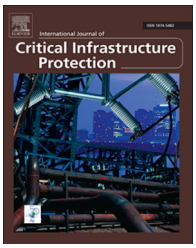
Available online at www.sciencedirect.com

ScienceDirect

www.elsevier.com/locate/ijcip



Risk mitigation strategies for critical infrastructures based on graph centrality analysis

George Stergiopoulos[a], Panayiotis Kotzanikolaou[b], Marianthi Theocharidou[c,*], Dimitris Gritzalis[a]

[a]Information Security and Critical Infrastructure Protection Laboratory, Department of Informatics, Athens University of Economics and Business, 76 Patission Avenue, GR-10434 Athens, Greece
[b]Department of Informatics, University of Piraeus, 85 Karaoli and Dimitriou, GR-18534 Piraeus, Greece
[c]Security Technology Assessment Unit, Institute for the Protection and the Security of the Citizen, European Commission Joint Research Centre, via E. Fermi 2749, I-21027 Ispra, Italy

ARTICLE INFO

Article history:
Received 8 January 2015
Received in revised form
4 May 2015
Accepted 13 May 2015
Available online 22 May 2015

Keywords:
Critical infrastructures
Dependency risk graphs
Graph centrality
Cascading failure
Information gain
Feature selection
Mitigation

ABSTRACT

Dependency risk graphs have been proposed as a tool for analyzing cascading failures due to critical infrastructure dependency chains. However, dependency chain analysis is not by itself adequate to develop an efficient risk mitigation strategy – one that specifies which critical infrastructures should have high priority for applying mitigation controls in order to achieve an optimal reduction in the overall risk. This paper extends previous dependency risk analysis research to implement efficient risk mitigation. This is accomplished by exploring the relation between dependency risk paths and graph centrality characteristics. Graph centrality metrics are applied to design and evaluate the effectiveness of alternative risk mitigation strategies. The experimental evaluations are based on random graphs that simulate common critical infrastructure dependency characteristics as identified by recent empirical studies. The experimental results are used to specify an algorithm that prioritizes critical infrastructure nodes for applying controls in order to achieve efficient risk mitigation.

1. Introduction

(Inter)dependencies between critical infrastructures are a key factor in critical infrastructure protection because they may allow a failure that is seemingly isolated in one critical infrastructure to cascade to multiple critical infrastructures. One way to analyze critical infrastructure dependencies is to use dependency risk graphs whose nodes represent critical infrastructures or components and directed edges represent the potential risk that the destination node may suffer due to its dependency on the source node in the event of a source node failure.

Kotzanikolaou et al. [7–9,14,15] have proposed a risk based methodology for assessing the cumulative risk of dependency risk paths – chains of critical infrastructure nodes that are (inter)connected due to their (inter)dependencies. The methodology uses as input existing risk assessment results from critical infrastructure operators and, based on the first-order dependencies between critical infrastructures, assesses the potential risk values of all the n^{th}-order dependency risk

*Corresponding author.
E-mail address: marianthi.theocharidou@jrc.ec.europa.eu (M. Theocharidou).

http://dx.doi.org/10.1016/j.ijcip.2015.05.003

chains. By computing and sorting all the dependency risk paths, a risk assessor may identify the most critical dependency chains by examining all potential dependency risk paths with cumulative risk above a predefined threshold.

Although the identification of the most important dependency chains is a key step towards efficient risk mitigation, certain open problems exist. A simple mitigation strategy is to apply security controls at the root node of each critical dependency chain. However, this may not always be a cost effective strategy because the effects on some nodes may not be measured properly. For example, consider the case where a small subset of nodes (not necessarily the roots of critical chains) affect a large number of critical dependency paths. Decreasing the probability of failure of these nodes by selectively applying security controls to them may result in a greater overall risk reduction. Another example is when nodes of high importance exist outside the most critical dependency paths (such as nodes that simultaneously affect many other nodes that belong to the set of most critical paths, or nodes that affect the overall dependency risk of the entire structure/graph).

Most current risk mitigation methodologies are empirical in nature and typically focus on critical infrastructures that initiate cascading failures (e.g., energy and information and communications infrastructures). By studying actual large-scale failures, it is clear that the two sectors often initiate serious cascading effects in interdependent critical infrastructures (e.g., the famous California blackouts of 2000 and 2001, and the Northeast Blackout of 2003). Nevertheless, focusing on some sectors and on potential initiators may not be suitable in every case. Therefore, the impact of each dependency should be considered along with the position of each critical infrastructure within the network of interdependent critical infrastructures.

The systematic identification of the most important nodes and the prioritization of nodes for applying security controls can be complex tasks. The need for a high-level and efficient risk mitigation technique that considers multiple characteristics of interdependent critical infrastructures is clear [3]. An optimal risk mitigation methodology would help identify the smallest subset of critical infrastructures that yields the highest overall risk reduction in a risk graph, although the candidate nodes may not initiate critical paths or may not even belong to the most critical paths.

This paper explores the use of graph centrality metrics in dependency risk graphs in order to prioritize critical infrastructures when applying risk mitigation controls. Alternative risk mitigation strategies are implemented algorithmically and subsequently evaluated empirically using simulations. The ultimate goal is to identify the minimum subset of critical infrastructure nodes in a dependency risk graph whose risk treatment (application of security controls) would result in the maximum risk reduction in the entire graph of interdependent critical infrastructures. The approach incorporates three key extensions: (i) data mining techniques to identify correlations between centrality metrics and high impact critical infrastructure nodes in a dependency risk graph; (ii) optimum centrality metrics to develop and test various risk mitigation strategies that maximize risk reduction; and (iii) analysis of nodes with high numbers of inbound (sinkholes) and outbound connections and the comparison of their risk reduction results.

Experiments were conducted on hundreds of random graphs with randomly selected dependencies in order to validate the proposed mitigation strategies. The random graphs were created to satisfy the constraints encountered in real-world interconnected critical infrastructures. The experimental results demonstrate the efficiency of the proposed risk mitigation strategies. The mitigation strategies can be used proactively to analyze interconnections in large-scale interdependent infrastructures and pinpoint underestimated or ignored infrastructures that are, in fact, critical to reducing the overall risk.

