Efficiently computing the likelihoods of cyclically interdependent risk scenarios
Steve Muller, Carlo Harpes, Yves Le Traon, Sylvain Gombault, Jean-Marie Bonnin

To cite this version:
Steve Muller, Carlo Harpes, Yves Le Traon, Sylvain Gombault, Jean-Marie Bonnin. Efficiently computing the likelihoods of cyclically interdependent risk scenarios. Computers & Security, 2017, 64, pp.59 - 68. 10.1016/j.cose.2016.09.008 . hal-01427488

HAL Id: hal-01427488
https://hal.science/hal-01427488
Submitted on 26 Apr 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Efficiently computing the likelihoods of cyclically interdependent risk scenarios

Steve Mullera,b,c*, Carlo Harpes,a, Yves Le Traona,b, Sylvain Gombaultc, Jean-Marie Bonninb

laughs

Quantitative risk assessment provides a holistic view of risk in an organisation, which is, however, often biased by the fact that risk shared by several assets is encoded multiple times in a risk analysis. An apparent solution to this issue is to take all dependencies between assets into consideration when building a risk model. However, existing approaches rarely support cyclic dependencies, although assets that mutually rely on each other are encountered in many organisations, notably in critical infrastructures. To the best of our knowledge, no author has provided a provably efficient algorithm (in terms of the execution time) for computing the risk in such an organisation, notwithstanding that some heuristics exist.

This paper introduces the dependency-aware root cause (DARC) model, which is able to compute the risk resulting from a collection of root causes using a poly-time randomised algorithm, and concludes with a discussion on real-time risk monitoring, which DARC supports by design.

1. Introduction

Risk management constitutes an important aspect of decision taking, especially if the outcome is uncertain or has a large-scale impact on an organisation, which is why it forms a basis for many information security standards, including ISO/IEC 27xxx (International Organization for Standardization, 2014). Risks can be evaluated in two ways (Laboratory, 2012): qualitatively and quantitatively. Furthermore, some authors have suggested combining (Mangan et al., 2004) or converting (Cox et al., 2005) both methods to get better results, but this topic is beyond the scope of this paper.

In a qualitative assessment, risk scenarios are identified and then estimated in terms of probability and impact on a discrete (and often abstract) scale, which consists of some few values, such as “low”, “normal” and “high”. A previously defined set of unacceptable tuples (probability, impact) permits to distinguish between risk scenarios for which counter-measures need to be implemented (so as to reduce risk) and scenarios that are critical for an organisation (see Fig. 1).

In contrast, quantitative assessments express risk in terms of the potential damage that is inflicted to an organisation, e.g. in financial terms. So instead of qualifying a risk scenario as above, its likelihood and impact are expressed numerically; for example, by stating that “scenario X is estimated to occur very

* Corresponding author.

E-mail addresses: steve.muller@itrust.lu (S. Muller), harpes@itrust.lu (C. Harpes), yves.letraon@uni.lu (Y. Le Traon), sylvain.gombault@telecom-bretagne.eu (S. Gombault), jm.bonnin@telecom-bretagne.eu (J.-M. Bonnin).
5 years and when it does, it causes a loss of 10000 Euro", leading to an expected loss of 2000 Euro per year. Note that unlike above, quantitative risk analyses provide an integral view of risk faced by an organisation since all scenarios can be inspected and compared to one another at once, thanks to the numerical value of the expected losses. By consequence, the urgency of securing a specific asset in favour of another one, can be readily deduced, which is not so obvious to achieve in qualitative analyses.

There is a major drawback of many risk assessment methods regarding their support for interdependent assets. For instance, in the case of a server hosting a critical service, the efficiency of the responsible software is not only threatened on the one hand by vulnerabilities (bugs, security flaws) of the service itself, but also, on the other hand, by any down-time of the server. This relationship can be accounted for in several ways, the flaws and strengths of which are summarised in Fig. 2:

One can analyse the common risk scenario ("service not provided") for each of the assets ("server is down", "software has vulnerabilities"); however, the risk scenario in question is going to be accounted for multiple times in the global risk analysis, the outcome of which thereby becomes distorted.

Another option is to eliminate any redundancy from the risk assessment, for instance by assigning each scenario only to the most related asset. Now the global risk analysis features every risk scenario exactly once, but the view on an individual asset is no longer complete, since it does not take care of every possible risk scenario.

