Journal of Artificial Intelligence Research 75 (2022) 1293-1322 Submitted 08/2022; published 12/2022

Towards Evidence Retrieval Cost Reduction in Abstract Argumentation Frameworks with Fallible Evidence

Andrea Cohen AC@CS.UNS.EDU.AR
Sebastian Gottifredi SG@CS.UNS.EDU.AR
Alejandro J. García AJG@CS.UNS.EDU.AR
Guillermo R. Simari GRS@CS.UNS.EDU.AR
Institute for Computer Science and Engineering (CONICET-UNS) &
Department of Computer Science and Engineering, Universidad Nacional del Sur
San Andrés 800 - Campus Palihue, Bahía Blanca, Buenos Aires, Argentina

Abstract

Arguments in argumentation systems cannot always be considered as standalone entities, requiring the consideration of the pieces of evidence they rely on. This evidence might have to be retrieved from external sources such as databases or the web, and each attempt to retrieve a piece of evidence comes with an associated cost. Moreover, a piece of evidence may be available in a given scenario but not in others, and this is not known beforehand. As a result, the collection of active arguments (whose entire set of evidence is available) that can be used by the argumentation machinery of the system may vary from one scenario to another. In this work, we consider an Abstract Argumentation Framework with Fallible Evidence that accounts for these issues, and propose a heuristic measure used as part of the acceptability calculus (specifically, for building pruned dialectical trees) with the aim of minimizing the evidence retrieval cost of the arguments involved in the reasoning process. We provide an algorithmic solution that is empirically tested against two baselines and formally show the correctness of our approach.

1. Introduction

Computational Argumentation is a form of considering and making effective the task of reasoning that has proved to be valuable in different domains, such as logic-based environments (Besnard & Hunter, 2001; García & Simari, 2004), decision-making and negotiation (Black & Hunter, 2009; Ferretti et al., 2017), Artificial Intelligence & Law (Al-Abdulkarim et al., 2014; Prakken & Sartor, 2015), and has led to the development of argumentation-based recommender and decision support systems (Briguez et al., 2014; Bedi & Vashisth, 2014; Gómez et al., 2016). Briefly, it is a form of reasoning where a piece of information (claim) is accepted or rejected after considering the reasons (arguments) for and against that acceptance, providing a reasoning mechanism able to handle contradictory, incomplete, and often uncertain information. There exists a variety of approaches to argumentation-based reasoning, among which we can distinguish abstract (with the work initiated by Dung (1995) being the most prominent) and structured ones; we refer the reader to (Besnard et al., 2014) for an overview.

Given its query-answering nature, a structured argumentation system like DeLP (García & Simari, 2004) can be effectively used to implement decision-support or recommender systems (Briguez et al., 2014). In such systems, the knowledge used for building arguments

is usually encoded through rules of the form *premises-conclusion*. However, the rules used for building the arguments involved in the system's reasoning process may not be applicable in every scenario. Instead, the system will only be able to use a particular rule when it is capable of retrieving every piece of information (from hereon referred to as *evidence*) corresponding to this rule's premises. Taking this situation into consideration, it could be the case that a piece of evidence required for building an argument is available from one source at a given time, but not from others, or not even from the same source at a different time. As a result, it is possible that one argument could be built at one moment (because all its associated evidence is available, in which case we will say it is *active*) but not at others (since some associated evidence is not available making the argument *inactive*); thus, there is an inherently dynamic component in the argumentative process.

As argued in (Skiba et al., 2020), many real-life application scenarios for argumentation systems require the consideration of arguments not as standalone entities, but as relying on pieces of evidence that provide the basis for their construction (in other words, for their activation). As an example, they consider an online forum discussion addressing the spread of the COVID-19 virus. Many arguments being exposed in online discussions or debates are based on premises or facts, which need to be backed by some evidence. For instance, for the COVID-19 discussion case, the evidence could correspond to articles by the World Health Organization (WHO) or other resources of authority used for backing the claims being made.

Regarding the pieces of evidence associated with arguments, they could correspond to information mined from an external source such as a database (Deagustini et al., 2013, 2017) or, more generally, the web (Lippi & Torroni, 2016). Then, the process of building an argument comes with an additional associated cost, which may be the financial cost of accessing a particular database, the time for resolving a query, *et cetera*. Going back to the COVID-19 online forum discussion example, to assess the validity of the arguments, the participants could visit the linked pieces of evidence to verify the claims. However, this verification step involves time and effort (*i.e.,* each piece of evidence associated with the argument has a cost), and the retrieval of pieces of evidence may fail, because a web server may be down or the article is no longer available. As a result, if we account for the cost of building an argument, as well as the dynamic nature mentioned above, a reasonable approach for determining the acceptance status of an argument would try to minimize the associated cost (in particular, the cost of the evidence retrieval), while accounting for the fact that arguments may not be active.

In this work, we propose an approach for determining the acceptance status of arguments that have an associated set of evidence, based on the construction and pruning of dialectical trees (Besnard & Hunter, 2001; García & Simari, 2004). Moreover, our approach will aim at reducing the evidence retrieval cost paid for building the arguments in the tree, trying to avoid fetching unnecessary evidence or building arguments that do not affect the acceptance status of the queried argument.

Given a queried argument $\mathcal{A}$ whose acceptance status has to be determined in a scenario where the set of available evidence is $\mathbb{E}$ (though, as stated before, not known beforehand), the problem of identifying the "cheapest" set of evidence that would have to be retrieved in order to be able to determine whether $\mathcal{A}$ is accepted or rejected (while not leaving active arguments aside) can be considered as an optimization problem. As discussed in (Skiba

et al., 2020), this problem is equivalent to the problem of determining the cheapest set of evidence such that for every argument $\mathcal{B}$ that is not active but would change the acceptance status of the queried argument $\mathcal{A}$, we ensure that $\mathcal{B}$ cannot be constructed at all by trying to retrieve at least one piece of evidence of $\mathcal{B}$ that is not available in that scenario. Moreover, as shown in (Skiba et al., 2020), the complexity results of decision variants of this optimization problem mirror classical complexity results for abstract argumentation semantics (Dvorák & Dunne, 2018), but most problems are lifted one level up in the polynomial hierarchy. In particular, for the grounded semantics, whose similarity with semantics based on the construction of dialectical trees is discussed in (García et al., 2020), finding the optimal solution for this optimization problem is NP-complete. Consequently, heuristics become handy for tackling these kind of problems and thus, our proposed approach will make use of a heuristic measure based on the evidence retrieval cost.

Specifically, we will work with an Abstract Argumentation Framework with Fallible Evidence (AFFE), inspired on the Dynamic Argumentation Framework (DAF) proposed in (Rotstein et al., 2010). Briefly, the AFFE extends Dung's framework (Dung, 1995) so that each argument is associated with a set of evidence, which sets the argument to be active or inactive. In particular, each piece of evidence comes with an associated cost (that has to be paid in order to attempt to retrieve it) and can be available or unavailable at a given time. The semantics we will consider for determining the acceptance status of arguments in an AFFE is based on the construction of dialectical trees, similarly to (Rotstein et al., 2010). Then, by accounting for the arguments in an AFFE (regardless of their activation status), we will consider the construction of *potential* dialectical trees. Based on those trees, we will characterize a heuristic measure to guide the construction of the *active* dialectical trees (*i.e.,* the dialectical trees that include only active arguments) while trying to minimize the cost of retrieving the evidence associated with the arguments in those trees. Furthermore, as will be shown later, this measure will be helpful to select branches to prune in the active trees that do not affect the acceptance status of the argument in the tree's root.

The rest of this paper is organized as follows. Section 2 introduces the theoretical basis for our approach, including the formalization of the Abstract Argumentation Framework with Fallible Evidence (AFFE) and a characterization of potential, active, and pruned active dialectical trees. In Section 3, we introduce the notion of related evidence for arguments in an AFFE and propose a heuristic measure to be used for guiding the construction of pruned dialectical trees in active scenarios. Section 4 presents and describes the algorithms implementing the construction of pruned active dialectical trees, showing how the heuristic measure is effectively used for that purpose. It also illustrates our algorithmic solution through examples and finishes by formally showing the correctness of our approach. Then, Section 5 introduces the experiments we performed for testing our approach against two baselines, discussing and illustrating the main results. Finally, in Section 6 we comment on related works, highlight the differences between this paper and the preliminary work reported in (Cohen et al., 2019), discuss possible directions for future work, and provide some concluding remarks.

2. Acceptability Calculus in AFFEs through Dialectical Trees

Dung's abstract argumentation framework (Dung, 1995) has become the current standard to explore, analyze, and apply new ideas to any argumentation-based setting that does not consider the arguments' structure. In this paper, we adopt a less abstract framework that allows for the representation of active (respectively, inactive) arguments in a dynamic scenario. The developement of our approach is based on the consideration of a *Abstract Argumentation Framework with Fallible Evidence (AFFE)*, inspired on the *Dynamic Argumentation Framework* proposed in (Rotstein et al., 2010). In particular, the AFFE considers a universal set of *potential arguments*, which holds every conceivable argument, along with the attack relation defined over that set. Each potential argument is also associated with a set of *evidence*, which has to be retrieved for the argument to be active at a given scenario. The different scenarios will be represented through *sessions*, and they will be identified with natural numbers ($\mathbb{N}$); furthermore, every session may go through different *states*, representing the evolution in the search for the necessary evidence during that session.

Active arguments in an AFFE are the only ones that can be used by the argumentation machinery to make inferences and compute the acceptance status of arguments in a given session. Therefore, in each scenario, arguments from the universal set that are inactive represent reasons that, although structurally valid, cannot be taken into consideration because the necessary evidence that is required for their activation is not available in the context corresponding to the current session.

Notwithstanding this, active arguments in one session could become inactive and unavailable to reason in later sessions, *e.g.,* when some of their associated evidence is not present; analogously, inactive arguments may also become active later. Having this in mind, the AFFE will also account for a function aiming at retrieving the pieces of evidence that are available in a given session, consequently allowing us to determine the active arguments in that session. Specifically, since the set of available evidence in each session is not known beforehand, this function will fail in case the piece of evidence that was attempted to be retrieved is not available in that session. Moreover, since attempting to retrieve a piece of evidence comes with an associated cost, and this cost may vary from one piece of evidence to another, we will also equip the AFFE with a function to determine the evidence retrieval cost (expressed in natural numbers) in a given session. Finally, note that, while retrieving a piece of evidence, its associated cost has to be paid regardless of whether the requested evidence is available or not in the given session.

Definition 1 (Abstract Argumentation Framework with Fallible Evidence)**.** *An* Abstract Argumentation Framework with Fallible Evidence (AFFE) *is a tuple* $\langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$*, where* $\mathbb{U}$ *is the universal set of arguments,* $\hookrightarrow \subseteq \mathbb{U} \times \mathbb{U}$ *is an attack relation,* $\mathbb{E}$ *is the universal set of evidence,* $\Theta : \mathbb{E} \times \mathbb{N} \mapsto \{\top, \bot\}$ *is the evidence retrieval function,* $\Gamma : \mathbb{E} \mapsto \mathbb{N}$ *is the evidence cost function, and* $\mathfrak{ev} : \mathbb{U} \mapsto 2^{\mathbb{E}}$ *is a function determining the evidence required by each argument.*

Definition 2 (Active Arguments)**.** *Let* $\langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ *be an AFFE,* $\mathcal{A} \in \mathbb{U}$ *and* $\mathtt{s}$ *a session. We say that* $\mathcal{A}$ *is* active *in* $\mathtt{s}$ *iff* $\forall \epsilon \in \mathfrak{ev}(\mathcal{A}) : \Theta(\epsilon, \mathtt{s}) = \top$*. The set of active arguments in a session* $\mathtt{s}$ *is denoted* $\mathbb{A}_{\mathtt{s}} \subseteq \mathbb{U}$*.*

In order to determine whether an argument of an AFFE is active in a given session, we need to try to retrieve its evidence in that session. As mentioned before, given an argument $\mathcal{A}$, some pieces of evidence $\epsilon, \epsilon' \in \mathfrak{ev}(\mathcal{A})$, and a session $\mathtt{s}$, it might be the case that ϵ is available in $\mathtt{s}$ but ϵ' is not (*i.e.,* $\Theta(\epsilon, \mathtt{s}) = \top$ and $\Theta(\epsilon', \mathtt{s}) = \bot$). As a result, since every attempt of retrieving a piece of evidence through the function Θ comes with an associated cost (as determined by the function Γ), we want to minimize the use of Θ. Hence, to be able to keep track of the evidence that has been attempted to be retrieved so far in a session (both successfully and unsuccessfully), we define the notion of *session state* as follows.