2. Building blocks

The proposed methodology has three building blocks: (i) dependency risk graphs and a multi-risk dependency analysis methodology for modeling cascading failures; (ii) graph centrality metric based analysis of dependency risk graphs; and (iii) feature selection techniques that help evaluate the effects of centrality metrics on risk mitigation. This section briefly describes the three building blocks.

2.1. *Multi-risk dependency analysis methodology*

A dependency is defined as "the one-directional reliance of an asset, system, network or collection thereof – within or across sectors – on an input, interaction or other requirement from other sources in order to function properly" [17]. The methodology described in this paper extends the dependency risk methodology of Kotzanikolaou et al. [7,8], which was developed to analyze multi-order cascading failures.

2.1.1. *First-order dependency risk*

Kotzanikolaou et al. [7,8] initially consider first-order dependencies between critical infrastructures and go on to model n^{th}-order dependencies. Each dependency from a node CI_i to a node CI_j is assigned an impact value $I_{i,j}$ and likelihood value $L_{i,j}$ of a disruption. Note that the impact and likelihood values are assumed to be obtained from organization-level risk assessments performed by critical infrastructure operators. The product of $I_{i,j}$ and $L_{i,j}$ is the dependency risk $R_{i,j}$ of infrastructure CI_j due to its dependency on CI_i. Dependencies are visualized in a graph $G = (N, E)$ where N is the set of nodes (or infrastructures or components) and E is the set of edges (or dependencies). The graph is directional and a destination critical infrastructure receives a risk from a source critical infrastructure as a result of its dependence on the source infrastructure.

2.1.2. *Extension to n^{th}-order dependency risk*

Let $\mathbb{CI} = (CI_1, \ldots, CI_m)$ be the set of all the considered critical infrastructures. An algorithm proposed by Kotzanikolaou et al. [7,8] examines each critical infrastructure as a potential root node of a cascading effect. Let CI_{Y_0} denote a critical infrastructure that is the root of a dependency chain and $CI_{Y_0} \rightarrow CI_{Y_1} \rightarrow \ldots \rightarrow CI_{Y_n}$ denote the corresponding chain of length n. Then, the algorithm computes the cumulative

dependency risk of the n^{th}-order dependency chain as

$$DR_{Y_0,\ldots,Y_n} = \sum_{i=1}^{n} R_{Y_0,\ldots,Y_i} \equiv \sum_{i=1}^{n} \left(\prod_{j=1}^{i} L_{Y_{j-1},Y_j} \right) \cdot I_{Y_{i-1},Y_i} \quad (1)$$

Informally, this equation computes the dependency risk contributed by each affected node in the chain due to a failure of the source node. The computation of the risk is based on a risk matrix that combines the likelihood and the incoming impact values of each node in the chain. Interested readers are referred to [7] for additional details.

2.2. Graph centrality metrics

Graph centrality analysis attempts to quantify the position of a node in relation to other nodes in a graph. In other words, graph centrality metrics are used to estimate the relative importance of a node in a graph. Several centrality metrics have been specified, each measuring a different characteristic:

- *Degree centrality*: This metric measures the number of edges attached to each node. Given a node u, the degree centrality is defined as $C_d(u) = deg(u)$ where $deg(u)$ is the total number of outbound and inbound edges of node u.
- *Closeness centrality*: This metric quantifies the intuitive central or peripheral placement of a node in a two-dimensional region based on geodesic distances. It is defined as $C_c(u) = \sum_{\forall v \in V(G)} \delta(u,v)$ where $\delta(u,v)$ is the average shortest path between node u and any other node in the graph.
- *Betweenness centrality*: This metric measures the number of paths in which a node participates. It is defined as $C_b(u) = \sum_{u \neq i \neq j \in V} \delta_{ij}(u)$ where $\delta_{ij}(u) = \sigma_{ij}(u)/\sigma_{ij}$. Note that $\sigma_{ij}(u)$ denotes the number of geodesic distances from i to j in which node u is present and σ_{ij} is the number of geodesic distances from i to j.
- *Bonacich (eigenvector) centrality*: This metric [2] measures the influence of a node in a network as $c_i(\alpha,\beta) = \sum_j (\alpha - \beta c_i) R_{i,j}$ where α is a scaling factor, β reflects the extent to which centrality is weighted, R is the node adjacency matrix, I is the identity matrix and l is the matrix of ones. An adjacency matrix is an $N \times N$ matrix with each element assigned a value of one if an edge exists between the corresponding nodes, and zero otherwise.
- *Eccentricity centrality*: This metric is similar to closeness centrality. It corresponds to the greatest distance from among all the shortest paths between a node u and other nodes (based on geodesic distances).

2.3. Feature selection

Feature selection, also called variable selection or attribute selection, is used in data mining for selecting attributes in a data set that are most relevant to a predictive modeling problem. Feature selection is critical for the effective analysis of characteristics in data sets because they frequently contain redundant information that is not necessary to build predictive models [11]. Removing non-informative terms according to corpus statistics [19] increases both effectiveness and efficiency.

This work considers centrality metrics as potential features in a dependency risk graph. A predictive model is used to detect which features (i.e., centrality metrics) can be used to characterize nodes that significantly affect the cumulative risk of dependency paths. Specifically, the model helps identify the centrality metrics that are the most useful (i.e., provide the most information) about whether or not a critical infrastructure node greatly affects the overall risk of a dependency risk graph. Features (centrality metrics) are ranked lower if they are common to the positive set (i.e., nodes with high impacts on risk paths) and the negative set (i.e., nodes with low impacts on risk paths). Features are ranked higher if they are effective discriminators for a class. The feature selection tests described in this paper employ two well-known measures, information gain and gain ratio.

Information gain [1] is a statistical measure that has been successfully applied to feature selection in information retrieval problems. Several researchers [4,16] have used this approach for feature selection before training a binary classifier. The approach is used in this work to classify centrality metrics that tend to appear often in nodes that greatly affect the cumulative risk of dependency risk graphs.