Finally, several authors (Brændeland et al., 2010; Breier, 2014; Pederson et al., 2006; Tong and Ban, 2014; Utne et al., 2011; Wang et al., 2008) incorporate the assets and/or risk scenarios together with their interdependencies into a hierarchical graph, based on which they deduce the risk for an asset, a group of assets or the whole organisation by reading off all subordinate risk scenarios. However, hardly any risk assessment model supports cycles in the dependency chain (e.g. A depends on B depends on C depends on A), although such cycles exist in every (sub-) system where the compromise of one component affects the whole (sub-) system. This is especially true in the context of Industrial Control Systems (ICS), where cascading effects can be of devastating order; for instance, the power grid and a voltage control system (requiring electricity to work) constitute an example of interdependent assets. As a second illustration, consider a poorly designed web service hosting confidential and valuable data (e.g. medical information) where the administrator can change any user passwords and can retrieve any of the regularly made backups of the user account database; then unauthorised access to the administrator interface allows an attacker to fetch a back-up, read out and disclose the user passwords, which again leads to unauthorised access. This scenario is depicted in Fig. 3: note the cycle "admin interface – backup location – user database" (dotted lines).

Some authors (Homer et al., 2009; Wang et al., 2008) propose a solution to deal with cyclic dependency graphs using graph unfolding techniques, but they fail at providing a complexity analysis for their methods.

This paper introduces a novel approach for computing the risk faced by cyclically dependent assets. Indeed, the proposed algorithm is based on a randomised (non-deterministic) simulation and – in contrast to other algorithms – provably efficient in terms of their execution time (which is important when doing real-time risk monitoring).

Section 2 presents related papers dealing with (cyclic) dependencies in risk assessments. The risk model used by the algorithm is defined in Section 3, along with the algorithm itself, whereas the proof of its correctness and running time can be found in the Appendix. The conducted experiments and their results are exposed in Section 4, Section 5 deliberates a generalisation of the model to also support more complex dependencies, and the paper closes with a conclusion in Section 6.

2. Related work

The model presented in this paper combines the concepts developed by various authors.
Breier (2014) encodes information security assets and their dependencies in a directed graph, where edges denote causal relations between nodes. The model supports the use of logical AND (for assets that depend on all parents) and OR (for assets that depend on one of the parents) operations. McQueen et al. (2006) also use attack graphs to encode dependencies (in their case, between services in a SCADA environment) and express risk as the time needed to compromise a component. Utne et al. (2011) focus on cascading effects in critical infrastructures by analysing the interdependencies between several high-level services (such as electricity) and their impact to the society. They also provide a framework to quantify the several magnitudes involved in the risk assessment, allowing an explicit computation of risk. Most interestingly, the risk itself is expressed as expected number of people affected by an incident.

Similarly, the risk assessment methodology supported by the Spanish government, MAGERIT (Amutio et al.), also deals with asset dependencies by embedding them into a graph. However, it does not directly link related assets, but their security objectives (such as confidentiality, integrity and availability), whenever they have an impact on each other. Rahmad et al. (2012) aim at improving on MAGERIT by combining it with thoughts from Fenz et al. (2009). They use an exhaustive list of threat scenarios instead of security objectives, which considerably increases the size of the model. This paper further generalises this concept to arbitrary and user-definable security incidents, which shows that quantitative assessments do not necessarily have to be expressed in financial terms.

To structure the assets (and thus the risk analysis) in a sensible way, Aubigny et al. (2011) establish a risk ontology for highly interdependent (critical) infrastructures, based on the estimation of quality of service (QoS). The proposed model supports risk prediction and incorporates a data structure which allows QoS information to be shared among interconnected infrastructures. In a similar spirit, Xin and Xiaofang (2014) classify assets into three layers, namely business, information and system. On the lowest layer, risk is computed traditionally as risk = impact × likelihood. Dependencies appear in the model as weighted impact added to the risk of dependent higher-level assets.

Moreover, in order to reduce the workload on risk assessors, several authors suggest models where risk can be computed from little input data. For instance, Homer et al. (2009) adapt the concepts and algorithms known from Bayesian networks to the realm of risk assessments and apply a graph unfolding technique to generalise the model for cyclic dependency graphs. The running time of this process is exponentially large in general, though, and thus only works for small or sparse graphs.

