Expert Systems With Applications 262 (2025) 125623

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

How to assess measurement capabilities of a security monitoring infrastructure and plan investment through a graph-based approach

Alessandro Palma[a,*], Andrea Sorrentino[b], Silvia Bonomi[a]

[a] *Department of Computer, Control, and Management Engineering "Antonio Ruberti", Sapienza University of Rome, Via Ariosto 25, Rome, 00185, Italy*
[b] *Leonardo S.p.A., Piazza Monte Grappa, 4, Rome, 00195, Italy*

A R T I C L E I N F O

Keywords:
Cybersecurity
Security monitoring
Security investment
Graph-based model
Monitoring infrastructure

A B S T R A C T

Security monitoring is a crucial activity in managing cybersecurity for any organization, as it plays a foundational role in various security processes and systems, such as risk identification and threat detection. To be effective, security monitoring is currently implemented by orchestrating multiple data sources to provide corrective actions promptly. Poor monitoring management can compromise an organization's cybersecurity posture and waste resources. This issue is further exacerbated by the fact that monitoring infrastructures are typically managed with a limited resource budget. This paper addresses the problem of supporting security experts in managing security infrastructures efficiently and effectively by considering the trade-off cost-benefit between using specific monitoring tools and the benefit of including them in the organization's infrastructure. To this aim, we introduce a graph-based model named *Metric Graph Model* (MGM) to represent dependencies between security metrics and the monitoring infrastructure. It is used to solve a set of security monitoring problems: (i) *Metrics Computability*, to assess the measurement capabilities of the monitoring infrastructure, (ii) *Instrument Redundancy*, to assess the utility of the instruments used for the monitoring, and (iii) *Cost-Bounded Constraint*, to identify the optimal monitoring infrastructure in terms of cost-benefit trade-off. We prove the NP-hardness of some of these problems, propose heuristics for solving them based on the Metric Graph Model and provide an experimental evaluation that shows their better performance than existing solutions. Finally, we present a usage scenario based on an instance of the Metric Graph Model derived from a state-of-the-art security metric taxonomy currently employed by organizations. It demonstrates how the proposed approach supports an administrator in optimizing the security monitoring infrastructure in terms of saving resources and speeding up the decision-making process.

1. Introduction

In the contemporary digital landscape, where systems and networks are always more complex and interconnected, preventing security threats is becoming paramount. In this scenario, security monitoring is one of the most critical activities while managing cybersecurity for every organization because it supports the security posture to provide mitigation actions in case of suspicious activities (Han, Choi, Choi, Lee, & Whinston, 2023). It is the first activity in security by design systems and its quality impacts all the following activities in cascade (Ahmad, Ong, Liew, & Norhashim, 2019; Von Solms & Van Niekerk, 2013), such as intrusion detection and risk assessment. To correctly monitor the cybersecurity situation and support the decision-making process, companies need to collect data and compute security metrics providing a quantitative evaluation of the current situation. Indeed, "*if you cannot measure it, you cannot improve it.*" is a quote from Lord Kelvin, also known as William Thompson stressing the importance of correctly identifying and managing metrics in every domain.

Applied to cybersecurity, this concept gains even more importance as the complexity of the situation under analysis is huge and heterogeneous variables play a relevant role. Among the monitoring systems currently used by enterprises, Security Information and Event Management (SIEM) systems (Kayhan, Agrawal, & Shivendu, 2023) collect and analyze security event data from various sources across an organization's IT infrastructure. They correlate events to identify potential security threats and provide real-time monitoring, threat detection, and incident response capabilities. Network Security Monitoring systems (Ghafir, Prenosil, Svoboda, & Hammoudeh, 2016) analyze network packets for anomalies, suspicious activities, and potential breaches to measure security metrics for risk assessment. Cloud Security Monitoring systems (Kamble & Bhutad, 2018) employ different sensors to identify and mitigate risks associated with cloud environments.

* Corresponding author.
E-mail address: palma@diag.uniroma1.it (A. Palma).

https://doi.org/10.1016/j.eswa.2024.125623
Received 5 July 2024; Received in revised form 23 October 2024; Accepted 23 October 2024
Available online 30 October 2024

Physical Security Management systems (Ashibani & Mahmoud, 2017) integrate and correlate data from various physical devices, such as surveillance cameras, and access control systems to detect the possible entry points for an attacker.

A first problem in this scenario is that these monitoring systems may include thousands of sensors and security monitoring involves more than one of these systems, thus making the monitoring infrastructure very large. This corresponds to hundreds of billions of dollars to manage monitoring infrastructure, even for medium-sized environments (Hayat et al., 2019), and planning its related investments becomes complex, especially due to external constraints such as limited budget and resources (Srinidhi, Yan, & Tayi, 2015).

In fact, many different sensors exist that measure security metrics, which must be considered in combination to reconstruct the monitoring situation and support the decision-making process. To this aim, metrics measurement is supported by instruments or software tools collecting necessary data from the environment. The selection of instrument instances is too often made by considering mainly technical aspects, consequently privileging the number of the used independent tools over their real contribution to the quality of the monitoring process (Ahmad et al., 2019; Bowen & Lupo, 2020). This raises a second emerging problem, which is the bad management of the monitoring infrastructure that may result in a waste of resources and investments. While a great effort exists in the literature to support the security monitoring infrastructure for attack detection (Ge, Hong, Guttmann, & Kim, 2017; George & Thampi, 2018; Vassilev, Sowinski-Mydlarz, Gasiorowski, Ouazzane, & Phipps, 2021) and data correlation (Hwoij, Khamaiseh, & Ababneh, 2021; Lavrova, 2016; Liu et al., 2020), it is still missing a formalization of the security monitoring problems and their solution for resource management and investment planning.

To address the reported problems and bridge the gap between comprehensive security monitoring and budgeted resources, this paper proposes a novel approach to support security administrators in managing the security monitoring infrastructure efficiently and effectively. To reach this goal, we formalize three security monitoring problems to assess (i) if the current collection of instruments allows the correct computation of a predefined set of security metrics (i.e., *Metric Computability* problem), (ii) if the available instruments are used inefficiently i.e., avoiding additional resources without getting any benefit (i.e., *Instrument Redundancy* problem) and (iii) the set of instruments to support the computation of security metrics according to a limited budget (i.e., *Cost-Bounded Constraint* problem). To address such problems, we introduce the *Metric Graph Model* (MGM), a graph-based representation of the existing relationships between security metrics, data needed to compute them, and instruments/tools that can be used to gather such data. We analyzed each problem's computational complexity, proposed heuristics based on the MGM, and studied their performance and scalability through an experimental evaluation. Finally, we employed the heuristics in a monitoring management system to support security administrators in making decisions about metrics and instruments for security monitoring. We contribute a usage scenario inspired by real-world needs that shows the capabilities of the proposed approach for the state-of-the-art taxonomy of security metrics. Summarizing, this work contributes the following:

- The design of MGM, a graph-based model to support security monitoring management;
- The formalization of three metric-related problems in security monitoring;
- Three heuristics solving the introduced problems based on the modeled graph;
- The experimental validation of the proposed heuristics to study their performance and scalability;
- The design of a system supporting security monitoring management and a usage scenario showing the capabilities of the proposed approach with open source code.[1]

The rest of this paper is organized as follows. Section 2 reports the related work on security monitoring and graph-based approaches. A motivating scenario and the overview of the approach are described in Section 3, while Section 4 presents the formalization of the system model and the three monitoring problems. Sections 5, 6, and 7 include the proofs of NP-hardness and the corresponding proposed heuristics for the three security monitoring problems. Section 8 presents the experimental validation of the heuristics, while Section 9 shows a usage scenario for security posture assessment. Finally, Section 10 highlights limitations and promising research directions, and Section 11 concludes the paper.

2. Related work

The related work is organized following two main areas that intersect with our contribution: research related to monitoring systems and graph-based approaches supporting them.

2.1. Monitoring approaches

Among the different monitoring systems and approaches existing in the literature (López Velásquez, Martínez Monterrubio, Sánchez Crespo, & Garcia Rosado, 2023), Liu and Lu (2003) highlight the importance of having cost-effective monitoring systems. They propose a distributed monitoring paradigm in which distributed agents detect updates without incurring heavy loading services. Given the complexity of monitoring, Swamynathan, Kannan, and Geetha (2006) explore a formal language to facilitate monitoring tasks in complex systems through a rule-based approach, while Coppolino, D'Antonio, Formicola, and Romano (2011) introduce an approach to oversee the activity and condition to monitor a dam environment. These works emphasize the importance of controlling and measuring critical areas of complex systems to ensure prompt alerts for any irregularities, thus high attention has been given to security monitoring. In fact, Chernov, Butakova, and Karpenko (2015) explain problems and tasks of monitoring security information, highlighting the insufficient performance when operating in networks with many sources of security events and the inability to correlate them.

To this aim, the literature increased the effort to support security monitoring, such as Hindy, Brosset, Bayne, Seeam, and Bellekens (2019) who adopt a Machine Learning solution to detect anomalies and classify the different failures (e.g. sensor failures), sabotage, and cyber-attacks (e.g. DoS and Spoofing). Similarly, Formicola, Di Pietro, Alsubaie, D'Antonio, and Marti (2014) propose a collector mechanism based on the mixed holistic reductionist methodology that models the relationships between functional components of critical infrastructures and the provided services to assess cyber risks. In the context of the Internet of Things (IoT), Lavrova (2016) proposes a mathematical framework for assessing and monitoring security incidents by integrating IoT event sources for comprehensive analysis, with Hwoij et al. (2021) elucidating IoT monitoring and its integration in smart cities. Finally, Liu et al. (2020) introduce a distributed intrusion detection framework for security monitoring, which focuses on integrating physical systems with information systems through the knowledge representation of the process system model.

All these systems focus on the detection and data correlation aspects of monitoring, while only a few works defined models for security monitoring infrastructure to manage its resources. Although a first step towards this problem is proposed by Armenia, Angelini, Nonino, Palombi, and Schlitzer (2021) who investigate the trade-off between security risks and investment, they do not consider monitoring in the investigation. Thus, a formalization of security monitoring problems

[1] https://github.com/ds-square/monitoring_mgm.

and their relation with investments is necessary to handle the "relatively slow progress in the theoretical formulation of the cybersecurity concepts" (Vassilev et al., 2021), that can help formalize appropriate solutions, as we propose in this paper. Different works found a promising solution in graph-based approaches to address this problem.

2.2. Graph-based monitoring

Among the graph-based approaches to support monitoring, Xie, Li, Ou, Liu, and Levy (2010) present Bayesian networks to model the security uncertainty. They model networks for attack structures, attacker actions, and alerts to calculate the conditional probability supporting the analysis of an attack. Differently, Bi and Zhang (2017) propose a graph-based framework to analyze the automation of energy system operations, highlighting the importance of monitoring the system's operational status in real-time and making decisions accordingly. They model smart grids as graphs and use state-of-the-art algorithms for the analyses (e.g., max-flow min-cut). Contrary to these works, we propose a graph model for security analysis that is not strictly related to a specific environment.

