

Analysing Mission-critical Cyber-physical Systems with AND/OR Graphs and MaxSAT

MARTÍN BARRÈRE and CHRIS HANKIN, Institute for Security Science and Technology,
Imperial College London, UK

Cyber-Physical Systems (CPS) often involve complex networks of interconnected software and hardware components that are logically combined to achieve a common goal or mission; for example, keeping a plane in the air or providing energy to a city. Failures in these components may jeopardise the mission of the system. Therefore, identifying the minimal set of critical CPS components that is most likely to fail, and prevent the global system from accomplishing its mission, becomes essential to ensure reliability. In this article, we present a novel approach to identifying the *Most Likely Mission-critical Component Set (MLMCS)* using AND/OR dependency graphs enriched with independent failure probabilities. We address the MLMCS problem as a Maximum Satisfiability (MaxSAT) problem. We translate probabilities into a negative logarithmic space to linearise the problem within MaxSAT. The experimental results conducted with our open source tool LDA4CPS indicate that the approach is both effective and efficient. We also present a case study on complex aircraft systems that shows the feasibility of our approach and its applicability to mission-critical cyber-physical systems. Finally, we present two MLMCS-based security applications focused on system hardening and forensic investigations.

CCS Concepts: • **Networks** → **Cyber-physical networks**; **Network reliability**; **Network security**; • **Mathematics of computing** → **Combinatorial optimization**; **Graph theory**; • **Security and privacy** → **Formal methods and theory of security**; **Distributed systems security**; • **Computer systems organization** → **Embedded and cyber-physical systems**; **Dependable and fault-tolerant systems and networks**; • **Applied computing** → **Avionics**;

Additional Key Words and Phrases: AND/OR graphs, maximum satisfiability, security metric, mission-critical systems, most-likely mission-critical set, system hardening, forensic investigations, aircraft systems

ACM Reference format:

Martin Barrère and Chris Hankin. 2021. Analysing Mission-critical Cyber-physical Systems with AND/OR Graphs and MaxSAT. *ACM Trans. Cyber-Phys. Syst.* 5, 3, Article 30 (July 2021), 29 pages.
<https://doi.org/10.1145/3451169>

1 INTRODUCTION

Cyber-Physical Systems (CPS) are characterised by a deep integration among cyber elements (e.g., computers, algorithms, data), physical components (e.g., sensors, actuators), and processes.

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE).

Author's address: M. Barrère and C. Hankin, Institute for Security Science and Technology, Imperial College London, South Kensington Campus, London SW7 2AZ, UK; emails: {m.barrere, c.hankin}@imperial.ac.uk.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2021 Copyright held by the owner/author(s).

2378-962X/2021/07-ART30

<https://doi.org/10.1145/3451169>

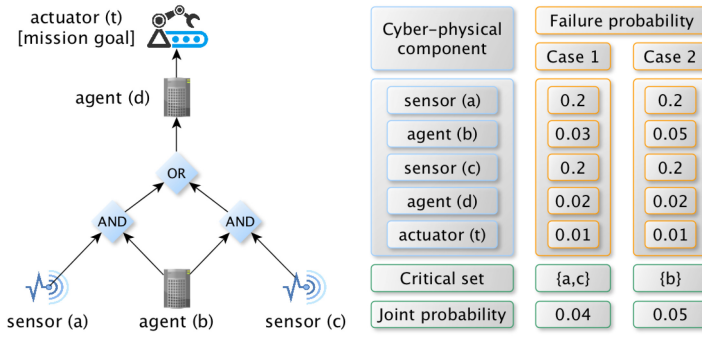


Fig. 1. Most likely mission-critical set (MLMCS) examples.

Accordingly, CPS environments usually expose complex networks of dependencies among cyber and physical components designed to accomplish a particular mission. Such complexity makes it challenging to understand what are the most likely points of failure, which is essential to ensure reliability [41]. AND/OR graphs are particularly suitable to represent these types of networks, since they are flexible enough to capture intricate logical dependencies among CPS components. However, the general problem of identifying critical nodes in AND/OR graphs, i.e., a minimal vertex cut, is an NP-complete problem [27, 28, 29, 43]. In this work, we consider AND/OR graphs enriched with node weights that represent independent failure probabilities, for example, due to internal component faults. More specifically, we address the following problem: Given a weighted AND/OR graph representing the operational dependencies of a mission-critical cyber-physical system, what is the minimal set of vital CPS components that is most likely to fail and prevent the global system from delivering the mission it was built for? In other words, we look for the set of critical nodes that maximises the probability of failure of the system's mission. We call this set the **Most Likely Mission-critical Component Set (MLMCS)**, or most likely critical set for short. Let us illustrate the problem with a simple example.

1.1 Motivating Example

Figure 1 shows an AND/OR graph involving five CPS components in the form of atomic nodes, namely, two sensors (*a* and *c*), two software agents (*b* and *d*), and one actuator (*t*). In addition, the graph includes two AND nodes and one OR node that express the logical dependencies among the CPS components. The mission of the system is to keep actuator *t* operational. The graph reads as follows: The actuator *t* depends on the output of software agent *d*, e.g., a **programmable logic controller (PLC)**. Agent *d* in turn has two alternatives to work properly; it can use either the readings of sensor *a* and the output from agent *b* together, or the output from agent *b* and the readings of sensor *c* together.

In addition, each CPS component has an associated value (or weight) that represents its probability of failure, which is independent from other components. The table in Figure 1 shows two cases with different failure probability assignments (discussed later). A failed component is understood as a component unable to fulfil its purpose, thus affecting the components that depend on it. In that context, we define a mission-critical set as a set of components that, should they fail simultaneously, will lead to the failure of the system's mission. For example, if sensors *a* and *c* fail simultaneously, then they would disrupt both AND nodes and subsequently the OR node, which in turn will affect agent *d* and finally the actuator *t*. Since the failure of {*a*, *c*} would compromise the mission of the system, we say that {*a*, *c*} is a mission-critical node set. A mission-oriented cyber-physical system may have many critical sets, though: if agent *b* fails, then both AND nodes will be affected and thus the dependencies on which *t* relies. Therefore, the set {*b*} is also a

mission-critical set. In this example, both critical sets are said to be minimal, since the removal of any of their nodes would allow the system to continue to fulfil its mission.

Now, given the probability of failure for each component, the objective of our methodology is to identify the MLMCS, i.e., the minimal set of components with the highest probability of compromising the system's mission. Note that the problem not only relates to the structural vertex cut in the AND/OR graph but also to the failure probabilities assigned to the CPS components. For example, if we consider the probability assignment depicted in Case 1, Figure 1, the MLMCS is the critical set $\{a, c\}$ with a joint probability of failure of $0.2 \times 0.2 = 0.04$. However, keeping the same graph structure and now considering the assignment from Case 2, the critical set with maximum probability is $\{b\}$ with a probability of 0.05.

1.2 Scope of the Article and Methodology

The identification of MLMCS in cyber-physical systems allows designers to assess the strength of a system's design as well as conducting tradespace analysis [62]. In addition, it enables security practitioners to prioritise which critical nodes should be made more robust first, e.g., by adding redundancy, to decrease the total probability of failure and sustain operational continuity. As explained in Section 2, previous works have addressed related graph cut problems in well-known approaches such as **fault tree analysis (FTA)** [37], widely used in CPS safety analysis [54], and attack trees [66], extensively used in network security [47]. In this work, however, we deal with general AND/OR graphs that cover tree-based cases as well as **directed acyclic graphs (DAGs)**, and also allow the presence of cycles.

In particular, our methodology focuses on the static analysis of input AND/OR dependency graphs enriched with independent failure probabilities. We do not cover dependent failures or cases where, for example, a sensor would be more likely to fail if it is located in the proximity of another sensor that has failed due to a fire in the bay area of an aircraft. These aspects are beyond the scope of this article and have been scheduled as future work. However, the model does cover the impact that independent failures can have on the overall mission of the system, for example, due to inherent device faults. To address the MLMCS problem, we leverage and extend the technique presented in Reference [14], which aims at identifying minimum-effort attack strategies in industrial control systems. As opposed to Reference [14], in this work, we incorporate stochastic aspects into the model that are applicable to a wide range of cyber-physical systems, a richer model that allows designers to describe functional blocks and abstract entities, thus enabling more expressive specifications, and two MLMCS-based security applications focused on system hardening and forensic investigations.

We address the MLMCS problem from a dependency analysis perspective and model it as a Weighted Partial MaxSAT problem. From an optimisation standpoint, the problem addressed in this article requires maximising the multiplication of decision variables (probabilities), which makes it a non-linear problem. Since MaxSAT approaches are additive in nature (as explained in Section 5), we adapt the problem by transforming the domain of node weights into a negative logarithmic space, which can be treated in an additive manner, and then translating the results back to the original stochastic domain. The outcome of our approach can be understood as a metric that complements the risk assessment machinery for cyber-physical systems.

1.3 Main Contributions

In this work, we make the following contributions: (1) we present a novel MaxSAT-based probabilistic approach to solve the problem of identifying the **most likely mission-critical set (MLMCS)** on mission-oriented cyber-physical systems modelled with AND/OR dependency graphs; (2) we provide an open-source tool called LDA4CPS that implements the proposed

methodology, is able to run multiple SAT-solvers in parallel to increase efficiency, and is fully documented online [8]; (3) we present a thorough analysis of the method's performance and scalability, and experimentally apply our technique on a control system for the pitch function of an aircraft where we analyse different configurations and failure scenarios to illustrate its potential use on either manual inspection for decision making or automated quantitative analysis, and (4) we describe two MLMCS-based security applications focused on system hardening and forensic investigations.

2 RELATED WORK

Over the past few years, there has been a considerable amount of research focused on improving the dependability, security, and safety of cyber-physical systems [41, 54, 56]. **Fault tree analysis (FTA)** constitutes a well-known assessment tool where tree nodes model events such as component failures and the root of the tree represents an undesired state of the system called the top-level event [37]. Leaves and internal nodes are combined using logic gates (e.g., AND, OR) as a mechanism to model how such events can lead to the top-level event. FTA focuses mostly on safety and reliability and is used in a wide variety of industries such as aerospace, power plants, nuclear plants, and other high-hazard engineering fields [17, 22, 64]. Attack trees also constitute a major security research area [24, 26, 66, 75]. Structurally speaking, attack trees are similar to fault trees where nodes representing attack steps are logically combined to achieve the attacker's goal (root of the attack tree). Hybrid fault/attack tree models have also been explored as an effort to bring together safety and security aspects [48, 63].

The survey presented in Reference [64] provides a comprehensive analysis of current techniques and tools used in FTA, including Boolean manipulation and **binary decision diagrams (BDDs)** [2] to find standard **minimal cut sets (MCSs)**. An MCS is defined as a minimal set of events that together cause the top-level event. As such, MCSs play a fundamental role in the structural analysis of fault trees. Common FTA approaches to compute all MCSs include classic algorithms such as MOCUS and MICSUP, as well as other BDD-based methods [64]. More recently, the work presented in Reference [53] also deals with MCSs and SAT-based techniques. In particular, the authors extend an existing SAT-solving framework to compute all MCSs during the SAT resolution process. However, large trees may involve hundreds of MCSs, and therefore, computation costs can become prohibitively expensive. In addition, even when MCSs can be normally categorised according to their size [44], i.e., first-order MCSs (or Single Points of Failure, SPOFs), second-order MCSs (MCSs with two events), and so on, complex scenarios may involve a very large number of MCSs in each one of these size-based categories. The methodology proposed in this work can be used to decrease such complexity in FTA by efficiently identifying MCSs with maximum probability in fault trees. As briefly discussed in Reference [10], its specific application to FTA requires a slight semantic variation, since cut sets in fault trees are intended to satisfy the root of the tree, while the present approach looks for cut sets that falsify the target of AND/OR dependency graphs. Generally speaking, our methodology differs from previous works in that: (1) we consider weights and perform a MaxSAT-based analysis to find minimal stochastic cut sets in general AND/OR graphs (instead of all MCSs); and (2) we provide a general framework that supports many SAT solvers in parallel instead of a particular one.

In structural terms, strict logical trees are a particular case of AND/OR graphs, and therefore, our approach is able to cover tree-based models as well as DAGs and general dependency graphs. In addition, the use of tree forests to separately model different parts of the system requires a delicate analysis of components that are shared across multiple trees [64]. Note that DAGs cannot have cycles by definition [47]. However, real cyber-physical models might also be cyclic, thus presenting the interdiction problem [4]. Our approach also allows AND/OR graphs with cycles between

components, as discussed in Section 5. Regarding stochastic aspects, a common measure used on these tree-based models is the aggregate probability of failure for a given top-level event. Our metric is complementary to such a measure and also provides a clear indication of how the MLMCS failure probability could be decreased due to the direct identification of the critical components.

