Computational Models of Argument
P. Baroni et al. (Eds.)

doi:10.3233/978-1-61499-686-6-107

On Efficiently Enumerating Semi-Stable Extensions via Dynamic Programming on Tree Decompositions

Bernhard BLIEM [a] Markus HECHER [a], Stefan WOLTRAN [a]
[a] *Institute of Information Systems, TU Wien, Austria*

Abstract. Many computational problems in the area of abstract argumentation are intractable. For some semantics like preferred and semi-stable, important decision problems can even be hard for classes of the second level of the polynomial hierarchy. One approach to deal with this inherent difficulty is to exploit structure of argumentation frameworks. In particular, algorithms that run in linear time for argumentation frameworks of bounded treewidth have been proposed for several semantics. In this paper, we contribute to this line of research and propose a novel algorithm for the semi-stable semantics. We also present an implementation of the algorithm and report on some experimental results.

Keywords. Abstract Argumentation, Fixed-Parameter Tractability, Dynamic Programming on Tree Decompositions

1. Introduction

Dung's abstract argumentation frameworks [5] are a central concept in many argumentation formalisms and systems. Efficient and versatile methods for abstract argumentation are therefore important for further advances in the field. For some important semantics like the preferred and semi-stable extensions (see [7,12]), the high worst-case complexity is a major obstacle to finding algorithms that evaluate argumentation frameworks (AFs) of real-world size in reasonable time. In fact, standard algorithms tend to be problematic for larger instances even if the inherent structure of the frameworks remains simple; a situation that is likely to appear when frameworks are obtained during some instantiation process. It is thus valuable to design alternative algorithms, where the size of the framework has less influence on the runtime.

The field of parameterized complexity theory [4] formally captures this intuition and is based on the following observation: Many hard problems become tractable if some problem parameter is bounded by a constant. This property is referred to as *fixed-parameter tractability* (FPT). One important parameter of graphs is the treewidth, which measures the "tree-likeness" of a graph and is thus also applicable to AFs. In the field of argumentation, research in this direction was initiated by Dunne [6] who showed that many intractable problems can be solved in linear time for argumentation frameworks of bounded treewidth. Later these results were extended to the more general structural parameter of clique-width [9]. Further parameterized complexity results include [8,15].

Showing that a problem parameterized by treewidth is FPT often does not yield a practically useful algorithm automatically. For obtaining such algorithms, dynamic programming (DP) algorithms that operate on a tree decomposition (TD) of the input usually have to be designed. For admissible, preferred, and ideal semantics, such algorithms have been presented in [11], and the system DYNPARTIX [3] implements algorithms for admissible, preferred, stable and complete semantics. However, semi-stable semantics has not been considered so far. This semantics is challenging, as it is among the most complex ones for abstract argumentation, with credulous acceptance being Σ_2^{P}-complete and skeptical acceptance being Π_2^{P}-complete [12].

In this work, we present a DP algorithm that computes semi-stable extensions in linear time on AFs of bounded treewidth. We briefly report on an implementation of our algorithms and some experimental evaluation.

2. Background

Abstract Argumentation. We first review the Dung argumentation framework [5].

Definition 2.1. *An* argumentation framework *(AF) is a pair $F = (A_F, R_F)$, where A_F is a set of* arguments *and $R_F \subseteq A \times A$ is a set of* attacks. *Instead of $(a,b) \in R_F$, we write $a \rightarrowtail^{R_F} b$, and we sometimes omit R_F if it is clear from the context. For any set $S \subseteq A_F$, we write $S \rightarrowtail^{R_F} b$ if there is some $a \in S$ s.t. $a \rightarrowtail^{R_F} b$. We say that a is* defended *by S if $S \rightarrowtail^{R_F} b$ for each $b \in A_F$ with $b \rightarrowtail^{R_F} a$. We call $S^+_{R_F} = S \cup \{b \mid S \rightarrowtail^{R_F} b\}$ the* range *of S.*

A semantics characterizes so-called *extensions* of an AF, i.e., sets of arguments that are acceptable. For a semantics $\psi \in \{\text{conflict-free}, \text{admissible}, \text{preferred}, \text{semi-stable}, \text{stable}\}$ and an AF F, we write $\psi(F)$ to denote the set of ψ-extensions in F.

Definition 2.2. *Let F be an AF and $S \subseteq A_F$. We define (a) $S \in \text{conflict-free}(F)$ if there are no $a,b \in S$ with $a \rightarrowtail^{R_F} b$; (b) $S \in \text{admissible}(F)$ if $S \in \text{conflict-free}(F)$ and each $a \in S$ is defended by S; (c) $S \in \text{preferred}(F)$ if $S \in \text{admissible}(F)$ and $S' \not\supset S$ holds for each $S' \in \text{admissible}(F)$; (d) $S \in \text{semi-stable}(F)$ if $S \in \text{admissible}(F)$ and $S'^+_{R_F} \not\supset S^+_{R_F}$ holds for each $S' \in \text{admissible}(F)$; (e) $S \in \text{stable}(F)$ if $S \in \text{conflict-free}(F)$ and $A_F = S^+_{R_F}$.*

One can show that for every AF F it holds that $\text{stable}(F) \subseteq \text{semi-stable}(F) \subseteq \text{preferred}(F) \subseteq \text{admissible}(F) \subseteq \text{conflict-free}(F)$.

Tree decompositions (TDs). A parameterized problem is a problem whose instances are accompanied by an integer that represents a certain parameter of the instance. Such a problem is called *fixed-parameter tractable* (FPT) if it is solvable in time $f(k) \cdot n^{\mathcal{O}(1)}$, where n is the input size and f is a function that only depends on the value k of the parameter [4]. The parameter we consider is *treewidth*, which is defined by means of tree decompositions, originally introduced in [18]. The intuition behind TDs is to obtain a tree from a (potentially cyclic) graph by subsuming multiple vertices under one node and thereby isolating the parts responsible for cyclicity.

