Cost-damage analysis of attack trees

Milan Lopuhaä-Zwakenberg
University of Twente
m.a.lopuhaa@utwente.nl

Mariëlle Stoelinga
University of Twente & Radboud University
m.i.a.stoelinga@utwente.nl

***Abstract*—Attack trees (ATs) are a widely deployed modelling technique to categorize potential attacks on a system. An attacker of such a system aims at doing as much damage as possible, but might be limited by a cost budget. The maximum possible damage for a given cost budget is an important security metric of a system. In this paper, we find the maximum damage given a cost budget by modelling this problem with ATs, both in deterministic and probabilistic settings. We show that the general problem is NP-complete, and provide heuristics to solve it. For general ATs these are based on integer linear programming. However when the AT is tree-structured, then one can instead use a faster bottom-up approach. We also extend these methods to other problems related to the cost-damage tradeoff, such as the cost-damage Pareto front.**

***Index Terms*—Attack trees, Pareto front, cost-damage analysis, integer linear programming**

I. Introduction

Attack trees. Attack trees (ATs) are a prominent methodology in security analysis. They aid security specialists in identifying, analyzing and prioritizing (cyber)risks. ATs are included in several popular system engineering frameworks, e.g., *UMLsec* [1] and *SysMLsec* [2], and are supported by industrial tools such as Isograph's *AttackTree* [3]. ATs have been used in many scenarios, such as military information infrastructure [4], electronic voting [5], and IoT insider threats [6]. Their popularity is owed to their simplicity, which allows for a range of applications, and their analyzability.

An AT is an hierarchical diagram that describes a system's vulnerabilities to attacks. Despite the name, an AT is a rooted directed acyclic graph (DAG). Its root represents the adversary's goal, while leaves represent basic attack steps (BASs) undertaken by the adversary. Other nodes represent intermediate attack goals and are labeled with an OR-gate or AND-gate, determining how its activation depends on that of its children. An example is given in Fig. 1.

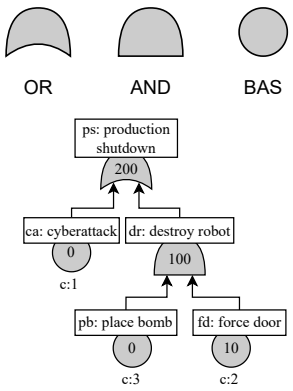


Fig. 1: Attack tree for a factory. Production can be stopped by a cyberattack or by destroying the production robot, for which an attacker forces their way inside and places a bomb. Damage values (in 1000 USD) are inscribed in the nodes, and cost values are below the BASs.

Quantitative analysis. Besides describing possible attacks on a system, ATs can also be used to analyze quantitative information about such attacks. Many *attack metrics* exist, such as the damage, required cost, or required skill of an attack. Such metrics are key performance indicators that formalize a system's security performance.

These metrics do not exist in isolation, and their interplay is important for quantitative security analysis. For instance, one attack may be cheaper than another, but require more time, or a more skilled attacker. Therefore, it is essential to understand the tradeoff between different security metrics. To understand and quantify such tradeoffs, one considers the *Pareto front* of multiple metrics [7], which includes all attacks that are not dominated by another attack in all metrics. For instance, in Fig. 1 the attack $\{\texttt{ca}\}$ does damage 200 for cost 1, which is preferable over $\{\texttt{fd}\}$ which does 10 damage for cost 2.

Cost-damage analysis. In this paper we consider the interplay between two important attack metrics: *attack cost* [8], describing an attacker's budget in, e.g., money or time; and *attack damage* [9], representing the damage done to the system, e.g., in terms of monetary value. The larger the cost budget available to an attacker, the more damaging an attack can be. While damage is the most relevant metric to the system owner, knowing the cost of an attack helps them understand the likelihood of such an attack. This fits within the perspective that likelihood and impact both play an important role in risk analysis [10]. For a comprehensive risk assessment of a system's security, it is therefore paramount to solve the following problems:

Problem statement. Given an attack tree T, solve the following problems:
- DgC) Find the most **D**amaging attack **g**iven a **C**ost budget.
- CgD) Find the **C**heapest attack **g**iven a **D**amage threshold.
- CDPF) Find the **C**ost-**D**amage **P**areto **F**ront.

Existing approaches to calculating the Pareto front of multiple AT metrics [7], [11], [12] cannot be applied to cost-damage problems for two reasons: First, existing methods assume that only BASs are assigned metric values. For damage, this

This research has been partially funded by ERC Consolidator grant 864075 CAESAR and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101008233.

assumption is not realistic, as the internal nodes often represent disabled subsystems, which also have an associated damage value. For instance, in Fig. 1, the attack $\{\texttt{ca}\}$ and $\{\texttt{pb}, \texttt{fd}\}$ both shut down production, but the latter does so by destroying the production robot, leading to greater monetary loss. Second, existing methods only consider *successful attacks*, i.e., attacks that activate the top node of the AT. In the case of cost-damage analysis, however, attacks not reaching the top node can still do quite some damage on intermediate nodes, and should be considered in the analysis. For instance, an attacker can try to rob an ATM by forcing it with explosives. Even if the attacker fails in stealing the money, the explosives still cause significant damage to the ATM owner. Thus existing work cannot solve cost-damage problems in the generality required to model realistic scenarios. For these reasons new approaches and algorithms need to be developed.

Approach. This paper introduces three novel methods to solve the problems stated above. We first consider a deterministic setting, where BASs always succeed. We then consider a probabilistic setting, where BASs may fail with a given probability.

NP-completeness: We first prove two important negative results, showing that even the simplest cost-damage problems do not have 'easy' solutions. Cost-damage problems are similar to binary knapsack problems [13]; we use this to prove that even the simplest type of cost-damage analysis is NP-complete. Unfortunately, this similarity cannot be exploited to apply heuristics for knapsack problems or their many extensions [14]–[16] to cost-damage problems: All extensions assume properties of the damage function (i.e., the function assigning a damage value to each attack) that are not met in our setting. In fact, we prove that the damage function can be any nondecreasing function. This highlights the need for the completely new methods for cost-damage analysis in ATs.

As common, our algorithms distinguish between tree- and DAG-shaped attack trees. Further, we consider deterministic versus probabilistic failure behaviour in the leaves.

Bottom-up algorithm for treelike ATs: Existing approaches to the Pareto front of two metrics work bottom-up, discarding non-optimal attacks at every node [7]. This does not work for damage, as intermediate nodes also carry damage values. Hence attacks that are non-optimal at a certain node may do more damage at a higher node, becoming optimal there.

To solve problem CDPF above, we describe a new bottom-up method for finding the Pareto front in both the deterministic and probabilistic setting. The key insight is to perform a bottom-up Pareto analysis in an *extended cost-damage domain*, by adding a dimension for the current top node's activation (or activation probability in the probabilistic setting); this dimension signifies an attack's 'potential' to do more damage at higher nodes. As shown in our experiments, these bottom-up methods drastically reduce computation time from multiple hours to less than 0.1 second.

For the single-objective problems DgC and CgD we cannot do a 'simpler' bottom-up approach in which only the optimal attack is propagated, as one needs the overview of the full AT to decide which attack is optimal. Instead, we still need to propagate (part of) the Pareto front, and we gain our solution for DgC and CgD from minor adaptations to the CDPF approach.

	Tree	DAG
Deterministic	bottom-up (Theorem 4)	BILP (Theorem 6)
Probabilitistic	bottom-up (Theorem 9)	*open problem*

TABLE I: Overview of this paper's algorithmic contributions.

Integer linear programming for DAG-like ATs: It is well-known [12], [17] that bottom-up algorithms do not work for DAG-like ATs: since nodes may have multiple parents, their cost/damage being counted twice. We introduce a novel method for the deterministic setting by translating cost-damage problems into the *bi-objective integer linear programming* (BILP) framework [18]; we can then apply existing BILP solvers to solve them [19]. This translation is nontrivial, as damage is a nonlinear function of the adversary's attack, as we will show in Section V. The key insights behind our algorithm are that (1) damage is linear in terms of the *structure function* that describes which AT nodes are reached by an attack and (2) the constraints defining the structure function can be phrased as linear constraints.

We use existing biobjective methods and solvers to solve CDPF [20], and single-objective solvers to solve DgC and CgD [21]. This does not extend to the probabilistic setting, where equations become nonlinear; we leave the analysis of probabilistic DAG-like ATs as an open problem.

Finally, in experiments we show our methods can be used for risk analysis by applying them to two systems: a wireless sensor device tracking wildlife in a giant panda reservation, and a data server in a network behind a firewall. The ATs of these systems are taken from the literature [22], [23]. We use the cost-damage Pareto front to assess the vulnerabilities of these systems. Furthermore, we also measure the computing time in the case studies and on 500 random ATs: both bottom-up and BILP methods vastly outperform the existing enumerative approach. This shows that our methods present an enormous speedup compared to the status quo.

Contributions. Summarized, our contributions are:

1) A formal definition of cost-damage problems in ATs;
2) A proof that these problems are NP-complete (Sec. V);
3) A proof that cost-damage problems cannot be reduced to common extensions of the binary knapsack problem; (Sec. V);
4) A bottom-up method to solve the deterministic and probabilistic cost-damage problems for treelike attack trees (Sec. VI & IX);
5) An integer linear programming-based method to solve the deterministic cost-damage problems for DAG-like attack trees (Sec. VII).
6) An experimental evaluation of the above methods on two realistic cases from the literature (Sec. X).

The Matlab code for the experiments, as well as a version of this paper with an appendix containing all proofs, can be found at [24]. It will also be put on ArXiv.

II. Related work

In the literature, there are multiple approaches to decorating an AT with cost and damage values. Existing work concerning damage (also called *impact*) on ATs can be divided into three categories: works in which only BASs have a damage attribute [9], [11], [25], [26], works in which only the root node has a damage attribute [27], and works in which every node can have a damage attribute [28]. In the same manner, in some works intermediate nodes are allowed to have an associated cost [11], [29], while in others only BASs have costs [11], [12], [30]. In this paper, every node has a damage attribute, while only BASs have a cost attribute. We choose this because it is the simplest model for the most expressivity; as we will show in Section IV, cost values on internal nodes can be modeled by adding dummy BASs, but damage values cannot.

Most of the work listed above only considers one metric at a time. For instance, in [25] binary decision diagrams (BDDs) are used to calculate both the minimal cost of a succesful attack and the maximal damage, but the tradeoff between the two metrics is not investigated. Other methods for calculating single metrics include bottom-up methods for treelike ATs [12] and priced-timed automata [29]. Of the works that consider cost-damage tradeoffs, some focuse on modeling rather than algorithms [9], [28]. One approach to the Pareto front is via priced-timed automata [11]; however, we cannot directly apply this to our setting as in that work only BASs have a damage attribute. In [27], cost and damage are used to define a single attack parameter *outcome*, which is optimized heuristically.

Other works on ATs consider the Pareto front between two generic metrics. A bottom-up method for calculating Pareto fronts for treelike ATs, and under some additional assumption for DAG-like ATs, is given in [7]. Furthermore, a BDD-based approach for DAG-like ATs is developed in [12]. However, damage does not satisfy the conditions for either of these two approaches, and these cannot be used for our CgD, DgC and CDPF problems. Overall, we can conclude that none of the existing literature is able to solve cost-damage problem in the general model discussed in this paper.

Another approach to multi-objective optimization is to approximate the Pareto front, for example using genetic algorithms [31], [32]. This has also been applied to ATs with cost [26]. While such an approach would be interesting for cost-damage ATs, in this paper we instead focus on provably optimal solutions, corresponding to provable security guarantees.

III. Preliminaries

Let $\mathbb{B}$ be the set $\{0,1\}$, with logical operators $\wedge, \vee$.

Definition 1. *An* attack tree *is a rooted directed acyclic graph $T = (N, E)$ where each node $v \in N$ has a type $\gamma(v) \in \{\mathrm{BAS}, \mathrm{OR}, \mathrm{AND}\}$, such that $\gamma(v) = \mathrm{BAS}$ if and only if v is a leaf.*

Contrary to terminology an AT is not necessarily a tree. When the DAG T is actually a tree, it is called *treelike*; the general case is referred to as *DAG-like*. The root of T is denoted R_T. For a node v we denote its set of children by $\mathrm{Ch}(v) = \{w \mid (v,w) \in E\}$; we also say that v is an *ancestor* of w, and w a *descendant* of v, if there is a path $v \to w$ in T. When $\mathrm{Ch}(v) = \{v_1, \dots, v_n\}$, we write $v = \mathrm{OR}(v_1, \dots, v_n)$ or $v = \mathrm{AND}(v_1, \dots, v_n)$ depending on $\gamma(v)$. The set of BASs a.k.a. leaves is denoted by B. For instance, in the AT T from Fig. 1 one has $B = \{\mathtt{ca}, \mathtt{pb}, \mathtt{fd}\}$, $\mathtt{dr} = \mathrm{AND}(\mathtt{pb}, \mathtt{fd})$, and $\mathrm{R}_T = \mathtt{ps} = \mathrm{OR}(\mathtt{ca}, \mathtt{dr})$. Note that T is treelike.

Notation	Explanation	page
$\mathbb{B}$	$\{0,1\}$	3
$T = (N, E)$	Attack Tree	3
B	BASs of T	3
$\gamma(v)$	Type of node v	3
$\mathrm{Ch}(v)$	Children of node v	3
$(\mathcal{A}, \preceq)$	Poset of attacks	3
$\mathrm{S}(\mathbf{x}, v)$	Structure function of T	3
$\mathrm{c}(v)$	Cost of BAS v	4
$\mathrm{d}(v)$	Damage of node v	4
$\hat{\mathrm{c}}(\mathbf{x})$	Cost of attack $\mathbf{x}$	4
$\hat{\mathrm{d}}(\mathbf{x})$	Damage of attack $\mathbf{x}$	4
$\underline{\min} X$	Set of minima of X	4
$(\mathbb{R}^2_{\geq 0}, \sqsubseteq)$	Poset of attribute pairs	4
$\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}$	Attribution map	4
$\mathrm{PF}(T)$	Pareto-front of T	4
CDPF	Cost-damage Pareto front	4
DgC	Maximal damage given cost	4
CgD	Minimal cost given damage	4
$(\mathtt{DTrip}, \sqsubseteq)$	Deterministic attribute triples	6
$\underline{\min}_U$	Cost-restricted $\underline{\min}$	6,9
$\mathcal{C}^{\mathrm{D}}_U(v)$	Incomplete deterministic PF at v	6
$\mathrm{p}(v)$	Probability of BAS v	8
$Y_{\mathbf{x}}$	Actualized attack	8
$\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})$	Expected damage of attack $\mathbf{x}$	8
CEDPF	Cost-expected damage Pareto front	8
EDgC	expected damage given cost	8
CgED	cost given expected damage	8
$\mathrm{PS}(\mathbf{x}, v)$	Probabilistic structure function	8
$(\mathtt{PTrip}, \sqsubseteq)$	Probabilistic attribute triples	9
$\mathcal{C}^{\mathrm{P}}_U(v)$	Incomplete probabilistic PF at v	9

TABLE II: Notation used in this paper.

An attacker performs an attack by activating a chosen set of BASs, represented by a *status vector* $\mathbf{x} \in \mathbb{B}^B$; the status x_v of a BAS v equals 1 if v is activated, and 0 if it is not. Such a status vector can also be regarded as a subset of B. Transposing the partial order $\subseteq$ to status vectors yields a partial order $\preceq$.

Definition 2. *An* attack *on T is a vector $\mathbf{x} \in \mathbb{B}^B$; we let $\mathcal{A} = \mathbb{B}^B$ be the set of all attacks. This has a partial order $\preceq$ given by $\mathbf{x} \preceq \mathbf{y}$ iff $x_v \leq y_v$ for all $v \in B$.*

An attack propagates upwards from the BASs. A node is reached by an attack depending on its type OR or AND, and whether any/all of its children are reached by the attack. This idea is formalized by the structure function S. Given an attack vector $\mathbf{x}$, and a node v, $\mathrm{S}(\mathbf{x}, v)$ indicates whether v is reached by $\mathbf{x}$, i.e., if $\mathrm{S}(\mathbf{x}, v) = 1$.

