbol{z}}_\lambda$. After that, Random-Search will choose a new edge $e_{p_{a2}}$ of the path $\hat{\boldsymbol{z}}_\lambda$ to interdict and repeat the searching process. The process ends when the blocking of edge $e_{p_{an}}$ leads $\hat{V}_\lambda > \lambda\underline{V}$. Figure 2 shows the execution of Random-Search.

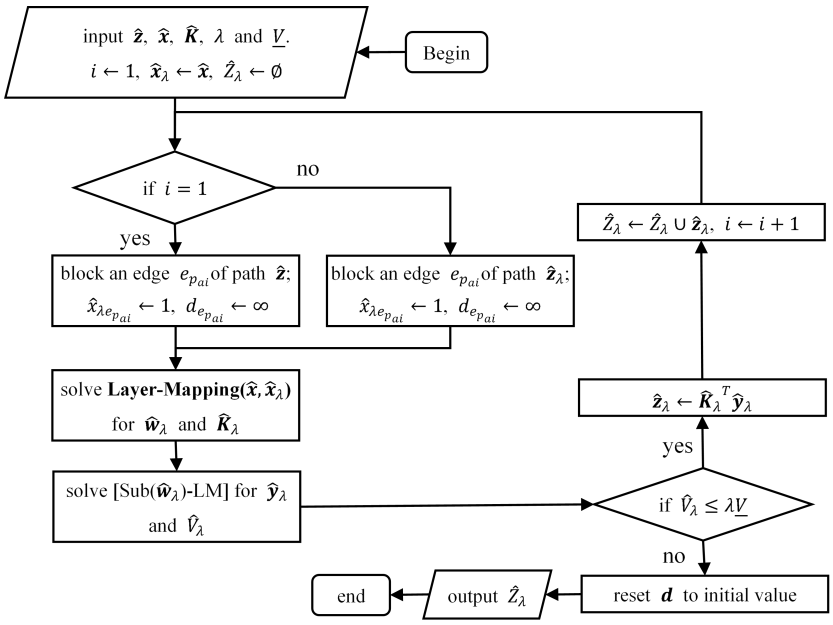


Figure 2. The process of Random-Search.

As $\hat{\boldsymbol{z}}_\lambda \in Z$ is naturally established, adding constraint $V \le \boldsymbol{c}^T\hat{\boldsymbol{z}}_\lambda + \boldsymbol{x}^T\boldsymbol{D}\hat{\boldsymbol{z}}_\lambda$ to the master problem will not eliminate any optimal solution. We can still use [Master($\hat{Z}$)-SVI] with the $\hat{Z}$ extended by Random-Search even if (22) and (23) are not supervalid (for convenience, we name these inequalities "λ-SVIs"). Since we keep $\hat{V}_\lambda > \lambda\underline{V}$ in every iteration, adding corresponding λ-SVIs to [Master($\hat{Z}$)-SVI] will finally lead to an approximate objective value V^*_λ which satisfies $V^*_\lambda \ge \frac{1}{\lambda}V^*$. We give the properties of λ-SVIs and prove it as follows.

Theorem 1. *For an interdiction plan $\hat{\boldsymbol{x}}$ given by [Master($\hat{Z}$)] during an iteration of Algorithm 2, let $\hat{\boldsymbol{z}}_\lambda$ denote a feasible solution of [Sub($\hat{\boldsymbol{x}}$)] that may be non-optimal. Let $\hat{V}_\lambda = \sum_{e_p \in \boldsymbol{A}_p}(c_{e_p} + \hat{x}_{e_p}d_{e_p})\hat{z}_{\lambda_{e_p}}$ and let V^* denote the global optimal objective value of [Master(Z)]. Then Type-I inequality $\hat{\boldsymbol{z}}^T_\lambda\boldsymbol{x} \ge 1$ does not eliminate all optimal solutions of [Master(Z)] unless the incumbent solution $\hat{\boldsymbol{x}}$ leads to a lower bound $\underline{V} \ge \frac{1}{\lambda}V^*$, providing that $\hat{V}_\lambda \le \lambda\underline{V}$.*

Proof of Theorem 1. Let $\boldsymbol{x}^*$ denote the global optimal solution of the defender's interdiction plan. Assuming that $\underline{V} < \frac{1}{\lambda}V^*$ during an iteration, if Type-I inequality $\hat{\boldsymbol{z}}^T_\lambda\boldsymbol{x} \ge 1$ eliminates all optimal solutions, which means $\hat{\boldsymbol{z}}^T_\lambda\boldsymbol{x}^* = 0$, then

$$\begin{aligned} V^* &\le \boldsymbol{c}^T\hat{\boldsymbol{z}}_\lambda + \boldsymbol{x}^{*T}\boldsymbol{D}\hat{\boldsymbol{z}}_\lambda \quad \text{(is naturally established)} \\ &= \boldsymbol{c}^T\hat{\boldsymbol{z}}_\lambda \quad \text{(because } \hat{\boldsymbol{z}}^T_\lambda\boldsymbol{x}^* = 0\text{)} \\ &= \hat{V}_\lambda - \hat{\boldsymbol{x}}^T D\hat{\boldsymbol{z}}_\lambda \\ &\le \hat{V}_\lambda \end{aligned}$$

$$\begin{aligned} &\leq \lambda \underline{V} \\ &< V^* \text{ (which is a contradiction)} \end{aligned}$$

□

In fact, the two types of SVIs and their corollaries can be modified to apply to a non-optimal pathfinding result $\hat{\boldsymbol{z}}$ if $\hat{\boldsymbol{z}}$ is an approximate optimal solution of the current subproblem [Sub($\hat{\boldsymbol{x}}$)]. The modified inequalities are not supervalid, which means that the inequalities may eliminate all optimal solutions. However, when this happens, we will already have an approximate global optimal interdiction solution.