Another close research area to our problem includes the domain of attack graphs [46, 51]. Attack graphs are mainly focused on depicting the many ways in which an attacker may compromise assets in a computer network. Among the many existing classes of attack graphs, **logical attack graphs (LAGs)** are one of the most commonly used [5, 15, 16, 57, 74]. Mathematically, AND/OR graphs are similar to the structures considered in LAGs. The difference in this work is that we use AND/OR graphs from an operational dependency perspective to identify the set of critical components that are most likely to fail, i.e., a vertex cut, and therefore compromise the entire mission. Attack graphs, however, are focused on modelling how multi-stage attacks can be carried out through a network towards the attacker's objective. Bayesian attack graphs also constitute an important research area [55, 59]. As opposed to finding potential critical points of failure, these works are focused on understanding the likelihood of an attacker compromising a given asset. SAT-based techniques have also been used in the area of IT network hardening and security management [9, 39, 40]. Our work is complementary to attack graphs, though we plan to incorporate this technique as a means to estimate potential failures due to cyber attacks.

The identification of critical nodes in complex networks is a well-studied problem [6, 18, 49, 67]. From a graph-theoretical perspective, our approach looks for a minimal weighted (in log-space) vertex cut in AND/OR graphs. This is an NP-complete problem as shown in References [27, 28, 29, 43]. While well-known algorithms such as Max-flow Min-cut [33] and variants of it could be used to estimate such a metric over OR graphs in polynomial time, their use for general AND/OR graphs is neither evident nor trivial, as they may fail to capture the underlying logical semantics of the graph. In that context, we leverage state-of-the-art techniques that excel in the domain of logical satisfiability and Boolean optimisation problems [23]. Other attempts to identify critical components have been made in the domain of network centrality measurements [25]. While useful in many types of scenarios [69], most of them are focused on OR-only graph-based models for IT networks and do not cover AND/OR semantics. Other important approaches targeting cyber-physical systems include probabilistic model checking and formal methods [30, 42, 73, 78], security metrics and graph-based methods [34, 61, 70, 72], system design analysis [19], real-time monitoring [12, 38], structural controllability aspects [3, 52, 58, 60], among others. The focus of this work is on the identification of CPS components that are most likely to jeopardise the system's mission.

3 PRELIMINARY CONCEPTS AND DEFINITIONS

3.1 Mission-oriented Cyber-physical Systems

The concept of *mission*, as used in this article, is formalised as follows:

Definition 1 (Mission). A mission is a goal-oriented process that involves a clearly defined objective and a success/failure criterion or measuring mechanism that is used to evaluate the outcome of the process with regard to its objective.

In the context of cyber-physical systems, the goal of a mission can be decomposed into sub-goals representing more specific objectives. These sub-goals are then logically combined to deliver the main mission's goal. Normally, any sub-system that implements a part of a mission is termed *critical* if its failure implies the failure of the mission. However, cyber-physical systems often involve complex interdependencies between components that might encompass multiple sub-goals simultaneously. Therefore, the failure of a single component may potentially affect different parts of the entire system. This requires a holistic analysis able to capture the mission's potential for

success/failure. The focus of this work is to analyse the most likely critical components from a system's perspective. In particular, we study the robustness of a mission design from a *dependency analysis* point of view.

We consider two classes of constructs within a mission's model: (1) goals and sub-goals representing the ability of the system to deliver a specific function or operational aspect (e.g., the ability of an aircraft to roll based on the operation of specific primary control surfaces) and (2) cyber-physical components (e.g., sensors, actuators, software agents) used to achieve these goals. The functional dependencies among components and goals are logically modelled using AND/OR graphs, as formalised in the next section.

3.2 AND/OR Dependency Graphs

We model a mission-oriented cyber-physical system M as a tuple $\langle G, t \rangle$ where $G = (V, E)$ is a directed AND/OR graph that represents the operational dependencies between cyber-physical components and goals in M , and $t \in V$ represents the goal of the system's mission. The graph G involves two main classes of vertices (atomic nodes and logical connectors), denoted as $V(G) = V_{AT} \cup L$.

- The set of atomic nodes $V_{AT} = V_{CP} \cup F$ models the different classes of cyber-physical components (V_{CP}) and goals (F) in the system. In particular, V_{CP} involves sensor nodes, actuators, and software agents (e.g., software running inside a PLC). The set F models notional nodes representing abstract concepts such as system function blocks or goals required to fulfil the system's mission (e.g., aircraft's roll control function).
- The set $L = \bigwedge \cup \bigvee$ represents the set of AND/OR nodes that model logical dependencies between atomic nodes. Namely, $\bigwedge$ represents the set of logical AND nodes, and $\bigvee$ represents the set of logical OR nodes.

$E(G)$ corresponds to the set of edges connecting nodes, and their semantics depend on the type of nodes they connect. We consider two main classes of edges:

- An edge $(v, w) : v, w \in V_{AT}$ between two atomic nodes represents that node w requires the functionality provided by node v to operate normally.
- Edges involving AND or OR nodes are interpreted from a logical perspective. A node v immediately preceded by an OR node l , i.e., $(l, v) : l \in \bigvee \wedge v \in V_{AT}$, means that the operational purpose of v can be satisfied (v operates normally) if *at least one* of the incoming connecting nodes to the OR node l is also satisfied. Similarly, a node v immediately preceded by an AND node m , i.e., $(m, v) : m \in \bigwedge \wedge v \in V_{AT}$, will be satisfied if *all* of the incoming nodes to the AND node are also satisfied. Edges between logical nodes, i.e., $(l, m) : l, m \in L$, are also allowed, and their semantics follow the same logical interpretation as with atomic nodes.

Identifying critical sets of components whose simultaneous failure may cause the failure of the system mission is essential to address the MLMCS problem. In the next section, we recall this concept in the context of AND/OR dependency graphs.

3.3 Critical CPS Component Sets in AND/OR Graphs

Definition 2 (Critical set). Given an AND/OR graph G and a target node t , a node set X is *critical* with regard to t if its removal from G logically disconnects t from G . Mathematically, X is critical if the following predicate holds:

$$Q(G, t, X) \equiv t \notin V(\sigma(G, X)), \quad (1)$$

where function $\sigma(G, X)$ removes from G each node $x \in X$ and the nodes that depend on them following a logic-style propagation. In simple terms, the removal of a critical set X from G produces a recursive elimination of the nodes that depend on them, which in turn affects other nodes and finally impacts on the target node t , thus removing t from G . Algorithm 1, taken from previous work described in Reference [14], implements $\sigma(G, X)$.

ALGORITHM 1: Node set removal and dependency impact propagation

```

Name:  $\sigma(G, X)$ 
Input: Graph  $G = (V, E)$ , Nodes to remove  $X$ 
Output: Updated graph  $G = (V, E)$ 

1 while  $X$  is not empty do
2   Node  $n \leftarrow X.pop()$                                 // get first node
3    $M \leftarrow \{x \in V : (n, x) \in E\}$                 // nodes reached by  $n$ 
4   for  $x \in M$  do
5     if  $(x \in V_{AT})$  or  $(x \in \wedge)$  or  $(x \in \vee \wedge d_{in}(x) = 1)$  then
6        $X.append(x)$                                      //  $x$  must be removed
7     end
8   end
9    $V = V - \{n\}$                                        // remove  $n$  from  $G$ 
10   $E = E - \{(v, w) \in E : v = n \vee w = n\}$         // remove edges
11 end
12 return  $G = (V, E)$ 
  
```

Algorithm 1 takes as input an AND/OR graph G and a set X of nodes to remove, and returns G after deleting the nodes in X as well as the nodes that logically depend on them, and every edge related to the removed nodes. Essentially, for each node n that must be removed, Algorithm 1 analyses the nodes that depend on n (set M , line 3). A node x that depends on n will be affected only if x is an atomic node; an AND node; or an OR node with only one input left, calculated with function $d_{in} : V \rightarrow N$ that indicates the number of incoming edges for a given node (line 5). In any of these cases, node x is queued for removal (line 6). In other words, only an OR node with more than one input will remain in the graph when one of its inputs is removed. Finally, each node marked for removal is deleted from G along with its respective edges (lines 9–10).

4 PROBLEM SPECIFICATION

In this section, we formalise the MLMCS problem in AND/OR dependency graphs.

4.1 Problem Formalisation: Most Likely Mission-critical Set (MLMCS)

Let $p : V_{AT} \rightarrow [0, 1]$ be a probability function over the space of atomic nodes $V_{AT} = V_{CP} \cup F$. Function p denotes that each cyber-physical component $v \in V_{CP}$ has an independent probability of failure $p(v)$. Abstract concepts such as goals and sub-goals modelled by F do not have an inherent failure probability; they rely on cyber-physical components (V_{CP}) to be fulfilled. Therefore, we consider $p(v) = 0, \forall v \in F$. The objective is to identify, given an AND/OR dependency graph G and a mission goal t , the set of critical components $X \subseteq V_{CP}$ with maximum probability of joint failure that can compromise the mission. More formally, we define the most likely mission-critical component set as follows:

Definition 3 (Most likely mission-critical set (MLMCS)). Let C be the set of critical sets, thus $\forall Y \in C, Q(G, t, Y)$. The most likely mission-critical set is the critical node set $X \in C$ such that its joint probability of failure is maximum among all sets in C . Mathematically, $X \in C$ is MLMCS if:

$$\forall Y \in C, \prod_{x_i \in X} p(x_i) \geq \prod_{y_i \in Y} p(y_i). \quad (2)$$

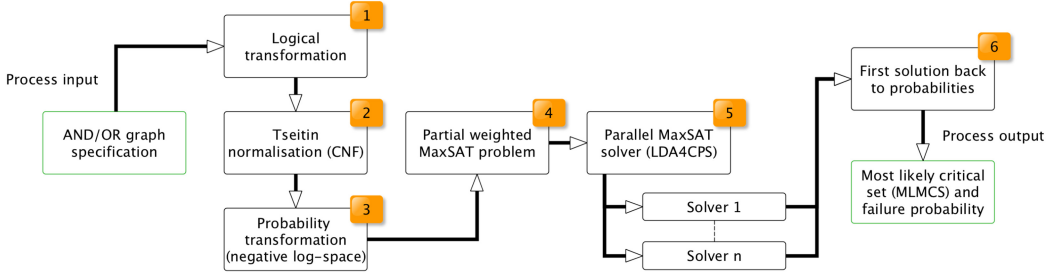


Fig. 2. Overall MLMCS resolution process.

We address this optimisation problem from a satisfiability perspective. In particular, we transform the AND/OR graph into a logical formulation in **conjunctive normal form (CNF)** that is used to build a **Satisfiability (SAT)** problem [20]. A SAT problem essentially looks for an assignment of truth values to the variables of a logical formula such that the formula evaluates to *true*. When a CNF formula also involves weights on each clause, the problem is called MaxSAT [23]. A MaxSAT problem consists in finding a truth assignment that maximises the weight of the satisfied clauses. Equivalently, MaxSAT minimises the weight of the clauses it falsifies [23]. When a set of clauses must be forcibly satisfied (called *hard clauses*), the problem is denominated Partial MaxSAT and it works on a subset of clauses (denominated *soft clauses*) that can be falsified if necessary. If the *soft clauses* have non-unit weights, then the problem is called Weighted Partial MaxSAT and it will try to minimise the penalty induced by falsified weighted variables. We use the latter to address the MLMCS problem.

5 MLMCS Resolution Strategy

In this section, we describe the overall MLMCS resolution process, which consists of six main steps, as illustrated in Figure 2. We have developed an open source tool called **Logical Dependency Analyser for Cyber-Physical Systems (LDA4CPS)**, which implements this process and is publicly available at Reference [8]. More technical details about LDA4CPS can be found in Annex A.

5.1 Logical Transformation

Given a target node t , the input graph G can be used as a map to decode the dependencies that node t relies on. Since these dependencies are presented as a logical combination of components connected with AND and OR operators, we say that node t is fulfilled (or can operate normally) if the logical combination is satisfied. In turn, these dependencies may also have previous dependencies, and therefore, they must be also satisfied. In that sense, G can be traversed backwards to produce a propositional formula that represents the different ways in which node t can be fulfilled. We call this transformation $f_G(t)$. To illustrate this idea, let us consider the example shown in Figure 1. In this case, $f_G(t)$ returns the formula:

$$f_G(t) = t \wedge (d \wedge ((a \wedge b) \vee (b \wedge c))).$$

Starting at the target node, the transformation recursively expands its logical dependencies until nodes with no incoming edges are reached. Cycles are handled by keeping a record of nodes already visited during each expansion. A simple algorithm to implement this transformation can be found at Reference [14]. The adversarial goal, however, is precisely the opposite, i.e., to disrupt

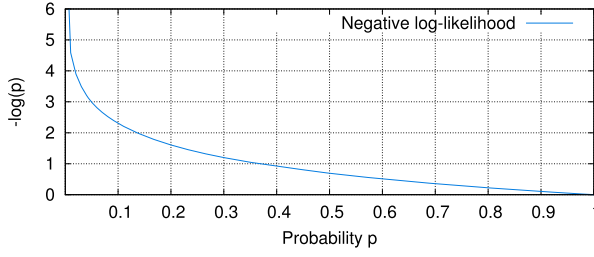


Fig. 3. Negative logarithmic space behaviour.

node t somewhere along the graph. Therefore, we are actually interested in satisfying $\neg f_G(t)$, which describes the means to disable t , as follows:

$$\neg f_G(t) = \neg(t \wedge (d \wedge ((a \wedge b) \vee (b \wedge c)))).$$

Under that perspective, a logical assignment such that $\neg f_G(t) = \text{true}$ (i.e., a SAT problem) will indicate which nodes must be compromised (i.e., logically falsified) to disrupt the operation of the system.