Definition 2.3. *A* tree decomposition *of a graph $G = (V,E)$ is a pair $T = (\mathcal{T}, \mathcal{X})$ where $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ is a (rooted) tree and $\mathcal{X} = \{X_{t_1}, \ldots, X_{t_n}\}$ assigns to each node $t \in V_{\mathcal{T}}$ a subset X_t of V (called the* bag *of t) as follows: (1) For each vertex $v \in V$, there is a node*

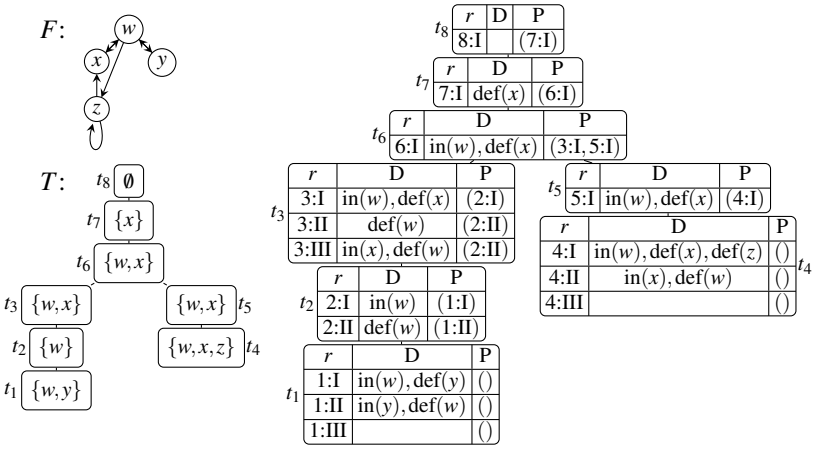

Figure 1: DP computation of stable(F) w.r.t. normalized TD $T = (\mathscr{T}, \mathscr{X})$.

$t \in V_{\mathscr{T}}$ such that $v \in X_t$; (2) For each edge $e \in E$, there is a node $t \in V_{\mathscr{T}}$ such that $e \subseteq X_t$; (3) For each $v \in V$, the subgraph of $\mathscr{T}$ induced by $\{t \in V_{\mathscr{T}} \mid v \in X_t\}$ is connected. We call $\max_{t \in V_{\mathscr{T}}} |X_t| - 1$ *the* width *of T. The* treewidth *of G is the minimum width over all its TDs.*

For $\mathscr{T} = (V_{\mathscr{T}}, E_{\mathscr{T}})$ we often write $t \in \mathscr{T}$ instead of $t \in V_{\mathscr{T}}$. We only consider TDs of the following form that can be achieved in linear time without increasing the width [16].

Definition 2.4. *We call a TD $(\mathscr{T}, \mathscr{X})$* normalized *if its root has an empty bag and each node $t \in \mathscr{T}$ is of one of the following types.* LEAF: *t is a leaf of $\mathscr{T}$.* FORGET: *t has only one child t' and $X_t = X_{t'} \setminus \{v\}$ for some $v \in X_{t'}$.* INSERT: *t has only one child t' and $X_t = X_{t'} \cup \{v\}$ for some $v \notin X_{t'}$.* JOIN: *t has two children t', t'' and $X_t = X_{t'} = X_{t''}$.*

Example 2.5. *Figure 1 depicts a normalized TD T (having width 2) of the AF F.*

Dynamic programming (DP) on Tree decompositions. Algorithms for dynamic programming on TDs generally traverse the TD in bottom-up order. At each node, partial solutions for the subgraph induced by the vertices encountered so far are computed and stored in a table associated with the node (cf. [17]). The size of each table is typically bounded by the width of the TD, and the number of TD nodes is linear in the input size. Hence, if the width is bounded by a constant, the search space for the subproblem is constant as well, and the number of subproblems only grows by a linear factor for larger instances. Each row in such a table corresponds to partial solutions that take only a part of the instance into account, namely the part of the instance that has been "encountered" during the bottom-up traversal:

Definition 2.6. *Let F be an AF, $(\mathscr{T}, \mathscr{X})$ be a TD of F, and $t \in \mathscr{T}$. We use $X_{\geq t}$ to denote the union of all bags $X_s \in \mathscr{X}$ such that s occurs in the subtree of $\mathscr{T}$ rooted at t. Moreover, $X_{>t}$ denotes $X_{\geq t} \setminus X_t$. We define F_t as the AF $(A_F \cap X_t, R_F \cap X_t^2)$ and call it the* subframework in *t. Finally, we define $F_{\geq t}$ as the AF $(A_F \cap X_{\geq t}, R_F \cap X_{\geq t}^2)$ and call it the* subframework induced by *t. (Note that $F_{\geq t} = F$ holds if t is the root of the TD.)*

Example 2.7. *We illustrate the DP for stable extensions of the example AF F in Figure 1. The tables in that figure are computed as follows. For a TD node t, each table row r*

consists of data $\mathrm{D}(r)$*, which may assign a status to arguments in* X_t*. For any argument* a*,* $\mathrm{D}(r)$ *contains* $in(a)$ *or* $def(a)$ *if for each set* S *of arguments represented by* r *it holds that* $a \in S$ *or* $S \rightarrowtail a$*, respectively. The set* $\mathrm{P}(r)$ *contains so-called extension pointer tuples (EPTs) that denote the rows in the children of* t *where* r *was constructed from. By following these pointers, we can obtain the sets represented by* r*.*

We make sure that all sets represented by a row are conflict-free. For instance, at node t_1*,* $X_{t_1} = \{w, y\}$ *holds and these arguments attack each other. Hence the table at* t_1 *contains a row for each of the three conflict-free subsets of* X_{t_1}*. At* t_2*, the child rows are extended and the status assignment is updated by removing the status of arguments that are not contained in* X_{t_2}*. Observe that row* $1{:}III$ *is not extended by any row at* t_2*. This is because the removed argument* y *(i.e.,* $y \in X_{t_1} \setminus X_{t_2}$*) has neither status "in" nor "def": Any solution* S *that is constructed using* $1{:}III$ *would satisfy neither* $y \in S$ *nor* $S \rightarrowtail y$*, so* S *would not be a stable extension. At* t_3*, we extend child rows and guess a status for the introduced argument* x*. Note that we must discard rows containing both* $in(w)$ *and* $in(x)$ *because otherwise the partial solutions represented by such rows would not be conflict-free. At* t_6*, only rows that agree on the status of the common arguments may be joined. We continue this procedure recursively until we reach the TD's root.*

