Pareto Efficient Solutions of Attack-Defence Trees

Zaruhi Aslanyan and Flemming Nielson

DTU Compute, Technical University of Denmark, Denmark
{zaas,fnie}@dtu.dk

Abstract. Attack-defence trees are a promising approach for representing threat scenarios and possible countermeasures in a concise and intuitive manner. An attack-defence tree describes the interaction between an attacker and a defender, and is evaluated by assigning parameters to the nodes, such as probability or cost of attacks and defences. In case of multiple parameters most analytical methods optimise one parameter at a time, e.g., minimise cost or maximise probability of an attack. Such methods may lead to sub-optimal solutions when optimising conflicting parameters, e.g., minimising cost while maximising probability.

In order to tackle this challenge, we devise automated techniques that optimise all parameters at once. Moreover, in the case of conflicting parameters our techniques compute the set of all optimal solutions, defined in terms of Pareto efficiency. The developments are carried out on a new and general formalism for attack-defence trees.

Keywords: Attack-defence trees, attack trees, countermeasures, security assessment, Pareto efficiency, multiple criteria.

1 Introduction

Nowadays fast growing technologies influence our everyday life and increase our productivity. Unfortunately, we witness with alarming frequency that they also increase the risk of physical and cyber attacks to a wide range of targets, from personal devices to systems of public concern. The growing number of threats demands a thorough investigation of the security properties of a system when deployed in a given environment. To this end, various formal graphical models have been studied.

Fault trees, introduced in the early 1980's, are one of the first and most prominent graphical representations for analysing the safety of a system. They represent a system failure in terms of the failure of its components [1].

Fault trees inspired a similar approach to security. In 1991, Weiss used trees in security analysis and presented threat-logic trees as a graphical attack-modelling technique [2]. Later, in 1999, Schneier introduced *attack trees* as a tool to evaluate the security of complex systems in a structured, hierarchical way. Attack trees allow to analyse the possible attack scenarios and reason about the security of

R. Focardi and A. Myers (Eds.): POST 2015, LNCS 9036, pp. 95–114, 2015.
DOI: 10.1007/978-3-662-46666-7_6

the whole system in a formal, methodical way, by splitting a complex goal into sub-goals and basic attacks [3]. However, attack trees evaluate only the attacker's behaviour and do not consider possible defences undertaken to avoid the attacks.

To overcome this limitation, further extensions of attack trees for capturing the defender's behaviour have been studied. Such extensions have been explored in several dimensions. Some enrich an attack tree model by integrating appropriate defender's actions against specific attacks only at leaf level [4], while others combine attack and defence models and present a methodology to compute specific parameters [5]. Kordy [6] introduced a more general tool, called attack-defence trees, to represent the interaction between an attacker and a defender.

Attack-defence trees are extensions of attack trees with countermeasures. They illustrate in a graphical way the possible actions an attacker can perform in order to attain a given goal, and the feasible countermeasures a defender can undertake to counter such actions. Attack-defence trees are used for analysing attack-defence scenarios. Analyses are performed by considering specific aspects or properties of the scenario. The evaluation assigns values to the parameters of the leaves and the tree is traversed from the leaves to the root.

Most analyses of attack-defence trees focus on one specific aspect of the system, such as feasibility or cost of an attack or a defence. They do not consider multiple parameters and the subsequent need for optimising all of them at once. Moreover, optimisation of multiple parameters might lead to incomparable values, in which case such methods may result in sub-optimal solutions. However, in many real-life scenarios a single parameter might not be adequate for the analysis of complex attack-defence scenarios.

In order to address multi-parameter optimisation of attack-defence trees, we present evaluation techniques that characterise the leaves of a tree with more than one parameter, such as the success probability *and* the cost of an attack. Our techniques compute different aspects of the scenario and handle multiple parameters, thus optimising all of them at once. Multi-parameter optimisation becomes necessary in case of conflicting parameters, as there is no single best solution but rather a set of optimal solutions. We handle conflicting parameters by computing the set of efficient solutions, defined in terms of Pareto efficiency. Thus, Pareto efficiency handles the multi-criteria optimisation problem, as well as parameters with incomparable values.

Our developments are performed on a new language-based formalism for attack-defence trees. Furthermore, we study the issue in both Boolean and probabilistic settings. For each such setting, we first consider the problem of feasibility of the attack or the defence, and then we extend our techniques to compute optimal attacks or defences in presence of multiple costs. Moreover, for each case, we first define the solution considering all possible player interactions, obtaining a natural but exponential characterisation. Then, we improve dramatically on the complexity devising an algorithmic evaluation that is linear in the size of the tree and yet sound for an expressive sub-class of models.

Organisation of the paper. In Sect. 2 we introduce our formalism for attack-defence trees and provide evaluation techniques for feasibility queries. Sect. 3 extends the model with a single cost and presents evaluation techniques for computing minimum cost. We extend the single cost model to multiple costs in Sect. 4. The results of evaluation are discussed on a case study for a Radio-Frequency Identification system managing goods in a warehouse. We describe related work in Sect. 5 and conclude in Sect. 6.

2 Formal Model of Attack-Defence Trees

In the following, we present our formalism for attack-defence trees. We start by defining the syntax and the terminology used throughout the paper. Then, we describe the evaluation techniques for investigating the feasibility of attacks and defences both in Boolean and probabilistic settings. The Boolean case is thoroughly explained in Sect. 2.2. The developments are generalised to the probabilistic setting in Sect. 2.3.

2.1 Syntax and Well-Formednes

Syntax and intended semantics. We construe an attack-defence tree as an interaction between two players (denoted by τ), the proponent ($\tau = p$) and the opponent ($\tau = o$), in the wake of [6]. A player can be either an attacker or a defender. We associate the proponent with the player at the root, and the opponent with the opposite player. Each player has an associated goal, such as minimising or maximising the overall probability of an attack or a defence.

The root of the tree represents the main goal of an attack-defence scenario for a given player τ. The leaves represent the basic actions that a player can perform to achieve his/her goal. The internal nodes show how those actions can be combined. In order to simplifying the technical developments, we assume that the players' actions are independent.

The abstract syntax of an attack-defence tree t is presented in Table 1. A tree is either a leaf or the application of a tree operator to one or two sub-trees.

Based on the player type, a leaf a is either a basic action of the proponent or of the opponent. We denote the set of proponent's and opponent's basic actions by Act_p and Act_o, respectively. We assume that these two sets are disjoint, $Act_p \cap Act_o = \emptyset$. We denote by Act the set of all basic actions, $Act = Act_p \cup Act_o$.

There are two special types of leaves; $\&_{\mathrm{true}}$ represents a trivially-successful action, and $\&_{\mathrm{false}}$ represents a trivially-failed action.

As standard in the literature, tree operators include conjunction and disjunction, while we introduce negation and a novel construct for player alternation. The conjunction operator $t = \&_{\wedge}(t_1, t_2)$ requires that the goals of t_1, t_2 are achieved in order for the goal of t to be achieved. The disjunction operator $t = \&_{\vee}(t_1, t_2)$ requires that the goal of at least one sub-tree is achieved in order for the goal of t to be achieved.