As a workaround, Wang et al. (2008) provide a simplified (and efficiently computable) probability metric for Bayesian networks, which they generalise to cyclic graphs, as well. However, the chosen metric does not properly take dependent events into account, so that the computed probabilities do not reflect reality.

Although Baiardi and Sgandurra (2013) work on attack trees rather than on causal graphs, the randomised algorithm they provide can be generalised in such a way that the probabilities of arbitrary causal graphs can be efficiently computed – which is done in this paper.

3. Modelling the risk analysis context

3.1. Risk assessment

The context of a risk analysis is primarily determined by the set of assets, which can be virtually anything of value to a company or institution. In terms of risk, each asset is characterised by its impact on business when one of its (security-related) properties can no longer be guaranteed. Such properties are called security aspects in the following, and include most notably the three notions (Bishop, 2012) below. Note that on the one hand, some of these aspects might not be applicable to certain assets. On the other hand, it is sometimes sensible to add further properties, or to be more specific about existing ones – especially if the impact considerably changes when doing so. For instance, one usually wants to distinguish between temporary unavailability, causing business interruption, and permanent loss, which may be fatal to business.

- Confidentiality – the assumption that sensible information is known only to a well-defined group of people.
- Integrity – the state that an asset is guaranteed to remain in a well-defined state.
- Availability – the property that an asset can be accessed in the way that was previously defined.

The impact itself is expressed in financial terms; this approach has the notable advantage that estimates have an objective meaning and can thus be easily compared to each other. More precisely, the impact is defined to be the financial damage caused by the threatened security aspect, or the amount of money necessary to recover back to the original state. That way, one can compute the loss expectancy (LE) which represents the total losses to be expected in a given period, typically a year:

\[
LE = \sum_{s \in \text{scenario}} \text{impact}(s) \cdot \text{likelihood}(s),
\]

where \( \text{likelihood}(s) \) denotes the number of times that scenario \( s \) is expected to occur in the given period (i.e., the expected frequency) – quantity usually to be estimated by a risk assessor, a methodology or a tool. This paper will present an algorithm for efficiently computing the likelihood function; see Section 3.5.

3.2. Dependencies

A considerable flaw in many risk management models is the lack of understanding of asset dependencies. Indeed, consider a hard disk hosting valuable data (and suppose, for the sake of the example, that no back-up is available), then a hard disk failure does not only require the physical disk to be replaced (which is cheap), but also implies the complete loss of core data (which may be business-ending). By consequence, dependency-unaware models do not give disk health monitoring the attention it deserves.

In fact, asset dependencies represent nothing else than relations of cause and consequence of security incidents on given assets. Causal graphs are thus a natural candidate for encoding
these relationships in a mathematical model. Recall that a
causal graph on a vertex set of events is a directed graph such
that two incidents are linked whenever they cause one another.

3.3. Likelihoods and probabilities

Whereas many approaches found in the literature (e.g. Breier,
2014; Liu et al., 2011; Loloi et al., 2012) use an abstract scale
(like “low”, “medium”, “high”) for describing the likelihood of
an event, this paper relies on concrete physical magnitudes that
support the direct use in a computation, see Fig. 4.

A reason why one prefers to use an abstract scale over a
number is because precise values are rarely known, so only a
rough estimate can be given. However, to make the assessment
task easier, one can still restrict to a given set of discrete
values to choose from for orientation, and interpolate to express
slight nuances. One such possible mapping is given in Fig. 5,
but can (and has to) be adapted to the setting in question.

Note the fundamental difference between “likelihood” and
“probability”. Probabilities are only meaningful when characterising a random event \( a \) priori: the event either happens
or it does not with a certain chance. Likelihoods, however,
express the \( a \) posteriori statistical occurrence (or expected fre-
quency) of events over time. Mathematically, probabilities are
unit-less and lie within \([0,1]\), whereas likelihoods can be ar-
bitrarily large and have as unit \( \frac{1}{\text{time}} \).

Both notions of “probability” and “likelihood” can be easily
confused, because they are somewhat related. It is meaning-
less to say that a risk scenario occurs with a certain probability,
though. Probabilities do not scale linearly and are bounded by
100\%, which is not in line with the common understanding of
risk. Indeed, consider the statement “there is a 10% chance of
fire”, then how to encode events that occur 100 times more of-
fen. If, however, one estimates the likelihood of fire to be “on
average once every 10 years”, then the expected damage is trivi-
ally \( 10^6 \text{EUR} \cdot \frac{1}{10^y} = 10000 \text{EUR/y} \).