To decide whether there is a stable extension, it suffices to check if the table in the root node is non-empty. The overall procedure is in FPT time because the number of nodes in the TD is bounded by the input size, and each node t *is associated with a table of size at most* $\mathcal{O}(3^{|X_t|})$ *(i.e., the number of possible status assignments). The AF* F *has a stable extension due to existence of row* $8{:}I$*. Solutions (stable extensions of* F*) can be enumerated with linear delay by starting at the root and following the EPTs while collecting arguments with status "in" according to the extended rows. Solution* $\{w\}$ *is constructed by starting at* $8{:}I$ *and following EPTs* $(7{:}I)$*,* $(6{:}I)$*,* $(3{:}I, 5{:}I)$*,* $(2{:}I)$*,* $(1{:}I)$ *and* $(4{:}I)$*. The union of the arguments having status "in" according to these rows is* $\{w\}$*. It is easy to see that* $\{w\}$ *is the only stable extension of* F*.*

3. Algorithm for Admissible Semantics

We first provide an algorithm for admissible semantics that uses DP on TDs, modifying concepts from [11], and then extend it to semi-stable semantics in Section 4. The adaption is needed, since we require to distinguish partial solutions not only with respect to the status of already processed arguments, but have to guess whether arguments might be attacked or not by arguments appearing later. As we will see in Section 4, this allows us to relate partial solutions to those which possess a larger range. For space reasons, we only provide proof sketches and refer to [14] for full proofs. First we adapt the concept of restricted-admissible sets from [11] for our purposes.

Definition 3.1. *Let* $F = (A_F, R_F)$ *be an AF and* B *a set of arguments. A tuple* (S, D) *satisfying* $S, D \subseteq A_F$ *and* $S \cap D = \emptyset$ *is a* B-restricted admissible tuple for F *if (1)* S *is conflict-free in* F *and* S *defends itself in* F *against all elements of* $A_F \cap B$*, and (2) for each* $a \in A_F$*, whenever* $S \rightarrowtail^{R_F} a$ *or* $a \rightarrowtail^{R_F} S$*, then* $a \in D$*. We call* S *a* B*-restricted admissible set for* F *if there is a set* D *such that* (S, D) *is a* B*-restricted admissible tuple for* F*.*

Note that for $A_F \subseteq B$, B-restricted admissible sets of AF $F = (A_F, R_F)$ are just admissible sets for F. For $A_F \cap B = \emptyset$, B-restricted admissible sets are just the conflict-free

sets for F. Intuitively, if (S,D) is a B-restricted admissible tuple, then D consists of at least those arguments (different from S) that are defeated or still require defeating by S.

Example 3.2. *Consider the framework F and TD T given in Figure 1. Let $F' = (A_{F'}, R_{F'})$ be the subframework induced by node t_3 of T minus the attack (y,w), i.e., $A_{F'} = \{w,x,y\}, R_{F'} = \{(w,x),(x,w),(w,y)\}$. The $\{x,y\}$-restricted admissible sets are $\emptyset$, $\{w\}$, $\{x\}$, $\{y\}$ and $\{x,y\}$. The set $\{y\}$, however, is not $\{w\}$-restricted admissible, since $w \rightarrowtail^{R_{F'}} y$ but y does not defend itself against w. From the stated $\{x,y\}$-restricted admissible sets we obtain $\{x,y\}$-restricted admissible tuples by adding the required second component. Since condition (2) of Definition 3.1 is always trivially satisfied if D contains all arguments, we only state the smallest sets D (w.r.t. subset inclusion) that satisfy condition (2). These are $(\emptyset,\emptyset)$, $(\{w\},\{x,y\})$, $(\{x\},\{w\})$, $(\{y\},\{w\})$ and $(\{x,y\},\{w\})$.*

The following concept of (valid) colorings helps to prove correctness of our algorithm.

Definition 3.3. *Let $(\mathscr{T},\mathscr{X})$ be a TD of an AF $F = (A_F, R_F)$ and $t \in \mathscr{T}$. We call $C : X_t \to \{in, attc, def, out\}$ a* coloring *and define $[C] = \{a \mid C(a) = in\}$ and $[[C]] = \{a \mid C(a) = def$ or $C(a) = attc\}$. Moreover, we define $e_t(C)$ as the collection of $X_{>t}$-restricted admissible tuples (S,D) for $F_{\geq t}$ that satisfy the following properties for each $a \in X_t$.*

$$
\begin{array}{rlll}
\textit{(i)} & C(a) = in & \iff & a \in S \\
\textit{(ii)} & C(a) = def & \iff & S \rightarrowtail^{R_F} a \\
\textit{(iii)} & C(a) = attc & \impliedby & S \not\rightarrowtail^{R_F} a \text{ and } a \rightarrowtail^{R_F} S \\
\textit{(iv)} & C(a) = out & \implies & S \not\rightarrowtail^{R_F} a \text{ and } a \not\rightarrowtail^{R_F} S \\
\textit{(v)} & C(a) \in \{def, attc\} & \iff & a \in D
\end{array}
$$

If $e_t(C) \neq \emptyset$, C is called a valid coloring *for t; $\mathscr{C}_t$ denotes the set of valid colorings for t. For convenience we use $e'_t(C) := \{S \mid (S,D) \in e_t(C)\}$.*

Intuitively, the color "in" means that the argument a is in the $X_{>t}$-restricted admissible set S and "def" means a is defeated by S. Arguments that attack S without being defeated (yet) have color "attc", but this color may also be assigned to other arguments that don't need to be colored "in" or "def". The color "attc" means that the argument is expected to be defeated in the future. Finally, all remaining arguments are assigned color "out". Any valid coloring C_t for a TD node t forms exactly one admissible tuple (S,D), with S being the $X_{>t}$-restricted admissible set and D being those arguments that are either defeated by S (color *def*) or attack S or shall be defeated by it (both color *attc*).