5.2 CNF Conversion with Tseitin

Normally, SAT formulations consider the input formula in **conjunctive normal form (CNF)**. To avoid exponential computation times during the CNF transformation process, we use the Tseitin transformation to produce, in polynomial time, a new formula in CNF that is not strictly equivalent to the original formula (because it introduces new variables) but is equisatisfiable [71].

In our particular problem, the weights that accompany each atomic graph node are independent probabilities of failure denoted as $p(v_i), \forall v_i \in V_{AT}$. However, MaxSAT is additive in nature, i.e., during the optimisation process, the values of the decision variables are added rather than multiplied. Therefore, we need to translate the problem to model joint probabilities, which are normally represented as the product of their values. While the multiplication of these decision variables is not a linear program, the product of these probabilities becomes a linear sum in logarithmic space.

5.3 Log-space Transformation

Since $\log(p(v_i) \times p(v_j)) = \log(p(v_i)) + \log(p(v_j))$, the product $p(v_i) \times p(v_j)$ can be computed in logarithmic space as a linear sum, and the actual result reconstructed as $p(v_i) \times p(v_j) = \exp(\log(p(v_i)) + \log(p(v_j)))$. Therefore, to maximise the product of weighted decision variables in MaxSAT, we transform the weights $p(v_i)$ into $w_i = -\log(p(v_i))$. The negative logarithm is used to produce positive values, as shown in Figure 3.

Figure 3 shows the behaviour of the negative log-likelihood function. This means that the lower a probability $p(v_i)$, the higher its negative log value w_i . Conversely, the higher the probability, the lower the corresponding $-\log$ value. Since the MaxSAT process will minimise the sum of the weights of the falsified variables, the solution (set of falsified variables) will have the lowest $-\log$ values, which correspond to the highest probabilities. Details about how LDA4CPS handles weights and decimal values can be found in Annex A. Table 1 shows the $-\log$ values (rounded up) of our base example illustrated in Figure 1, Case 1.

5.4 Weighted Partial MaxSAT Problem Specification

The normalised CNF formula obtained at step 2 represents the adversarial objective and has the form $h(v) = \text{CNF}(\neg f_G(v))$, where $h(v) = (v_{1i} \vee \dots \vee v_{1j}) \wedge \dots \wedge (v_{hi} \vee \dots \vee v_{hj})$. Therefore, we build the problem specification by constructing a soft clause for each decision variable (atomic

Table 1. Probability Assignments and $-\log$ Values (Case 1)

Probability / Node v_i	a	b	c	d	t
$p(v_i)$	0.2	0.03	0.2	0.02	0.01
$w_i = -\log(p(v_i))$	1.60944	3.50656	1.60944	3.91202	4.60517

node in the graph) $v_i \in V_{AT}$. These soft clauses tell the solver that each node v_i can be falsified with a certain penalty $w_i = -\log(p(v_i))$, which corresponds to the transformed failure probability of node v_i as exemplified in Table 1 for Case 1, Figure 1. The MaxSAT solver tries to minimise the total weight of falsified variables, and therefore, a solution to this problem yields a minimum vertex cut of the graph in logarithmic space. Since the lowest logarithmic values correspond to the highest probabilities of failure, the solution indicates the critical set with maximum joint probability of failure, i.e., the MLMCS. In our example (Case 1), the MLMCS is $\{a, c\}$ where the total penalty is $w_a + w_c = 1.60944 + 1.60944 = 3.21888$ (the lowest penalty among all possible cuts). In Case 2, Figure 1, $p(b) = 0.05$ and thus $w_b = 2.99573$, which makes $\{b\}$ the MLMCS.

5.5 Parallel MaxSAT Solver

As discussed later in Section 6, current solvers use a wide variety of techniques and optimisation methods to address SAT problems [23]. Depending on the composition and size of the AND/OR graph, we have experimentally observed that, quite often, solvers are very good at some instances and not that good at others. To address this issue, our tool LDA4CPS executes multiple pre-configured solvers in parallel and picks the solution of the solver that finishes first. This method provides a more stable behaviour in terms of performance and scalability. More details about our parallel SAT solving approach are reported in Appendix A.

5.6 Reverse Log-space Transformation

Once the MaxSAT solution has been found for a given system G with regard to a target node t , we compute the joint probability of failure $P_F(G, t)$ of the MLMCS by simply performing the reverse log-space transformation as follows: $P_F(G, t) = \exp(-1 \times \sum_i w_i)$, where i denotes the critical components found in the MaxSAT solution. This solution (MLMCS) has the highest probability among all critical sets capable of cutting the dependency graph with regard to its mission. Therefore, a joint failure of these components will prevent the cyber-physical system from delivering the mission it was built for. In our example, we have that for Case 1, $P_F(G, t) = \exp(-(1.60944 + 1.60944)) = 0.04$, while for Case 2, $P_F(G, t) = \exp(-2.99573) = 0.05$.

6 EXPERIMENTAL EVALUATION

In this section, we present an analytical evaluation of the proposed approach regarding performance and scalability over randomly generated AND/OR graphs of different size and composition. These experiments have been performed with our tool LDA4CPS [8] using a MacBook Pro 2018, 2.9 GHz Intel Core i9, 32 GB 2400 MHz DDR4. The construction procedure for an AND/OR graph of size n (n nodes) is as follows: We first create the target node. Afterwards, we create a predecessor that has one of the three types (atomic, AND, OR) according to a probability given by a compositional configuration predefined for the experiment. For example, a configuration of (60, 20, 20) means 60% of atomic nodes, 20% of AND nodes and 20% of OR nodes. We repeat this process creating children on the respective nodes until we approximate the desired size of the graph, n . The first set of experiments, illustrated in Figure 4, shows the behaviour of our approach as AND/OR graphs become larger, up to 20,000 nodes.

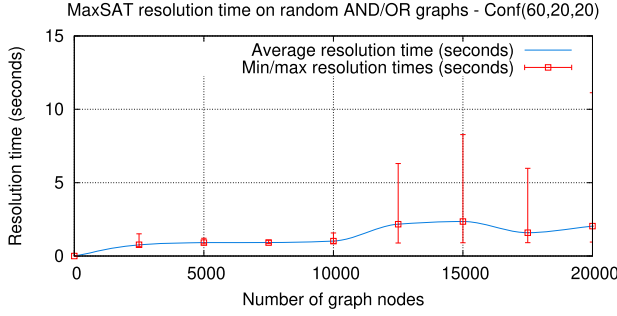


Fig. 4. Scalability/performance evaluation up to 20,000 nodes.

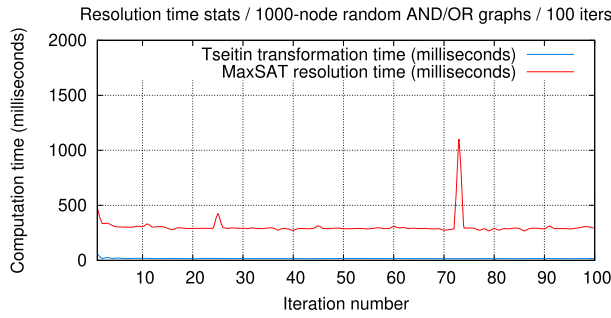


Fig. 5. MaxSAT analysis over 1,000-node AND/OR graphs.

Each experiment is focused on a specific graph size $n \in [0, 2,500, 5,000, \dots, 20,000]$, in which we evaluate 10 different random graphs of size n and measure maximum, minimum, and average MaxSAT resolution times. In general terms, we have observed that the performance of the approach is relatively stable despite its NP-complete nature, and also presents good scalability properties, being able to tackle graphs with thousands of nodes in a matter of seconds. The bars represent minimum and maximum resolution times for each experiment, which means that different AND/OR graphs with the same size may have quite different resolution times. This also means that some larger graphs may be solved faster than some smaller graphs. To further analyse this aspect, Figure 5 takes a closer look at the method's performance over 100 different graphs with the same size (1,000 nodes).

We can observe that, while the Tseitin transformation time is pretty much constant for all cases, the MaxSAT resolution is quite stable with the exception of a couple of cases where times are considerably higher. This is due to the structural composition of each specific graph, which translates into a different logical formulation. These formulas may vary in the distribution of variables and the connectivity (AND/OR) among them, and therefore, some formulations are easier to solve than others [14]. In addition, we have observed that different solvers may behave quite differently with the same AND/OR graphs. We have submitted a significant part of our evaluation benchmark over AND/OR graphs to the MaxSAT Evaluation 2019¹ [13], where our dataset has been part of the body of optimisation problems used to evaluate the participant MaxSAT solvers. Interestingly, none of the nine solvers evaluated with our dataset performed better than the others on every instance. The reason is that distinct MaxSAT solvers generally use very different resolution techniques. As a consequence, the result obtained from diversity (using solvers in parallel) is in

¹<https://maxsat-evaluations.github.io/2019>.

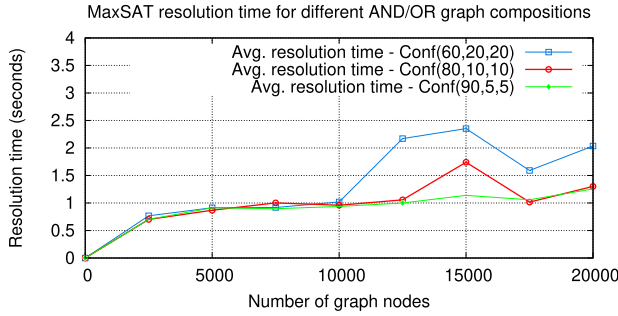


Fig. 6. Resolution times for different graph compositions.

fact quite good. Our tool LDA4CPS currently uses two different MaxSAT solvers, namely, SAT4J [65] and a Python-based linear programming approach using Gurobi [36]. We have also observed that the composition of AND/OR graphs also affects the average resolution time. Figure 6 shows resolution times over graphs up to 20,000 nodes for three different graph compositions.

The first composition type, previously shown in Figure 4, has a configuration of (60,20,20) and presents the longest computation time on average. However, a configuration with more atomic nodes and fewer AND/OR nodes, e.g., (80,10,10), actually decreases the average time substantially. This is explained by the fact that these graphs have more dependent nodes in sequence with fewer AND/OR nodes among them. Therefore, solvers are able to speed up the resolution process and find the appropriate vertex cut. Due to the lack of large real-world models and datasets, it is hard to actually analyse the performance of the proposed approach for every potential scenario. In particular, randomness may not always ensure complete coverage regarding corner or core cases. However, our results indicate that the proposed approach is, in general, very efficient and capable of dealing with large graphs involving tens of thousands of nodes in seconds. In the next section, we present the results of our method over a realistic scenario involving complex aircraft systems.

7 CASE STUDY ON COMPLEX AIRCRAFT SYSTEMS

This case study is focused on the analysis of quad-redundant flight control systems as presented in Reference [35], which considers an AND/OR dependency model similar to the one proposed in this work. More specifically, we study the complex interdependencies involved in the pitch control function as an example.

7.1 Flight Control Systems Overview

The flight dynamics of a typical fixed-wing aircraft are governed by three main *functions* that allow the vehicle to rotate in three different dimensions with regard to its gravity centre, as illustrated in Figure 7 (base plane image taken from Reference [21]). These functions are:

- **Pitch**, which controls the Y axis (from wing tip to wing tip), allowing the aircraft to move its nose up or down, thus increasing or decreasing its *angle of attack*,
- **Roll**, which controls the X axis (from nose to tail), allowing the aircraft to rotate according to its longitudinal axis, and
- **Yaw**, which controls the Z axis (from top to bottom), allowing the aircraft to move the nose left or right.