Definition 3 (Session State)**.** *Let* $\langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ *be an AFFE and* $\mathtt{s}$ *a session. A* session state *is a tuple* $\sigma = (\mathtt{s}, \mathsf{CE}, \mathsf{ME})$*, where* $\mathsf{CE}, \mathsf{ME} \subseteq \mathbb{E}$*,* $\forall \epsilon \in \mathsf{CE} : \Theta(\epsilon, \mathtt{s}) = \top$*, and* $\forall \epsilon' \in \mathsf{ME} : \Theta(\epsilon', \mathtt{s}) = \bot$*.*

As Definition 3 shows, a session state has an associated session number and two sets of evidence representing, respectively, the evidence that has already been collected in the session (current evidence, CE), and the evidence that has been attempted to be retrieved but found missing in that session (missing evidence, ME). Therefore, when using the function Θ to try to retrieve a piece of evidence ϵ in a session state $\sigma = (\mathtt{s}, \mathsf{CE}, \mathsf{ME})$, a new session state will be obtained depending on the function's outcome. That is, if $\Theta(\epsilon, \mathtt{s}) = \top$, then the new session state will be $\sigma' = (\mathtt{s}, \mathsf{CE} \cup \{\epsilon\}, \mathsf{ME})$; otherwise, the new session state will be $\sigma'' = (\mathtt{s}, \mathsf{CE}, \mathsf{ME} \cup \{\epsilon\})$.

As it is the case for Dung's framework (Dung, 1995), the AFFE yields a graph of arguments connected by the attack relation. Given a session, an *active subgraph* could be considered, containing only active arguments. In argumentation-based reasoning, the challenge consists in finding out which arguments prevail after all things considered (*i.e.,* identifying the arguments that are accepted under some criterion). To this end, the notion of argumentation semantics has been extensively studied in the literature (Baroni et al., 2018). However, it should be noted that, in order to use the existing family of argumentation semantics, we have to consider the entire set of active arguments and the attacks between them. That is, to determine the acceptance status of an argument $\mathcal{A}$, every active argument would have to be built, including those arguments that may have no effect in $\mathcal{A}$'s acceptance or rejection. Hence, it can be argued that the use such semantics may lead to incurring unnecessary costs (the cost of attempting to build arguments like those mentioned above). Therefore, in this paper we will adopt an alternative approach that consists on building tree structures for determining the acceptance status of a queried argument, by considering only the arguments that are directly or indirectly related to it through the attack relation (García & Simari, 2004; Besnard & Hunter, 2001; Rotstein et al., 2010).

We define a *potential dialectical tree* as a tree structure where every node is associated with an argument, and the children of each node correspond to attackers of the associated argument. Also, the tree is such that no argument can be considered more than once within the same branch. Furthermore, the tree has the characteristic of being exhaustive, in the sense that if an argument (node) satisfies the above mentioned requirements, then it should be added to the tree structure; in other words, the tree should be exhaustive in the consideration of attackers for each argument.

Definition 4 (Potential Dialectical Tree)**.** *Let* $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ *be an AFFE and* $\mathcal{A} \in \mathbb{U}$*. The* potential dialectical tree *TP*$(\mathcal{A})$ *for* $\mathcal{A}$ *is a tree structure where:*

(*i*) *The root of $TP(\mathcal{A})$ is labeled with argument $\mathcal{A}$.*

(*ii*) *Given a node N labeled with $\mathcal{B}$ in $TP(\mathcal{A})$, $\forall \mathcal{C} \in \mathbb{U}$ such that $(\mathcal{C}, \mathcal{B}) \in \hookrightarrow$, if there is no ancestor of N in $TP(\mathcal{A})$ labeled with $\mathcal{C}$, then there exists a node N' in $TP(\mathcal{A})$ such that N' is labeled with $\mathcal{C}$ and N' is a child of N.*

Note that there will be a single potential tree for each argument in $\mathbb{U}$. We refer to the tree as *potential* since it accounts for every argument in the universal set and thus, for every attack in the attack relation. Also, it should be noted that there might be different nodes in a potential dialectical tree labeled with the same argument. This is because the same argument may attack various arguments, which are associated with nodes in different branches of the tree. Thus, the attacking argument will be the label of different child nodes in those branches.

As stated before, a potential dialectical tree for an argument $\mathcal{A}$ is a structure accounting for every attacker of $\mathcal{A}$, every attacker of those attackers, and so on (*i.e.,* it considers every argument and attack in an AFFE). Notwithstanding this, since arguments (also, the attacks involving them) may be active or not depending on the session, we need to account for the active scenario (hence, the active arguments in the session) in order to determine the acceptance status of $\mathcal{A}$. For this purpose, the notion of *active dialectical tree* is introduced. Briefly, the active dialectical tree for an argument $\mathcal{A}$ in a session $\mathtt{s}$ contains a subset of the nodes of the potential dialectical tree for $\mathcal{A}$, which are labeled with active arguments in $\mathtt{s}$.

Definition 5 (Active Dialectical Tree)**.** *Let $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ be an AFFE, $\mathtt{s}$ a session, $\mathbb{A}_{\mathtt{S}}$ the set of active arguments in $\mathtt{s}$, and $\mathcal{A} \in \mathbb{A}_{\mathtt{S}}$. The* active dialectical tree *$T_{\mathtt{S}}(\mathcal{A})$ for $\mathcal{A}$ in $\mathtt{s}$ is a tree structure where:*

(*i*) *The root of $TP(\mathcal{A})$ is labeled with argument $\mathcal{A}$.*

(*ii*) *Given a node N labeled with $\mathcal{B}$ in $TP(\mathcal{A})$, $\forall \mathcal{C} \in \mathbb{A}_{\mathtt{S}}$ such that $(\mathcal{C}, \mathcal{B}) \in \hookrightarrow$, if there is no ancestor of N in $T_{\mathtt{S}}(\mathcal{A})$ labeled with $\mathcal{C}$, then there exists a node N' in $T_{\mathtt{S}}(\mathcal{A})$ such that N' is labeled with $\mathcal{C}$ and N' is a child of N.*

Similarly to the potential dialectical trees, given an active argument $\mathcal{A} \in \mathbb{A}_{\mathtt{S}}$, there will be a unique active dialectical tree $T_{\mathtt{S}}(\mathcal{A})$. Then, to determine the acceptance status of argument $\mathcal{A}$ in session $\mathtt{s}$, a marking criterion is applied over $T_{\mathtt{S}}(\mathcal{A})$ as defined next.

Definition 6 (Marking Criterion)**.** *Let $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ be an AFFE, $\mathtt{s}$ a session, $\mathbb{A}_{\mathtt{S}}$ the set of active arguments in $\mathtt{s}$, $\mathcal{A} \in \mathbb{A}_{\mathtt{S}}$ and $T_{\mathtt{S}}(\mathcal{A})$ the active dialectical tree for $\mathcal{A}$ in $\mathtt{s}$. We define the* marking criterion *on $T_{\mathtt{S}}(\mathcal{A})$ as follows:*

(*i*) *All leaves in $T_{\mathtt{S}}(\mathcal{A})$ are marked as* **U** *(undefeated).*

(*ii*) *Let N be a non-leaf node of $T_{\mathtt{S}}(\mathcal{A})$. The node N is marked as* **U** *iff every child of N is marked as* **D** *(defeated); otherwise, N is marked as* **D** *(iff at least one of its children is marked as* **U***).*

Finally, a query for an argument $\mathcal{A}$ in a session $\mathtt{s}$ is resolved by applying the marking criterion on its active dialectical tree, and then looking at the marking of the root in the marked tree: if the root of $T_{\mathtt{S}}(\mathcal{A})$ is marked as **U**, argument $\mathcal{A}$ is accepted in session $\mathtt{s}$; otherwise, it is rejected in that session.

Definition 7 (Query and Answer)**.** *Let $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ be an AFFE, $\mathtt{s}$ a session, $\mathbb{A}_{\mathtt{s}}$ the set of active arguments in $\mathtt{s}$, and $\mathcal{A} \in \mathbb{U}$. Also, if $\mathcal{A} \in \mathbb{A}_{\mathtt{s}}$, let $T_{\mathtt{s}}(\mathcal{A})$ be the active dialectical tree for $\mathcal{A}$ in $\mathtt{s}$, to which we apply the marking criterion of Definition 6. The answer for a query about argument $\mathcal{A}$ in session $\mathtt{s}$ is **accepted** iff the root of $T_{\mathtt{s}}(\mathcal{A})$ is marked as* **U***, or **rejected** iff $\mathcal{A} \notin \mathbb{A}_{\mathtt{s}}$ or the root of $T_{\mathtt{s}}(\mathcal{A})$ is marked as* **D***.*

To illustrate these notions, let us consider the following example.

Example 1. *Let us consider an AFFE $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ and $TP(\mathcal{A})$, the potential dialectical tree for an argument $\mathcal{A} \in \mathbb{U}$ in the context of τ, as illustrated in Figure 1(a). Each node in the tree is depicted as a triangle; the argument labeling the node is inside the triangle, and the pieces of evidence associated with the argument are depicted below the triangle. Let us now suppose that Θ is such that the every piece of evidence, except from e_4, can be retrieved in a session S_1. Similarly, let us assume that the only pieces of evidence that cannot be retrieved in a session S_2 are e_6 and e_9. Then, Figures 1(b_1) and 1(b_2), respectively, illustrate the active dialectical tree for $\mathcal{A}$ in sessions S_1 and S_2.*

In the case of $T_{S_1}(\mathcal{A})$, we can note that argument $\mathcal{B}$ is not active because the piece of evidence e_4 is missing. Thus, the node for $\mathcal{B}$ is discarded in $T_{S_1}(\mathcal{A})$. As a result, the marking of the root in $T_{S_1}(\mathcal{A})$ is **U***, meaning that the answer for a query about $\mathcal{A}$ in session S_1 is **accepted**.*

On the other hand, if we consider $T_{S_2}(\mathcal{A})$, we note that arguments $\mathcal{D}$ and $\mathcal{E}$ are not active (thus, not included in the active tree) since they both have pieces of evidence that are not available in session S_2. Notwithstanding this, the example illustrates a difference between the status of the pieces of evidence e_6 and e_7 required by argument $\mathcal{D}$. When trying to build the node for $\mathcal{D}$ (thus, when attempting to retrieve the evidence associated with it), suppose we first fetch e_5 and then find out that e_6 is missing since it cannot be retrieved in session S_2. Hence, the node corresponding to argument $\mathcal{D}$ is discarded at that point, and no attempt to retrieve the piece of evidence e_7 is made. Finally, given that the root of $T_{S_2}(\mathcal{A})$ is marked as **D***, we can conclude that the answer to the query about argument $\mathcal{A}$ in session S_2 is **rejected**.*

As specified by the marking criterion introduced in Definition 6, whenever a node is marked as **U**, its parent node can be marked as **D**. Hence, the marking criterion can be optimized, so that the children of a node N are considered only up to the point where we find a child N' of N that is marked as **U**. In order to capture this behavior, we introduce the notion of *pruned active dialectical tree*, following the *and-or* pruning technique of (Chesñevar et al., 2000).

