Heuristic Algorithms for Credulous and Skeptical Reasoning Problems in Abstract Argumentation

Matthias Thimm
Artificial Intelligence Group, University of Hagen, Germany

Abstract

We consider problems of credulous and skeptical reasoning in abstract argumentation under a variety of semantics and present algorithms for *heuristically* solving these, i. e., we present algorithms that do not necessarily always give the correct answer but are more performant than correct algorithms. Our algorithms are based on using grounded semantics as a proxy for deciding acceptability wrt. other semantics and on bounded search for defenders. We perform a comprehensive experimental evaluation that shows competitive performance of our approaches.

1 Introduction

Abstract argumentation frameworks (AAFs) [14] are approaches for modelling argumentative scenarios that represent arguments as nodes in a directed graph, where a directed edge represents an attack from one argument to another. Reasoning in AAFs consists of identifying sets of arguments (*extensions*) that form a *plausible* outcome of the argumentation. Several different semantical approaches for formalising plausibility in this context have been proposed, see e. g. [14, 2]. Central to most of these approaches are the concepts of *conflict-freeness* and *admissibility*, which model the requirements that a plausible outcome should be free of internal conflicts and defend itself against threats from the outside, respectively. Many different semantics have been proposed over the years based on these notions, but also based on different concepts such as rankings [1, 32], weights [16, 5], or probabilities [26, 21].

In this work, we are concerned with *algorithmic* aspects of AAFs. Most interesting reasoning problems, such as deciding whether there exists an admissible set containing a given argument, are computationally hard [17] and therefore requiring sophisticated algorithms to be solved effectively. In scenarios, where runtime performance is of a larger concern than correctness, *heuristic algorithms* can be used.[1] Interest into heuristic algorithms for abstract argumentation in particular, but also other areas concerned with symbolic reasoning [31, 6, 38],

[1]Note that we will be using the term *heuristic algorithms* here in a broad sense to denote algorithms that solve some decision problem faster than a verified correct algorithm, but give no formal guarantee whether the result is correct. Note that in previous works on heuristic

has increased quite a bit with the availability of powerful machine learning-based models. Works such as [24, 10, 27] use neural networks, which are trained on large sets of already classified instances, to accurately predict the correct answers to reasoning problems such as deciding whether there exists an admissible set containing a given argument. However, in this paper, we actually argue against the need to use machine learning-based models for heuristically solving abstract argumentation problems and present a series of simple heuristic algorithms, based on insights of the structure of abstract argumentation frameworks and the different semantics. More precisely, we will be using *grounded semantics* [14] as a proxy for deciding reasoning problems with a series of other semantics, motivated by insights from [8]. Furthermore, we present additional algorithms that use *bounded search* to solve instances that cannot be clearly decided by relying on grounded semantics. These algorithms implement simple exhaustive search algorithms for finding, in particular, admissible sets, but bound the search to not consider all possibilities and thereby keep the runtime low. We experimentally compare our approaches with state-of-the-art algorithms and show that our algorithms are competitive.

To summarise, the contributions of this paper are as follows:

- We present and discuss algorithms using grounded semantics to heuristically solve a wide range of reasoning problems (Section 4).
- We present and discuss algorithms based on bounded search to heuristically solve such problems (Section 5).
- We experimentally evaluate these algorithms (Section 6).

In addition, Section 2 presents preliminaries on abstract argumentation, Section 3 discusses related works, and Section 7 concludes.

Note that parts of the work described in this paper have already been documented in the (non-peer reviewed) system descriptions [33, 34]. In addition to a more detailed discussion of these approaches, the present paper also contains a comprehensive experimental evaluation.

2 Abstract argumentation

An *abstract argumentation framework* F is a tuple $F = (A, R)$ where A is a (finite) set of arguments and R is a relation $R \subseteq A \times A$ [14]. For two arguments $a, b \in A$ the relation aRb means that argument a attacks argument b. For a set

algorithms for abstract argumentation, e. g., [24, 10, 27], such algorithms have been presented using the term *approximation algorithm*. However, this term is actually not adequate, as an approximation algorithm is an algorithm for solving an *optimisation problem* and has a guaranteed theoretical approximation quality, cf. [37]. Here, we are concerned with *decision problems* for which we can usually not provide any formal guarantees on correctness, so we will be using the term *heuristic algorithm*, see also [35].