Corollary 1. *For a feasible solution $\hat{\boldsymbol{z}}_\lambda$ of [Sub($\hat{\boldsymbol{x}}$)], which leads to an objective value $\hat{V}_\lambda \leq \lambda \underline{V}$. Order the $d_m \hat{z}_{\lambda_m} > 0$ so that $d_{m_1} \hat{z}_{\lambda m_1} \geq d_{m_2} \hat{z}_{\lambda m_2} \ldots \geq d_{m_H} \hat{z}_{\lambda m_H}$. Then, if $\lambda \underline{V} \geq \boldsymbol{c}^T \hat{\boldsymbol{z}}_\lambda + \sum_{h=1}^{p} d_{m_h} \hat{z}_{\lambda m_h}$ for $p \leq H$, the Type-I inequality of Theorem 1 can be tightened to $\hat{\boldsymbol{z}}_\lambda^T \boldsymbol{x} \geq p + 1$*

Proof of Corollary 1. Assuming that $\hat{\boldsymbol{z}}_\lambda^T \boldsymbol{x} \leq p$, then

$$\begin{aligned} V^* &\leq \boldsymbol{c}^T \hat{\boldsymbol{z}}_\lambda + \boldsymbol{x}^{*^T} \boldsymbol{D} \hat{\boldsymbol{z}}_\lambda \\ &\leq \boldsymbol{c}^T \hat{\boldsymbol{z}}_\lambda + \sum_{h=1}^{p} d_{m_h} \hat{z}_{\lambda_{m_h}} \text{ (because } \hat{\boldsymbol{z}}_\lambda^T \boldsymbol{x} \leq p \text{ and } \hat{z}_{\lambda_{m_h}} \in \mathbb{N} \text{ imply that} \\ &\quad \text{the number of edges selected by both } \hat{\boldsymbol{z}}_\lambda \text{ and } \boldsymbol{x}^* \text{ is no more than } p) \\ &\leq \lambda \underline{V} < V^* \text{ (which is a contradiction)} \end{aligned}$$

□

Theorem 2. *For an interdiction plan $\hat{\boldsymbol{x}}$ given by [Master($\hat{Z}$)] during an iteration of Algorithm 2, let $\hat{\boldsymbol{z}}_\lambda$ denote a feasible solution of [Sub($\hat{\boldsymbol{x}}$)] that may be non-optimal. Let $\tilde{\boldsymbol{z}}_\lambda = (diag(\mathbf{1} - \hat{\boldsymbol{x}}))\hat{\boldsymbol{z}}_\lambda$. Then, the Type-II inequality $\tilde{\boldsymbol{z}}_\lambda^T \boldsymbol{x} \geq 1$ does not eliminate all optimal solutions of [Master(Z)] unless the incumbent solution $\hat{\boldsymbol{x}}$ leads to a lower bound $\underline{V} \geq \frac{1}{\lambda} V^*$, providing that $\hat{V}_\lambda \leq \lambda \underline{V}$.*

Proof of Theorem 2. Assuming that $\underline{V} < \frac{1}{\lambda} V^*$, if Type-II inequality $\tilde{\boldsymbol{z}}_\lambda^T \boldsymbol{x} \geq 1$ eliminates all optimal solutions, which means $\tilde{\boldsymbol{z}}_\lambda^T \boldsymbol{x}^* = \boldsymbol{x}^{*^T} (diag(\mathbf{1} - \hat{\boldsymbol{x}}))\hat{\boldsymbol{z}}_\lambda = \boldsymbol{x}^{*^T} (\boldsymbol{I} - diag(\hat{\boldsymbol{x}}))\hat{\boldsymbol{z}}_\lambda = 0$, we can obtain that $\boldsymbol{x}^{*^T} (\boldsymbol{I} - diag(\hat{\boldsymbol{x}}))D\hat{\boldsymbol{z}}_\lambda = 0$ because all elements in vector $\boldsymbol{x}^{*^T} (\boldsymbol{I} - diag(\hat{\boldsymbol{x}}))$ are 0 or 1. Then

$$\begin{aligned} V(\boldsymbol{x}^*, \hat{\boldsymbol{z}}_\lambda) &= \boldsymbol{c}^T \hat{\boldsymbol{z}}_\lambda + \boldsymbol{x}^{*^T} \boldsymbol{D} \hat{\boldsymbol{z}}_\lambda \\ &= \boldsymbol{c}^T \hat{\boldsymbol{z}}_\lambda + \boldsymbol{x}^{*^T} (diag(\hat{\boldsymbol{x}})) \boldsymbol{D} \hat{\boldsymbol{z}}_\lambda \\ &\leq \boldsymbol{c}^T \hat{\boldsymbol{z}}_\lambda + \hat{\boldsymbol{x}} \boldsymbol{D} \hat{\boldsymbol{z}}_\lambda = \hat{V}_\lambda \\ &\leq \lambda \underline{V} \\ &< V^* \text{ (which contradicts the fact that} \\ &\quad \boldsymbol{x}^* \text{ and } V^* \text{ are global optimal solutions)} \end{aligned}$$