The negation operator $t = \&_{\neg}(t')$ requires that the goal of the sub-tree t' is not achieved in order for the goal of t to be achieved. This operator negates the goal

Table 1. The type system for defining well-formed trees

$$t ::= a \mid \&_\wedge(t_1, t_2) \mid \&_\vee(t_1, t_2) \mid \&_\neg(t) \mid \&_\sim(t) \mid \&_{\text{true}} \mid \&_{\text{false}}$$

$$\vdash a : p \text{ if } a \in Act_p \qquad \vdash a : o \text{ if } a \in Act_o$$

$$\frac{\vdash t_1 : \tau \quad \vdash t_2 : \tau}{\vdash \&_\wedge(t_1, t_2) : \tau} \qquad \frac{\vdash t_1 : \tau \quad \vdash t_2 : \tau}{\vdash \&_\vee(t_1, t_2) : \tau} \qquad \frac{\vdash t : \tau}{\vdash \&_\neg(t) : \tau} \qquad \frac{\vdash t : \tau}{\vdash \&_\sim(t) : \tau'} \; \tau' = \tau^{-1}$$

of t' and leaves the player unchanged. Such an operator allows to analyse a wider range of attack and defence scenarios, including the cases of unrecoverable and conflicting actions, thus making trees more flexible and expressive. For instance, cutting a communication wire might be unrecoverable, and after having cut a wire a player might not be able to communicate with a given device.

The changing player operator $t = \&_\sim(t')$ changes the goal of t' by changing the type of the player. Note that in this case the goal belongs to the opposite player. For instance, if t' belongs to an attacker with the corresponding goal (e.g., minimising), then the tree t belongs to a defender with the corresponding goal (e.g., maximising). Thus, the changing player operator flips the player from an attacker to a defender and vice versa, as highlighted by the side-condition of the corresponding rule, where $p^{-1} = o$ and $o^{-1} = p$.

The syntax of Table 1 is overly liberal for it does not associate players to nodes. The simple type system, showed in the second section of the table, enforces such association defining a well-formedness condition. We denote by $Tree_\tau$ the set of well-formed attack-defence trees whose root belongs to a player τ. Based on the type of the player, we have $Tree_p$ when τ is the proponent and $Tree_o$ when τ is the opponent and $Tree = Tree_p \cup Tree_o$. In the following, we will refer to them as attack-defence trees or simply trees. Moreover, we introduce below the notion of polarity consistency, to be exploited in the technical developments.

Polarity-Consistent Tree. We say that an action a occurs negatively in a tree, if a is under an odd number of negations. Otherwise, we say that an action a occurs positively. Such *polarities* are denoted with the symbols $-$ and $+$, respectively.

Definition 1. *An attack-defence tree t is polarity-consistent iff there is no action that occurs both positively and negatively in t.*

A sufficient (but not necessary) condition for polarity-consistency is that all actions are "uniformly good" or "uniformly bad" for the proponent. If t is a polarity-consistent tree, then the polarity of each action is uniquely determined.

Running Example. Let us introduce an example that we will develop throughout the paper. We consider a fragment of a Radio-Frequency Identification (RFID) system managing goods in a warehouse, studied in [7]. Particularly, we consider an attacker (proponent) whose goal consists in removing the RFID tags from

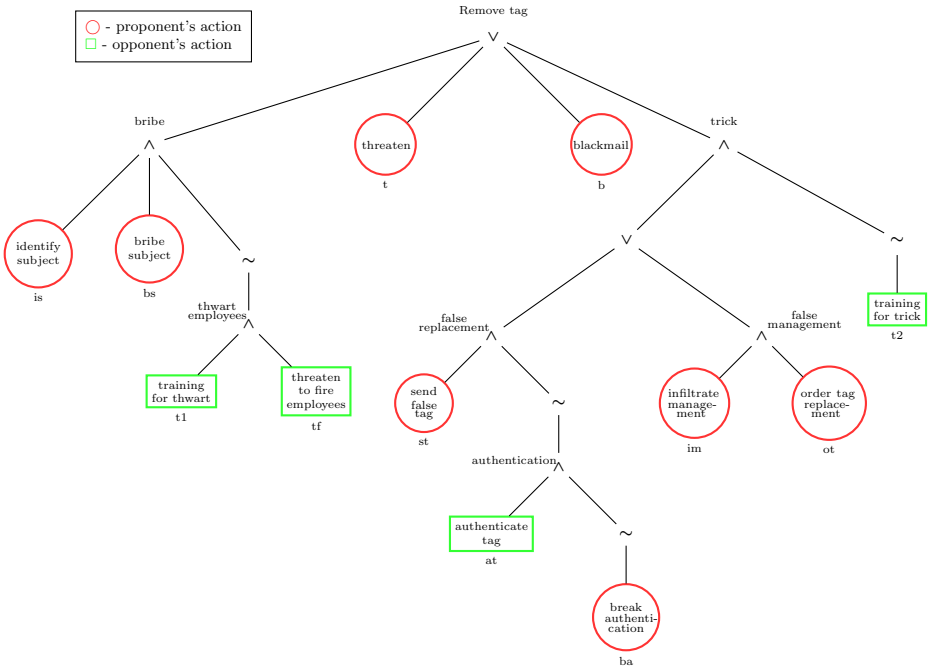

Fig. 1. Attack-defence tree for removing tag

goods with the help of an insider. In order to enable a direct comparison with the evaluation techniques in the literature, our attack-defence tree does not contain negation. We will see, however, that as far as the calculation is concerned, negation would be treated similarly to the changing player operator.

In order to attain the goal, the attacker can "bribe", "threaten", "blackmail", or "trick" the insider. For bribing a person the attacker has to "identify a corruptible subject" and "bribe the subject". The defender (opponent) can protect against bribery by "thwarting employees", which can be done by "training for security" and by "threatening to fire the employees".

In case the attacker decides instead to "trick" the insider by placing a fake tag, they can either "send false replacement tags" or give a "false management order" to do it. The latter can be done by "infiltrating the management" and "ordering replacement tags". To fight such attacks, the defender can provide the employees with "training for trick".

The corresponding attack-defence tree is given in Figure 1. We decorate internal nodes with labels to keep track of sub-goals, hence making the tree more informative and human-readable. We label the leaves to refer to them easily.

The t, displayed in Figure 1, is represented by the following syntactic term:

$$t = \&_\vee(\&_\wedge(\text{is}, \&_\wedge(\text{bs}, \&_\sim(\&_\wedge(\text{t1},\text{tf})))),$$
$$\&_\vee(\text{t}, \&_\vee(\text{b}, \&_\wedge(\&_\vee(\&_\wedge(\text{st}, \&_\sim(\&_\wedge(\text{at}, \&_\sim(\text{ba}))))), \&_\wedge(\text{im},\text{ot})), \&_\sim(\text{t2})))))$$

2.2 Semantics in the Boolean Case

As mentioned above, we construe an attack-defence tree as an interaction between the proponent and the opponent. In the Boolean setting the investigation of the feasibility of a scenario is related to answering questions such as "Is the scenario satisfiable?" or "Is there an always-successful attack/defence?".

In this setting, we associate with each basic action a value from the Boolean set $\mathbb{B}$, where *true* corresponds to performing and *false* corresponds to not performing the action. We consider $\mathbb{B}$ to be ordered such that $\max\{tt, ff\} = tt$ and $\min\{tt, ff\} = ff$.

We define a Boolean assignment of basic actions for a given player τ as follows: a Boolean assignment m_τ is an arbitrary function that assigns a value $b \in \mathbb{B}$ to each basic action $a \in Act_\tau$; $m_\tau : Act_\tau \to \mathbb{B}$. Thus, the Boolean assignment m is a pair (m_p, m_o), but we allow to write $m(a)$ as a shorthand for $m_p(a)$ when $a \in Act_p$ and $m_o(a)$ when $a \in Act_o$. We say that the main goal described by a tree succeeds if the Boolean assignment evaluates the tree to *true*.