Other works, such as the one by George and Thampi (2018), propose a graphical model representing the vulnerability relations in the IoT network. They model nodes as vulnerabilities and edges as dependencies between them, so that paths represent attack steps enabling the analysis of attack path enumeration and hot-spot finding. Similarly, Ge et al. (2017) introduce a formal framework for graphical modeling and assessment of IoT security. They model a 3-layer attack graph based on the level of the IoT networks where links between layers define Attack Trees and evaluate security metrics, such as attack cost, node degree, and mean-time-to-compromise. These works support the security analysts in assessing the threats in the IoT environment (Ge et al., 2017; George & Thampi, 2018) and uncertainty modeling (Xie et al., 2010), differently to our proposal that addresses security monitoring from a broader point of view, not only attack-based.

To handle more comprehensive security monitoring, Vassilev et al. (2021) propose a solution for cyber-systems by using ontologies that include threat, event, situation, and security item concepts. They provide a framework for policy checking based on logical rules and address the formal security analysis. However, they do not inform about resource optimization for measurement capabilities, as we do in this paper. Additionally, Sheeraz et al. (2023) propose a comprehensive modular architecture of the SIEM system to support end users in making better decisions when selecting an SIEM system, including data correlation, visualization, analysis, and storage. Differently, Sabur, Chowdhary, Huang, and Alshamrani (2022) address the problem of distributing firewalls in the cloud environment to optimize the network resources. They use a divide-and-conquer approach based on network segmentation and attack graphs. In particular, they use attack graphs to analyze the network's critical points and to introduce firewalls accordingly. These graph-based models solve optimization problems as we also propose in this paper, but they focus on specific scenarios such as firewall (Sabur et al., 2022) and SIEM systems (Sheeraz et al., 2023), while we consider a broader perspective for security monitoring. Collins (2011) collects different analyses using graphs, demonstrating their application for scan detection, and identification of hitlist attackers and spammers, while Khaleel and Al-Shumam (2020) review the latest theoretical graph-based applications for information security, highlighting their usage for cryptography algorithms, network monitoring, and hardware security. Contrarily, to the best of the authors' knowledge, graph-based approaches to address monitoring infrastructure problems combined with investment optimization are still missing in the literature.

Table 1 reports a comparative analysis of the proposed work with the current state of the art of security monitoring. It highlights whether the existing works address security monitoring problems related to the management of (i) metrics an organization is capable of computing (*Metrics Computability*), (ii) instruments not providing any benefits to the monitoring infrastructure (*Instrument Redundancy*), and (iii) optimal investment of monitoring resources according to a limited budget (*Cost-Bounded Constraint)*. Additionally, it reports the methodology used by the different works, their support to security problems, application domain, and whether they provide open-source code. A first emerging aspect is that none of the works addresses the three security monitoring problems together, thus hindering a comprehensive assessment of monitoring management and investments. In fact, they provide a single perspective to assess the measurement capabilities of an organization, mostly related to anomaly detection. In contrast, this paper proposes heuristics tailored to support monitoring infrastructure management effectively considering different perspectives, addressing all three security monitoring problems, and thus supporting the decision-making process more thoroughly.

Another aspect is that graph-based modeling is predominant in the security monitoring field. Existing works model the system as a graph and use state-of-the-art algorithms to address specific problems. In contrast, we designed a novel graph-based model, namely the Metric Graph Model, and heuristics that outperform state-of-the-art algorithms as they are tailored for the monitoring problems (see Section 8). This difference is because most of the literature provides security support in terms of attack and anomaly detection, thus focusing on data aggregation and correlation and lacking support for monitoring infrastructure management. Only two works consider a higher-level perspective including security investments (Armenia et al., 2021) and policy validation (Vassilev et al., 2021). However, none of them consider a multi-perspective analysis including measurement, resource, and investment management, as we propose in this paper.

Finally, it is worth noting that half of the reported works apply only to specific domains (e.g., IoT, wireless sensors, and power grid) and there is a lack of open-source code for reproducibility and comparability. In contrast, we contribute an approach that is independent of the application domain and release open-source code.

3. Proposed approach

In this section, we present a motivating scenario illustrating the security monitoring management problem and provide an overview of the proposed solution for this problem.

3.1. Motivating scenario

Let us consider an organization that wants to assess its posture through *dependability* attributes (Laprie, 1992). Given the many metrics necessary to measure dependability — varying from the quality of services, reliability, and security — the organization uses a Security Information and Event Management (SIEM) system to evaluate its dependability. It collects, aggregates, stores, and correlates events generated by the infrastructure from multiple and heterogeneous sensors (González-Granadillo, González-Zarzosa, & Diaz, 2021). Fig. 1 reports the architecture of the SIEM which integrates six different monitored systems (Anastasov & Davcev, 2014):

(i) *perimeter device monitoring* to regulate traffic in the network: it includes firewalls, virtual private networks (VPNs), and intrusion detection and prevention systems (IDS/IPS).

(ii) *operating system monitoring*, to collect data from security systems, domain name system (DNS) servers, and file replication services.

(iii) *endpoint device monitoring*, to collect data from user activities and interconnected devices.

(iv) *application monitoring* on various applications such as databases, web server applications, and in-house applications to perform specific functions.

(v) *proxy monitoring* to collect valuable information about usage statistics and browsing behavior of endpoint users. For example, it helps monitor the length of packets exchanged through the proxy server.

Table 1
Comparison of the literature about monitoring approaches. The columns denote if the works addressed metrics computability, instrument redundancy, or cost-bounded constraint problems. Additionally, the table reports the methodology used by the existing works, their support to the security and domain application, and whether open source code is available.

	Metrics computability	Instrument redundancy	Cost-bounded constraint	Methodology	Security support	Application	Code
Formicola et al. (2014)	✓	×	×	Mixed holistic reductionist	Attack detection and risk assessment	Wireless sensors	×
Lavrova (2016)	×	✓	×	Graph-based algorithms	Data aggregation and correlation	IoT	×
Bi and Zhang (2017)	✓	×	×	Graph-based algorithms	Attack injection and network observability	Power grid	×
George and Thampi (2018)	×	×	×	Graph modeling	Attack detection and threat reduction	IoT	×
Hindy et al. (2019)	×	×	×	Machine learning	Anomaly detection	General	×
Liu et al. (2020)	✓	×	×	Gaussian mixture model	Intrusion detection	General	×
Armenia et al. (2021)	×	×	✓	Simulation models	Security investments	General	×
Hwoij et al. (2021)	×	×	×	Splunk-based architecture	Real-time monitoring	IoT	×
Vassilev et al. (2021)	×	×	✓	Ontology-based	Policy validation	General	×
Sheeraz et al. (2023)	×	×	×	SIEM architecture	SIEM design	General	×
Proposed work	✓	✓	✓	**Graph-based heuristics**	**Monitoring management and investments**	**General**	✓

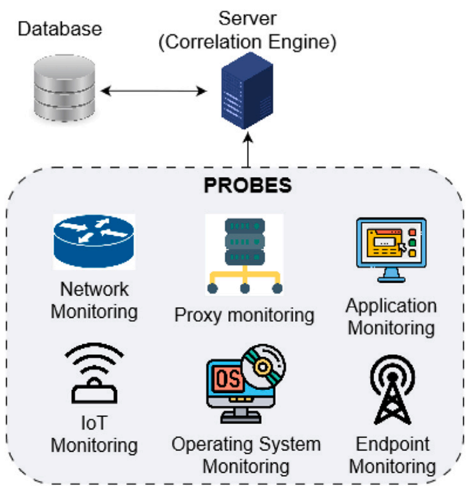


Fig. 1. Architecture of the SIEM for the motivating example.

(vi) *IoT monitoring* from embedded sensors and software to enable data collection, processing, and transmission.

The SIEM under consideration is employed with these six monitoring levels and has thousands of data sources, measured through instruments, each providing different metrics, that must be combined to collect probes of the overall environment (Sheeraz et al., 2023). Since each instrument has its own cost and the organization has a limited budget, the security administrator must provide an effective plan to optimize the company resources to assess the security posture through the SIEM. To accomplish this, she can access the asset inventory, which outlines the metrics measured by each instrument and provides a list of various tools and their costs. It is worth noting that a single instrument may measure multiple metrics, and different tools for the same instrument may vary in characteristics beyond their cost, such as accuracy and the metrics they measure.

In light of these considerations, deciding the optimal plan for monitoring the dependability situation of the organization is far from trivial as the administrator should manually analyze the asset inventory for the dependability metrics. To perform this task, she first checks whether the set of available instruments is enough to measure the dependability metrics (Problem 1).

Given that typically instruments are bought separately for the six monitoring systems in the SIEM, this causes their possible redundancy. For example, the proxy monitoring team may buy a network analyzer to measure the response time, while the IoT team buys an IDS that measures the response time as well. The security administrator must optimize the redundant instruments, i.e., the ones that can be safely removed to save the organization's resources (Problem 2).

Finally, the administrator must define an optimal plan founded on a limited budget. The budget does not always allow the inclusion of all necessary instruments and deciding which instrument to exclude is complex because of the trade-off between metrics measurement and instrument cost (Problem 3).

Addressing these problems is challenging for a human expert as they require considering and correlating different aspects (instruments, metrics, and costs) to determine optimal decisions. The problem is even harder if we consider that the monitoring system includes a large volume of heterogeneous data (Kang, Templeton, Lee, & Um, 2024). For this reason, this paper proposes a novel approach to address the above security monitoring problem, whose overview is described in the next section.

3.2. Overview of the approach

Fig. 2 reports the high-level overview of the proposed approach. The *inputs* are the monitoring infrastructure and the budget constraints. The former includes all the organization's instruments used for measurements, their instances (e.g., tools, software, hardware) and their costs. The latter defines constraints concerning the monitoring infrastructure (e.g., the maximum monetary investment and the maximum amount of resources).

The first step of the approach consists of *modeling* the monitoring infrastructure and its constraints. We design the Metric Graph Model (MGM), a graph-based representation of the dependencies between the instruments, their instances, and the metrics to measure. The graph-based structure facilitates the analysis of the relationships between the current infrastructure and the measurement capabilities.

We formalize three *monitoring problems* according to the MGM to assess measurement capabilities (Metrics Computability), optimize redundant instruments (Instrument Redundancy), and determine optimal

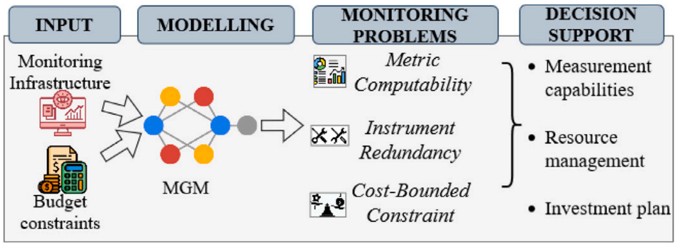


Fig. 2. High-level overview of the proposed approach.

investment plans (Cost-Bounded Constraint). These problems can be formulated as decision problems and we prove their NP-hardness. Given their complexity, we design heuristic algorithms to solve the monitoring problems efficiently and effectively.