Definition 3. *The* structure function $\mathrm{S} \colon \mathcal{A} \times N \to \mathbb{B}$ *of T is defined recursively:*

$$\mathrm{S}(\mathbf{x}, v) = \begin{cases} x_v & \text{if } \gamma(v) = \mathrm{BAS}, \\ \bigvee_{v' \in \mathrm{Ch}(v)} \mathrm{S}(\mathbf{x}, v') & \text{if } \gamma(v) = \mathrm{OR}, \\ \bigwedge_{v' \in \mathrm{Ch}(v)} \mathrm{S}(\mathbf{x}, v') & \text{if } \gamma(v) = \mathrm{AND}. \end{cases}$$

IV. Deterministic cost-damage problems for ATs

In this section we formulate this paper's problem; solutions are presented in Sections VI and VII. This section deals with a deterministic setting, where a BAS's success is guaranteed; its probabilistic equivalent is presented in Section VIII.

The attacker's goal is to disrupt the system as much as possible, which is measured by a *damage* value representing financial cost, downtime, etc. Each node v has a damage value $\mathrm{d}(v)$, and an attack's total damage $\hat{\mathrm{d}}(\mathbf{x})$ is the sum of the damage value of all nodes reached by $\mathbf{x}$. At the same time, an attacker may have only limited resources. Each BAS v has a *cost* value $\mathrm{c}(v)$ representing e.g. the money, time or resources the attacker has to spend to activate it. The total cost $\hat{\mathrm{c}}(\mathbf{x})$ of an attack is the sum of the costs of the activated BASs.

Definition 4. *A* cd-AT *is a triple* $(T, \mathrm{c}, \mathrm{d})$ *of an AT* T *and maps* $\mathrm{c}\colon B \to \mathbb{R}_{\geq 0}$ *and* $\mathrm{d}\colon N \to \mathbb{R}_{\geq 0}$. *Define the total cost and damage functions* $\hat{\mathrm{c}}, \hat{\mathrm{d}}\colon \mathcal{A} \to \mathbb{R}_{\geq 0}$ *by*

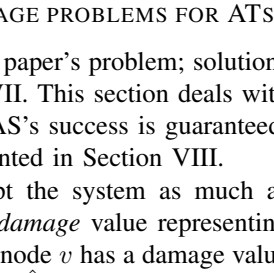
$$\hat{\mathrm{c}}(\mathbf{x}) = \sum_{v \in B} x_v \, \mathrm{c}(v), \qquad \hat{\mathrm{d}}(\mathbf{x}) = \sum_{v \in N} \mathrm{S}(\mathbf{x}, v) \, \mathrm{d}(v).$$

As opposed to other works in quantitative analysis on ATs [7], [12], we do not only consider so-called *successful* attacks, i.e., $\mathbf{x}$ for which $\mathrm{S}(\mathbf{x}, \mathrm{R}_T) = 1$. The reason is that in our model damage can be done at any level, not just at the top node. It is therefore important to know the damaging capabilities of an attacker, even when that attacker's limited resources mean that they cannot damage the top node. Furthermore, an attacker may try different avenues towards success, and while a given path may be discarded without reaching the top node, side effects may remain. We therefore assign damage values not only to the top node, but also to internal nodes.

Example 1. *Consider the AT* T *from Fig. 1, repeated below, and its cost and damage functions. Then the functions* $\hat{\mathrm{c}}$ *and* $\hat{\mathrm{d}}$ *are calculated as in the following table.*

$x_{\mathtt{ca}}$	0	0	0	0	1	1	1	1
$x_{\mathtt{pb}}$	0	0	1	1	0	0	1	1
$x_{\mathtt{fd}}$	0	1	0	1	0	1	0	1
$\hat{\mathrm{c}}(\mathbf{x})$	0	2	3	5	1	3	4	6
$\hat{\mathrm{d}}(\mathbf{x})$	0	10	0	310	200	210	200	310

Some works also assign cost values to internal nodes [11], [29], the interpretation being that an internal node is only activated if enough of its children are activated and its cost is paid. However, this can be simulated by adding a dummy BAS which holds the associated cost, as in Fig. 2. However, the same cannot be done for damage: moving the damage to the dummy BAS leads to a situation where *only* the dummy needs to be activated to do the damage. For full expressivity we thus allow internal nodes to have damage values, but not cost values.



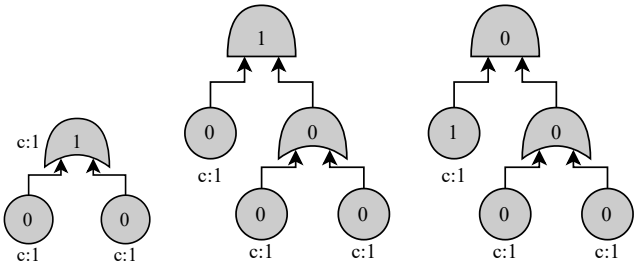


Fig. 2: An example showing that damage values on internal nodes are necessary, but cost values on internal nodes are not. The cost value on the internal node in the AT on the left is replaced by a dummy BAS in the middle AT, which is equivalent: both ATs require cost 2 to perform 1 damage. In the right AT, the damage is also moved to the dummy BAS, but the result is not equivalent: 1 cost already yields 1 damage.

A. Cost damage problems

In ATs, there is a tradeoff between resource utilization and damage: the higher the cost budget an attacker has at their disposal, the more damage they may cause. This tradeoff can be analyzed via the *Pareto front*: the cost and damage values of all attacks that are not dominated by other attacks, where $\mathbf{x}$ dominates $\mathbf{y}$ if $\mathbf{x}$ is cheaper than $\mathbf{y}$ while doing more damage. An attack $\mathbf{x}$ in the Pareto front is called *Pareto optimal*, and it is the most damaging attack if the attacker cannot exceed cost $\hat{\mathrm{c}}(\mathbf{x})$. Thus the Pareto front gives a full overview of the system's vulnerability to any attacker.

For a general poset $(X, \preceq)$, we define its set of minimal elements as

$$\underline{\min}_{\preceq} X = \{x \in X \mid \forall x' \in X. x' \not\prec x\}.$$

We drop the subscript $\preceq$ if it is clear from the context. We consider the domain of *attribute pairs*, i.e., the set $\mathbb{R}^2_{\geq 0}$ with a partial order $\sqsubseteq$ given by $(a, a') \sqsubseteq (b, b')$ if and only if $a \leq b$ and $a' \geq b'$. For a cd-AT $(T, \mathrm{c}, \mathrm{d})$, we define the evaluation map $\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}\colon \mathcal{A} \to \mathbb{R}^2_{\geq 0}$ by $\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}(\mathbf{x}) = \binom{\hat{\mathrm{c}}(x)}{\hat{\mathrm{d}}(x)}$ (we represent elements of $\mathbb{R}^2_{\geq 0}$ as column vectors). Note that $\mathbf{x}$ dominates $\mathbf{y}$ if and only if $\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}(\mathbf{x}) \sqsubset \binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}(\mathbf{y})$.

The aim of this paper is to find the cost-damage Pareto front, as well as two related single-objective problems. Mathematically, these are formulated as follows:

Problems. Given a cd-AT $(T, \mathrm{c}, \mathrm{d})$, solve the following problems:

CDPF Cost-damage Pareto front: find $\underline{\min}_{\sqsubseteq} \binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}(\mathcal{A}) \subseteq \mathbb{R}^2_{\geq 0}$.

DgC Maximal damage given cost constraint: Given $U \in \mathbb{R}_{\geq 0}$, find $d_{\mathrm{opt}} = \max_{\mathbf{x}\colon \hat{\mathrm{c}}(\mathbf{x}) \leq U} \hat{\mathrm{d}}(\mathbf{x})$.

CgD Minimal cost given damage constraint: $L \in \mathbb{R}_{\geq 0}$, find $c_{\mathrm{opt}} = \min_{\mathbf{x}\colon \hat{\mathrm{d}}(\mathbf{x}) \geq L} \hat{\mathrm{c}}(\mathbf{x})$.

From CDPF one can solve DgC and CgD via

$$d_{\mathrm{opt}} = \max\{d \in \mathbb{R}_{\geq 0} \mid \exists c \in [0, U]. \binom{c}{d} \in \mathrm{PF}(T)\}, \quad (1)$$

$$c_{\mathrm{opt}} = \min\{c \in \mathbb{R}_{\geq 0} \mid \exists d \in \mathbb{R}_{\geq L}. \binom{c}{d} \in \mathrm{PF}(T)\}. \quad (2)$$

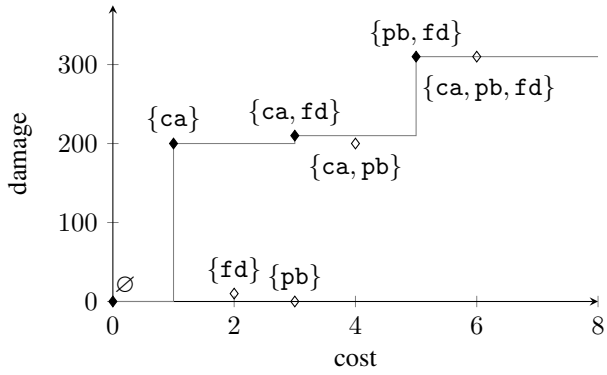


Fig. 3: CDPF for Examples 1 and 2. Filled nodes are Pareto-optimal attacks.

These problems are relevant in security analysis: DgC can be used to determine the damaging capabilities of different attacker profiles [11], [26]. CDPF can be used to give an overview over all attacker profiles. For a security operations center monitoring a network, a cost-damage analysis (with cost measured in time) provides insight in whether the response time is sufficient to stop damaging attacks.

Example 2. *In Example 1,* $\binom{\hat{c}}{\hat{d}}(\mathcal{A})$ *is given by the lower two rows of the table. A number of these attacks are not Pareto optimal: we have* $\binom{1}{200} \sqsubset \binom{2}{10}, \binom{3}{0}, \binom{4}{200}$*, and furthermore* $\binom{5}{310} \sqsubset \binom{6}{310}$*. It follows that (see Fig. 3):*

$$\mathrm{PF}(T) = \left\{\binom{0}{0}, \binom{1}{200}, \binom{3}{210}, \binom{5}{310}\right\}. \tag{3}$$

From this we find, for instance that the solution to DgC for $U = 2$ *is given by* $d_{\mathrm{opt}} = 200$*.*

In what follows, we present novel methods to solve CDPF, DgC and CgD. As in many problems related to calculating AT metrics, an important factor in the complexity of solutions is whether the AT is treelike or not [12]. We introduce a bottom-up method for treelike ATs in Section VI, and a method based on integer linear programming for DAG-like ATs in Section VII.

V. Relation to knapsack problems and NP-completeness

In this section, we prove two important negative results, based on the similarity of cost-damage problems to binary knapsack problems. First, we show that even the simplest cost-damage problem is NP-complete. Second, we show that cost-damage problems are considerably more general than (extended) knapsack problems, which means that existing heuristics for knapsack problems cannot be applied to our situation. Both results emphasize the importance of finding new heuristics for cost-damage problems.

DgC is a generalisation of the binary knapsack problem [13], which is

$$\text{minimize}_{\mathbf{x}\in\mathbb{B}^n} \quad f(\mathbf{x}) \quad \text{subject to} \quad g(\mathbf{x}) \leq b$$

where $b \in \mathbb{R}$ and $n \in \mathbb{N}$ are constants and the objective and constraint functions f and g are *linear*, i.e., $f(\mathbf{x}) = \sum_{i=1}^{n} f_i \mathbf{x}_i$ for some constants $f_i \in \mathbb{R}$. In DgC, $n = |B|$, $b = U$, and the objective and constraint functions are $-\hat{\mathrm{d}}$ and $\hat{\mathrm{c}}$. Although $\hat{\mathrm{c}}$ is linear, $-\hat{\mathrm{d}}$ is not; for instance, in the AT $\mathrm{AND}(a, b)$, one has $\hat{\mathrm{d}}(\mathbf{x}) = \mathrm{d}(a)x_a + \mathrm{d}(b)x_b + \mathrm{d}(\mathrm{R}_T)(x_a \wedge x_b)$. To show NP-completeness, consider the *decision problem* associated to CDPF, DgC and CgD:

Problem (Cost-damage decision problem (CDDP))**.** *Given a cd-AT* $(T, \mathrm{c}, \mathrm{d})$*, a cost upper bound* U *and a damage lower bound* L*, decide whether there exists an attack* $\mathbf{x} \in \mathcal{A}$ *such that* $\hat{\mathrm{c}}(\mathbf{x}) \leq U$ *and* $\hat{\mathrm{d}}(\mathbf{x}) \geq L$*.*

CDDP can be reduced to CDPF, DgC or CgD. Theorem 1 shows that the knapsack decision problem can be reduced to the CDDP (in fact, a *treelike* AT with n BASs and a root suffices). Since the knapsack decision problem is known to be NP-complete [33] and it is straightforard to show that CDDP is in NP, we find the following result:

Theorem 1. *CDDP is NP-complete, even when restricted to treelike ATs.*

The binary knapsack decision problem (and by extension CDDP) is known to be NP-complete [34]. It should come as no surprise that we do not give polynomial-time methods to solve CDPF, DgC, and CgD, but instead introduce heuristic methods. These methods discard infeasible solutions throughout the computation instead of at the end, making them faster than the naive approach.

In the literature, many extensions of the binary knapsack problem have been considered that allow less restrictive types of objectives functions, such as quadratic [14], cubic [15] and submodular [16] objective functions. However, the following theorem shows that objective functions $\hat{\mathrm{d}}$ arising from cd-ATs form the even larger class of *nondecreasing* functions (i.e., $\mathbf{x} \preceq \mathbf{y}$ implies $f(\mathbf{x}) \leq f(\mathbf{y})$, see Definition 2).

Theorem 2. *Let* X *be a finite set, and let* $f\colon \mathbb{B}^X \to \mathbb{R}_{\geq 0}$ *be any nondecreasing function. Then there is a cd-AT* $(T, \mathrm{c}, \mathrm{d})$ *with* $B = X$ *and* $\hat{\mathrm{d}} = f$*.*

It follows that we cannot use existing binary knapsack approaches to solve DgC, since these approaches [14]–[16] put some assumptions on $\hat{\mathrm{d}}$. Instead, we develop new techniques, based on bottom-up methods and integer linear programming. These techniques exploit the structure of the cd-AT from which the objective $\hat{\mathrm{d}}$ originates.

VI. Treelike ATs, deterministic setting

For treelike ATs in the deterministic setting we focus on CDPF. DgC and CgD then follow from (1) and (2) respectively. These single-objective problems cannot be computed easier because, as we will demonstrate below, we need to propagate (part of) the Pareto front bottom-up, rather than a single damage/cost value, to solve these problems.

A. CDPF

A naive way to solve CDPF (and with it DgC and CgD) is by calculating $\hat{\mathrm{c}}(\mathbf{x})$ and $\hat{\mathrm{d}}(\mathbf{x})$ for each $\mathbf{x} \in \mathcal{A}$. Since

$|\mathcal{A}| = 2^{|B|}$, this is impractical for large ATs, and new heuristics are needed. We solve CDPF via a bottom-up approach in which only a small set of attacks is handled at each node, and infeasilibity is determined at each node rather than at the end. The key insight to make this work is that at intermediate nodes, we perform Pareto analysis in an extended domain DTrip, and we only project to $\mathbb{R}^2_{\geq 0}$ at the root.