For evaluating the feasibility of an attack-defence tree, we present two evaluation techniques, termed semantic and algorithmic evaluations respectively.

The semantic evaluation $M(t)$ of an attack-defence tree $t \in Tree_p$ is presented in Table 2. The evaluation analyses the tree t by considering all possible Boolean assignments of values to the basic actions of t. It computes the pair of minimum and maximum success values of the proponent. If the proponent is an attacker, then it computes the minimum and the maximum values of an attack. Otherwise, it computes the minimum and maximum values of a defence. We observe that if the main goal of the scenario (represented by the root of the tree) is successful for the proponent, then it is not successful for the opponent. Similarly, if the proponent wants to maximise the success of the main goal, then the opponent wants to minimise it. Thus, the players have opposite goals. We integrate this consideration into our technique by minimising the value of t over all opponent's Boolean assignments m_o, and then maximising it over all proponent's Boolean assignments m_p. This is illustrated in the second component of $M(t)$ in Table 2, which computes the maximum success value of the proponent. The first component of $M(t)$ computes the minimum success value of the proponent. Therefore, the computation maximises the value over all m_o's and then minimises it over all m_p's.

The analysis $\mathcal{B}[\![t]\!]m$ of the tree t, displayed in the second part of the Table 2, is performed recursively on the structure of t. Observe that even though the negation and changing player operators work on the same assignment m, the former only changes the polarity of a tree whereas the latter changes the optimisation objective by swapping the players.

The result of the semantic evaluation, when the proponent is an attacker, is interpreted as follows.

- If both the minimum and the maximum values of t are *false* (ff, ff), then the system is always secure despite the attacker's actions.

Table 2. The Boolean semantic evaluation of an attack-defence tree

$$M(t) = (\min\{\max\{\mathcal{B}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\} \mid m_p \text{ Boolean assignment}\},\ \max\{\min\{\mathcal{B}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\} \mid m_p \text{ Boolean assignment}\})$$

$$\begin{aligned}
\mathcal{B}[\![a]\!]m &= m(a)\\
\mathcal{B}[\![\&_\wedge(t_1,t_2)]\!]m &= \mathcal{B}[\![t_1]\!]m \wedge \mathcal{B}[\![t_2]\!]m\\
\mathcal{B}[\![\&_\vee(t_1,t_2)]\!]m &= \mathcal{B}[\![t_1]\!]m \vee \mathcal{B}[\![t_2]\!]m\\
\mathcal{B}[\![\&_\neg(t)]\!]m &= \neg\mathcal{B}[\![t]\!]m\\
\mathcal{B}[\![\&_\sim(t)]\!]m &= \neg\mathcal{B}[\![t]\!]m\\
\mathcal{B}[\![\&_{\mathrm{true}}]\!]m &= tt\\
\mathcal{B}[\![\&_{\mathrm{false}}]\!]m &= ff
\end{aligned}$$

- If the minimum value of t is *false* and the maximum value of t is *true* (ff, tt), then the system is vulnerable. In other words, there exist actions (a Boolean assignment m) such that an attack on the system is feasible.
- If both the minimum and the maximum values of t are *true* (tt, tt), then the system is flawed. In other words, despite the attacker's actions (for all Boolean assignments m) an attack on the system is always successful.

The result is interpreted likewise, when the proponent is a defender.

The semantic evaluation characterises the analysis in a natural way, for it explicitly considers all the interactions interwoven in a tree in terms of assignments to the leaves. Nonetheless, it gives rise to an exponential computation already in the Boolean case, the satisfiability problem being NP-complete. Therefore, evaluation techniques that enjoy a lower complexity are needed. In particular, we face the problem of defining those restrictions on attack-defence trees under which the more efficient methods are sound with respect to the semantic evaluation, our gold standard.

The algorithmic evaluation $INT(t)$ of an attack-defence tree $t \in Tree_p$ is presented in Table 3. Similarly to the semantic evaluation, it computes the pair of minimum and maximum success values of the proponent. It considers the values of basic actions and propagates them up to the root. The propagation is performed according to the rules given in Table 3. The first rule assigns the minimum and the maximum success values to the actions based on the player type. Observe that, as the players have opposite goals, the success values are also opposite. The next four rules define the computation for operators. Conjunction and disjunction are treated in the standard way, hence let us focus on the negation and changing player operators. Both operators change the goal of the player. The negation operator negates the goal without changing the player, while the changing player operator changes the goal by changing the player. Thus, in both rules we first swap the minimum and maximum values, and then apply negation. The last two rules are independent from the players and represent always successful and failed actions.

The semantic and algorithmic evaluations might lead to different results, as we can see by considering the attack-defence tree $t = \&_\wedge(a, \&_\neg(a))$, where $a \in Act_p$.

Table 3. The Boolean algorithmic evaluation of an attack-defence tree

$$
\begin{array}{ll}
INT(a) & = \begin{cases} (\mathit{ff}, tt) & \text{if } a \in Act_p \\ (tt, \mathit{ff}) & \text{if } a \in Act_o \end{cases} \\
INT(\&_\wedge(t_1, t_2)) & = let\ (min_i, max_i) = INT(t_i), i \in \{1, 2\} \\
 & \quad in\ (min_1 \wedge min_2, max_1 \wedge max_2) \\
INT(\&_\vee(t_1, t_2)) & = let\ (min_i, max_i) = INT(t_i), i \in \{1, 2\} \\
 & \quad in\ (min_1 \vee min_2, max_1 \vee max_2) \\
INT(\&_\neg(t)) & = let\ (min, max) = INT(t) \\
 & \quad in\ (\neg max, \neg min) \\
INT(\&_\sim(t)) & = let\ (min, max) = INT(t) \\
 & \quad in\ (\neg max, \neg min) \\
INT(\&_{\text{true}}) & = (tt, tt) \\
INT(\&_{\text{false}}) & = (\mathit{ff}, \mathit{ff})
\end{array}
$$

The result of the semantic evaluation is $M(t) = (\mathit{ff}, \mathit{ff})$, while the results of the algorithmic evaluation is $INT(t) = (\mathit{ff}, tt)$. However, observe that t is not polarity-consistent. As a matter of fact, if we restrict to polarity-consistent trees, then the two evaluations are equivalent.

Theorem 1. *If $t \in Tree$ is a polarity-consistent tree, then $M(t) = INT(t)$.*

The semantic evaluation considers all possible Boolean assignments m, thus being exponential in the size of t. The implementation of the algorithmic evaluation consists in a bottom-up traversal of t, and thus is linear in the size of the tree. Therefore, in case of polarity-consistent trees, our method offers a dramatic improvement in performance hence in scalability. This is in line with the development of, e.g., [8].

2.3 Semantics in the Probabilistic Case

The probabilistic setting generalises the Boolean one. In the following, we give a brief explanation of the setting and the evaluations, focusing on the novelties and omitting redundant details.

In the probabilistic setting, we consider the interval $[0, 1]$, where 1 corresponds to success and 0 corresponds to failure. The questions tackled in the probabilistic setting are, e.g., "What is the maximum probability of an attack?" or "How vulnerable is the system to the attack?".