Definition 8 (Pruned Active Dialectical Tree)**.** *Let $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ be an AFFE, $\mathtt{s}$ a session, $\mathbb{A}_{\mathtt{s}}$ the set of active arguments in $\mathtt{s}$, $\mathcal{A} \in \mathbb{A}_{\mathtt{s}}$ and $T_{\mathtt{s}}(\mathcal{A})$ the active dialectical tree for $\mathcal{A}$ in $\mathtt{s}$. Also, let N be the set of nodes in $T_{\mathtt{s}}(\mathcal{A})$, and E be the set of edges in $T_{\mathtt{s}}(\mathcal{A})$. A* pruned active dialectical tree *$P_{\mathtt{s}}(\mathcal{A})$ for $\mathcal{A}$ in session $\mathtt{s}$ is a tree whose root is labelled with $\mathcal{A}$, and has a set of nodes $\mathsf{N_p} \subseteq \mathsf{N}$ and a set of edges $\mathsf{E_p} \subseteq \mathsf{E}$ such that $\forall N \in \mathsf{N_p}$: if N has a child N' marked as* **U** *in $T_{\mathtt{s}}(\mathcal{A})$, then N has exactly one child marked as* **U** *in $P_{\mathtt{s}}(\mathcal{A})$.*

We should remark that, given an active dialectical tree, different pruned versions of it can be obtained. The construction of such pruned trees depends on discovering opportunities

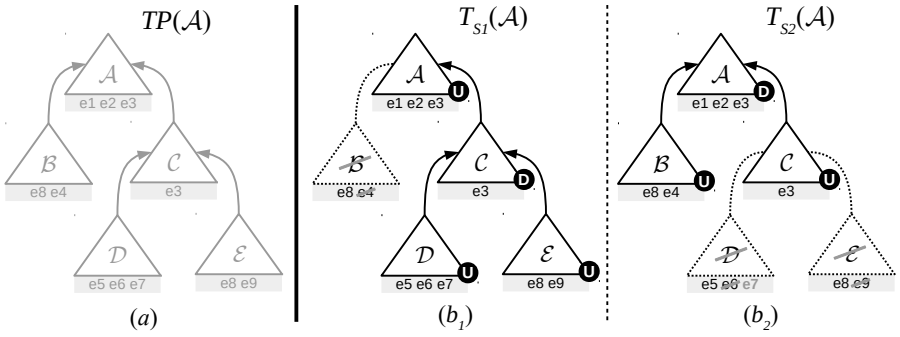

Figure 1: (a) The potential dialectical tree $TP(\mathcal{A})$ for argument $\mathcal{A}$, and two active dialectical trees for $\mathcal{A}$: (b_1) $T_{S_1}(\mathcal{A})$ in session S_1 and (b_2) $T_{S_2}(\mathcal{A})$ in session S_2.

for pruning. That is, during the construction of dialectical trees, the possibility of pruning a branch depends on the order in which the attackers of a given argument (node) are considered. Finally note that, although this pruning technique is not too restrictive and could even be used quite often, finding all the opportunities for pruning (thus, obtaining the smallest pruned active dialectical trees) requires just plain luck. This is because undefeated attackers must be found first at each step, and undefeatedness is not predictable — indeed, its discovery is the reason for building dialectical trees in the first place, to determine the acceptance status of arguments. To illustrate this, let us consider the following example.

Example 2. *First, it is important to highlight the difference between the parts of a pruned active dialectical tree that have been cut off (thus, unexplored) and those corresponding to failed attempts for building arguments. The former are depicted using light grey color and striking the pruned branch with a double line; the same color-coding is applied to the pieces of evidence of arguments that were not attempted to be retrieved. In contrast, the latter (i.e., arguments for which some piece of evidence is detected to be missing) are depicted as in Example 1 using dotted outlines; furthermore and the name of the argument and the corresponding piece of missing evidence are striked with a single line.*

Let $T_{S_1}(\mathcal{A})$ and $T_{S_2}(\mathcal{A})$ be the active dialectical trees for argument $\mathcal{A}$ in sessions S_1 and S_2, introduced in Example 1. The pruned active dialectical tree $P_{S_1}(\mathcal{A})_1$, depicted in Figure 2(a), is the pruned version of $T_{S_1}(\mathcal{A})$ obtained by considering the child $\mathcal{B}$ of $\mathcal{A}$ before $\mathcal{C}$, and the child $\mathcal{D}$ of $\mathcal{C}$ before $\mathcal{E}$. Therefore, by having the argument node $\mathcal{D}$ marked as **U**, *we can prune its sibling ($\mathcal{E}$) and establish the mark of $\mathcal{C}$ as* **D**. *Alternatively, if $\mathcal{E}$ is chosen before $\mathcal{D}$ when considering the children of $\mathcal{C}$, we obtain the pruned active dialectical tree $P_{S_1}(\mathcal{A})_2$, shown in Figure 2(b).*

Regarding $T_{S_2}(\mathcal{A})$, if the argument node $\mathcal{C}$ is chosen before $\mathcal{B}$ when considering the children of $\mathcal{A}$ in session S_2, we obtain $P_{S_2}(\mathcal{A})_1$, as depicted in Figure 3(a). Contrastedly, by choosing $\mathcal{B}$ before $\mathcal{C}$, we obtain the tree $P_{S_2}(\mathcal{A})_2$ illustrated in Figure 3(b).

On the one hand, the two pruned trees from Figure 2 attempted to build argument $\mathcal{B}$ and failed while trying to retrieve the piece of evidence e_4 in session S_1; hence, argument $\mathcal{B}$ is illustrated with a dotted outline. In contrast, Figures 2(a) and (b), pruned the branch

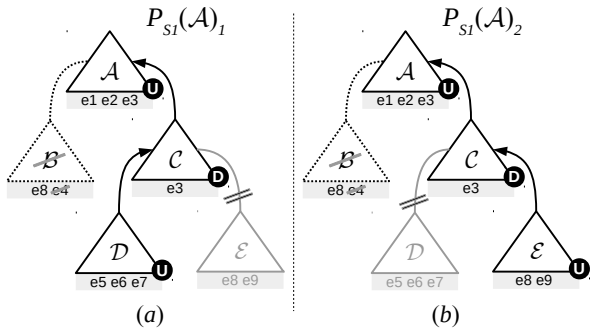

Figure 2: Two pruned active dialectical trees for $T_{S_1}(\mathcal{A})$, corresponding to Example 2.

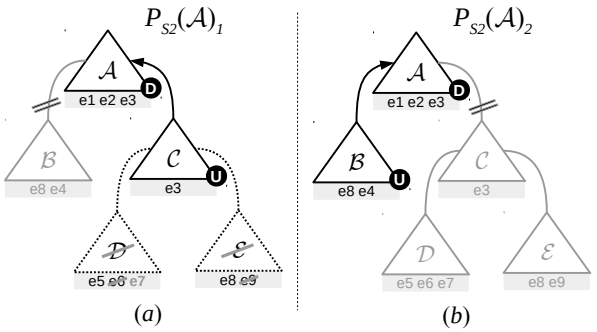

Figure 3: Two pruned active dialectical trees for $T_{S_2}(\mathcal{A})$, corresponding to Example 2.

corresponding to argument $\mathcal{E}$ or $\mathcal{D}$, respectively. Thus, the subtree corresponding to the pruned branch was not explored at all (as shown by the light grey color in the figure).

On the other hand, the pruned tree from Figure 3(a) corresponds to the case where argument $\mathcal{C}$ was chosen as the first attacker of $\mathcal{A}$ and then, its children could not be built since the pieces of evidence e_6 and e_9 were not available in session S_2; therefore, arguments $\mathcal{D}$ and $\mathcal{E}$ have a dotted outline in Figure 3(a). Again, it is important to note the difference between the pieces of evidence associated with argument $\mathcal{D}$: whereas e_5 was fetched and retrieved, and there was a failed attempt to retrieve e_6, no search for e_7 was carried out (hence, e_7 is depicted with a light grey color in the figure). Finally, Figure 3(b) illustrates the pruned active dialectical tree obtained by choosing $\mathcal{B}$ as the first child of $\mathcal{A}$ in session S_2. Then, since $\mathcal{B}$ is built and has no attackers, it is marked as **U** *and the branch corresponding to argument $\mathcal{C}$ is pruned.*

As mentioned before, in this work we aim at building dialectical trees for determining the acceptance status of arguments to resolve queries, while trying to minimize the evidence retrieval costs for obtaining them. For this purpose, we will introduce a pruning technique based on the evidence retrieval cost. Driven by this goal, in the next section we propose a heuristic measure that can be used for guiding the construction of pruned active dialectical trees.

3. Argument Related Evidence and Heuristic Measure

Here, we will propose a heuristic measure for guiding the construction of active dialectical trees in order to prune those subtrees with the highest evidence retrieval cost. Briefly, given a node corresponding to an argument in a potential dialectical tree and a session, this measure will provide an estimate of the cost for building the dialectical subtree rooted in that argument, given the evidence retrieved so far in that session. Thus, the measure will be computed using information from the potential and active scenarios, allowing for a partial pre-computation before the argumentation inference machinery starts running.

For that purpose, given a node N labeled with an argument $\mathcal{A}$ in a potential dialectical tree, we will first determine the set of evidence related to that argument, which includes every piece of evidence required to build the potential subtree rooted in it. Then, the cost associated with the related evidence set will be determined, accounting for the cost of attempting to retrieve each piece of evidence in the set, as specified by the function Γ.

Recall that a piece of evidence may be associated to multiple arguments (*i.e.,* different arguments may share the same piece of evidence). Moreover, since the same argument may appear more than once within the same dialectical tree (through the existence of different nodes in alternative branches of the tree that are labeled with the same argument), the same piece of evidence might be required at different points in a tree. As a result, when calculating the related evidence and its cost, every piece of evidence will be accounted for only once, regardless of the amount of argument nodes in the tree requiring it.

On the other hand it should be noted that, given the possibility of pruning active dialectical trees, an argument located in a deep level of a potential tree is less likely to be constructed in an active scenario than another argument located closer to the root of the tree. Consequently, in an active scenario, it would be less likely to attempt to retrieve a piece of evidence required by such arguments in deep levels. In general, the probability of trying to retrieve a piece of evidence in an active scenario decreases as the depth of the argument requiring that piece of evidence increases. Hence, when determining the related evidence of an argument in a potential tree, we estimate its cost accounting for these issues. This calculus is formalized in the following definition.

Definition 9 (Related Evidence and Cost)**.** *Let $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ be an AFFE, N a node labeled with an argument $\mathcal{A} \in \mathbb{U}$ in a potential dialectical tree TP and $\mathsf{CIF} \in (0, 1]$ a constant representing the cost impact factor of a piece of evidence. We define the set of* related evidence *of N in TP and its* cost *as:*

$$RelEv(N, TP) = \{(\epsilon, \Gamma(\epsilon) * \mathsf{CIF}^L) \mid \epsilon \in \mathfrak{ev}(Arg), Arg \in (\{\mathcal{A}\} \cup \mathsf{desc}(N, TP))\}$$

where $L = \mathsf{minLevel}(\epsilon, \mathsf{subTree}(TP, N))$ is the lowest level of a node in the subtree of TP rooted in $\mathcal{A}$, that is labeled with an argument requiring the piece of evidence ϵ[1]; and $\mathsf{desc}(N, TP)$ returns the set of arguments that label the descendants of N in the potential tree TP.

1. We consider that the level numbering in a dialectical tree starts with 0 (*i.e.,* the root of the tree is in level 0, its children are in level 1, and so on).

The related evidence cost given in Definition 9 provides an estimation of the actual cost of building an active dialectical tree. In particular, the reduction by CIF aims at adjusting the impact the cost of a piece of evidence has in the final cost, depending on its location on the tree. In other words, the cost of every piece of evidence is reduced by the cost impact factor CIF as many times as the lowest level number (*i.e.,* the closest to the root) of an argument requiring that piece of evidence. Finally, the reason why we consider the lowest level, is that it corresponds to the level of the argument requiring that piece of evidence that is more likely to be constructed.

Example 3. *Let us consider the potential dialectical tree $TP(\mathcal{A})$ from Example 1. For simplicity, since every node in $TP(\mathcal{A})$ is labeled with a different argument, we will refer to the nodes in the tree directly by their associated arguments. Let us assume a cost impact factor $\mathsf{CIF} = 0.5$, and suppose that the evidence cost function Γ is such that: $\Gamma(e_1) = 8$, $\Gamma(e_2) = 7$, $\Gamma(e_3) = 2$, $\Gamma(e_4) = 20$, $\Gamma(e_5) = 9$, $\Gamma(e_6) = 3$, $\Gamma(e_7) = 1$, $\Gamma(e_8) = 6$, $\Gamma(e_9) = 4$. The related evidence and cost for the different argument nodes in $TP(\mathcal{A})$ is:*

$$\begin{aligned}
\mathit{RelEv}(\mathcal{A}, TP(\mathcal{A})) = \quad & \{(e_1, 8 * 0.5^0), (e_2, 7 * 0.5^0), (e_3, 2 * 0.5^0), (e_8, 6 * 0.5^1), \\
& (e_4, 10 * 0.5^1), (e_5, 9 * 0.5^2), (e_6, 3 * 0.5^2), (e_7, 1 * 0.5^2), \\
& (e_9, 4 * 0.5^2)\} = \{(e_1, 8), (e_2, 7), (e_3, 2), (e_8, 3), (e_4, 5), \\
& (e_5, 2.25), (e_6, 0.75), (e_7, 0.25), (e_9, 1)\}
\end{aligned}$$

$$\mathit{RelEv}(\mathcal{B}, TP(\mathcal{A})) = \quad \{(e_8, 6 * 0.5^0), (e_4, 10 * 0.5^0)\} = \{(e_8, 6), (e_4, 10)\}$$

$$\begin{aligned}
\mathit{RelEv}(\mathcal{C}, TP(\mathcal{A})) = \quad & \{(e_3, 2 * 0.5^0), (e_5, 9 * 0.5^1), (e_6, 3 * 0.5^1), (e_7, 1 * 0.5^1), \\
& (e_8, 6 * 0.5^1), (e_9, 4 * 0.5^1)\} = \{(e_3, 2), (e_5, 4.5), (e_6, 1.5), \\
& (e_7, 0.5), (e_8, 3), (e_9, 2)\}
\end{aligned}$$

$$\begin{aligned}
\mathit{RelEv}(\mathcal{D}, TP(\mathcal{A})) = \quad & \{(e_5, 9 * 0.5^0), (e_6, 3 * 0.5^0), (e_7, 1 * 0.5^0)\} = \{(e_5, 9), \\
& (e_6, 3), (e_7, 1)\}
\end{aligned}$$

$$\mathit{RelEv}(\mathcal{E}, TP(\mathcal{A})) = \quad \{(e_8, 6 * 0.5^0), (e_9, 4 * 0.5^0)\} = \{(e_8, 6), (e_9, 4)\}$$

It should be noted that the calculus of the related evidence and cost only uses information from potential dialectical trees; specifically, it accounts for the pieces of evidence required by the arguments labeling the descendant nodes and the level of such nodes in the corresponding subtree. As a result, for every node in every potential tree, it is possible to determine its related evidence and cost during precompilation, avoiding any runtime overhead during the system's query-answering process in an active scenario.

