Tractable Abstract Argumentation via Backdoor-Treewidth

**Wolfgang Dvořák, Markus Hecher, Matthias König,
André Schidler, Stefan Szeider, Stefan Woltran**

TU Wien, Institute of Logic and Computation
{dvorak, hecher, mkoenig, woltran}@dbai.tuwien.ac.at, {aschidler, sz}@ac.tuwien.ac.at

Abstract

Argumentation frameworks (AFs) are a core formalism in the field of formal argumentation. As most standard computational tasks regarding AFs are hard for the first or second level of the Polynomial Hierarchy, a variety of algorithmic approaches to achieve manageable runtimes have been considered in the past. Among them, the backdoor-approach and the treewidth-approach turned out to yield fixed-parameter tractable fragments. However, many applications yield high parameter values for these methods, often rendering them infeasible in practice. We introduce the backdoor-treewidth approach for abstract argumentation, combining the best of both worlds with a guaranteed parameter value that does not exceed the minimum of the backdoor- and treewidth-parameter. In particular, we formally define backdoor-treewidth and establish fixed-parameter tractability for standard reasoning tasks of abstract argumentation. Moreover, we provide systems to find and exploit backdoors of small width, and conduct systematic experiments evaluating the new parameter.

1 Introduction

Argumentation is used to resolve conflicts in potentially inconsistent or incomplete knowledge. Argumentation frameworks (AFs), introduced in his influential paper by Dung (1995), turned out to be a versatile system for reasoning tasks in an intuitive setting. In AFs we view arguments just as abstract entities, represented by nodes in a directed graph, independent from their internal structure. Conflicts are modelled in form of attacks between these arguments, constituting the edges of said graph representation. The semantics are defined via so called *extensions*—sets of arguments that can be jointly accepted.

Evaluating AFs w.r.t. these semantics is a central task in argumentation (Cerutti et al. 2018). However, many common reasoning and counting problems regarding extensions turned out to be computationally hard by classical notions (Dvořák and Dunne 2017; Fichte, Hecher, and Meier 2019; Baroni, Dunne, and Giacomin 2010). For this reason, various approaches utilizing structural parameters have been introduced (Dunne 2007) and shown to be fruitful for AFs, in particular treewidth (Dvořák, Pichler, and Woltran 2012) and backdoor-distance (Dvořák, Ordyniak, and Szeider 2012). These concepts support efficient (i.e., polynomial time) reasoning, provided the respective parameter is small. In this paper, we combine these two approaches to achieve a smaller parameter while still maintaining polynomial runtime in the size of the framework.

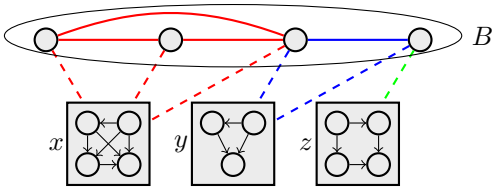


Figure 1: Illustration of the backdoor-treewidth approach

To illustrate the usefulness of this concept consider the following example scenario, where two parties argue about a general agreement, where diverging opinions on several subtopics have to be resolved. The discussions about the subtopics might be located in easy fragments of AFs (e.g., symmetric or acyclic frameworks). Then there is a meta-discussion that goes along the lines "if we agree on argument a in subtopic x, we shall have in turn argument b in subtopic y accepted", etc. One can expect that the meta-discussion, while not being in an easy fragment itself, exhibits certain structural features. A high-level illustration of this concept is given in Figure 1, where we have subtopics x, y, z and the meta-discussion B. Here, B indicates the backdoor functioning of this part, i.e. removing B yields an AF composed of easy fragments. In order to employ the backdoor-treewidth approach, we have to use the *torso graph* of B, where any pair of nodes $a, b \in B$ adjacent to same component (x, y, or z) needs to be connected; it is this graph for which we require small treewidth.

Our main contributions can be summarised as follows.

- We define the concept of *backdoor-treewidth* (Ganian, Ramanujan, and Szeider 2017b) for AFs. Specifically, we present two backdoor-types that allow us to exploit small treewidth, and two backdoor-types that do not.
- Moreover, we argue that given such a backdoor of small treewidth, the reasoning tasks on AFs are *fixed-parameter tractable* (FPT) parameterized by the backdoor's treewidth in complete and stable semantics, ex-

tending the known FPT results for the classical notion of treewidth. Fixed-parameter tractability for a problem of input size n and parameter k refers to solvability in time $f(k)n^c$ where f is a (possibly exponential) function of k and c is a constant (Gottlob and Szeider 2008).
- We utilize SAT-solvers both to compute a suitable backdoor set of small treewidth and to solve the instance. The latter is done via a tree decomposition-guided reduction (Fichte et al. 2021) from the argumentation-specific problems to propositional satisfiability. The resulting propositional formula is then evaluated with an adaptation of the NestHdb system for dynamic programming (Hecher, Thier, and Woltran 2020).
- Our experimental results indicate that backdoor-treewidth for argumentation is promising compared to existing techniques based on the measures treewidth and backdoor-size. In particular, it turns out that in practice, backdoor-treewidth can solve many additional instances, where both treewidth and backdoor-size is insufficient.

Some proofs are only given in full length in the appendix.[1]

2 Background

Argumentation. An argumentation framework (AF) due to Dung (1995) is a pair $F = (A, R)$ where A is a non-empty and finite set of arguments, and $R \subseteq A \times A$ is the attack relation. We write $S \mapsto_R b$ if there is some $a \in S$ such that $(a, b) \in R$. Likewise, $S \mapsto_R S'$ denotes $S \mapsto_R b$ for some $b \in S'$. Given an AF $F = (A, R)$, a set $S \subseteq A$ is *conflicting* in F if $S \mapsto_R a$ for some $a \in S$. A set $S \subseteq A$ is *conflict-free* in F, if S is not conflicting in F, i.e. if $\{a, b\} \not\subseteq S$ for each $(a, b) \in R$. An argument $a \in A$ is *defended* (in F) by $S \subseteq A$ if for each $B \subseteq A$, $B \mapsto_R a$ implies $S \mapsto_R B$. A set T of arguments is defended (in F) by S if each $a \in T$ is defended by S (in F). Let $S \subseteq A$ be a conflict-free set in F. Then, S is *admissible* in F, denoted by $S \in adm(F)$, if S defends itself in F; S is *stable* in F ($S \in stb(F)$), if S attacks every argument in $A \setminus S$; S is *complete* for F ($S \in com(F)$), if $S \in adm(F)$ and contains every argument it defends; S is *preferred* in F ($S \in pref(F)$), if $S \in adm(F)$ and there is no $T \in adm(F)$ with $T \supset S$. Standard reasoning tasks are *credulous/skeptical acceptance* (is an argument in one/all extensions?).