□

Corollary 2. *For a Type-II inequality $\tilde{\boldsymbol{z}}_{\lambda_1}^T \boldsymbol{x} \geq 1$, let $\tilde{M}_{\lambda_1} = \{m | \tilde{z}_{\lambda_{1m}} = 1\}$ and let $\tilde{M}_{\lambda_2}$ be any subset of $\tilde{M}_{\lambda_1}$ such that $\sum_{m \in \tilde{M}_{\lambda_2}} d_m + (\boldsymbol{c}^T \hat{\boldsymbol{z}}_{\lambda_1} + \hat{\boldsymbol{x}}_1^T \boldsymbol{D} \hat{\boldsymbol{z}}_{\lambda_1}) \leq \lambda \underline{V}$. Then, the Type-II SVI of Theorem 2 can be tightened to $\tilde{\boldsymbol{z}}_{\lambda_2}^T \boldsymbol{x} \geq 1$, where $\tilde{z}_{\lambda_{2m}} = \tilde{z}_{\lambda_{1m}}$ for all $m \in \tilde{M}_{\lambda_1} \setminus \tilde{M}_{\lambda_2}$ and 0 otherwise.*

Proof of Corollary 2. Assuming that $\underline{V} < \frac{1}{\lambda}V^*$, if Type-II inequality $\tilde{\boldsymbol{z}}_{\lambda_2}^T \boldsymbol{x} \geq 1$ eliminates all optimal solutions, which means $\tilde{\boldsymbol{z}}_{\lambda_2}^T \boldsymbol{x}^* = 0$, let $\hat{M}_{\lambda_1} = \{m | \hat{z}_{\lambda_{1m}} \geq 1\}$, then

$$\begin{aligned}
V(\boldsymbol{x}^*, \hat{\boldsymbol{z}}_{\lambda_1}) &= \boldsymbol{c}^T \hat{\boldsymbol{z}}_\lambda + \boldsymbol{x}^{*^T} \boldsymbol{D} \hat{\boldsymbol{z}}_\lambda \\
&= \sum_{m \in (\hat{M}_{\lambda_1} \setminus \tilde{M}_{\lambda_1})} (c_m \hat{z}_{\lambda_m} + x_m^* d_m \hat{z}_{\lambda_m}) + \sum_{m \in \tilde{M}_{\lambda_2}} (c_m \hat{z}_{\lambda_m} + x_m^* d_m \hat{z}_{\lambda_m}) + \\
&\quad \sum_{m \in (\tilde{M}_{\lambda_1} \setminus \tilde{M}_{\lambda_2})} (c_m \hat{z}_{\lambda_m} + x_m^* d_m \hat{z}_{\lambda_m}) \\
&= \sum_{m \in (\hat{M}_{\lambda_1} \setminus \tilde{M}_{\lambda_1})} (c_m \hat{z}_{\lambda_m} + x_m^* d_m \hat{z}_{\lambda_m}) + \sum_{m \in \tilde{M}_{\lambda_2}} (c_m \hat{z}_{\lambda_m} + x_m^* d_m \hat{z}_{\lambda_m}) \\
&= \sum_{m \in \tilde{M}_{\lambda_2}} x_m^* d_m \hat{z}_{\lambda_m} + \left[\sum_{m \in (\hat{M}_{\lambda_1} \setminus \tilde{M}_{\lambda_1})} (c_m \hat{z}_{\lambda_m} + x_m^* d_m \hat{z}_{\lambda_m}) + \sum_{m \in \tilde{M}_{\lambda_2}} c_m \hat{z}_{\lambda_m} \right] \\
&\leq \sum_{m \in \tilde{M}_{\lambda_2}} d_m + (\boldsymbol{c}^T \hat{\boldsymbol{z}}_{\lambda_1} + \hat{\boldsymbol{x}}_1^T \boldsymbol{D} \hat{\boldsymbol{z}}_{\lambda_1}) \\
&\leq \lambda \underline{V} < V^* \text{ (which leads to a contradiction)}
\end{aligned}$$

□

Based on Algorithm 3, we add the Random-Search module and get Algorithm 4 as follows:

Algorithm 4 Improved decomposition algorithm with Layer-Mapping and Random-Search

Input: An instance of LPNSPI and a tolerable approximation ratio $\frac{1}{\lambda}$
Output: A tolerable near-optimal interdiction plan $\boldsymbol{x}_\lambda^*$