Next, using the information about the related evidence and cost for a given argument node in a potential dialectical tree (which, as mentioned before, can be obtained prior to the system's execution), we define the *heuristic evidence cost* for that node. This measure will be used for guiding the construction of active dialectical trees in a particular session, with the aim of minimizing the evidence retrieval cost for building them.

Definition 10 (Heuristic Evidence Cost)**.** *Let $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ be an AFFE, N a node labeled with an argument $\mathcal{A} \in \mathbb{U}$ in a potential dialectical tree TP and $\sigma = (\mathsf{s}, \mathsf{CE}, \mathsf{ME})$*

a session state. We define the heuristic evidence cost *of N in TP and σ as follows:*

$$HeurEvCost(N, TP, \sigma) = \begin{cases} \infty & \text{if } (\mathsf{ME} \cap \mathfrak{ev}(\mathcal{A})) \neq \emptyset \\ \sum_{(\epsilon,c)\in RelEv(N,TP)\wedge\epsilon\notin\mathsf{CE}} c & \text{otherwise} \end{cases}$$

Example 4. *Continuing with our example, let $\sigma = (S_1, \{e_1, e_2, e_3\}, \emptyset)$ be the session state after building argument node $\mathcal{A}$ in session S_1. Then, for instance, we can determine the heuristic evidence cost of argument nodes $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$, and $\mathcal{E}$ in the potential dialectical tree for $\mathcal{A}$ in σ as follows:*

- $HeurEvCost(\mathcal{B}, TP(\mathcal{A}), \sigma) = 6 + 10 = 16$.
- $HeurEvCost(\mathcal{C}, TP(\mathcal{A}), \sigma) = 4.5 + 1.5 + 0.5 + 3 + 2 = 11.5$ *(note that e_3 is in the current evidence set, so its associated cost is disregarded).*
- $HeurEvCost(\mathcal{D}, TP(\mathcal{A}), \sigma) = 9 + 3 + 1 = 13$.
- $HeurEvCost(\mathcal{E}, TP(\mathcal{A}), \sigma) = 6 + 4 = 10$.

Differently from the calculus of the related evidence and cost, the heuristic evidence cost requires the consideration of information from the session state and thus, cannot be obtained during precompilation time. However, since the biggest part of the calculus involves determining the related evidence and its cost, the computation during execution time simply reduces to sum up the cost of the pieces of related evidence (determined during precompilation) that are not in the set of current evidence from the given session state. Otherwise, if the corresponding argument has a piece of evidence that was already found missing in the session, the heuristic evidence cost is set to infinity; the aim of doing this is to avoid the selection of such argument nodes when building the active dialectical trees in the given session, since they cannot be built. Consequently, the overhead of computing the heuristic evidence cost during the system's execution is significantly reduced in every session.

4. A Heuristic Pruning Approach for Building Active Dialectical Trees

In this section, we propose an approach for using the heuristic measure introduced in Section 3 to guide the construction of dialectical trees in an active scenario in order to determine the acceptance status of queried arguments. This measure will help to decide which branch of a tree should be explored at each time (thus, which arguments are to be built next). Also, the tree-building process will exploit the possibility of pruning branches that can be dismissed since they do not change the acceptance status of the root argument.

We will start by addressing the construction of a single argument in a session state. Given the abstract nature of arguments in our approach, this task reduces to determining whether the argument is active or not. As mentioned in Section 2, an argument will be active in a session state if all the evidence required by it can be retrieved in that session. Therefore, we have to find out whether we can successfully gather all its evidence in the given session.

Algorithm 1: Attempts to build the given argument, checking and collecting the necessary evidence

Function: `canBeBuilt`($\mathcal{A}$)
Input: An argument $\mathcal{A}$
Global: An AFFE $\langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$, and a session state $\sigma = (\mathtt{s}, \mathsf{CE}, \mathsf{ME})$
Result: *true* if all evidence associated with $\mathcal{A}$ is available in σ; *false* otherwise

```
1  EvidToFetch ← ev(A) \ CE;
2  if EvidToFetch ∩ ME ≠ ∅ then
3      return false;
4  foreach ϵ ∈ ev(A) do
5      if Θ(ϵ, s) = ⊤ then
6          σ ← (s, CE ∪ {ϵ}, ME);
7      else
8          σ ← (s, CE, ME ∪ {ϵ});
9          return false;
10 return true;
```

The process of building an argument will incur the costs associated with the retrieval of its associated evidence. These costs come from the use of the function Θ; thus, regardless of whether we are successful in the attempt of retrieving a piece of evidence or fail in the process (in which case the piece of evidence is considered to be missing), the associated cost is paid. As a result, since our goal is to minimize the evidence retrieval costs, we will try to use the function Θ as little as possible. For this purpose, when building an argument, we will take advantage of the information gathered while previously trying to build arguments in the corresponding session. On the one hand, our strategy will reutilize every piece of evidence that was already fetched during the session, avoiding to pay their retrieval cost more than once. On the other hand, if we are attempting to build an argument that has a piece of evidence we already found to be missing in the session, such an argument will be immediately discarded. Finally, for this strategy to work, every time the function Θ is used to attempt to retrieve a piece of evidence (paying the corresponding cost), the sets of current and missing evidence are updated, leading to a new session state. The process of determining whether an argument can be built in the current session is captured by the function `canBeBuilt`, illustrated in Algorithm 1.

The construction process of active dialectical trees, shown in Algorithm 2, follows a Depth-First Search strategy. Once an argument node N is built, we need to account for the children of N in the potential tree and then decide which subtree we will attempt to build next. In particular, the choice between different children will be guided by the Heuristic Evidence Cost of the corresponding argument nodes.

Since the construction of an active tree involves the construction of its subtrees, the algorithm is designed for building the subtree rooted in an argument, given its ancestors. In particular, given an argument $\mathcal{A}$, if $\mathcal{A}$ is established as the root and we consider an empty set of ancestors, the algorithm will in turn build the active dialectical tree rooted in $\mathcal{A}$. The structure of a node in Algorithm 2 contains an argument, its marking, and a set of

Algorithm 2: Builds the pruned active subtree of a given argument

Function: `prunedActiveSubTree`($\mathcal{A}$, *Root*, *Ancestors*)
Input: An argument $\mathcal{A}$, the root argument *Root* of an active dialectical tree, and a set of arguments *Ancestors* representing the ancestors of argument node $\mathcal{A}$ in the active tree rooted in *Root*
Global: An AFFE $\langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$
Result: A Pruned (sub)Tree

```
1  NodeA ← createNode(A, U);
2  Attackers ← {B | (B, A) ∈ ↪};
3  OrderedAttackers ← heuristicSort(Attackers, Root);
4  while OrderedAttackers ≠ ∅ do
5      B ← getFirst(OrderedAttackers);
6      if B ∉ Ancestors then
7          if canBeBuilt(B) then
8              NodeB ← prunedActiveSubTree(B, Root, Ancestors ∪ {A});
9              NodeA ← addChild(NodeA, NodeB);
10             if mark(NodeB) = U then
11                 NodeA ← setMark(NodeA, D);
12                 return NodeA;
13     OrderedAttackers ← remove(B, OrderedAttackers);
14     OrderedAttackers ← heuristicSort(OrderedAttackers);
15 NodeA ← setMark(NodeA, U);
16 return NodeA;
```

child nodes; therefore, a tree is represented directly by the node corresponding to its root argument.

It should be noted that the active tree returned by Algorithm 2 is a pruned tree. That is, once an argument node has been marked as **U** in the active tree, its unexplored siblings from the corresponding potential tree are dismissed. As a result, by guiding the selection of argument nodes through their heuristic evidence cost, the final cost of building the active pruned trees is reduced. Also, even though Algorithm 2 does not make explicit reference to a session, information from the session state is used by Algorithms 1 and 3 (which are in turn used by Algorithm 2). Finally, Algorithm 3 shows how the heuristic measure is calculated and then used for sorting a set of attackers from lowest to highest value. In particular, as mentioned in Section 3, to calculate an argument's heuristic evidence cost, the function `heuristicSort` makes use of the current session state, more specifically, of the set of current evidence. Finally, we note that `sort` is a standard sorting function taking a set of pairs (x, y) and ordering them from lowest to highest, depending on the value of y.

Example 5. *Let us consider the resolution of a query about argument $\mathcal{A}$ through the construction of its pruned active dialectical tree in session S_1. After building the argument node $\mathcal{A}$ we obtain the session state $\sigma = (S_1, \{e_1, e_2, e_3\}, \emptyset)$ and we have the empty set of ancestors. We establish $\{\mathcal{B}, \mathcal{C}\}$ as the set of attackers of $\mathcal{A}$ and obtain a list sorted*

Algorithm 3: Sorts a set of attackers using their heuristic evidential cost

Function: `heuristicSort`($Attackers, Root$)
Global: An AFFE $\langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$, a session state σ, and the potential dialectical tree $TP(Root)$ of $Root$
Input: A set of arguments $Attackers$, the root argument $Root$ of an active dialectical tree
Result: A sorted list of arguments

```
1 ArgsCost ← ∅;
2 foreach A ∈ Attackers do
3 |   CostA ← HeurEvCost(A, TP(Root), σ));
4 |   if CostA ≠ ∞ then
5 |   |   ArgsCost ← ArgsCost ∪ {(A, CostA)};
6 return sort(ArgsCost);
```

by their heuristic evidence cost (lines 2 and 3 in Algorithm 2). As shown in Example 4, $HeurEvCost(\mathcal{B}, TP(\mathcal{A}), \sigma) = 16$ *and* $HeurEvCost(\mathcal{C}, TP(\mathcal{A}), \sigma) = 11.5$. *As a result, the sorted list returned by Algorithm 3 is* $[\mathcal{C}, \mathcal{B}]$ *and* $\mathcal{C}$ *is chosen next (Algorithm 2, line 5). Then, since argument* $\mathcal{C}$ *is not in the set of ancestors and is buildable in* S_1 *(because* e_3*, the only piece of evidence it requires, is in the current evidence set), we proceed to build the pruned dialectical subtree rooted in* $\mathcal{C}$ *(Algorithm 2, line 8). The process of building the subtree for* $\mathcal{C}$ *is then analogous, by considering the set of ancestors* $\{\mathcal{A}\}$ *and the new session state; however, since the only piece of evidence required by* $\mathcal{C}$ *was already in the current evidence set, the session state obtained after building* $\mathcal{C}$ *continues to be* σ. *The set of attackers of* $\mathcal{C}$ *is* $\{\mathcal{D}, \mathcal{E}\}$ *and, by Example 4, we have* $HeurEvCost(\mathcal{D}, TP(\mathcal{A}), \sigma) = 13$ *and* $HeurEvCost(\mathcal{E}, TP(\mathcal{A}), \sigma) = 10$. *So, the ordered list of attackers is* $[\mathcal{E}, \mathcal{D}]$ *and argument* $\mathcal{E}$ *is chosen next. We are then able to build argument* $\mathcal{E}$ *(because its pieces of evidence* e_8 *and* e_9 *are available in session* S_1*) and, since it has no attackers to be considered, it is marked as* **U**. *Consequently,* $\mathcal{C}$ *can be marked as* **D**, *pruning the branch corresponding to the argument node* $\mathcal{D}$. *As a result, we go back to the construction of the tree rooted in* $\mathcal{A}$ *and consider the remaining attacker* $\mathcal{B}$. *Then, since the piece of evidence* e_4 *cannot be retrieved in session* S_1*, argument* $\mathcal{B}$ *is not buildable. Finally, since* $\mathcal{A}$ *has no remaining attackers we come up with the pruned dialectical active tree* $P_{S_1}(\mathcal{A})_2$*, depicted in Figure 2(b), whose root is marked as* **U** *meaning that argument* $\mathcal{A}$ *is* ***accepted*** *in session* S_1.