For a node v, we let T_v be the sub-AT of T with root v, and we let B_v be its set of BASs. At the node v, we are interested in the cost and damage of attacks on T_v, which are elements of $\mathbb{B}^{B_v}$. Suppose that $\mathrm{Ch}(v) = \{v_1, v_2\}$. Since T is treelike, one has $B_{v_1} \cap B_{v_2} = \varnothing$. So $\mathbb{B}^{B_v} = \mathbb{B}^{B_{v_1}} \times \mathbb{B}^{B_{v_2}}$, and an attack $\mathbf{x}$ on T_v can be written $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ for attacks $\mathbf{x}_1$ on T_{v_1} and $\mathbf{x}_2$ on T_{v_2}. With regards to cost and damage, we find

$$\hat{\mathrm{c}}(\mathbf{x}) = \hat{\mathrm{c}}(\mathbf{x}_1) + \hat{\mathrm{c}}(\mathbf{x}_2), \tag{4}$$

$$\hat{\mathrm{d}}(\mathbf{x}) = \hat{\mathrm{d}}(\mathbf{x}_1) + \hat{\mathrm{d}}(\mathbf{x}_2) + \mathrm{S}(\mathbf{x}, v)\,\mathrm{d}(v) \tag{5}$$

where we recall that $\mathrm{S}(\mathbf{x}, v)$ is defined as

$$\mathrm{S}(\mathbf{x}, v) = \begin{cases} x_v, & \text{if } \gamma(v) = \mathrm{BAS}, \\ \mathrm{S}(\mathbf{x}_1, v_1) \vee \mathrm{S}(\mathbf{x}_2, v_2), & \text{if } \gamma(v) = \mathrm{OR}, \\ \mathrm{S}(\mathbf{x}_1, v_1) \wedge \mathrm{S}(\mathbf{x}_2, v_2), & \text{if } \gamma(v) = \mathrm{AND}. \end{cases}$$

Thus, in order to correctly calculate the cost and damage of attacks as we combine them, we need to store each attack $\mathbf{x}$ as an *attribute triple* in the *deterministic attribute triple domain*:

$$\begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}) \\ \hat{\mathrm{d}}(\mathbf{x}) \\ \mathrm{S}(\mathbf{x},v) \end{pmatrix} \in \texttt{DTrip} := \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathbb{B}.$$

Example 3. *Consider the AT of Example 1. Each BAS has only two possible attacks (activating that BAS or not) so for* pb *we have* $\left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}3\\0\\1\end{pmatrix}\right\} \subset \texttt{DTrip}$*, and* $\left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}2\\10\\1\end{pmatrix}\right\} \subset \texttt{DTrip}$ *for* fd*. Combining these, we have four possible attacks on the* AND*-gate* dr*, which is the set*

$$\left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}3\\0\\0\end{pmatrix}, \begin{pmatrix}2\\10\\0\end{pmatrix}, \begin{pmatrix}5\\110\\1\end{pmatrix}\right\} \subset \texttt{DTrip}.$$

After finding the values of all attacks on v by combining those on v_1 and on v_2, we discard the infeasible ones. Infeasibility is based on two conditions:

1) In DgC, if $\hat{\mathrm{c}}(\mathbf{x}) > U$, then $\mathbf{x}$ is infeasible.
2) Other than that, feasibility is determined by Pareto optimality on the poset $(\texttt{DTrip}, \sqsubseteq)$, where $\begin{pmatrix}c\\d\\b\end{pmatrix} \sqsubseteq \begin{pmatrix}c'\\d'\\b'\end{pmatrix}$ if and only if $c \leq c'$, $d \geq d'$ and $b \geq b'$. The first two inequalities are to be expected from cost-damage optimality. The third inequality is introduced for the following reason: if $\mathbf{x}$ and $\mathbf{x}'$ are two attacks on v corresponding to $(c, d, 0)^\intercal$ and $(c', d', 1)^\intercal$, respectively, then potentially $\mathbf{x}'$ can reach nodes higher up in T, and thereby eventually do more damage than $\mathbf{x}$. However, whether this happens or not cannot be detected at the level of v, and therefore we need to keep both triples.

Example 4. *We continue Example 3. At* dr*, we have* $\begin{pmatrix}0\\0\\0\end{pmatrix} \sqsubset \begin{pmatrix}3\\0\\0\end{pmatrix}$*, so the latter is infeasible and discarded, leaving us with the Pareto front*

$$\left\{\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}2\\10\\0\end{pmatrix}, \begin{pmatrix}5\\110\\1\end{pmatrix}\right\} \subset \texttt{DTrip}.$$

This example shows why we need the third dimension: if not, we would have discarded the attack $\binom{3}{0}$ *at* pb *for being infeasible:* $\binom{0}{0}$ *does the same damage at lower cost. However, had we done so at* pb*, we would have concluded that it is always optimal not to activate* pb*, thereby missing out on the attack* $\binom{5}{110}$ *at* dr*. By also storing the top node's activation, we ensure that activating* pb *is still considered feasible.*

This approach can be formally defined as follows. Let $U \in [0, \infty]$. For each $v \in N$, we define a Pareto front $\mathcal{C}^{\mathrm{D}}_U(v) \subseteq \texttt{DTrip}$ (for *D*eterministic) of feasible attacks on v. To do this, we define a map $\underline{\min}_U \colon \mathcal{P}(\texttt{DTrip}) \to \mathcal{P}(\texttt{DTrip})$ given by

$$\underline{\min}_U(X) = \underline{\min}_{\sqsubseteq} \left\{ \begin{pmatrix}c\\d\\b\end{pmatrix} \in X : c \leq U \right\}$$

which returns the Pareto-optimal elements (w.r.t. the partial order $\sqsubseteq$ of DTrip) of a set X that do not exceed the cost constraint. From now we assume that T is *binary*, i.e., $|\,\mathrm{Ch}(v)| \in \{0, 2\}$ for all v. Since every AT is equivalent to a binary one this assumption is purely to simplify notation. We then recursively define the Pareto front $\mathcal{C}^{\mathrm{D}}_U(v)$ of attribute triples, by combining elements of $\mathcal{C}^{\mathrm{D}}_U(v_1)$ and $\mathcal{C}^{\mathrm{D}}_U(v_2)$ via (4) and (5) and then discarding the nonfeasible triples:

$$\mathcal{C}^{\mathrm{D}}_U(v) = \begin{cases} \left\{ \begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}\mathrm{c}(v)\\\mathrm{d}(v)\\1\end{pmatrix} \right\}, & \text{if } \gamma(v) = \mathrm{BAS} \text{ and } \mathrm{c}(v) \leq U, \\ \left\{ \begin{pmatrix}0\\0\\0\end{pmatrix} \right\}, & \text{if } \gamma(v) = \mathrm{BAS} \text{ and } \mathrm{c}(v) > U, \end{cases}$$

$$\mathcal{C}^{\mathrm{D}}_U(\mathrm{AND}(v_1, v_2)) = \underline{\min}_U \left\{ \begin{pmatrix} c_1 + c_2 \\ d_1 + d_2 + (b_1 \wedge b_2) \cdot \mathrm{d}(v) \\ b_1 \wedge b_2 \end{pmatrix} \in \texttt{DTrip} \,\middle|\, \begin{pmatrix}c_i\\d_i\\b_i\end{pmatrix} \in \mathcal{C}^{\mathrm{D}}_U(v_i) \right\},$$

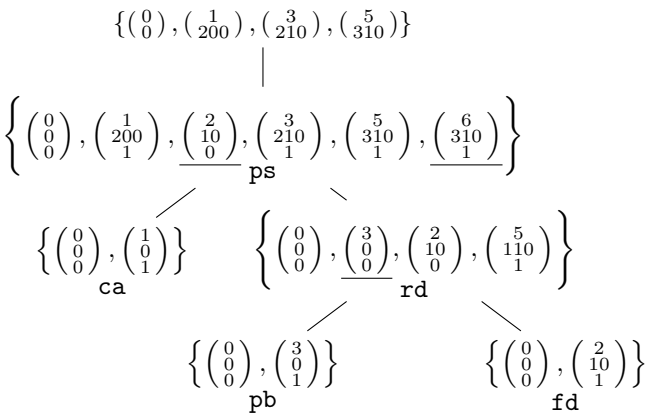
$$\mathcal{C}^{\mathrm{D}}_U(\mathrm{OR}(v_1, v_2)) = \underline{\min}_U \left\{ \begin{pmatrix} c_1 + c_2 \\ d_1 + d_2 + (b_1 \vee b_2) \cdot \mathrm{d}(v) \\ b_1 \vee b_2 \end{pmatrix} \in \texttt{DTrip} \,\middle|\, \begin{pmatrix}c_i\\d_i\\b_i\end{pmatrix} \in \mathcal{C}^{\mathrm{D}}_U(v_i) \right\}.$$

These theorems show the validity of this approach.

Theorem 3. *The solution to DgC is given by* $\max \left\{ d \in \mathbb{R}_{\geq 0} \,\middle|\, \begin{pmatrix}c\\d\\b\end{pmatrix} \in \mathcal{C}^{\mathrm{D}}_U(\mathrm{R}_T) \right\}$.

Theorem 4. *The solution to CDPF is given by* $\min \pi(\mathcal{C}^{\mathrm{D}}_\infty(\mathrm{R}_T))$*, where* $\pi \colon \texttt{DTrip} \to \mathbb{R}^2_{\geq 0}$ *is the projection map onto the first two components.*

Example 5. *We continue Examples 3 and 4, for* $U = \infty$*, in which we calculated* $\mathcal{C}^{\mathrm{D}}_\infty(\texttt{dr})$*. Below shows the calculation for* $\mathcal{C}^{\mathrm{D}}_\infty(v)$ *for every node; underlined vectors are infeasible and are not part of* $\mathcal{C}^{\mathrm{D}}_\infty(v)$*. The top set is the solution to CDPF.*



B. DgC and CgD

For DgC we still have to compute a Pareto front at every node v, instead of taking the most damaging attack satisfying the cost constraint $\hat{\mathrm{c}}(\mathbf{x}) \leq U$, for the following reason. Suppose $\binom{c}{d}, \binom{c'}{d'}$ are the attribute triples of two attacks on a node v with $c, c' \leq U$ and $d > d'$. If we would just keep the most damaging attack, we would have to discard $\binom{c'}{d'}$; however, similar to Example 4, this could cause us to miss high damage attacks later on in the bottom-up process. Thus a 'simple' bottom-up approach, in which only a single attack value is propagated, does not work; the best we can do is exclude attacks that at a node already exceed the cost budget. For CgD even this is impossible, as attacks that do not yet satisfy the minimum damage at a certain node may yet do so later.

C. Complexity

Theorem 5 states that the approaches of Theorems 3 and 4 are of exponential complexity.

Theorem 5. *The complexity of solving DgC and CDPF via Theorems 3 and 4 is $\mathcal{O}(2^{|B|})$. For CDPF this cannot be improved.*

The fact that CDPF cannot be computed in less than exponential time can be seen from Example 6 below, from the simple reason that the Pareto front may be of exponential size. For DgC, we have improved on the efficiency of CDPF in practice by disregarding attacks that exceed the cost constraint at any node. This does not work for CgD: if an attack does not reach the damage goal at node v, that is no reason to regard it as infeasible, because it may be combined with other attacks to increase damage. Therefore we need the full Pareto front to solve CgD in this fashion.

Example 6. *Let T be the AT given by $\mathrm{R}_T = \mathrm{OR}(v_0, \ldots, v_{n-1})$, where $\gamma(v_i) = \mathrm{BAS}$ and $\mathrm{c}(v_i) = \mathrm{d}(v_i) = 2^i$ for all $i < n$; furthermore $\mathrm{d}(\mathrm{R}_T) = 0$. Then*

$$\forall \mathbf{x} \in \mathcal{A} \colon \begin{pmatrix} \hat{\mathrm{c}} \\ \hat{\mathrm{d}} \end{pmatrix}(\mathbf{x}) = \begin{pmatrix} \sum_{i \colon x_{v_i}=1} 2^i \\ \sum_{i \colon x_{v_i}=1} 2^i \end{pmatrix},$$

so each $\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}(\mathbf{x})$ is optimal in $\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}}(\mathcal{A}) = \left\{ \binom{k}{k} \in \mathbb{R}^2_{\geq 0} \middle| k \in \{0, \ldots, 2^n - 1\} \right\}$. It follows that $|\mathrm{PF}(T)| = |\mathcal{A}| = 2^{|B|}$.

VII. DAG-like ATs, deterministic setting

If T is DAG-like, then the approach outlined in the previous section does not give the correct answer to DgC and CDPF. This is because for a node v with children v_1, v_2, the sub-ATs T_{v_1} and T_{v_2} may no longer be disjoint, and so equations (4) and (5) no longer hold. In particular, if $v_1 = v_2$, we have $\hat{\mathrm{c}}(\mathbf{x}) = \hat{\mathrm{c}}(\mathbf{x}_1) = \hat{\mathrm{c}}(\mathbf{x}_2)$ rather than (4).

A. CDPF

Instead, to solve CDPF, we introduce a novel approach based on *Biobjective Integer Linear Programming* (BILP) [18], i.e., an integer linear programming problem with two objective functions. A BILP problem is of the following form:

$$\text{minimise}_{\mathbf{y} \in \mathbb{Z}^n} \quad \binom{c_1 \cdot \mathbf{y}}{c_2 \cdot \mathbf{y}} \quad \text{subject to} \quad A \cdot \mathbf{y} \leq 0 \tag{6}$$

where $c_1, c_2 \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m \times n}$ for some integers m, n. The solution to this BILP problem is the Pareto front

$$\underline{\min} \left\{ \binom{c_1 \cdot \mathbf{y}}{c_2 \cdot \mathbf{y}} \in \mathbb{R}^2 \middle| \mathbf{y} \in \mathbb{Z}^n, A\mathbf{y} \leq 0 \right\},$$

where $\underline{\min}$ is taken in the poset $(\mathbb{R}^2, \leq)$. Solvers for BILP problems work by repeatedly solving single-objective integer linear programming problems [18]. In our case, as variables we use $\mathbf{y} \in \{0,1\}^N$, where we want y_v to represent $\mathrm{S}(\mathbf{x}, v)$ for an attack $\mathbf{x}$ (so $y_v = x_v$ for $v \in B$). Then the objective functions are

$$\hat{\mathrm{c}}(\mathbf{x}) = \sum_{v \in B} \mathrm{c}(v) y_v, \qquad -\hat{\mathrm{d}}(\mathbf{x}) = -\sum_{v \in N} \mathrm{d}(v) y_v.$$

We now need to describe the linear constraints on $\mathbf{y}$. If v is an AND-gate, we introduce a constraint $y_v \leq y_w$ for all children w of v; this ensures that $y_v = 0$ whenever at least one of the children has $y_w = 0$. If v is an OR-gate, then we introduce a constraint $y_v \leq \sum_{w \in \mathrm{ch}(v)} y_w$. Together, these constraints ensure that $y_v \leq \mathrm{S}(\mathbf{x}, v)$ for all v. Extra constraints that ensure the equality $y_v = \mathrm{S}(\mathbf{x}, v)$ are not necessary, because taking $\mathbf{y}$ such that equality holds turns out to be always Pareto optimal. This then leads to the following result:

Theorem 6. *CDPF is solved by solving the BILP problem given by*

$$\begin{aligned} \text{minimise}_{\mathbf{y} \in \{0,1\}^N} \quad & \begin{pmatrix} -\sum_{v \in N} \mathrm{d}(v) y_v \\ \sum_{v \in B} \mathrm{c}(v) y_v \end{pmatrix} \\ \text{subject to} \quad & \forall v \in \{v' \in V \mid \gamma(v') = \mathrm{AND}\}. \\ & \qquad \forall w \in \mathrm{Ch}(v).\ y_v \leq y_w, \\ & \forall v \in \{v' \in V \mid \gamma(v') = \mathrm{OR}\}. \\ & \qquad y_v \leq \sum_{w \in \mathrm{ch}(v)} y_w \end{aligned} \tag{7}$$

Example 7. *Applying Theorem 6 to the AT and cost/damage values of Example 1 yields the following BILP problem:*

$$\begin{aligned} \text{minimise}_{\mathbf{y} \in \{0,1\}^N} \quad & \begin{pmatrix} y_{\mathtt{ca}} + 3y_{\mathtt{pb}} + 2y_{\mathtt{fd}} \\ -10y_{\mathtt{fd}} - 100y_{\mathtt{dr}} - 200y_{\mathtt{ps}} \end{pmatrix} \\ \text{subject to} \quad & y_{\mathtt{dr}} \leq y_{\mathtt{fb}}, \\ & y_{\mathtt{dr}} \leq y_{\mathtt{fd}}, \\ & y_{\mathtt{ps}} \leq y_{\mathtt{ca}} + y_{\mathtt{dr}}. \end{aligned}$$

B. CgD and DgC

We solve CgD and DgC, by deriving *constrained single-objective optimization problems* from (7). Associated to a BILP problem (6) one has these single-objective problems:

$$\begin{aligned} &\text{minimize}_{\mathbf{y}\in\mathbb{Z}^n} \quad c_1\cdot\mathbf{y} \qquad \text{subject to} \quad A\cdot\mathbf{y}\le 0,\\ &\qquad\qquad\qquad\qquad\qquad\quad\ c_2\cdot\mathbf{y}\le C_2,\\ &\text{minimize}_{\mathbf{y}\in\mathbb{Z}^n} \quad c_2\cdot\mathbf{y} \qquad \text{subject to} \quad A\cdot\mathbf{y}\le 0,\\ &\qquad\qquad\qquad\qquad\qquad\quad\ c_1\cdot\mathbf{y}\le C_1. \end{aligned}$$

These are standard integer linear program (ILP) problems, for which efficient solvers exist [35]. By applying this to (7), we can formulate DgC and CgD as single-objective ILP problems, which can be fed to a solver.