Once suitable heuristics are developed for the security monitoring problems, their output is used for *decision support* about improvements to the monitoring infrastructure. The solution to the Metrics Computability problem informs about the metrics the organization is (not) capable of measuring with the current monitoring infrastructure, highlighting missing instruments for a complete measurement and assessing measurement capabilities. The solution to the Instrument Redundancy problem informs about the instruments that are unnecessary in the monitoring infrastructure and that can be safely removed to save resources. The solution to the Cost-Bounded Constraint problem determines the optimal monitoring infrastructure components for a limited budget.

4. System model

We consider an organization as an entity with a specific mission (e.g., providing energy to the city, producing goods, providing digital services to citizens). It relies on several processes running on a digital infrastructure i.e., a network of interconnected devices gathering, analyzing, processing, and storing information needed to implement digital services necessary to support business process provision. As a valuable asset, the organization wants to protect its digital infrastructure, thus it is interested in evaluating and assessing its security posture to support the decision-making process better concerning security resources utilization and investment. To monitor its cyber-security posture and develop its situation awareness, the organization relies on the computation and evaluation of a set of *security metrics* $M = \{m_1, m_2 \dots m_n\}$ where each metric m_j is an indicator quantifying certain security attributes based on certain scales (e.g., nominal, ordinal, interval) (Böhme & Freiling, 2008).

To quantify a metric we need to take *measurements* over the environment. This is done by using a set of specific *instruments* $I = \{i_1, i_2 \dots i_k\}$, i.e., devices or software sensors used to gauge the quantity of one or more metrics. Let us note that an instrument i_j may contribute to measuring different metrics.

Each instrument may have different *instances* each one characterized by a different implementation, with its own acquisition and deployment costs, and providing potentially different guarantees in terms of quality of measurement. We denote as T the set of pairs $\langle t_j, c(t_j)\rangle$ representing the available instrument instances where t_j represent a particular instance of the instrument and $c(t_j)$ is a function assigning a *cost* to the usage of t_j.

Let us note that some metrics may measure the same quantity but in different contexts (e.g., time measures both response time and repair time metrics). As a consequence, metrics and instruments within an environment may have interdependent relationships which, if not properly managed, can cause inefficiencies in security resource management. For this purpose, we introduce the notion of *instruments bunch* (or simply *bunch*) of a metric, that is the set of instruments that are used together to measure the metric.

Table 2
Summary of the system model notation.

Symbol	Definition
$M = \{m_1, \dots, m_n\}$	Set of metrics to measure
$B = \{b_1, \dots, b_x\}$	Set of instrument bunches
$I = \{i_1, \dots, i_k\}$	Set of instrument in the infrastructure
$T = \{t_1, \dots, t_h\}$	Set of instrument instances in the infrastructure
MGM = (V,E)	Metric graph model with vertices V and edges E
MC	Metric computability problem
IR	Instrument redundancy problem
CBC	Cost-bounded constraint problem
$\mathcal{U}$	Universe in the (weighted) set cover formulation
$\mathcal{S}$	Collection of sets in the (weighted) set cover formulation
W	Maximum budget constraint

Definition 1. An *instrument bunch B* for a metric *m*, denoted as B_m is a set of instruments $i_1, i_2 \dots i_k$ that, *all together*, are necessary to measure *m*.

Example 1. Let us consider an organization that wants to assess its posture through *dependability* attributes (Laprie, 1992). Let *MTTF*, *MTBF*, and *MTTR* be the mean time to failure, mean time between failures, and mean time to repair respectively. Dependability is composed of: (i) Reliability (i.e., continuity of correct service), measured as $MTTF$; (ii) Availability (i.e., readiness for correct service), measured as $\frac{MTTF}{MTBF}$; (iii) Maintainability (i.e., the ability for easy maintenance and repair), measured as $MTTR$; (iv) Safety (i.e., absence of catastrophic consequences on the environment), measured as the number of incidents in a given period; (v) Integrity (i.e., absence of improper system alteration), measured as the number of service shutdowns in a given period. The metrics measuring dependability are MTTF, MTBF, MTTR, number of incidents, and number of shutdowns. The organization has three instruments: a *system log* to detect shutdown failures, a *clock* to measure the time, and an *antivirus* tool to protect against malware. The system log can be *syslog-ng* tool with a cost of 3k$ or the one provided by Windows, *WinSysLog* with a cost of 0k$. The clock can be globally synchronized (i.e., *UTC*) with a cost of 4k$ per year or *locally managed* with a cost of 7k$ per year. The antivirus can be chosen between *Norton* and *Avast*, with a cost of 5k$ and 9k$, respectively. Although Avast has a higher cost, it also integrates a syslog. The metrics and instruments inventories are reported in Table 3. According to the definition of bunch, *incident log* and *clock* are the instruments belonging to the bunch b_1 which is used to measure MTTF, MTBF, and MTTR. The number of shutdowns is measurable with a bunch b_2 including only the system log, while the antivirus belongs to a bunch b_3.

In the rest of this section, we define the Metrics Graph Model and the three monitoring problems formally. Table 2 reports the notation used for the formalization.

4.1. Metrics graph model

Let us note that metrics, instruments, and bunches are related between them. In particular, a metric $m_j \in M$ is measurable by one or more bunches $b_1, \dots, b_x \in B$ and we call this relation *disposition*.

Table 3
Metrics inventory (on the left) and instruments inventories (on the right) related to Example 1.

Metrics inventory	Instruments inventory	
	Instrument	**Cost**
MTTF	clock UTC	4k$
MTBF	clock local	7k$
MTTR	syslog (ng)	3k$
num. incidents	syslog (win)	0$
num. shutdowns	antivirus (Norton)	5k$
	antivirus (Avast)	9k$

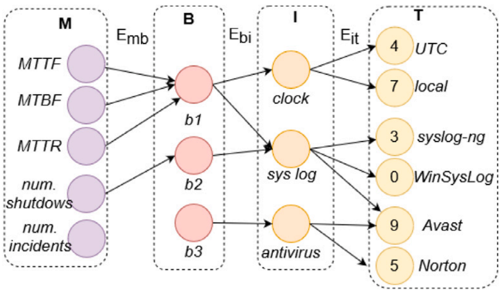


Fig. 3. Representation of the Metrics Graph Model for dependability.

A bunch $b_j \in B$ is composed by one or more instruments $i_1, \dots, i_k \in I$ and we call this relation *composition*.

An instrument $i_j \in I$ is implemented in one or more instances $t_1, \dots, t_h \in T$ and we call this relation *implementation*.

In order to capture these relationships and consider them while supporting security experts in the decision-making process, we contribute MGM, a graph-based representation of metrics, instruments, bunches, and their relationships. In particular, each instrument instance node in T has an associated weight corresponding to its cost.

Definition 2 (*MGM*)**.** A *Metrics Graph Model* (MGM) is a directed multipartite node-weighted graph $MGM = (V, E)$ where $V = \{M \cup B \cup I \cup T\}$ and $E = \{E_{mb} \cup E_{bi} \cup E_{it}\}$ with

- E_{mb} being the set of edges following from the *deploy* relation (i.e., the set of directed edges e_{jk} such that metric m_j is measurable by instrument bunch B_k);
- E_{bi} being the set of edges following from the *composition* relation (i.e., the set of directed edges e_{jk} such that the instrument bunch B_j includes the instrument i_k);
- E_{it} being the set of edges following from the *implementation* relation (i.e., the set of directed edges e_{jk} such that the instrument i_j has instance t_k).

Example 2. Let us consider again the dependability equipment of Example 1 (Table 3). The corresponding MGM is the one depicted in Fig. 3, in which the partitions and their bipartite relations are shown and annotated. The numbers in the nodes of partition T represent the costs of instrument instances.

4.2. Problem statement

Our scope is to support security experts in (i) evaluating the level of suitability of the current *security monitoring infrastructure* (i.e., the set of instrument instances currently deployed in the organization) with respect to the security posture evaluation needs (i.e., the set of security metrics needed to determine the current security posture) and (ii) planning for investment on instrument resources. To this aim, we leverage the MGM and formalize three more specific sub-problems.

The first problem (*Metrics Computability*) deals with the identification of the metrics that can be measured with a valid instrument, where the validity of an instrument is the existence of at least one instance of it. Indeed, if a metric has not a valid instrument, then it cannot be measured.

Problem 1 (*Metrics Computability Problem MC*)**.** Let M' be the set of metrics that the organization wants to measure (with $M' \subseteq M$), let T' be the set of instrument instances actually used by the organization, and let $MGM = (V, E)$ be organization metrics graph model (where $M', T' \subset V$). Identify all the metrics $m' \in M'$ that can be computed with the actually available instrument instances.

Let us note that an efficient security monitoring infrastructure can compute all metrics of interest without wasting resources. Thus, for the organization is important to avoid redundancy intended as the overuse of the available resources. To this aim, we define the *Instrument redundancy* problem.

Problem 2 (*Instruments Redundancy Problem IR*)**.** Let M' be the set of metrics that the organization wants to measure (with $M' \subseteq M$), let I be the set of instruments of the security monitoring infrastructure, and let $MGM = (V, E)$ be the organization metrics graph model (where $M', I \subset V$). Identify the *minimum subset* of instruments $I' \subset I$ that allows the computation of all the metrics in M'.

Finally, we introduce the *Cost-Bounded constraint* problem that improves resource utilization by considering the cost of the entire equipment needed to measure the security posture.

Problem 3 (*Cost-Bounded Constraint Problem CBC*)**.** Let M' be the set of metrics that the organization wants to measure (with $M' \subseteq M$), let T be the set of instrument instances of the security monitoring infrastructure, and let $MGM = (V, E)$ be the organization metrics graph model (where $M', T \subset V$). Let W be the budget available to the organization for its security monitoring infrastructure. Identify the subset of instrument instances $T' \subset T$ with *minimum cost* that allows the computation of all the metrics in M', without exceeding the given budget W.

Example 3. Let us consider the dependability equipment of Example 1 and the corresponding MGM of Example 2. The solution to the MC problem identifies that the organization cannot measure the number of incidents, thus hindering complete dependability monitoring. The solution to the IR problem supports the administrators in identifying that the antivirus instrument is redundant, therefore the organization can save 5/9k$. Finally, the solution to the CBC problem identifies the optimal investment of instruments and tools to measure all the measurable metrics. For the sake of the example, let us consider a budget of 6k$, then the optimal investment is the UTC clock and WinSysLog. With the remaining budget of 2k$, the administrator can propose an incident log to measure the number of incidents and complete dependability monitoring.

5. Metrics computability problem

Let us recall that the MC problem aims to identify the metrics that can be computed with the available instrument instances. To show the complexity of the problem, we can consider its corresponding decision problem which is evaluating whether the instrument instances currently deployed and used inside the organization are enough to satisfy the monitoring needs through a set of identified metrics M'.

Leveraging the MGM, this problem can be translated into a particular traversal of the MGM. In the following, we will show that the MC problem is in the complexity class P and then we propose an algorithm to visit the MGM and solve the MC problem efficiently.

Theorem 1. $MC \in P$.