Gain ratio is a variation of information gain. The difference is that the information gain measure prefers to select attributes with large values [6]. Gain ratio overcomes this bias by taking into consideration the statistical appearance of a feature in the negative and positive sets (i.e., if a specific centrality metric is rarely detected in dangerous critical infrastructure nodes but is never found in safe nodes, then it has a rather low information gain rank but a very high gain ratio rank). The gain ratio measure is used to identify centrality metrics and combinations of centrality metrics that rarely appear in graphs, but significantly affect the cumulative risk in paths when they are present.

3. Centrality metrics in dependency risk graphs

This section explores the effects of centrality metrics on the importance of nodes in dependency risk graphs. Note that the edges in the graphs considered here denote directed risk relations between nodes, not topological connections. Intuitively, nodes with high centrality measures are expected to have high effects on the overall dependency risk. The question that needs to be answered is: Which centrality metrics have higher effects on risk propagation? Nodes with the highest centrality values are good candidates for implementing risk mitigation controls. The goal is to design an algorithm that is capable of identifying these nodes and to use it to articulate a cost-effective risk mitigation strategy.

3.1. Analysis of centrality metrics

Each of the centrality metrics is considered in order to analyze its effect on the overall risk in a dependency risk graph.

3.1.1. *Degree centrality*

A node with high degree centrality has many dependencies. Since the edges in a risk graph are directional, degree centrality should be examined for two cases:

- *Inbound degree centrality*: This is the number of edges that end at a node. A high inbound degree centrality indicates a cascade resulting node [10].
- *Outbound degree centrality*: This is the number of edges that start at a node. A high outbound centrality indicates a cascade initiating node [10].

Nodes with high inbound degree centrality in a risk graph are natural "sinkhole" points of incoming dependency risk. Nodes with high outbound degree centrality appear to be suitable nodes to consider when prioritizing mitigation controls. Indeed, if proper mitigation controls are applied to these nodes, then multiple cumulative dependency risk chains can be reduced simultaneously. This could be a very cost-effective approach for applying controls to particular high risk edges or high risk paths. Clearly, it is not guaranteed that applying one or more security controls to a node with high outbound degree centrality would positively impact many (or all) outgoing dependency chains that involve the node. However, it would be beneficial if the mitigation strategy could initially examine the possible security controls.

3.1.2. *Closeness centrality*

Nodes with high closeness centrality have short average distances from most nodes in a graph. In the case of a dependency risk graph, nodes with high closeness centrality tend to be part of many dependency chains; sometimes, these nodes may even initiate dependency chains. Since cascading effects tend to affect relatively short chains (empirical evidence indicates that cascading failures rarely propagate deeply [18]), intuitively, nodes with high closeness centrality would have a greater effect on the overall risk of a dependency chain than nodes with low closeness centrality. To formalize this, consider Eq. (1) that computes the cumulative risk of a dependency chain. The closer a node is to the initiator of a cascading event, the more effect it would have on the cumulative dependency risk because the likelihood of its outgoing dependency would affect all the partial risk values of subsequent dependencies (edges).

A more effective way to exploit closeness centrality in mitigation decisions is to compute the closeness of every node with respect to the subset of the most important initiator nodes. Regardless of the approach that is used, the *a priori* knowledge and intuition of risk assessors could be leveraged. In general, risk assessors have an understanding of which nodes are most important in cascading failure scenarios. For example, empirical results show that energy nodes and information and communications nodes are the most common cascade initiators [18].

In addition, nodes with high outbound degree centrality are likely to participate in multiple dependency risk chains. Thus, it is possible to first identify the subset of the most important nodes with regard to cascading failures and then compute the closeness of all other nodes with respect to this subset of nodes as a secondary criterion for mitigation prioritization.

3.1.3. *Eccentricity centrality*

Similar to closeness centrality, a node has higher eccentricity centrality if it has a small maximum distance to every other reachable node. The small maximum distance is computed as the greatest distance from among all the shortest paths between the node and all other nodes (based on geodesic distances). If a critical infrastructure node has high eccentricity, then all the other critical infrastructure nodes are proximal to it.

3.1.4. *Betweenness centrality*

In a dependency risk graph, a node with high betweenness centrality lies on a high proportion of dependency risk paths. This means that, although the nodes may not be initiating nodes of cascading failures (high outbound centrality) or may not belong to a path with high cumulative dependency risk, they tend to contribute to multiple risk paths and, therefore, play an important role in the overall risk. Applying mitigation measures to these nodes is likely to decrease the dependency risk of multiple chains simultaneously.

Upon comparing closeness centrality with betweenness centrality, it appears that closeness should precede betweenness as a mitigation criterion. Although a node that is in multiple paths will eventually affect multiple chains, it is possible that such a node, because it tends to be at the end of the chains, will not affect the cumulative dependency risks of the chains (recall that nodes with high-order dependencies are rarely affected by cascading failures).

3.1.5. *Bonacich (eigenvector) centrality*

A node with high Bonacich (eigenvector) centrality [2] is adjacent to nodes with very high or very low influence, depending on whether the parameter β is greater or less than zero. In a risk dependency graph, nodes with high eigenvector centrality (when $\beta \leq 0$) are of interest because these nodes are connected to other nodes that also have high connectivity. This is an interesting measure for critical infrastructure risk graphs as these nodes not only can cause cascading failures to more nodes, but they can cause multiple cascading chains of high risk. In contrast, a less connected node means that it shares fewer dependencies with other nodes and is only affected by specific nodes in the graph. This means that applying mitigation measures to the node may not affect the overall risk in a significant manner. On the other hand, if mitigation controls are applied to a node with high eigenvector centrality (when $\beta \leq 0$), then the most powerful (or critical) node is modified and this, in turn, affects several other important nodes.

3.2. *Classifying centrality metrics*

In order to validate the intuitive analysis of centrality metrics in dependency risk graphs presented in Section 3.1, feature selection from the discipline of data mining was used to empirically study possible relations between the risk levels of nodes in dependency paths (as assessed using the methodology of [8])

and the various node centrality metrics. Specifically, an experiment was conducted to demonstrate how often nodes simultaneously appear in (i) the set of the most critical paths (i.e., paths with the highest cumulative dependency risk values); and (ii) sets of nodes with the highest centrality values in a dependency risk graph.