In the remainder of the paper, we shall restrict our investigation to linear trees, inspired by Girard's linear logic [9] and defined as follows:

Definition 2. *An attack-defence tree t is linear iff no action occurs twice in t.*

The notion of linearity is stronger than polarity-consistency, as the latter does not forbid to have multiple occurrences of the same action with the same polarity. Let us focus on the occurrences of an action a in the following polarity-consistent tree: $t = \&_\vee(\&_\wedge(a, b), \&_\wedge(a, c))$. In the tree t the action a is performed once but

Table 4. The probabilistic semantic evaluation of an attack-defence tree

$$M(t) = (\min\{\max\{\mathcal{P}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\} \mid m_p \text{ Boolean assignment}\},\ \max\{\min\{\mathcal{P}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\} \mid m_p \text{ Boolean assignment}\})$$

$$\begin{aligned}
\mathcal{P}[\![a]\!]m &= \begin{cases} M_2(a) & \text{if } m(a) = tt \\ M_1(a) & \text{if } m(a) = ff \end{cases} \\
\mathcal{P}[\![\&_\wedge(t_1, t_2)]\!]m &= \mathcal{P}[\![t_1]\!]m \cdot \mathcal{P}[\![t_2]\!]m \\
\mathcal{P}[\![\&_\vee(t_1, t_2)]\!]m &= 1 - (1 - \mathcal{P}[\![t_1]\!]m) \cdot (1 - \mathcal{P}[\![t_2]\!]m) \\
\mathcal{P}[\![\&_\neg(t)]\!]m &= 1 - \mathcal{P}[\![t]\!]m \\
\mathcal{P}[\![\&_\sim(t)]\!]m &= 1 - \mathcal{P}[\![t]\!]m \\
\mathcal{P}[\![\&_{\text{true}}]\!]m &= 1 \\
\mathcal{P}[\![\&_{\text{false}}]\!]m &= 0
\end{aligned}$$

occurs more than once, and the success of each sub-tree containing a depends on the actions b, c. However, observe that in t the actions of sub-trees $\&_\wedge(a, b)$ and $\&_\wedge(a, c)$ are not independent. The assumption of linearity ensures independence of actions, thereby guaranteeing the soundness of the computations explained below.

We assume that each basic action $a \in Act$ has two associated success probabilities; success probability $M_1(a)$ in case of not performing a, and success probability $M_2(a)$ in case of performing a, such that $M_1(a) < M_2(a)$. For instance, an attacker might succeed to disable a security camera with a given probability M_2, or the security camera might be disabled due to some external conditions with a given probability M_1, which for the attacker will be the probability of succeeding without performing the action. We consider the Boolean assignment m_τ as defined in the previous section, $m_\tau : Act_\tau \to \mathbb{B}$, and assume that an action a has a probability of success $M_1(a)$ if $m(a)$ is *false* and has a probability of success $M_2(a)$ if $m(a)$ is *true*. Choosing $M_1(a) = 0$ and $M_2(a) = 1$ coincides with the Boolean case.

The evaluation of attack-defence trees in the probabilistic setting follows the development for the Boolean setting: first, we characterise the solution to our problem in a top-down fashion, and then we investigate what limitations on the model allow to devise an algorithmic approach with lower complexity.

The semantic evaluation $M(t)$ of an attack-defence tree $t \in Tree_p$ is illustrated in Table 4. It computes the minimum and the maximum success probability of a scenario by analysing the tree t over all Boolean assignments from which the probability values are inferred. Observe that also here the players have opposite goals, e.g., the proponent wants to maximise the overall probability of success, while the opponent wants to minimise it.

The result of the computation, when the proponent is an attacker, is interpreted as follows. The maximum success probability p shows the existence of an attack with probability p. In this case, we say that the system is p-vulnerable.

The algorithmic evaluation $INT(t)$ of an attack-defence tree $t \in Tree_p$ is given in Table 5. It traverses the tree from the leaves to the root and propagates

Table 5. The probabilistic algorithmic evaluation of an attack-defence tree

$$
\begin{array}{ll}
INT(a) & = \begin{cases} (M_1(a), M_2(a)) & \text{if } a \in Act_p \\ (M_2(a), M_1(a)) & \text{if } a \in Act_o \end{cases} \\
INT(\&_\wedge(t_1, t_2)) & = let\ (min_i, max_i) = INT(t_i), i \in \{1, 2\} \\
 & \quad in\ (min_1 \cdot min_2, max_1 \cdot max_2) \\
INT(\&_\vee(t_1, t_2)) & = let\ (min_i, max_i) = INT(t_i), i \in \{1, 2\} \\
 & \quad in\ (1 - (1 - min_1) \cdot (1 - min_2), 1 - (1 - max_1) \cdot (1 - max_2)) \\
INT(\&_\neg(t)) & = let\ (min, max) = INT(t) \\
 & \quad in\ (1 - max, 1 - min) \\
INT(\&_\sim(t)) & = let\ (min, max) = INT(t) \\
 & \quad in\ (1 - max, 1 - min) \\
INT(\&_{\text{true}}) & = (1, 1) \\
INT(\&_{\text{false}}) & = (0, 0)
\end{array}
$$

Table 6. The values of probability and cost for the basic actions of the example

Label	**Name of the Node**	M_1	M_2	c
is	identify subject	0.2	0.8	80
bs	bribe subject	0	0.7	100
t1	training for thwart	0.1	0.3	0
tf	threaten to fire employees	0.1	0.4	0
t	threaten	0	0.7	160
b	blackmail	0	0.7	150
st	send false tag	0	0.5	50
at	authenticate tag	0.1	0.7	0
ba	break authentication	0.1	0.6	85
im	infiltrate management	0	0.5	70
ot	order tag replacement	0	0.6	0
t2	training for trick	0.1	0.4	0

the values of the basic actions. Similarly to the Boolean case, as the negation and changing player operators change the goal of the player, we first swap the minimum and maximum values before applying negation.

The restriction of linear trees is adequate for showing the equivalence of the two evaluations.

Theorem 2. *If $t \in Tree$ is a linear tree, then $M(t) = INT(t)$.*

Hence, in the probabilistic setting linearity allows to scale from an exponential to a linear complexity.

Running Example. Consider the attack-defence tree t presented in Figure 1. Observe that t is a linear tree, thus we can apply the algorithmic evaluation for computing the maximum probability of success at the root. Table 6 lists possible probability values for basic actions (the last column is for later reference). Providing a realistic estimate is a research topic in itself and falls outside the scope of this work. Following the algorithmic computation, at the root we obtain $INT(t) = (0, 0.97)$, that is, the system is 0.97 vulnerable.

3 Attack-Defence Trees with Cost

In this section we extend our evaluation techniques by considering a single cost for basic actions. We evaluate the minimum cost of an attack or defence in the Boolean and probabilistic settings. Similarly to the previous section, the Boolean case is thoroughly explained in Sect. 3.1. The developments are generalised to the probabilistic setting in Sect. 3.2.

3.1 Cost in the Boolean Case

In the following, we extend the model of attack-defence trees described in Sect. 2.2 with a single cost for basic actions. Therefore, each basic action is associated with a pair of Boolean and cost values.

When we consider costs, we can focus on questions such as "What is the minimum cost of an attack?" or "How much does it cost to protect a system in a given scenario?". Observe that such questions are player-dependent, meaning that the model is evaluated from a given player's perspective. Since we assumed that the basic actions are independent, we need to consider one player's values only. For instance, for computing the minimum cost of an attack we need only the cost of the attacker's actions and do not require the cost of the defender's actions. Thus, in our evaluation techniques we consider only the cost of the proponent's actions, and do not consider the cost of the opponent's actions.

In the following we consider the set $D = \mathbb{B} \times \mathbb{R}_{\geq 0}$. In order to link the cost parameter to the existing model in the Boolean setting, we assume that each basic action of the proponent $a \in Act_p$ has two associated costs (non-negative real numbers). One is the cost of not performing a (0 in the following), the other is the cost c of performing a. We set both costs of the opponent actions to 0.