Backdoors to Tractability. Reasoning on AFs turned out to be NP/coNP-hard for most cases (we assume the reader to be familiar with complexity classes P, NP, coNP, and Π_2^P; see the work of Dvořák and Dunne (2017) for an introduction to complexity theory in the context of argumentation). Hence, work has been done to identify means of achieving computational speedups (Coste-Marquis, Devred, and Marquis 2005; Dunne 2007; Dunne and Bench-Capon 2001), which lead to the discovery of *tractable fragments of argumentation*. We consider here acyclicity, even-cycle-freeness, bipartiteness, and symmetry, denoted by *ACYC*, *NOEVEN*, *BIP*, and *SYM*, respectively, and defined in a standard way for directed graphs (Dvořák and Dunne 2017). As these fragments restrict the possible argumentation structures, to exploit their computational advantages while retaining expressiveness we consider arbitrary (fixed) distances.

We use the *backdoor*-approach by Dvořák, Ordyniak, and Szeider (2012): let $F = (A, R)$ be an AF and let $\mathcal{C}$ be a tractable fragment. We call a set $S \subseteq A$ a $\mathcal{C}$-backdoor if $(A', R \cap (A' \times A'))$ for $A' = A \setminus S$ belongs to $\mathcal{C}$. We denote the size of a smallest $\mathcal{C}$-backdoor by $\mathtt{bd}_{\mathcal{C}}(F)$. If it is clear what fragment $\mathcal{C}$ is meant, we shall drop the subscript. Reasoning w.r.t. stable, complete, and preferred semantics is tractable for the fragments *ACYC* and *NOEVEN* if $\mathtt{bd}_{\mathcal{C}}(F)$ is fixed (Dvořák, Ordyniak, and Szeider 2012).

Treewidth. We recall the notion of *treewidth* (Robertson and Seymour 1986). Let $F = (A, R)$ be an AF. Let $U^F = (V, E)$ be the corresponding undirected graph, i.e., $V = A$ and there is an edge between two vertices $a, b \in V$ iff $a \neq b$ and there is an attack (a, b) in R. A *tree decomposition (TD)* of F is a pair $(\mathcal{T}, \mathcal{X})$, where $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is a tree and $\mathcal{X} = (X_t)_{t \in V_\mathcal{T}}$ is a set of bags (a bag is a subset of A) such that (1) $\bigcup_{t \in V_\mathcal{T}} X_t = V$; (2) for each $v \in V$, the subgraph induced by v in $\mathcal{T}$ is connected; and (3) for each $\{v, w\} \in E$, $\{v, w\} \subseteq X_t$ for some $t \in V_\mathcal{T}$. The width of a TD is $\max\{|X_t| \mid t \in V_\mathcal{T}\} - 1$, the *treewidth* of F, denoted by $\mathtt{tw}(F)$, is the minimum width of all TDs for F. Reasoning on an AF F is fixed-parameter tractable parameterized by its treewidth $\mathtt{tw}(F)$ (Dvořák, Pichler, and Woltran 2012). For utilizing these results, weaker notions of extensions exist:

Definition 1 (Restricted Extensions)**.** *Let $F = (A, R)$ be an AF and $C, S \subseteq A$ be sets of arguments. We call S a C-*restricted stable set*, if S is conflict-free in F and S attacks all arguments $a \in C$. Then, S is a C-*restricted complete set *if S is conflict-free in F, defends all arguments in $C \cap S$ and contains every argument in C that is defended by S.*

3 Towards Backdoor-Treewidth

In this section, we formally define backdoor-treewidth for abstract argumentation. Intuitively, this notion allows us to "hide" subframeworks (that belong to tractable fragments) and decompose the graph, called *torso graph*, containing "remaining" arguments (i.e., *backdoor arguments*). Ultimately, we utilize a tree decomposition of this torso graph to perform dynamic programming (DP). To ensure correct interaction between these backdoor arguments and the tractable fragments, the whole neighborhood of such a subframework is completely connected in the torso graph. This forces that neighboring arguments of subframeworks appear in a common bag of the TD and enables solving.

Torso Graphs and their Width. We start with the notion of S-components, i.e., components that stay connected (ignoring directions of attacks) after deleting a set S of arguments. Let $F = (A, R)$ be an AF, $S \subseteq A$ be a set of arguments and let $U^F = (V, E)$ be the *corresponding undirected graph*. An *S-component* is a connected component (maximal connected subgraph) of the graph $U^{F'}$, which we obtain from U^F by deleting the vertices S and their incident edges. The torso graph is an aggregated representation of F.

Definition 2 (Torso Graph)**.** *Let $F = (A, R)$ be an AF and $S \subseteq A$ a set of arguments. The S-*torso graph *G_F^S is defined*

[1] Find the appendix at github.com/mk-tu/argBTW.

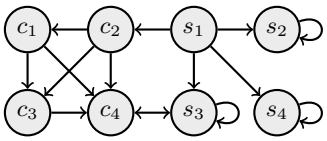
as the (undirected) graph with S as vertices, where two vertices s, t are adjacent iff there is an S-component that s and t are adjacent to in U^F, or an attack (s,t) or (t,s) in R.

This definition allows us to define backdoor-treewidth in terms of TDs over all possible torso graphs.