The model introduced in this paper makes use of both
notions, that is, on the one hand, the likelihood of an incident
or risk scenario, and on the other hand, the probability that it
entails another (dependent) incident.

In order to distinguish between the two, write \( P \) for the prob-
bility measure and \( L \) for the likelihood.

3.4. The dependency-aware root cause (DARC) model

Pick a set of nodes \( V \), each representing a security incident. A
possible choice is to opt for \( V \subseteq A \times S \), where \( A \) is the set of
all assets and \( S \) the set of security properties. For instance,
\( S = \{ C, I, A \} \) could comprise confidentiality, integrity and avail-
ability threat scenarios. In that case, for readability, write \( a.s \)
instead of \((a,s)\in A \times S\) to denote the security aspect \( s \) of an
asset \( a \).

Let \( E \subseteq V \times V \) be the set of (directed) edges such that \((a,\beta)\in E \)
iff security incident \( a \) has an impact on incident \( \beta \). For read-
ability, write \( a \rightarrow \beta \) instead of \((a,\beta)\) if the set of edges \( E \) is
understood from the context. Illustrating the notation, the
example above can be rewritten in a very short and intuitive
form:

\[
\text{HDD} \cdot A \rightarrow \text{DATA} \cdot A, 
\]

which reads “if the availability of the hard-drive is compro-
mised, then so is the availability of any data stored on it”.

The tuple \((V, E)\) constitutes a directed graph which is hence-
forth referred to as the (asset) dependency graph. It is not required
to be acyclic.

Whereas manual estimation of the full probability distrib-
ution of the several incidents is theoretically possible, statistical
experiments in real-world systems are usually infeasible, for
it would require the simulation of threats on business pro-
cesses. Instead, the proposed model will use a slightly simplified
approach by basing itself on estimating the probability that a
particular incident causes another, independently of possible
other causes. In particular, the parent causes of an event are
related by a boolean OR operation, but the model can be ex-
tended to support arbitrary boolean formulas as well; see
Section 5. Formally, this describes a mapping \( p : E \rightarrow [0,1] \) where
\( p(\alpha \rightarrow \beta) \) denotes the probability that \( \beta \) is entailed by \( \alpha \). Note
that this is not the same as \( \Pr[\beta | \alpha] \), since \( \beta \) could also be en-
tailed by other events. \( p(\alpha \rightarrow \beta) \) is often denoted \( \Pr[\beta | \text{Ho}(\alpha)] \) or
\( \Pr[\beta | \text{set}(\alpha)] \) in the literature (Pearl, 2000).

Define a root cause to be a vertex without parent nodes in
the dependency graph. These are the causes for which an ex-
plicit likelihood needs to be specified later on, whereas for non-
root nodes it is deduced from the model.

An example of a dependency graph is depicted in Fig. 6: the
dge weights represent the values of \( p \), and \( A \) and \( I \) stand for
the availability and integrity properties of the assets, respec-
tively. Note how the model does not make a difference between
external threats (circled) and security properties of assets
(boxed).

3.5. Computing the probability distribution

The complexity of the DARC model lies in the fact that one is
interested in the probability that a certain sequence of events
cause each other, which is different from the problem of finding such a sequence. Indeed, for the former, one needs to determine all such sequences and compute the probability that one of them occurs. For this, simply listing those sequences and adding up their probabilities of occurrence yields wrong results, since some edges are accounted for twice.

3.5.1. The acyclic case
If the dependency graph is acyclic, a well-established theory can be used to describe the full probability distribution. Indeed, by its definition, the dependency graph together with the associated probability distribution constitutes a Bayesian network. This case has been extensively studied (Idika and Bhargava, 2012), notably in Poolsappasit et al. (2012) and Homer et al. (2009). This paper will thus focus on cyclic dependencies.

It is important to note that already in this simpler case, it is computationally infeasible to determine the full probability distribution of general Bayesian networks (Cooper, 1990). If one assumes further properties of the graph, efficient algorithms do exist, though (Zhang and Poole, 1994). A new approach will thus be required to support cyclic graphs as well.