It should be noted that, each time an attacker has to be selected (Algorithm 2, line 5) when building an active tree, the remaining set of attackers is re-ordered (for the first time: Algorithm 2, line 3; subsequent times: Algorithm 2, line 14). This is of special interest since, as mentioned before, Algorithm 3 recomputes the heuristic evidence cost accounting for the current (updated) session state. Thus, it could be the case that a previous ordering among attacking arguments is changed afterwards. On the one hand, it could be the case that one of the attackers required a piece of evidence that was already found to be missing, in which case its heuristic evidence cost will be set to ∞ and the function `heuristicSort` will directly dismiss such attacker. On the other hand, it could be the case that one of the attackers required a piece of evidence that was already fetched (hence, it is in the current

evidence set) and thus, its heuristic evidence cost is now reduced. These advantages of Algorithm 2 are illustrated by the following example.

Example 6. *Let us consider an AFFE* $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ *and* $TP(\mathcal{A})$*, the potential dialectical tree for an argument* $\mathcal{A} \in \mathbb{U}$ *in the context of* τ*, as illustrated in Figure 4. Let us now suppose that* Θ *is such that every piece of evidence, except from* e_4*, can be retrieved in a session* $\mathtt{s}$*. Then, Figure 5 illustrates the active dialectical tree for* $\mathcal{A}$ *in session* $\mathtt{s}$*. Also, let us assume a cost impact factor* $\mathsf{CIF} = 0.5$ *and suppose that the evidence cost function* Γ *is such that:* $\Gamma(e_1) = 2$*,* $\Gamma(e_2) = 3$*,* $\Gamma(e_3) = 7$*,* $\Gamma(e_4) = 4$*,* $\Gamma(e_5) = 7$*,* $\Gamma(e_6) = 9$*,* $\Gamma(e_7) = 8$*,* $\Gamma(e_8) = 12$*,* $\Gamma(e_9) = 11$*. The related evidence and cost for the argument nodes corresponding to the children of* $\mathcal{A}$ *in* $TP(\mathcal{A})$ *is:*

$$\begin{aligned} \mathit{RelEv}(\mathcal{B}, TP(\mathcal{A})) &= \{(e_4, 4 * 0.5^0), (e_5, 7 * 0.5^0)\} \\ &= \{(e_4, 4), (e_5, 7)\} \end{aligned}$$

$$\begin{aligned} \mathit{RelEv}(\mathcal{C}, TP(\mathcal{A})) &= \{(e_5, 7 * 0.5^0), (e_6, 9 * 0.5^0), (e_7, 8 * 0.5^0)\} \\ &= \{(e_5, 7), (e_6, 9), (e_7, 8)\} \end{aligned}$$

$$\begin{aligned} \mathit{RelEv}(\mathcal{D}, TP(\mathcal{A})) &= \{(e_4, 4 * 0.5^0), (e_8, 12 * 0.5^0)\} \\ &= \{(e_4, 4), (e_8, 12)\} \end{aligned}$$

$$\begin{aligned} \mathit{RelEv}(\mathcal{E}, TP(\mathcal{A})) &= \{(e_9, 11 * 0.5^0), (e_7, 8 * 0.5^0), (e_1, 2 * 0.5^1)\} \\ &= \{(e_9, 11), (e_7, 8), (e_1, 1)\} \end{aligned}$$

Let us now consider the construction of $P_{\mathtt{s}}(\mathcal{A})$*, the pruned active dialectical tree for* $\mathcal{A}$ *in session* $\mathtt{s}$*, through Algorithm 2. We start by building the argument node* $\mathcal{A}$ *and, after that, we reach the session state* $\sigma = (\mathtt{s}, \{e_1, e_2, e_3\}, \emptyset)$*. Then, we consider the attackers of* $\mathcal{A}$ *and sort them by their heuristic evidence cost, obtained as follows:*

- $\mathit{HeurEvCost}(\mathcal{B}, TP(\mathcal{A}), \sigma) = 4 + 7 = 11$*.*
- $\mathit{HeurEvCost}(\mathcal{C}, TP(\mathcal{A}), \sigma) = 7 + 9 + 8 = 24$*.*
- $\mathit{HeurEvCost}(\mathcal{D}, TP(\mathcal{A}), \sigma) = 4 + 12 = 16$*.*
- $\mathit{HeurEvCost}(\mathcal{E}, TP(\mathcal{A}), \sigma) = 11 + 8 = 19$ *(note that* e_1 *is in the set of current evidence of* σ*, so its associated cost is disregarded).*

Therefore, the sorted list of attackers is $[\mathcal{B}, \mathcal{D}, \mathcal{E}, \mathcal{C}]$ *and* $\mathcal{B}$ *is chosen next. Now, suppose that while attempting to build argument* $\mathcal{B}$ *the piece of evidence* e_5 *is fetched first and then,* e_4 *cannot be retrieved in session* $\mathtt{s}$ *preventing the construction of argument* $\mathcal{B}$*. Consequently, the new session state is* $\sigma' = (\mathtt{s}, \{e_1, e_2, e_3, e_5\}, \{e_4\})$*. Next, as stated by Algorithm 2 (lines 13–14) argument* $\mathcal{B}$ *is removed from the set of attackers and the list of remaining attackers of* $\mathcal{A}$ *is reordered, for which their heuristic values are recomputed. Given the new session state* σ'*, it holds that:*

- $\mathit{HeurEvCost}(\mathcal{C}, TP(\mathcal{A}), \sigma') = 9 + 8 = 17$ *(*e_5 *is in the set of current evidence of* σ'*, so its associated cost is disregarded).*

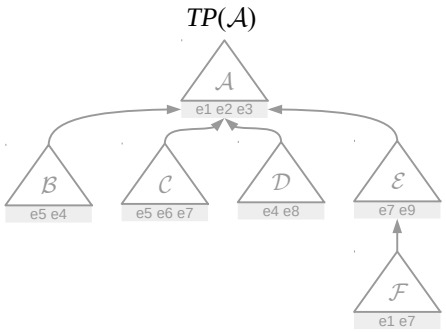

Figure 4: Potential dialectical tree $TP(\mathcal{A})$ from Example 6.

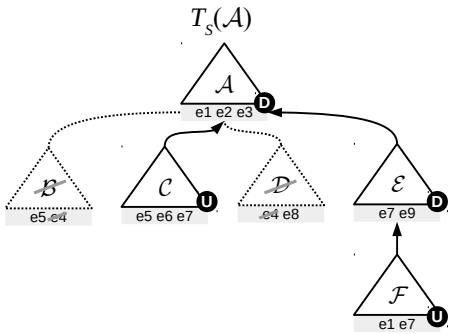

Figure 5: Active dialectical tree $T_{\mathtt{s}}(\mathcal{A})$ from Example 6.

- *HeurEvCost*$(\mathcal{D}, TP(\mathcal{A}), \sigma) = \infty$ *(because* e_4 *is in the set of missing evidence of* σ'*, so* $\mathcal{D}$ *cannot be built in session* s*).*
- *HeurEvCost*$(\mathcal{E}, TP(\mathcal{A}), \sigma) = 11 + 8 = 19$ *(*e_1 *is still in the set of current evidence of* σ'*, so its associated cost is disregarded).*

As a result, the new list of ordered attackers is $[\mathcal{C}, \mathcal{E}]$*, with* $\mathcal{D}$ *being removed because it cannot be built in session* s*. Now, we select* $\mathcal{C}$ *as the next attacker to be considered and we proceed to build the pruned dialectical subtree rooted in* $\mathcal{C}$ *(Algorithm 2, line 8), which is analogous to the process of building the tree rooted in* $\mathcal{A}$ *while considering the set of ancestors* $\{\mathcal{A}\}$ *and the new session state* σ'*. Finally, since argument* $\mathcal{C}$ *can be built in session* s *and has no attackers, the corresponding node will be marked as* **U** *and* $\mathcal{A}$ *can be marked as* **D***, pruning the branches corresponding to the argument nodes* $\mathcal{E}$ *and* $\mathcal{D}$*. That is, the pruned active dialectical tree returned by Algorithm 2 is the one depicted in Figure 6, whose root is marked as* **D***, meaning that argument* $\mathcal{A}$ *is* ***rejected*** *in session* s*.*

Finally, we show the correctness of Algorithm 2, by showing that it indeed returns a tree structure corresponding to a pruned active dialectical tree.

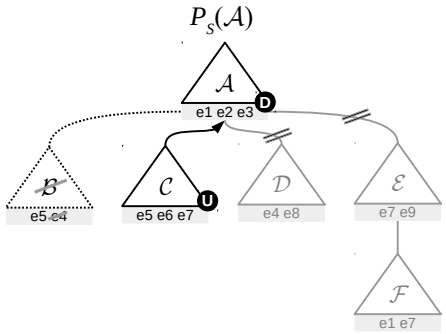

Figure 6: Pruned active dialectical tree $P_{\mathtt{S}}(\mathcal{A})$ from Example 6.

Theorem 1. *Let $\tau = \langle \mathbb{U}, \hookrightarrow, \mathbb{E}, \Theta, \Gamma, \mathfrak{ev} \rangle$ be an AFFE,* s *a session, $\mathbb{A}_{\mathtt{S}}$ the set of active arguments in* s*, $\mathcal{A} \in \mathbb{A}_{\mathtt{S}}$ and $T_{\mathtt{S}}(\mathcal{A})$ the active dialectical tree for $\mathcal{A}$ in* s*. The result of applying Algorithm 2 as* `prunedActiveSubTree`$(\mathcal{A}, \mathcal{A}, \emptyset)$ *is a tree structure $P_{\mathtt{S}}(\mathcal{A})$, corresponding to a pruned active dialectical tree for $\mathcal{A}$ in* s*.*

Proof. By hypothesis, Algorithm 2 is called as `prunedActiveSubTree`$(\mathcal{A}, \mathcal{A}, \emptyset)$. Then, by lines 1, 8, 9, 12 and 16 of Algorithm 2, the output tree structure is rooted in $\mathcal{A}$. Also, it follows directly that the output of Algorithm 2 is a tree structure, since every recursive call creates a new subtree whose root node is set as a child of its parent (lines 8–9). Moreover, the output tree structure is built using a set of nodes $\mathsf{N_p} \subseteq \mathsf{N}$, where N is the set of nodes in $T_{\mathtt{S}}(\mathcal{A})$. This is because, for every node X of the output tree structure, every attacker of the argument labelling X is considered (lines 2–3); also, by line 6, the attackers already appearing in the set of ancestors of X are discarded. Therefore, from the resulting set of attackers (which coincides with the set of children of X in $T_{\mathtt{S}}(\mathcal{A})$) only a subset will be added as children of X in the output tree structure (line 9). Finally, by lines 8–12, it can be seen that for every node X, its children will be added to the output tree structure up to the point in which one of X's children is marked as **U**, thus discarding (*i.e.,* pruning) the remaining children. □

5. Empirical Testing

In this section we will present the results we obtained by empirically testing our approach, from hereon referred to as heuristic evidence cost-guided pruned active tree building process (**CGPT**). We tested it against two baseline approaches: a non-guided process for building pruned active dialectical trees (**NGPT**) and another heuristic-based pruning technique based on arguments' strength (**SGPT**). In particular, the comparison between the results obtained for our approach and the two baselines will help us in understanding the improvement achieved by our method.

On the one hand, the non-guided approach imitates the usual process for the tree's construction. The main difference between **NGPT** and the other two approaches is that the former does not sort the set of attackers of a node, but randomly (hence, blindly) selects one attacker from the set as the next node to be considered. The implementation of

NGPT can be obtained from Algorithm 2, by by directly using the set *Attackers* instead of *OrderedAttackers* and avoiding the heuristic sorting process.

On the other hand, the strength-guided pruning approach (**SGPT**), proposed in (Rotstein et al., 2010), uses a heuristic measure encoding the arguments' strength. This measure is pre-computed for nodes in the potential dialectical trees, determined by the following formula, originally considered in (Besnard & Hunter, 2001):

$$Str(\mathcal{A}) = \frac{1}{1 + \sum_{\mathcal{B} \hookrightarrow \mathcal{A}} Str(\mathcal{B})}$$

that is, $\mathcal{B}$ ranges over the set of the attackers of $\mathcal{A}$. Briefly, this formula captures the intuition that an argument is as strong as weak are its attackers. Then, the implementation of **SGPT** corresponds to Algorithm 2 but modifying the function `heuristicSort` so that attackers of a node are decreasingly ordered by their strength (*i.e.,* stronger attackers are considered first).