$S \subseteq A$ we define

$$S^+_F = \{a \in A \mid \exists b \in S, bRa\}$$
$$S^-_F = \{a \in A \mid \exists b \in S, aRb\}$$

If S is a singleton set, i. e., $S = \{a\}$ for some $a \in A$, then we also just write a^+_F (resp. a^-_F) for $\{a\}^+_F$ (resp. $\{a\}^-_F$).

We say that a set $S \subseteq A$ is *conflict-free* if for all $a, b \in S$ it is not the case that aRb. A set S *defends* an argument $b \in A$ if for all a with aRb there is $c \in S$ with cRa. A conflict-free set S is called *admissible* if S defends all $a \in S$.

Different semantics [2] can be phrased by imposing certain constraints on sets of arguments. In particular, a set E

- is a *complete* (co) extension iff it is admissible and for all $a \in A$, if E defends a then $a \in E$,
- is a *grounded* (gr) extension iff it is complete and minimal,
- is a *stable* (st) extension iff it is conflict-free and $E \cup E^+_F = A$,
- is a *preferred* (pr) extension iff it is admissible and maximal.
- is a *semi-stable* (sst) extension iff it is complete and $E \cup E^+_F$ is maximal.
- is a *stage* (stg) extension iff it is conflict-free and $E \cup E^+_F$ is maximal.
- is an *ideal* (id) extension iff it is admissible, $E \subseteq E'$ for each preferred extension E', and E is maximal.

All statements on minimality/maximality are meant to be with respect to set inclusion.

Given an abstract argumentation framework $F = (A, R)$ and a semantics $\sigma \in \{\mathsf{co}, \mathsf{gr}, \mathsf{st}, \mathsf{pr}, \mathsf{sst}, \mathsf{stg}, \mathsf{id}\}$ we are interested in the following computational problems [36, 17]:

DC-σ: For a given argument a, decide whether a is in at least one σ-extension of F.

DS-σ: For a given argument a, decide whether a is in all σ-extensions of F.

Here DC stands for the <u>d</u>ecision problem for <u>c</u>redulous reasoning and DS stands for the <u>d</u>ecision problem for <u>s</u>keptical reasoning.

Note that DC-σ=DS-σ for $\sigma \in \{\mathsf{gr}, \mathsf{id}\}$ as both the ideal and the grounded extension is uniquely defined [2]. Moreover, we have DC-co=DC-pr as every preferred extension is complete and every complete extension can be extended to a preferred extension [14]. Furthermore, DS-co=DC-gr as the intersection of all complete extensions is the grounded extension [14]. Since reasoning with grounded semantics can be done in polynomial time [17], we do not consider it here for the purpose of developing heuristic algorithms. Due to DC-id=DS-id, we only consider DS-id and not DC-id. In the remainder of this paper, we will therefore consider the computational problems

$$\texttt{Prob}_C = \{\mathsf{DC\text{-}co}, \mathsf{DC\text{-}st}, \mathsf{DC\text{-}sst}, \mathsf{DC\text{-}stg}\}$$

for credulous reasoning and the problems

$$\texttt{Prob}_S = \{\mathsf{DS\text{-}pr}, \mathsf{DS\text{-}st}, \mathsf{DS\text{-}sst}, \mathsf{DS\text{-}stg}, \mathsf{DS\text{-}id}\}$$

for skeptical reasoning. Let furthermore $\texttt{Prob} = \texttt{Prob}_C \cup \texttt{Prob}_S$.

3 Related works

There exist several approaches for solving problems in Prob *exactly*, as witnessed by the International Competition on Computational Models of Argumentation[2] (ICCMA). Most popular and successful are approaches based on reductions to the satisfiability problem or using answer set programming, see [7] for a survey and [22] for some recent developments.

The first works on *heuristically* solving (some) problems in Prob are based on using neural networks [24, 10, 27]. These works use (standard or specifically tailored versions of) *graph convolutional neural networks* [23] to represent problems in Prob as classification problems. These approaches have to be trained on automatically generated training data, which can introduce bias issues [25]. In our experimental evaluation (see Section 6), we include the AFGCNv2 solver [28], which was the only participant in the approximate track of ICCMA23 that was based on a neural network architecture.