The physical manipulation of these main functions is achieved through the aircraft's *control surfaces* [31], as illustrated in Figure 7. These are aerodynamic devices that produce torques about

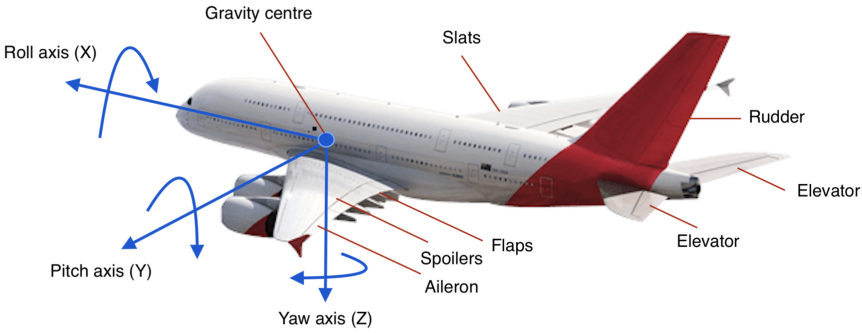


Fig. 7. Primary flight functions and control surfaces.

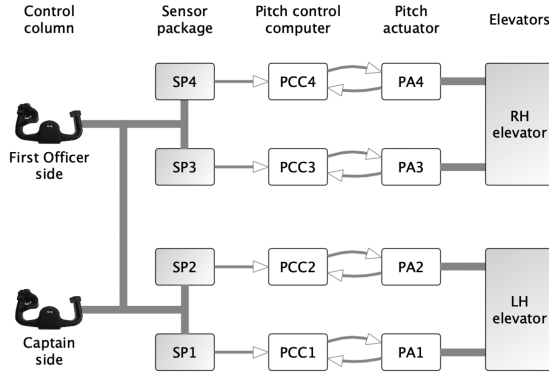


Fig. 8. System block diagram of a quad-redundant pitch control function [35].

the axes when deflected and allow pilots to control the aircraft’s motion about the three axes of rotation. Each function has a primary control surface: the *elevators* are the primary control surface for pitch, the *rudder* controls the aircraft’s yaw, and roll is achieved by operating the *ailerons* (see Figure 7).

Secondary control surfaces include *spoilers*, *flaps*, and *slats*, among others. In some cases, these devices can be used with or instead of primary control surfaces. For example, flight spoilers are devices intended to reduce the lift on the wings by deflecting (spoiling) the air stream passing over them. Under certain circumstances, spoilers can be used to control roll. In these cases, the roll rate may not be optimal but still effective. This simplified example aims at illustrating the fact that a given flight function might be also achieved by logically combining other aircraft functions and components.

As explained in Reference [35], the **Minimum Acceptable Control Criterion (MACC)** for the mission **Continued Safe Flight and Landing (CSFL)** requires all three main functions (pitch, roll, yaw) in operation. For the sake of clarity and simplicity, we only focus on the pitch control function as an example.

7.1.1 Pitch Control Function. Figure 8 shows a typical quad-redundant fly-by-wire system to control the pitch axis of an aircraft. As opposed to mechanical flight control systems, fly-by-wire systems involve intermediate computers that interpret electronic signals coming through wires from the movement of flight controls and determine how to operate actuators and control surfaces [32].

As explained in Reference [35], the aircraft pitch axis is controlled by the pilots via the control columns installed on the flight deck, as illustrated in Figure 8. A **position sensor package (SP)** senses the movement of the control column and sends its position to the **pitch control computer (PCC)**, which uses this information to control the **elevator surface actuators (PA)**. Position sensors located at the actuator allow closed loop control of the pitch actuator. The pitch control function involves four redundant control channels, thus *quad-redundant*, with two actuators moving each one of the two **elevator surfaces (RH and LH)**. In this case, the pitch MACC for CSFL is fulfilled if at least one control channel is available to operate a control surface.

7.2 AND/OR Graph Model and What-if Analysis

Based on the material provided in Reference [35], we have built a simplified AND/OR-based model of the pitch control function (the mission) of a quad-redundant flight control system. In this example, the aircraft has two main engines (turbines EN1 and EN2) that deliver power to its electric power supplies and hydraulic power units. For the sake of simplicity, we have omitted some intermediate electrical units and we consider only two hydraulic power supplies (HPS1 and HPS2). The fly-by-wire computers have a dual redundant electrical power supply (PCCx_PWR1 and PCCx_PWR2) and an energy supply installation (PCCx_INST). The surface actuators are hydraulically powered.

The objective of this section is to exemplify the use of MLMCS as a quantitative analysis tool to evaluate different system configurations (an activity that is normally conducted at design time to assess system attributes and requirements [62]). To do so, we use our tool LDA4CPS to analyse the pitch function model over six scenarios with different failure probability assignments (Ⓔ to Ⓕ), as shown in Table 2. The probabilities involved in our what-if analysis are synthetic. In practice, however, these probabilities should correspond to the appropriate failure estimates for each CPS component (e.g., per mission, per hour), depending on the analysis being conducted.

7.2.1 Pitch Control Function Analysis. The first scenario, Ⓔ, constitutes a base probability assignment in which our tool LDA4CPS identifies the elevator surfaces LH and LR (marked with • in Table 2) as the MLMCS with a joint failure probability of 0.0016. Recall that any other critical set under this configuration will have, by definition, a lower probability of failure. During execution, LDA4CPS consumes a JSON file representing the input AND/OR model and outputs a new JSON file with the solution. This output file is used to graphically display the AND/OR graph along with the MLMCS (red dashed circles) on a web browser, as shown in Figure 9. While visualisation aspects are not the main objective of this work, Figure 9 exemplifies the use of our technique for visual representations and also shows the complexity that this type of system may pose for analysis to the naked eye. In that context, the main goal of the proposed approach and LDA4CPS is to provide a machine-readable analytical output that can be used by administrators and external systems for risk analysis purposes.

System upgrades. A natural use of the proposed metric is to make decisions based on the identified critical components. Scenario Ⓕ shows a hypothetical case where both elevator surfaces LH and LR have been upgraded and now have a lower failure probability (0.001). Interestingly, the updated metric now indicates a considerably different MLMCS involving sensor packages SP1, SP2, SP3, and SP4. As expected, the MaxSAT resolution yields a much lower joint probability of failure (0.00000625) for this critical set.

Component failures. Understanding whether the mission can still be fulfilled under the presence of failures is vital during design. Our approach can easily model failed components v_i by simply considering $p(v_i) = 1.0$. Scenario Ⓕ involves a *what-if* situation where two different actuators (PA3 and PA4) have failed (marked with ▼ in Table 2) due to freezing conditions. In this

Table 2. Critical Sets (Marked ●) on What-if Scenarios with Different Failure Probabilities and Failed Components (Marked ▼)

Component	Description	Ⓐ	Ⓑ	Ⓒ	Ⓓ	Ⓔ	Ⓕ
LH	Left elevator surface	0.04 (●)	0.0001	0.0001	0.0001	0.0001	0.0001
RH	Right elevator surface	0.04 (●)	0.0001	0.0001	0.0001	0.0001	0.0001
PA1	Surface actuator 1 (moves LH)	0.002	0.002	0.002	0.002	0.002	0.002
PA2	Surface actuator 2 (moves LH)	0.002	0.002	0.002	0.0002	0.002	0.002
PA3	Surface actuator 3 (moves RH)	0.002	0.002	1.0 (▼)	1.0 (▼)	1.0 (▼)	1.0 (▼)
PA4	Surface actuator 4 (moves RH)	0.002	0.002	1.0 (▼)	1.0 (▼)	1.0 (▼)	1.0 (▼)
PCC1	Pitch control computer 1 (controls PA1)	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
PCC2	Pitch control computer 2 (controls PA2)	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
PCC3	Pitch control computer 3 (controls PA3)	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
PCC4	Pitch control computer 4 (controls PA4)	0.0005	0.0005	0.0005	0.0005	0.0005	0.0005
PCC1_PWR1	PCC1's electrical power supply 1	0.03	0.03	0.03	0.03	0.03	0.03
PCC1_PWR2	PCC1's electrical power supply 2	0.03	0.03	0.03	0.03	0.03	0.03
PCC1_INST	Energy supply installation for PCC1	0.01	0.01	0.01	0.01	0.01	0.01 (●)
PCC2_PWR1	PCC2's electrical power supply 1	0.03	0.03	0.03	0.03	0.03	0.03
PCC2_PWR2	PCC2's electrical power supply 2	0.03	0.03	0.03	0.03	0.03	0.03
PCC2_INST	Energy supply installation for PCC2	0.01	0.01	0.01	0.01	0.01	0.01
PCC3_PWR1	PCC3's electrical power supply 1	0.03	0.03	0.03	0.03	0.03	0.03
PCC3_PWR2	PCC3's electrical power supply 2	0.03	0.03	0.03	0.03	0.03	0.03
PCC3_INST	Energy supply installation for PCC3	0.01	0.01	0.01	0.01	0.01	0.01
PCC4_PWR1	PCC4's electrical power supply 1	0.03	0.03	0.03	0.03	0.03	0.03
PCC4_PWR2	PCC4's electrical power supply 2	0.03	0.03	0.03	0.03	0.03	0.03
PCC4_INST	Energy supply installation for PCC4	0.01	0.01	0.01	0.01	0.01	0.01
CTRL_CAP	Captain's control column	0.00001	0.00001	0.00001	0.00001	0.00001	0.00001
CTRL_OFF	1st Officer's control column	0.00001	0.00001	0.00001	0.00001	0.00001	0.00001
SP1	Position sensor package 1 (Captain's column)	0.05	0.05 (●)	0.05 (●)	0.05 (●)	0.05 (●)	0.001
SP2	Position sensor package 2 (Captain's column)	0.05	0.05 (●)	0.05 (●)	0.05 (●)	0.05	0.05
SP3	Position sensor package 3 (1st Officer's column)	0.05	0.05 (●)	0.05	0.05	0.05	0.001
SP4	Position sensor package 4 (1st Officer's column)	0.05	0.05 (●)	0.05	0.05	0.05	0.05
EN1	Engine 1	0.000001	0.000001	0.000001	0.000001	0.000001	0.000001
EN2	Engine 2	0.000001	0.000001	0.000001	1.0 (▼)	0.000001	0.000001
HPS1	Hydraulic power supply 1	0.000001	0.000001	0.000001	0.000001	0.000001	0.000001
HPS2	Hydraulic power supply 2	0.000001	0.000001	0.000001	0.000001	1.0 (▼)	1.0 (▼)
MLMCS failure probability:		0.0016	0.00000625	0.0025	0.0025	0.05	0.01

case, the mission can still be achieved, since the left elevator can be operated by the remaining actuators PA1 and PA2. However, the MLMCS is now composed of only sensor packages SP1 and SP2 with a joint failure probability of 0.0025 (higher than that of Ⓐ).

Scenario Ⓓ considers the case where, besides actuators PA3 and PA4, engine EN2 has also failed (e.g., bird strike). In that context, one would think that the remaining engine EN1 would constitute the MLMCS. However, under this specific configuration, sensor packages SP1 and SP2 have a higher joint probability of failure. Since they are essential to operate the remaining actuators PA1 and PA2 (see graph), the fact of having just one operational engine (with a failure probability of 0.000001) degrades to a lower-criticality concern.

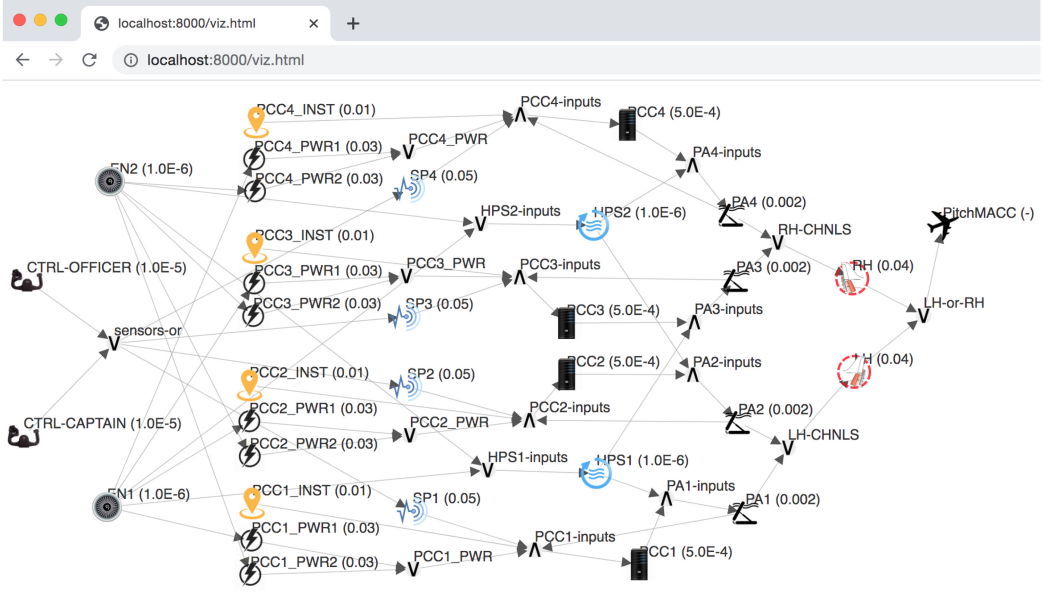


Fig. 9. LDA4CPS output graph of pitch MACC mission for scenario S1.