Theorem 7. *DgC and CgD are solved by solving the constrained single-objective optimization problems derived from* (7) *with respective added constraints*

$$\sum_{v\in B}\mathrm{c}(v)y_v\le U, \qquad -\sum_{v\in N}\mathrm{d}(v)y_v\le -L.$$

Note that to solve DgC and CgD via Theorem 7, one does not need to first solve the BILP problem (7), but one can directly solve the single-objective problem.

VIII. Probabilistic cost-damage Pareto front

So far, we have assumed that any BAS undertaken by the attacker will succeed. However, in reality an attempted BAS may or may not succeed. Following earlier work [12], [36] we now assume a *probabilistic setting* in which each BAS v has a success probability $\mathrm{p}(v)$. More precisely, we assume:

1) The activation of the BASs may or may not succeed;
2) The successes of different BASs are independent;
3) The attacker pays the cost of a BAS, whether its activation succeeds or not;
4) All BASs are attempted simultaneously and paid for in advance;
5) Each BAS can only be attempted once.

The independence assumption is standard [12], [36], while the other assumptions lead to the most straightforward setting. Extensions are possible: for instance, the attacker might recoup some of the costs of failed activations, or BASs are attempted one by one and the attacker may choose to reallocate their budget based on BASs that have succeeded or failed their activation thusfar. Such extensions lead to more complicated models, and are left to future work.

Definition 5. *A* cdp-AT *is a tuple* $(T,\mathrm{c},\mathrm{d},\mathrm{p})$ *of an AT* T *and maps* $\mathrm{c}\colon B\to\mathbb{R}_{\ge 0}$, $\mathrm{d}\colon N\to\mathbb{R}_{\ge 0}$, *and* $\mathrm{p}\colon B\to[0,1]$.

In a cdp-AT, the damage done by an attack is a random variable: its value depends on the *actualized attack*, i.e., the BASs that succeed. Therefore, an attacker is interested in the *expected damage* of an attack.

Definition 6. *Let* $(T,\mathrm{c},\mathrm{d},\mathrm{p})$ *be a cdp-AT. For* $\mathbf{x}\in\mathcal{A}$, *define the* actualized attack *to be the random variable* $Y_{\mathbf{x}}$ *on* $\mathcal{A}$ *given by*

$$\mathbb{P}(Y_{\mathbf{x}}=\mathbf{y})=\begin{cases}\prod_{v\colon x_v=1}\mathrm{p}(v)^{y_v}(1-\mathrm{p}(v))^{1-y_v}, & \text{if } \mathbf{y}\preceq\mathbf{x},\\ 0, & \text{otherwise.}\end{cases}$$

We define the expected damage *of an attack to be* $\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})=\mathbb{E}[\hat{\mathrm{d}}(Y_{\mathbf{x}})]\in\mathbb{R}_{\ge 0}$.

Example 8. *We return to the setting of Example 1. We extend the cd-AT* $(T,\mathrm{c},\mathrm{d})$ *with a probability map* $\mathrm{p}\colon B\to[0,1]$ *given by* $\mathrm{p}(\mathtt{ca})=0.2$, $\mathrm{p}(\mathtt{pb})=0.4$ *and* $\mathrm{p}(\mathtt{fd})=0.9$. *We use this to calculate the function* $\hat{\mathrm{d}}_{\mathrm{E}}$; *we write an attack* $\mathbf{x}$ *as the vector* $(x_{\mathtt{ca}},x_{\mathtt{pb}},x_{\mathtt{fd}})$. *Then the random variable* $Y_{(0,1,1)}$ *is given by*

$$\begin{aligned}\mathbb{P}[Y_{(0,1,1)}=(0,0,0)]&=0.6\cdot 0.1=0.06,\\ \mathbb{P}[Y_{(0,1,1)}=(0,0,1)]&=0.6\cdot 0.9=0.54,\\ \mathbb{P}[Y_{(0,1,1)}=(0,1,0)]&=0.4\cdot 0.1=0.04,\\ \mathbb{P}[Y_{(0,1,1)}=(0,1,1)]&=0.4\cdot 0.9=0.36.\end{aligned}$$

Similar to $\hat{\mathrm{d}}_{\mathrm{E}}$, we also define $\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}_{\mathrm{E}}}(\mathbf{x})=(\hat{\mathrm{c}}(\mathbf{x}),\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x}))\in\mathbb{R}^2_{\ge 0}$. We then have the following probabilistic counterparts of CDPF, DgC, and CgD:

Problems. Given a cdp-AT $(T,\mathrm{c},\mathrm{d},\mathrm{p})$, solve the following problems:

- **CEDPF** Cost-expected damage Pareto front: find $\underline{\min}_{\sqsubseteq}\binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}_{\mathrm{E}}}(\mathcal{A})\subseteq\mathbb{R}^2_{\ge 0}$.
- **EDgC** Maximal expected damage given cost constraint: Given $U\in\mathbb{R}_{\ge 0}$, find $d_{\mathrm{E,opt}}=\max_{\mathbf{x}\colon\hat{\mathrm{c}}(\mathbf{x})\le U}\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})$.
- **CgED** Minimal cost given expected damage constraint: $L\in\mathbb{R}_{\ge 0}$, find $c_{\mathrm{E,opt}}=\min_{\mathbf{x}\colon\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})\ge L}\hat{\mathrm{c}}(\mathbf{x})$.

Example 9. *We continue Example 8. Using the definition of* $\hat{\mathrm{d}}$ *from the table in Example 1, we find* $\hat{\mathrm{d}}_{\mathrm{E}}(0,1,1)=0.06\cdot 0+0.54\cdot 0+0.04\cdot 10+0.36\cdot 310=112$.

Solving CDEPF naively is more involved than CDPF: not only do we have to calculate $\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})$ for exponentially many $\mathbf{x}$, but a single $\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})$ also requires $\mathbb{P}(Y_{\mathbf{x}}=\mathbf{y})\cdot\hat{\mathrm{d}}(\mathbf{y})$ for exponentially many $\mathbf{y}$. Therefore, we introduce new methods to solve CDEPF for treelike ATs in Section IX, by adapting the deterministic method of Section VI to account for probabilities. For DAG-like ATs, we cannot simply adapt the BILP method of Section VII, as (7) becomes nonlinear, and CDEPF, EDgC and CgED for DAG-like ATs are left to future work.

IX. Treelike ATs, probabilistic setting

EDgC and CEDPF for treelike ATs can be solved similar to the approach of Section VI. The main difference is that instead of working with the structure function $\mathrm{S}(\mathbf{x},v)$, we work with the *probabilistic structure function* $\mathrm{PS}(\mathbf{x},v):=\mathbb{P}(\mathrm{S}(Y_{\mathbf{x}},v)=1)$. With this notation we can write

$$\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})=\sum_{v\in N}\mathrm{PS}(\mathbf{x},v)\,\mathrm{d}(v).$$

Let v be a node with children v_1, v_2, and let $\mathbf{x} \in \mathcal{A}$. Since T is treelike, v_1 and v_2 do not have shared BASs. Since the truth values of the BASs in $Y_{\mathbf{x}}$ are independent of each other, this means that the random variables $\mathrm{S}(Y_{\mathbf{x}}, v_1)$ and $\mathrm{S}(Y_{\mathbf{x}}, v_2)$ are independent, and so we find

$$\begin{aligned}&\mathrm{PS}(\mathbf{x}, \mathrm{OR}(v_1, v_2))\\ &\quad = \mathrm{PS}(\mathbf{x}, v_1) + \mathrm{PS}(\mathbf{x}, v_2) - \mathrm{PS}(\mathbf{x}, v_1)\,\mathrm{PS}(\mathbf{x}, v_2), \qquad (8)\\ &\mathrm{PS}(\mathbf{x}, \mathrm{AND}(v_1, v_2))\\ &\quad = \mathrm{PS}(\mathbf{x}, v_1)\,\mathrm{PS}(\mathbf{x}, v_2). \qquad (9)\end{aligned}$$

On the other hand, we can express $\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})$ as

$$\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x}) = \hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x}_1) + \hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x}_2) + \mathrm{PS}(\mathbf{x}, v)\,\mathrm{d}(v). \qquad (10)$$

Combining this with (8) and (9) we can calculate the attributes $\hat{\mathrm{c}}$, $\hat{\mathrm{d}}_{\mathrm{E}}$, PS of attacks on v from their constituent attacks on v_1 and v_2. From here, we continue akin to Section VI. More precisely, we consider the *probabilistic attribute triple domain*, which is the poset $(\mathtt{PTrip}, \sqsubseteq)$ given by $\mathtt{PTrip} = \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \times [0,1]$ and $(c,d,p) \sqsubseteq (c',d',p')$ if and only if $c \leq c', d \geq d'$ and $p \geq p'$. For every node v we define a set $\mathcal{C}^{\mathrm{P}}_U(v) \subseteq \mathcal{P}(\mathtt{PTrip})$ of attribute triples. Just as in the deterministic case, we add the requirement $p \geq p'$ in determining feasibility because a greater activation probability of a node may lead to more damage higher up in the AT. As in Section VI, we define a map $\underline{\min}_U \colon \mathcal{P}(\mathtt{PTrip}) \to \mathcal{P}(\mathtt{PTrip})$ by $\underline{\min}_U(X) = \underline{\min}\left\{\begin{pmatrix} c \\ d \\ p \end{pmatrix} \in X : c \leq U\right\}$. Define $\star \colon [0,1]^2 \to [0,1]$ by $p \star p' = p + p' - pp'$. Then we again assume that T is binary, and we define $\mathcal{C}^{\mathrm{P}}_U(v)$ recursively by

$$\mathcal{C}^{\mathrm{P}}_U(v) \qquad (11)$$
$$= \begin{cases} \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \mathrm{c}(v) \\ \mathrm{p}(v)\,\mathrm{d}(v) \\ \mathrm{p}(v) \end{pmatrix} \right\}, & \text{if } \gamma(v) = \mathrm{BAS} \text{ and } \mathrm{c}(v) \leq U,\\ \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right\}, & \text{if } \gamma(v) = \mathrm{BAS} \text{ and } \mathrm{c}(v) > U, \end{cases}$$

$$\mathcal{C}^{\mathrm{P}}_U(\mathrm{OR}(v_1, v_2)) \qquad (12)$$
$$= \underline{\min}_U \left\{ \begin{pmatrix} c_1 + c_2 \\ d_1 + d_2 + (p_1 \star p_2)\cdot \mathrm{d}(v) \\ p_1 \star p_2 \end{pmatrix} \in \mathtt{PTrip} \middle| \begin{pmatrix} c_i \\ d_i \\ p_i \end{pmatrix} \in \mathcal{C}^{\mathrm{P}}_U(v_i) \right\},$$

$$\mathcal{C}^{\mathrm{P}}_U(\mathrm{AND}(v_1, v_2)) \qquad (13)$$
$$= \underline{\min}_U \left\{ \begin{pmatrix} c_1 + c_2 \\ d_1 + d_2 + p_1 p_2\,\mathrm{d}(v) \\ p_1 p_2 \end{pmatrix} \in \mathtt{PTrip} \middle| \begin{pmatrix} c_i \\ d_i \\ p_i \end{pmatrix} \in \mathcal{C}^{\mathrm{P}}_U(v_i) \right\}.$$

Then similar to the results in Section VI one can prove:

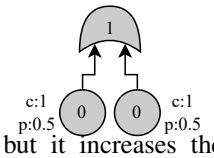
Theorem 8. *The solution to EDgC is given by* $\max\left\{ d \in \mathbb{R}_{\geq 0} \middle| \begin{pmatrix} c \\ d \\ p \end{pmatrix} \in \mathcal{C}^{\mathrm{P}}_U(\mathrm{R}_T) \right\}$.

Theorem 9. *The solution to CEDPF is given by* $\min \pi(\mathcal{C}^{\mathrm{P}}_\infty(\mathrm{R}_T))$*, where* $\pi \colon \mathtt{PTrip} \to \mathbb{R}^2_{\geq 0}$ *is the projection map onto the first two coefficients.*

In the worst case, the complexity of this approach will be the same as in Section VI; the Pareto frontier can still be of exponential size. Typically, however, $\mathcal{C}^{\mathrm{P}}_U(v)$ will be larger than $\mathcal{C}^{\mathrm{D}}_U(v)$; in the deterministic model, it is often nonoptimal to add BASs with no damage but with extra costs to an attack, when that attack already activates their parent nodes. However, in the probabilistic model, attempting extra BASs that are not needed in the deterministic model typically leads to a higher probability of activating the parent nodes, giving another way of increasing the cost of an attack to increase its expected damage.

Example 10. *Consider the AT with* $w = \mathrm{R}_T = \mathrm{OR}(v_1, v_2)$*, with* $\gamma(v_i) = \mathrm{BAS}$*,* $\mathrm{c}(v_i) = 1$*,* $\mathrm{d}(v_i) = 0$*,* $\mathrm{p}(v_i) = 0.5$ *for* $i = 1, 2$*, and* $\mathrm{d}(w) = 1$*. For* $U \geq 2$ *the incomplete Pareto fronts* $\mathcal{C}^{\mathrm{D}}_U$ *and* $\mathcal{C}^{\mathrm{P}}_U$ *are given in the table below.*

Node	$\mathcal{C}^{\mathrm{D}}_U$	$\mathcal{C}^{\mathrm{P}}_U$
v_1, v_2	$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0.5 \end{pmatrix} \right\}$
w	$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$	$\left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 2 \\ 0.75 \\ 0.75 \end{pmatrix} \right\}$



In the deterministic case one v_i suffices to reach w, and activating the other comes with extra costs without benefit, which is infeasible. In the probabilistic case attempting both v_i instead comes at the same extra cost, but it increases the expected damage because it increases the probability of w being reached.

For DAG-like ATs in the probabilistic setting one cannot transpose our BILP approach of Section VII, because the associated equations become nonlinear. For instance, if we introduce a vector $\vec{y} \in [0,1]^N$ where y_v represents $\mathrm{PS}(\mathbf{x}, v)$, then for $v = \mathrm{AND}(v_1, v_2)$ we get a constraint $y_v = y_{v_1} \cdot y_{v_2}$, which is nonlinear. In Section VII, this issue was circumvented because this equation can be linearized if one knows $y_v \in \{0, 1\}$, but in general this is not possible. Therefore, we leave CEDPF, CgED and EDgC for DAG-like ATs as an open problem.

X. Experiments

We tested the validity of our methods by executing them on two established ATs from the literature; these model the attacks on private information of valuable assets in a wireles sensor network [22] and on a data server on a network behind a firewall [23]. We also evaluate computation time on a suite of randomly generated ATs. As discussed in Section II, existing approaches cannot be applied to solve the Cg(E)D, (E)DgC and C(E)DPF problems; instead, we compare computation time to an enumerative method that goes through all attacks to find the Pareto optimal ones.

The methods are implemented in Matlab and executed on PC with an Intel Core i7-10750HQ 2.8GHz processor and 16GB memory. The source code can be found at [24]. The BILP problems are solved by translating them into single-objective problems via the methods of [18] in the YALMIP environment [21], which translates them into the Gurobi solver [19], a state-of-the-art optimizer that can handle ILP problems. We find that our methods compute C(E)DPF considerably faster than the naive method, and that the resulting Pareto front provides valuable insight into the weak points of the system.