Proof. To prove the claim we simply show a deterministic polynomial algorithm that leverages the MGM to solve the MC problem. The pseudo-code of the algorithm is shown in Algorithm 1. The basic idea is the following:

- **Step 1**: check that all the metrics we are interested in are modeled in the MGM representing the current equipment status of the organization. If not, the output of the algorithm is the value `false` as there is at least one metric not represented in the MGM and thus not computable with the currently available instruments.
- **Step 2**: starting from the available instances in T' we prune the MGM by checking families of available instruments we can use in the organization and we remove edges and instrument nodes that are not currently available. At the same time, we check if unavailable instruments are part of an instrument bunch to effectively take the measure and, in this case, it also removes the instrument bunch node from the MGM.
- **Step 3:** check that all metrics have a path in the pruned MGM to an instrument instance. This is done by running the Depth First Search (DFS) (Tarjan, 1972) algorithm and checking if at least one leaf obtained from the spanning tree is an instrument instance. The algorithm terminates when it checks all the metrics in M' with a positive assessment or as soon as it identifies a metric that is not computable.

Algorithm 1: Polynomial time decider for metrics computability

```
Input: The organization metrics graph model $MGM = (V, E)$ where
       $V = (M \cup B \cup I \cup T)$ and $E = (E_{mb} \cup E_{bi} \cup E_{it})$
Input: The set of target metrics $M'$
Input: The set of available instruments instances $T'$
Output: A Boolean value: true if $M'$ can be computed starting from $T'$, false
        otherwise

if $M' \not\subseteq V$ then
    return false;
else
                                   ▷ Prune the MGM due to unavailable instrument instances
    for $e_{j,k} \in E_{it}$ do
        if $v_k \notin T'$ then
            $E_{it} \leftarrow E_{it} \setminus \{e_{j,k}\}$;          ▷ Check unavailable instruments
        end
    end
    for $v_i \in I$ do
        if out_degree($v_i$) == 0 then
            $I \leftarrow I \setminus \{v_i\}$;    ▷ Remove Instruments without available instances
        end
        for $e_{j,i} \in E_{bi}$ do
            $B \leftarrow B \setminus \{v_j\}$;    ▷ Remove bunches requiring unavailable instances
        end
    end
    for $m_i \in M'$ do
                                                   ▷ Check metrics computability
        $T_{m_i} \leftarrow$ compute_DFS($m_i$);
        Let $L_{T_{mi}}$ be the leaves of $T_{m_i}$;
        if $L_{T_{mi}} \cap T' == \emptyset$ then
            return false;
        end
    end
    return true;
end
```

Let us note that the correctness of the algorithm directly follows from the definition of the MGM and the DFS algorithm. □

The time complexity derives from the highest complexity among the three steps. The procedure in step 1 is a repeated search over the set of metrics M, the one in step 2 is a repeated search over the set of instruments I and their instances T, while step 3 is a DFS on the whole MGM. Consequently, the procedures in step 1 and step 2 have a time complexity of O(M) and O($I + T$), respectively. In contrast, the time complexity of step 3 is O($M \cdot (V + E)$) since we perform a graph traversal through DFS (O($V + E$)) per each metric (O(M)), and it dominates the time complexity of the whole algorithm. Thus, we can conclude that the time complexity of Algorithm 1 is O($M \cdot (V + E)$), where $V = \{M \cup B \cup I \cup T\}$ by the definition of the MGM.

6. Instruments redundancy problem

Let us recall that the IR problem deals with the identification of the minimal subset of instruments I' that allows computing a set of metrics of interest M'. Differently from the MC problem, we will show that IR is an NP-hard problem. As such, we leverage a heuristic to solve it.

Theorem 2. *The IR problem is NP-hard.*

Proof. To prove our claim, let us simply note that the IR problem is basically a *minimum set cover problem* (also known as *min set cover*) (Karp, 1972). It considers a set of elements $\mathcal{U}$ and a collection S of sets whose union is equal to $\mathcal{U}$, and the problem is finding a set covering that uses the fewest sets. More formally:

Definition 3 (*Set Cover*). Given a set of elements $\{e_1, e_2, \ldots, e_n\}$ (called *universe*) and a collection S of m sets whose union equals the universe, the set cover problem is to identify the smallest sub-collection of S whose union equals the universe.

Min set cover is NP-complete as a decision problem (Bernhard & Vygen, 2008). Thus, to show the reduction of the IR problem to min set cover, we formulate it as a decision problem, that is: "decide whether there exists a minimum subset of instruments such that all the metrics nodes are connected to instruments nodes".

Now let us define the instruments redundancy problem considering the set M of metrics $\{m_1, \ldots, m_n\}$, and the set B of instrument bunches $B_1, \ldots, B_k$, such that the bunch B_i measures the subset of metrics $M'_i \subseteq M$. Let us note that, due to the construction of the MGM and the one-to-many relationship between bunches and instruments, finding the minimum set of Instruments I' can be trivially achieved by finding the minimum set of bunches B'.

According to this definition, we can associate each instrument bunch B_i with one subset of metrics M'_i to optimize the coverage. Thus, we map each metric $m_i \in M$ to an element of $x_i \in X$, and each subset of metric $M'_i \in M$ to a subset $S_i \in S$. As a consequence, if there exists a solution for the min set cover (i.e., all the elements of X can be covered with k subsets of S), then there exist k bunches covering all the metrics (i.e., a solution for the IR problem). Vice versa, if there exists a solution for the IR problem (i.e., all the metrics in M can be measured with k bunches), then there exist k subsets in S covering all the elements in X (i.e., a solution for the min set cover). □

A Heuristic algorithm for the IR problem. In the literature, there exist several heuristics for the min-set-cover that propose a Lagrangian relaxation and subgradient optimization heuristic (e.g., Beasley (1990) and Ceria, Nobili, and Sassano (1998)). Among the others, we leverage the solution proposed by Dasgupta, Papadimitriou, and Vazirani (2008) because it best suits the MGM structure. The idea of our algorithm is to proceed greedily, adding one set at a time to the set cover until every metric is "covered" by at least one instrument bunch. Thus, we choose the set that covers the most elements that remain uncovered at each ensuing step. Algorithm 2 reports the pseudo-code.

It first sorts the instrument bunches by in-degree, so that it chooses the set that covers the most elements as part of the solution (through the *select&remove* function). Upon all the metrics being covered by a subset of instrument bunches, the algorithm returns their minimal subset, and thus all the redundant instruments that are not in the minimal set. Note that the function *get_Instruments*(E_{bi}, b_j) returns the instruments belonging to bunch b_j.

The time complexity of Algorithm 2 is polynomial in the number of metrics (M) and instruments (I). In fact, we can perform each

Algorithm 2: Instrument Redundancy

```
Input: The organization metrics graph model $MGM = (V, E)$ where
       $V = (M \cup B \cup I \cup T)$ and $E = (E_{mb} \cup E_{bi} \cup E_{it})$
Input: The set of target metrics $M'$
Output: A set of Instrument $I'$

$B_{ord} \leftarrow$ sort_by_Indegree($B$);
$B' \leftarrow \emptyset$;
while $M' \neq \emptyset$ do
    $B_j \leftarrow$ select&remove_first($B_{ord}$);
    $B' \leftarrow B' \cup B_j$                ▷ Include set $B_j$ into the set cover ;
    $M_{temp} = \{m_i \in M' \mid \exists\ e_{i,j} \in E_{mb} \ \wedge \ v_j \in B\}$;
    $M' \leftarrow M' \setminus M_{temp}$;
end
$I' \leftarrow \emptyset$;
for $b_j \in B'$ do
    $I_{temp} \leftarrow$ get_Instruments($E_{bi}, b_j$);
    $I' \leftarrow I' \cup I_{temp}$;
end
return $I'$
```

iteration inside the while loop in time O($M \cdot B$) because we select and remove an instrument bunch for each iteration on the metrics. The for loop has a time complexity of O($B \cdot I$) because we select the instruments for each iteration on the bunches. Given that the number of instrument bunches (B) is at most equal to the number of metrics (M) by definition of MGM, then the upper bound for the time complexity of the proposed heuristic is $O(M^2 + M \cdot I)$. Concerning the approximation, Algorithm 2 is proven to be an O(log n)-approximation of the optimal solution (Dasgupta et al., 2008). For the scope of the instrument redundancy problem, such an approximation is reasonable because the aim is to inform about the most relevant instrument bunch, and a log approximation does not significantly change the exact solution, as we have shown in the experimental evaluation in Section 8.

7. Cost-bounded constraint problem

The CBC supports the planning of investments as it deals with identifying the subset of instrument instances T' with *minimum cost* that allows the computation of all the metrics in M'. To solve this problem, we enriched the MGM with a cost function $W(t_j)$ that assigns to each instrument instance $t_j \in T$ a cost e.g., the cost of buying the instance t_j or the cost of using the instance t_j. Not surprisingly, also the CBC problem can be proven to be NP-hard, thus we propose a heuristic to solve it.

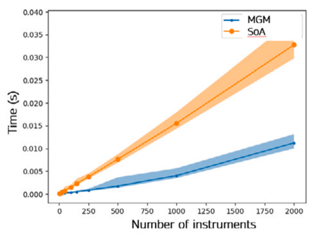
Theorem 3. *The CBC problem is NP-hard.*

Proof. The CBC problem can be reduced to the weighted set cover problem (Vazirani, 2001) i.e., the problem of minimum set cover with the addition of costs associated with each set.

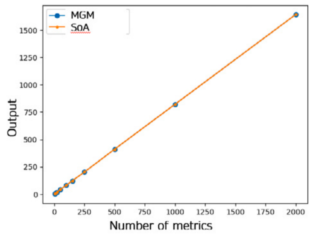
Let $\mathcal{U}$ be the set of all the elements, $\mathcal{S}$ a collection of sets whose union is equal to $\mathcal{U}$ and each one with its weight, and w the total weight. The problem deals with deciding whether there exists a minimum weight sub-collection of $\mathcal{S}$ whose union is $\mathcal{U}$ and total weight w. This problem is known to be NP-complete (Vazirani, 2001), and we leverage it to define the less costly subset of instrument instances to measure all the metrics.

We map the weighted set cover problem to the cost-bounded constraint one, where $\mathcal{U}$ corresponds to the set of metrics and $\mathcal{S}$ to the subsets of metrics covered by each bunch, each one with cost corresponding to the sum of the instrument instances belonging to that bunch. Through this mapping, if a solution exists for the weighted set cover, it also exists for the cost-bounded constraint, and vice versa. □

A Heuristic algorithm for the CBC problem. While several heuristics exist for min-set-cover problems, few coped with the weighted-set-cover. Vazirani (2001) proposes an f-approximation of the problem



(a) Performance experiments for MC problem.



(b) Validation experiments for MC problem.

Fig. 4. Performance and validation experiments of the metrics computability heuristic and path existence algorithm.

where f is the maximum number of occurrences of an element in different sets. Vasko, Lu, and Zyma (2016) propose a comparison of greedy heuristics for the weighted-set-cover problem, showing that avoiding redundancy significantly improves performance with respect to classical approaches. Such heuristics focus on variations of the cost function to find the optimal one by leveraging the integer linear programming version of the problem. In contrast, Golab, Korn, Li, Saha, and Srivastava (2015) introduce a generalization of weighted-set-cover in which they add both cost and size constraints.