Extending the model with costs and evaluating the pairs of success *and* cost values lead to multi-parameter optimisation. Moreover, such pairs are incomparable in case the goal of a player is to maximise one parameter while minimising the other. In order to address multi-parameter optimisation in the case of incomparable values, we resort to Pareto efficiency and define two functions for computing the sets of Pareto efficient solutions. A solution is called Pareto efficient if it is not dominated by any other solution in the ordering relation [10].

We assume that the goal of the proponent is to maximise the success value while minimising the cost of an attack or defence. In order to compute the set of pairs of efficient solutions, where we want to maximise the first argument while minimising the second, we define function MR^{+-}. The function computes the set of all pairs that have higher value for the first argument *or* lower value for the second argument with respect to the other pairs in the set.

$$\begin{aligned} MR^{+-}(Z) &= \{(x,y) \in Z \mid \forall (x',y') \in Z : x' \sqsupseteq x \wedge y' \sqsubseteq y \Rightarrow x' = x \wedge y' = y\} \\ &= \{(x,y) \in Z \mid \forall (x',y') \in Z : (x \sqsupseteq x' \vee y \sqsubset y') \ \wedge \ (x \sqsupset x' \vee y \sqsubseteq y')\} \end{aligned}$$

where $Z \subseteq D$.

Note that the sign "+" indicates the maximisation and the sign "-" indicates the minimisation, and their position refer to the parameter of the maximisation/minimisation.

In Sect. 2.2 we discussed how the negation and changing player operators change the goal, e.g., maximisation is turned into minimisation. Thus, if the proponent's goal is to maximise the success value and minimise the cost of an attack or defence, then under negation the goal is to minimise the success value and minimise the cost of an attack or defence. Observe that the goal for the cost does not change, as we assume to deal with rational players. Therefore, we define function MR^{--} to compute the set of all pairs that have lower values for both arguments with respect to the other pairs in the set.

$$\begin{aligned} MR^{--}(Z) &= \{(x,y) \in Z \mid \forall (x',y') \in Z : x' \sqsubseteq x \wedge y' \sqsubseteq y \Rightarrow x' = x \wedge y' = y\} \\ &= \{(x,y) \in Z \mid \forall (x',y') \in Z : (x \sqsubseteq x' \vee y \sqsubset y') \ \wedge \ (x \sqsubset x' \vee y \sqsubseteq y')\} \end{aligned}$$

where $Z \subseteq D$.

Observe that the "+" symbol in MR^{+-} corresponds to the outer-most max operator in Table 2, and the first "-" symbol in MR^{--} corresponds to the outer-most min operator.

Following the previous developments, we present two evaluation techniques.

The semantic evaluation $M(t, A)$ of an attack-defence tree $t \in \mathit{Tree}_p$ and a set of actions A is illustrated in Table 7. It computes a pair, where the first argument is a set computed by the function MR^{--} and consists of all pairs that have lower success value and lower cost of the proponent actions compared to other pairs, and the second argument is the set computed by the function MR^{+-} and consists of all pairs that have higher success value and lower cost of the proponent actions compared to other pairs.

As we discussed in Sect. 2.2, if the proponent wants to maximise the success of the main goal, then the opponent wants to minimise it. In other words, the players affect the computation of the success value of the main goal in opposite ways, e.g., when one wants to maximise, the other wants to minimise and vice versa. The functions MR^{--} and MR^{+-} evaluate the success values based on the goal of the proponent. In order to consider the effect of the opponent with the opposite goal, we define functions $f_1^{m_p}(t)$ and $f_2^{m_p}(t)$ given in Table 7. The functions compute respectively the maximum and minimum success values over all Boolean assignments m_o for a given Boolean assignment m_p.

As the costs of the opponent's actions are 0 and do not influence the overall cost of the proponent, we consider only the cost of the proponent. The cost is represented with the concept of a budget, denoted by b_p, for associating with each success value the corresponding budget of the proponent and the Boolean assignment m_p. The budget $b_p \in \mathbb{R}_{\geq 0}$ takes values from 0 to infinity in an increasing manner. For a given budget b_p we take m_p such that the cost of the proponent for m_p is not greater than b_p, and the corresponding success value for m_p is computed. The cost for a given m_p is computed with the function *cost*, defined in Table 7.

Table 7. The Boolean semantic evaluation of an attack-defence tree with cost

$$M(t,A) = (MR^{--}(\{\ (f_1^{m_p}(t), b_p) \mid cost(m_p, A) \leq b_p\}),\\ MR^{+-}(\{\ (f_2^{m_p}(t), b_p) \mid cost(m_p, A) \leq b_p\}))$$

$$f_1^{m_p}(t) = \max\{\mathcal{B}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\}$$
$$f_2^{m_p}(t) = \min\{\mathcal{B}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\}$$

$$cost(m_p, A) = \sum_{a \in A} \begin{cases} c(a), & \text{if } m_p(a) = tt \\ 0, & \text{if } m_p(a) = ff \end{cases}$$

The result of the semantic evaluation is a pair of sets of Pareto efficient solutions for the given optimisation criteria of the goal.

The algorithmic evaluation $INT(t)$ for an attack-defence tree $t \in Tree_p$ is given in Table 8. It again computes a pair, where the first argument consists of all pairs that have lower success value and cost of the proponent actions, and the second argument consists of all pairs that have higher success value and lower cost of the proponent actions. Such sets are computed in the bottom-up fashion, defined by the rules presented in Table 8. The rules extend the ones for the success value computation, presented in Sect. 2.2, Table 3.

The first rule assigns the sets MR^{--} and MR^{+-} to the basic actions. Observe that the cost of not performing the proponent's actions, as well as both costs of the opponent's actions, is 0. The rules for conjunction and disjunction use the common computation for success values and sum the costs. The negation and changing player operators evaluate the success value as described in Sect. 2.2, while leaving the cost unchanged. The last two rules correspond to always successful and failed actions, which are independent from the players, and thus have a cost equal to 0. Applying the functions MR^{--} and MR^{+-} in each rule of the evaluation allows to reduce the size of the sets in each step.

We denote by $yield(t) \subseteq Act$ the set of actions that are in the leaves of t. By considering the polarity-consistent tree $t = \&_\wedge(\&_\vee(a,b), \&_\vee(a, \&_\sim(c)))$, where $a, b \in Act_p, c \in Act_o$, $c(a) = 2$ and $c(b) = 1$, we see that the two evaluation techniques lead to different results. The semantic evaluation gives the result $M(t, yield(t)) = (\{(ff, 0)\}, \{(ff, 0), (tt, 2)\})$, while the algorithmic evaluation gives the result $INT(t) = (\{(ff, 0)\}, \{(ff, 0), (tt, 3)\})$.

Therefore, the assumption of polarity consistency, considered in the Boolean case in Sect. 2.2, is no longer adequate when dealing with the notion of cost. In order to show the equivalence of the two evaluations we resort to a stronger assumption of linearity.

Theorem 3. *If $t \in Tree$ is a linear tree, then $M(t, yield(t)) = INT(t)$.*

We measure complexity of the two evaluation methods by calculating the number of set operations. The semantic evaluation is exponential in the size of t, as it considers all Boolean assignments. The algorithmic evaluation is linear and hence presents a dramatic improvement in the case of linear trees.