Example 3.4. *Let F and T be the AF and TD, respectively, from Figure 1. Observe that $X_{t_3} = \{w,x\}$, $X_{>t_3} = \{y\}$ and $F_{\geq t_3} = (\{w,x,y\}, \{(w,x),(x,w),(w,y),(y,w)\})$. Let C be the coloring for t_3 s.t. $C(w) = in$, $C(x) = def$. The only tuple in $e_{t_3}(C)$ that is $X_{>t_3}$-restricted admissible for $F_{\geq t_3}$ and satisfies the conditions of Definition 3.3 is $(\{w\},\{x,y\})$.*

We now show that valid colorings indeed correspond to admissible extensions.

Proposition 3.5. *Assuming that r is the root of a normalized TD of an AF F and ε is the valid coloring for r, we have that $e'_r(\varepsilon) = admissible(F)$.*

Proof sketch. Let r be the root of the TD (recall that $X_r = \emptyset$) and ε the (trivial) coloring for r. For $A_F \subseteq B$, B-restricted admissible sets of AF $F = (A_F, R_F)$ are just admissible sets for F. Hence $e'_r(\varepsilon) = \text{admissible}(F)$ follows immediately from Definitions 3.1 and 3.3. □

Because of this correspondence, our goal is to efficiently compute $e'_r(\varepsilon)$ for the root r of a TD T of an AF F using coloring ε for r, as we have seen that $e'_r(\varepsilon) = \text{admissible}(F)$. However, to guarantee FPT w.r.t. treewidth, we cannot afford to compute $e_t(\cdot)$ explicitly. In the following, we show how we can compute compact representations of $e_t(\cdot)$. For this, we first define the following operations that we will use in our algorithm.

Definition 3.6. *Let $(\mathscr{T}, \mathscr{X})$ be a TD of an AF $F = (A_F, R_F)$, and let C' and C'' be colorings for nodes t' and t'', respectively, in $\mathscr{T}$. We define the following operations.*

$$(C' - a)(b) = \quad C'(b) \quad \text{for each } b \in X_{t'} \setminus \{a\}$$

$$(C' +_{attc} a)(b) = \begin{cases} C'(b) & \text{if } b \in X_{t'} \\ def & \text{if } a = b \text{ and } [C'] \rightarrowtail^{R_F} a \\ attc & \text{otherwise} \end{cases}$$

$$(C' \dot{+}_{out} a)(b) = \begin{cases} C'(b) & \text{if } b \in X_{t'} \\ out & \text{otherwise} \end{cases}$$

$$(C' \dot{+}_{in} a)(b) = \begin{cases} in & \text{if } a = b \text{ or } C'(b) = in \\ out & \text{if } a \neq b, (a,b) \notin R_F, (b,a) \notin R_F, C'(b) = out \\ def & \text{if } a \neq b \text{ and } ((C'(b) = attc \text{ and } (a,b) \in R_F) \text{ or } C'(b) = def) \\ attc & \text{otherwise} \end{cases}$$

$$(C' \bowtie C'')(b) = \begin{cases} in & \text{if } C'(b) = C''(b) = in \\ out & \text{if } C'(b) = C''(b) = out \\ def & \text{if } C'(b) = def \text{ or } C''(b) = def \\ attc & \text{otherwise} \end{cases}$$

In our algorithm, we will use the "$-$" operation at FORGET nodes, the three "$+$" operations at INSERT nodes and "$\bowtie$" at JOIN nodes. The intuitions behind the three different "$+$" operations are the following: The operation $C +_{\text{attc}} a$ causes that a newly introduced atom a is considered an attacking candidate (attc); hence a will have to be defeated (def) by a resulting extension S (i.e., $S \rightarrowtail^{R_F} a$). The operation $C \dot{+}_{\text{in}} a$ puts the new atom a in the resulting extension. Finally, $C \dot{+}_{\text{out}} a$ results in a being neither in the extension nor being an attacking candidate of it. Using these operations, we now define our algorithm that traverses a TD in a bottom-up manner to compute *vcolorings*, which serve as compact representations of valid colorings.

Definition 3.7. *Let $(\mathscr{T}, \mathscr{X})$ be a TD of an AF F, and let $t \in \mathscr{T}$. If t has exactly one child, let t' denote this child; if t has two children, let them be denoted by t' and t''. We define the set of* vcolorings *for t depending on the node type of t:*

LEAF: *A coloring $C' : X_t \to \{in, out, def, attc\}$ is a vcoloring for t if the following conditions hold for all $x \in X_t$: (i) $C'(x) = in \implies C'(y) \in \{def, attc\}$ for all $y \rightarrowtail x$; (ii) $C'(x) = def \iff \exists y : C'(y) = in$ and $y \rightarrowtail x$*

FORGET ($X_t = X_{t'} \setminus \{a\}$ for some argument a): *If C' is a vcoloring for t' and $C'(a) \neq attc$, then $C' - a$ is a vcoloring for t.*

INSERT ($X_t = X_{t'} \cup \{a\}$ for some argument a):

(i) *If C' is a vcoloring for t', then $C' +_{attc} a$ is a vcoloring for t.*

(ii) *If C' is a vcoloring for t', $[C'] \not\rightarrowtail a$ and $a \not\rightarrowtail [C']$, then $C' \dot{+}_{out} a$ is a vcoloring for t.*

(iii) *If C' is a vcoloring for t', $a \not\rightarrow a, [C'] \not\rightarrow a$, $a \not\rightarrow [C']$ and $[[C']] = [[C' \dot{+}_{in} a]]$, then $C' \dot{+}_{in} a$ is a vcoloring for t.*

JOIN: *If C' and C'' are vcolorings for t' and t'', respectively, and $[C'] = [C'']$ as well as $[[C']] = [[C'']]$ hold, then $C' \bowtie C''$ is a vcoloring for t.*

Example 3.8. *Let F and $T = (\mathscr{T}, \mathscr{X})$ be the AF and TD, respectively, from Figure 1. Figure 2 illustrates the bottom-up computation of the vcolorings for the AF and TD of Figure 1. Each row r contains a vcoloring in the set $\mathrm{D}(r)$, and $\mathrm{P}(r)$ contains the EPTs as described in Section 2. By following the EPTs from the row 8:I in the root table, we obtain the sets $\emptyset$, $\{w\}$ and $\{y\}$, which are indeed exactly the admissible sets.*

We now illustrate how to compute the tables in Figure 2 from the bottom up. Consider LEAF node t_1 with bag $\{w,y\}$ and $F_{\geq t_1} = (\{w,y\}, \{(w,y),(y,w)\})$. Its table contains six vcolorings, which correspond to $\emptyset$-restricted admissible tuples (that encode conflict-free sets) for $F_{\geq t_1}$, namely $(\emptyset,\emptyset)$, $(\emptyset,\{w\})$, $(\emptyset,\{y\})$, $(\emptyset,\{w,y\})$, $(\{w\},\{y\})$ and $(\{y\},\{w\})$.