In this paper, we leverage the graph-based structure of the MGM to solve the CBC problem. We propose a novel heuristic that uses the information aggregated into each instrument bunch (i.e., how many metrics a given bunch may measure and how costly are the instrument instances in it). The pseudo-code of the algorithm solving the cost-bounded constraint problem is Algorithm 3.

The idea of the algorithm is to choose the instrument bunches with the smallest cost (*aggregate_cost* variable) and the highest number of measured metrics (*covered_metrics* variable) as part of the solution, i.e., it tends to optimize the trade-off cost-coverage (expressed in the *if-condition* inside the *for-loop*). To do so, we sort the instrument bunches in ascending order per cost and descending order per number of covered metrics. Then, the selected bunch is removed together with all its connected nodes and edges (i.e., the connected metric, instrument, and instances nodes). Finally, the aggregated cost of all the removed instrument instances is subtracted from the unspent budget and the procedure is repeated until the cost constraint is violated or the graph is fully visited.

In the code, the function *get_connected_metrics(v_j)* returns the metrics nodes connected to v_j, while *connected(v_j)* all the connected components of v_j. The function *compute_aggregate_cost(v_j)* returns the cost associated with a specific instrument bunch v_j. Finally, the algorithm returns the instrument bunches respecting the cost constraint.

The proposed algorithm reduces the complexity of naive solutions because we do not consider any combination of instrument bunches, which requires a time complexity of O(B!) to evaluate all possible permutations without repetition of paths involving each instrument bunch. Rather we start by considering the less costly bunches and maximizing the covered metrics. In this way, at each step of the algorithm, we consider the subgraph obtained by removing the chosen instrument bunches, which reduces the graph size at each iteration, guaranteeing the time complexity of Algorithm 3 of O(2^B) dominated by the possible combinations of instrument bunches.

8. Experimental evaluation

In this section, we evaluate the proposed solutions for MC, IR, and CBC problems and compare them with other state-of-the-art heuristics that can be used to solve the graph traversal, min set cover, and weighted set cover problems respectively. In particular, we evaluated the computation time and the quality of the output by comparing

Algorithm 3: Cost-Bounded Constraint

```
Input: The organization metrics graph model $MGM = (V, E)$ where
       $V = (M \cup B \cup I \cup T)$ and $E = (E_{mb} \cup E_{bi} \cup E_{it})$
Input: The set of target metrics $M'$
Input: The cost function $C$
Input: The Budget $W$
Output: A set of Instrument Instances $T'$

$unspent \leftarrow W$;                          ▷ keep track the remaining budget
$B' \leftarrow \emptyset$;
while $MGM$ is not empty OR $unspent > 0$ do
    $minW \leftarrow Infinite$;
    $maxM \leftarrow 0$;
    $sort(B)$;
                                      ▷ sort by $aggregate\_cost$ and $covered\_metrics$
    for $v_j \in B$ do
        $aggregate\_cost \leftarrow$ compute_aggregate_cost($v_j$);
        $covered\_metrics \leftarrow$ get_connected_metrics($v_j$) ;
        if $aggregate\_cost \leq unspent$ AND $aggregate\_cost \leq minW$ and
          $covered\_metrics > maxM$ then
            $minW \leftarrow aggregate\_cost$;
            $maxM \leftarrow covered\_metrics$;
            $B' \leftarrow B' \cup \{v_j\}$;
        end
    end
    $MGM \leftarrow MGM \setminus connected(v_j)$ ;
    $B' \leftarrow B' \cup B_j$;
    $currW \leftarrow currW - minW$;
end
return $B'$
```

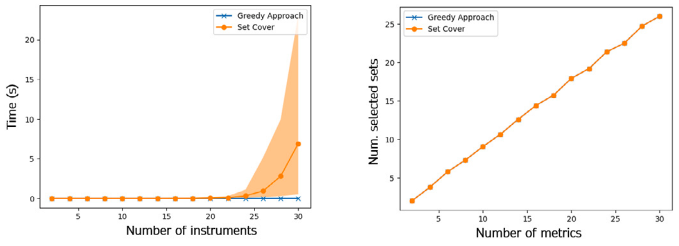
the solutions of our heuristics with the optimal ones. The aim is to analyze the improvement in performance, accuracy of results, and their trade-off.

To perform the evaluation, we develop a benchmark of graphs considering different scenarios in terms of size, topology, and cost functions for instrument instances of a security monitoring infrastructure. More in detail, we vary the number of instruments from 10 to 2000 to emulate both small and large-scale monitoring systems that may have thousands of sensors (e.g., smart cities) (Puiu et al., 2016). Concerning the topology, we considered the worst-case scenario represented by the *complete graphs*, i.e., all nodes in a partition are connected to the nodes in the connected partition. It represents a homogeneous environment in which all the metrics are measurable by all the instruments (i.e., maximum redundancy of instruments). For the best-case scenario, we modeled *path graphs*, i.e., the case in which each metric is connected to one bunch, each bunch is connected to one instrument, and each instrument is connected to one instrument instance. It represents a heterogeneous environment in which each metric is measurable by one and only one instrument (i.e., no redundancy of instruments). Additionally, we consider an average scenario through the *Erdos-Renyi random graph* (ERDdS & R&wi, 1959). It is obtained through the random connection of nodes such that each edge is included in the graph with probability $p = 0.5$, independently from every other edge. Finally, to consider different possibilities of costs associated with the instrument instances, we consider four different cost distributions including binomial, Poisson, geometric, and normal ones. For each configuration of these parameters, we executed 100 iterations of the experiment and we considered the average and the upper and lower quartiles of the output results. This benchmark evaluation resulted in a total of 12,000 experiments. The algorithms are implemented in Python (Van Rossum & Drake, 1995) and executed on a PC with an Intel Core i7-11800H 2.3 GHz processor and 16 GB memory.

8.1. Metrics computability

The first experimental setting deals with performance evaluation and validation of the metrics computability problem.

Classical approaches for solving this kind of problem typically refer to checking the existence of a path from a source (i.e., metric) to a



(a) Performance experiments for IR problem.

(b) Validation experiments for IR problem.

Fig. 5. Performance and validation experiments of the min-set cover algorithm and the greedy approach.

destination (i.e., instrument instance) node, keeping track of the information acquired during the traversal (e.g., visited nodes and edges) to refine the next traversal iterations (Jin, Ruan, Dey, & Xu, 2012; Korf, 1985). This typically leads to an increased space complexity required to maintain such information. In contrast, we proposed an algorithm that prunes the MGM according to its structure (see Section 5), which reduces space complexity rather than increasing it. Thus, to validate our approach, we compare our algorithm with a standard polynomial-time decider for path existence (Kleinberg & Tardos, 2006), which makes a DFS once per metric over the MGM.

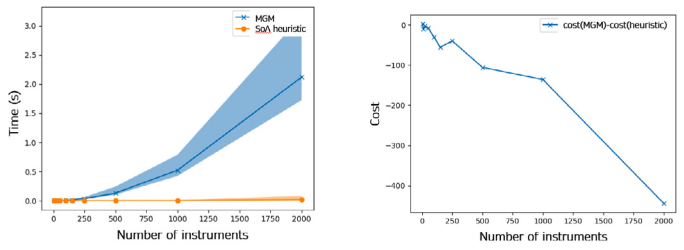
The experiments are done by varying the instrument inventory size from 10 to 2000 instruments. Fig. 4(a) shows the computation time (in seconds) to perform the metrics computability problem. The lines in the plot represent the mean values of the experiments, while the shape around the lines highlights the min–max variation. We considered the existing poly-time decider algorithm (SoA in Fig. 4) and our proposed solution (MGM in Fig. 4). We can notice that our approach outperforms the other algorithm as it is much faster. This is due to the fact that instead of visiting the full MGM, we just visit metrics (M) and instrument (I) partitions once. Indeed, by the definition of the MGM, the total number of vertices (V) is expected to be much higher than the number of vertices in each partition (i.e., metrics and instruments). Another interesting result is that our proposed solution introduces less variability than the existing one: this indicates more reliable performance results.

Concerning the validation of the proposed approach, we compare the set of computable metrics produced by the two algorithms. Fig. 4(b) reports the number of computable metrics for each experiment. The result is that in all the cases the outputs of our solution and the path existence algorithm are the same, as we can expect since we provide an exact solution.

Result 1. *The proposed algorithm for the metrics computability problem has better performance than existing traversal algorithms with an improvement of 35% of the computation time, and the output is valid in 100% of the cases.*

8.2. Instruments redundancy

We create a second experimental setting for the IR problem. In this case, we leverage the greedy approach proposed by Dasgupta et al. (2008) suitably adapted to the MGM model (Algorithm 2). The improvement in the performance of the greedy approach is formally proved by the authors, who also prove that it is a log approximation of the optimal result. The experiments confirm the improvement in performance with respect to algorithms without approximation. Fig. 5 shows the performance and validation results of the instruments redundancy problem dependent on the number of instruments and metrics, respectively. The lines in the plot represent the mean values of the experiments, while the shape around the lines highlights the min–max



(a) Performance experiments for CBC problem.

(b) Validation experiments for CBC problem.

Fig. 6. Performance and validation experiments of weighted set cover algorithms and the proposed heuristic.

variation. We compare the standard min set cover algorithm (Dasgupta et al., 2008) and the greedy approximation. The main goal of this experiment is to show that the log approximation, applied to the MGM is reasonably acceptable. On one side, the greedy approach outperforms the optimal algorithm as it is much faster. This is an expected result as the log approximation is a trade-off for a faster min-set-cover computation.

On the other hand, Fig. 5(b) reports the outputs of the two algorithms dependent on the number of metrics. The outputs correspond to the number of subsets included in the solution. The plot shows that the output size of the greedy approach is the same as the optimal one in 100% of the experiments. The fact that there are no experiments in which the solution is different from the optimal one can be explained by the fact that MGM does not consider the metrics as independent elements, but the modeled relations support the covering of similar metrics into the same set, making the greedy algorithm outcome very close to the optimal one.

Result 2. *The greedy approach is suitable for the instrument redundancy problem because it outperforms the optimal algorithm in terms of performance and even gives results very close to the optimal ones (the same in 100% of the tested cases).*

8.3. Cost-bounded constraint

The last experimental setting includes the experiments for the cost-bounded constraint problem. We compare our heuristic of Algorithm 3 with the existing greedy heuristic for the weighted set cover problem (Vasko et al., 2016). The goal of this validation is to show that the greedy approximation is not always the best choice in terms of results, although it is faster. To validate the algorithms we compare the calculated cost of the selected bunches.