Table 8. The Boolean algorithmic evaluation of an attack-defence tree with cost

$$
\begin{array}{lll}
INT(a) & = \begin{cases} (MR^{--}(\{(ff,0),(tt,c(a))\}), & \\ \ MR^{+-}(\{(ff,0),(tt,c(a))\})) & \text{if } a \in Act_p \\ (MR^{--}(\{(tt,0)\}), MR^{+-}(\{(ff,0)\})) & \text{if } a \in Act_o \end{cases} \\
INT(\&_\wedge(t_1,t_2)) & = let\ (V_i,W_i) = INT(t_i), i \in \{1,2\} & \\
 & \quad in\ (MR^{--}(\{(b \wedge b', c+c') \mid (b,c) \in V_1, (b',c') \in V_2\}), & \\
 & \qquad MR^{+-}(\{(b \wedge b', c+c') \mid (b,c) \in W_1, (b',c') \in W_2\})) & \\
INT(\&_\vee(t_1,t_2)) & = let\ (V_i,W_i) = INT(t_i), i \in \{1,2\} & \\
 & \quad in\ (MR^{--}(\{(b \vee b', c+c') \mid (b,c) \in V_1, (b',c') \in V_2\}), & \\
 & \qquad MR^{+-}(\{(b \vee b', c+c') \mid (b,c) \in W_1, (b',c') \in W_2\})) & \\
INT(\&_\neg(t)) & = let\ (V,W) = INT(t) & \\
 & \quad in\ (MR^{--}(\{(\neg b, c) \mid (b,c) \in W\}), MR^{+-}(\{(\neg b, c) \mid (b,c) \in V\})) & \\
INT(\&_\sim(t)) & = let\ (V,W) = INT(t) & \\
 & \quad in\ (MR^{--}(\{(\neg b, c) \mid (b,c) \in W\}), MR^{+-}(\{(\neg b, c) \mid (b,c) \in V\})) & \\
INT(\&_{\text{true}}) & = (\{(tt,0)\}, \{(tt,0)\}) & \\
INT(\&_{\text{false}}) & = (\{(ff,0)\}, \{(ff,0)\}) &
\end{array}
$$

3.2 Cost in the Probabilistic Case

In this section we briefly generalise our development to the probabilistic setting, concentrating on the differences with respect to the Boolean setting.

In the probabilistic setting the cost-related questions are the same as in the Boolean setting, and the same observation regarding the cost to the proponent and to the opponent applies. Here we consider the set $D = [0,1] \times \mathbb{R}_{\geq 0}$. The cost is integrated to the basic actions in the same way. Thus, to extend the probabilistic model with costs, we assume that each basic action of the proponent player $a \in Act_p$ has two associated costs, 0 in case of not performing a, and c in case of performing a. We set both costs of the opponent's actions equal to 0.

As in the Boolean case, by considering probability *and* cost and focusing on maximising the first while minimising the other, we face a multi-parameter optimisation issue with incomparable values. Similarly to the Boolean setting, we provide two evaluation techniques based on Pareto efficiency by considering the functions MR^{--} and MR^{+-}. We compute the set of Pareto efficient solutions for answering questions such as "What is the maximum probability and the minimum cost of an attack?".

The semantic evaluation $M(t,A)$ of an attack-defence tree $t \in Tree_p$ and a given set A is illustrated in Table 9. The evaluation follows the corresponding one for the Boolean case, described in the Sect. 3.1. The only difference is that the tree t is evaluated over the Boolean assignments by considering the probabilistic analysis $\mathcal{P}[\![t]\!]$ instead of the Boolean $\mathcal{B}[\![t]\!]$ one (and considering the corresponding

Table 9. The probabilistic semantic evaluation of an attack-defence tree with cost

$$M(t,A) = (MR^{--}(\{\ (f_1^{m_p}(t), b_p) \mid cost(m_p, A) \leq b_p\}),\\ MR^{+-}(\{\ (f_2^{m_p}(t), b_p) \mid cost(m_p, A) \leq b_p\}))$$

$$f_1^{m_p}(t) = \max\{\mathcal{P}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\}$$
$$f_2^{m_p}(t) = \min\{\mathcal{P}[\![t]\!](m_p, m_o) \mid m_o \text{ Boolean assignment}\}$$

$$cost(m_p, A) = \sum_{a \in A} \begin{cases} c(a), & \text{if } m_p(a) = tt \\ 0, & \text{if } m_p(a) = ff \end{cases}$$

probabilistic values for each action). The result of the evaluation is the pair of the sets MR^{--} and MR^{+-}, corresponding to the set of Pareto efficient solutions.

The algorithmic evaluation $INT(t)$ for a tree $t \in Tree_p$ is given in Table 10. It traverses the tree from the leaves to the root according to the rules presented in Table 10. The rules follow the corresponding ones of the Boolean setting.

Analogously to the previous section, we shall restrict to linear trees in order to show the equivalence of the two evaluations.

Theorem 4. *If $t \in Tree$ is a linear tree, then $M(t, yield(t)) = INT(t)$.*

Again, a syntactic restriction allows to develop a sound evaluation technique that is linear in the size of the tree as opposed to the exponential complexity that characterises the general case.

Running Example. Consider the linear attack-defence tree t discussed in Figure 1. Table 6 lists possible values for probability and cost for basic actions.

In order to detect the attacks with maximum probability of success and minimum cost, we apply the algorithmic evaluation. At the root we obtain: $INT(t) = (\{(0,0)\}, \text{“The plot in Figure 2”}\})$. The overall result of the evaluation, i.e., the set of efficient solutions for the goal representing the Pareto frontier of the problem, is displayed in Figure 2. The probability of successful attacks ranges from 0 to 0.97 and the corresponding cost ranges from 0 to 695. The intermediate points on the Pareto frontier indicate other optimal solutions. We can conclude that the system under study is (p,c)-vulnerable for all the incomparable pairs in the Pareto frontier. In particular, the attack is not trivially attainable (all pairs with probability greater than zero require a cost greater than zero).

4 Attack-Defence Trees with Multiple Cost

In this section we extend further the model to deal with multiple costs for basic actions. Observing that the Boolean setting is a special case of the probabilistic one, in the following we describe the extended model only in the probabilistic setting, focusing on the extensions with respect to a single cost model.

Table 10. The probabilistic algorithmic evaluation of an attack-defence tree with cost

$$
\begin{array}{ll}
INT(a) & = \begin{cases} (MR^{--}(\{(M_1(a),0),(M_2(a),c(a))\}), & \\ \ MR^{+-}(\{(M_1(a),0),(M_2(a),c(a))\})) & \text{if } a \in Act_p \\ (MR^{--}(\{(M_2(a),0)\}), MR^{+-}(\{(M_1(a),0)\})) & \text{if } a \in Act_o \end{cases} \\
INT(\&_\wedge(t_1,t_2)) & = let\ (V_i,W_i) = INT(t_i), i \in \{1,2\} \\
 & \quad in\ (MR^{--}(\{(p\cdot p', c+c') \mid (p,c) \in V_1, (p',c') \in V_2\}), \\
 & \quad\quad MR^{+-}(\{(p\cdot p', c+c') \mid (p,c) \in W_1, (p',c') \in W_2\})) \\
INT(\&_\vee(t_1,t_2)) & = let\ (V_i,W_i) = INT(t_i), i \in \{1,2\} \\
 & \quad in\ (MR^{--}(\{(1-(1-p)(1-p'), c+c') \mid (p,c) \in V_1, (p',c') \in V_2\}), \\
 & \quad\quad MR^{+-}(\{(1-(1-p)(1-p'), c+c') \mid (p,c) \in W_1, (p',c') \in W_2\})) \\
INT(\&_\neg(t)) & = let\ (V,W) = INT(t) \\
 & \quad in\ (MR^{--}(\{(1-p,c) \mid (p,c) \in W\}), MR^{+-}(\{(1-p,c) \mid (p,c) \in V\})) \\
INT(\&_\sim(t)) & = let\ (V,W) = INT(t) \\
 & \quad in\ (MR^{--}(\{(1-p,c) \mid (p,c) \in W\}), MR^{+-}(\{(1-p,c) \mid (p,c) \in V\})) \\
INT(\&_{\text{true}}) & = (\{(1,0)\}, \{(1,0)\}) \\
INT(\&_{\text{false}}) & = (\{(0,0)\}, \{(0,0)\})
\end{array}
$$

We consider the set $D = [0,1] \times \mathbb{R}^n_{\geq 0}$. We assume that each basic action of the proponent $a \in Act_p$ has a vector of n associated costs, a vector of 0's in case of not performing a, and a vector $\gamma : Act_p \to \mathbb{R}^n_{\geq 0}$ in case of performing a. When adding costs we resort to point-wise summation of vectors. We set the cost of the opponent's actions to vectors of 0's.