The next node t_2 is of type FORGET and removes y ($X_{>t_2} = \{y\}$). The vcolorings for t_2 are obtained from vcolorings for t_1 except for C' with $C'(y) = attc$. Intuitively, such colorings are not extended further because y is still an undefeated attacking candidate (i.e., y requires defeating). By properties (2) and (3) of TDs, y is not attacked by any argument outside $X_{\geq t_2}$, ie., y will not be defeated in nodes further toward the root. The colorings for t_2 correspond to the $X_{>t_2}$-restricted admissible tuples for $F_{\geq t_2} = F_{\geq t_1}$.

Node t_3 introduces argument x. Consider the coloring C' of t_2 with $C'(w) = attc$. We have now three possibilities for argument x (corresponding to our "$+$" operations).

- $C = C' +_{attc} x$: *This results in $C(x) = attc$ and $C(w) = attc$.*
- $C = C' \dot{+}_{in} x$: *If we set $C(x) = in$, this leads to $C(w) = def$ since $x \rightarrowtail^{R_F} w$.*
- $C = C' \hat{+}_{out} x$: *This leads to $C(x) = out$ and $C(w) = attc$.*

The only JOIN node is t_6, which combines subframeworks $F_{\geq t_3}$ and $F_{\geq t_5}$. Let C' and C'' be colorings for t_3 and t_5, respectively, and let $C'(w) = in = C''(w)$ and $C'(x) = def = C''(x)$. Since $[C'] = [C'']$ and $[[C']] = [[C'']]$, the colorings coincide on $X_{\geq t_3} \cap X_{\geq t_5}$ and we can join these colorings without any conflict, leading to $C = C' \bowtie C''$ with $C(x) = C'(x) = C''(x)$ and $C(w) = C'(w) = C''(w)$ for node t_6. Now consider coloring C^* for node t_3 with $C^*(w) = def$ and $C^*(x) = in$. It holds that $[C''] \neq [C^*]$, and $[C''] \cup [C^*] = \{w,x\}$ are in conflict, leading to the fact that C'' and C^* do not result in a vcoloring for node t_6. In fact, there is no resulting vcoloring for node t_7 originating from C^*.

Together with Proposition 3.5, the following theorem proves that the algorithm described in Definition 3.7 is sound.

Theorem 3.9. *Let $(\mathscr{T}, \mathscr{X})$ be a TD of an AF F. For each coloring C for a node $t \in \mathscr{T}$, C is a valid coloring for t if and only if C is a vcoloring for t.*

Proof sketch. The proof is by structural induction over the given TD and shows equivalence between valid colorings and vcolorings for all node types, see [14]. □

Recall that the A_F-restricted admissible sets for an AF $F = (A_F, R_F)$ are the admissible sets for F. Because of Theorem 3.9 and Proposition 3.5, we can construct a valid coloring ε for the root r of any TD T by computing vcolorings in a bottom-up manner. This allows us to enumerate admissible sets via $e'_r(\varepsilon)$. Observe that $\emptyset$ is always an admissible extension, so ε trivially exists, but for enumerating $e'_r(\varepsilon)$, the vcolorings for

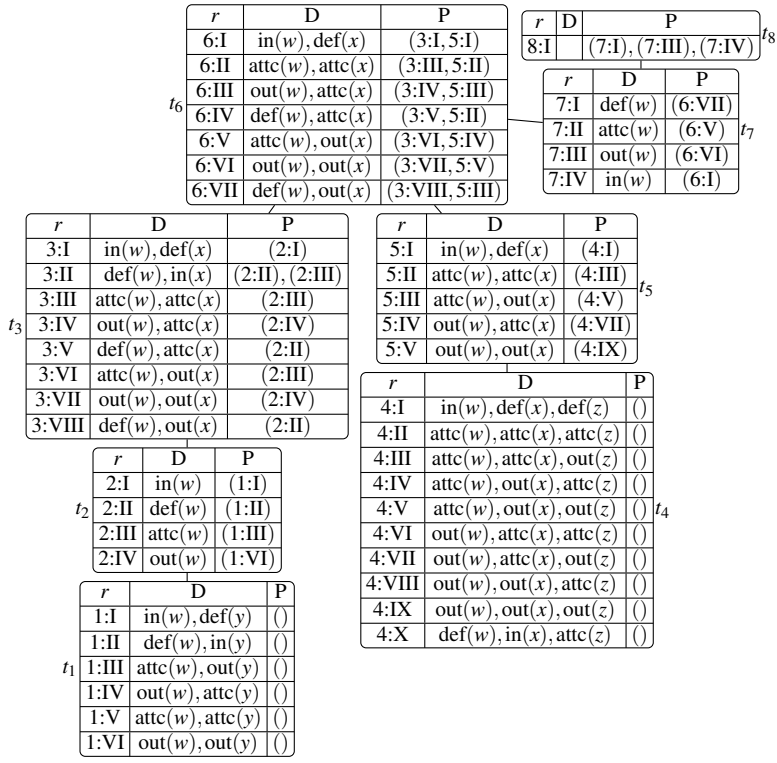

Figure 2: DP computation of vcolorings for $F = (A_F, R_F)$ w.r.t. T (see Figure 1).

all the nodes of T are required. Enumeration can be done with linear delay by combining vcolorings from the different nodes of T according to the EPTs [1].