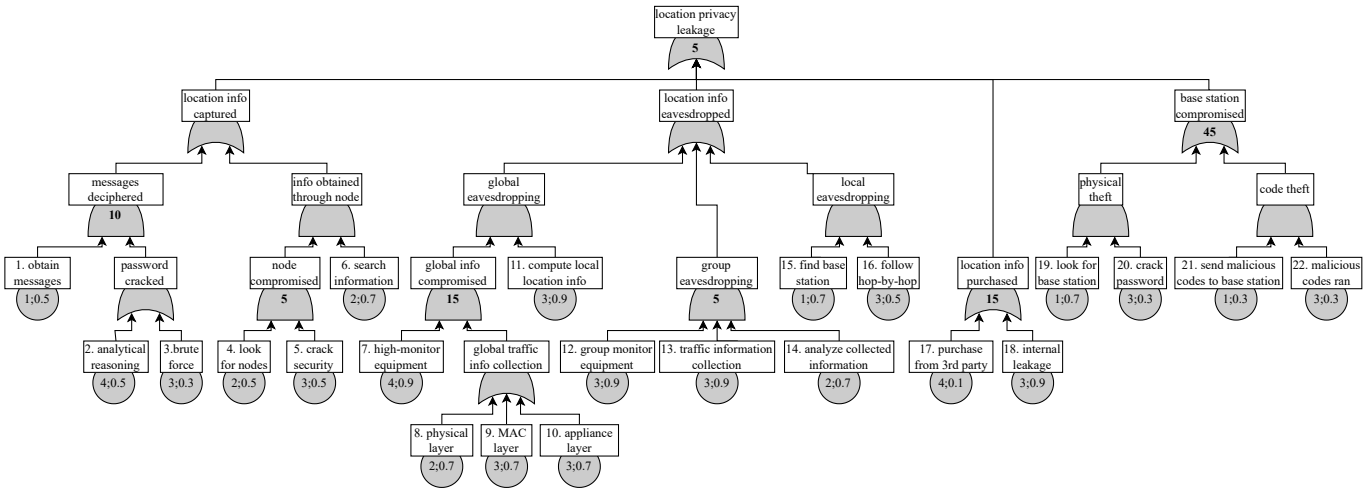


Fig. 4: Attack tree for privacy attacks on a giant panda preservation IOT monitoring system [22]. Nonzero damage values (in million USD) are in **bold**, BASs have cost values (unitless) and probabilities inscribed.

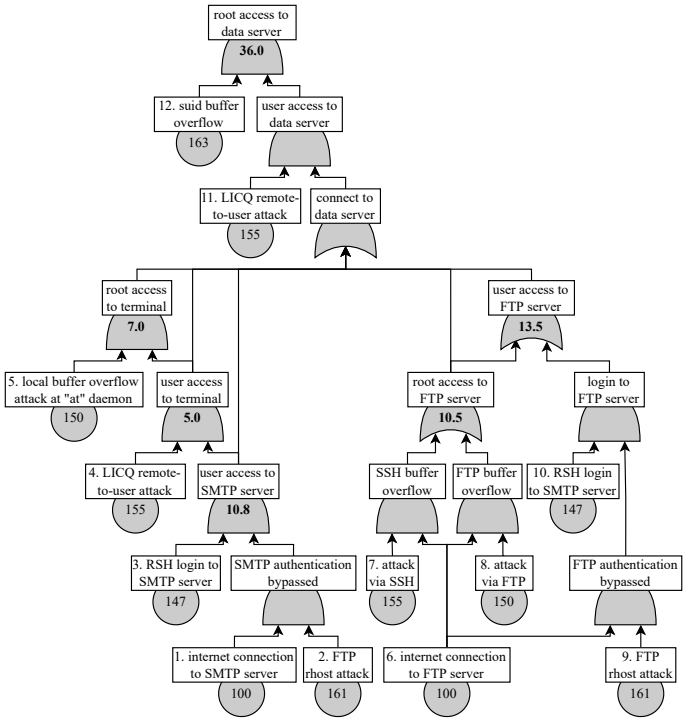


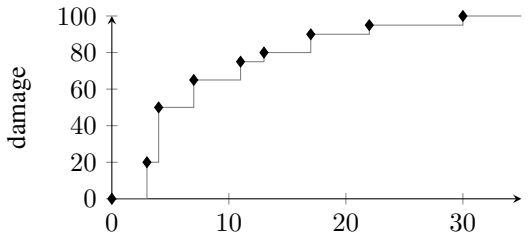
Fig. 5: Attack tree for a data server on a network [23]. Nonzero damage values (unitless) are in **bold**, BASs have cost (in seconds) inscribed.

A. IoT sensor network for wildlife observation

The first AT [22] is treelike (Fig. 4). It shows attacks on a wireless IoT sensor network that have the goal of obtaining the location information of valuable assets; in this case, giant pandas in a reservation in China [22]. The costs of BASs are given in [22] as unitless values 1–5. Detection probability is also given as a value 1–5; we take this as the BAS's success probability by converting it to a value 0.1–0.9. The work [22] does not contain damage values; instead, we estimate these from the economic value of giant pandas and the average reservation size [37]. The top event (the location information of one giant panda) only does minor damage compared to some of the internal nodes; e.g, if the base station is compromised, all pandas' location information is leaked.

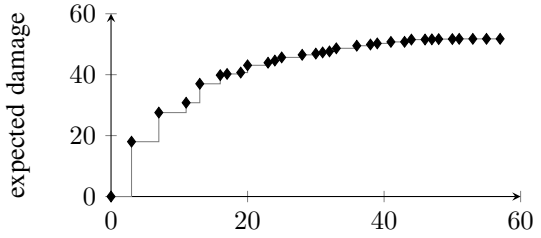
On this AT, we first disregard probability and calculate the cost-damage Pareto front bottom-up via Theorem 4. The resulting Pareto front is shown in Fig. 6a, and the corresponding Pareto-optimal attacks are listed as subsets of B (where b_i is the BAS numbered i in Fig. 4). As we can see, only a few of the 2^{22} possible attacks are Pareto optimal. Furthermore, every optimal attack contains at least one of the minimal attacks $\{b_{18}\}, \{b_{19}, b_{20}\}$ and $\{b_{21}, b_{22}\}$, and many contain two of them. These three minimal attacks do a lot of damage at relatively small cost; indeed, after these the curve tapers off, and extra cost beyond this has less damage impact. Thus, these attacks require the most defense, and security improvements should focus on location information leakage by internal sources (b_{18}) and base station compromise by either physical theft (b_{19}, b_{20}) or code theft (b_{21}, b_{22}). After defenses are put in place, a new cost-damage analysis is needed to see whether attack risks have been mitigated satisfactorily.

We also calculate the cost-expected damage Pareto front via Theorem 9. It has 31 Pareto-optimal attacks; this increase compared to the deterministic situation comes from the fact that in the probabilistic case it is beneficial to activate multiple children of an OR-gate, as in Example 10. Again the attack $\{b_{18}\}$ is Pareto-optimal at $(3, 18)$; however, $\{b_{19}, b_{20}\}$ and $\{b_{21}, b_{22}\}$ have expected damage 10.5 and 4.5, respectively, and at cost 4 are no longer Pareto-optimal. Instead, the next Pareto-optimal attack is $\{b_{18}, b_{19}, b_{20}\}$, which targets two valuable low-level nodes. In this probabilistic setting, we see that internal local information leakage (b_{18}) is part of every Pareto-optimal attack, which suggests this is the most important attack to defend against.



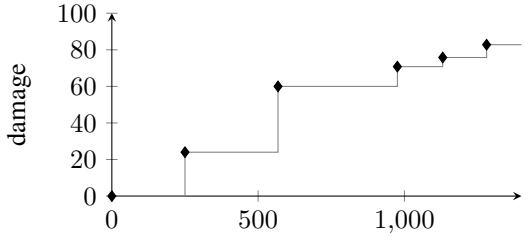
Attack	BASs	cost	damage	top
A_1	$\{b_{18}\}$	3	20	y
A_2	$\{b_{19}, b_{20}\}$ or $\{b_{21}, b_{22}\}$	4	50	y
A_3	$A_1 \cup A_2$	7	65	y
A_4	$A_3 \cup \{b_1, b_3\}$	11	75	y
A_5	$A_3 \cup \{b_7, b_8\}$	13	80	y
A_6	$A_4 \cup A_5$	17	90	y
A_7	$A_6 \cup \{b_4, b_5\}$	22	95	y
A_8	$A_7 \cup \{b_{11}, b_{12}, b_{13}\}$	30	100	y

(a) Cost-damage Pareto front for Fig. 4.



Attack	BASs	cost	damage	top
A_1	$\{b_{18}\}$	3	18.0	y
A_2	$A_1 \cup \{b_{19}, b_{20}\}$	7	27.6	y
A_3	$A_2 \cup \{b_{21}, b_{22}\}$	11	30.8	y
A_4	$A_2 \cup \{b_7, b_8\}$	13	37.0	y
A_5	$A_4 \cup \{b_9\}$	16	39.8	y
$\vdots$	$\vdots$	$\vdots$	$\vdots$	

(b) Cost-expected damage Pareto front for Fig. 4



Attack	BASs	cost	damage	top
A_1	$\{b_6, b_8\}$	250	24	n
A_2	$A_1 \cup \{b_{11}, b_{12}\}$	568	60	y
A_3	$A_2 \cup \{b_1, b_2, b_3\}$	976	70.8	y
A_4	$A_3 \cup \{b_4\}$	1131	75.8	y
A_5	$A_4 \cup \{b_5\}$	1281	82.8	y

(c) Cost-damage Pareto front for Fig. 5.

Fig. 6: Pareto fronts for the example ATs, together with the corresponding attacks as subsets of B. Except for A_1 of (c) all optimal attacks reach the top node.

B. Data server on a network

The second AT we consider represents the attack on a data server through a firewall using known exploits [23]. Since is DAG-like we only consider the deterministic case. The damage values are from [23] and represent unitless composites aggregating lost revenue, non-productive downtime, damage recovery, public embarassment, and law penalty. The cost is measured in time spent by the attacker, and the values are taken from [38], where the time taken by similar attacks is modeled via exponential distributions; we take the expected value as each node's duration. The rates in [38] are unitless, so we assume they are in $\frac{1}{100\,\mathrm{s}}$; this does not affect the Pareto front except for scaling. We have slightly changed the AT compared to [23] as the presentation there focused on vulnerabilities rather than attacks. Note that some nodes, such as `UserAccessToTerminal`, are superfluous if one only cares about activating the top node since they require `UserAccessToSMTPServer`, but they do play a role in cost-damage analysis since they carry damage values.

The results are depicted in Fig. 6c. There are 5 nonzero Pareto-optimal attacks. Furthermore, every Pareto-optimal attack contains the previous one. This implies that FTP buffer overflow attacks on the FTP server (b_6, b_8) are the most important BASs to defend against, followed by b_{11} and b_{12}, etc. Note that of these Pareto optimal attacks only A_2 would have been found by a minimal attack analysis.

C. Computation time: Case studies

We also measure the computation time of both our bottom-up and BILP methods for our analyses where applicable. We also compare it to an enumerative approach in which we calculate the cost and damage for each possible attack, and keep only the Pareto-optimal ones. For Fig. 4, this amounts to $2^{22} \approx 4 \cdot 10^6$ attacks. The bottom-up method is about $10\times$ as fast as BILP, and both outperform the enumerative method by an enormous margin, especially in the larger Fig. 4.

To check the robustness of our timing results, we also evaluate our methods on the same ATs, but with random $\mathrm{c}, \mathrm{d}, \mathrm{p}$ values on each node ($\mathrm{c}(v) \in \{1, \ldots, 10\}$, $\mathrm{d}(v) \in \{0, \ldots, 10\}$, $\mathrm{p}(v) \in \{0.1, 0.2, \ldots, 1.0\}$). The average computation time and standard deviations are given in Table III. For the bottom-up methods, the results conform to our earlier results, but *BILP is* slower; this may be because the random ATs contain considerably more nonzero values. The enumerative method is skipped because it is a lot slower than our new approaches; we compare it to our methods more comprehensively below.

D. Computation time: Randomly generated ATs

We also apply our methods to a suite of ATs, randomly generated through a method adapted from [39]. More specifically, we generate ATs by taking literature ATs (see Table IV) and combining them in one of the three following ways (see [39]):

1) We take a random BAS from the first AT and replace it with the root of the second AT, thus joining the two ATs;
2) We give the roots of the two ATs a common parent with a random type;

AT	True c, d, (p) time (BU)	time (BILP)	time (enumerative)	Random c, d, (p) time (BU)	time (BILP)	time (enumerative)
Fig. 4 deterministic	0.044s	0.438s	34h	0.037s±0.004s	3.144s±0.526s	
Fig. 4 probabilistic	0.047s	n/a	49h	0.051s±0.012s	n/a	
Fig. 5 deterministic	n/a	0.380s	79.53s	n/a	1.558s±0.252s	84.19s±4.79s

TABLE III: Computation time for C(E)DPF for the given ATs using bottom-up methods (Theorems 4 & 9), BILP (Theorem 6) and enumerative methods for their given $\mathrm{c}, \mathrm{d}, \mathrm{p}$ values, and average and standard deviations over 100 random $\mathrm{c}, \mathrm{d}, \mathrm{p}$ values.

Source	$\lvert N\rvert$	treelike	Source	$\lvert N\rvert$	treelike
[11] Fig. 1	12	no	[40] Fig. 3	8	yes
[11] Fig. 8	20	no	[40] Fig. 5	21	yes
[11] Fig. 9	12	no	[40] Fig. 7	25	yes
[8] Fig. 1	16	no	[41] Fig. 2	20	yes
			[17] Fig. 1	15	yes

TABLE IV: ATs from the literature used as building blocks. The trees from [41] and [17] are attack-defense trees; only the root component of the attack part was used for these trees.

3) Same as the previous, but we also identify two randomly chosen BASs, one from each AT.

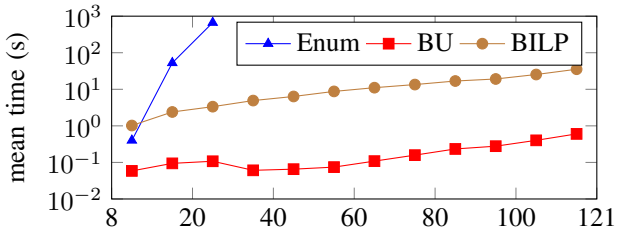
For each integer $1 \leq n \leq 100$, we combine ATs from Table IV via a method randomly drawn from the three above until the resulting AT satisfies $|N| \geq n$. We do this five times for each n, yielding a testing suite $\mathcal{T}_{\mathrm{DAG}}$ of 500 DATs, with random $\mathrm{c}, \mathrm{d}, \mathrm{p}$ as above. To test our bottom-up methods, we also create a suite $\mathcal{T}_{\mathrm{tree}}$ of treelike ATs, using the first two combining methods above and only the treelike ATs from Table IV.

We evaluate computation times and average the results in groups of ATs grouped by $\lfloor N \rfloor/10$; see Fig. 7. We only evaluated the enumerative method for the first 3 groups. Again, BU is faster than BILP, and both are considerably faster than the enumerative approach. For large ATs probabilistic BU is slower than deterministic BU, which is not yet seen from the case study ($N = 38$). This is probably because not only there are exponentially many attacks to consider, each attack also considers exponentially many actualized attacks to calculate expected damage, see Example 8.

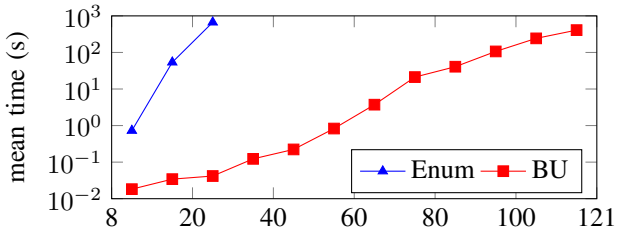
XI. Conclusion

This paper introduced two novel methods to solve cost-damage problems for attack trees, both by optimizing damage (resp. cost) under a cost (resp. damage) constraint, and by calculating the cost-damage Pareto front. For treelike ATs, this is done via bottom-up methods, both in the deterministic and the probabilistic case. For DAG-like ATs in the deterministic case, we introduce a method based on integer linear programming.