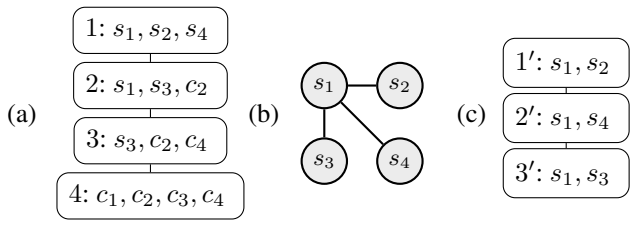
Definition 3 (Backdoor-Treewidth)**.** *The $\mathcal{C}$-backdoor-treewidth $\mathtt{btw}_{\mathcal{C}}(F)$ of an AF F is the minimal width of all tree decompositions of the torso graphs G^S_F for all $\mathcal{C}$-backdoors S of F.*

For the ease of presentation, in this work we assume TDs, where each subset-maximal bag appears (also) in the bag of a leaf of the TD. Indeed, this is not a hard restriction and any TD of the torso graph can be transformed such that this property holds, without increasing its width. The arguments of tractable subframeworks not contained in the torso graph are then considered in leaf node bags containing relevant backdoor arguments (i.e., those adjacent to the S-component). To this end, let $F = (A, R)$ be an AF, $S \subseteq A$ be a backdoor and $(\mathcal{T}, \mathcal{X})$ be a tree decomposition of G^F_S. Then, for a leaf node $t \in \mathcal{T}$, we denote by *components*(t) the *union of all arguments in S-components*, where all adjacent vertices that are in S are also in X_t. As the neighborhood of every S-component s is a clique, there is at least one such leaf node t such that X_t contains the arguments in s.

Example 1. Consider the following AF $F = (A, R)$, consisting of an acyclic 4-clique c_1, c_2, c_3, c_4 connected to a star s_1, s_2, s_3, s_4 with self-attacking leaves.



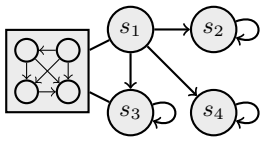
Because of the self-attacks, the minimum *ACYC*-backdoor has size at least 3. In fact, one can verify $B = \{s_2, s_3, s_4\}$ is such an *ACYC*-backdoor. As F contains a 4-clique, the treewidth of F is at least 3. Indeed, the following tree decomposition (a) of width 3 assures $\mathtt{tw}(F) = 3$.



Now consider $B' = \{s_1, s_2, s_3, s_4\}$. Obviously, B' is an *ACYC*-backdoor (but not minimal, as $B' \supset B$). The torso graph w.r.t. B' is exactly the star graph with s_1, s_2, s_3, s_4 as its vertices, i.e., it has backdoor-treewidth 1 (see (b), (c) above). We have *components*$(3') = \{c_1, c_2, c_3, c_4\}$.

Observe that we could add arbitrarily many new arguments to the clique and by doing so increasing the treewidth, while still remaining $\mathtt{btw}_{ACYC}(F) = 1$. Likewise, we can add self-attacking arguments as leaves to the star, increasing the minimum backdoor size of F, while, again, the backdoor-treewidth stays the same. Moreover, we want to highlight that the backdoor B' is not minimal - in fact, the cardinality-minimal backdoor B has a higher width.

We achieve this lower width by hiding easy acyclic subframeworks. In (b), the acyclic B'-component is adjacent only to s_1 and s_3, which assures a small width in the torso graph $G^{B'}_F$ (see illustration below).



Moreover, the backdoor-treewidth $\mathtt{btw}_{\mathcal{C}}(F)$ of an AF $F = (A, R)$ is bounded by $\min(\mathtt{bd}_{\mathcal{C}}(F), \mathtt{tw}(F))$: as A as a whole is a backdoor to any tractable fragment, U^F is the torso G^A_F. Moreover, the tree decomposition of G^B_F (for B being a minimum size backdoor) with only one bag containing all of B has width $|B| - 1$. From this it follows:

Proposition 1. *For an AF F and a tractable class $\mathcal{C}$,*
1. $\mathtt{bd}_{\mathcal{C}}(F)$ and $\mathtt{tw}(F)$ can be arbitrarily large even while $\mathtt{btw}_{\mathcal{C}}(F)$ remains constant,
2. $\mathtt{btw}_{\mathcal{C}}(F) < \mathtt{bd}_{\mathcal{C}}(F)$, and
3. $\mathtt{btw}_{\mathcal{C}}(F) \leq \mathtt{tw}(F)$.

In the following, we use *chordal* AFs: an AF $F = (A, R)$ is *chordal* if each set $C \subseteq A$ of arguments inducing a directed cycle of F contains a set of arguments C', $|C'| \leq 3$, that induces a directed cycle of F as well. Hence Proposition 1 also applies in the special case of chordal AFs.

4 Finding Backdoors of Minimum Width

Unfortunately, computing backdoor-treewidth exactly is a nontrivial task since it is insufficient to restrict to sets of size at most k. The non-parameterized version of this problem for *ACYC*-backdoors is NP-hard.

Proposition 2. *Deciding whether a given AF F has an ACYC-backdoor of width $\leq k$ is NP-complete (if k is part of the input).*

However, it is known that finding the minimum backdoor-treewidth for backdoors in other contexts such as SAT or CSP is fixed-parameter tractable parameterized by backdoor-treewidth (Ganian, Ramanujan, and Szeider 2017a,b). We utilize this result to obtain fixed-parameter tractability results for *SYM* in the general case and for *ACYC* for chordal AFs. The proofs for these theorems are given in the appendix and utilize respective results for CSP (Ganian, Ramanujan, and Szeider 2017b).

Theorem 1. *Given an AF F and a parameter k, it is fixed-parameter tractable to either return a SYM-backdoor of width $\leq k$ or correctly conclude that the SYM-backdoor-treewidth of F exceeds k.*

To highlight the relevance of Theorem 2, recall Proposition 1 also holds for chordal AFs. In particular, in this fragment, both treewidth and minimum *ACYC*-backdoor size can be arbitrarily larger than the *ACYC*-backdoor-treewidth.

Theorem 2. *Given a chordal AF F and a parameter k, it is fixed-parameter tractable to either return an ACYC-backdoor of width $\leq k$ or correctly conclude that the ACYC-backdoor-treewidth of F exceeds k.*

5 Utilizing Small Backdoor-Treewidth

In the following we present FPT results for AFs parameterized by backdoor-treewidth. This is established by means of dynamic programming (DP) algorithms for AFs that explicitly utilize backdoor-treewidth. In such DP algorithms, extensions are often times captured by *colorings* that are computed for each bag of a TD in post-order (bottom-up). These colorings give rise to invariants that need to be fulfilled for every node. Maintaining colorings is similar to existing DP algorithms (Dvořák, Pichler, and Woltran 2012) for treewidth, but for backdoor-treewidth one also needs to take care of tractable components induced by leaf nodes.

Next, we show that reasoning on AFs is fixed-parameter tractable parameterized for backdoor-treewidth to *ACYC* or *NOEVEN*, if a backdoor of minimum width is given.