Fig. 6(a) shows the computational time (in seconds) for the cost-bounded constraint problem. The lines in the plot represent the mean values of the experiments, while the shape around the lines highlights the min–max variation. We can notice that the greedy approach outperforms the proposed heuristic as it is faster. However, it is worth noting that for 2000 instruments the heuristic computes the solution in the order of a few seconds. This is a result that we can tolerate to trade-off performance and validation, as we show in Fig. 6(b). It represents the difference between the cost calculated with our approach and the one calculated with the existing greedy approach over the same graph (i.e., for a given experiment, let c_h be the output cost of the heuristic and c_g the one of the existing greedy solution. Then, the chart plots $c_h - c_g$). Thus, negative values in the plot correspond to the fact that the solution of our heuristic is less costly than the greedy one. The plot shows that the greedy approximation tends to assign a cost higher than the one assigned by our algorithm, and the difference increases proportionally to the number of considered metrics. This is because the greedy weighted-set-cover algorithm does not consider the relation of a metric with the *different* instruments and their instances, as our heuristic does.

Result 3. *The greedy approach provides costs that are 30% higher, on average, than the proposed heuristic, although its performance is better. However, the main goal of this problem is to save the cost in the security resources used to measure security posture, therefore the worse performance is a trade-off we can tolerate to gain advantages in cost saving.*

9. Usage scenario: Security posture

To show the capabilities and advantages of the security monitoring problems and solutions proposed in this paper, we developed a software system composed of: (i) a back-end application implementing all the proposed algorithms; (ii) a front-end application to support the usability of the approach. The algorithms are implemented in Python programming language (Van Rossum & Drake, 2009), while the fronted has been developed using Flask (Grinberg, 2018). The system includes a set of APIs that implement the solutions to the MC, IR, and CBC problems so that the system can be extended with different external interfaces. The input is a CSV file composed of tuples each one with a metric, the instrument implementing it, the instrument instance, and its cost. This information can be retrieved directly from the asset inventory of a company. Nowadays different management systems are available to collect this information (e.g., Corporation, 2023a, 2023b). Fig. 7 reports the Graphical User Interface (GUI) of the proposed software system, which is designed together with a security expert working in the monitoring team of a Security Operation Center (SOC).

The first pane on the left (Fig. 7-A) is the admin control pane. It allows the user to upload a new dataset for analysis and create its related project. In each project, the user can save and view the list of analyses. The pane on the top (Fig. 7-B) is the tab selection, in which the user can decide which analysis should be performed among the three presented in this paper. From left to right, the user can decide to view the graph (according to the MGM model), to perform metrics computability, instrument redundancy, or cost-bounded constraint analysis. The core part of the interface is the selection pane in Fig. 7-C, where the user can view the legend, insert the maximum budget available, and get a short summary of the analysis related to the selected problem. Finally, the great part of the interface is the graph view pane (Fig. 7-D), which includes the graphical representation of the results. The graph view is interactive and the user can select the nodes and apply the analysis only to a subset of elements. For example, one may want to run the analysis of the metrics computability only on a subset of metrics (e.g., the ones into a given sub-networks or the ones measuring cyber risks). This interaction supports more sophisticated and customized analyses.

In the rest of this section, we provide a usage scenario in which the user is a security administrator who wants to monitor the security awareness of her company. In this context, security awareness monitoring is intended as the analysis of the organization's security posture (Siponen, 2000). To this aim, we assume that the organization has a list of security metrics, suitably organized in a taxonomy, to evaluate the ability to prevent, defend, and respond to cyber-attacks. An example of such a taxonomy, used in this scenario as it is the most comprehensive in the current literature of security monitoring, is provided by Pendleton, Garcia-Lebron, Cho, and Xu (2016), who describe in detail and classify 63 security metrics. They investigate the security metrics for information systems to describe the security level according to four categories: (i) metrics regarding *vulnerabilities*, which includes all those metrics that measure the vulnerabilities of a system in terms of user, interface-induced, and software vulnerabilities; (ii) metrics on *defensive* capabilities, which determine the effectiveness of a system to defend against attacks; (iii) metrics of *attack* or threat severity, which measure the strength of attacks performed against a system; (iv) *situation* metrics, which reflect the comprehensive manifestation of attack-defense interactions for an enterprise or computer system,

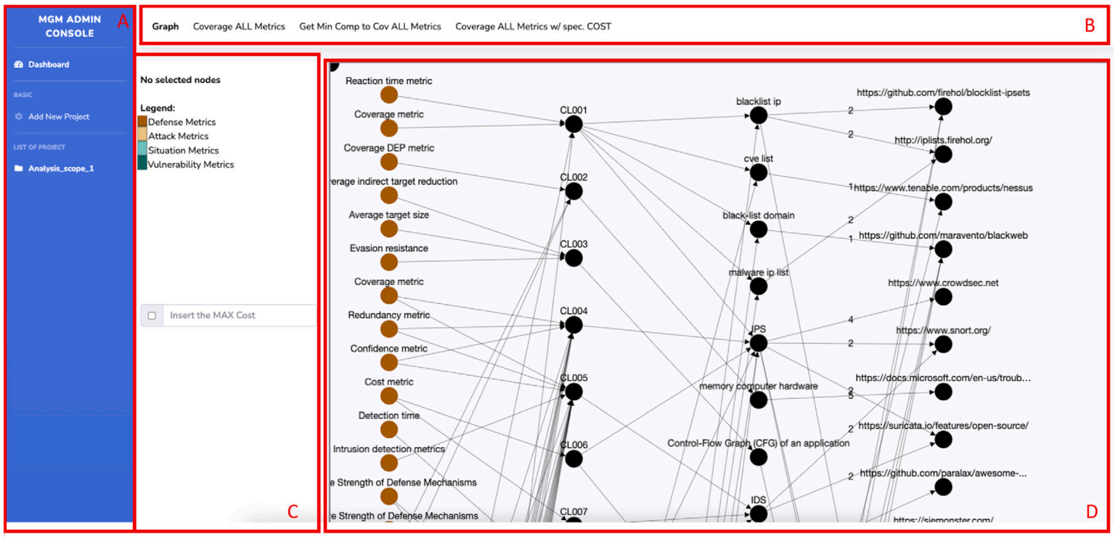


Fig. 7. User interface of the metrics management support system.

especially in terms of incidents and investments. Let us note that these metrics categories can be easily employed in the proposed MGM as node attributes, providing additional information that can be used, for example, to inform the visualization design. In this paper, we are focusing on the methodological approaches, while we leave the MGM enhanced with attributes and its implication to different disciplines (e.g., visualization) as future works.

In the usage scenario, we consider a small-medium enterprise (SME) with the following instruments: Intrusion Detection and Prevention Systems (IDS/IPS) (Liao, Lin, Lin, & Tung, 2013), log monitoring, penetration test reports, password manager tool, antivirus, ad blocker tool, and firewall management tool. It is a realistic scenario inspired by the existing surveys on the security approaches and instruments put in place in SMEs, which highlight that nearly 80% of the companies in Europe and the US employ the above instruments (Dimopoulos, Furnell, Jennex, & Kritharas, 2004). Consequently, the dataset used in input for the system is the asset inventory of an organization with the following information: (i) the set of metrics to measure and, for each metric, the instrument that measures it; (ii) the set of instruments available to the organization and, for each instrument, the set of instances that implement it; (iii) the set of instrument instances (e.g., tools, software) with their corresponding cost. We report the organization's instruments, their instances, and sources in Table 4, while we make available the full dataset and inventory for the research community for the sake of example and reproducibility.[2]

Table 4
Monitoring infrastructure reporting the list of instruments, their instances (e.g., tools, software, hardware), and the sources of the products used in the usage scenario.

Instrument	Instrument instance	Source
Intrusion detection system	Snort	www.snort.org/
Intrusion detection system	Suricata	www.suricata.io/
Intrusion prevention system	Crowdsec	www.crowdsec.net/
Log monitor	Service now	www.servicenow.com/
Password manager	Password safe	www.pwsafe.org/
Penetration testing system	Acunetix	www.acunetix.com
Penetration testing companion	SecList	https://seclists.org/
Antivirus	Norton	www.ie.norton.com/
Ad blocker	uBlock	www.ublockorigin.com/
Firewall manager	SkyBox	www.skyboxsecurity.com/

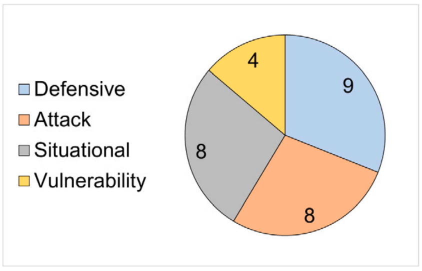


Fig. 8. Distribution of non-computable metrics according to their security category.

9.1. Metrics computability analysis

The first analysis the administrator needs to perform is to understand which metrics the organization is not able to measure. Performing the metrics computability analysis, the administrator discovers that 29 metrics over 63 cannot be measured with the organization's resources. We report the output of this analysis in Fig. 8 and the non-computable metrics in Table 5.

Analyzing the security categories of the non-measurable metrics (Fig. 8), s/he found that 9 of these metrics deal with defense and this implies that the organization is able to measure 47% of the defensive metrics; 8 metrics deal with attack measurement, implying just 30% of measured attack metrics; 8 metrics deal with situation measurement, resulting in 58% of situation metrics measurement; finally, 4 metrics are about vulnerabilities, and the organization can measure 64% of vulnerability metrics. On average, only 49.75% of the metrics can be measured by the organization, meaning that more than half of the metrics cannot be measured, resulting in incomplete security coverage. Among the metrics the organization is not able to measure, particular attention should be put to the "policy coverage" metric, which deals with compliance with security governance standards. This implies potentially higher risks, inability to keep the risks under control, and absence of policies to get security certification (e.g., ISO 27001 ISO Central Secretary, 2013).

In addition, the organization is not able to measure any of the attack metrics dealing with zero-day attacks (Bilge & Dumitraş, 2012), which are those attacks that have not been disclosed publicly, and therefore more difficult to manage. This means that the organization can respond

[2] https://github.com/ds-square/monitoring_mgm/blob/main/inventory_metrics.csv.

Table 5
List of non-computable metrics and their categories: defensive and vulnerability metrics on the left, attack and situational metrics on the right.

Category	Metric	Category	Metric
Defensive	Policy coverage	Attack	Lifetime of 0-day attacks
Defensive	Data execution prevention	Attack	Victims by 0-day attacks
Defensive	Monitoring level	Attack	Targeted threat index
Defensive	Network diversity	Attack	Botnet size
Defensive	Memory entropy	Attack	Botnet efficiency
Defensive	Relative strength of defense	Attack	Adversarial ML attacks
Defensive	Collective strength of defense	Attack	Obfuscation prevalence
Defensive	Moving target defense level	Attack	Structural complexity
Defensive	Avg indirect target reduction	Situational	Network maliciousness
Vulnerability	Password guessability	Situational	Rogue network metric
Vulnerability	Interface induced susceptibility	Situational	Control-plane reputation
Vulnerability	Vulnerability lifetime	Situational	Probability of compromise
Vulnerability	Effort metric	Situational	Encounter rate
		Situational	Breach size
		Situational	Delay in incident detection
		Situational	Network present value

to such attacks only *a posteriori*, i.e. after the attack happened, without the possibility to prevent them.