We ran a simulation involving the generation of AFFEs and potential dialectical trees, according to the following parameters:

- `TreeNodeCount`: Amount of nodes in the potential dialectical tree. Values used: 100, 300, 500, 700, 900.
- `MaxBranchFactor`: Maximum branching factor in the potential dialectical tree. Values used: 5, 10, 20.
- `EvidUnivSize`: Coefficient by which `TreeNodeCount` is multiplied, in order to obtain the size of the evidence universe, expressed as the amount of pieces of evidence. Values used: 0.5, 1, 2.
- `MaxArgEvid`: Maximum number of pieces of evidence allowed per argument. Values used: 10, 20.
- `CIF`: Cost Impact Factor, as referred to in Definition 9. Value used in all tests: 0.5.
- `MaxEvidCost`: Maximum cost associated with each piece of evidence. Values used: 1, 10, 100, 1000.
- `DeactQuota`: Percentage of missing/deactivated evidence in a given session (w.r.t. `EvidUnivSize`). Values used: 1, 10, 20, 30.

In order to run the tests, we implemented the **CGPT**, **NGPT** and **SGPT** approaches in Prolog and used the SWI-Prolog interpreter. We also implemented an AFFE benchmark generator, taking into account the above described parameters.[2]

For each combination of values we built 500 AFFEs and performed 100 evidence deactivations (with the corresponding % of `DeactQuota`) per AFFE, each of which was considered for building a potential dialectical tree (*i.e.,* 500 potential trees were built, each of which was

2. The source code and the bash script shell file used for running the empirical evaluation described in this section are available at `https://github.com/nonicohen/HeurEvCost`

considered under 100 scenarios with different sets of available evidence). First of all, before starting the construction of an AFFE, the universal set of evidence (set $\mathbb{E}$ in Definition 1) was built by generating pieces of evidence numbered from 1 to `TreeNodeCount` $\times$`EvidUnivSize`, rounded down.

In order to facilitate running the experiments and ensure the amount of nodes in a potential dialectical tree built from an AFFE, as specified by the `TreeNodeCount` parameter, we chose to build each AFFE as a tree structure. Consequently, each of the 500 AFFEs being generated actually corresponds to a potential dialectical tree considered in the experiments. In that way, the key parameters associated with the structure of an AFFE are `TreeNodeCount` and `MaxBranchFactor`, where the former establishes the number of arguments in the AFFE (which, in particular, is a potential dialectical tree), and the latter establishes an upper bound for the number of attackers per argument in the AFFE. Also, this choice ensures that the same potential dialectical tree is tested against the 100 scenarios with different sets of deactivated evidence.

Specifically, for the AFFE construction, we generated arguments numbered from 1 to `TreeNodeCount`. Then, starting from argument 1 (set to be the root node), arguments were successively considered to be associated with their evidence and attackers in the following way. On the one hand, the corresponding pieces of evidence were associated to the argument by choosing a random number between 0 and `MaxArgEvid`, and the cost of each piece of evidence was determined by choosing a random number between 1 and `MaxEvidCost`. On the other hand, to associate an argument with its attackers, we first determined whether the argument was attacked at all, considering an 80% probability of being attacked.[3] Then, for the attacked arguments, the number of attackers was selected by choosing a random number between 1 and `MaxBranchFactor`, and each attacker was successively taken from the node count. For instance, if argument 1 has four attackers, these are arguments 2, 3, 4, and 5; then, if, for instance, argument 2 has three attackers, these correspond to arguments 6, 7 and 8.

Regarding the selection of attackers, we should remark that each argument can only be considered once as an attacker of another argument; this is to ensure that the AFFE being built actually has `TreeNodeCount` different nodes and, furthermore, to ensure that the AFFE is indeed a potential dialectical tree according to Definition 4 (because an argument cannot appear twice in a branch of the potential dialectical tree). Finally, in case of reaching a point where attackers for an argument have to be selected but all arguments have been already chosen as attackers of other arguments (*i.e.,* no arguments are available to be selected as new attackers), the process simply continues by leaving that argument unattacked.

It should be noted that, since every argument to be included in the AFFE under generation has an 80% probability of being attacked, in theory, it could be the case that the resulting AFFE has less than `TreeNodeCount` argument nodes. However, in practice, we evidenced that all generated AFFE instances had `TreeNodeCount` nodes. Notwithstanding this, during the AFFE generation, we explicitly checked that this was the case and, if not, the instance being generated was discarded and a new one was obtained.

3. It should be noted that, for the first argument (*i.e.,* the root argument, numbered by 1) we did not perform this, as the root argument always has to be attacked; otherwise, we would get a potential dialectical tree with just one node.

Before moving to discuss the experiments performed over the AFFEs, we would also like to highlight the following. The fact that AFFEs are built as tree structures, satisfying the conditions imposed over potential dialectical trees in Definition 4, do not make our results any less general. If we had considered AFFEs with generic graph structures, which might include cycles, Definition 4 would have prevented the appearance of the same argument more than once within the same branch of the potential dialectical trees built from those AFFEs. This behavior (*i.e.,* the non-repetition of arguments within the same branch) is guaranteed though our AFFE generation process. On the other hand, let us analyze the construction of a potential dialectical tree from an AFFE with a graph-like structure. According to Definition 4, the same argument from an AFFE could appear in different branches of a potential dialectical tree built from it. In contrast, this is not the case with our AFFE construction process, since all arguments in the AFFE (which, as stated before, coincides with a potential dialectical tree) are different. Notwithstanding this, the behavior of having the same argument in different branches of the tree can be achieved by having different arguments in alternative branches with exactly the same pieces of related evidence and equivalent sub-trees (*i.e.,* equivalent attackers, attackers of those attackers, and so on). So, in summary, our AFFE generation process has the advantage of simplifying the construction of potential dialectical trees and ensuring the amount of nodes per tree, while keeping the behavior of general graph-like structures.

For each generated AFFE or potential tree and scenario, we built the pruned active dialectical trees following the **CGPT**, **NGPT** and **SGPT** approaches. Finally, for each combination of parameters, we obtained the average cost of building an active tree under each approach (referred to as `TreeEvidCost`), where the cost associated with the construction of an active tree is determined by adding up the cost of the retrieved pieces of evidence and the cost the pieces of evidence found missing (*i.e.,* pieces of evidence with failed retrieval attempts). Next, we contrast the results obtained for our approach against those for the two baselines, illustrated in Figures 7–12.

When contrasting **CGPT** vs. **NGPT**, the tests showed that **CGPT** leads to obtaining a significant reduction of `TreeEvidCost` in all cases. In general, the cost reduction obtained with **CGPT** over **NGPT** increases with the tree size. This reduction also increases on trees with a high branching factor (`MaxBranchFactor`). Furthermore, the deactivation quota (`DeactQuota`) also affected the results, with smaller percentages of missing evidence leading to obtain greater cost reductions. Finally, even though a variation of other parameters affected the results regarding `TreeEvidCost`, they were not as significant as the ones obtained with the previously discussed parameters. For instance, a change in `MaxEvidCost` did not seemingly affect the results, leading to obtain similar values of `TreeEvidCost` for different tree sizes; this was also the case when varying the evidence universe size (`EvidUnivSize`). On the other hand, the tests showed that increasing the amount of evidence per argument (`MaxArgEvid`) reduces, although not significantly, the gain associated with `TreeEvidCost`.

The results of the comparison between **CGPT** and **SGPT** indicate that, in general, our approach leads to obtaining lower values of `TreeEvidCost` than those obtained with **SGPT**. However, these results do not show, in general, tendencies as sharp as those observed when contrasting **CGPT** with **NGPT**. In particular, the tests revealed that `MaxBranchFactor` is still a key parameter, with better results obtained as the trees' branching factor increased. Also, like in the previous case, the maximum cost associated with each piece of evidence

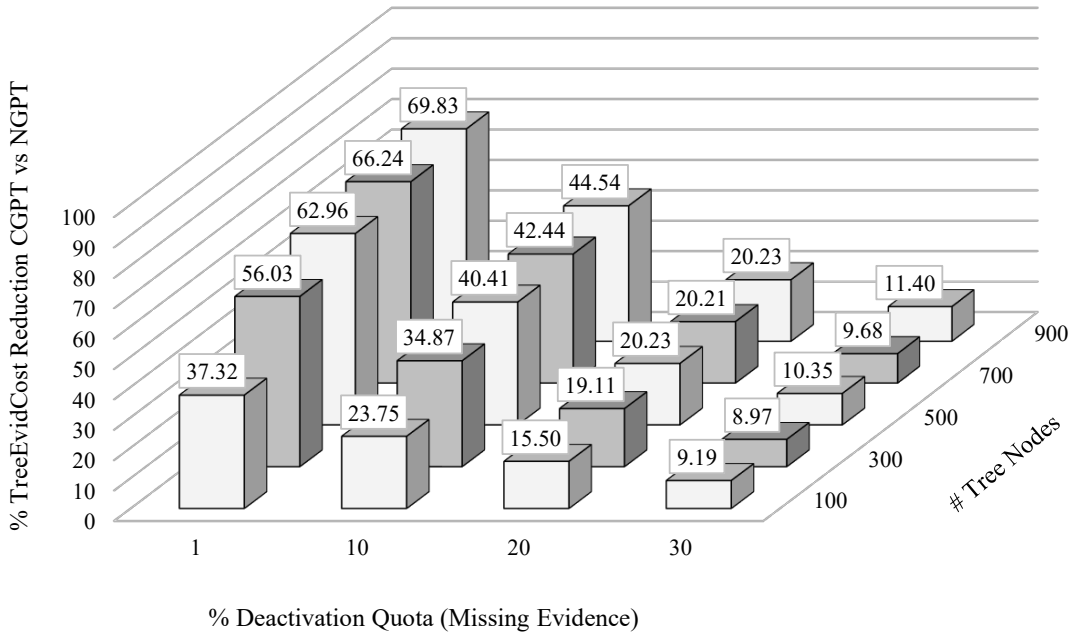

Figure 7: **CGPT** vs. **NGPT**, with `MaxBranchFactor`= 5, `EvidUnivSize`= 0.5, `MaxArgEvid` = 10, `MaxEvidCost` = 10

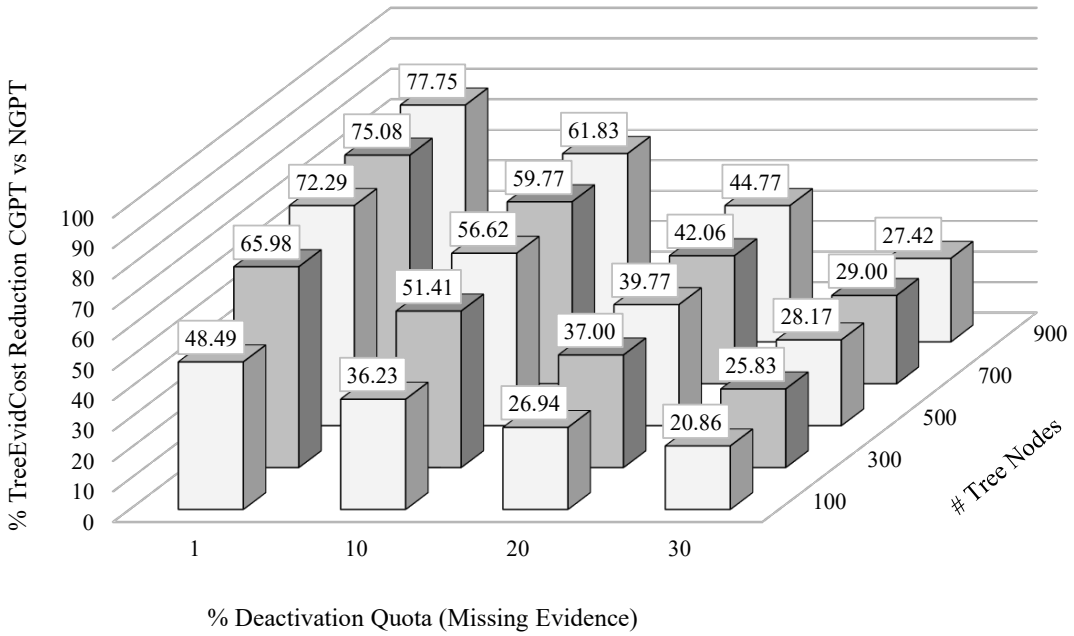

Figure 8: **CGPT** vs. **NGPT**, with `MaxBranchFactor` = 10, `EvidUnivSize`= 0.5, `MaxArgEvid` = 10, `MaxEvidCost`= 10