4. Algorithm for Semi-Stable Semantics

We now present our algorithm for semi-stable semantics by re-using concepts from the algorithm for admissible semantics from Section 3. First we define the counterparts of valid colorings and vcolorings for semi-stable semantics, namely *valid pairs* and *vpairs*.

Definition 4.1. *Let $(\mathscr{T}, \mathscr{X})$ be a TD of an AF F, $t \in \mathscr{T}$, and (C, Γ) a pair with C being a coloring for t and Γ being a set of colorings for t. We call (C, Γ) simply a* pair *for t and define $e_t(C, \Gamma)$ as the collection of tuples (S, D) that satisfy the following conditions.*

(i) $(S, D) \in e_t(C)$;
(ii) For all $C' \in \Gamma$, *there is an* $(E, U) \in e_t(C')$ *such that* $S \cup D \subset E \cup U$;
(iii) For all tuples (E, U) *that are* $X_{>t}$*-restricted admissible for* $F_{\geq t}$ *s.t.* $S \cup D \subset E \cup U$, *there is a* $C' \in \Gamma$ *with* $(E, U) \in e_t(C')$.

If $e_t(C, \Gamma) \neq \emptyset$, we call (C, Γ) a valid pair *for t. We define $e'_t(C, \Gamma) = \{S \mid (S, D) \in e_t(C, \Gamma)\}$.*

Given a pair (C, Γ) for a TD node t, the coloring C again represents admissible sets. Recall that an admissible set S is a semi-stable extension if there is no admissible set S'

whose range is a proper superset of the range of S. The colorings in Γ represent exactly such sets S'. Thus, in our algorithm for semi-stable semantics, we will again compute all colorings that represent admissible sets, but for each such coloring C we store colorings in Γ that cause all solution candidates represented by C to be rejected.

Definition 4.2. *Let $(\mathcal{T},\mathcal{X})$ be a TD of an AF $F=(A_F,R_F)$, and let Γ and Δ be sets of colorings for nodes t' and t'', respectively, in $\mathcal{T}$. We define (similar to Definition 3.6):*

$$\begin{aligned}
\Gamma - a &= \{C' - a \mid C' \in \Gamma, C'(a) \neq attc\}\\
\Gamma +_{attc} a &= \{C' +_{attc} a \mid C' \in \Gamma, [C'] \not\rightarrow^{R_F} a, a \not\rightarrow^{R_F} [C']\}\\
\Gamma \dot{+}_{in} a &= \{C' \dot{+}_{in} a \mid C' \in \Gamma, [C'] \not\rightarrow^{R_F} a, a \not\rightarrow^{R_F} [C'], a \not\rightarrow^{R_F} a \text{ and } [[C']] = [[C' \dot{+}_{in} a]]\}\\
\Gamma \hat{+}_{out} a &= \{C' \hat{+}_{out} a \mid C' \in \Gamma\}\\
\Gamma \bowtie \Delta &= \{C' \bowtie C'' \mid C' \in \Gamma, C'' \in \Delta, [C'] = [C''] \text{ and } [[C']] = [[C'']]\}
\end{aligned}$$

Definition 4.3. *Let $(\mathcal{T},\mathcal{X})$ be a TD of an AF F and let $t \in \mathcal{T}$ be a node with t', t'' its possible children. Depending on the node type of t, we define a* vpair *for t.*

LEAF: *Each (C,Γ) with $C \in \mathscr{C}_t$ and $\Gamma = \{C' \in \mathscr{C}_t \mid [C] \cup [[C]] \subset [C'] \cup [[C']]\}$ is a vpair for t.*

FORGET ($X_t = X_{t'} \setminus \{a\}$ for some argument a): *If (C',Γ') is a vpair for t' and $C'(a) \neq attc$, then $(C'-a,\Gamma'-a)$ is a vpair for t.*

INSERT ($X_t = X_{t'} \cup \{a\}$ for some argument a): *If (C',Γ') is a vpair for t' and $C' \dot{+}_{in} a$ is a vcoloring for t, then $(C' \dot{+}_{in} a, (\Gamma' +_{attc} a) \cup (\Gamma' \dot{+}_{in} a))$ is a vpair for t; if moreover $C' \hat{+}_{out} a$ is a vcoloring for t, then $(C' \hat{+}_{out} a, (\{C'\} +_{attc} a) \cup (\{C'\} \dot{+}_{in} a) \cup (\Gamma' +_{attc} a) \cup (\Gamma' \dot{+}_{in} a) \cup (\Gamma' \hat{+}_{out} a))$ is a vpair for t; $(C' +_{attc} a, (\Gamma' +_{attc} a) \cup (\Gamma' \dot{+}_{in} a))$ is a vpair for t.*

JOIN: *If (C',Γ') is a vpair for t', (C'',Γ'') is a vpair for t'', $[C'] = [C'']$ and $[[C']] = [[C'']]$, then $(C' \bowtie C'', (\Gamma' \bowtie \Gamma'') \cup (\{C'\} \bowtie \Gamma'') \cup (\Gamma' \bowtie \{C''\}))$ is a vpair for t.*

Example 4.4. *Figure 3 illustrates the computation of the vpairs for the AF F and the TD from Figure 1. For each row r, we store a set Γ of references to rows r' from the same table such that r' represents admissible sets whose range is a proper superset of the range of each admissible set represented by r. By following the EPTs, we obtain exactly one set, namely $\{w\}$, which is in fact the only semi-stable extension.*

For our correctness proof, we need another lemma and a proposition.