Finally, the organization is not able to measure the costs associated with cyber incidents (i.e., it is not able to quantify the impacts of an attack), nor the susceptibility to phishing attacks, which are the most common ones (Open Web Application Security Project, 2021) and exposing the organization to high risk of attack.

In summary, the administrator understands that the overall security monitoring of the organization is not good (49% of all the necessary metrics) and many of the metrics that cannot be measured are essential for the organization's security. To improve security posture, the next step is the analysis of the resources, which helps her to understand whether some instruments are redundant (and can be saved) and which are instruments necessary to measure the most critical security threats.

9.2. Instrument redundancy analysis

In this phase, the administrator is interested in understanding whether some instruments are not properly used, so as to save resources to invest differently in security monitoring. S/he performs the instrument redundancy analysis and, as the first step, studies how many instruments are necessary to measure all the metrics. In the proposed scenario, all 63 metrics can be measured using 26 instruments, including Intrusion Detection and Prevention Systems (IDS and IPS), Incident Management Systems, NVD knowledge bases,[3] and phishing detectors.

The organization owns just 7 instruments over the 26 that are necessary to measure all the metrics. Among the instruments that the organization does not own, some are freely available (e.g., CVE list, phishing detector), while others require the purchase of subscriptions (e.g., Incident Management systems) or physical devices (e.g., honeypot). Thus, the organization's equipment includes just 27% of the necessary instruments.

However, some strategies can be put in place to improve this lack, depending on which are the most important metrics that the organization needs to measure. To solve this problem, the administrator can select such metrics in the interactive system and perform the instrument redundancy analysis on them.

For example, the organization may have the following requirement: "*provide proactive strategies against phishing, malware, botnet attacks, and the incidents should be logged*". In such a case, the administrator selects the related metrics (i.e., phishing susceptibility, malware susceptibility, number of affected assets, malware infection rate, botnet sizes and efficiency, and number of incidents) and the system outputs the set of instruments necessary to cover such metrics. In this way, s/he identifies which are the missing resources that, in the proposed scenario, correspond to a phishing detector, malware, and botnet analyzer.

[3] https://cve.mitre.org/.

On the other hand, the administrator discovers the presence of 2 redundant instruments, which are the IDS (whose metrics are already measured by the IPS in this usage scenario) and the ad blocker. Both these instruments come with a subscription, thus, by removing them, s/he saves resources and money.

This analysis supports the administrator in deciding about the assets that the organization should accomplish to improve its security posture, which consists of buying the missing instruments and removing the redundant ones. However, such decisions may be bounded by a limited budget. For this reason, the last analysis considers the cost-bounded constraint problem.

9.3. Cost-bounded constraint analysis

The last step is the analysis of the budget to measure as many metrics as possible. Let us note that such a budget may be either the economic value of the instrument instances, their power consumption, or any other kind of cost. In this usage scenario, let us consider that the organization has a budget of 43k$ allocated for security investments.

The first information the administrator should take into account is the total cost of the instrument instances necessary to measure all the metrics. We assume it is 87k$, corresponding almost to double the allocated budget.[4] Given such a limited budget, it is crucial for the organization to maximize the number of measurable metrics according to their cost.

By performing the cost-bounded constraint analysis, the administrator retrieves the optimal configuration of the budget allocation (i.e., 43k$) is sufficient to buy 18 instruments, which increases the number of measurable metrics from 34 of the initial configuration to 43. In addition, 5 metrics can be measured with free tools. Thus, the administrator can reach an improvement of 22% of measurable metrics with the same budget allocated for the initial security monitoring infrastructure.

The automatic system implementing the proposed algorithms performs the analyses in a few seconds. Without it, the administrator should first formalize the problems to solve according to the organization's mission. Then, s/he has to understand which data is necessary to collect. Finally, s/he should manually analyze the data. It is worth noting that the security monitoring process is long, and performing it manually is time-consuming and prone to human errors (Siponen & Willison, 2009). This software system supports, based on the proposed MGM model and its related algorithms, the analyses of security monitoring problems while saving time and resources.

[4] It is an assumption inspired by the report by VentureBeat https://venturebeat.com/security/report-only-10-of-orgs-had-higher-budget-for-cybersecurity-despite-increased-threat-landscape/.

10. Discussion

This paper addressed the existing research gap between security monitoring and resource management in planning investments. We define a set of three security monitoring problems to fill this gap, that informs decisions about the design of monitoring infrastructures. We showed that these problems are indeed decision problems and we designed a graph-based model to support their solutions by suitably designing heuristics for each of them. We validated the heuristics through a comparison with the current state-of-the-art solutions and we proposed a usage scenario based on realistic data and informed by a security expert from a Security Operation Center.

In terms of opportunities, we believe that the formalization of security monitoring problems deserves attention to inform decisions about security investments. This is especially relevant if we consider the budget constraints the organizations face every day, especially in the cybersecurity sector (Armenia et al., 2021). Formalizing these problems aids researchers and practitioners in developing efficient solutions accordingly, as we propose in this paper. The consequent effect of this formalization is the development of systems and tools to support the decisions and analysis of security monitoring, as we showed in the proposed usage scenario. This paper moved a step towards this problem and in the rest of this section, we discuss its opportunities and limitations.

Generality of the approach. Although we modeled the problems, their solutions, the graph-based model, and the usage scenario in the context of security monitoring, we believe that the approach can be easily generalized to other monitoring scenarios. For example, in the case of smart cities, several sensors and components collect data from the environment to support analytical tasks to ensure Quality of Service (QoS). The application of the proposed approach to such a scenario informs the administrator about the optimal planning of monitoring devices to ensure QoS according to predefined Service Level Agreements (SLAs). In that case, the IoT community can benefit from the approach to save storage resources and costs to support QoS.

Another example is the application to Cyber Threat Intelligence (CTI), which is the process of gathering and analyzing information relating to cyber threats (Lee, 2023). It consists of integrating different sources of information which are shared between organizations. Thus, it is crucial to analyze the computability of each source and its monitoring infrastructure to assess the comprehensiveness of the CTI. The MGM model allows for analyzing measurement capabilities of CTI metrics across different organizations and planning investments to optimize CTI systems, which are real current needs of the organizations (Wagner, Mahbub, Palomar, & Abdallah, 2019). We foresee efforts from ourselves and other researchers to apply the proposed approach in specific scenarios to inform expert systems.

Usability of the system. The paper presented a basic GUI system to support the security administrators in the analysis of the monitoring problems. Although visualization is an important aspect of decision support systems, the focus of this paper is more on the formalization and algorithmic aspects, for which we proposed a performance evaluation and usage scenario. In addition, although the usage scenario is designed with the support of a domain expert, he does not have expertise in visualization design. For this reason, another opportunity is the design of a visual analytics system informed by proposed problems and heuristics to support security monitoring decision-making. In particular, we plan to involve visual analysts in the design of the interface and provide a structured user study.

Application to real-world scenarios. Solving security monitoring problems involves the description of all the assets, tools, and metrics of an organization, as well as its business objectives (e.g., SLAs). Since this is sensible information for an organization, it is difficult to retrieve real data and show a real use case. For this reason, we designed a usage scenario with a security expert to make it as realistic as possible. The problem of lack of real publicly available data opens the opportunity for looking at benchmark-based approaches to automatically collect the metrics and asset inventories from real environments, potentially applied to different contexts (e.g., security environment, Cyber-Physical Systems, smart cities).

On the other hand, the usage scenario is inspired by the real need of a cybersecurity company, anonymized for the sake of privacy, and informal interviews with its security practitioners highlighted the practical applicability of the proposed approach to monitoring management. Their current need is to manage the complexity of the systems, which requires them to manage the monitoring of subparts of the organization's systems individually (e.g., subnetworks). This typically results in the redundancy of instruments in the different subparts of the organization. The proposed approach allows both reducing such a redundancy and managing the optimization of the monitoring infrastructure even in complex systems, automatically. In this way, security administrators can provide investment plans for the organization as a whole even if its infrastructure is complex. This provides an enhancement of the operational efficiency from two different perspectives: first, the overall monitoring infrastructure can be managed in less time, facilitating the works of security administrators; second, the data-driven nature of the approach makes the human decisions more informed than the manual management of the monitoring infrastructure, thus reducing the potential bias of an assessment exclusively based on human evaluation.

11. Conclusions

This paper considered the problem of supporting security experts in evaluating the effectiveness of their monitoring infrastructure, in terms of the suitability of the deployed tools. Indeed, gathering and providing useful data for enabling the computation of security metrics, and potentially planning new investments for monitoring equipment is a complex process.

To support security experts in this complexity, we considered three decision tasks and we formalized the related problems namely the *metric computability*, *instrument redundancy*, and *cost-bounded constraint* problems. We introduced the *Metric Graph Model* MGM i.e., a graph-based representation of the relevant relationships between metrics of interest, measurements/data needed to compute the considered metric, and instruments needed to gather such data. Leveraging the MGM, we analyzed the computational complexity of the considered problems and we proposed heuristics to solve them. We evaluated our proposal by comparing our solutions with other state-of-the-art heuristics. Such empirical evaluation shows that the proposed heuristics perform well and represent the first step toward the design and implementation of more sophisticated decision support tools. Finally, we presented a software system and usage scenario that showed the capabilities of the proposed problems and solutions. This enables more analyses that can be carried out. For example, one may analyze the optimal placement of instrument bunches to allocate the resources in a distributed manner in which each bunch is a distributed host; or it is possible to analyze the best combination of instrument instances to select the most suitable providers.

In future work, we plan to enlarge the covered problems according to the existing systematization of security posture analysis (Jones, 2022). We will improve the analyses by providing multi-objective problems combining multiple approaches. This will support a more comprehensive analysis. Finally, as some of the presented solutions are indeed approximated algorithms, we will include the concept of uncertainty to provide more robust analyses. To this aim, we may model the proposed MGM as a Bayesian Network and leverage probabilistic approaches.

CRediT authorship contribution statement

Alessandro Palma: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Andrea Sorrentino:** Conceptualization, Data curation, Investigation, Methodology, Software. **Silvia Bonomi:** Conceptualization, Formal analysis, Investigation, Methodology, Project administration, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by the SERICS project (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Data availability

Data and code are openly available and linked in the paper.

References

Ahmad, Z., Ong, T. S., Liew, T. H., & Norhashim, M. (2019). Security monitoring and information security assurance behaviour among employees: An empirical analysis. *Information & Computer Security*, *27*(2), 165–188.

Anastasov, I., & Davcev, D. (2014). SIEM implementation for global and distributed environments. In *2014 world congress on computer applications and information systems* WCCAIS, (pp. 1–6). http://dx.doi.org/10.1109/WCCAIS.2014.6916651.

Armenia, S., Angelini, M., Nonino, F., Palombi, G., & Schlitzer, M. F. (2021). A dynamic simulation approach to support the evaluation of cyber risks and security investments in SMEs. *Decision Support Systems*, *147*, Article 113580. http://dx.doi.org/10.1016/j.dss.2021.113580.

Ashibani, Y., & Mahmoud, Q. H. (2017). Cyber physical systems security: Analysis, challenges and solutions. *Computers & Security*, *68*, 81–97.