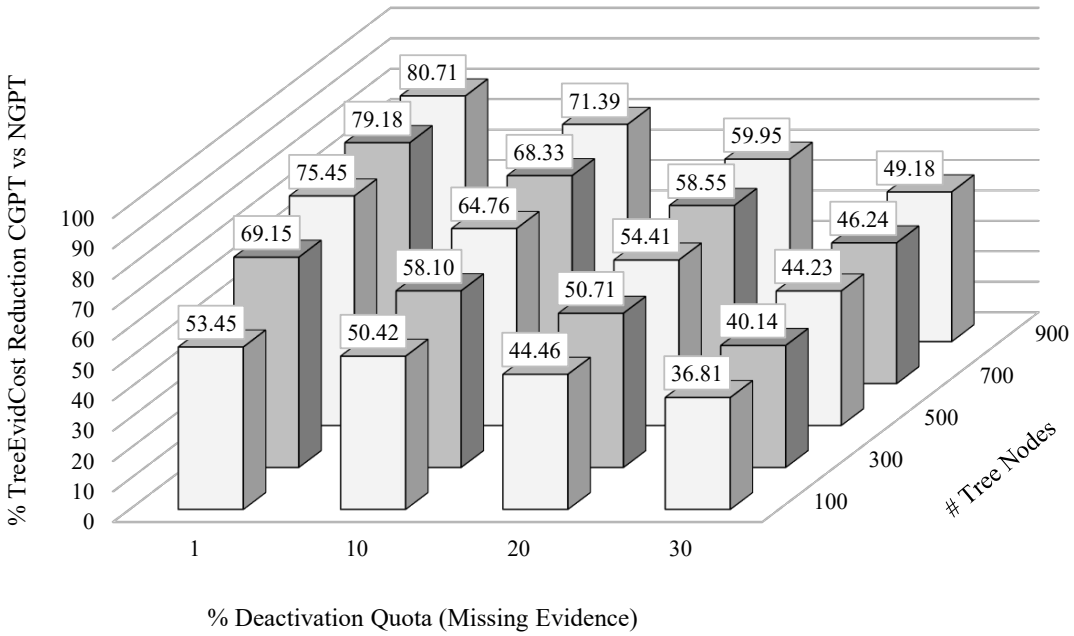

Figure 9: **CGPT** vs. **NGPT**, with `MaxBranchFactor` $= 20$, `EvidUnivSize` $= 0.5$, `MaxArgEvid` $= 10$, `MaxEvidCost` $= 10$

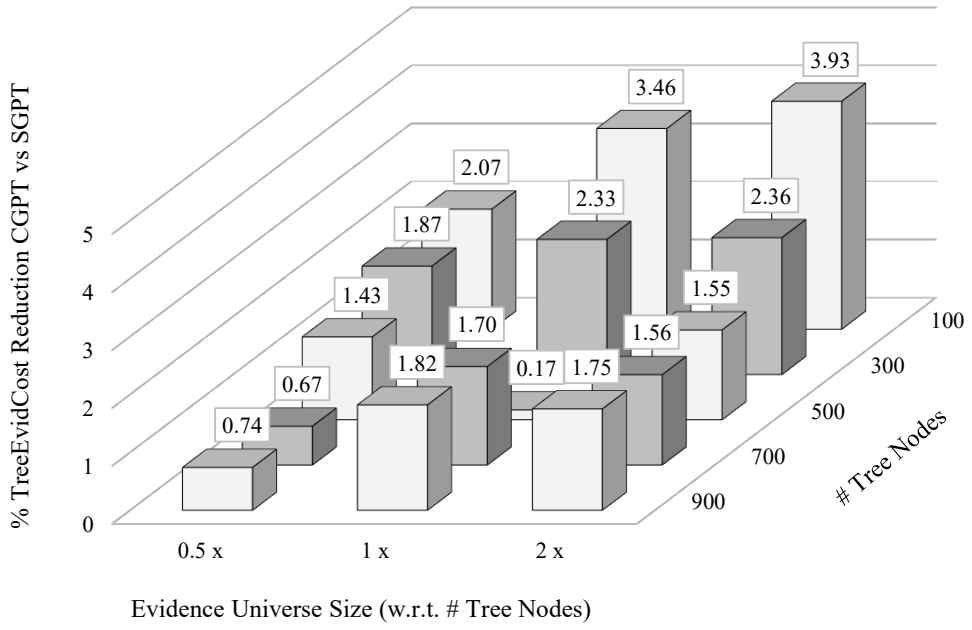

Figure 10: **CGPT** vs. **SGPT**, with `MaxBranchFactor` $= 5$, `DeactQuota` $= 10$, `MaxArgEvid` $= 10$, `MaxEvidCost` $= 10$

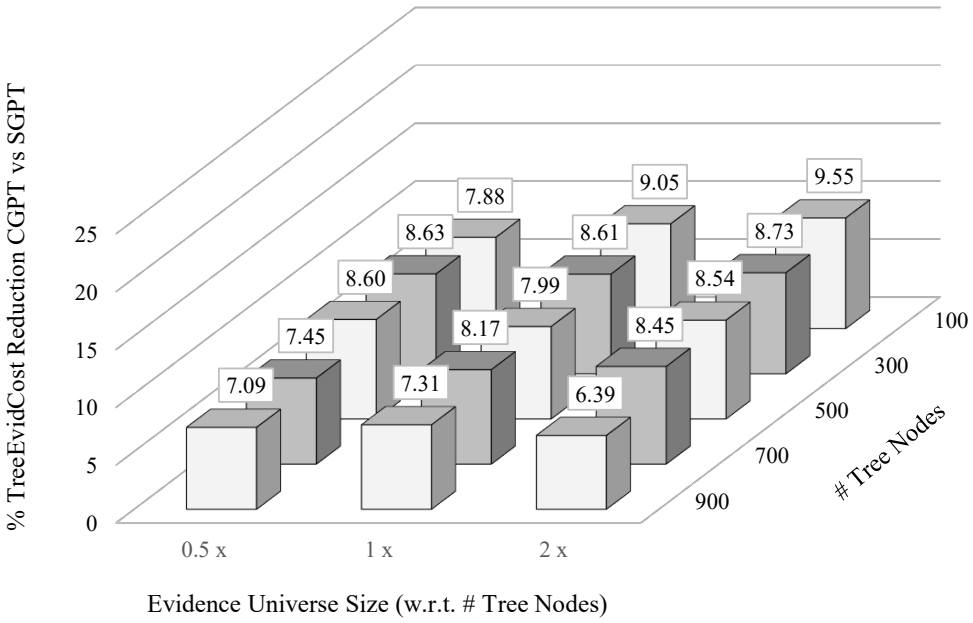

Figure 11: **CGPT** vs. **SGPT**, with `MaxBranchFactor` = 10, `DeactQuota` = 10, `MaxArgEvid` = 10, `MaxEvidCost` = 10

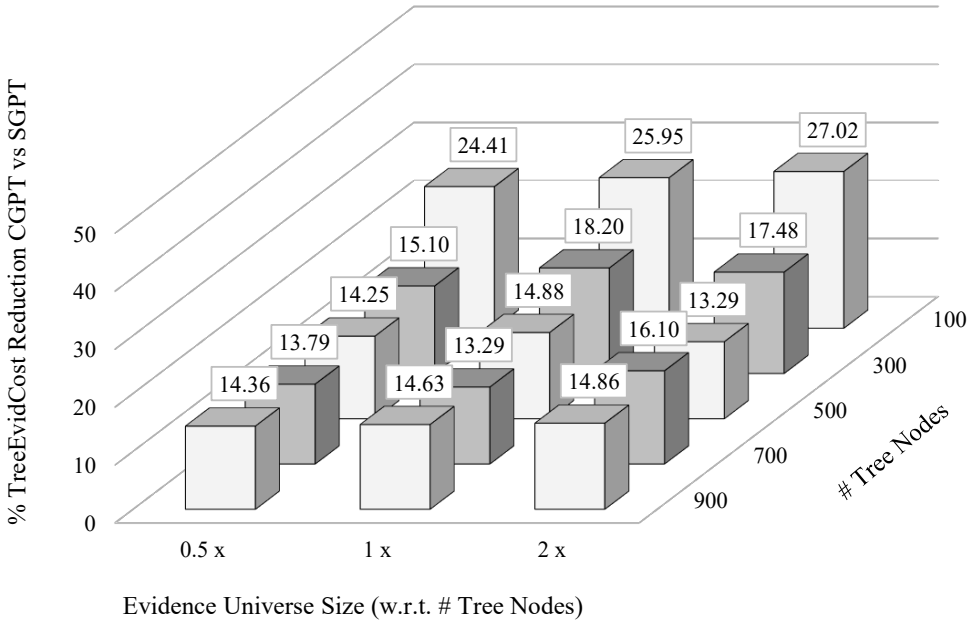

Figure 12: **CGPT** vs. **SGPT**, with `MaxBranchFactor` = 20, `DeactQuota` = 10, `MaxArgEvid` = 10, `MaxEvidCost` = 10

did not significantly affect the results, since similar gains were obtained when varying the `MaxEvidCost` parameter over tests considering the same tree size. On the other hand it was noted that, differently from before, better results (*i.e.,* greater cost reductions) were obtained for smaller trees. In general, a smaller amount of evidence per argument (*i.e.,* a lower `MaxArgEvid` value) resulted in the **CGPT** approach increasing the cost reduction over the **SGPT** approach. Differently from before, the deactivation quota (`DeactQuota`) was not a key parameter. This is because, when considering a branching factor of 10 or 20, the cost reduction of **CGPT** over **SGPT** was more or less the same for trees of the same size, regardless of the deactivation quota percentage. Moreover, it was observed that the cost reduction did not vary much when considering the same deactivation quota and different tree sizes, with a branching factor of 5 or 10. Then, for trees with a low branching factor (`MaxBranchFactor` = 5) and a nearly insignificant percentage of missing evidence (`DeactQuota` = 1), the **SGPT** approach outperformed ours. Nevertheless, remarkably, among all the tests performed (*i.e.,* by considering every combination and variation of the parameters), this setting was the only one where our approach did not yield a cost reduction with respect to **NGPT** or **SGPT**. Finally, similarly to before, except from isolated cases, variations in the evidence universe size (`EvidUnivSize` parameter) did not affect the results.

The most significant results of the comparison between **CGPT** and **NGPT** are illustrated in Figures 7–9, and correspond to tests varying the `MaxBranchFactor`, `TreeNodeCount` and `DeactQuota` parameters. On the other hand, Figures 10–12 illustrate the test cases where the highest cost reductions of **CGPT** over **SGPT** were obtained, with variations of the `MaxBranchFactor`, `TreeNodeCount` and `EvidUnivSize` parameters.

It can be noted that the branching factor and the tree size are the two common parameters that impact the performance of our approach against the two baselines. Thus, we will end this section by discussing the reasons why we believe this is the case. On average, bigger trees offer more opportunities for pruning. In addition, increasing the branching factor over trees having the same size yields wider and less deep trees; therefore, each prune performed over such trees has the effect of avoiding the exploration of a bigger amount of arguments (the wider sub-trees) and thus, avoiding to pay their (higher) evidence retrieval costs. Taking this into account, it is not surprising that **CGPT** outperforms **NGPT**, and that the advantages of our approach are further exploited for bigger trees and higher branching factor values. This is because, in general, bigger trees are associated with higher evidence retrieval costs. Thus, guiding the exploration of arguments by their heuristic evidential cost offers the possibility of early pruning those branches with higher retrieval costs.

In contrast, when considering the outcome relating **CGPT** and **SGPT**, we note that the improvement of our approach over the strength-guided one is smaller, and that **SGPT** bridges the gap as trees become bigger. We believe that this has to do with the fact that **SGPT** guides the exploration of arguments by their strength, avoiding to choose arguments that would end up being defeated (*i.e.,* marked as **D** in the corresponding dialectical tree) and avoiding to unnecessarily pay the evidence retrieval cost of their corresponding subtrees; as a result, **SGPT** has the effect of pruning more siblings of each argument at a time. Consequently, for bigger trees, the **SGPT** strategy compensates a little for the non-consideration of the evidence retrieval costs. Notwithstanding this, when increasing the branching factor, the children of an argument are more likely to have similar strength values and thus, **SGPT** cannot make up for the non-consideration of evidence retrieval

costs. This is because, when having to choose between two or more arguments with the same strength, **SGPT** would select any of them (somehow randomly), whereas **CGPT** will always aim at minimizing the evidence retrieval cost. Therefore, as in the case of the comparison between **CGPT** and **NGPT**, higher branching factors allow to further exploit the advantages of our approach.