3.5.2. The general case
For cyclically related security incidents, computing their likelihoods constitutes a more delicate problem than it seems. Based on how the model is defined, an event (i.e., a node) is not triggered multiple times throughout the course of the experiment, but only once. For a concrete instance of the random experiment, the issue consists in finding all events that are activated by any of the root causes. See for example Fig. 7: event E can be caused by C or F, but inspecting the situation in more detail, E is only caused by either of the two event chains A → B → C → E or F → E. In particular, E can only by triggered by C if C is not already triggered by the chain F → E → D → B → C.

By consequence, the probability that a risk scenario occurs cannot only be expressed by the probability of its direct parents, but has to involve all cycle-free paths from a root node. The computation effort for enumerating all such paths can be huge (in the worst case, namely in complete graphs, the runtime is exponential in the number of vertices), which is also why any efforts in finding an efficient deterministic algorithm failed. Instead, the randomised algorithm given in Algorithm 1 aims to give a good approximation; note that risk assessments as introduced in this paper do not require input data to be precise and are stable with respect to fluctuations.

The running time of Algorithm 1 is polynomial in its input data. More precisely, it is bounded by

$$O\left(n \cdot m \cdot \ln\left(\frac{2n}{\delta}\right) \epsilon^{-3}\right),$$

where $n$ is the number of vertices, $m$ is the number of edges, $\delta$ is the probability that the algorithm output is wrong and $\epsilon$ is an upper bound for the absolute error of the computed values. Observe the logarithmic dependency on $\delta$, which permits amplifying the algorithm accuracy without significantly increasing its running time.

<table>
<thead>
<tr>
<th>Algorithm 1 Compute probability matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Graph $G = (V, E)$ with root nodes $V_R \subseteq V$</td>
</tr>
<tr>
<td><strong>Input:</strong> Probability map $p : E \rightarrow [0, 1]$</td>
</tr>
<tr>
<td><strong>Input:</strong> $\epsilon &gt; 0$, $\delta &gt; 0$</td>
</tr>
<tr>
<td><strong>Output:</strong> Probabilities $C : V_R \times V \rightarrow [0, 1]$ that a root node causes a node, each value with absolute error at most $\epsilon$. The algorithm will fail with probability at most $\delta$.</td>
</tr>
</tbody>
</table>

1. $\gamma := \frac{1}{\sqrt{\pi \cdot \epsilon}}$
2. $N := \frac{n}{2} \ln \left(\frac{2n^2}{\delta}\right)$ where $n := |V|$
3. // $N$ is chosen large enough so that there are guaranteed bounds on the error probabilities — those are formally proven in Proposition 1.
4. // error probabilities — those are formally proven in Proposition 1.
5. for $(v_r, v) \in V_R \times V$ do
6. $C(v_r, v) \leftarrow 0.$
7. loop $N$ times
8. Sample a random graph $G'$ from $G$ according to $p$
9. for $v_r \in V_R$ do
10. for $v \in V(G')$ do
11. if $\exists$ path in $G'$ from $v_r$ to $v$ then
12. $C(v_r, v) \leftarrow C(v_r, v) + 1/N$
13. return $C$

The proof of correctness, running time and error probability is given in the Appendix.

3.6. Real-time risk monitoring
The DARC model paves the way for real-time risk monitoring in the sense that the likelihoods of the root nodes, which usually have to be estimated manually, can be automatically determined by external sources such as intrusion detection systems (IDS) or security information and event management (SIEM) appliances. Note that “real-time” denotes a process where an explicit bound on the running time is known.

Observe that the function $C : V_R \times V \rightarrow \mathbb{R}$ computed in Algorithm 1 only depends on the probability weights $p$ encoded into the graph, but not on the likelihoods of the root causes. By consequence, since the model (including $p$) is not supposed to change during the risk monitoring phase, the above

Fig. 6 – An example illustrating the representation of the DARC model as a graph.

Fig. 7 – A simple cyclic dependency graph.
algorithm is invoked once, namely after the model design phase. As such, \( C \) can be considered static.

Since \( C \) expects two arguments, it can be viewed as a two-dimensional matrix \( C \in \mathbb{R}^{W \times V} \) where each row represents a root node \( v \in V_R \) and each column an arbitrary node \( v \in V \). In fact, an entry of \( C \) denotes the probability that a given root node (row) entails any given node (column).