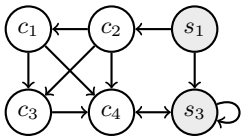
Theorem 3. *Reasoning and counting on AFs F in stable and complete semantics is fixed-parameter tractable when parameterized by $\mathtt{btw}_{\mathcal{C}}(F)$ for $\mathcal{C} \in \{ACYC, NOEVEN\}$ if a respective backdoor is given.*

This result is established by adapting the existing DP algorithm in consideration of our torso graphs. In particular, we consider the tractable S-components to be "attached" to the sub-problems in the leaves of a torso tree decomposition. Then, tractability is preserved by applying the backdoor algorithm approach in the leaf nodes as in related work (Dvořák, Ordyniak, and Szeider 2012). To this end, for a set $S \subseteq A$, a tree decomposition $(\mathcal{T}, \mathcal{X})$ of torso graph G_F^S, and $t \in \mathcal{T}$, let $X_{\geq t}$ be the union of all bags $X_s \in \mathcal{X}$ and sets *components*(s) for every node s that occurs in the subtree of $\mathcal{T}$ rooted at t. Further, $X_{>t}$ shall denote $X_{\geq t} \setminus X_t$.

Observe that for leaf nodes t of $\mathcal{T}$, we have $X_{>t} =$ *components*(t). Consequently, in the leaf nodes of $\mathcal{T}$, we guess a coloring on the backdoor-arguments X_t in the bag and check whether this coloring yields $X_{>t}$-restricted stable/complete extensions, cf. Definition 1. Note that as X_t for a leaf node $t \in \mathcal{T}$ is a backdoor set, this check can be performed in polynomial time. This follows with slight adaptations from the respective results by Dvořák, Ordyniak, and Szeider (2012). The remainder of the adapted DP algorithms proceeds as in the original algorithms for stable/complete extensions. In particular, every node maintains those colorings yielding $X_{>t}$-restricted stable/complete sets. Then, one can extend the correctness proofs for the modified algorithms returning valid colorings (see appendix for details).

Example 1 ctd. We evaluate the leaf node $3'$ of the given TD of the torso $G_F^{B'}$ for stable semantics on the AF $F_{\geq 3}$ (see below). Following the backdoor approach (Dvořák, Ordyniak, and Szeider 2012), we guess colors on the backdoor arguments $B' = \{s_1, s_3\}$ (gray background in illustration below), such that the set of arguments colored *in* is conflict-free in $F_{\geq 3}$, and each argument that is attacked by an *in*-argument is colored *def* ("defeated").

$F_{\geq 3}$:



Then, for each such guessed coloring λ we compute the *propagation* λ^* on the remaining non-backdoor arguments $x \in$ *components*$(3') = \{c_1, c_2, c_3, c_4\}$ (white background in illustration above): for stable semantics it suffices to guess whether an argument is *in* or *def*. Assume we guess $\lambda(s_1) =$ *in*, $\lambda(s_3) =$ *def*. Consequently, by applying the propagation rules, we obtain $\lambda^*(c_2) =$ *def*, $\lambda^*(c_1) =$ *in*, and $\lambda^*(c_3) = \lambda^*(c_4) =$ *def*. This corresponds to the B'-restricted stable set $\{s_1, c_1\}$. Now assume we guess $\lambda(s_1) = \lambda(s_3) =$ *def*. We obtain $\lambda^*(c_2) =$ *in*, and $\lambda^*(c_1) = \lambda^*(c_3) = \lambda^*(c_4) =$ *def*. This corresponds to the B'-restricted stable set $\{c_2\}$.

Limitations of Backdoor-Treewidth. The backdoor-treewidth approach, however, does not always work. In the following, we present several cases that do not admit fixed-parameter tractability when parameterized by backdoor-treewidth (under standard assumptions of complexity theory). In particular, this holds for the tractable fragments *BIP* and *SYM* (i.e., even though Theorem 1 guarantees us *finding SYM*-backdoors is fixed-parameter tractable, *reasoning* on an AF of constant *SYM*-backdoor-treewidth remains hard).

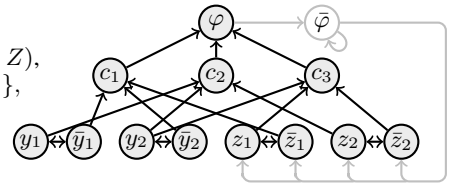
Proposition 3. *Reasoning on AFs F with $\mathtt{btw}_{\mathcal{C}}(F) = 0$ in $\sigma \in \{adm, com, pref, stb\}$ and $\mathcal{C} \in \{SYM, BIP\}$ remains* NP/coNP*-hard.*

Several fragments have the property that the otherwise Π_2^{P}-hard skeptical acceptance problem becomes coNP-easy, such as (a) odd-cycle-freeness (Dunne and Bench-Capon 2002), (b) no even-cycles of length ≥ 4 (Dvořák, König, and Woltran 2021) (subsuming the class of AFs that have no cycles longer than 3 arguments). We will denote these by (a) *ODDFREE* and (b) *EVENFREE-4*, respectively.

Proposition 4. *Skeptical acceptance for preferred semantics is Π_2^{P}-hard even for AFs F with $\mathtt{btw}_{\mathcal{C}}(F) = 0$ for $\mathcal{C} \in \{ODDFREE, EVENFREE\text{-}4\}$.*

Proof. Dvořák et. al. (2014) showed Π_2^{P} hardness for *Skept*$_{pref}$ for AFs F with $\mathtt{bd}_{ODDFREE}(F) = 1$ in a different context. The same reduction F_Φ (see example below) proves hardness for *EVENFREE-4* for $\mathtt{bd}_{EVENFREE\text{-}4}(F) = 1$.

F_Φ for
$\Phi = \forall Y \exists Z \varphi(Y, Z)$,
$\varphi = \{\{\bar{y}_1, \bar{y}_2, \bar{z}_1\},$
$\{y_1, y_2, z_2\},$
$\{y_2, z_1, \bar{z}_2\}\}$.



It suffices to remove the highlighted argument $\bar{\varphi}$, hence, $\mathtt{bd}_{\mathcal{C}}(F_\Phi) = 1$ and $\mathtt{btw}_{\mathcal{C}}(F_\Phi) = 0$. □

6 Systems and Benchmarks

We implemented both a system to find the exact (e.g., minimal) width of all *ACYC*-backdoors, as well as an argumentation problem solver using the backdoor-treewidth approach. In the latter, we solve the "Count Extensions"-problem in semantics $\sigma \in \{stb, adm\}$, where given a framework F we ask for the number of extensions $|\sigma(F)|$. This problem is gaining importance in the community, and has recently been added to the ICCMA competition (Mailly et al. 2021).