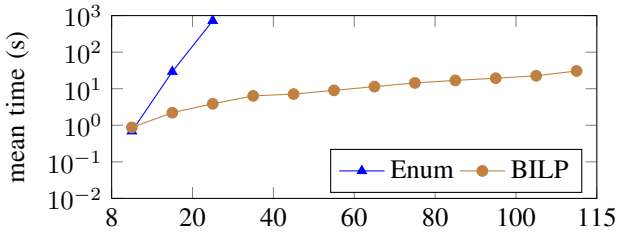
There are multiple avenues for further research. An obvious one is the probabilistic case for DAG-like ATs, which is not discussed in this paper. One approach would be to use a bottom-up approach, but in a polynomial ring with formal variables for nodes that occur multiple times, rather than in the real numbers. In that way, one can keep track of which nodes occur twice, and tweak addition to prevent double counting. Another extension is to compare the formal, provably optimal approach presented in this paper with a genetic algorithm approach to multiobjective optimization to approximate the Pareto front [32]. From experiments it could be established to



(a) $\mathcal{T}_{\mathrm{tree}}$ deterministic



(b) $\mathcal{T}_{\mathrm{tree}}$ probabilistic



(c) $\mathcal{T}_{\mathrm{DAG}}$ deterministic

		BU	BILP	enum*
$\mathcal{T}_{\mathrm{tree}}$ det.	min	<0.01s	0.563s	0.266s
	mean	0.159s	11.12s	86.87s
	max	1.141s	44.86s	3917s
$\mathcal{T}_{\mathrm{tree}}$ prob.	min	<0.01s		0.313s
	mean	42.30s		426.0s
	max	1335s		3853s
$\mathcal{T}_{\mathrm{DAG}}$ prob.	min		0.781s	0.313s
	mean		11.09s	296.3s
	max		50.08s	5619s

(d) Overall statistics. *Only ATs with $N < 30$.

Fig. 7: Computation time on randomly generated ATs. Means over subsets grouped by $\lfloor N/10 \rfloor$.

what extent the performance gain (if any) from using genetic algorithms comes at an accuracy cost. Finally, the cost and damage values may not be precisely known, but carry some uncertainty. A more elaborate analysis can incorporate this uncertainty, for example using fuzzy numbers, to obtain a robust version of the cost-damage Pareto front.

References

[1] Y. Roudier and L. Apvrille, "SysML-Sec: A model driven approach for designing safe and secure systems," in *MODELSWARD*, IEEE, 2015, pp. 655–664, ISBN: 978-989-758-136-6.

[2] L. Apvrille and Y. Roudier, "SysML-sec: A sysML environment for the design and development of secure embedded systems," in *APCOSEC*, 2013. [Online]. Available: http://www.eurecom.fr/publication/4186.

[3] Isograph, *AttackTree*. [Online]. Available: https://www.isograph.com/software/attacktree/.

[4] J. R. Surdu, J. M. Hill, R. Dodge, S. Lathrop, and C. Carver, "Military academy attack/defense network simulation," in *Proceedings of advanced simulation technology symposium. Military, government, and aerospace simulation*, 2003.

[5] A. Yasinisac and J. H. Pardue, "A process for assessing voting system risk using threat trees," *Journal of Information Systems Applied Research*, vol. 4, no. 1, p. 4, 2011.

[6] F. Kammüller, J. R. Nurse, and C. W. Probst, "Attack tree analysis for insider threats on the iot using isabelle," in *International Conference on Human Aspects of Information Security, Privacy, and Trust*, Springer, 2016, pp. 234–246.

[7] B. Fila and W. Wideł, "Efficient attack-defense tree analysis using pareto attribute domains," in *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, IEEE, 2019, pp. 200–20 015.

[8] F. Arnold, D. Guck, R. Kumar, and M. Stoelinga, "Sequential and Parallel Attack Tree Modelling," in *SAFECOMP*, ser. LNCS, vol. 9338, Springer International Publishing, 2015, pp. 291–299. DOI: 10.1007/978-3-319-24249-1_25.

[9] V. Saini, Q. Duan, and V. Paruchuri, "Threat modeling using attack trees," *Journal of Computing Sciences in Colleges*, vol. 23, no. 4, pp. 124–131, 2008.

[10] P. Hopkin, *Fundamentals of risk management: understanding, evaluating and implementing effective risk management*. Kogan Page Publishers, 2018.

[11] R. Kumar, E. Ruijters, and M. Stoelinga, "Quantitative attack tree analysis via priced timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*, Springer, 2015, pp. 156–171.

[12] C. E. Budde and M. Stoelinga, "Efficient algorithms for quantitative attack tree analysis," in *CSF*, IEEE Computer Society, 2021, pp. 501–515. DOI: 10.1109/CSF51468.2021.00041.

[13] K. Dudziński and S. Walukiewicz, "Exact methods for the knapsack problem and its generalizations," en, *European Journal of Operational Research*, vol. 28, no. 1, pp. 3–21, Jan. 1987, ISSN: 03772217. DOI: 10.1016/0377-2217(87)90165-2. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0377221787901652 (visited on 02/24/2022).

[14] G. Gallo, P. L. Hammer, and B. Simeone, "Quadratic knapsack problems," in *Combinatorial Optimization*, M. W. Padberg, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1980, pp. 132–149, ISBN: 978-3-642-00802-3. DOI: 10.1007/BFb0120892. [Online]. Available: https://doi.org/10.1007/BFb0120892.

[15] R. J. Forrester and L. A. Waddell, "Strengthening a linear reformulation of the 0-1 cubic knapsack problem via variable reordering," en, *Journal of Combinatorial Optimization*, Jan. 2022, ISSN: 1382-6905, 1573-2886. DOI: 10.1007/s10878-021-00840-z. [Online]. Available: https://link.springer.com/10.1007/s10878-021-00840-z (visited on 02/24/2022).

[16] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004, ISSN: 0167-6377. DOI: https://doi.org/10.1016/S0167-6377(03)00062-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167637703000622.

[17] B. Kordy and W. Wideł, "On quantitative analysis of attack–defense trees with repeated labels," in *International Conference on Principles of Security and Trust*, Springer, 2018, pp. 325–346.

[18] M. Özlen and M. Azizoğlu, "Multi-objective integer programming: A general approach for generating all non-dominated solutions," *European Journal of Operational Research*, vol. 199, no. 1, pp. 25–35, 2009.

[19] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2022. [Online]. Available: https://www.gurobi.com.

[20] T. Stidsen, K. A. Andersen, and B. Dammann, "A branch and bound algorithm for a class of biobjective mixed integer programs," *Management Science*, vol. 60, no. 4, pp. 1009–1032, 2014.

[21] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, IEEE, 2004, pp. 284–289.

[22] R. Jiang, J. Luo, and X. Wang, "An attack tree based risk assessment for location privacy in wireless sensor networks," in *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing*, IEEE, 2012, pp. 1–4.

[23] R. Dewri, I. Ray, N. Poolsappasit, and D. Whitley, "Optimal security hardening on attack tree models of networks: A cost-benefit analysis," *International Journal of Information Security*, vol. 11, no. 3, pp. 167–188, 2012.

[24] Anonymous. "Cost-damage analysis of attack trees." (2022), [Online]. Available: https://drive.google.com/drive/folders/1Ww2iEeH80akFlehtLK8TBuk764gAhJbw?usp=share_link.

[25] A. Bobbio, L. Egidi, and R. Terruggia, "A methodology for qualitative/quantitative analysis of weighted attack

trees," *IFAC Proceedings Volumes*, vol. 46, no. 22, pp. 133–138, 2013.

[26] A. Lenin, J. Willemson, and D. P. Sari, "Attacker profiling in quantitative security assessment based on attack trees," in *Secure IT Systems*, K. Bernsmed and S. Fischer-Hübner, Eds., Cham: Springer International Publishing, 2014, pp. 199–212, ISBN: 978-3-319-11599-3.

[27] A. Jürgenson and J. Willemson, "Computing exact outcomes of multi-parameter attack trees," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, Springer, 2008, pp. 1036–1051.

[28] T. Ingoldsby, "Fundamentals of capabilities-based attack tree analysis," *Amenaza Technologies Limited*, pp. 406–917, 2005.

[29] É. André, D. Lime, M. Ramparison, and M. Stoelinga, "Parametric analyses of attack-fault trees," *Fundamenta Informaticae*, vol. 182, no. 1, pp. 69–94, 2021.

[30] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *International Conference on Information Security and Cryptology*, Springer, 2005, pp. 186–198.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[32] S. Ali, P. Arcaini, and T. Yue, "Do quality indicators prefer particular multi-objective search algorithms in search-based software engineering?" In *International Symposium on Search Based Software Engineering*, Springer, 2020, pp. 25–41.

[33] M. R. Garey and D. S. Johnson, *Computers and intractability*. San Francisco: Freeman, 1979.

[34] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, Springer, 1972, pp. 85–103.

[35] D.-S. Chen, R. G. Batson, and Y. Dang, *Applied integer programming: modeling and solution*. John Wiley & Sons, 2011.

[36] A. Rauzy, "New algorithms for fault trees analysis," *Reliability Engineering & System Safety*, vol. 40, no. 3, pp. 203–211, 1993.

[37] F. Wei, R. Costanza, Q. Dai, *et al.*, "The value of ecosystem services from giant panda reserves," *Current Biology*, vol. 28, no. 13, pp. 2174–2180, 2018.

[38] F. Zhao, H. Huang, H. Jin, and Q. Zhang, "A hybrid ranking approach to estimate vulnerability for dynamic attacks," *Computers & Mathematics with Applications*, vol. 62, no. 12, pp. 4308–4321, 2011.

[39] M. Lopuhaä-Zwakenberg and M. Stoelinga, "Attack time analysis in dynamic attack trees via integer linear programming," *arXiv preprint arXiv:2111.05114*, 2021.

[40] F. Arnold, H. Hermanns, R. Pulungan, and M. Stoelinga, "Time-dependent analysis of attacks," in *International Conference on Principles of Security and Trust*, Springer, 2014, pp. 285–305.

[41] M. Fraile, M. Ford, O. Gadyatskaya, R. Kumar, M. Stoelinga, and R. Trujillo-Rasua, "Using attack-defense trees to analyze threats and countermeasures in an atm: A case study," in *IFIP Working Conference on The Practice of Enterprise Modeling*, Springer, 2016, pp. 326–334.

APPENDIX

A. Proof of Theorem 2

Theorem 2. *Let X be a finite set, and let $f\colon \mathbb{B}^X \to \mathbb{R}_{\geq 0}$ be any nondecreasing function. Then there is a cd-AT $(T, \mathrm{c}, \mathrm{d})$ with $B = X$ and $\hat{\mathrm{d}} = f$.*

Proof. Define $n = |X|$. Let $f\colon \mathbb{B}^X \to \mathbb{R}_{\geq 0}$ be a nondecreasing function, and let $\mathbf{x}^1, \ldots, \mathbf{x}^{2^n}$ be an ordering of $\mathbb{B}^n$ such that $f(\mathbf{x}^i) \leq f(\mathbf{x}^{i+1})$ for all $i < 2^n$, and such that $\mathbf{x}^i \preceq \mathbf{x}^j$ implies $i \leq j$ for all i and j. Since f is nondecreasing, these two conditions can be fulfilled simultaneously. Furthermore, define an AT T by having $B = X$ and non-leaf nodes $\{A_i, O_i\}_{i \leq 2^n}$ and R_T given by

$$\begin{aligned} A_i &= \mathrm{AND}(\{v \in B | x_v^i = 1\}), \\ O_j &= \mathrm{OR}(\{A_i | i \geq j\}), \\ \mathrm{R}_T &= \mathrm{AND}(\{O_j \mid j \leq 2^n\}). \end{aligned}$$

Furthermore, we define $d\colon N \to \mathbb{R}_{\geq 0}$ by

$$\begin{aligned} \mathrm{d}(v) &= 0 && \forall v \in B, \\ \mathrm{d}(A_i) &= 0 && \forall i \leq 2^n, \\ \mathrm{d}(O_1) &= f(\mathbf{x}^1), && \\ \mathrm{d}(O_{j+1}) &= f(\mathbf{x}^{j+1}) - f(\mathbf{x}^j) && \forall j < 2^n, \\ \mathrm{d}(\mathrm{R}_T) &= 0. && \end{aligned}$$

Note that R_T does not play a role in the cost-damage analysis and is here purely to satisfy the condition of T having a root. Now consider an attack $\mathbf{x}_i$; then $\mathrm{S}(\mathbf{x}^i, A_j) = 1$ if and only if $\mathbf{x}^j \preceq \mathbf{x}^i$. It follows that $\mathrm{S}(\mathbf{x}^i, O_j) = 1$ if and only if there is an $\mathbf{x}^k$ with $j \leq k$ and $\mathbf{x}^j \preceq \mathbf{x}^i$. The latter can only happen when $j \leq i$, and so $\mathrm{S}(\mathbf{x}^i, O_j) = 1$ if and only if $j \leq i$. It follows that

$$\hat{\mathrm{d}}(\mathbf{x}^i) = \sum_{j \leq i} \mathrm{d}(O_j) = f(\mathbf{x}^i)$$

and since this holds for all i, we have $f = \hat{\mathrm{d}}$. □

B. Proof of Theorem 1

Theorem 1. *CDDP is NP-complete, even when restricted to treelike ATs.*

Proof. First, we show that CDDP is in NP. A witness of a CDDP problem determined by $(T, \mathrm{c}, \mathrm{d}, U, L)$ is given by an attack $\mathbf{x} \in A$; to verify this we need to calculate $\hat{\mathrm{c}}(\mathbf{x})$ and $\hat{\mathrm{d}}(\mathbf{x})$. By Definition 4, the former can be calculated in $\mathcal{O}(|B|)$ time. The latter can be calculated in $\mathcal{O}(|N| + |E|)$ time, as calculating $\mathrm{S}(\mathbf{x}, v)$ for all v takes $\mathcal{O}(|N| + |E|)$ time via Definition 3. We conclude that CDDP is in NP.

Consider a binary knapsack decision problem, i.e. two linear functions $f, g\colon \mathbb{B}^n \to \mathbb{R}_{\geq 0}$, an upper bound U and a lower

bound L; the problem is to determine whether there exists a $\mathbf{x} \in \mathbb{B}^n$ such that $f(\mathbf{x}) \geq L$ and $g(\mathbf{x}) \leq U$. This problem is known to be NP-complete [33], and we prove that the cost-damage decision problem by transforming the binary knapsack decision problem into it.

Since f and g are linear and their images lie in $\mathbb{R}_{\geq 0}$, there exist $f_1, \ldots, f_n, g_1, \ldots, g_n \in \mathbb{R}_{\geq 0}$ such that $f(\mathbf{x}) = \sum_{i=1}^n f_i x_i$ and $g(\mathbf{x}) = \sum_{i=1}^n f_i x_i$. Now define a cd-AT $(T, \mathrm{c}, \mathrm{d})$ by taking $N = \{v_1, \ldots, v_n, \mathrm{R}_T\}$, with the v_i BASs and $\mathrm{R}_T = \mathrm{AND}(v_1, \ldots, v_n)$. Furthermore, take $\mathrm{c}(v_i) = g_i$, and $\mathrm{d}(v_i) = f_i$ and $\mathrm{d}(\mathrm{R}_T) = 0$. Then $\hat{\mathrm{c}}(\mathbf{x}) = g(\mathbf{x})$ and $\hat{\mathrm{d}}(\mathbf{x}) = f(\mathbf{x})$, and so a solution to the cost-damage decision problem is a solution to the binary knapsack decision problem. Since $(T, \mathrm{c}, \mathrm{d})$ is polynomial (even linear) in the size of the original knapsack problem, this shows that the cost-damage decision problem is NP-complete. □

C. Proof of Theorem 5

Before we prove the theorem, we first consider the following auxiliary lemma.