Analogously to the previous sections, we deal with a multi-parameter optimisation with incomparable values. We give two evaluation techniques by using Pareto efficiency. In order to generalise the functions MR^{--} and MR^{+-}, defined in the previous section, we introduce polarity modifications of the comparison operators as follows: $\sqsupseteq^+$ is $\sqsupseteq$, $\sqsupset^+$ is $\sqsupset$, $\sqsupseteq^-$ is $\sqsubseteq$ and $\sqsupset^-$ is $\sqsubset$. The sign "+" corresponds to the maximisation of the parameters and keeps the operator as it is, while the sign "−" corresponds to the minimisation of the parameters, therefore it changes the operator.

We define a general frontier function, where $s_i \in \{+,-\}$ and $Z \subseteq D$, as follows:

$$
\begin{aligned}
MR^{s_0,\cdots,s_n}(Z) &= \{(x_0,\cdots,x_n) \in Z \mid \forall (x'_0,\cdots,x'_n) \in Z : \\
&\quad x'_0 \sqsupseteq^{s_0} x_0 \wedge \cdots \wedge x'_n \sqsupseteq^{s_n} x_n \Rightarrow x'_0 = x_0 \wedge \cdots \wedge x'_n = x_n\} \\
&= \{(x_0,\cdots,x_n) \in Z \mid \forall (x'_0,\cdots,x'_n) \in Z : \\
&\quad ((x_0 \sqsupseteq^{s_0} x'_0) \vee (x_1 \sqsupset^{s_1} x'_1) \vee \cdots \vee (x_n \sqsupset^{s_n} x'_n)) \wedge \cdots \\
&\quad \cdots \wedge ((x_0 \sqsupset^{s_0} x'_0) \vee (x_1 \sqsupset^{s_1} x'_1) \vee \cdots \vee (x_n \sqsupseteq^{s_n} x'_n))\}
\end{aligned}
$$

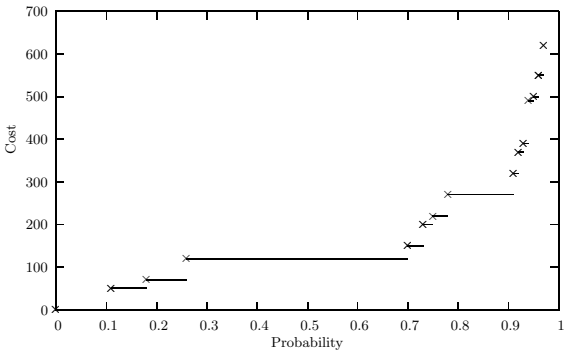

Fig. 2. Pareto efficient solutions for the attack-defence tree t

The function $MR^{s_0,\cdots,s_n}$ computes the efficient solutions for multiple parameters by maximising the parameter values if $s_i = +$ and minimising it if $s_i = -$. Note that each $\sqsupseteq^{s_i}$ is in fact a total order (on [0,1] or $\mathbb{R}_{\geq 0}$) and hence $\neg(x_i' \sqsupseteq^{s_i} x_i)$ is equivalent to $x_i \sqsupset^{s_i} x_i'$ (as in Sect. 3). Observe that with this notation we get MR^{--} when we take $n = 1$ and $s_0 = s_1 = -$, and we get MR^{+-} when we take $n - 1$ and $s_0 = +, s_1 = -$.

The definition of the semantic and algorithmic evaluations closely follows that of the corresponding ones in the Sect. 3.2. Similarly, we show their equivalence by restricting to linear trees.

For better understanding the extension to multiple costs, let us explain the rule for conjunction in the algorithmic evaluation. The rule is as follows:

$$\begin{aligned} INT(\&_\wedge(t_1,t_2)) = let\ (V_i,W_i) = INT(t_i), i \in \{1,2\} \\ in\ (MR^{--\cdots-}(\{(p\cdot p', c_1+c_1', \cdots, c_n+c_n') \mid \\ (p,c_1,\cdots,c_n) \in V_1, (p',c_1',\cdots,c_n') \in V_2\}), \\ MR^{+-\cdots-}(\{(p\cdot p', c_1+c_1', \cdots, c_n+c_n') \mid \\ (p,c_1,\cdots,c_n) \in W_1, (p',c_1',\cdots,c_n') \in W_2\})) \end{aligned}$$

First, it computes all possible combinations of pairs from both sub-trees t_1 and t_2 by multiplying probabilities and summing costs. Then, it applies functions $MR^{\cdots}$ in order to get the Pareto efficient solutions. This is sound due to the point-wise ordering of the set.

5 Related Work

We now expand on the comparison with related work given in Sect. 1. Different graphical approaches have been studied, for evaluating the security of a system. A historical overview on existing graph-based approaches for security threats is given by Pièvre-Cambacédès and Boussou [11]. Moreover, Kordy et al. summarise the existing methodologies for analysing attack and defence scenarios in [12].

As we mentioned in Sect. 1, Schneier developed attack trees as an approach to analyse the security of complex systems [3]. Further extensions of attack trees based on Schneier's model have been considered, such as attack graphs [13,14] and dynamic attack trees [15,16], as well as tools for modelling [17,18] and generating automatically [19] attack trees. Mauw and Oostdijk give a formal semantics of attack trees in [20]. Moreover, Buldas et al. developed a multi-parameter attack tree model for security analysis against rational attacks [21], subsequently extended in [22,23,24].

While attack trees focus on evaluating attack scenarios, other tree-structure representations incorporate countermeasures. Bistarelli et al. introduced an extension of attack trees with defender actions to the leaves of a tree [4]. Edge et al. proposed protection trees, a methodology for allocating appropriate protections against specified attacks such that the success probability of the defender is maximised [25]. Zonouz et al. [26] and Roy et al. [5] proposed a methodology for attack and defence modelling that combines analytical methods of attack and defence trees. They capture attacks and countermeasures at any node of a tree.

Finally, Kordy et al. [6] formalised attack-defence trees as an intuitive model for presenting attacks and countermeasures in a single view. For evaluating attack-defence trees the typical bottom-up approach of attack trees is extended. Attack-defence trees are interpreted with various semantics to answer questions such as the vulnerability of the system to an attack or the minimum cost of an attack [6]. Most evaluations [6,27,28] analyse a specific aspect of a scenario and do not consider trees with multiple parameters.

Further developments on attack-defence trees have been carried out, such as studying the relationship between such trees and two-player games [8] and combining the tree methodology with Bayesian networks for analysing probabilistic measures of attack-defence trees with dependent actions [28].

6 Conclusion

The growing centrality of technology requires a thorough investigation of the security properties of complex systems with respect to cyber and physical attacks, as well as consideration of possible defences undertaken to counter such attacks.