Benchmark Hardware. All our solvers ran on a cluster consisting of 12 servers. Each of these servers is equipped with two Intel Xeon E5-2650 CPUs, consisting of 12 physical cores that run at 2.2 GHz clock speed and have access to 256 GB shared main memory (RAM). Results are gathered on Ubuntu 16.04.1 LTS powered on kernel 4.4.0-139 with hyperthreading disabled using version 3.7.6 of Python3.

Benchmark Instances. As we focused on extreme values for backdoor size and treewidth, in addition to using instances provided by the ICCMA competition, we generated AFs to range from small to high backdoor sizes/treewidths. Then, we performed our experiments on a variety of composite frameworks. This is motivated by the modular nature of many frameworks that arise from applications. In particular, we combined instances provided by the ICCMA competition with frameworks designed to range from small to large backdoor-sizes and treewidths.

For scenario (A), we generated 960 instances that contain dense, directed subgraphs (high treewidth) and sparse subgraphs with many cycles (high backdoor size). For scenario (B), we combined small ICCMA instances with the generated frameworks from (A), resulting in 1134 instances:
(B1) 378 combinations of only ICCMA instances,
(B2) 378 combinations of only generated instances,
(B3) 378 combinations of ICCMA *and* generated instances.

Finding *ACYC*-Backdoors of Minimal Width

We propose a SAT encoding that, given an AF F and an integer k, produces a propositional formula $E(F, k)$ that is satisfiable if and only if $\texttt{btw}(F) \leq k$. We construct the formula in three steps: (i) we encode the definition of an *ACYC*-backdoor, (ii) we derive the corresponding torso graph, and (iii) we restrict the treewidth of the torso graph (Samer and Veith 2009). We then find the $\texttt{btw}(F)$ of an instance by incrementally increasing k until the SAT solver finds the formula satisfiable. For the 1134 instances of scenario B we computed $\texttt{btw}(F)$ using our SAT encoding. For 611 of the 1134 instances, that is more than half, $\texttt{btw}(F) < \texttt{tw}(F)$, and for most of them (519 out of 611), even $\texttt{btw}(F) \leq \texttt{tw}(F)/2$. Details are given in the appendix.

Utilizing Small Backdoor-Treewidth in Practice

We refer to the implemented system by argBTW[2]. It can easily be extended to the credulous/skeptical acceptance-problem, the existence of extensions-problem, etc. Our system argBTW uses *decomposition-guided reductions* to propositional satisfiability (Fichte et al. 2021). Such a reduction for argumentation reduces the respective argumentation problem to propositional logic such that the treewidth of the argumentation problem is (linearly) preserved. This way, we can adapt the existing NestHdb system (Hecher, Thier, and Woltran 2020) for propositional logic in order to efficiently implement dynamic programming for argumentation.

As an input, we get an AF in `apx` format. Then, four steps are performed (illustrated in Figure 2): (1) Find a suitable backdoor to *ACYC*. (2) Using this backdoor, compute the torso graph and decompose it into a suitable TD $(\mathcal{T}, \mathcal{X})$ of small width. (3) Perform tree decomposition guided reduction and get a propositional formula φ characterizing extensions of the AF. (4) Following the structure given by $\mathcal{T}$, we apply a dynamic programming algorithm on φ to determine the number of models of φ, i.e., the number of extensions.

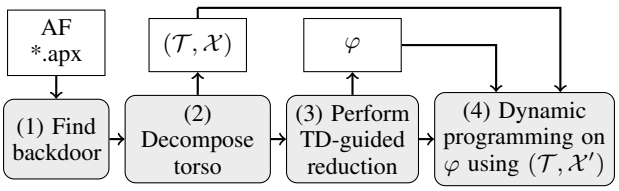


Figure 2: Flowchart of the implemented argBTW system.

In the previous section, we discussed an approach for computing the backdoor-treewidth exactly. For practical purposes and in order solve reasonably-sized instances efficiently, we rely on heuristics, i.e., Steps (1) and (2) as described above do not use exact methods. For (1) we apply an *Answer Set Programming (ASP)* encoding: for (2) we employ the htd system (Abseher, Musliu, and Woltran 2017).

For Step (1) we allotted at most 30 seconds and Step (2) is finished within a second per instance, due to the decomposer htd being reasonably fast. Step (4) also includes a simple preprocessing phase, which aims at simplifying instances quickly via, e.g., techniques like unit propagation.

Empirical Evaluation

The performed experiments aimed at identifying graph structures where the backdoor-treewidth approach is particularly beneficial. To this end, we considered two scenarios:
(A) We compared the backdoor-treewidth approach to an backdoor-only and a treewidth-only approach.
(B) As the backdoor-treewidth approach turned out to be the fastest of the three, we conducted experiments where we compared this method on counting extensions against an ASP approach (ASPARTIX, Dvořák et al. 2020).

For Scenario (A), we are interested in comparing our technique based on backdoor-treewidth with the special (degenerated) cases of treewidth and backdoor size. Then, the conducted experiments of Scenario (B) aim at examining the runtime behavior of the "Count Extensions"-problem. This problem is quite general and can very easily be adapted to decide credulous/skeptical acceptance of an argument, existence of a (non-empty) extension, and other well established argumentation problems. In our experiments, we mainly compare wall clock time and number of solved instances over the best of two runs for every solver and instance, where each run is allowed to use up to 1200s of wall clock time and 16GB of main memory (RAM).

The goal of the two scenarios (A) and (B), is to empirically confirm the following hypotheses.
(H1_A) There are instances where backdoor-treewidth based solvers outperform (tree)width and backdoor approaches.
(H2_A) In practice, the obtained backdoor-treewidths are often smaller than the computed widths and backdoor sizes.

[2]Find argBTW and benchmarks at github.com/mk-tu/argBTW.