More interesting, considering the above context and the particular AND/OR graph structure, a failure in the hydraulic power system might be more critical than the failure of a turbine. Scenario ⑤ shows the case where, besides PA3 and PA4, HS2 has failed instead of EN2. Because HS2 powers PA2 and PA4 (failed already), the failure of HS2 affects PA2, leaving only the option of PA1. Since PA1 requires SP1 to work properly, and SP1 has a higher failure probability, SP1 becomes the MLMCS with a failure probability of 0.05.

Diversification and failures. From a redundancy perspective, the use of different components might be beneficial to diversify potential failures under certain conditions and/or cyber attacks [50, 76]. Scenario ⑥ illustrates a case involving the same failures as in ⑤, but in addition, sensor packages SP1 and SP3 have been modified and now have a lower failure probability of 0.001 (instead of 0.05). In this case, the failure probability of PCC1_INST takes over, thus changing the most likely probability of failure to 0.01. This is because the impact of diversifying components SP1 and SP3 caused PCC1's energy supply installation (PCC1_INST) to become a critical component, since PCC1 is the only computer capable of controlling the surface actuator PA1 to manipulate the left surface elevator LH.

7.2.2 Mission Simulations. To validate the MLMCS predicted by our approach, we have also performed mission simulations for specific configurations. The objective is to analyse whether the most common critical set that causes the mission failures is in fact the MLMCS. We use scenario ⑤ as an example. During the experiment, we have performed five iterations involving 8,000 mission simulations each. On each simulation, we traverse the graph nodes to simulate their failure according to their failure probability. That is, for each node v_i , a real value $r \in [0, 1]$ is randomly selected and v_i is marked as failed if $r \leq p(v_i)$. Afterwards, we analyse whether the mission has failed, and if so, which failed components were involved. Figure 10 shows a heatmap indicating the active participation of each component on the missions that have failed.

In this scenario, the MLMCS is $\{LH, RH\}$ with a failure probability of 0.0016. The results indicate that, as predicted by our approach, LH and RH are the components with more joint participation

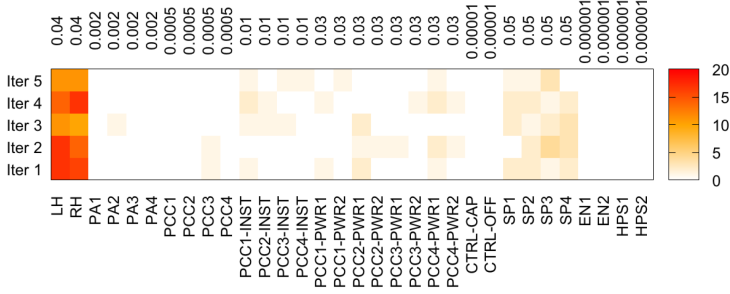


Fig. 10. Mission simulation heatmap.

in the failure of the simulated missions. Note that some missions have failed due to the failure of other components (lighter colours). However, the MLMCS has been the most dominant critical set (darker colours). In numbers, out of a total of 40,000 simulations, we have observed 78 failed missions, 63 of which correspond to the MLMCS. This gives a percentage of $63/40,000 = 0.001575$, which matches well the estimated MLMCS failure probability of 0.0016.

7.2.3 Discussion and Further Observations. The proposed methodology enables what-if analysis at different design stages and provides actionable information for decision making as exemplified in this section. In addition, the proposed AND/OR graph-based model enables composability and granularity refinement. This means that specific abstract components, goals, or functions, can be further detailed with their appropriate sub-entities. For example, we have simplified actuators as single components. In practice, however, actuators might involve redundant subcomponents in series or parallel configurations to achieve their goal [68]. Therefore, the interdependencies between these subcomponents can be also modelled using the proposed AND/OR graph methodology.

8 Extended MLMCS applications

8.1 System Hardening via Iterative MLMCS-based Improvements

During the design of a mission, it is common to define a baseline of what level of risk the system is willing to tolerate. Given the complexity of modern cyber-physical systems, improving multiple interconnected system components to achieve an acceptable level of failure risk can be troublesome and complex. In this section, we describe a threshold-oriented iterative strategy that uses MLMCS to take the failure probability of any system-critical set under a given threshold.

8.1.1 MLMCS-based Hardening Strategy. The overall hardening strategy is illustrated in Figure 11. The process receives as input an AND/OR graph describing the dependencies and failure probabilities of the system, and a threshold T that indicates the upper limit of acceptable failure risk for any MLMCS. The objective is to update the system through a sequence of improvements until the failure probability of any critical set that can compromise the mission is under said threshold.

The process first computes the MLMCS (set X) and its joint probability of failure $p(X) = \prod_{c \in X} p(c)$ for the specified AND/OR graph. If the obtained joint probability $p(X)$ is under the threshold T , then the process successfully ends. This is because, by definition, there is no other MLMCS with a probability of failure greater than $p(X)$. If this is not the case, then the critical components in X are sorted in descending order according to their failure probabilities and stored in the list L . When the MLMCS has just been computed, L is not empty and the process will pick and remove the first component from L , termed as c . Component c is then analysed for potential improvements, for example, using a different configuration to decrease its failure probability,

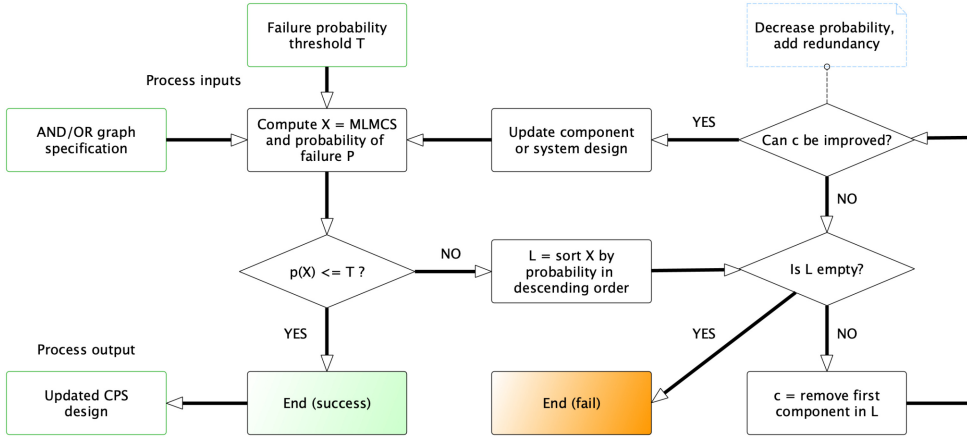


Fig. 11. Hardening strategy with MLMCS.

replacing it with a component from a different vendor, or adding redundant components in the system with the same role as c and implemented as a logical disjunction.

If c can be improved, then the MLMCS is recomputed over the updated system, since this change will decrease the current probability $p(X)$ and might point to a different MLMCS Y with a probability $p(Y)$ that, by definition, will be equal to or lower than the current value of $p(X)$. Note that the MLMCS in the next iteration can also be the same, but with a lower probability due to the update in c .

If c cannot be improved, then the next component in L is selected and the process continues as explained before. If list L is empty, then it means that every component in the current MLMCS has been analysed and none of them can be improved. Since the current MLMCS has the highest failure probability among all critical sets, and this probability cannot be lowered, the process exits with a *failed* output to indicate that there still exists a critical set of components with a joint probability of failure higher than the specified threshold. Therefore, the threshold cannot be satisfied unless improvements to the system can be granted and implemented.

8.1.2 Algorithm Termination. The hardening process always terminates as long as each improvement for a given component c either: (a) reduces its independent probability of failure p_c to p'_c such that $p'_c < p_c$, or (b) incorporates redundant components with c 's role in the form of a logical disjunction in which probabilities are multiplied and therefore the failure probability of the whole disjunction is much lower than that of component c alone.

The algorithm convergence is guaranteed, because each improvement monotonically decreases the joint probability of the current MLMCS X . To explain why, let $p(X_{BE})$ and $p(X_{AF})$ be the joint failure probability of the current MLMCS X before and after updating a component $c \in X$, respectively. By definition, an improvement in X implies that $p(X_{BE}) > p(X_{AF})$. If after updating c , the same MLMCS X is selected again in the next iteration, then it means that X still has the highest probability of failure among all critical sets, but now its value $p(X_{AF})$ is strictly lower than that of the previous iteration $p(X_{BE})$. Conversely, if a different critical set Y comes up in the next iteration, then it is because Y now has the highest probability of failure $p(Y)$. However, $p(Y)$ will always be lower (or equal in the case of a tie) than $p(X_{BE})$ from the previous iteration. In other words, $p(Y)$ can never be higher than $p(X_{BE})$, otherwise Y would have been selected as the MLMCS in the previous iteration instead of X . In the case of a tie between two MLMCS, one MLMCS will be processed first and after it is improved, its failure probability will be decreased

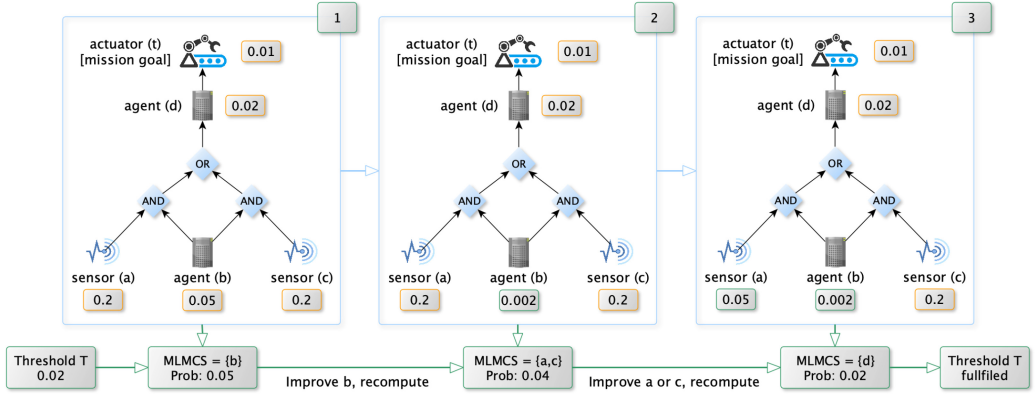


Fig. 12. Hardening example for initial scenario.

so the next one will take its place in the next iteration. As explained before, if no improvements can be made to any of the components within the current MLMCS, then the requested threshold cannot be satisfied and the process will terminate.

8.1.3 Example Execution. Figure 12 illustrates a simple example of the proposed hardening methodology based on our initial scenario.

The process begins with the input AND/OR graph illustrated in the first block of Figure 12 and a failure probability threshold T of 0.02. In the first iteration, the MLMCS is $\{b\}$ with a probability of 0.05. Since $p(b) > T$, b is improved and the process iterates. Block 2 illustrates the updated system where now b has a failure probability of 0.002 (improvement). The MLMCS for the second iteration is $\{a, c\}$ with a joint probability of failure of 0.04. Since both a and c have the same probability, the first component, i.e., a , is updated and its new probability is 0.05 (improvement), as shown in Block 3. The third and last iteration indicates that the MLMCS is $\{d\}$ with a failure probability of 0.02. Since $p(d) = T$, the algorithm terminates and outputs the last system configuration.

It is important to note that when a component c is improved within an MLMCS, the next iteration might yield a completely different MLMCS. This will normally depend on the type of improvement that can be performed on c (or the design itself), and the new probability of failure. However, an important feature of the strategy is that it has the ability to completely change the direction within the search space and systematically focus on the most relevant components.

8.1.4 Discussion about Tradespace Analysis and Improvement Costs. The decision of how to improve a specific component can be a challenging problem. A fundamental research domain dedicated to this problem is tradespace analysis [62], which can involve the study of multiple factors before deciding how such a component or system might be improved. Currently, our methodology does not cover specific tradespace analysis techniques. Instead, we assume that this activity is performed externally by a human analyst or another software system. However, our methodology clearly states where these activities take place, and we plan to include these in future work.

Another important aspect involves the inclusion of improvement costs. While our strategy does not directly focus on costs, it does provide a localised and reduced search space, i.e., the MLMCS itself, where more advanced analysis can be conducted over the set of MLMCS components at any given iteration in the algorithm. For example, our methodology currently sorts the MLMCS components by descending probability of failure to converge faster to the specific threshold. However, a different sorting tactic could use the following component valuation function to find a balance between the failure probability of each component c in MLMCS X and the cost required to improve

them:

$$f_{\alpha,\beta}(c) = \alpha \times p(c) + \beta \times (1 - g(c)), \quad (3)$$

where $g(c) = \text{cost}(c) / \sum_{c_i \in X} \text{cost}(c_i)$ normalises the costs of the involved components in X ; and $\alpha, \beta \in [0..1]$ are real coefficients such that $\alpha + \beta = 1$.