Lemma 4.5. *Let $(\mathcal{T},\mathcal{X})$ be a TD of an AF F and $t \in \mathcal{T}$. For each $X_{>t}$-restricted admissible tuple (S,D) for $F_{\geq t}$, there is a coloring $C \in \mathscr{C}_t$ s.t. $(S,D) \in e_t(C)$.*

Proposition 4.6. *Let r be the root of a TD $(\mathcal{T},\mathcal{X})$ of an AF $F=(A_F,R_F)$. It holds that $e'_r(\varepsilon,\emptyset) = \text{semi-stable}(F)$.*

Proof sketch. Recall that $e'_r(\varepsilon) = \text{admissible}(F)$ (see Proposition 3.5, Definitions 3.1 and 3.3). To show $e'_r(\varepsilon,\emptyset) \subseteq \text{semi-stable}(F)$, let (S,D) be an arbitrary tuple s.t. $(S,D) \in e_r(\varepsilon,\emptyset)$. By condition (i) from Definition 4.1, S is admissible for $F_{\geq r} = F$. Furthermore, by (iii) and the fact that $\Gamma = \emptyset$ we conclude that there is no admissible tuple (E,U) for F with $E \cup U$ being a proper superset of $S \cup D$, i.e., S is a semi-stable extension of F. It remains to show that $e'_r(\varepsilon,\emptyset) \supseteq \text{semi-stable}(F)$. Let $S \in \text{semi-stable}(F)$ be an arbitrary semi-stable

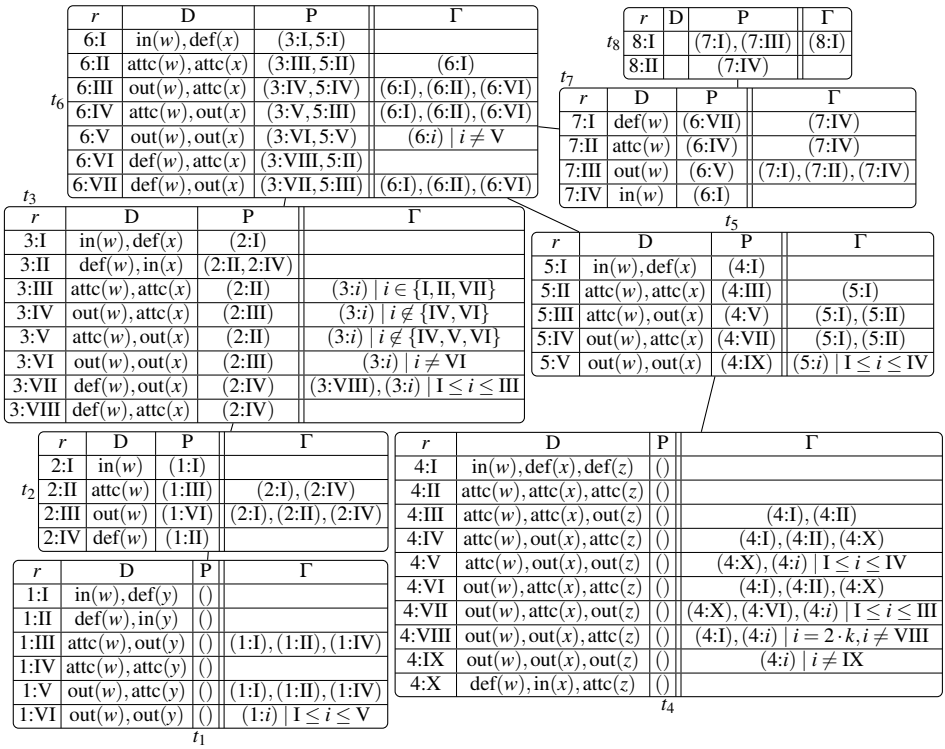

Figure 3: DP computation of vpairs for $F = (A_F, R_F)$ w.r.t. T (see Figure 1).

extension of F with range $S^+_{R_F}$. We set $D = S^+_{R_F} \setminus S$ to get the arguments that require defeating. It can be shown [14] that there exists a pair (C, Γ) such that $(S, D) \in e_r(C, \Gamma)$. Since the root node has an empty bag, $C = \varepsilon$, and furthermore, by condition (ii) from Definition 4.1 and the fact that $S \cup D$ is maximal (w.r.t. $\subseteq$) in F, $\Gamma = \emptyset$ holds as well. □

Finally, we state the main theorem of this section, which analgously to Theorem 3.9 can be proved by structural induction [14].

Theorem 4.7. *Let $(\mathscr{T}, \mathscr{X})$ be a TD of an AF F. For each pair (C, Γ) for a node t, it holds that (C, Γ) is a valid pair for t if and only if (C, Γ) is a vpair for t.*

Together with Proposition 4.6, this guarantees that we can compute semi-stable extensions via vpairs. For the root r of a TD T of a framework F, we can compute vpairs in a bottom-up manner along T and thus obtain valid pairs. For enumerating $e'_r(\varepsilon, \emptyset)$ (i.e., semi-stable extensions of $F_{\geq r} = F$), we combine vpairs from all nodes of T.

Proposition 4.8. *Let T be a TD of width w for an AF F of size n. The DP computation for semi-stable extensions according to Definition 4.3 is feasible in FPT time, i.e., in time $f(w) \cdot n^{\mathscr{O}(1)}$, for some function f that depends only on w.*

Proof sketch. For the induction base, let t be a LEAF node. There are up to $\mathscr{O}(4^w)$ many vcolorings and vpairs for t, which can be computed in time $g(w) \cdot n^{\mathscr{O}(1)}$, for some function g. For the induction step, let t be a FORGET, INSERT or JOIN node and k be the number

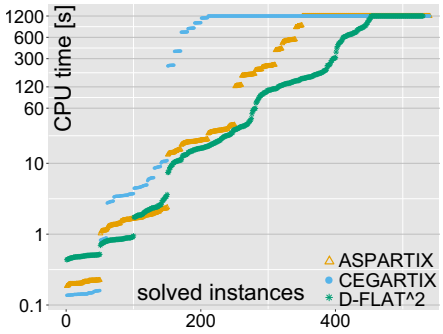

Figure 4: Plot on grids, TW ≤ 5, P 0.9.