The experiment simulated graphs that were randomized based on constraints proposed in empirical studies [10,18] in order to resemble real critical infrastructure dependencies. The following constraints were used:

- *Occasional tight coupling*: Some node relationships in the dependency risk graph must have high impact values. This was achieved by generating random risk values with relatively high lower and upper impact and likelihood bounds.
- *Interactive complexity*: Real-world critical infrastructure dependencies have high interactive complexity. Informally, this implies that it is not possible to foresee all the ways that things can go wrong. To achieve this, the experiment constructed random graphs of 50 nodes with high complexity; the critical paths up to fourth-order dependencies had 230,300 to 2,118,760 possible chains.
- *Number of dependencies*: The experiment randomly simulated critical infrastructure nodes with one to seven connections (dependencies).
- *Path length*: The experiment used critical paths with lengths of three to four hops.
- *Number of initiators*: The experiment used graphs in which more than 60% of the nodes acted as initiators.
- *Connectivity of initiators*: The experiment used initiator nodes that tended to have higher connectivity than other nodes.

The experiment used the Weka tool (version 3.6). All the simulations were conducted using the information gain and gain ratio based feature selection algorithms in Weka. Weka's Attribute Selection Ranker was employed to rank the output results.

3.3. *Feature selection results*

The data set employed in the experiment comprised 32,950 nodes extracted from about 700 graphs containing about 774,015,270 paths. Dependency graphs were created according to the aforementioned constraints. Feature selection algorithms were then used to detect correlations between high centrality metrics and critical infrastructure nodes that greatly affect the overall risk.

Weka's output rankings provide insights into how high centrality features characterize dangerous nodes that affect the risks of the top critical paths. The results of two sets of tests, inbound tests and outbound tests, are presented in Tables 1 and 2; the inbound tests used inbound degree centrality measures while the outbound tests used outbound degree centrality measures. Table 1 shows the results using information gain while Table 2 shows the results using gain ratio. Note that the two highest values in the inbound and outbound tests are highlighted using a bold font.

The Weka results provide valuable information about how nodes with high centrality values affect dependency risk graphs:

- The number of critical paths with at least one node with a high centrality metric is very high. An average of 74.4% of the most critical paths in all the generated graphs contained a node with at least one high centrality metric. This percentage increased dramatically to 99.9% for the top 10% of most critical paths, which leads to the conclusion that the top 10% of paths frequently pass through the same nodes as the top 1% of paths. When even larger sample sets (with more than 10% of critical paths) were analyzed, almost all the nodes with high centrality were found to belong to a critical path.
- The gain ratio results demonstrate that critical infrastructure nodes with high values for all the centrality metrics rarely appear in graphs but, when the nodes do appear, they definitely affect the cumulative risk of critical paths. Thus, critical infrastructure nodes with high values for all the centrality metrics are of high importance.
- Nodes with high closeness centrality appear to be the best critical infrastructure node candidates. However, the results clearly show that all the centrality metrics are useful indicators of important critical infrastructure nodes (the information gain is above 20% for all the features). Therefore, all the centrality metrics must be considered in each graph.
- Nodes with high closeness and/or degree centrality, both outbound and inbound (sinkholes), tend to mitigate high amounts of risk. An average of 49.6% of all dangerous nodes tested had high closeness centrality values and an average of 45% of the nodes had high degree centrality values.
- Risk mitigation is highly dependent on the graph being considered. For example, in graphs containing nodes that act as single points of failure, high centrality values are excellent indicators of important critical infrastructure nodes. However, in graphs containing nodes with equal degrees, the identification of nodes for risk mitigation is often less successful. Fortunately, critical infrastructure dependencies tend to have specific structural characteristics [10,18] and these characteristics can be leveraged to identify nodes for risk mitigation.

4. Risk mitigation algorithm

The experimental results demonstrate that even if nodes with high centrality are only a small fraction of the nodes in the most critical risk paths, they still affect all the critical paths greatly. Additionally, the feature analysis results suggest that, depending on the graph being analyzed, all the centrality metrics can play important roles. Thus, it is essential for a risk mitigation strategy to take into

Table 1 – Weka output ranking using information gain.

Information gain	Inbound test	Outbound test
Betweenness	0.259	0.277
Eccentricity	0.238	0.285
Closeness	**0.387**	**0.345**
Eigenvector	0.151	0.260
Intersection of all centralities	0.176	0.248
Inbound degree (sinkholes)	–	**0.302**
Outbound degree	**0.281**	–

consideration the centrality metrics that yield the best results (e.g., high closeness values) as well as the distinctive features of each graph and choose the correct combination of centrality metrics to identify the nodes for applying risk mitigation controls.

Algorithm 1. Decision-tree traversal using information gain to select nodes for risk mitigation.

Procedure choosingNodeSubsets ($\mathbb{C}, IG_c(D), N, D$)
Inputs
High centrality subsets: $\mathbb{C} = \{C_C, C_I, C_B, C_{Eg}, C_E\}$
Head set of all nodes: N
Number of nodes to be chosen from D: *PERCENT*
Subset of dangerous nodes: $D \subset N$
Information gain of a centrality subset c over D: $IG_c(D)$
Output
Percentage of nodes from $D' \subset N$ for risk mitigation
Step 1 – Start:
Mark all sets in $\mathbb{C}$ as unused
Position at the root node
Step 2 – Leaf:
If $\mathbb{C}$ is empty **then**
Output top nodes from D with highest centrality
END
else
if $IG_c(D) = 0\ \forall c \in \mathbb{C}$ **then**
Output top nodes from D with highest centrality
END
end if
end if
Step 3 – Decide case:
if nodes in $D <$ *PERCENT* **then**
Go to *Step* 4
else
Go to *Step* 5
end if
Step 4 – Backtrack:
if at the root **then**
STOP
else
Return to the node above the current node
Choose different $c \in \mathbb{C}$
Go to *Step* 3
end if
Step 5 – Decision:
if $IG_{c_1}(D) = IG_{c_2}(D)\ \forall c_1, c_2 \in \mathbb{C}$ **then**
Select the leftmost c from $\mathbb{C}$
else
Calculate information gain $IG_c(D)\ \forall c \in \mathbb{C}$
Select c from $\mathbb{C}$ with the highest $IG_c(D)$
end if
$N \leftarrow N \subset c$
$D \leftarrow D \subset c$
$\mathbb{C} \leftarrow \mathbb{C} - \{c\}$
Go to *Step 2*
end procedure

Table 2 – Weka output ranking using gain ratio.