$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{0}$; $\hat{Z} \leftarrow \emptyset$; $\underline{V} \leftarrow -\infty$; $\overline{V} \leftarrow \infty$
while $\overline{V} - \underline{V} > 0$ **do**
  if $\hat{\boldsymbol{x}} = \boldsymbol{0}$ **then**
    $\boldsymbol{x^0} \leftarrow \hat{\boldsymbol{x}}$; Calculate $\boldsymbol{w^0}$ and $\boldsymbol{K^0}$ using (4)-(6);
    $\hat{\boldsymbol{w}} \leftarrow \boldsymbol{w^0}$;
  else
    Solve **Layer-Mapping($\boldsymbol{x^0}$,$\hat{\boldsymbol{x}}$)** for $\hat{\boldsymbol{w}}$ and $\hat{\boldsymbol{K}}$;
  end if
  Solve [Sub($\hat{\boldsymbol{w}}$)-LM] for $\hat{\boldsymbol{y}}$ and the objective value $V_{\hat{\boldsymbol{w}}}$;
  $\hat{\boldsymbol{z}} = \hat{K}^T \hat{\boldsymbol{y}}$; $\hat{Z} \leftarrow \hat{Z} \cup \hat{\boldsymbol{z}}$;
  if $\underline{V} < V_{\hat{\boldsymbol{w}}}$ **then**
    $\boldsymbol{x}' \leftarrow \hat{\boldsymbol{x}}$; $\underline{V} \leftarrow V_{\hat{\boldsymbol{w}}}$;
  end if
  if $\overline{V} - \underline{V} \leq 0$ **then**
    break;
  end if
  Solve Random-Search($\hat{\boldsymbol{x}}, \hat{\boldsymbol{z}}, \hat{K}, \lambda, \underline{V}$) for $\hat{Z}_\lambda$;
  $\hat{Z} \leftarrow \hat{Z} \cup \hat{Z}_\lambda$;
  if [Master($\hat{Z}$)-SVI] is feasible **then**
    Solve [Master($\hat{Z}$)-SVI] for $\hat{\boldsymbol{x}}$ and the objective value $V_{\hat{Z}}$;
    $\overline{V} \leftarrow V_{\hat{Z}}$;
  else
    break;
  end if
end while

26: $\boldsymbol{x}_{\lambda}^{*} \leftarrow \boldsymbol{x}'$
27: **return** $\boldsymbol{x}_{\lambda}^{*}$

6. Computational Experiments

We tested our algorithms in a set of generated layered networks with directed random networks as their logical layers. Random networks, small-world networks, scale-free networks and grid networks are used as physical layer in these instances. The network in the logical layer are smaller than the physical-layer network in each instance, and each node of the logical layer correspond a randomly selected node of the physical layer, which means that these two nodes share the same entity. We generated random networks by connecting newly added nodes to previous nodes with a certain probability p. p is adjusted for each instance to ensure that all the random networks have the same expected average degree. The small-world networks are generated by first constructing a nearest-neighbor coupling network and then reconnecting the edges with different probabilities. For the scale-free case, we first generate a small random network, and then assign the connection probabilities according to the degrees of nodes, and finally preferentially connect the newly added nodes to the nodes with better connectivity in the light of the probabilities. As for the grid networks, conventional square lattice networks are used. The communication costs of edges c_{e_p}s and the interdiction increments d_{e_p}s in physical layer are integers that are randomly distributed on [1, 20] and [200, 1000], respectively. The resource consumption of interdiction $\boldsymbol{r}$ is set to $\mathbf{1}$ in practice. A time limit of 3600 s is set for each experiment.

Table 2 shows the parameters of the test problems we used. The blank cells repeat values from cells above. The numbers in the brackets of the column "N_l" and column "N_p" represent the average degree of the generated networks. For each problem we generated ten instances. We programmed the algorithms presented above using the MATLAB toolbox YALMIP and CPLEX 12.8 callable library. Computation is performed on a Windows 10 64-bit laptop with an Intel Core i7-9700K CPU (3.60 GHZ) and 16 GB of RAM.

Table 2. Test problem statistics.

Problem Name	Type of G_p	Type of G_l	N_l	N_p	R
rd2000	random	random	100 (3)	2000 (2)	2
rd5000			200 (3)	5000 (2)	3
rd10000			500 (3)	10,000 (2)	4
rd20000			1000 (3)	20,000 (2)	5
sf2000	scale-free		100(3)	2000 (3)	2
sf5000			200 (3)	5000(3)	3
sf10000			500 (3)	10,000 (3)	4
sf20000			1000 (3)	20,000 (3)	5
sw2000	small-world		100 (3)	2000 (4)	2
sw5000			200 (3)	5000 (4)	3
sw10000			500 (3)	10,000 (4)	4
sw20000			1000 (3)	20,000 (4)	5
gd1000	grid		100 (3)	1000 (2)	2
gd2000			100 (3)	2000 (2)	2

The value of λ in Algorithm 4 is set as 1.05, 1.10 and 1.15, respectively, and the error range of the corresponding optimal objective value is 95.2%, 90.9% and 87.0%. The basic results for LPNSPI are shown in Table 3. The column "instance" represents the number of instances we used for comparison in the problem of a certain scale. T, average solution time in seconds; $Std.T$, the standard deviation of the solution time; N, average iteration times; $Std.N$, the standard deviation of the iteration times. The numbers in brackets of the column "T" indicate the number of the instances which were solved within 3600 s. The "–"s of the same rows indicates "not applicable" because there was at least one instance which was not successfully solved within the allotted time. It can be intuitively seen from

Table 3 that the Layer-Mapping module make Algorithm 3 much faster than the basic decomposition algorithm with almost the same number of iterations, especially when the scale of the physical layer is large. However, despite this, Algorithm 3 cannot solve all the instances within the stipulated time. Combining Random-Search with Layer-Mapping, Algorithm 3 takes significantly less time and fewer iterations, and successfully solved all the instances. It should be noted that among the ten instances of 'gn1000', there are two extreme instances, which increase the average solving time of 'gn1000' problem (even more than the average solving time of 'gn2000'). However, it does not affect the comparison between the performance of different algorithms.

Table 3. The computational results for networks of different scale and different types.