If the vector \( L \in \mathbb{R}^{V_R} \) denotes the (estimated) likelihoods of root causes, then the likelihoods \( L \) of all the nodes can be computed as

\[
L = L^T \cdot C,
\]

where \( \cdot \) denotes matrix multiplication and \( L^T \) the transpose of the vector \( L \). Moreover, let \( I \in \mathbb{R}^V \) be the vector that holds the direct impact caused by each node. The global risk can finally be written as

\[
\text{risk} = L^T \cdot C \cdot I.
\]

In a more explicit fashion,

\[
\text{risk} = \sum_{v \in V_R} \sum_{u \in V} L_u \cdot C(u, v) \cdot I(v).
\]

\( C \) is computed from the model by Algorithm 1. The impacts \( I \) are manually estimated by a risk assessor for every security incident (including root nodes) in the graph – the value 0 describes incidents without impact. The likelihoods \( L \) of the root causes are either manually estimated (if they are static) or dynamically monitored by external sources (IDS, SIEM, etc.).

4. Experiments

The algorithm is constructed in such a way that it can be interrupted at any point, yielding a less precise, but complete solution. More precisely, if it is aborted after \( \alpha N \) steps, for \( 0 < \alpha < 1 \), then the relative error \( \varepsilon \) will increase at most by a factor \( \alpha^{-3} \); this estimate follows directly from the definition of \( N \) (the number of simulation iterations) and is formally proven in Lemma 1 in the Appendix.

Moreover, since all simulations are run in an independent manner, they can be perfectly run in parallel (profiting from multi-threading capabilities of a CPU) or in a distributed way.

To test the performance of the algorithm on “average” graphs, dependency graphs have been generated uniformly at random. A typical risk analysis may cover up to 50 different assets, each of which generally encounters 3–5 threats, so a related graph is composed of a few hundred nodes. It is sensible to assume that nodes are not connected (on average) to more than a few edges, so a typical graph will consist of a few thousand edges at most.

The simulation was performed on a dual-core 2.5 GHz processor (i7-3537U). The results are depicted in Figs 8–10 – as expected, they reflect the running time computed in Proposition 1 in the Appendix. The precise numbers can be found in Table A1 in the Appendix.

In order to compare the performance of Algorithm 1 to other approaches, similar experiments have been conducted with
It turns out that the multiplication of those matrices is very fast and insignificant with evaluating the matrix product \( \epsilon \delta \) iterations time (s) simulation experiments

<table>
<thead>
<tr>
<th>( V )</th>
<th>( \epsilon )</th>
<th>( \delta )</th>
<th>Iterations</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.1</td>
<td>0.01</td>
<td>57290</td>
<td>1.1163832</td>
</tr>
<tr>
<td>200</td>
<td>0.1</td>
<td>0.01</td>
<td>62764</td>
<td>1.8782055</td>
</tr>
<tr>
<td>300</td>
<td>0.1</td>
<td>0.01</td>
<td>65966</td>
<td>3.4242201</td>
</tr>
<tr>
<td>400</td>
<td>0.1</td>
<td>0.01</td>
<td>68238</td>
<td>4.0489518</td>
</tr>
<tr>
<td>500</td>
<td>0.1</td>
<td>0.01</td>
<td>70000</td>
<td>5.2410411</td>
</tr>
</tbody>
</table>

Algorithm 2 Compute probability matrix in a recursive way
Input: Graph \( G = (V, E) \) with root nodes \( V_R \subset V \)
Input: Probability map \( p : \epsilon \rightarrow [0, 1] \)
Output: Probab. : \( V_R \times \epsilon \rightarrow [0, 1] \) that a root node causes another node.

1. if \( \epsilon \in E \) with \( 0 < \epsilon(p) < 1 \) then
2. Set \( \rho_0 := p \) but \( \rho_0(\epsilon) = 0 \). Recursively run on \( (G, \rho_0) \) to get output \( \rho_0 \).
3. Set \( \rho_1 := p \) but \( \rho_1(\epsilon) = 1 \). Recursively run on \( (G, \rho_1) \) to get output \( \rho_1 \).
4. For all \( \epsilon \in V_R, y \in V \): \( C(x, y) := C_0(x, y) \cdot (1 - \rho(\epsilon)) + C_1(x, y) \cdot \rho(\epsilon) \)
5. else
6. For all \( \epsilon \in V_R, y \in V \): \( C(x, y) := 1 \) if there is a path \( x \rightarrow y \), otherwise 0.