Gain ratio	Inbound test	Outbound test
Betweenness	0.08	0.101
Eccentricity	0.08	0.101
Closeness	**0.14**	**0.120**
Eigenvector	0.06	0.09
Intersection of all centralities	**0.458**	**0.550**
Inbound degree (sinkholes)	–	0.103
Outbound degree	0.101	–

These results are incorporated in a node selection algorithm for efficient risk mitigation (Algorithm 1). The heuristic algorithm recursively chooses the best set of nodes by limiting its selection options until it reaches a set of selected nodes of a predefined size.

4.1. *Modeling feature selection results as algorithm restrictions*

Three observations were made in Section 3.3: (i) nodes with high values for all the centrality metrics always affect the overall risk of a graph; (ii) nodes with high closeness and betweenness centralities appear to have greater impacts than nodes with high values for the other metrics; and (iii) nodes with high values for more centrality metrics tend to have higher impacts on a risk graph. Taking these observations into account, the following steps are incorporated in the dynamic node selection algorithm for risk mitigation:

1. *Model definition*: Define the subsets of nodes with the highest values for each centrality metric and compute their information gain over the entire node set.
2. *Setup*: Pre-compute all the centrality metric values for each node in the graph. Using the centrality values, generate all the high centrality subsets. These subsets act as features for feature selection. In addition, pre-compute all the dependency risk paths in the graph and sort them according to their weights (total cumulative risk value of each path).
3. *Using decision trees and information gain for node selection*: Define the information gain method and use it to split the node sets.
4. *Traversing a decision tree using information gain splits*: Traverse a decision tree and select the optimal subset of nodes to apply risk mitigation controls.

4.1.1. *Model definition*

Let C_A denote the subset of top X% of nodes with the highest centrality values for all the centrality sets, C_E denote the subset of the top X% of nodes with the highest eccentricity centrality, C_C denote the subset of the top X% of nodes with the highest closeness centrality, C_I denote the subset of the top X% of nodes with the highest inbound degree centrality (sinkholes), C_B denote the subset of the top X% of nodes with the highest betweenness centrality, and C_{Eg} denote the subset of the top X% of nodes with the highest eigenvector centrality.

Let N denote the set of nodes in a risk graph. Let $D \subset N$ denote the subset of nodes that participate in the top 20 critical risk paths in the graph. Finally, let $IG_x(N)$ denote the

information gain when choosing subset C_x as a feature of the node set N. For example, $IG_B(N)$ denotes the information gain when using high betweenness centrality as a feature of set N.

Information gain values were calculated for each centrality feature subset separately. The high centrality subset C_x with the highest information gain over N is deemed to have the most impact on the risk graph in comparison with the other high centrality subsets.

For example, in a sample dependency risk graph, there may exist nodes that act as initiators of cascading events in the three most critical paths (i.e., critical paths with the highest cumulative dependency risk values). However, nodes that have less influence on the top three most critical paths also exist, and these nodes may affect the overall risk of the entire dependency risk graph more than the aforementioned initiators. The risk mitigation algorithm is designed to select these nodes over the three initiator nodes because it uses the information gain values of all the high centrality node subsets.

4.1.2. *Setup*

The centrality metric values of each node are pre-computed along with all the dependency risk paths in the graph. For each centrality metric, the algorithm uses the pre-computed centrality values and creates subsets C_E, C_C, C_I, C_B and C_{Eg}.

Next, the algorithm computes all the risk paths in the graph and sorts them according to their total accumulated risk values. All the outputs are used as input to the risk mitigation algorithm, which selects the most effective nodes for risk mitigation.

4.1.3. *Using decision trees and information gain for node selection*

A decision can be made (and a decision tree traversed) by creating a split of the head set of critical infrastructure nodes N to yield smaller subsets (graph leaves). Given a dependency risk graph, the algorithm selects the centrality node subset with the highest impact on the graph. Then, it outputs a new subset that is the intersection of the selected subset with the original set. This process is repeated for each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the information gain calculation at a specific point in the decision tree produces the same information gain value for the target subset of nodes, or when splitting no longer adds information to the prediction.

Algorithm 1 defines a sequence of high centrality subsets that act as features. The risk mitigation algorithm investigates this sequence to rapidly narrow down the group of critical infrastructure nodes with the highest impacts on the dependency risk graph. The feature selection increases the effectiveness and efficiency of the algorithm because it removes non-informative critical infrastructure nodes based on the corpus statistics [19].

4.1.4. *Traversing a decision tree using information gain splits*

The risk mitigation strategy can be simplified by always choosing the subset of nodes with the highest information gain at each step in the decision tree.

Assume that a dependency risk graph has all the centrality metric values calculated and all the high centrality subsets specified. Then, Algorithm 1 would run as follows:

- *Step* 1: Find the centrality feature that yields the most information gain for the root set of nodes $D \subset N$. The information gain is calculated for all four high centrality subsets (features):
  - $IG_C(D) = 0.246$ (Gain of closeness subset over D).
  - $IG_B(D) = 0.029$ (Gain of betweenness subset over D).
  - $IG_S(D) = 0.051$ (Gain of sinkholes subset over D).
  - $IG_{Eg}(D) = 0.151$ (Gain of eigenvector subset over D).
  - $IG_E(D) = 0.048$ (Gain of eccentricity subset over D).

  The high closeness centrality subset has the highest information gain, so it is used as the decision attribute in the root set D. From here on, D' is the subset $D \cap C_C$ of nodes and N becomes the $N \cap C_C$ subset.
- *Step* 2: Determine the feature that is tested in the second branch. Since C_C is already used as the root, it is only necessary to consider the remaining four centrality subsets, betweenness, sinkholes, eigenvector and eccentricity:
  - $IG_B(D) = 0.246$
  - $IG_{Eg}(D) = 0.029$
  - $IG_S(D) = 0.051$
  - $IG_E(D) = 0.000$

  In this step, the betweenness centrality subset has the highest information gain, so it is used as the decision node. From here on, D' is the subset $D \cap C_B$ of nodes and N' becomes the $N \cap C_B$ subset.
- *Steps* 3 *to N*: The process is repeated until all the nodes are classified and D has three nodes remaining, or the algorithm runs out of centrality attributes.