Problem	Algorithm 1				Algorithm 3				Algorithm 4 ($\lambda = 1.05$)			
	T	*Std.T*	*N*	*Std.N*	*T*	*Std.T*	*N*	*Std.N*	*T*	*Std.T*	*N*	*Std.N*
rd2000	5.7	2.5	17.3	6.9	3.5	1.4	17.3	6.9	3.0	1.2	8.9	4.8
rd5000	22.4	12.9	30.4	16.5	8.7	5.8	30.4	16.5	6.9	3.2	11.7	5.7
rd10000	163.5	113.7	64.3	43.6	39.2	33.5	64.3	43.6	16.5	9.3	14.3	10.6
rd20000	(7)	–	–	–	(8)	–	–	–	86.5	132.8	25.0	26.7
sf2000	8.4	2.7	24.1	6.8	5.3	1.8	24.1	6.8	3.4	1.2	11.1	3.2
sf5000	39.9	20.3	47.1	22.5	17.8	11.4	47.1	22.5	8.4	3.3	15.6	5.5
sf10000	406.2	272.4	127.2	75.1	134.6	118.9	127.2	75.1	28.5	8.9	23.3	6.7
sf20000	(8)	–	–	–	(9)	–	–	–	95.4	39.9	26.4	7.2
sw2000	7.2	2.9	19.7	6.6	4.4	1.6	19.7	6.6	2.6	0.6	8.7	2.3
sw5000	46.1	30.9	49.1	28.0	20.2	15.7	49.1	28.0	6.3	2.0	12.8	4.0
sw10000	560.2	518.8	145.2	111.2	224.2	264.6	145.2	111.2	26.6	14.8	23.9	12.4
sw20000	(6)	–	–	–	(7)	–	–	–	84.0	41.3	27.7	12.7
gn1000	933.2	969.7	579.6	359.8	812.7	867.4	579.6	359.8	192.0	144.8	57.1	31.0
gn2000	757.3	675.9	488.4	267.8	655.5	622.1	488.4	276.8	157.5	126.7	45.0	24.9

To analyze the specific running time of each part (i.e., the master problem, the subproblem and the Random-Search part) of the three algorithms, we selected some "easy" instances of the hardest problems of each network type in Table 3 (that is, rd20000, sf20000, sw20000 , gn1000 and gn2000), where "easy" means that all three algorithms can solve the problem within 3600 s. The results are shown in Table 4. $T.M$ is the running time of the master problem; $T.S$ is the running time of the subproblem; $T.RS$ is the time spent on Random-Search. Initialization, formatting, and other parts of the program only account for a small portion of the time and are not listed. Err is the average error rate of the results obtained by Algorithm 4, and the numbers in parentheses indicate the number of instances where Algorithm 3 find a near-optimal solution rather than an exact optimal solution. $Std.Err$ is the standard deviation of Err. As Table 4 shows, Algorithm 3 takes almost the same time in solving the master problem as Algorithm 1, but takes much less time to solve the subproblem. With Random-Search, Algorithm 3 greatly reduces the time of the master problem and the subproblem, but it also takes a considerable amount of time in Random-Search to find near-optimal paths.

Table 4. The running time of the main parts and the error of Algorithm 4.

Problem	Instance	Algorithm 1		Algorithm 3		Algorithm 4 ($\lambda = 1.05$)				
		T.M	*T.S*	*T.M*	*T.S*	*T.M*	*T.S*	*T.RS*	*Err*	*Std.Err*
rd20000	7	108.7	695.0	107.2	16.9	6.9	9.3	13.6	0.28% (2)	0.005
sf20000	8	300.9	1435.0	313.4	67.4	19.5	14.8	48.9	0.11% (1)	0.003
sw20000	6	321.0	1473.0	337.1	29.4	20.7	12.3	29.0	0 (0)	0
gn1000	10	746.7	183.5	740.6	69.0	97.9	6.5	87.2	0.23% (5)	0.003
gn2000	10	582.7	171.2	597.8	54.2	91.2	4.4	61.4	0.30% (4)	0.005

A suitable value of λ allows Random-Search to find a large number of near-optimal paths, but at the same time, the final result may have some errors from the actual optimal solution. In our test instances, Algorithm 3 solved many instances accurately, with an overall average accuracy of more than 99.7%. Of course, we can set the value of λ to 0, which will make the final result the global optimal solution, but also make it difficult for Random-Search to find near-optimal paths. We tested Algorithm 3 on the case when $\lambda = 0$, and found that the running time results are not so satisfying because much fewer near-optimal paths were found.