Lemma 1. *Let $T = (N, E)$ be a binary tree, i.e., $\mathrm{Ch}(v) \in \{0, 2\}$ for all $v \in N$. For a non-leaf node v, let $b(v)$ be the number of leaf descendants of v (with edges pointing away from the root). Let $v_1, \ldots, v_K$ be an enumeration of the non-leaf nodes of T such that $b(v_i) \leq b(v_j)$ whenever $i \leq j$. Then $b(v_i) \leq i + 1$.*

Proof. We prove this by induction on $|N|$; it is clearly true if $|N| = 3$, where the single internal node, the root, has two leaves as children. Let T be a full binary tree with $|N| > 3$, and let R_T be its root; let a_1, a_2 be its children. Furthermore, let $T_i = (N_i, E_i)$ be the subtree consisting of a_i and its descendants. Let $v_1, \ldots, v_K$ be an enumeration of the non-leaf nodes of N such that $b(v_i) \leq b(v_j)$ whenever $i \leq j$. Let $i \leq K$; we aim to prove that $b(v_i) \leq i+1$. If $v_i \in N_1$, define $r := |\{j \leq i \mid v_j \in N_1\}|$. Then $r \leq i$. On the other hand, if we restrict the sequence $v_1, \ldots, v_K$ to just elements of N_1, the resulting sequence satisfies the conditions of the Lemma for T_1, and v_i is the r-th element in this sequence. It follows from the induction hypothesis that $b(v_i) \leq r + 1$; hence certainly $b(v_i) \leq i+1$. The case that $v_i \in N_2$ is handled similarly. The last remaining case is $v_i = \mathrm{R}_T$; but this happens when $i = K$ as R_T is necessarily the last element of the sequence. Since T is a binary tree, one has $K = |B| - 1$, from which this case also follows. □

Theorem 5. *The complexity of solving DgC and CDPF via Theorems 3 and 4 is $\mathcal{O}(2^{|B|})$. For CDPF this cannot be improved.*

Proof. At a node v with children v_1, v_2, we have to do at most $|\mathcal{C}_U^{\mathrm{D}}(v_1)| \cdot |\mathcal{C}_U^{\mathrm{D}}(v_2)|$ computations. We also have $|\mathcal{C}_U^{\mathrm{D}}(v)| \leq |\mathcal{C}_U^{\mathrm{D}}(v_1)| \cdot |\mathcal{C}_U^{\mathrm{D}}(v_2)|$. Since $|\mathcal{C}_U^{\mathrm{D}}(v)| \leq 2$ if $v \in B$, one straightforwardly proves by induction that for inner nodes one has $|\mathcal{C}_U^{\mathrm{D}}(v)| \leq 2^{b(v)}$, where $b(v)$ is the number of BAS descendants of v as in Lemma 1. It also follows that at v we have to do at most $2^{b(v)}$ computations. Let $v_1, \ldots, v_{|B|-1}$ be an enumeration of $N \setminus B$ such that $b(v_i) \leq b(v_j)$ whenever $v_i \leq v_j$; then the total number of computations is equal to

$$\sum_{i=1}^{|B|-1} 2^{b(v)} \leq \sum_{i=1}^{|B|-1} 2^{i+1} = 2^{|B|+1} - 2, \tag{14}$$

where the inequality follows from Lemma 1. This shows the statement about complexity. The fact that this cannot be improved for CDPF follows from Example 6, which shows that the Pareto front can be of size $2^{|B|}$; in particular, outputting it takes at least that much time. □

D. Proofs of Theorems 6 and 7

We only prove Theorem 6, as the proof of Theorem 7 is essentially the same but less involved, as the optimization is only done in one dimension.

Theorem 6. *CDPF is solved by solving the BILP problem given by*

$$\begin{aligned} \text{minimise}_{\mathbf{y} \in \{0,1\}^N} \quad & \begin{pmatrix} -\sum_{v \in N} \mathrm{d}(v) y_v \\ \sum_{v \in B} \mathrm{c}(v) y_v \end{pmatrix} \\ \text{subject to} \quad & \forall v \in \{v' \in V \mid \gamma(v') = \mathrm{AND}\}. \\ & \qquad \forall w \in \mathrm{Ch}(v).\ y_v \leq y_w, \\ & \forall v \in \{v' \in V \mid \gamma(v') = \mathrm{OR}\}. \\ & \qquad y_v \leq \sum_{w \in \mathrm{ch}(v)} y_w \end{aligned} \tag{7}$$

Proof. Let $\mathbf{y} \in \mathbb{B}^N$ satisfy the conditions of (6), and let $\mathbf{x} \in \mathbb{B}^B$ be the attack given by $x_v = y_v$ for $v \in B$. We claim that $y_v \leq \mathrm{S}(\mathbf{x}, v)$ for all $v \in N$, and we prove this by induction; clearly it is true for BASs. Suppose the claim is true for $v_1, \ldots, v_n$, and consider $v = \mathrm{AND}(v_1, \ldots, v_n)$. Then (6) amounts to $y_v \leq \min\{y_{v_i} \mid i \leq n\}$. By the induction hypothesis we then have

$$y_v \leq \min\{y_{v_i} \mid i \leq n\} \leq \min\{\mathrm{S}(\mathbf{x}, v_i) \mid i \leq n\} = \mathrm{S}(\mathbf{x}, v). \tag{15}$$

Similarly, if $v = \mathrm{OR}(v_1, \ldots, v_n)$, we get

$$\begin{aligned} y_v &\leq \min\left\{1, \sum_{i \leq n} y_{v_i}\right\} \\ &\leq \min\left\{1, \sum_{i \leq n} \mathrm{S}(\mathbf{x}, v_i)\right\} \\ &= \begin{cases} 0, & \text{if } \mathrm{S}(\mathbf{x}, v_i) = 0 \text{ for all } i, \\ 1, & \text{otherwise} \end{cases} \\ &= \mathrm{S}(\mathbf{x}, v). \end{aligned}$$

This proves the claim.

We now continue with the proof of the theorem. Let $\mathcal{F}$ be the set of $\mathbf{y}$ satisfying the conditions of (6), and write for $\mathbf{y} \in \mathcal{F}$:

$$f(\mathbf{y}) = \sum_{v \in B} \mathrm{c}(v) y_v, \qquad g(\mathbf{y}) = -\sum_{v \in N} \mathrm{d}(v) y_v.$$

Our aim is to prove the following equality:

$$\{\binom{c}{-\hat{d}} | \binom{c}{\hat{d}} \in \mathrm{PF}(T)\} = \underline{\min}\left\{ \begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix} \middle| \mathbf{y} \in \mathcal{F} \right\}, \tag{16}$$

where the $\underline{\min}$ on the RHS is taken in the poset $(\mathbb{R}^2, \leq)$. We first prove "$\supseteq$". Let $\mathbf{y} \in \mathcal{F}$ be such that $\begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix}$ is minimal. Let $\mathbf{x} \in \mathcal{A}$ be such that $x_v = y_v$ for all $v \in B$, and let $\mathbf{y}' \in \mathbb{B}^N$ be given by $y'_v = \mathrm{S}(\mathbf{x}, v)$ for all v. A straightforward induction proof shows that $\mathbf{y}' \in \mathcal{F}$, and by the claim we have $y_v \leq y'_v$ for all $v \in N$, with equality when $v \in B$. It follows that $f(\mathbf{y}) = f(\mathbf{y}')$ and $g(\mathbf{y}) \geq g(\mathbf{y}')$. Since $\begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix}$ is minimal, this must be an equality, and we get

$$\begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix} = \begin{pmatrix} f(\mathbf{y}') \\ g(\mathbf{y}') \end{pmatrix} = \begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}) \\ -\hat{\mathrm{d}}(\mathbf{x}) \end{pmatrix}.$$

It remains to be shown that $\begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}) \\ \hat{\mathrm{d}}(\mathbf{x}) \end{pmatrix} \in \mathrm{PF}(T)$. Let $\mathbf{x}'' \in \mathcal{A}$ be such that $\hat{\mathrm{c}}(\mathbf{x}') \leq \hat{\mathrm{c}}(\mathbf{x})$ and $\hat{\mathrm{d}}(\mathbf{x}'') \geq \hat{\mathrm{d}}(\mathbf{x})$. Let $\mathbf{y}'' \in \mathbb{B}^N$ be such that $y''_v = \mathrm{S}(\mathbf{x}'', v)$ for all v; then again $\mathbf{y}'' \in \mathcal{F}$, and

$$\begin{pmatrix} f(\mathbf{y}'') \\ g(\mathbf{y}'') \end{pmatrix} = \begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}'') \\ -\hat{\mathrm{d}}(\mathbf{x}'') \end{pmatrix} \leq \begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}) \\ -\hat{\mathrm{d}}(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix}.$$

Since $\begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix}$ is minimal, this means that equality must hold here; this shows that $\mathbf{x}$ is Pareto optimal, and so we have shown "$\supseteq$" in (16).

The argument to prove "$\subseteq$" is very similar. Let $\mathbf{x} \in \mathcal{A}$ be such that $\begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}) \\ \hat{\mathrm{d}}(\mathbf{x}) \end{pmatrix} \in \mathrm{PF}(T)$, and let $\mathbf{y} \in \mathbb{B}^N$ be given by $y_v = \mathrm{S}(\mathbf{x}, v)$. Then $\mathbf{y} \in \mathcal{F}$ and $\begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix} = \begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}) \\ -\hat{\mathrm{d}}(\mathbf{x}) \end{pmatrix}$; we need to show that this vector is minimal. Let $\mathbf{y}' \in \mathcal{F}$ be such that $\begin{pmatrix} f(\mathbf{y}') \\ g(\mathbf{y}') \end{pmatrix} \leq \begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix}$. Define $\mathbf{x}'' \in \mathcal{A}$ by $x''_v = y'_v$ for all $v \in B$, and define $\mathbf{y}'' \in \mathbb{B}^N$ by $\mathbf{y}''_v = \mathrm{S}(\mathbf{x}'', v)$ for all $v \in N$. Similar to the above we have $\mathbf{y}'' \in \mathcal{F}$ and $\begin{pmatrix} f(\mathbf{y}'') \\ g(\mathbf{y}'') \end{pmatrix} \leq \begin{pmatrix} f(\mathbf{y}') \\ g(\mathbf{y}') \end{pmatrix}$. It follows that we have

$$\begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}'') \\ -\hat{\mathrm{d}}(\mathbf{x}'') \end{pmatrix} = \begin{pmatrix} f(\mathbf{y}'') \\ g(\mathbf{y}'') \end{pmatrix} \leq \begin{pmatrix} f(\mathbf{y}') \\ g(\mathbf{y}') \end{pmatrix} \leq \begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix} = \begin{pmatrix} \hat{\mathrm{c}}(\mathbf{x}) \\ -\hat{\mathrm{d}}(\mathbf{x}) \end{pmatrix}.$$

Since $\mathbf{x}$ is Pareto optimal by assumption, the above must have equalities throughout. In particular $\begin{pmatrix} f(\mathbf{y}') \\ g(\mathbf{y}') \end{pmatrix} = \begin{pmatrix} f(\mathbf{y}) \\ g(\mathbf{y}) \end{pmatrix}$, which proves that $\mathbf{y}$ is minimal. This shows "$\subseteq$" in (16), completing the proof. □

E. Proofs of Theorems 3, 4, 8 and 9

We will show that all these theorems follow from a shared main result, namely Theorem 10 below. In order to formulate it, we first need a little more notation. For a node v, we let $T_v = (N_v, E_v)$ be the sub-DAG of T consisting of v and all its descendants, together with its set of BASs B_v, set of attacks $\mathcal{A}_v$, cost, damage, and expected damage functions $\hat{\mathrm{c}}_v$, $\hat{\mathrm{d}}_v$ and $\hat{\mathrm{d}}_{\mathrm{E},v}$:

$$\begin{aligned} N_v &= \{w \in N \mid \exists \text{ path } v \to w\}, \\ E_v &= E \cap (N_v \times N_v), \\ T_v &= (N_v, E_v), \\ B_v &= B \cap N_v, \\ \mathcal{A}_v &= \mathbb{B}^{B_v}, \end{aligned}$$

$$\begin{aligned} \hat{\mathrm{c}}_v(\mathbf{x}) &= \sum_{w \in B_v} \mathrm{c}(w) x_w, && \text{for } \mathbf{x} \in \mathcal{A}_v, \\ \hat{\mathrm{d}}_v(\mathbf{x}) &= \sum_{w \in N_v} \mathrm{d}(w)\, \mathrm{S}(w), && \text{for } \mathbf{x} \in \mathcal{A}_v, \\ \hat{\mathrm{d}}_{\mathrm{E},v}(\mathbf{x}) &= \mathbb{E}\left[\hat{\mathrm{d}}_v(Y_{\mathbf{x}})\right], && \text{for } \mathbf{x} \in \mathcal{A}_v. \end{aligned}$$

Furthermore, we let $\mathrm{PS}_v \colon \mathcal{A}_v \times N_v \to \mathbb{B}$ be the probabilistic structure function of T_v, analogous to Definition 3. Based on the equations above we define, for a node v, the *extended expected attribute map* $\mathrm{EA}_v \colon \mathcal{A}_v \to \mathtt{PTrip}$ by

$$\mathrm{EA}(\mathbf{x}) = \begin{pmatrix} \hat{\mathrm{c}}_v(\mathbf{x}) \\ \hat{\mathrm{d}}_{\mathrm{E},v}(\mathbf{x}) \\ \mathrm{PS}(\mathbf{x}, v) \end{pmatrix}.$$

Then as we will show below, Theorems 3, 4, 8 and 9 all follow from the following result:

Theorem 10. *For every $v \in N$ one has*

$$\mathcal{C}^{\mathrm{P}}_U(v) = \underline{\min}\left\{\mathrm{EA}_v(\mathbf{x}) \in \mathtt{PTrip} | \mathbf{x} \in \mathcal{A}_v, \hat{\mathrm{c}}_v(\mathbf{x}) \leq U\right\}.$$

To prove this theorem we first need a few auxiliary lemmas, as well as some definitions. For $\begin{pmatrix} c_1 \\ d_1 \\ p_1 \end{pmatrix}, \begin{pmatrix} c_i \\ d_i \\ p_i \end{pmatrix} \in \mathtt{PTrip}$ and $d \in \mathbb{R}_{\geq 0}$, we define

$$\begin{aligned} \begin{pmatrix} c_1 \\ d_1 \\ p_1 \end{pmatrix} \triangle_d \begin{pmatrix} c_2 \\ d_2 \\ p_2 \end{pmatrix} &= \begin{pmatrix} c_1 + c_2 \\ d_1 + d_2 + p_1 p_2 d \\ p_1 p_2 \end{pmatrix}, \\ \begin{pmatrix} c_1 \\ d_1 \\ p_1 \end{pmatrix} \triangledown_d \begin{pmatrix} c_2 \\ d_2 \\ p_2 \end{pmatrix} &= \begin{pmatrix} c_1 + c_2 \\ d_1 + d_2 + (p_1 \star p_2) d \\ p_1 \star p_2 \end{pmatrix}. \end{aligned}$$

Slightly abusing notation, we write $X \triangle_d Y = \left\{ \begin{pmatrix} c_1 \\ d_1 \\ p_1 \end{pmatrix} \triangle_d \begin{pmatrix} c_2 \\ d_2 \\ p_2 \end{pmatrix} \middle| \begin{pmatrix} c_1 \\ d_1 \\ p_1 \end{pmatrix} \in X, \begin{pmatrix} c_2 \\ d_2 \\ p_v \end{pmatrix} \in Y \right\}$ and likewise for $\triangledown_d$. We then have the following result:

Lemma 2. *For an internal node $v \in N \setminus B$ one has*

$$\begin{aligned} &\mathrm{EA}_v(\mathcal{A}_v) \\ &= \begin{cases} \mathrm{EA}_{v_1}(\mathcal{A}_{v_1}) \triangle_{\mathrm{d}(v)} \mathrm{EA}_{v_2}(\mathcal{A}_{v_2}), & \text{if } \gamma(v) = \mathrm{AND}, \\ \mathrm{EA}_{v_1}(\mathcal{A}_{v_1}) \triangledown_{\mathrm{d}(v)} \mathrm{EA}_{v_2}(\mathcal{A}_{v_2}), & \text{if } \gamma(v) = \mathrm{OR}. \end{cases} \end{aligned} \tag{17}$$