5. Algorithm validation

This section empirically analyzes the efficiency of the risk mitigation algorithm (Algorithm 1) presented in Section 4. A total of 2000 random experiments were conducted. In each experiment, a graph containing up to 50 nodes was generated based on the constraints described in Section 3.2.

5.1. *Efficiency analysis*

The previous results have shown that centrality measures can indeed characterize the important critical infrastructure nodes that significantly affect the overall risk in a dependency risk graph. To evaluate the effectiveness of the proposed risk mitigation strategy, it is necessary to emulate the implementation of mitigation controls in the nodes identified as being important (i.e., dangerous) by Algorithm 1. Examples of controls are the repair prioritization of nodes (i.e., which node to send a repair crew first), increase in asset redundancy at a node, or any other control that reduces the consequences and/or likelihood of failure.

The experiments emulated the implementation of mitigation controls at a node CI_i by reducing the likelihood value $L_{i,j}$ of cascading failure from node CI_i to node CI_j for all nodes CI_j

that depend on node CI_i. Specifically, the likelihood values $L_{i,j}$ of all the dependencies originating from CI_i were reduced by 20%. To assess the result of applying risk mitigation controls on each selected subset of nodes from a graph, the dependency risk values were computed before and after the implementation of risk mitigation, and the corresponding risk reduction was computed in each case.

The efficiency of the proposed risk mitigation algorithm was assessed by calculating the percentage risk reductions in the graphs based on the aforementioned emulation of mitigation controls. The effects on three risk metrics were examined using Algorithm 1: (i) risk reduction achieved for the most critical path; (ii) risk reduction achieved for the sum of the risks of the top 20 paths with the highest cumulative dependency risks; and (iii) risk reduction achieved for the sum of the risks of all graph paths. Only acyclic paths were considered to reduce overlap. For example, if a high risk path was cyclic, its acyclic part would most likely also be a high risk path. Thus, the computation of cyclic paths was avoided in the experiments.

Mitigation controls were implemented on three out of the 50 critical infrastructure nodes in each experiment. These corresponded to 6% of the nodes in the entire graph.

5.1.1. *Effect on the most critical path*

An average risk reduction of 12.1% was achieved when using Algorithm 1 to select critical infrastructure nodes for implementing mitigation controls. The highest risk reduction achieved over all the experiments was 43.7%, where almost half the risk of the most critical path was mitigated. This risk mitigation was the highest achieved from among all the mitigation strategies tested. A more thorough comparison of the results is provided later in this section.

As shown in Fig. 1, no risk reduction in the most critical path was achieved in about 25% of the experiments (498 out of 2000). This is not a big concern because the goal is to reduce the overall risk of the dependency risk graph and not necessarily the risks of the nodes in the most critical path, which are easily identified. Note, however, that when mitigation controls were applied to the nodes in the most critical path, the algorithm achieved better risk reduction results than when using well-known mitigation strategies such as implementing controls at the initiator node in the most critical path.

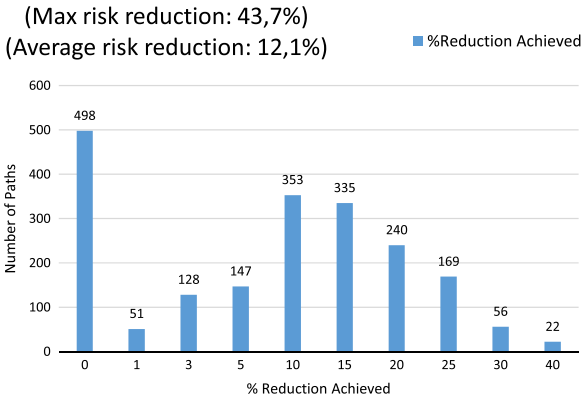


Fig. 1 – Risk reduction for the most critical path.

5.1.2. *Effect on the top 20 risk paths*

Fig. 2 shows that the highest risk reduction of 37.5% was achieved for the sum of all the risk values derived from the top 20 critical paths. The average risk reduction achieved in the graphs using the algorithm was 9.8%. In addition, note that only 12 graphs out of 2000 had structures that prevented the algorithm from achieving any risk reduction in the top 20 paths. Specifically, 99.6% of the algorithm applications achieved some risk mitigation and 47.3% of the applications achieved high risk reduction (10% or more).

These are interesting results. The top 20 most critical paths comprised 1000 nodes in each experiment. Although the paths share critical infrastructure nodes, being able to choose only three out of 50 nodes for risk mitigation in 20 paths is far more complex than choosing the correct nodes only in the most critical path comprising five nodes. Again, the algorithm had the best performance for the three different mitigation strategies tested.

By monitoring algorithm execution and the critical infrastructure node choices, it was possible to validate some feature analysis results. In particular, the risk mitigation algorithm mostly chooses critical infrastructure nodes with high values for all the centrality metrics (i.e., the intersection of all the high centrality sets for all the centrality metrics). The second most frequent combination of centrality types was nodes with high degree centrality (e.g., sinkholes) and high closeness centrality.

5.1.3. *Effect on the entire dependency risk graph*

Finally, the efficiency of the algorithm was analyzed when attempting to reduce the risk in the entire risk graph, not just for the top 20 paths. To accomplish this, the risk mitigation achieved using the node selection algorithm was evaluated for the sum of all the paths in the dependency graphs. Fig. 3 shows the results for 2000 experiments. The highest risk reduction achieved was 12.2% with an average risk reduction of 7.5% over all 2000 experiments. Interestingly, in 86.4% of the experiments, the risk reduction achieved was between 5% and 10% with an average of around 8.7%. Given the low dispersion of the results, statistical probability suggests that the algorithm output shows strong signs of uniformity and homogeneity. Again, the algorithm had the best performance of the three mitigation strategies tested.