Unfortunately, such algorithms have exponential running time and take more than a few minutes already for small graphs (\( \mathcal{O}(2^n) \), \( \mathcal{O}(10^{10}) \)). All other attempts to solving the problem in a deterministic way resulted in similarly bad execution times. More advanced algorithms (see for instance Homer et al., 2009) suffer from the same problem.

The simulations revealed that \( \epsilon = 0.1 \) and \( \delta = 0.01 \) are sensible choices for typical dependency graphs (\( n \leq 1000, m \leq 5000 \)). Indeed, the running time is still relatively low (1–2 minutes), but the computed values lie very close (±5%) to the ones produced by \( \epsilon = 0.01 \) and \( \delta = 0.001 \), although the computation of the latter took much longer (38 hours). The values computed by setting \( \epsilon = 0.05 \) lie within ±5% of the high-precision results, but the running time is increased to 15 minutes; depending on the context, this may or may not be acceptable. In general, reducing \( \epsilon \) is more costly (in terms of running time) than reducing \( \delta \), but has a much greater impact on the precision of the output values.

Computing the risk faced by an organisation requires three steps: manually collecting impacts \( I \) and likelihoods \( L \), computing the probability matrix \( C \) using Algorithm 1, and finally evaluating the matrix product \( L \cdot C \cdot I \). It turns out that the multiplication of those matrices is very fast and insignificant with respect to the running time of Algorithm 1 for all choices of \( \epsilon < 0.5 \) and \( \delta < 0.1 \).

5. Extension to boolean formulas

The dependency graph is based on the concept of causality; that is, the parents of a node represent alternative causes, each of which can engender the consequential scenario. Formally, the dependency relationship of a vertex \( v \) and its parent nodes \( P_v \subset V(G) \) can be expressed as

\[
\rho(v) := \bigvee_{x \in P_v} I_x,
\]

where \( I_x \) denotes the boolean variable encoding whether the event \( x \) occurs or not.

The beauty of Algorithm 1 lies in the fact that it does not depend at all on the topology of the graph or on the form of the dependencies. In fact, generalising the “OR” relations to arbitrary boolean expressions \( \rho(x) \) is straightforward and does not change the main lines or the proof of the algorithm; yet
considerable adaptations have to be made in order to find whether a node is triggered or not (line 10 in the algorithm). For general boolean formulas the effort for computing the likelihoods is considerably higher, since a recursive search might no longer be possible.

In order to evaluate $A \land X$, one needs to evaluate both parents, including $X$ and thus $Y \land B$ and $Y$ (see Fig. 11). But $Y$ can only be evaluated if $A \land X$ is known already. It is not so clear how to proceed in such a case: here, one solution is to set the likelihood to zero for all non-reachable nodes, because the cycle can never be entered; however, this might not be sensible in all situations. A different theory, such as boolean satisfiability (Clote and Kranakis, 2013), is required in these matters.

The comparatively good running time of Algorithm 1 was due to the fact that evaluating the likelihood (sc. finding all cycle-free paths) can be implemented in an efficient way. For more complex boolean formulas this may not necessarily be the case (indeed, the boolean satisfiability problem\footnote{Given an arbitrary boolean formula $\rho$ on variables $x_1, \ldots, x_n$, the \texttt{SAT} problem consists in determining whether there is a value $\sigma \in \{0, 1\}^n$ for all of these variables $x$ such that $\rho(x_1 \equiv \sigma_1, \ldots, x_n \equiv \sigma_n) = 1$.} \texttt{SAT} is \texttt{NP}-complete (Clote and Kranakis, 2013)) so that deterministic (and possibly even error-free) algorithms could outperform the simulation variant.