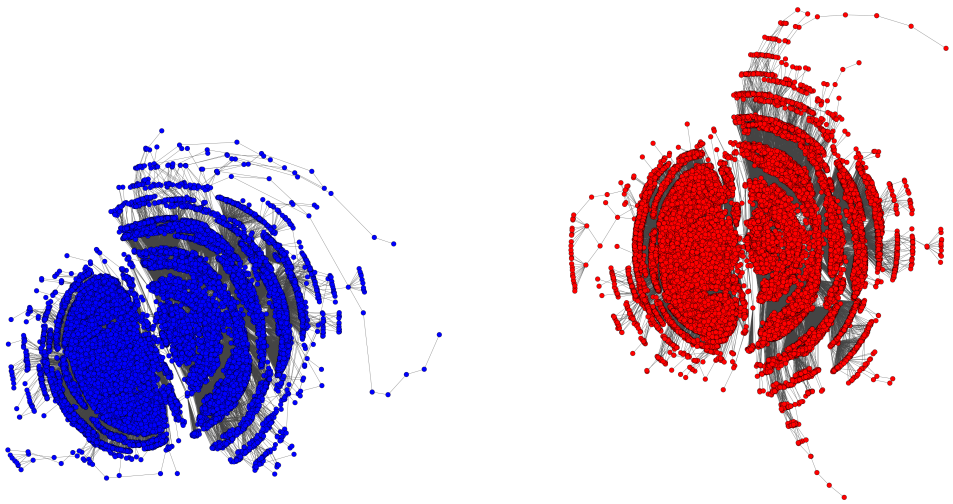
To study the effect of the value of λ on the solution speed of Algorithm 3, we set three cases of $\lambda = 1.05$, $\lambda = 1.1$, and $\lambda = 1.15$, and tested the algorithm on all the instances of the problems in Table 4. Table 5 compares the algorithm time, number of iterations, and errors of the near-optimal solution in the three cases. We used all ten instances of each problem when comparing the running time and the number of iterations of the algorithm. However, for "rd20000", "sf20000" and "sw20000", there were instances where Algorithm 1 and Algorithm 2 could not give the exact optimal results. The numbers in parentheses in the column "Err" indicate the number of instances used in calculating the average error rate and the number of non-optimal solutions given by Algorithm 3. For example, "0.25%(2 in 8)" means that in 8 verifiable instances, Algorithm 3 gives 2 approximate solutions (and 6 exact solutions), with an average error of 0.25%.

Table 5. Results for Algorithm 4 when λ changes.

Problem	**Algorithm 4**								
	$\lambda = 1.05$			$\lambda = 1.1$			$\lambda = 1.15$		
	T	***N***	***Err***	***T***	***N***	***Err***	***T***	***N***	***Err***
rd20000	86.5	25.0	0.25% (2 in 8)	48.9	14.3	0.15% (1 in 8)	36.4	9.7	0.46% (2 in 8)
sf20000	95.4	26.4	0.10% (1 in 9)	83.3	19.7	0.66% (4 in 9)	75.1	15.4	1.91% (5 in 9)
sw20000	84.0	27.7	0 (0 in 7)	66.2	18.5	0 (0 in 7)	51.0	11.7	0.50% (1 in 7)
gn1000	192.0	57.1	0.23% (5 in 10)	110.1	29.6	0.69% (7 in 10)	75.6	17.0	1.17% (8 in 10)
gn2000	157.5	45.0	0.30% (4 in 10)	89.9	25.1	0.68% (6 in 10)	74.0	16.7	0.81% (8 in 10)

Due to the randomness of the pathfinding results of Random-Search, the number of near-optimal paths found in a specific iteration will fluctuate, and the contribution of these paths and the corresponding SVIs to the master problem is also uncertain. In some instances, the algorithm took more time with a larger λ than with a smaller one. However, in general, as Table 5 shows, a slightly larger λ makes the algorithm converge faster.

Furthermore, we examine our improving methods in commercial data of a bi-layer network which is obtained from scanning over the Internet. This bi-layer network includes a physical layer of 36,409 nodes and 49,084 edges, and a logical layer of 32,490 nodes and 51,340 edges. The physical nodes contain switching equipment, terminal equipment, storage equipment, control equipment, etc., and they are connected by physical links which consist of optical fibers, twisted-pair lines, coaxial cables, wireless media links, etc. On the logical layer, nodes represent operating systems and other software, and links represent information transfers, remote logins, network sessions, etc. The network structure is shown in Figure 3.



(a) logical layer (b) physical layer

Figure 3. The structure of a real bi-layer IoT network (after data cleaning).

After data cleaning of the original network (remove isolated nodes and connected components which do not include the source-destination pair), we designed experiments to test the changes of the algorithm running time and blocking effect with the increase of interdiction resources. The time limit is set to 3600 s.

As shown in Table 6, the processing time of the algorithms increases rapidly when the number of interdicted edges (R) grows larger. Both Algorithm 3 and Algorithm 4 solve the problem with much less time than Algorithm 1 (benchmark) when interdicted edges are no more than 8, which confirms the effectiveness of Layer-Mapping. As a large number of near-optimal paths are found by Random-Search, when R is small, the solution of Algorithm 3 is faster than that of Algorithm 4. However, when R grows up to 9, neither Algorithm 1 nor Algorithm 3 can solve the problem within 3600 s. Algorithm 3 with both Layer-Mapping and Random-Search successfully solves the problem when R is no more than 12. The interdiction effects of the three algorithms are compared in Figure 4. The objective function, which is the length of the shortest s_l-t_l path, increases when R grows. We find that even if the λ in Algorithm 4 is set to 1.15, which leads to a potential error range of 87%, accurate optimal solutions are found in comparable cases (i.e., when R is no more than 8).

Table 6. The processing time when defender's resources increase.