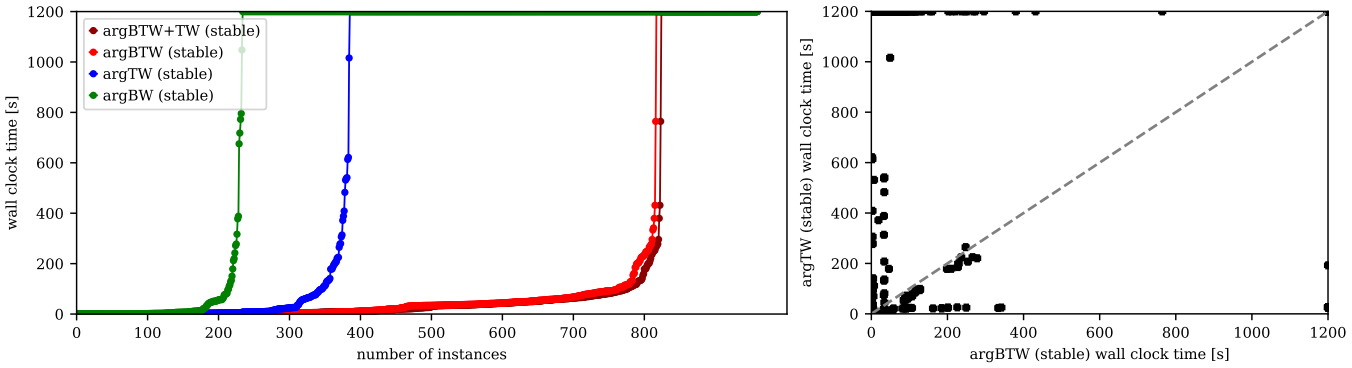


Figure 3: Performance comparison of argBTW, argTW and argBW for Scenario (A). The cactus plot (left) shows the number of solved instances (x-axis) that is sorted in ascending order for each approach individually according to runtime. The scatter plot (right) depicts a one-to-one comparison between the runtime (in seconds) of every instance for BTW and TW.

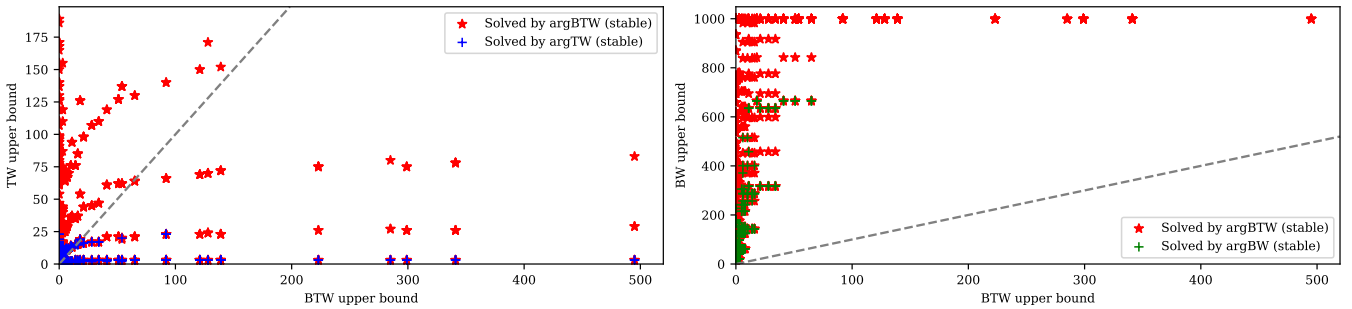


Figure 4: Measures among solved instances of Scenario (A). We compare (left) upper bounds of backdoor-treewidth (BTW) against treewidth (TW) as well as (right) upper bounds of backdoor-treewidth (BTW) against backdoor-size (BW).

(H_B) For counting extensions, the backdoor-treewidth based solver and suitable portfolio configurations combining argBTW and argTW often outperform existing approaches based on ASP.

Benchmarked Systems. In our experiments, we mainly compare the performance of the following configurations.

- clingo[3] (stable) and clingo (admissible): these configurations are based on clingo version 5.4.0 and we used arguments "-q" and "-n 0" when counting answer sets. On top we used standard encodings for admissible and stable semantics of ASPARTIX (Dvořák et al. 2020).
- argBTW (stable) and argBTW (admissible): see above.
- argBW (stable) and argBW (admissible): this is a special case of argBTW, where instead of computing and decomposing a torso of an AF, after finding a backdoor B we solve the instance via the backdoor-only approach.
- argTW (stable) and argTW (admissible): in this special case of argBTW, we take all arguments A of an AF F as a backdoor, obtaining the undirected graph U^F as a torso, which is then solved with the treewidth-only approach.
- argBTW+TW (stable) and argBTW+TW (admissible): This is a portfolio that emerged from our study. First, 200s are used for solving via argTW and then if unsuccessful, the remaining 1000s are put into argBTW.

[3]ASP solver, see https://potassco.org/

Results of Scenario (A). An overview of the results for Scenario (A) is depicted in Figure 3. Figure 3 (left) shows a cactus plot over the different configurations of our solver for computing stable extensions. Such a cactus plot depicts for each configuration the number of solved instances (x-axis) according to their runtime (y-axis) in ascending order. Consequently, one sees that overall the backdoor-treewidth based approach argBTW outperforms the other configurations. The picture is similar for admissible extensions (not shown in the plot). To enable a one-to-one comparison, Figure 3 (right) depicts a scatter plot, where the runtime of each instance is compared between argBTW (x-axis) and the second best configuration of Figure 3 (left), namely argTW (y-axis). The argBTW approach is often faster and solves more instances, thereby confirming Hypothesis ($H1_A$), however, there are also some instances below the dashed diagonal, which indicates instances, where argTW is faster. This scatter plot gives rise to the portfolio configuration argBTW+TW in order to further improve the performance, since almost all of these dots can be solved by argTW in a runtime below or even well below 200 seconds. While the runtime benefits of argBTW are certainly interesting, in order to evaluate Hypothesis ($H2_A$), we analyze the obtained upper bounds of backdoor-treewidths, treewidths and backdoor sizes. The values of these measures among all solved instance are compared in Figure 4, where (left) compares