	ASPARTIX		CEGARTIX		D-FLAT^2	
	t[s]	t/outs	t[s]	t/outs	t[s]	t/outs
TW 4, P 0.9	443.9	(100)	712	(160)	**232.9**	**(49)**
TW 4, P 0.7	280	(40)	555.4	(90)	**34.4**	**(0)**
TW 4, P 0.5	17.3	**(0)**	127	(20)	**2.1**	**(0)**
TW 5, P 0.9	551.4	(90)	938.5	(170)	**358.2**	**(29)**
TW 5, P 0.7	365.9	(60)	611.6	(110)	**174.2**	**(4)**
TW 5, P 0.5	**41.6**	**(0)**	278.7	(50)	88.2	**(0)**
TW 6, P 0.9	**705.1**	**(100)**	1200	(200)	1168.5	(190)
TW 6, P 0.7	**394.1**	**(57)**	640.2	(70)	1137.3	(176)
TW 6, P 0.5	**31.1**	**(0)**	200.9	(30)	1076.7	(157)

Table 1.: Tabular results on grids.

of children of t, and assume that the time required for computing the tables of the children is $g_i(w) \cdot n^{\mathcal{O}(1)}$, for $1 \leq i \leq k$ and some function g_i. There are at most $\mathcal{O}(4^w)$ many vcolorings for t, and there can be at most $\mathcal{O}(4^w \cdot 2^{4^w})$ vpairs, which can be computed in time $g(w) \cdot n^{\mathcal{O}(1)} \cdot \Pi_{i=1}^{k} g_i(w)$, for some function g, as described in in Definition 4.3. Since we may assume that T has size $\mathcal{O}(n)$, the claim holds. □

Our algorithm for semi-stable semantics can be easily turned into an algorithm for preferred semantics (as an alternative to [11]) by simplifying Definitions 4.1 and 4.3.

5. Preliminary Evaluation

We implemented the algorithm of Section 4 as an ASP encoding for the D-FLAT^2 system[1]. This is an extended version of the D-FLAT system [1] and capable of efficiently solving problems from the second level of the polynomial hierarchy if the treewidth is small. We compared D-FLAT^2 1.0.3 with CEGARTIX 0.4 [10] and ASPARTIX [13]. D-FLAT^2 internally uses ASP systems Gringo 4.5.4 and Clasp 3.1.4; we also used these versions for ASPARTIX. DYNPARTIX [3] cannot compute semi-stable extensions yet.

DP on TDs makes sense on instances with small treewidth, but usually yields poor performance if the treewidth is very large. For instances of the International Competition on Computational Models of Argumentation (ICCMA), we observed widths between 60 and 200, which is too much for our system. Hence we used randomly generated instances obtained from grids: Vertices are arranged on an $n \times m$ matrix, and edges connect horizontally, vertically and diagonally neighboring vertices with a certain probability (P).

We considered the problem of enumerating semi-stable extensions and compared the systems on instances with ≤ 70 nodes and treewidth (TW) ≥ 4; the observed widths of the TDs are ≤ 11. Each D-FLAT^2 instance was run ten times with different TDs, and every run was limited to twenty minutes and three GB of memory. Figure 4 shows a cactus plot, and Table 1 lists running times in seconds and the number of timeouts. D-FLAT^2 exhibited the best performance, while CEGARTIX and ASPARTIX often time out, especially on larger instances. On the other hand, the performance of D-FLAT^2 becomes worse with increasing treewidth, thus reflecting our runtime estimation from Proposition 4.8. For detailed results, we refer to [14].

[1]The system [2] is open source and available at https://github.com/hmarkus/dflat-2

6. Conclusion

We presented a new algorithm for computing semi-stable semantics using dynamic programming on tree decompositions that runs in linear time on AFs of bounded treewidth. For this purpose, we extended the concept of restricted-admissible sets [11]. Our experimental results indicate performance advantages over existing systems in case of bounded treewidth. It should be noted that such DP algorithms should not be seen as general solvers that outperform standard techniques on average. Instead, DP algorithms qualify as an alternative approach when instances are structurally rather "close" to trees.

Acknowledgements This research has been supported by the Austrian Science Fund (FWF) through projects Y698, P25607, I1102 and I2854.

References

[1] M. Abseher, B. Bliem, G. Charwat, F. Dusberger, M. Hecher, and S. Woltran. D-FLAT: Progress report. Technical Report DBAI-TR-2014-86, TU Wien, 2014.
[2] B. Bliem, G. Charwat, M. Hecher, and S. Woltran. D-FLAT^2: Subset minimization in dynamic programming on tree decompositions made easy. In *ASPOCP*, 2015.
[3] G. Charwat and W. Dvořák. dynPARTIX 2.0 - Dynamic programming argumentation reasoning tool. In *COMMA*, volume 245 of *FAIA*, pages 507–508. IOS Press, 2012.
[4] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, 1999.
[5] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–357, 1995.
[6] P. E. Dunne. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.*, 171(10-15):701–729, 2007.
[7] P. E. Dunne and T. J. M. Bench-Capon. Coherence in finite argument systems. *Artif. Intell.*, 141(1/2):187–203, 2002.
[8] W. Dvorák, S. Ordyniak, and S. Szeider. Augmenting tractable fragments of abstract argumentation. *Artif. Intell.*, 186:157–173, 2012.
[9] W. Dvořák, S. Szeider, and S. Woltran. Abstract argumentation via monadic second order logic. In *SUM*, volume 7520 of *LNCS*, pages 85–98. Springer, 2012.
[10] W. Dvořák, M. Järvisalo, J. P. Wallner, and S. Woltran. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.*, 206:53 – 78, 2014.
[11] W. Dvořák, R. Pichler, and S. Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186:1–37, 2012.
[12] W. Dvořák and S. Woltran. Complexity of semi-stable and stage semantics in argumentation frameworks. *Inf. Process. Lett.*, 110:425–430, 2010.
[13] U. Egly, S. A. Gaggl, and S. Woltran. Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, 1(2):147–177, 2010.
[14] M. Hecher. Optimizing Second-Level Dynamic Programming Algorithms; The D-FLAT^2 System: Encodings and Experimental Evaluation. Master's thesis, Vienna University of Technology, 2015.
[15] E. J. Kim, S. Ordyniak, and S. Szeider. Algorithms and complexity results for persuasive argumentation. *Artif. Intell.*, 175(9-10):1722–1736, 2011.
[16] T. Kloks. *Treewidth: Computations and Approximations*, volume 842 of *LNCS*. Springer, 1994.
[17] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. OUP, 2006.
[18] N. Robertson and P. D. Seymour. Graph minors. III. Planar tree-width. *J. Comb. Theory, Ser. B*, 36(1):49–64, 1984.