Proof. Suppose $\gamma(v) = \mathrm{AND}$. Since T is treelike, one has $B_v = B_{v_1} \cup B_{v_2}$ and $B_{v_1} \cap B_{v_2} = \varnothing$. It follows that we can identify $\mathcal{A}_v = \mathbb{B}^{B_v} = \mathbb{B}^{B_{v_1}} \times \mathbb{B}^{B_{v_2}} = \mathcal{A}_{v_1} \times \mathcal{A}_{v_2}$. Let $\mathbf{x}_1 \in \mathcal{A}_{v_1}$ and $\mathbf{x}_2 \in \mathcal{A}_{v_2}$, and let $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2)$ be the corresponding element of $\mathcal{A}$, i.e., $x_w = x_{i,w}$ when $w \in B_{v_i}$ for $i = 1, 2$. Then the attributes of $\mathbf{x}$ are given by

$$\begin{aligned} \hat{\mathrm{c}}_v(\mathbf{x}) &= \sum_{w \in B_v} x_w\, \mathrm{c}(w) \\ &= \sum_{w \in B_{v_1}} x_{1,w}\, \mathrm{c}(w) + \sum_{w \in B_{v_2}} x_{2,w}\, \mathrm{c}(w) \\ &= \hat{\mathrm{c}}_{v_1}(\mathbf{x}_1) + \hat{\mathrm{c}}_{v_2}(\mathbf{x}_2), \\ \mathrm{PS}_v(\mathbf{x}, v) &= \mathrm{PS}_v(\mathbf{x}, v_1)\, \mathrm{PS}_v(\mathbf{x}, v_2) \\ &= \mathrm{PS}_{v_1}(\mathbf{x}_1, v_1)\, \mathrm{PS}_{v_2}(\mathbf{x}_2, v_2), \\ \hat{\mathrm{d}}_v(\mathbf{x}) &= \sum_{w \in B_v} \mathrm{PS}_v(\mathbf{x}, w)\, \mathrm{d}(w) \\ &= \sum_{w \in B_{v_1}} \mathrm{PS}_{v_1}(\mathbf{x}_1, w)\, \mathrm{d}(w) \end{aligned}$$

$$+ \sum_{w \in B_{v_1}} \mathrm{PS}_{v_2}(\mathbf{x}_2, w)\,\mathrm{d}(w) + \mathrm{PS}_v(\mathbf{x}, v)\,\mathrm{d}(v)$$
$$= \hat{\mathrm{d}}_{v_1}(\mathbf{x}_1) + \hat{\mathrm{d}}_{v_2}(\mathbf{x}_2) + \mathrm{PS}_{v_1}(\mathbf{x}_1, v_1)\,\mathrm{PS}_{v_2}(\mathbf{x}_2, v_2)\,\mathrm{d}(v).$$

We can write this more succinctly as

$$\mathrm{EA}_v(\mathbf{x}) = \mathrm{EA}_{v_1}(\mathbf{x}_1) \triangle_{\mathrm{d}(v)} \mathrm{EA}_{v_2}(\mathbf{x}_2).$$

Ranging over all $\mathbf{x}_1$ and $\mathbf{x}_2$ (and consequently over all $\mathbf{x}$) now proves the lemma. The case that $\gamma(v) = \mathrm{OR}$ is completely analogous. □

Furthermore, for $X \subseteq \texttt{PTrip}$, we define

$$H_U(X) := \left\{ \begin{pmatrix} c \\ d \\ p \end{pmatrix} \in X \middle| c \leq U \right\}.$$

This function, along with the known function $\underline{\min}$, satisfies the following properties:

Lemma 3. *For $X, Y \subseteq \texttt{PTrip}$ and $d \in \mathbb{R}_{\geq 0}$ the following hold:*

$$H_U(\underline{\min}(X)) = \underline{\min}(H_U(X)), \tag{18}$$
$$H_U(X \triangle_d H_U(Y)) = H_U(X \triangle_d Y), \tag{19}$$
$$H_U(X \triangledown_d H_U(Y)) = H_U(X \triangledown_d Y), \tag{20}$$
$$\underline{\min}(X \triangle_d \underline{\min}(Y)) = \underline{\min}(X \triangle_d Y), \tag{21}$$
$$\underline{\min}(X \triangledown_d \underline{\min}(Y)) = \underline{\min}(X \triangledown_d Y). \tag{22}$$

Proof. We tackle these equations one by one, starting with (18). Let $x = \begin{pmatrix} c \\ d \\ p \end{pmatrix} \in H_U(\underline{\min}(X))$; then $x \in \underline{\min}(X)$. Furthermore, $c \leq U$, and so $x \in H_U(X)$. Suppose that $x \notin \underline{\min}(H_U(X))$; then there exists an $x' \in H_U(X)$ such that $x' \sqsubset x$. But this contradicts the fact that $x \in \underline{\min}(X)$, and such x' cannot exist; this proves $H_U(\underline{\min}(X)) \subseteq \underline{\min}(H_U(X))$.

Now let $x = \begin{pmatrix} c \\ d \\ p \end{pmatrix} \in \underline{\min}(H_U(X))$. Suppose $x \notin \underline{\min} X$; then there exists an $x' = \begin{pmatrix} c' \\ d' \\ p' \end{pmatrix}$ such that $x' \sqsubset x$. By definition of $\sqsubset$ this means that $c' \leq c$, and so $x' \in H_U(X)$; but then this contradicts the fact that $x \in \underline{\min}(H_U(X))$. We can conclude that $x \in \underline{\min}(X)$. Since $x \in H_U(X)$, we also know that $c \leq U$, and so $x \in H_U(\underline{\min}(X))$. This proves $H_U(\underline{\min}(X)) \supseteq \underline{\min}(H_U(X))$.

Next, we prove (19) (equation (20) is proven analogously to this and is therefore skipped). Since $H_U(Y) \subseteq Y$ it is clear that $H_U(X \triangle_d H_U(Y)) \subseteq H_U(X \triangle_d Y)$; we now prove the opposite direction. Let $x = \begin{pmatrix} c_1 \\ d_1 \\ p_1 \end{pmatrix} \in X$, $y = \begin{pmatrix} c_2 \\ d_2 \\ p_2 \end{pmatrix} \in Y$ be such that $x \triangle_d y \in H_U(X \triangle_d Y)$. Then $c_1 + c_2 \leq U$, and since $c_1, c_2 \in \mathbb{R}_{\geq 0}$ this implies that $c_2 \in U$. Hence $y \in H_U(Y)$, and this proves $H_U(X \triangle_d H_U(Y)) \supseteq H_U(X \triangle_d Y)$.

Finally, we consider (21) and (22); since they can be proven completely analogous we only consider the former. First, we note that $\triangle_d$ preserves $\sqsubseteq$ in the following sense: if $y \sqsubseteq y'$, then $x \triangle_d y \sqsubseteq x \triangle_d y'$ for all $x, y, y' \in \texttt{PTrip}$. Now let $x \in X$ and $y \in \underline{\min}(Y)$ be such that $x \triangle_d y \in \underline{\min}(X \triangle_d \underline{\min}(Y))$. Suppose that $x \triangle_d y \notin \min(X \triangle_d Y)$; then there exist $x' \in X$, $y' \in Y$ such that $x' \triangle_d y' \sqsubset x \triangle_d y$. Let $y'' \in \underline{\min}(Y)$ be such that $y'' \sqsubseteq y'$; then

$$X \triangle_d \underline{\min}(Y) \ni x' \triangle_d y'' \sqsubseteq x' \triangle_d y' \sqsubset x \triangle_d y,$$

which contradicts the fact that $x \triangle_d y \in \underline{\min}(X \triangle_d \underline{\min}(Y))$. Hence $x \triangle_d y \in \underline{\min}(X \triangle_d Y)$, and this proves $\underline{\min}(X \triangle_d \underline{\min}(Y)) \subseteq \underline{\min}(X \triangle_d Y)$.

Now let $x \in X$ and $y \in Y$ such that $x \triangle_d y \in \min(X \triangle_d Y)$. Let $y' \in \underline{\min}(Y)$ such that $y' \sqsubseteq y$. Then $x \triangle_d y' \sqsubseteq x \triangle_d y$. Since the latter is assumed to be minimal in $X \triangle_d Y$, it follows that $x \triangle_d y' = x \triangle_d y$. In particular, $x \triangle_d y \in X \triangle_d \underline{\min}(Y)$. Since $x \triangle_d y$ is minimal in $X \triangle_d Y$, it is certainly minimal in the smaller set $X \triangle_d \underline{\min}(Y)$. This proves $\underline{\min}(X \triangle_d \underline{\min}(Y)) \supseteq \underline{\min}(X \triangle_d Y)$. □

Proof of Theorem 10. We prove this via induction on v. From (11) it is clear that it holds for BASs. Now suppose $v = \mathrm{AND}(v_1, v_2)$, and that the statement holds for v_1 and v_2. We can then write (13) as

$$\mathcal{C}^{\mathrm{P}}_U(v) = \underline{\min}(H_U(\mathcal{C}^{\mathrm{P}}_U(v_1) \triangle_d \mathcal{C}^{\mathrm{P}}_U(v_2)))$$

and what we need to prove as

$$\mathcal{C}^{\mathrm{P}}_U(v) \stackrel{?}{=} \underline{\min}(H_U(\mathrm{EA}_v(\mathcal{A}_v))).$$

Using the induction hypothesis and Lemmas 2 and 3, we find

$$\begin{aligned}
&\mathcal{C}^{\mathrm{P}}_U(v) \\
&= \underline{\min} H_U[\mathcal{C}^{\mathrm{P}}_U(v_2) \triangle_d \mathcal{C}^{\mathrm{P}}_U(v_2)] \\
&\stackrel{\text{IH}}{=} \underline{\min} H_U\left[\underline{\min} H_U(\mathrm{EA}_{v_1}(\mathcal{A}_{v_1})) \triangle_d \underline{\min} H_U(\mathrm{EA}_{v_2}(\mathcal{A}_{v_2}))\right] \\
&\stackrel{(18)}{=} H_U \underline{\min}\left[\underline{\min} H_U(\mathrm{EA}_{v_1}(\mathcal{A}_{v_1})) \triangle_d \underline{\min} H_U(\mathrm{EA}_{v_2}(\mathcal{A}_{v_2}))\right] \\
&\stackrel{(21)}{=} H_U \underline{\min}\left[H_U(\mathrm{EA}_{v_1}(\mathcal{A}_{v_1})) \triangle_d H_U(\mathrm{EA}_{v_2}(\mathcal{A}_{v_2}))\right] \\
&\stackrel{(18)}{=} \underline{\min} H_U\left[H_U(\mathrm{EA}_{v_1}(\mathcal{A}_{v_1})) \triangle_d H_U(\mathrm{EA}_{v_2}(\mathcal{A}_{v_2}))\right] \\
&\stackrel{(19)}{=} \underline{\min} H_U\left[\mathrm{EA}_{v_1}(\mathcal{A}_{v_1}) \triangle_d \mathrm{EA}_{v_2}(\mathcal{A}_{v_2})\right] \\
&\stackrel{(17)}{=} \underline{\min} H_U\left[\mathrm{EA}_v(\mathcal{A}_v)\right],
\end{aligned}$$

which is what needed to be shown. The case that $v = \mathrm{OR}(v_1, v_2)$ is completely analogous. □

We are now in a position to prove Theorems 3, 4, 8 and 9. Note that the deterministic scenario can be reduced to the probabilistic scenario, by taking $\mathrm{p}(v) = 1$ for all $v \in B$; this ensures that $\hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x}) = \hat{\mathrm{d}}(\mathbf{x})$ for all $\mathbf{x}$. It also causes the definition of $\mathcal{C}^{\mathrm{D}}_U(v)$ to coincide with that of $\mathcal{C}^{\mathrm{P}}_U(v)$ for all v. Therefore it suffices to prove 8 and 9.

Theorem 8. *The solution to EDgC is given by* $\max\left\{ d \in \mathbb{R}_{\geq 0} \middle| \begin{pmatrix} c \\ d \\ p \end{pmatrix} \in \mathcal{C}^{\mathrm{P}}_U(\mathrm{R}_T) \right\}$.

Proof. Let $d_{\mathrm{E,opt}}$ be the solution to EDgC, i.e., there exists an $\mathbf{x}_0 = \mathcal{A}$ such that $c_0 := \hat{\mathrm{c}}(\mathbf{x}) \leq U$ and $d_{\mathrm{E,opt}} = \hat{\mathrm{d}}_{\mathrm{E}}(\mathbf{x})$, and $d_{\mathrm{E,opt}}$ is maximal under this constraint. Let $x_0 = \mathrm{EA}(\mathbf{x})$; then certainly $x_0 \in H_U(\mathrm{EA}(\mathcal{A}))$. Let $x' = \begin{pmatrix} c' \\ d' \\ p' \end{pmatrix} \in \underline{\min} H_U(\mathrm{EA}(\mathcal{A}))$ with $x' \sqsubset x$. Then $c' \leq c_0 \leq U$ and $d' \geq d_{\mathrm{E,opt}}$; since $d_{\mathrm{E,opt}}$ was assumed to be optimal given $c_0 \leq U$, we conclude that $d = d'$. It follows that

$$d_{\mathrm{E,opt}} = \max\left\{ d \middle| \exists c, p.\ \begin{pmatrix} c \\ d \\ p \end{pmatrix} \in \underline{\min} H_U(\mathrm{EA}(\mathcal{A})) \right\},$$

$$= \max\left\{ d \middle| \exists c, p.\, \begin{pmatrix} c \\ d \\ p \end{pmatrix} \in \mathcal{C}_U^{\mathrm{P}}(\mathrm{R}_T) \right\},$$

where the second equation follows from Theorem 10. This is what was needed to be proven. □

Theorem 9. *The solution to CEDPF is given by* $\min \pi(\mathcal{C}_\infty^{\mathrm{P}}(\mathrm{R}_T))$*, where* $\pi\colon \texttt{PTrip} \to \mathbb{R}^2_{\geq 0}$ *is the projection map onto the first two coefficients.*

Proof. We claim that $\underline{\min} \circ \pi \circ \underline{\min} = \underline{\min} \circ \pi$ as maps $\mathcal{P}(\texttt{PTrip}) \to \mathcal{P}(\mathbb{R}^2_{\geq 0})$. To show this, let $X \subseteq \texttt{PTrip}$, and let $x \in \underline{\min}(X)$ be such that $\pi(x) \in \underline{\min}(\pi(\underline{\min}(X)))$. Then $\pi(x) \in \pi(X)$; suppose $\pi(x)$ is not minimal in $\pi(X)$, and there exists an $x' \in X$ such that $\pi(x') \sqsubset \pi(x)$. Let $x'' \in \underline{\min} X$ be such that $x'' \sqsubseteq x'$. Since π is order-preserving, we have $\pi(x'') \sqsubseteq \pi(x') \sqsubset \pi(x)$. However, this contradicts the fact that $\pi(x)$ is minimal in $\pi(\underline{\min}(X))$. Hence our assumption that $\pi(x)$ is not minimal in $\pi(X)$ does not hold, and we can conclude $\underline{\min}(\pi(\underline{\min}(X))) \subseteq \underline{\min}(\pi(X))$.

Now let $x \in X$ be such that $\pi(x) \in \underline{\min}(\pi(X))$. Let $x' \in \underline{\min}(X)$ be such that $x' \sqsubseteq x$. Since π is order-preserving, we find $\pi(x') \sqsubseteq \pi(x)$, but since $\pi(x)$ is minimal, this is an equality; hence $\pi(x) = \pi(x') \in \pi(\underline{\min}(X))$. Furthermore, since $\pi(x)$ is minimal in $\pi(X)$, it is certainly minimal in $\pi(\underline{\min}(X))$. We conclude $\underline{\min}(\pi(\underline{\min}(X))) \supseteq \underline{\min}(\pi(X))$, which proves the claim.

Let us now return to the proof of 9. Note that H_∞ is just the identity on $\mathcal{P}(\texttt{PTrip})$, and that $\pi \circ \mathrm{EA} = \binom{\hat{\mathrm{c}}}{\hat{\mathrm{d}}_{\mathrm{E}}}$. It follows that

$$\begin{aligned}
\underline{\min}\pi\mathcal{C}_\infty^{\mathrm{PT}}(\mathrm{R}_T) &= \underline{\min}\pi\underline{\min}H_\infty\,\mathrm{EA}(\mathcal{A}) \\
&= \underline{\min}\pi\,\mathrm{EA}(\mathcal{A}) \\
&= \underline{\min}\begin{pmatrix} \hat{\mathrm{c}} \\ \hat{\mathrm{d}}_{\mathrm{E}} \end{pmatrix}(\mathcal{A}) \\
&= \mathrm{EPF}(T). \qquad \square
\end{aligned}$$