6. Conclusion and outlook

This paper provides a simple and lightweight approach for encoding asset dependencies into a graph structure. Since that graph is not assumed to be acyclic (which allows to model situations where asset $A$ depends on $B$, which depends on $C$, which depends on $A$ again), the model can also be used in environments with interdependencies, such as in Industrial Control Systems (ICS) or Critical Infrastructures (CI). The major contribution of this piece of work (apart from the DARC model) is Algorithm 1, which computes the resulting risk in such a case (indeed, the boolean satisfiability problem consists in determining whether there is a path from $v_i$ to $v_j$ in the $i$-th random experiment. Observe that

$$\frac{1}{N} \sum_{i=1}^{N} E[X_i(u)] = E[X(u)] = P[X(u) = 1],$$

that is, the quantity approximated by the algorithm (left hand-side) equals the probability that node $v_i$ is reachable by $v_j$. So if the random experiments do not deviate too much from their expectations, the algorithm output is correct up to a certain relative error, which is determined in the following.

Define $\gamma$ and $N$ as in the algorithm. Note that $\gamma < \epsilon < 1$.

Fix $v_i \in V$. Suppose for now that $\mu(v_i) \geq \gamma$. Using a two-sided Chernoff bound (Motwani and Raghavan, 2010),

$$\mathbb{P}\left[\sum_{i=1}^{N} X_i(u) - N\mu(u) > N\mu(u)\epsilon\right] \leq 2\exp\left(-\frac{\epsilon^2}{3}N\gamma\right)$$

$$= \frac{\delta}{N}.$$ (by the choice of $N$)

**Fig. 11** – Endless loop in dependencies for general boolean formulas.
If however $\mu(v) \leq \gamma < \epsilon$, using a one-sided Chernoff bound,

$$
P\left[ \frac{1}{N} \sum_{i=1}^{n} X_i(v) > \epsilon \right] = P\left[ \frac{1}{N} \sum_{i=1}^{n} X_i(v) > 1 + \epsilon - \mu(v) \right] N \mu(v) \\
\leq 2 \exp\left( -\frac{\epsilon - \mu(v)^2}{\mu(v)} \right) N \mu(v) \\
\leq 2 \exp\left( -\frac{\epsilon - \gamma^2}{3 \gamma} \right) N \\
= 2 \exp\left( -\frac{\epsilon - \gamma^2}{3 \gamma} \ln(2n^2/\delta) \right).
$$

By the definition of $\gamma$ it holds that $\epsilon - \gamma^2 > 0$ and thus $\epsilon^2 > \frac{\delta}{n^2}$. To summarise, with probability at least $\delta n^{-2}$ the following two statements hold:

- If $\mu(v) \geq \gamma$ then the relative error of the random experiment is at most $\epsilon$; however, since $\mu(v) < 1$, this also implies that the absolute error $e(v) = \sum_{i=1}^{n} X_i(v) - N \mu(v)$ is at most $\epsilon$.
- If $\mu(v) \leq \gamma$ then the absolute error $e(v)$ is at most $\epsilon$.

Notes that the statements above hold for any fixed vertex $v$ and any fixed root node $v_0$. Using a union bound,

$$P[\exists v, \exists e(v) > \epsilon] \leq n^2 \cdot P[\epsilon(\nu_0) > \epsilon] = \delta$$

yielding the desired error probability for the algorithm.

Regarding the runtime, note that the inner for-loop can be implemented (e.g. using a breadth-first search) in linear time $O(m)$ for each of the at most $n$ root nodes, whereas the sampling requires time $O(m)$, resulting in a total execution time of $O(n \cdot m \cdot N)$.

Lemma 1. For fixed $\delta$, if Algorithm 1 is aborted after $aN$ iterations, for $0 < \alpha < 1$, then the relative error of the algorithm output increases at most by a factor $\alpha^2$.

Proof. Suppose Algorithm 1 requires $N_0$ iterations to achieve a relative error $\epsilon_0$. If one picked $\epsilon = \beta \epsilon_0$, it would require

$$
\frac{1 + \sqrt{\epsilon}}{\epsilon^2} \cdot 6 \log\frac{2n}{\delta} = \frac{1 + \sqrt{\beta \epsilon_0}}{\beta^2 \epsilon_0^2} \cdot 6 \log\frac{2n}{\delta} = \beta^3 \cdot \frac{1 + \sqrt{\epsilon}}{\epsilon} \cdot N_0 \leq \beta^3 \cdot N_0
$$

iterations instead, for any $0 < \beta < 1$. Since the outputs get more precise the longer the algorithm runs, running it precisely $\beta^3 N_0$ times will yield a relative error $\beta \epsilon_0$ (or even better). Setting $\beta = \alpha^2$ concludes the proof.

References