Attack-defence trees are a useful tool to study attack-defence scenarios and present the interaction between an attacker and a defender in an intuitive way. Moreover, such models are relied on to develop quantitative analyses of attacks and defences. Many evaluation methods consider one-parameter trees or analyse multi-parameter trees focusing on one specific aspect of the scenario, such as probability of success or cost. Nonetheless, in case of multi-parameter models, conflicting objectives may lead to incomparable values, which require to optimise all parameters at once on pain of computing sub-optimal solutions.

In order to tackle this issue, we have presented evaluation techniques for multi-parameter attack-defence trees that optimise all parameters at once, leveraging the concept of Pareto efficiency. Our developments have been carried out on a new language-based formalism for attack-defence trees, which extends standard trees with negation and with a novel operator for player alternation. In this

language, the interaction between an attacker and a defender is made explicit by associating a player to each node thanks to a simple type system. We have called *proponent* the player at the root and *opponent* the other player.

We have developed analyses of attack-defence scenarios both in the Boolean and in the probabilistic settings, investigating aspects such as the feasibility and the cost of an attack or a defence. For each case we have illustrated the natural semantic evaluation technique as well as an algorithmic evaluation which enjoys a dramatic improvement in complexity, and we have proven under which conditions the latter can be relied on in place of the former. Both methods characterise the goal of the scenario with a set of Pareto efficient solutions.

Our current methods focus on the players independently: for evaluating the cost of the proponent p, we set the cost of the opponent to 0 and assign a budget to p. In future work, we plan to extend the model with a budget for the opponent, so as to compute the optimal solutions for both players at once.

Acknowledgment. Part of the research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318003 (TREsPASS). Special thanks to Roberto Vigo and Alessandro Bruni for valuable comments and inspiring discussions.

References

1. Vesely, W., Roberts, N., Haasl, D., Goldberg, F.: Fault Tree Handbook. Number v. 88 in Fault Tree Handbook. Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission (1981)
2. Weiss, J.D.: A system security engineering process. In: Proceedings of the 14th National Computer Security Conference, pp. 572–581 (1991)
3. Schneier, B.: Attack Trees: Modeling Security Threats. Dr. Dobb's Journal of Software Tools 24(12), 21–29 (1999)
4. Bistarelli, S., Fioravanti, F., Peretti, P.: Defense trees for economic evaluation of security investments. In: Availability, Reliability and Security, pp. 416–423 (2006)
5. Roy, A., Kim, D.S., Trivedi, K.S.: Attack countermeasure trees (ACT): Towards unifying the constructs of attack and defense trees. Security and Communication Networks 5(8), 929–943 (2012)
6. Kordy, B., Mauw, S., Radomirović, S., Schweitzer, P.: Foundations of attack–defense trees. In: Degano, P., Etalle, S., Guttman, J. (eds.) FAST 2010. LNCS, vol. 6561, pp. 80–95. Springer, Heidelberg (2011)
7. Bagnato, A., Kordy, B., Meland, P.H., Schweitzer, P.: Attribute decoration of attack-defense trees. IJSSE 3(2), 1–35 (2012)
8. Kordy, B., Mauw, S., Melissen, M., Schweitzer, P.: Attack-defense trees and two-player binary zero-sum extensive form games are equivalent. In: Alpcan, T., Buttyán, L., Baras, J.S. (eds.) GameSec 2010. LNCS, vol. 6442, pp. 245–256. Springer, Heidelberg (2010)
9. Girard, J.Y.: Linear logic: Its syntax and semantics. In: Proceedings of the Workshop on Advances in Linear Logic, pp. 1–42. Cambridge University Press (1995)
10. Legriel, J., Le Guernic, C., Cotton, S., Maler, O.: Approximating the pareto front of multi-criteria optimization problems. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 69–83. Springer, Heidelberg (2010)

11. Piètre-Cambacédès, L., Bouissou, M.: Beyond attack trees: Dynamic security modeling with boolean logic driven markov processes (BDMP). In: Eighth European Dependable Computing Conference, EDCC-8 2010, pp. 199–208 (2010)
12. Kordy, B., Piètre-Cambacédès, L., Schweitzer, P.: Dag-based attack and defense modeling: Don't miss the forest for the attack trees. CoRR abs/1303.7397 (2013)
13. Sheyner, O., Haines, J.W., Jha, S., Lippmann, R., Wing, J.M.: Automated generation and analysis of attack graphs. In: IEEE S&P 2002, pp. 273–284 (2002)
14. Jha, S., Sheyner, O., Wing, J.M.: Two formal analyses of attack graphs. In: 15th IEEE Computer Security Foundations Workshop (CSFW-15 2002), pp. 49–63 (2002)
15. Arnold, F., Hermanns, H., Pulungan, R., Stoelinga, M.: Time-dependent analysis of attacks. In: Abadi, M., Kremer, S. (eds.) POST 2014. LNCS, vol. 8414, pp. 285–305. Springer, Heidelberg (2014)
16. Khand, P.: System level security modeling using attack trees. In: Computer, Control and Communication, IC4 2009, pp. 1–6 (2009)
17. Amenaza: SecurITree, `http://www.amenaza.com`
18. Isograph: AttackTree+, `http://www.isograph.com/software/attacktree/`
19. Vigo, R., Nielson, F., Riis Nielson, H.: Automated Generation of Attack Trees. In: 27th Computer Security Foundations Symposium (CSF 2014), pp. 337–350. IEEE (2014)
20. Mauw, S., Oostdijk, M.: Foundations of attack trees. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 186–198. Springer, Heidelberg (2006)
21. Buldas, A., Laud, P., Priisalu, J., Saarepera, M., Willemson, J.: Rational Choice of Security Measures Via Multi-Parameter Attack Trees. In: López, J. (ed.) CRITIS 2006. LNCS, vol. 4347, pp. 235–248. Springer, Heidelberg (2006)
22. Jürgenson, A., Willemson, J.: Computing exact outcomes of multi-parameter attack trees. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1036–1051. Springer, Heidelberg (2008)
23. Jürgenson, A., Willemson, J.: On fast and approximate attack tree computations. In: Kwak, J., Deng, R.H., Won, Y., Wang, G. (eds.) ISPEC 2010. LNCS, vol. 6047, pp. 56–66. Springer, Heidelberg (2010)
24. Buldas, A., Lenin, A.: New efficient utility upper bounds for the fully adaptive model of attack trees. In: Das, S.K., Nita-Rotaru, C., Kantarcioglu, M. (eds.) GameSec 2013. LNCS, vol. 8252, pp. 192–205. Springer, Heidelberg (2013)
25. Edge, K., Dalton, G., Raines, R., Mills, R.: Using attack and protection trees to analyze threats and defenses to homeland security. In: MILCOM 2006, pp. 1–7. IEEE (2006)
26. Zonouz, S.A., Khurana, H., Sanders, W.H., Yardley, T.M.: RRE: A game-theoretic intrusion response and recovery engine. In: DSN 2009, pp. 439–448 (2009)
27. Kordy, B., Pouly, M., Schweitzer, P.: Computational aspects of attack-defense trees. In: Bouvry, P., Kłopotek, M.A., Leprévost, F., Marciniak, M., Mykowiecka, A., Rybiński, H. (eds.) SIIS 2011. LNCS, vol. 7053, pp. 103–116. Springer, Heidelberg (2012)
28. Kordy, B., Pouly, M., Schweitzer, P.: A probabilistic framework for security scenarios with dependent actions. In: Albert, E., Sekerinski, E. (eds.) IFM 2014. LNCS, vol. 8739, pp. 256–271. Springer, Heidelberg (2014)