The objective of these coefficients is to emphasise the importance of each factor (failure probability vs. costs) within the new valuation function $f_{\alpha,\beta}(c)$. When $\alpha = 0$, i.e., $f_{0,1}(c)$, the function is completely focused on the budget. Conversely, when $\beta = 0$, i.e., $f_{1,0}(c)$, the function gives us the original function focused on the failure probability. In that context, we can provide a balance between the two by parameterising the function with intermediate values $\alpha = 0.5$ and $\beta = 0.5$ ($f_{0.5,0.5}$). Ideally, we want to start the analysis with the component that has the highest failure probability and the lowest cost, thus we consider $1 - g(c)$ in Equation (3) to reverse costs (i.e., the highest costs become the lowest costs and vice versa). By using function $f_{\alpha,\beta}(c)$ within the sorting mechanism, we can sort components by descending order to provide a balance between convergence speed and cost reduction.

Both cost optimisation and tradespace analysis are key areas for the design of more secure and resilient cyber-physical systems. As future work, we plan to further investigate these aspects from an integrated perspective as a way to find an optimal balance between improvement types, costs, and failure probability reduction.

8.2 Forensic Investigations

MLMCS can also be used to prioritise components during root cause analysis and forensic investigations. When a mission fails, it is fundamental to understand why and how it happened to learn and avoid the same errors in future missions and system designs. The identification of the MLMCS in a failed mission can provide guidance to prioritise which components should be analysed first to look for failures or potential cyber compromise, and then determine what caused the failure of the mission. In this section, we describe an MLMCS-based forensic methodology to investigate the cause of mission failures and present a comprehensive experimental evaluation over mission simulations with different characteristics.

8.2.1 Forensic Methodology. In general terms, the proposed methodology receives as input the corresponding AND/OR graph $G(V, E)$, performs a systematic forensic search by leveraging the identification of MLMCS to guide the analysis, and returns a set of failed nodes F that conjunctively are deemed as the cause of the mission failure. Algorithm 2 describes the proposed strategy.

Algorithm 2 starts by computing the MLMCS M of the input AND/OR graph (line 3) and then sorts its components by probability in descending order, i.e., most likely failed nodes first (line 4). The algorithm traverses the list of components in M to understand whether they have failed or not. By definition, if the whole set of components within an MLMCS fails, then the mission fails. Therefore, the algorithm stops if the current MLMCS under analysis has been fully explored (lines 7–9). Otherwise, the first component n is picked and removed from M (line 11). To analyse each potential component only once, the forensic process stores in set A the nodes that have been already analysed (line 13). If after performing a forensic analysis over component n (line 14) the investigation finds it to be non-functional due to an inherent fault or cyber compromise, then it is marked as failed in the graph, i.e., $p(n) = 1.0$, and added to the set of failed nodes (lines 15–18). Otherwise, the component is marked as functional and the MLMCS is recomputed, since the current MLMCS cannot be the cause of the mission failure (lines 20–22). With this new information stored in the graph, the new MLMCS will redirect the search towards the new most likely failed component set, thus providing guidance and speeding up the investigation by reducing the search space. The algorithm

ALGORITHM 2: Forensic search

```

Name: search(G)
Input: Graph  $G = (V, E)$ 
Output: Component set  $F$  that caused the mission failure

1   $A \leftarrow \{\}$  // initialise set of analysed nodes
2   $F \leftarrow \{\}$  // initialise set of failed nodes
3   $M \leftarrow \text{MLMCS}(G)$  // compute the MLMCS
4  sort( $M$ ) // sort  $M$  by descending probability
5  finished  $\leftarrow$  false // set up Boolean flag
6  while not finished do
7      if  $M$  is empty then
8          finished  $\leftarrow$  true // the last MLMCS has been fully explored
9      end
10     else
11         Node  $n \leftarrow M.\text{removeFirst}()$  // get first node from MLMCS
12         if  $n \notin A$  then
13              $A \leftarrow A \cup \{n\}$  // add node  $n$  to set of analysed nodes
14             Status  $s \leftarrow \text{forensics}(n)$  // determine if node  $n$  has failed or not
15             if  $s == \text{failed}$  then
16                  $p(n) \leftarrow 1.0$  // mark node  $n$  as failed
17                  $F \leftarrow F \cup \{n\}$  // add node  $n$  to set of failed nodes
18             end
19             else
20                  $p(n) \leftarrow 0.0$  // mark node  $n$  as functional
21                  $M \leftarrow \text{MLMCS}(G)$  // recompute the MLMCS
22                 sort( $M$ ) // sort  $M$  by descending probability
23             end
24         end
25     end
26 end
27 return  $F$ 

```

terminates when all components within the last computed MLMCS have been analysed so list M is empty, and returns the set of failed nodes that have caused the failure of the mission.

8.2.2 Simple Example. Let us consider the AND/OR graph corresponding to scenario ⑤ described in the case study (Section 7.2.1). Let us now consider that the mission has failed due to the failure of the following set of components: $\{SP1, PCC2_{INST}, RH\}$. In this case, the methodology proceeds as described in Table 3.

It can be observed from the example that the strategy traverses the current MLMCS and will only enter into a new iteration if a suspected node in the current MLMCS has not failed. At the same time, failed nodes are stored in F and the algorithm terminates when the last MLMCS has been fully explored. The set of analysed nodes A allows the algorithm to skip these nodes, since they have been previously examined and declared as operational ($p(n) = 0.0$) or failed ($p(n) = 1.0$).

8.2.3 Analytical Experiments and Simulations. We have conducted a thorough ensemble of experiments to determine the feasibility and behaviour of the proposed forensic methodology. In this section, we present the experiments and simulations divided in three categories: (a) analysis of mission simulations over scenario ⑤ with random failures; (b) mission simulations over scenario ⑤ with random probabilities and failures; and (c) mission simulations on large random graphs.

(a) Case study scenario ⑤ with random failures. Figure 13 shows the results obtained after conducting 10,000 mission simulations using the AND/OR graph and failure probabilities described in Scenario ⑤, in the case study section. In this scenario, out of 10,000 simulations, only 25 missions

Table 3. Forensic Investigation Example

Iterations	MLMCS	Queries	Actions
Iteration 1	$MLMCS = \{LH, RH\}$	LH failed? $\rightarrow$ No	Set $p(LH) = 0.0$ $A = \{LH\}$ Recompute $MLMCS$
Iteration 2	$MLMCS = \{SP1, SP2, RH\}$	$SP1$ failed? $\rightarrow$ Yes	Set $p(SP1) = 1.0$ $A = \{LH, SP1\}$ $F = \{SP1\}$
		$SP2$ failed? $\rightarrow$ No	Set $p(SP2) = 0.0$ $A = \{LH, SP1, SP2\}$ Recompute $MLMCS$
Iteration 3	$MLMCS = \{PCC2_{INST}, RH\}$	$PCC2_{INST}$ failed? $\rightarrow$ Yes	Set $p(PCC2_{INST}) = 1.0$ $A = \{LH, SP1, SP2, PCC2_{INST}\}$ $F = \{SP1, PCC2_{INST}\}$
		RH failed? $\rightarrow$ Yes	Set $p(RH) = 1.0$ $A = \{LH, SP1, SP2, PCC2_{INST}, RH\}$ $F = \{SP1, PCC2_{INST}, RH\}$
		$MLMCS = \{\}$	End process
Cause of mission failure:		$F = \{SP1, PCC2_{INST}, RH\}$	

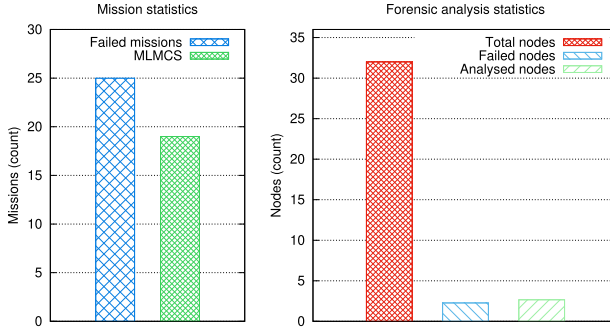


Fig. 13. Mission simulations over case study scenario with random failures.

failed, of which 19 corresponded to the MLMCS (i.e., the MLMCS was responsible for the failure of the mission). The graph structure in this case involves 32 cyber-physical components (atomic nodes). The results show that each failed mission involves, on average, 7.056% of failed nodes, while the forensic procedure analysed only 8.277% of the system nodes to find the critical set that led to the failure of the mission.

(b) Case study scenario with random probabilities and random failures. Figure 14 shows the results of mission simulations over the same AND/OR graph structure considered in our case study but assigning a random failure probability to each system component within a pre-established probability range. We have experimented with five probability ranges $[0.0, X]$ where $X \in \{0.02, 0.05, 0.1, 0.2, 0.4\}$. We have defined the largest range as $[0.0, 0.4]$ to avoid a massive number of failures in the system components. For each experiment, we have run 10,000 mission simulations where each one involves a different random probability assignment within the

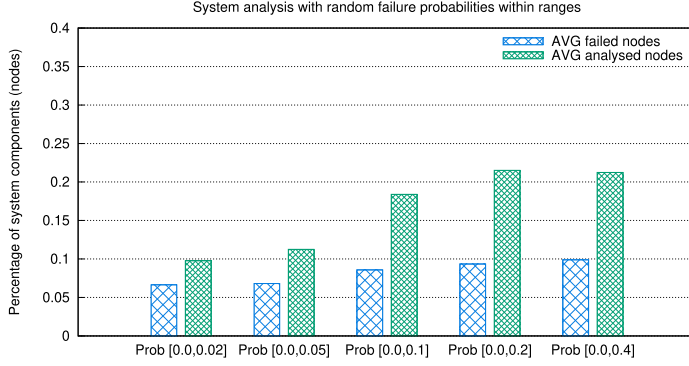


Fig. 14. Mission simulations over case study scenario with random probabilities and random failures.

corresponding range. Each simulation is executed as previously described in Section 7.2.2, that is, each node v_i is marked as failed if, given a randomly selected value $r \in [0, 1]$, it holds that $r \leq p(v_i)$.

The obtained results show that the MLMCS-based forensic approach significantly reduces the number of nodes that need to be analysed during the investigation. For example, in the first experiment (Prob [0.0, 0.02]), failed missions involve 6.64% of failed nodes on average while the percentage of analysed nodes approximates 9.76%. In the last experiment (Prob [0.0, 0.4]), 9.88% of the nodes failed, while the investigation required the analysis of 21.24% nodes in average to find out the cause of the mission failure. In general, we have observed that, as opposed to the naïve exploration where the whole set of nodes (100%) should be analysed in the worst-case scenario, the MLMCS-based search strategy is able to significantly speed up the overall forensic process.

(c) Mission simulations over large random graphs. Our last set of experiments involves the analysis of the proposed methodology over larger random graphs. We have experimented with three different sizes, i.e., 1,000, 3,000, and 5,000 nodes, involving a configuration of (60%, 20%, 20%) (see Section 6). For each graph size, we have run 100 simulations, each one considering a new random graph. Failure probabilities are in the range [0.0, 0.02]. The results are illustrated in Figure 15.

We have observed that the number of nodes involved in the set of failed components tends to be quite small compared to the size of the graph. In addition, the analysed nodes during the forensic search only amount to a small fraction of the nodes of the graph, thus providing an important gain in terms of time and exploration resources. Due to the probabilistic nature of the experiments, we have also observed that the number of MLMCS occurrences (i.e., cases where the MLMCS was the cause of the mission failure) does not always relate to the size of the graph. For example, the experiment with 1,000 nodes involved more MLMCS occurrences than the experiment with 3,000 nodes. This is an interesting observation because, even with less MLMCS, the performance of the experiment with 3,000 nodes was similar to that of the other experiments in terms of analysed nodes. This is explained by the fact that the forensic procedure changes the focus of the search exploration (by computing a new MLMCS) as soon as it detects that a node of the current MLMCS (the set that is currently believed to have caused the failure) has not failed. In other words, even if a mission has failed due to a set of components that does not coincide with the initial MLMCS, the incorporation of this new information early in the process allows the forensic strategy to quickly pivot towards the actual components that caused the failure of the mission.