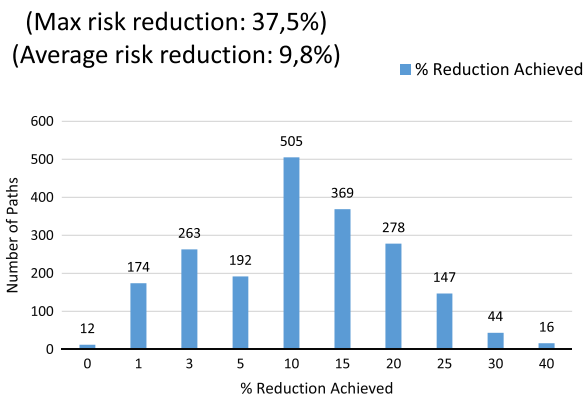


Fig. 2 – Risk reduction for the top 20 critical paths.

5.2. *Comparing alternate mitigation strategies*

This section compares the results obtained using the risk mitigation algorithm (Algorithm 1) with the results obtained using alternative mitigation mechanisms. The two alternative mitigation mechanisms used are based on well-known scenarios.

5.2.1. *Using initiators from the most critical paths*

The first strategy is based on the notion that initiator nodes are critical points of failure in dependency risk chains. The first alternative mitigation strategy is to implement mitigation controls in critical infrastructure nodes that initiate critical dependency paths. Thus, in the experiments, three out of the 50 critical infrastructure nodes (6% of the nodes in the graph) were chosen as initiators of the corresponding top three critical paths.

5.2.2. *Using nodes with high degree centrality (sinkholes and outbound)*

The second strategy is based on the commonly held notion that critical infrastructure nodes with high inbound and outbound degree centrality values (i.e., nodes with larger numbers of incoming and/or outgoing connections) significantly affect dependency risk paths. For this reason, the second mitigation strategy chooses the top 6% of nodes with the highest inbound and outbound degree centrality values to apply mitigation controls.

5.2.3. *Comparison results*

Table 3 shows the results obtained using all the mitigation strategies. The highest values achieved are highlighted using a bold font. Clearly, the decision making algorithm using information gain yields the best results. Using the top initiators strategy yields a higher average reduction risk for the top 20 critical paths, albeit only marginally.

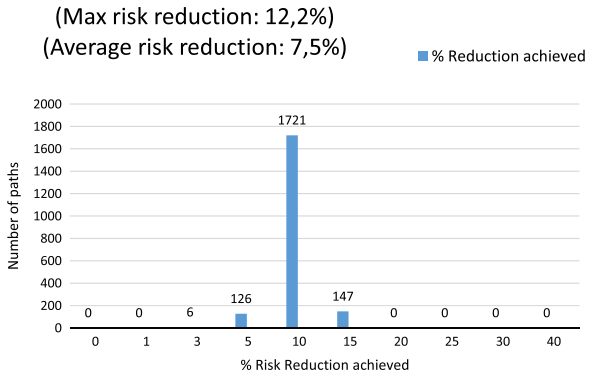


Fig. 3 – Risk reduction for the complete set of paths.

5.3. *Implementation details*

All the experiments were performed using a computer with an Intel Core i-7, 2.7 GHz processor with four cores and 16 GB RAM. An automatic dependency risk graph generator was written in Java to test the proposed risk mitigation strategies. The program randomly generates risk graphs and executes the mitigation algorithms on the generated graphs. Also, it computes the risk reduction percentage achieved using each of the mitigation mechanisms.

The Neo4j graph database [12] was used to implement the dependency analysis modeling tool due to its adaptability, scalability and efficiency [5,13]. Neo4j builds on the property graph model; nodes may have various labels and each label can serve as an informational entity. Nodes are connected via directed, typed relationships, and nodes and relationships can hold arbitrary properties (key-value pairs). Neo4j facilitates the representation of thousands of dependent critical infrastructures in a weighted, directed graph. The developed application is standalone: it creates a random dependency risk graph using the aforementioned constraints to model real-world critical infrastructure dependencies. It then precalculates all the necessary graph data and executes the selected risk mitigation algorithm. When an algorithm completes its execution, it outputs the percentage of risk reduction achieved in the dependency risk graph.

6. Conclusions

The methodology presented in this paper extends the dependency risk methodology of Kotzanikolaou et al. [7,8] by engaging graph centrality measures. The centrality measures are used as additional criteria to identify the critical infrastructure nodes in a dependency risk graph that significantly affect the critical risk paths in the graph and are, therefore, good candidates for applying mitigation controls. The results of the feature selection algorithms confirm that the most critical paths in a dependency risk graph tend to involve nodes with high centrality values. However, several centrality metrics are available and they contribute to node selection and overall risk mitigation in various degrees. For this reason, numerous simulation experiments were conducted to determine the best metrics and combinations of metrics. The

Table 3 – Comparison of mitigation strategy results.

Risk metrics	Information gain strategy	Top initiators strategy	Top sinkholes strategy
Most critical path	**43.7% (max)** **12.1% (avg)**	38.4% (max) 11.8% (avg)	34.5% (max) 10.3% (avg)
Top 20 critical paths	**37.5% (max)** 9.8% (avg)	28.7% (max) **10.0% (avg)**	29.8% (max) 7.3% (avg)
Entire graph	**12.2% (max)** **7.5% (avg)**	10.1% (max) 5.3% (avg)	10.8% (max) 6.7% (avg)

experimental results demonstrate that aggregating all the centrality sets to identify nodes with high overall centrality values yields a good, and intuitively appealing, mitigation strategy. Another result is that critical infrastructure nodes with high closeness and degree centrality values appear to have the highest impact on the overall risk of a dependency risk graph.

Still, simply choosing nodes from these sets is not always a viable choice because some dependency graphs may have no nodes in specific high centrality sets or there may be practical reasons that inhibit the application of controls at these nodes. For this reason, a risk mitigation mechanism was developed. The mechanism uses a decision tree built from node subsets with high centrality values extracted from a graph. The information gain approach is then used to choose subsets of high centrality nodes that best characterize the graph, enabling the dynamic identification of the best high centrality nodes for overall risk mitigation.