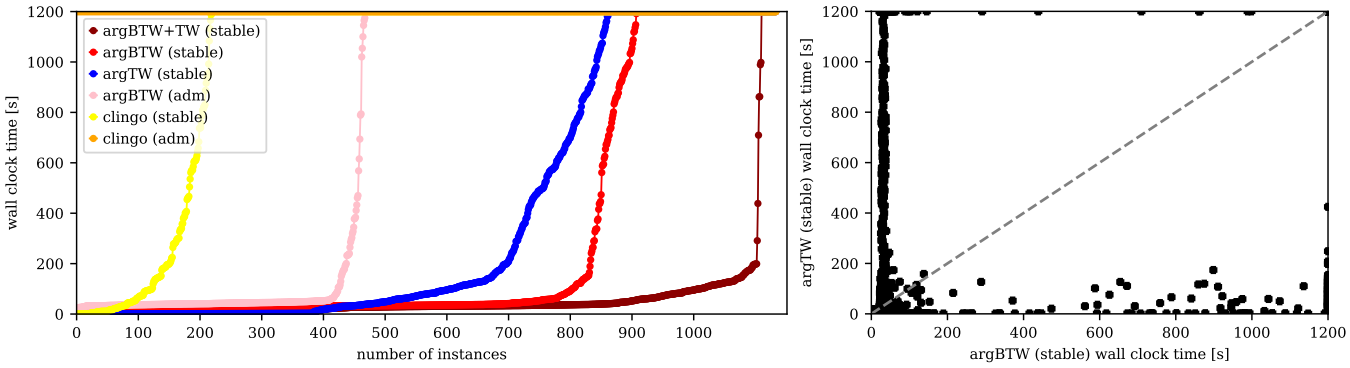


Figure 5: Performance of compared solvers for Scenario (B). The cactus plot (left) shows the BTW approach for admissible and stable semantics. The scatter plot (right) allows a one-to-one comparison of solved instances for BTW vs. TW approach.

the backdoor-treewidth upper bounds (x-axis) to the used widths (y-axis) and (right) compares backdoor-treewidth upper bounds (x-axis) with used backdoor sizes (y-axis). These scatter plots allow a one-to-one comparison of these measured values for each instance solved by the respective configuration. Overall, one can observe that while there are some instances where we obtain larger backdoor-treewidth than treewidth upper bounds (cf. dots below the diagonal of Figure 4 (left)), the obtained backdoor size is always dominated by backdoor-treewidth upper bounds as shown in Figure 4 (right). Both observations confirm Hypothesis $H2_A$.

Results of Scenario (B). We took the best configurations of Scenario (A) in order to count extensions over instances of different kind. Figure 5 presents the overall results of this experiment, where in (left) we show a cactus plot comparing our approach with the treewidth-based technique as well as solutions based on logic programming (ASP). Further, Figure 5 (right) completes this analysis by a one-to-one comparison of argBTW vs. argTW and explains why the portfolio argBTW+TW is promising. In particular, this plot also reveals that there is still further potential for improving our heuristics on approximating decent backdoor-treewidth upper bounds. We provide detailed results in Table 1, revealing that the portfolio argBTW+TW shows indeed a significant performance increase compared to argBTW. Overall, we can confirm Hypothesis H_B for counting extensions.

Solver	$\sum$	Width Range				Time[h]
		max(width)	0-5	5-10	>10	
		B1				
argTW (stable)	**378**	**16.0**	**29**	**163**	**186**	**0.18**
argBTW (stable)	279	15.0	**29**	158	92	43.45
argBTW (adm)	37	14.0	23	12	2	116.72
clingo (stable)	31	10.0	17	14	0	117.54
clingo (adm)	0	0	0	0	0	126.0
		B2				
argBTW (stable)	**377**	**3.0**	**377**	0	0	**3.72**
argBTW (adm)	**377**	**3.0**	**377**	0	0	4.71
argTW (stable)	184	**3.0**	184	0	0	94.5
clingo (stable)	129	**3.0**	129	0	0	91.03
clingo (adm)	0	0	0	0	0	126.0
		B3				
argTW (stable)	**300**	**16.0**	36	102	**162**	**31.79**
argBTW (stable)	252	15.0	**46**	**133**	73	52.19
clingo (stable)	60	**16.0**	28	23	9	109.33
argBTW (adm)	54	11.0	44	9	1	110.67
clingo (adm)	0	0	0	0	0	126.0
		Σ				
argBTW+TW (stable)	**1110**	**16.0**	**452**	**296**	**362**	**19.99**
argBTW (stable)	**908**	15.0	**452**	291	165	99.36
argTW (stable)	862	**16.0**	249	265	348	126.47
argBTW (adm)	468	14.0	444	21	3	232.1
clingo (stable)	220	**16.0**	174	37	9	317.9
clingo (adm)	0	0	0	0	0	378.0

Table 1: Detailed benchmark data for Scenario (B), where we show for each configuration the number of solved instances, grouped by respective width upper bounds, as well as the overall runtime (timeouts account for 1200s).

7 Conclusion

In this work, we introduced the parameter backdoor-treewidth for argumentation, which dominates the well established parameters treewidth and minimum backdoor size. We gave algorithms to compute backdoors of width $\leq k$ that establish fixed-parameter tractability (w.r.t. k) for the fragment *SYM* in the general case, and for the fragment *ACYC* for chordal AFs. Moreover, we showed fixed-parameter tractability for solving several problems associated with AFs if a suitable backdoor to one of the fragments *ACYC* or *NOEVEN* is given. On the other hand, we established computational hardness for these problems in other fragments, effectively pointing out limitations of the new backdoor-treewidth approach. Finally, we conducted systematic experiments, empirically evaluating the power of the newly introduced parameter. Here, we presented systems for both finding the exact backdoor-treewidth (as well as the associated backdoor set and a suitable tree decomposition of the torso), and a composite system for solving argumentation problems. We found several graph structures where backdoor-treewidth approach outperforms its "parent"-parameters treewidth and minimum backdoor size. However, as Figure 4 suggests, the hereby used heuristics for finding adequate backdoor sets can still be improved, i.e., there is further open potential for estimating tree decompositions of decent backdoor-treewidth. Further future work regards investigations for other semantics and fragments, as well as the connection to structured argumentation.

Acknowledgments

This research has been supported by the Vienna Science and Technology Fund (WWTF) through project ICT19-065, and by the Austrian Science Fund (FWF) through projects P30168, P32441, P32830, W1255, and Y698. Part of the project was carried out while Hecher, Schidler, and Szeider were visiting the Simons Institute for the Theory of Computing. Hecher is also affiliated with the university of Potsdam, Germany.

References

Abseher, M.; Musliu, N.; and Woltran, S. 2017. htd - A Free, Open-Source Framework for (Customized) Tree Decompositions and Beyond. In Salvagnin, D.; and Lombardi, M., eds., *Integration of AI and OR Techniques in Constraint Programming - 14th International Conference, CPAIOR 2017, Padua, Italy, June 5-8, 2017, Proceedings*, volume 10335 of *LNCS*, 376–386. Springer.