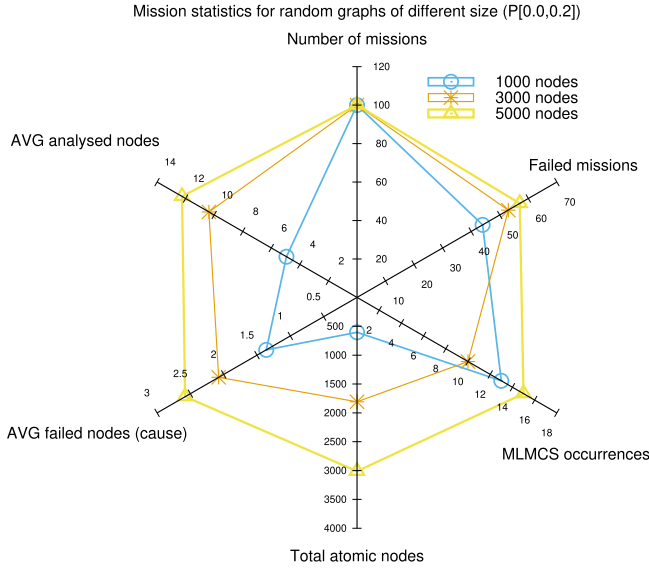


Fig. 15. Mission simulations over large random graphs.

8.2.4 Discussion. The proposed methodology assumes that forensic investigations are precise and can indicate whether a component has failed or not. In practice, this might not always be the case, as inconclusive results may be involved. For example, a burned controller might be too damaged for investigators to produce an accurate and detailed forensic cyber-physical evaluation. In that context, a planned extension in this direction is to integrate a more granular forensic scheme able to cover uncertainty aspects within the overall forensic methodology.

9 CONCLUSION AND FUTURE WORK

In this article, we have presented a novel approach to identifying the **Most Likely Mission-critical Component Set (MLMCS)** within mission-oriented cyber-physical systems. The approach considers AND/OR dependency graphs enriched with failure probabilities that represent internal component failures and performs a log-space transformation to solve the optimisation problem as a Weighted Partial MaxSAT problem. The experiments conducted with our tool LDA4CPS show that the approach is able to scale to large graphs in seconds. The tool is open source and is publicly available at Reference [8]. We have also presented a case study on representative aircraft systems that shows the feasibility of our approach as well as the need for this type of analysis to better understand and enhance the security and resilience of modern cyber-physical systems. Finally, we have described two MLMCS-based security applications focused on system hardening and forensic investigations.

As happens with most risk analysis approaches, the quality of the outcome heavily depends on the input data (model); in our case, failure probabilities. While vendors and historical data may provide useful information to estimate internal failures in physical devices, attack graph-based techniques may also be used to further improve the generation of failure/disruption probabilities that may result from cyber attacks. The integration of these two perspectives, however, poses hard challenges that include, among others, the estimation of conditional and unconditional probabilities within multi-stage cyber-physical attacks, scalability aspects on large and dense graphs, and the involvement of threat intelligence inputs to produce accurate prior probabilities. We plan to investigate these aspects within the avionics sector to refine the proposed approach and also

study its applicability to other classes of cyber-physical systems. In particular, we aim at further exploring industrial settings in critical sectors to overcome the lack of large real-world models that can be publicly accessed and use these outcomes to further extend and validate the application of our approach to real-world scenarios. We also aim at integrating dynamic aspects such as **mean time to failure (MTTF)** [77] to model systems whose failure properties change over time. Further research directions also include the study of tie-solving approaches for MLMCS with the same failure probability as well as the incorporation of additional logic operators such as voting gates [64] and standard fault-tolerant techniques like **triple modular redundancy (TMR)** [45]. Another research line includes the use of hypergraph-based techniques to address dependent failure probabilities and therefore deal with probabilistic relationships between components and external events (e.g., fire, freezing, flood) [14]. Human errors and failures should be also kept in the loop [1]. Finally, we also plan to include uncertainty aspects within our forensic methodology and further enhance the proposed MLMCS-based hardening technique to recommend optimal configurations to maximise reliability and reduce costs.

APPENDIX

A LDA4CPS - Implementation prototype

LDA4CPS (Logical Dependency Analyser for Cyber-Physical Systems) is a Java-based tool, built on top of META4ICS [7], that has been designed to identify the **most likely mission-critical component set (MLMCS)** using AND/OR dependency graphs enriched with independent failure probabilities. LDA4CPS mainly differs from META4ICS in its capacity to operate with stochastic values that represent the probability of failure of system components. LDA4CPS runs in the command line and consumes a JSON file that specifies the input AND/OR graph including failure probabilities and the target of the mission. Once executed, the tool stores the solution in another JSON file that is used to graphically display the AND/OR graph and the MLMCS as shown in Figure 9 for scenario ⑤ in Section 7. Figure 16 illustrates the output of LDA4CPS for this scenario when executed on the command line, indicating the MLMCS and its joint probability of failure.

Probabilities and decimal weights. Currently, many SAT solvers only support integer weights. To address this issue, LDA4CPS performs a transformation on the input decimal weights (i.e., probabilities in negative log space) by right shifting (multiplying by 10) every value until the smallest value is covered with an acceptable level of precision. For example, 0.001 and 0.00007 would become 100 and 7 (right shift five times). Under this approach, precision normally depends on the underlying representation capabilities of the language, which in this case is Java and is quite robust. However, this aspect should be further analysed depending on the expected probability ranges to avoid precision loss. Additional variables introduced by the Tseitin transformation have weight 0. After conversion, LDA4CPS specifies the problem as a Partial Weighted MaxSAT instance by assigning the transformed probability values as an integer penalty score for each decision variable. This methodology has been successfully used in our benchmarks for fault tree analysis over directed acyclic graphs in the MaxSAT Evaluation 2020² [11].

Parallel SAT solving. LDA4CPS runs multiple SAT solvers in parallel and picks up the solution of the solver that finishes first. This is implemented with Java's *ExecutorService*, where each solver is embedded in the form of a callable task that is invoked by the executor. The executor maintains a pool of threads, whose size can be preconfigured, and a thread queue in case there are more solvers than allocated threads. According to our experiments, Java's internal thread

²<https://maxsat-evaluations.github.io/2020>.

```
[mbarrere@Martin-mbp14 lda4cps]$ java -jar lda4cps.jar examples/aircraft-case1.json
== LDA4CPS v0.62.5 ==
== Started at 2020-05-25 12:21:22.938 ==

=> Loading problem specification... done in 260 ms (0 seconds).
-----
Problem source: _s_
Problem target: PitchMACC
-----
=> Performing Tseitin transformation... done in 132 ms (0 seconds).
[+] Solvers: [MaxSAT]
=====
=> BEST solution found by MAX-SAT-SOLVER for:
Source: _s_
Target: PitchMACC
=== Security Metric ===
Joint probability of failure: 0.0016
[+] Failed nodes: none
Total critical nodes: 2
[+] Most likely mission-critical set (MLMCS): (LH,0.04); (RH,0.04);
[*] Metric computation time: 83 ms (0 seconds).
=====
Solution saved in: ./view/sol.json
== LDA4CPS ended at 2020-05-25 12:21:23.444 ==
```

Fig. 16. LDA4CPS execution for scenario S1.

processing mechanism behaves fairly in terms of resource allocation for each thread. This has been also validated by independently analysing complex cases and measuring the time required for each individual solver to find the solution. LDA4CPS can be easily extended to add more pre-configured solvers by defining new callable tasks and setting up an appropriate thread pool size. It is important to note that for small cases, where solvers can find the solution in milliseconds, the use of multiple solvers can produce an overhead that can be noticeable in proportion to the execution time of each solver individually. However, this overhead has been measured in the order of milliseconds within our experiments and therefore has not posed a real problem. The true positive impact of combining multiple solvers can be observed on larger and complex cases where no solver performs better than the others on every instance, as shown with our benchmarks in the competitions MaxSAT Evaluation 2019 [13] and 2020 [11]. When used together, however, the overall performance increases significantly.

Open source. We have released LDA4CPS as an open source project and is publicly available and fully documented at Reference [8]. All of the JSON specification files used for the scenarios presented in this article as well as further graphical examples can also be found at Reference [8].

REFERENCES

- [1] S. Ahrenbach. 2017. Reasoning about safety-critical information flow between pilot and computer. In *NASA Formal Methods*, C. Barrett, M. Davies, and T. Kahsai (Eds.). Springer International Publishing, Cham, 342–356.
- [2] S. B. Akers. 1978. Binary decision diagrams. *IEEE Trans. Comput.* C-27, 6 (June 1978), 509–516.
- [3] C. Alcaraz and S. Wolthusen. 2014. Recovery of structural controllability for control systems. In *Critical Infrastructure Protection VIII*, J. Butts and S. Sheno (Eds.). Springer, Berlin, 47–63.
- [4] D. S. Altner, Ö. Ergun, and N. A. Uhan. 2010. The maximum flow network interdiction problem: Valid inequalities, integrality gaps, and approximability. *Oper. Res. Lett.* 38, 1 (2010), 33–38.
- [5] P. Ammann, D. Wijesekera, and S. Kaushik. 2002. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*. ACM Press, New York, 217–224.
- [6] A. Arulselvan, C. W. Commander, L. Eleftheriadou, and P. M. Pardalos. 2009. Detecting critical nodes in sparse graphs. *Comput. Oper. Res.* 36, 7 (2009), 2193–2200.
- [7] M. Barrère. 2019. META4ICS - Metric Analyser for Industrial Control Systems. Retrieved from <https://github.com/mbarrere/meta4ics>.
- [8] M. Barrère. 2020. LDA4CPS - Logical Dependency Analyser for Cyber-Physical Systems. Retrieved from <https://github.com/mbarrere/lda4cps>.

- [9] M. Barrère, R. Badonnel, and O. Festor. 2014. A SAT-based autonomous strategy for security vulnerability management. In *Proceedings of the IEEE Network Operations and Management Symposium (NOMS'14)*. IEEE, 1–9.
- [10] M. Barrère and C. Hankin. 2020. Fault tree analysis: Identifying maximum probability minimal cut sets with MaxSAT. In *Proceedings of the 50th IEEE-IFIP International Conference on Dependable Systems and Networks (DSN'20), Supplemental Volume (DSN-S)*. IEEE, 53–54. DOI : <https://doi.org/10.1109/DSN-S50200.2020.00029>
- [11] M. Barrère and C. Hankin. 2020. MaxSAT Evaluation 2020 - Benchmark: Identifying Maximum Probability Minimal Cut Sets in Fault Trees. In *Proceedings of the MaxSAT Evaluation 2020 (MSE'20)*. Retrieved from <https://arxiv.org/abs/2007.08255>.
- [12] M. Barrère, C. Hankin, A. Barboni, G. Zizzo, F. Boem, S. Maffei, and T. Parisini. 2018. CPS-MT: A real-time cyber-physical system monitoring tool for security research. In *Proceedings of the 24th IEEE International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA'18)*. IEEE, 240–241. DOI : <https://doi.org/10.1109/RTCSA.2018.00040>
- [13] M. Barrère, C. Hankin, N. Nicolaou, D. Eliades, and T. Parisini. 2019. MaxSAT Evaluation 2019 - Benchmark: Identifying Security-Critical Cyber-Physical Components in Weighted AND/OR Graphs. In *Proceedings of the MaxSAT Evaluation 2019 (MSE'19)*. Retrieved from <https://arxiv.org/abs/1911.00516>.
- [14] M. Barrère, C. Hankin, N. Nicolaou, D. Eliades, and T. Parisini. 2020. Measuring cyber-physical security in industrial control systems via minimum-effort attack strategies. *J. Inf. Secur. Applic.* 52 (June 2020), 1–17. DOI : <https://doi.org/10.1016/j.jisa.2020.102471>
- [15] M. Barrère and E. C. Lupu. 2017. Naggen: A network attack graph generation tool. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'17)*. IEEE, 378–379.
- [16] M. Barrère, R. V. Steiner, R. Mohsen, and E. C. Lupu. 2017. Tracking the bad guys: An efficient forensic methodology to trace multi-step attacks using core attack graphs. In *Proceedings of the 13th IEEE International Conference on Network and Service Management*. IEEE, 1–7.
- [17] C. A. Ericson II. 1999. Fault tree analysis - A history. In *Proceedings of the 17th International System Safety Conference*. System Safety Society, 1–9.
- [18] X. Chen. 2015. Critical nodes identification in complex systems. *Complex Intell. Syst.* 1, 1 (2015), 37–56.
- [19] E. Chung and J. S. Hanks. 2016. Fault tree analyses as a tool for flight control system architecture design. In *Proceedings of the Reliability and Maintainability Symposium (RAMS'16)*. IEEE, 1–6.
- [20] S. A. Cook. 1971. The complexity of theorem-proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*. ACM, 8.
- [21] B. Croft. 2015. Plane PNG Image, License Creative Commons (CC BY-NC 4.0) Retrieved from <https://freepngimg.com>. Retrieved from <https://freepngimg.com/png/996-plane-png-image>.
- [22] N. Das and W. Taylor. 2016. Quantified fault tree techniques for calculating hardware fault metrics according to ISO 26262. In *Proceedings of the IEEE Symposium on Product Compliance Engineering (ISPC'16)*. IEEE, 1–8.
- [23] J. Davies and F. Bacchus. 2011. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Principles and Practice of Constraint Programming – CP 2011*, Jimmy Lee (Ed.). Springer, Berlin, 225–239.
- [24] K. R. Davis, C. M. Davis, S. A. Zonouz, R. B. Bobba, R. Berthier, L. Garcia, and P. W. Sauer. 2015. A cyber-physical modeling and assessment framework for power grid infrastructures. *IEEE Trans. Smart Grid* 6, 5 (2015), 2464–2475.
- [25] Y. Deng, L. Song, Z. Zhou, and P. Liu. 2017. Complexity and vulnerability analysis of critical infrastructures: A methodological approach. *Math. Prob. Eng.* Vol. 2017 (2017), 1–12.
- [26] W. Depamelaere, L. Lemaire, J. Vossaert, and V. Naessens. 2018. CPS security assessment using automatically generated attack trees. In *Proceedings of the 5th International Symposium for ICS & SCADA Cyber Security Research*. BCS Learning & Development, Hamburg, Germany, 1–10.
- [27] Y. Desmedt and Y. Wang. 2002. Maximum flows and critical vertices in AND/OR graphs. In *Computing and Combinatorics*, O. H. Ibarra and L. Zhang (Eds.). Springer, Berlin, 238–248.
- [28] Y. Desmedt and Y. Wang. 2004. Analyzing vulnerabilities of critical infrastructures using flows and critical vertices in AND/OR graphs. *Int. J. Found. Comput. Sci.* 15, 1 (2004), 107–125.
- [29] U. dos Santos Souza, F. Protti, and M. Dantas da Silva. 2013. Revisiting the complexity of and/or graph solution. *J. Comput. Syst. Sci.* 79, 7 (2013), 1156–1163.
- [30] Y. Elderhalli, O. Hasan, W. Ahmad, and S. Tahar. 2018. Formal dynamic fault trees analysis using an integration of theorem proving and model checking. In *NASA Formal Methods*, A. Dutle, C. Muñoz, and A. Narkawicz (Eds.). Springer International Publishing, Cham, 139–156.
- [31] Federal Aviation Administration, U.S. Department of Transportation. 2016. *Airplane Flying Handbook: FAA-8083-3B*.
- [32] Federal Aviation Administration, U.S. Department of Transportation. 2016. *Pilot's Handbook of Aeronautical Knowledge: FAA-H-8083-25B*.
- [33] L. R. Ford and D. R. Fulkerson. 1962. *Flows in Networks*. RAND Corporation. lc62007401
- [34] I. Friedberg, K. McLaughlin, and P. Smith. 2017. A cyber-physical resilience metric for smart grids. In *Proceedings of the IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT'17)*. IEEE, 1–5.