Beasley, J. E. (1990). A lagrangian heuristic for set-covering problems. *Naval Research Logistics*, *37*(1), 151–164.

Bernhard, K., & Vygen, J. (2008). *Combinatorial optimization: Theory and algorithms* (3rd ed.). Springer, 2005.

Bi, S., & Zhang, Y. J. A. (2017). Graph-based Cyber Security Analysis of State Estimation in Smart Power Grid. *IEEE Communications Magazine*, *55*(4), 176–183.

Bilge, L., & Dumitraş, T. (2012). Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on computer and communications security* (pp. 833–844).

Böhme, R., & Freiling, F. C. (2008). On metrics and measurements. *Dependability Metrics: Advanced Lectures*, 7–13.

Bowen, L., & Lupo, C. (2020). The performance cost of software-based security mitigations. In *Proceedings of the ACM/SPEC international conference on performance engineering* ICPE, (pp. 210–217).

Ceria, S., Nobili, P., & Sassano, A. (1998). A Lagrangian-based heuristic for large-scale set covering problems. *Mathematical Programming*, *81*(2), 215–228.

Chernov, A. V., Butakova, M. A., & Karpenko, E. V. (2015). Security incident detection technique for multilevel intelligent control systems on railway transport in Russia. In *2015 23rd telecommunications forum telfor* TELFOR, (pp. 1–4). http://dx.doi.org/10.1109/TELFOR.2015.7377381.

Collins, M. P. (2011). Graph-based analysis in network security. In *2011 - MILCOM 2011 military communications conference* (pp. 1333–1337).

Coppolino, L., D'Antonio, S., Formicola, V., & Romano, L. (2011). Integration of a system for critical infrastructure protection with the OSSIM SIEM platform: A dam case study. In F. Flammini, S. Bologna, & V. Vittorini (Eds.), *Computer safety, reliability, and security* (pp. 199–212). Berlin, Heidelberg: Springer Berlin Heidelberg.

Corporation, Z. (2023a). *Zoho ManageEngine asset explorer*: *Tool*, Claymont, USA: ZOHO Corporation.

Corporation, I. (2023b). *Invgate service desk*: *Standard*, Geneva, CH: InvGate Corporation.

Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. (2008). *Algorithms*. McGraw-Hill Higher Education New York.

Dimopoulos, V., Furnell, S., Jennex, M. E., & Kritharas, I. (2004). Approaches to IT security in small and medium enterprises. In *AISM* (pp. 73–82). Citeseer.

ERDdS, P., & R&wi, A. (1959). On random graphs I. *Publicationes Mathematicae Debrecen*, *6*(290–297), 18.

Formicola, V., Di Pietro, A., Alsubaie, A., D'Antonio, S., & Marti, J. (2014). Assessing the impact of cyber attacks on wireless sensor nodes that monitor interdependent physical systems. In J. Butts, & S. Shenoi (Eds.), *Critical infrastructure protection VIII* (pp. 213–229). Berlin, Heidelberg: Springer Berlin Heidelberg.

Ge, M., Hong, J. B., Guttmann, W., & Kim, D. S. (2017). A framework for automating security analysis of the internet of things. *Journal of Network and Computer Applications*, *83*, 12–27.

George, G., & Thampi, S. M. (2018). A Graph-Based Security Framework for Securing Industrial IoT Networks From Vulnerability Exploitations. *IEEE Access*, *6*, 43586–43601.

Ghafir, I., Prenosil, V., Svoboda, J., & Hammoudeh, M. (2016). A survey on network security monitoring systems. In *2016 IEEE 4th international conference on future internet of things and cloud workshops (fiCloudW)* (pp. 77–82). IEEE.

Golab, L., Korn, F., Li, F., Saha, B., & Srivastava, D. (2015). Size-constrained weighted set cover. In *2015 IEEE 31st international conference on data engineering* (pp. 879–890). IEEE.

González-Granadillo, G., González-Zarzosa, S., & Diaz, R. (2021). Security information and event management (SIEM): Analysis, trends, and usage in critical infrastructures. *Sensors*, *21*(14), http://dx.doi.org/10.3390/s21144759.

Grinberg, M. (2018). *Flask web development: developing web applications with python*. O'Reilly Media, Inc..

Han, K., Choi, J. H., Choi, Y., Lee, G. M., & Whinston, A. B. (2023). Security defense against long-term and stealthy cyberattacks. *Decision Support Systems*, *166*, Article 113912. http://dx.doi.org/10.1016/j.dss.2022.113912.

Hayat, H., Griffiths, T., Brennan, D., Lewis, R. P., Barclay, M., Weirman, C., et al. (2019). The State-of-the-Art of sensors and environmental monitoring technologies in buildings. *Sensors*, *19*(17), http://dx.doi.org/10.3390/s19173648.

Hindy, H., Brosset, D., Bayne, E., Seeam, A., & Bellekens, X. (2019). Improving SIEM for critical SCADA water infrastructures using machine learning. In S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, A. Antón, S. Gritzalis, J. Mylopoulos, & C. Kalloniatis (Eds.), *Computer security* (pp. 3–19). Cham: Springer International Publishing.

Hwoij, A., Khamaiseh, A., & Ababneh, M. (2021). SIEM architecture for the Internet of Things and smart city. In *International conference on data science, e-learning and information systems 2021* DATA '21, (pp. 147–152). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3460620.3460747.

ISO Central Secretary (2013). *ISO/IEC 27001:2013-Information security management*: *Standard ISO/IEC TR 27001:2013*, Geneva, CH: International Organization for Standardization.

Jin, R., Ruan, N., Dey, S., & Xu, J. Y. (2012). Scarab: scaling reachability computation on large graphs. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data* (pp. 169–180).

Jones, A. (2022). Security posture: A systematic review of cyber threats and proactive security.

Kamble, A., & Bhutad, S. (2018). Iot based patient health monitoring system with nested cloud security. In *2018 4th international conference on computing communication and automation* ICCCA, (pp. 1–5). IEEE.

Kang, M., Templeton, G. F., Lee, E. T., & Um, S. (2024). A method framework for identifying digital resource clusters in software ecosystems. *Decision Support Systems*, *177*, Article 114085. http://dx.doi.org/10.1016/j.dss.2023.114085.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Kayhan, V. O., Agrawal, M., & Shivendu, S. (2023). Cyber threat detection: Unsupervised hunting of anomalous commands (UHAC). *Decision Support Systems*, *168*, Article 113928. http://dx.doi.org/10.1016/j.dss.2023.113928.

Khaleel, T. A., & Al-Shumam, A. A. (2020). A Study of Graph Theory Applications in IT Security. *Iraqi Journal of Science*, 2705–2714.

Kleinberg, J., & Tardos, E. (2006). *Algorithm design*. Pearson Education India.

Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, *27*(1), 97–109.

Laprie, J.-C. (1992). Dependability: Basic concepts and terminology. In *Dependability: basic concepts and terminology* (pp. 3–245). Springer.

Lavrova, D. S. (2016). An approach to developing the SIEM system for the Internet of Things. *Automatic Control and Computer Sciences*, *50*, 673–681.

Lee, M. (2023). *Cyber threat intelligence*. John Wiley & Sons.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16–24.

Liu, R.-L., & Lu, Y.-L. (2003). Distributed agents for cost-effective monitoring of critical success factors. *Decision Support Systems*, *35*(3), 353–366. http://dx.doi.org/10.1016/S0167-9236(02)00113-6.

Liu, J., Zhang, W., Ma, T., Tang, Z., Xie, Y., Gui, W., et al. (2020). Toward security monitoring of industrial cyber-physical systems via hierarchically distributed intrusion detection. *Expert Systems with Applications*, *158*, Article 113578. http://dx.doi.org/10.1016/j.eswa.2020.113578.

López Velásquez, J. M., Martínez Monterrubio, S. M., Sánchez Crespo, L. E., & Garcia Rosado, D. (2023). Systematic review of SIEM technology: SIEM-SC birth. *International Journal of Information Security*, *22*(3), 691–711.

Open Web Application Security Project (2021). *OWASP top ten: Standard*, OWASP Foundation.

Pendleton, M., Garcia-Lebron, R., Cho, J.-H., & Xu, S. (2016). A Survey on Systems Security Metrics. *ACM Computing Surveys*, *49*(4), 62:1–62:35.

Puiu, D., Barnaghi, P., Tönjes, R., Kümper, D., Ali, M. I., Mileo, A., et al. (2016). Citypulse: Large scale data analytics framework for smart cities. *IEEE Access*, *4*, 1086–1108.

Sabur, A., Chowdhary, A., Huang, D., & Alshamrani, A. (2022). Toward scalable graph-based security analysis for cloud networks. *Computer Networks*, *206*, Article 108795.

Sheeraz, M., Paracha, M. A., Haque, M. U., Durad, M. H., Mohsin, S. M., Band, S. S., et al. (2023). Effective security monitoring using efficient SIEM architecture. *Human-centric Computing and Information Sciences*, *13*, 1–18.

Siponen, M. T. (2000). A conceptual foundation for organizational information security awareness. *Information Management & Computer Security*.

Siponen, M., & Willison, R. (2009). Information security management standards: Problems and solutions. *Information & Management*, *46*(5), 267–270.

Srinidhi, B., Yan, J., & Tayi, G. K. (2015). Allocation of resources to cyber-security: The effect of misalignment of interest between managers and investors. *Decision Support Systems*, *75*, 49–62. http://dx.doi.org/10.1016/j.dss.2015.04.011.

Swamynathan, S., Kannan, A., & Geetha, T. (2006). Composite event monitoring in XML repositories using generic rule framework for providing reactive e-services. *Decision Support Systems*, *42*(1), 79–88. http://dx.doi.org/10.1016/j.dss.2004.10.001.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, *1*(2), 146–160.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Van Rossum, G., & Drake, F. L., Jr. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Vasko, F. J., Lu, Y., & Zyma, K. (2016). What is the best greedy-like heuristic for the weighted set covering problem? *Operations Research Letters*, *44*(3), 366–369.

Vassilev, V., Sowinski-Mydlarz, V., Gasiorowski, P., Ouazzane, K., & Phipps, A. (2021). Intelligence Graphs for Threat Intelligence and Security Policy Validation of Cyber Systems. In *Proc. of international conference on artificial intelligence and applications* (pp. 125–139). Singapore: Springer.

Vazirani, V. V. (2001). *Approximation algorithms: vol. 1*, Springer.

Von Solms, R., & Van Niekerk, J. (2013). From information security to cyber security. *Computers & Security*, *38*, 97–102.

Wagner, T. D., Mahbub, K., Palomar, E., & Abdallah, A. E. (2019). Cyber threat intelligence sharing: Survey and research directions. *Computers & Security*, *87*, Article 101589. http://dx.doi.org/10.1016/j.cose.2019.101589.

Xie, P., Li, J. H., Ou, X., Liu, P., & Levy, R. (2010). Using Bayesian networks for cyber security analysis. In *2010 IEEE/IFIP international conference on dependable systems & networks (DSN)* (pp. 211–220).