6. Related Work and Conclusions

In this paper we proposed the Abstract Argumentation Framework with Fallible Evidence (AFFE), an extension of Dung's framework (Dung, 1995) which accounts for the fact that arguments should be backed by evidence, and that each piece of evidence associated with an argument comes at a cost that has to be paid in order to (attempt to) retrieve it. Moreover, since it might be the case that some evidence cannot be retrieved (*e.g.,* in the case of an online article, because the corresponding server may be down), each argument can be active or inactive in a given scenario; we equipped the AFFE with a function enabling it to determine whether each piece of evidence can be retrieved or not in different sessions, which can represent different scenarios or time-frames. Taking this into account, we proposed a heuristic measure to be used as part of the acceptability calculus of an argumentation system based on the construction of dialectical trees.

Specifically, we used the heuristic measure to select the order in which arguments are explored for building the active dialectical trees (*i.e.,* the dialectical trees considering only active arguments), with the aim of pruning and minimizing the arguments' evidence retrieval cost in a given scenario. As mentioned before and shown in (Skiba et al., 2020), finding the optimal set of evidence that should be attempted to be retrieved in order to guarantee the acceptance status of a given argument, thus yielding the minimum evidence retrieval cost, is computationally hard. Consequently, heuristic-based approaches like ours become handy and could provide an appealing method for knowledge representation and reasoning in application scenarios where the consideration of evidence is natural, such as online forum debates or discussions.

We formally showed the correctness of our approach, and we empirically tested it against two baselines, namely a non-guided pruning technique (**NGPT**) and the strength-guided pruning technique (**SGPT**) proposed in (Rotstein et al., 2010). The results showed that our cost-guided pruning technique (**CGPT**) significantly outperforms the non-guided approach in all cases. In contrast, even though the evidence retrieval cost with **CGPT** is lower than that incurred with **SGPT**, the improvement is not as significant as that obtained against the non-guided approach. As discussed in Section 5, this has to do with the fact that **SGPT** aims at building smaller trees, seeking to explore the undefeated arguments first. Notwithstanding this, our approach still offers the advantage of yielding lower evidence retrieval costs than the other two baselines.

We want to highlight the main differences between this paper and (Cohen et al., 2019). In this work we improved the theoretical basis section, reshaping definitions and formalizing some notions that were only introduced intuitively in (Cohen et al., 2019), as well as changing the definition of the framework our approach is based on (*i.e.,* the AFFE). We added new examples and expanded the ones in (Cohen et al., 2019), and formally showed the correctness of our approach showing that it indeed returns a tree structure corresponding

to a pruned active dialectical tree. In addition, we presented and discussed the algorithms developed for our heuristic pruning technique. Also, regarding the empirical validation of our proposal, we incorporated a new section introducing and detailedly discussing the experiments we carried out and the results we obtained, where we tested our approach against two baselines. Furthermore, we added a detailed description of the generation process for the frameworks used in our experiments.

It should be noted that, since our proposal is such that the same piece of evidence may be available or unavailable in different sessions (leading to different sets of active arguments in each case), it has an inherently dynamic component. Recently, there has been an increasing interest in studying the dynamic nature of argumentation (Doutre & Mailly, 2018), and this has also become evident with the inclusion of a dedicated track in the latest editions of the *International Competition of Computational Models of Argumentation* (*ICCMA*)[4]. In particular, approaches like (Liao et al., 2011; Baroni et al., 2014; Alfano et al., 2017; Niskanen & Järvisalo, 2020) address the incremental recomputation of extensions of a Dung's abstract argumentation framework after some updates have been performed; moreover, this line of work has been further extended to consider extensions of Dung's framework which incorporate support relations or high-order interactions (Alfano et al., 2018, 2020, 2020). Our approach aligns with these works in the sense that, in order to determine the acceptance status of a given argument, we only seek to account for those arguments (also, the interactions) that affect it. However, since those works are aimed at identifying sets of extensions of a given framework (hence, the acceptance status of *every* argument in the framework), to determine the acceptance status of an argument $\mathcal{A}$ they may also require to consider (thus, attempt to build) arguments whose acceptance status is affected by $\mathcal{A}$, but which do not affect the acceptance status of $\mathcal{A}$; consequently, they may incur unnecessary costs.

Another difference between our proposal and the works mentioned above is that, in the other approaches, after the updates have been performed, the entire set of arguments of the framework is known to be active. In contrast, in our approach, the dynamic component is the available evidence; then, since the changes in the set of available evidence are not known beforehand, we still have to attempt to retrieve the evidence associated with the arguments, not knowing whether they will be active or not. As a result, our proposal relates to works like (Capobianco et al., 2005) and (Rotstein et al., 2010) that put more focus on how the set of active arguments changes. In particular, our work is most closely related to (Rotstein et al., 2010), since we considered the variation of the available evidence, determining the active arguments in each session.

Despite the fact that extension-based approaches for Dung's abstract argumentation frameworks may require to consider arguments in the framework that do not affect the acceptance status of the queried argument, extension-based approaches are the most widely used within the community of argumentation. In addition to the computation of extensions, another common task tackled by argumentation systems is the computation of skeptical or credulous acceptance of a single argument under a given semantics (Dung et al., 2007; Baumeister et al., 2021). For that purpose, dialectical proof procedures for determining credulous or skeptical acceptance of a single argument under some of the original

4. `http://argumentationcompetition.org`

Dung semantics were proposed; examples of such approaches are (Dung & Thang, 2009) and (Thang et al., 2009). With this in mind, in the future, to minimize the evidence retrieval cost and reduce the search space by discarding irrelevant arguments, we will seek to apply our heuristics-based pruning technique to those procedures.

Finally, as it was discussed before, our approach relates to (Gottifredi et al., 2013), where another heuristic strategy guiding the construction of dialectical trees was proposed. Our tests revealed that our heuristic measure outperforms the one proposed in (Gottifredi et al., 2013) in most cases, being more suitable for reducing the evidence retrieval costs. Notwithstanding this, as future work, we plan to explore an alternative heuristic measure that will combine the intuitions of our approach and those associated with the notion of argument strength.

Acknowledgments

The authors would like to thank Matthias Thimm for the helpful and insightful comments and discussions. This research was partially supported by PGI-UNS under grants 24/N046 and 24/ZN37.

References

Al-Abdulkarim, L., Atkinson, K., & Bench-Capon, T. J. M. (2014). Abstract dialectical frameworks for legal reasoning. In *Proc. of JURIX*, pp. 61–70.

Alfano, G., Cohen, A., Gottifredi, S., Greco, S., Parisi, F., & Simari, G. R. (2020). Dynamics in abstract argumentation frameworks with recursive attack and support relations. In *Proc. of ECAI*, pp. 577–584.

Alfano, G., Greco, S., & Parisi, F. (2017). Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach. In *Proc. of IJCAI*, pp. 49–55.

Alfano, G., Greco, S., & Parisi, F. (2018). A meta-argumentation approach for the efficient computation of stable and preferred extensions in dynamic bipolar argumentation frameworks. *Intelligenza Artificiale*, *12*(2), 193–211.

Alfano, G., Greco, S., & Parisi, F. (2020). Computing skeptical preferred acceptance in dynamic argumentation frameworks with recursive attack and support relations. In *Proc. of COMMA*, pp. 67–78.

Baroni, P., Caminada, M., & Giacomin, M. (2018). Abstract argumentation frameworks and their semantics. In Baroni, P., Giacomin, M., & van der Torre, L. (Eds.), *Handbook of Formal Argumentation - Volume 1*, pp. 159–236. College Publications, London.

Baroni, P., Giacomin, M., & Liao, B. (2014). On topology-related properties of abstract argumentation semantics. A correction and extension to dynamics of argumentation systems: A division-based method. *Artificial Intelligence*, *212*, 104–115.

Baumeister, D., Järvisalo, M., Neugebauer, D., Niskanen, A., & Rothe, J. (2021). Acceptance in incomplete argumentation frameworks. *Artificial Intelligence*, *295*, 103470.

Bedi, P., & Vashisth, P. B. (2014). Empowering recommender systems using trust and argumentation. *Information Sciences*, *279*, 569–586.

Besnard, P., García, A. J., Hunter, A., Modgil, S., Prakken, H., Simari, G. R., & Toni, F. (2014). Introduction to structured argumentation. *Argument & Computation*, *5*(1), 1–4.

Besnard, P., & Hunter, A. (2001). A logic-based theory of deductive arguments. *Artificial Intelligence*, *128*(1-2), 203–235.

Black, E., & Hunter, A. (2009). An inquiry dialogue system. *Autonomous Agents and Multi-Agent Systems*, *19*(2), 173–209.

Briguez, C. E., Budán, M. C. D., Deagustini, C. A. D., Maguitman, A. G., Capobianco, M., & Simari, G. R. (2014). Argument-based mixed recommenders and their application to movie suggestion. *Expert Systems with Applications*, *41*(14), 6467–6482.

Capobianco, M., Chesñevar, C. I., & Simari, G. R. (2005). Argumentation and the dynamics of warranted beliefs in changing environments. *JAAMAS*, *11*, 127–151.

Chesñevar, C. I., Simari, G. R., & García, A. J. (2000). Pruning search space in defeasible argumentation. In *Proc. of ATAI*, pp. 46–55.

Cohen, A., Gottifredi, S., & García, A. J. (2019). A heuristic pruning technique for dialectical trees on argumentation-based query-answering systems. In *Proc. of FQAS*, pp. 101–113.

Deagustini, C. A. D., Dalibón, S. E. F., Gottifredi, S., Falappa, M. A., Chesñevar, C. I., & Simari, G. R. (2017). Defeasible argumentation over relational databases. *Argument & Computation*, *8*(1), 35–59.

Deagustini, C. A. D., Dalibón, S. E. F., Gottifredi, S., Falappa, M. A., Chesñevar, C. I., & Simari, G. R. (2013). Relational databases as a massive information source for defeasible argumentation. *Knowledge-Based Systems*, *51*, 93–109.

Doutre, S., & Mailly, J. (2018). Constraints and changes: A survey of abstract argumentation dynamics. *Argument & Computation*, *9*(3), 223–248.

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, *77*(2), 321–358.

Dung, P. M., Mancarella, P., & Toni, F. (2007). Computing ideal sceptical argumentation. *Artificial Intelligence*, *171*(10-15), 642–674.

Dung, P. M., & Thang, P. M. (2009). A unified framework for representation and development of dialectical proof procedures in argumentation. In *Proc. of IJCAI*, pp. 746–751.

Dvorák, W., & Dunne, P. E. (2018). Computational problems in formal argumentation and their complexity. In Baroni, P., Giacomin, M., & van der Torre, L. (Eds.), *Handbook of Formal Argumentation - Volume 1*, pp. 631–687. College Publications, London.

Ferretti, E., Tamargo, L. H., García, A. J., Errecalde, M. L., & Simari, G. R. (2017). An approach to decision making based on dynamic argumentation systems. *Artificial Intelligence*, *242*, 107–131.

García, A. J., Prakken, H., & Simari, G. R. (2020). A comparative study of some central notions of ASPIC+ and delp. *Theory and Practice of Logic Programming*, *20*(3), 358–390.

García, A. J., & Simari, G. R. (2004). Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, *4*(1-2), 95–138.

Gómez, S. A., Goron, A., Groza, A., & Letia, I. A. (2016). Assuring safety in air traffic control systems with argumentation and model checking. *Expert Systems with Applications*, *44*, 367–385.

Gottifredi, S., Rotstein, N. D., García, A. J., & Simari, G. R. (2013). Using argument strength for building dialectical bonsai. *Annals of Mathematics and Artificial Intelligence*, *69*(1), 103–129.

Liao, B., Jin, L., & Koons, R. C. (2011). Dynamics of argumentation systems: A division-based method. *Artificial Intelligence*, *175*(11), 1790–1814.

Lippi, M., & Torroni, P. (2016). Argumentation mining: State of the art and emerging trends. *ACM Transactions on Internet Technology*, *16*(2), 10:1–10:25.

Niskanen, A., & Järvisalo, M. (2020). Algorithms for dynamic argumentation frameworks: An incremental sat-based approach. In *Proc. of ECAI*, pp. 849–856.

Prakken, H., & Sartor, G. (2015). Law and logic: A review from an argumentation perspective. *Artificial Intelligence*, *227*, 214–245.

Rotstein, N. D., Moguillansky, M. O., García, A. J., & Simari, G. R. (2010). A dynamic argumentation framework. In *Proc. of COMMA*, pp. 427–438.

Skiba, K., Thimm, M., Cohen, A., Gottifredi, S., & García, A. J. (2020). Abstract argumentation frameworks with fallible evidence. In *Proc. of COMMA*, pp. 347–354.

Thang, P. M., Minh, P., & Hung, N. D. (2009). Towards a common framework for dialectical proof procedures in abstract argumentation. *Journal of Logic and Computation*, *19*(6), 1071–1109.