The risk mitigation algorithm achieved an average mitigation of 9.8% and maximum of 37.5% for the top 20 critical paths in more than 99% of the risk dependency graphs tested. The average risk mitigation achieved for entire graphs was 7.5% and the maximum mitigation was 12.2%. The risk mitigation achieved for the most critical path was higher than all the other mitigation mechanisms tested (43.7%). However, not all experiments were successful in mitigating the risk of the most critical path – in about 25.6% of the graphs tested, the algorithm did not mitigate any risk. This is understandable because the goal of the algorithm is to pinpoint critical infrastructure nodes that impact the entire dependency graph, not just the most critical path.

The methodology has certain limitations. If the goal is to protect critical infrastructure nodes in the most critical path, then mitigation controls should be applied at the initiator node (and, if affordable, to the second and third nodes in the path). While this approach guarantees maximum risk mitigation in every possible scenario, the large number of experiments in which the mitigation approach did not lower the overall risk prevents it from being applied exclusively to the most critical path of a graph. This is because the algorithm focuses on critical infrastructure nodes that affect the entire risk dependency graph, not just a single path.

However, if the focus is on mitigating the cumulative risk of a percentage of the top critical paths in a dependency risk graph, then, in almost every case tested, the information gain approach is more efficient than manually choosing nodes. This result holds for all three mitigation mechanisms evaluated in this research.

The single biggest advantage of the risk mitigation algorithm is that it can identify critical infrastructure nodes that may not be in the top critical path, but still significantly affect the overall risk in a dependency graph. While the exhaustive computation of the complete set of dependency risk paths may provide useful information and reveal critical infrastructure nodes that are underestimated or ignored, risk assessors can use the algorithm to examine complex scenarios where the manual selection of nodes for mitigation controls is infeasible.

A key limitation of the methodology is its reliance on data from previous critical infrastructure risk assessments. This is an inherent problem in all empirical risk approaches because they analyze dependencies based on previous incidents. However, critical infrastructure coordinators and regulators often collect data related to specific sectors such as energy and information and communications. National critical infrastructure protection agencies as well as some critical infrastructure asset owners and operators are also beginning to collect data that can be used in risk assessment methodologies such as the one presented in this paper.

Acknowledgments

This project was partially supported by the European Union's Seventh Framework Programme for Research, Technological Development and Demonstration under (Grant no. 312450.) The support from the European Commission is gratefully acknowledged.

REFERENCES

[1] N. Abramson, *Information Theory and Coding*, McGraw-Hill, New York, 1963.
[2] P. Bonacich, Power and centrality: A family of measures, *American Journal of Sociology*, vol. 92(5), pp. 1170–1182, 1987.
[3] European Commission, The Post 2015 Hyogo Framework for Action: Managing Risks to Achieve Resilience, COM(2014) 216 Final, Brussels, Belgium, 2014.
[4] E. Glover, G. Flake, S. Lawrence, W. Birmingham, A. Kruger, C. Giles and D. Pennock, Improving category specific web search by learning query modification, *Proceedings of the IEEE Symposium on Applications and the Internet*, pp. 23–32, 2001.
[5] S. Jouili and V. Vansteenberghe, An empirical comparison of graph databases, *Proceedings of the International Conference on Social Computing*, pp. 708–715, 2013.
[6] A. Karegowda, A. Manjunath and M. Jayaram, Comparative study of attribute selection using gain ratio and correlation based feature selection, *International Journal of Information Technology and Knowledge Management*, vol. 2(2), pp. 271–277, 2010.
[7] P. Kotzanikolaou, M. Theoharidou and D. Gritzalis, Assessing n^{th}-order dependencies between critical infrastructures, *International Journal of Critical Infrastructures*, vol. 9(1/2), pp. 93–110, 2013.
[8] P. Kotzanikolaou, M. Theoharidou and D. Gritzalis, Cascading effects of common-cause failures in critical infrastructures, in *Critical Infrastructure Protection VII*, J. Butts and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 171–182, 2013.
[9] P. Kotzanikolaou, M. Theoharidou and D. Gritzalis, Interdependencies between critical infrastructures: Analyzing the risk of cascading effects, in *Critical Information Infrastructure Security*, S. Bologna, B. Hammerli, D. Gritzalis and S. Wolthusen (Eds.), Springer-Verlag, Berlin Heidelberg, Germany, pp. 104–115, 2013.
[10] E. Luiijf, A. Nieuwenhuijs, M. Klaver, M. van Eeten and E. Cruz, Empirical findings on critical infrastructure dependencies in Europe, in *Critical Information Infrastructure Security*, R. Setola and S. Geretshuber (Eds.), Springer-Verlag, Berlin Heidelberg, Germany, pp. 302–310, 2009.
[11] Microsoft Developer Network, Feature Selection (Data Mining), Redmond, Washington (msdn.microsoft.com/en-us/library/ms175382.aspx), 2015.
[12] Neo Technology, Neo4j Graph Database, San Mateo, California ⟨www.neo4j.org⟩, 2014.

[13] B. Shao, H. Wang and Y. Xiao, Managing and mining large graphs: Systems and implementations, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 589–592, 2012.

[14] M. Theoharidou, P. Kotzanikolaou and D. Gritzalis, A multi-layer criticality assessment methodology based on interdependencies, *Computers and Security*, vol. 29(6), pp. 643–658, 2010.

[15] M. Theoharidou, P. Kotzanikolaou and D. Gritzalis, Risk assessment methodology for interdependent critical infrastructures, *International Journal of Risk Assessment and Management*, vol. 15(2/3), pp. 128–148, 2011.

[16] S. Ugurel, R. Krovetz and C. Giles, What's the code? Automatic classification of source code archives, *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 632–638, 2002.

[17] U.S. Department of Homeland Security, National Infrastructure Protection Plan (NIPP) 2013: Partnering for Critical Infrastructure Security and Resilience, Washington, DC, 2013.

[18] M. van Eeten, A. Nieuwenhuijs, E. Luiijf, M. Klaver and E. Cruz, The state and the threat of cascading failure across critical infrastructures: The implications of empirical evidence from media incident reports, *Public Administration*, vol. 89(2), pp. 381–400, 2011.

[19] Y. Yang and J. Pederson, A comparative study on feature selection in text categorization, *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412–420, 1997.