R	Processing Time		
	Algorithm 1	**Algorithm 3**	**Algorithm 4 ($\lambda = 1.15$)**
1	81.3	18.7	48.4
2	187.8	33.5	55.5
3	275.5	51.6	74.5
4	299.0	92.4	103.5
5	258.2	81.8	116.9
6	265.0	115.8	83.6
7	605.2	276.4	211.6
8	828.2	403.0	292.9
9	-	-	384.2
10	-	-	499.5
11	-	-	1528.7
12	-	-	2456.1

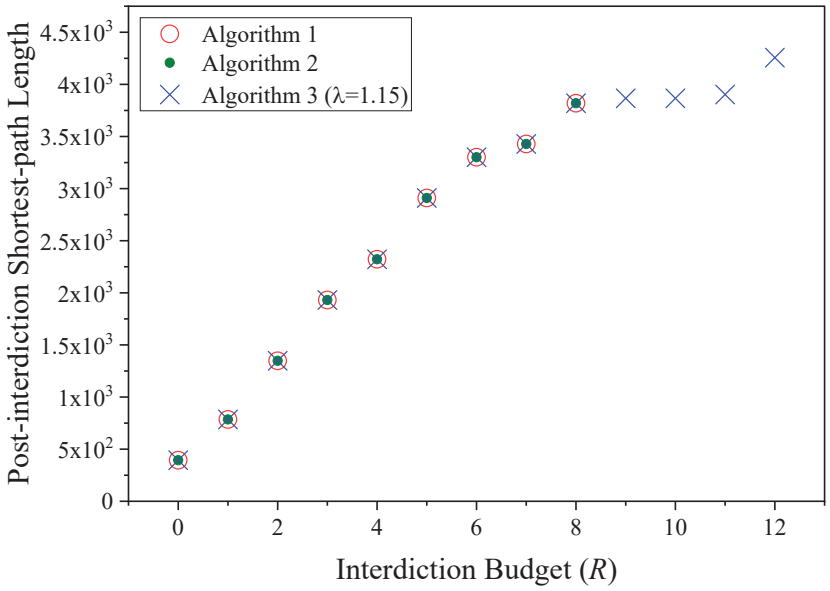


Figure 4. Interdiction effects in the bi-layer IoT network.

7. Conclusions

This paper focuses on blocking malicious network behaviors in IoT systems which can be modeled as logical–physical networks. The problem is represented as the shortest-path interdiction problem in layered networks, where the target paths and the interdiction behaviors are on different layers of the network. The attacker seeks to minimize the total communication cost of the attacking path from the source node to the target node on the logical layer, and the defender aims to maximize this path by interdicting edges on the physical layer. The interdiction of edges on the physical layer affects edges on the logical layer through the inter-layer relationship. In this problem, every node in the logical-layer network has a corresponding node on the physical layer, and the weight of each logical-layer edge is decided by a shortest path on the physical layer, with the endpoints of the logical-layer edge being the start node and the end node.

By referring to the experience of interdiction problems in monolayer networks, we model LPNSPI as a solvable form of Benders decomposition algorithm and apply "supervalid inequalities" (SVIs) on it. A Layer-Mapping module is proposed to deal with the explicit non-linearity of the objective function and reduce double counting of the subproblem. Layer-Mapping recalculates the status of the physical layer, basing on the current solution of the master problem and the initial network status. To accelerate the convergence of the decomposition algorithm, we raise Random-Search. By specifying an acceptable approximation range, Random-Search can randomly find multiple near-optimal paths in an iteration; as a result these paths and the corresponding SVIs can be added as constraints to the master problem. Computational results show the effectiveness of Layer-Mapping and Random-Search.