Arnborg, S.; Corneil, D. G.; and Proskurowski, A. 1987. Complexity of finding embeddings in a k-tree. *Siam Journal of Discrete Mathematics*, 8(2): 277–284.

Bailleux, O.; and Boufkhad, Y. 2003. Efficient CNF Encoding of Boolean Cardinality Constraints. In Rossi, F., ed., *Principles and Practice of Constraint Programming - CP 2003*, volume 2833 of *LNCS*, 108–122. Springer.

Baroni, P.; Dunne, P. E.; and Giacomin, M. 2010. On Extension Counting Problems in Argumentation Frameworks. In Baroni, P.; Cerutti, F.; Giacomin, M.; and Simari, G. R., eds., *Proceedings of the 3rd Conference on Computational Models of Argument (COMMA 2010)*, volume 216 of *Frontiers in Artificial Intelligence and Applications*, 63–74. Desenzano del Garda, Italy: IOS Press. ISBN 978-1-60750-618-8.

Bodlaender, H. L. 2005. Discovering treewidth. In *Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'05)*, volume 3381 of *LNCS*, 1–16. Springer.

Caminada, M.; and Gabbay, D. M. 2009. A logical account of formal argumentation. *Studia Logica*, 93(2): 109–145.

Cerutti, F.; Gaggl, S. A.; Thimm, M.; and Wallner, J. P. 2018. Foundations of Implementations for Formal Argumentation. In Baroni, P.; Gabbay, D.; Giacomin, M.; and van der Torre, L., eds., *Handbook of Formal Argumentation*, chapter 14, 688–767. College Publications. Also appears in IfCoLog Journal of Logics and their Applications 4(8):2623–2706.

Charwat, G. 2012. *Tree-decomposition based Algorithms for Abstract Argumentation Frameworks*. Master's thesis, TU Wien.

Coste-Marquis, S.; Devred, C.; and Marquis, P. 2005. Symmetric Argumentation Frameworks. In Godo, L., ed., *Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2005)*, volume 3571 of *LNCS*, 317–328. Springer. ISBN 3-540-27326-3.

Dung, P. M. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artif. Intell.*, 77(2): 321–358.

Dunne, P. E. 2007. Computational properties of argument systems satisfying graph-theoretic constraints. *Artif. Intell.*, 171(10-15): 701–729.

Dunne, P. E.; and Bench-Capon, T. J. M. 2001. Complexity and Combinatorial Properties of Argument Systems. Technical report, Dept. of Computer Science, University of Liverpool.

Dunne, P. E.; and Bench-Capon, T. J. M. 2002. Coherence in Finite Argument Systems. *Artif. Intell.*, 141(1/2): 187–203.

Dvořák, W.; and Dunne, P. E. 2017. Computational Problems in Formal Argumentation and their Complexity. *FLAP*, 4(8).

Dvořák, W.; Gaggl, S. A.; Rapberger, A.; Wallner, J. P.; and Woltran, S. 2020. The ASPARTIX System Suite. In *COMMA*, volume 326 of *Frontiers in Artificial Intelligence and Applications*, 461–462. IOS Press.

Dvořák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2014. Complexity-sensitive decision procedures for abstract argumentation. *Artificial Intelligence*, 206(0): 53 – 78.

Dvořák, W.; König, M.; and Woltran, S. 2021. On the Complexity of Preferred Semantics in Argumentation Frameworks with Bounded Cycle Length. In Bienvenu, M.; Lakemeyer, G.; and Erdem, E., eds., *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, November 3-12, 2021*, 671–675.

Dvořák, W.; Ordyniak, S.; and Szeider, S. 2012. Augmenting tractable fragments of abstract argumentation. *Artificial Intelligence*, 186(0): 157–173.

Dvořák, W.; Pichler, R.; and Woltran, S. 2012. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186: 1 – 37.

Fichte, J. K.; Hecher, M.; Mahmood, Y.; and Meier, A. 2021. Decomposition-Guided Reductions for Argumentation and Treewidth. In Zhou, Z., ed., *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, 1880–1886. ijcai.org.

Fichte, J. K.; Hecher, M.; and Meier, A. 2019. Counting Complexity for Reasoning in Abstract Argumentation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, 2827–2834. AAAI Press. ISBN 978-1-57735-809-1.

Ganian, R.; Ramanujan, M. S.; and Szeider, S. 2017a. Backdoor Treewidth for SAT. In Gaspers, S.; and Walsh, T., eds., *Theory and Applications of Satisfiability Testing - SAT 2017*, volume 10491 of *LNCS*, 20–37. Springer.

Ganian, R.; Ramanujan, M. S.; and Szeider, S. 2017b. Combining Treewidth and Backdoors for CSP. In Vollmer, H.; and Vallée, B., eds., *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, volume 66 of *LIPIcs*, 36:1–36:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Gottlob, G.; and Szeider, S. 2008. Fixed-parameter algorithms for artificial intelligence, constraint satisfaction, and database problems. *The Computer Journal*, 51(3): 303–325.

Hecher, M.; Thier, P.; and Woltran, S. 2020. Taming High Treewidth with Abstraction, Nested Dynamic Programming, and Database Technology. In *Theory and Applications of Satisfiability Testing - SAT 2020*, volume 12178 of *LNCS*, 343–360. Springer.

Kloks, T. 1994. *Treewidth, Computations and Approximations*, volume 842 of *LNCS*. Springer. ISBN 3-540-58356-4.

Mailly, J.-G.; Lonca, E.; Lagniez, J.-M.; and Rossit, J. 2021. International Competition on Computational Models of Argumentation (ICCMA) 2021. Available at: https://www.argumentationcompetition.org/2021.

Martins, R.; Joshi, S.; Manquinho, V.; and Lynce, I. 2014. Incremental Cardinality Constraints for MaxSAT. In *Principles and Practice of Constraint Programming*, 531–548. Springer International Publishing.

Robertson, N.; and Seymour, P. D. 1986. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7(3): 309–322.

Samer, M.; and Veith, H. 2009. Encoding Treewidth into SAT. In *Theory and Applications of Satisfiability Testing - SAT 2009. Proceedings*, volume 5584 of *LNCS*, 45–50. Springer.