- [35] K. Fritz. 2007. System dependency analysis for complex aircraft systems. In *Proceedings of the Aerospace Technology Conference and Exposition*. SAE International, 1–13.
- [36] Gurobi. 2020. Gurobi Optimizer. Retrieved from <https://www.gurobi.com/>.
- [37] D. F. Haasl, N. H. Roberts, W. E. Vesely, and F. F. Goldberg. 1981. *Fault Tree Handbook*. U.S. Nuclear Regulatory Commission.
- [38] C. Hankin and M. Barrère. 2020. Trustworthy inter-connected cyber-physical systems. In *Critical Information Infrastructures Security (CRITIS2020)*, A. Rashid and P. Popov (Eds.). Springer International Publishing, Cham, 3–13.
- [39] J. Homer and X. Ou. 2009. SAT-solving approaches to context-aware enterprise network security management. *IEEE J. Select. Areas Commun.* 27, 3 (2009), 315–322.
- [40] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah. 2011. Distilling critical attack graph surface iteratively through minimum-cost SAT solving. In *Proceedings of the 27th Computer Security Applications Conference (ACSAC'11)*. ACM, New York, NY, 31–40.
- [41] A. Humayed, J. Lin, F. Li, and B. Luo. 2017. Cyber-physical systems security—A survey. *IEEE Internet Things* 4, 6 (Dec. 2017), 1802–1831.
- [42] A. Ibrahim, S. Kacianka, A. Pretschner, C. Hartsell, and G. Karsai. 2019. Practical causal models for cyber-physical systems. In *NASA Formal Methods*, J. M. Badger and K. Y. Rozier (Eds.). Springer, Cham, 211–227.
- [43] G. Jakimoski and M. Burmester. 2004. Using Faulty Flows in AND/OR Graphs to Model Survivability and Reliability in Distributed Systems. Technical Report. Florida State University. Retrieved from <https://www.cs.fsu.edu/files/reports/TR-060318.pdf>.
- [44] S. Kabir. 2017. An overview of fault tree analysis and its application in model based dependability analysis. *Expert Syst. Applic.* 77 (2017), 114–135. DOI: <https://doi.org/10.1016/j.eswa.2017.01.058>
- [45] F. L. Kastensmidt, L. Sterpone, L. Carro, and M. S. Reorda. 2005. On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. In *Proceedings of the Conference on Design, Automation and Test in Europe*. IEEE, 6.
- [46] K. Kaynar. 2016. A taxonomy for attack graph generation and usage in network security. *J. Inf. Secur. Applic.* 29 (2016), 27–56.
- [47] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer. 2014. DAG-based attack and defense modeling: Don't miss the forest for the attack trees. *Comput. Sci. Rev.* 13–14 (2014), 1–38.
- [48] R. Kumar and M. Stoelinga. 2017. Quantitative security and safety analysis with attack-fault trees. In *Proceedings of the IEEE 18th International Symposium on High Assurance Systems Engineering (HASE'17)*. IEEE, 25–32.
- [49] M. Lalou, M. A. Tahraoui, and H. Kheddouci. 2018. The critical node detection problem in networks: A survey. *Comput. Sci. Rev.* 28 (2018), 92–117.
- [50] T. Li, C. Feng, and C. Hankin. 2018. Improving ICS Cyber Resilience through Optimal Diversification of Network Resources. Retrieved from <https://arxiv.org/abs/1811.00142>.
- [51] R. P. Lippmann and K. W. Ingols. 2005. *An Annotated Review of Past Papers on Attack Graphs*. MIT, Lincoln Lab.
- [52] Y.-Y. Liu and A.-L. Barabási. 2016. Control principles of complex systems. *Rev. Mod. Phys.* 88, 3 (Sep. 2016), 58.
- [53] W. Luo and O. Wei. 2017. WAP: SAT-based computation of minimal cut sets. In *Proceedings of the IEEE 28th International Symposium on Software Reliability Engineering (ISSRE'17)*. IEEE, 146–151. DOI: <https://doi.org/10.1109/ISSRE.2017.13>.
- [54] C. Mattarei, A. Cimatti, M. Gario, S. Tonetta, and K. Y. Rozier. 2015. Comparing different functional allocations in automated air traffic control design. In *Proceedings of the Formal Methods in Computer-aided Design Conference*. FMCAD, 112–119.
- [55] L. Muñoz-González, D. Sgandurra, M. Barrère, and E. C. Lupu. 2019. Exact inference techniques for the analysis of Bayesian attack graphs. *IEEE Trans. Depend. Sec. Comput.* 16, 2 (2019), 231–244.
- [56] D. M. Nicol, W. H. Sanders, and K. S. Trivedi. 2004. Model-based evaluation: From dependability to security. *IEEE Trans. Depend. Sec. Comput.* 1, 1 (2004), 48–65.
- [57] X. Ou, W. F. Boyer, and M. A. McQueen. 2006. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*. ACM Press, New York, NY, 336–345.
- [58] S. Pequito, F. Khorrami, P. Krishnamurthy, and G. J. Pappas. 2019. Analysis and design of actuation–sensing–communication interconnection structures toward secured/resilient LTI closed-loop systems. *IEEE Trans. Contr. Netw. Syst.* 6, 2 (2019), 667–678.
- [59] N. Poolsappasit, R. Dewri, and I. Ray. 2012. Dynamic security risk management using Bayesian attack graphs. *IEEE Trans. Depend. Sec. Comput.* 9, 1 (2012), 61–74.
- [60] M. A. Rahimian and A. G. Aghdam. 2013. Structural controllability of multi-agent networks: Robustness against simultaneous failures. *Automatica* 49, 11 (2013), 3149–3157.
- [61] M. A. Rahman, E. A. Shaer, and R. G. Kavasseri. 2014. Security threat analytics and countermeasure synthesis for power system state estimation. In *Proceedings of the 44th IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 156–167.

- [62] A. M. Ross, D. E. Hastings, J. M. Warmkessel, and N. P. Diller. 2004. Multi-attribute tradespace exploration as front end for effective space system design. *J. Spacecr. Rockets* 41, 1 (2004), 20–28. DOI : <https://doi.org/10.2514/1.9204>.
- [63] M. Roth and P. Liggesmeyer. 2013. Modeling and analysis of safety-critical cyber physical systems using state/event fault trees. In *Proceedings of the Workshop on Dependable Embedded and Cyber-physical Systems*. SAFECOMP, 1–10.
- [64] E. Ruijters and M. Stoelinga. 2015. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Comput. Sci. Rev.* 15–16 (2015), 29–62.
- [65] SAT4J. 2020. SAT4J. Retrieved from <https://www.sat4j.org/>.
- [66] B. Schneier. 1999. Attack Trees—Modeling security threats. Retrieved from <http://www.schneier.com/paper-attacktrees-ddj-ft.html>.
- [67] S. Shen and J. C. Smith. 2012. Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. *Networks* 60, 2 (2012), 103–119.
- [68] T. Steffen, F. Schiller, M. Blum, and R. Dixon. 2009. Increasing the reliability of high redundancy actuators by using elements in series and parallel. In *Computer Safety, Reliability, and Security*, B. Buth, G. Rabe, and T. Seyfarth (Eds.). Springer, Berlin, 270–282.
- [69] R. V. Steiner, M. Barrère, and E. Lupu. 2018. WSNs under attack! How bad is it? Evaluating connectivity impact using centrality measures. In *Proceedings of the IET Living in the Internet of Things Conference, Cybersecurity of the IoT*. IET, 1–8. DOI : <https://doi.org/10.1049/cp.2018.0011>.
- [70] N. O. Tippenhauer, W. G. Temple, A. H. Vu, B. Chen, D. M. Nicol, Z. Kalbarczyk, and W. H. Sanders. 2014. Automatic generation of security argument graphs. In *Proceedings of the IEEE 20th Pacific Rim International Symposium on Dependable Computing*. IEEE, 33–42. DOI : <https://doi.org/10.1109/PRDC.2014.13>
- [71] G. S. Tseitin. 1970. On the complexity of derivation in propositional calculus. In *Studies in Constructive Maths. and Mathematical Logic, Part II*, A. Slisenko (Ed.). Steklov Mathematical Institute, Russia.
- [72] C. Vellaithurai, A. Srivastava, S. Zonouz, and R. Berthier. 2015. CPIndex: Cyber-physical vulnerability assessment for power-grid infrastructures. *IEEE Trans. Smart Grid* 6, 2 (Mar. 2015), 566–575.
- [73] C. von Essen and D. Giannakopoulou. 2014. Analyzing the next generation airborne collision avoidance system. In *Tools and Algorithms for the Construction and Analysis of Systems*, E. Ábrahám and K. Havelund (Eds.). Springer, Berlin, 620–635.
- [74] L. Wang, S. Jajodia, and A. Singhal. 2017. *Network Security Metrics* (1st ed.). Springer International Publishing, 2017.
- [75] F. Xie, T. Lu, X. Guo, J. Liu, Y. Peng, and Y. Gao. 2013. Security analysis on cyber-physical system using attack tree. In *Proceedings of the 9th International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 429–432.
- [76] Y. Xue, Z. Yao, and W. Niu. 2016. The distributed dissimilar redundancy architecture of fly-by-wire flight control system. In *Proceedings of the 12th International Conference on Computational Intelligence and Security (CIS'16)*. IEEE, 287–293.
- [77] Y. Chen, A. M. Gillespie, M. W. Monaghan, M. J. Sampson, and R. F. Hodson. 2012. On component reliability and system reliability for space missions. In *Proceedings of the IEEE International Reliability Physics Symposium*. IEEE, 4B.2.1–4B.2.8.
- [78] Y. Zhao and K. Y. Rozier. 2014. Probabilistic model checking for comparative analysis of automated air traffic control systems. In *Proceedings of the IEEE/ACM International Conference on Computer-aided Design*. IEEE Press, 690–695.

Received May 2020; revised January 2021; accepted February 2021