Author Contributions: Conceptualization, J.Y. and K.X.; methodology, J.Y. and K.X.; software, J.Y.; validation, J.Y.; formal analysis, J.Y.; investigation, J.Y.; resources, J.Y. and C.Z.; data curation, J.Y.; writing—original draft preparation, J.Y.; writing—review and editing, J.Y., K.X., C.Z., J.W.; visualization, J.Y., K.X. and G.Y.; supervision, W.Z.; project administration, C.Z.; funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was in part supported by the National Natural Science Foundation of China under Grants Nos. 71571186 and 71871217, and in part by the Natural Science Foundation of Hunan Province under Grant No. 2019JJ20019.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gao, J.; Buldyrev, S.V.; Stanley, H.E.; Havlin, S. Networks formed from interdependent networks. *Nat. Phys.* **2011**, *8*, 40–48. [CrossRef]
2. Farivar, F.; Haghighi, M.S.; Jolfaei, A.; Alazab, M. Artificial Intelligence for Detection, Estimation, and Compensation of Malicious Attacks in Nonlinear Cyber-Physical Systems and Industrial IoT. *IEEE Trans. Ind. Inf.* **2020**, *16*, 2716–2725. [CrossRef]
3. Liu, L.; Ma, Z.; Meng, W. Detection of multiple-mix-attack malicious nodes using perceptron-based trust in IoT networks. *Future Gener. Comput. Syst.* **2019**, *101*, 865–879. [CrossRef]
4. Ahmed, A.; Latif, R.; Latif, S.; Abbas, H.; Khan, F.A. Malicious insiders attack in IoT based Multi-Cloud e-Healthcare environment: A Systematic Literature Review. *Multimedia Tools Appl.* **2018**, *77*, 21947–21965. [CrossRef]
5. Cowie, J.; Ogielski, A.; Premore, B.; Smith, E.; Underwood, T. *Impact of the 2003 Blackouts on Internet Communications: Preliminary Report*; Technical Report; Renesys: Tokyo, Japan, 2004 .
6. Kushner, D. The real story of stuxnet. *IEEE Spectr.* **2013**, *50*, 48 – 53. [CrossRef]
7. Cho, C.S.; Chung, W.H.; Kuo, S.Y. Cyberphysical Security and Dependability Analysis of Digital Control Systems in Nuclear Power Plants. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *46*, 356–369. [CrossRef]
8. Chen, B.; Ho, D.W.C.; Zhang, W.A.; Yu, L. Distributed Dimensionality Reduction Fusion Estimation for Cyber-Physical Systems Under DoS Attacks. *IEEE Trans. Syst. Man Cybern. Syst.* **2017**, *49*, 455–468. [CrossRef]
9. Xiao, K.; Zhu, C.; Xie, J.; Zhou, Y.; Zhu, X.; Zhang, W. Dynamic Defense Strategy against Stealth Malware Propagation in Cyber-Physical Systems. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 1790–1798. [CrossRef]
10. Deng, Y.; Wu, J.; Xiao, Y.; Zhang, M.; Yu, Y.; Zhang, Y. Optimal Disintegration Strategy With Heterogeneous Costs in Complex Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2018**, 1–9. [CrossRef]
11. Qi, M.; Deng, Y.; Deng, H.; Wu, J. Optimal disintegration strategy in multiplex networks. *Chaos Interdiscip. J. Nonlinear Sci.* **2018**, *28*, 121104. [CrossRef]
12. Bica, I.; Chifor, B.C.; Arseni, .C.; Matei, I. Multi-Layer IoT Security Framework for Ambient Intelligence Environments. *Sensors* **2019**, *19*, 4038. [CrossRef]
13. Suárez-Albela, M.; Fraga-Lamas, P.; Fernández-Caramés, T. A Practical Evaluation on RSA and ECC-Based Cipher Suites for IoT High-Security Energy-Efficient Fog and Mist Computing Devices. *Sensors* **2018**, *18*, 3868. [CrossRef] [PubMed]
14. Holme, P.; Kim, B.; Yoon, C.; Han, S.K. Attack Vulnerability of Complex Networks. *Phys. Review. E Stat. Nonlinear Soft Matter Phys.* **2002**, *65*, 056109. [CrossRef] [PubMed]
15. Liu, B.; Xu, H.; Zhou, X. Stackelberg Dynamic Game-Based Resource Allocation in Threat Defense for Internet of Things. *Sensors* **2018**, *18*, 4074. [CrossRef]
16. Lim, C.; Smith, J.C. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Trans.* **2007**, *39*, 15–26. [CrossRef]
17. Scaparra, M.P.; Church, R.L. A bilevel mixed-integer program for critical infrastructure protection planning. *Comput. Oper. Res.* **2008**, *35*, 1905–1923. [CrossRef]
18. Morton, D.P.; Pan, F.; Saeger, K.J. Models for nuclear smuggling interdiction. *IIE Trans.* **2007**, *39*, 3–14. [CrossRef]
19. Chen, L.; Wang, Z.; Li, F.; Guo, Y.; Geng, K. A Stackelberg Security Game for Adversarial Outbreak Detection in the Internet of Things. *Sensors* **2020**, *20*, 804. [CrossRef]
20. Washburn, A.; Wood, K. Two-Person Zero-Sum Games for Network Interdiction. *Oper. Res.* **1995**, *43*, 243–251. [CrossRef]
21. Goldberg, N. Non-zero-sum nonlinear network path interdiction with an application to inspection in terror networks. *Nav. Res. Logist. (NRL)* **2017**, *64*. [CrossRef]
22. Janjarassuk, U.; Linderoth, J. Reformulation and sampling to solve a stochastic network interdiction problem. *Networks* **2008**, *52*, 120–132. [CrossRef]
23. Lunday, B.; Sherali, H. A Dynamic Network Interdiction Problem. *Inf. Lith. Acad. Sci.* **2010**, *21*, 553–574. [CrossRef]

24. Rad, M.; Kakhki, H. Maximum dynamic network flow interdiction problem: New formulation and solution procedures. *Comput. Ind. Eng.* **2013**, *65*, 531–536. [CrossRef]
25. Yates, J.; Sanjeevi, S. A length-based, multiple-resource formulation for shortest path network interdiction problems in the transportation sector. *Int. J. Crit. Infrastruct. Prot.* **2013**, *6*, 107–119. [CrossRef]
26. Xiangyu, W.; Cheng, Z.; Kaiming, X.; Quanjun, Y.; Yabing, Z. Shortest Path Network Interdiction with Goal Threshold. *IEEE Access* **2018**, *6*, 29332–29343.
27. Yates, J.; Lakshmanan, K. A constrained binary knapsack approximation for shortest path network interdiction. *Comput. Ind. Eng.* **2011**, *61*, 981–992. [CrossRef]
28. Bayrak, H.; Bailey, M.D. Shortest path network interdiction with asymmetric information. *Networks* **2010**, *52*, 133–140. [CrossRef]
29. Claudio, M.R.S.; Ramirez-Marquez, J.E. A bi-objective approach for shortest-path network interdiction. *Comput. Ind. Eng.* **2010**, *59*, 232–240.
30. Kennedy, K.T. Synthesis, Interdiction, and Protection of Layered Networks. Ph.D. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, OH, USA, 2009.
31. Xiangyu, W.; Kaiming, X.; Wei, D. Shortest path network interdiction of bi-layer networks with goal threshold. In Proceedings of the 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2016; pp. 70–75.
32. Baycik, N.O.; Sharkey, T.C.; Rainwater, C.E. Interdicting layered physical and information flow networks. *IISE Trans.* **2018**, *50*, 316–331. [CrossRef]
33. Israeli, E.; Wood, R.K. Shortest-path network interdiction. *Netw. Int. J.* **2002**, *40*, 97–111. [CrossRef]