Further heuristic approaches are founded on the observation that, at least *empirically*, reasoning with any semantics often is very close to reasoning with grounded semantics [8]. More precisely, the grounded extension coincides quite often with the set of skeptically accepted arguments wrt. any semantics. The algorithmic approach of harper++ [34] is based on this observation and only uses a reasoner for grounded semantics (which runs in polynomial time) to heuristically solve all problems in Prob. We will discuss this approach in more detail in Section 4. Recently, Delobelle, Mailly, Rossit [11] refined this idea by considering additional heuristics for cases, where using only the grounded reasoning approach is insufficient. This work resulted in two working systems ARIPOTER-Degrees [12] and ARIPOTER-HCAT [13] that also participated in the approximate track of ICCMA23. For arguments that are neither included in nor attacked by the grounded extension, ARIPOTER-Degrees uses the ratio of in- and out-degree of the arguments to decide their acceptance (the larger the out-degree and smaller the in-degree, the more likely the argument is accepted). The solver ARIPOTER-HCAT uses the values from the *h-categorizer* approach [3] to decide the acceptance of those arguments instead (larger values of the h-categorizer approach indicate a more likely acceptance).

[2] http://argumentationcompetition.org

Algorithm 1 GR-DC algorithm for credulous reasoning wrt. $\sigma \in \{\mathsf{co}, \mathsf{st}, \mathsf{sst}, \mathsf{stg}\}$

```
Input:    $F = (A, R)$, $a \in A$
Output:   TRUE iff $a$ is contained in a $\sigma$-extension.
GR-DC($F, a$)
1: $S = \mathsf{grounded}(F)$
2: if $a \in S_F^+$ then return FALSE
3: return TRUE
```

Yet another approach is implemented in the FARGO solver [33], which also participated in the approximate track of ICCMA23. In contrast to ARIPOTER-DEGREES and ARIPOTER-HCAT, FARGO does not rely on graph-theoretic features but uses an incomplete approach to decide admissibility of sets. We will discuss that approach in detail in Section 5.

4 Using grounded semantics as a heuristic

It is well-known that the grounded extension E_{gr} of an abstract argumentation framework $F = (A, R)$ is contained in every complete, preferred, stable, semi-stable, and ideal extension [14, 15]. It follows that, if the answer to DC-gr for an argument a in a framework F is TRUE—so if $a \in E_{\text{gr}}$—, then the answer for DC-co, DC-sst, DS-pr, DS-st, DS-sst, and DS-id is TRUE as well.[3] Moreover, if we have an argument a that is *attacked* by the grounded extension—so if $a \in E_{\text{gr}}^+$—, then it is also clear that it cannot be contained in any complete, preferred, stable, semi-stable, and ideal extension (as such extensions contain E_{gr} and are conflict-free). So in that case, we necessarily have that the answer to DC-co, DC-st, DC-sst, DS-pr, DS-sst, and DS-id is FALSE.[4]

Besides the two theoretically valid facts from above, the work [8] also motivates the use of grounded semantics as a heuristic from an empirical perspective. In essence, [8] shows that for a series of random graph models (in particular those used in the ICCMA competitions) skeptically reasoning with, e. g., preferred semantics is identical (or very close) to reasoning with grounded semantics, since cases where an argument is contained in all preferred extensions but not in the grounded extension are very rare.

By the above motivation, we consider the algorithms GR-DC (see Algorithm 1) resp. GR-DS (see Algorithm 2) as heuristic approaches to solve all problems in $\texttt{Prob}_C$, resp. $\texttt{Prob}_S$. Note that although stage semantics is not included in the (theoretical) discussion above, we still consider it here and also empirically evaluate it in Section 6. Our approaches for credulous (GR-DC) and skeptical (GR-DS) reasoning based on grounded semantics are very similar and

[3]Note that this is not necessarily true for DC-st since F may not have a stable extension. However, note that the statement is indeed true for DS-st, as in the case $\mathsf{st}(F) = \emptyset$, every argument is, by definition, skeptically accepted wrt. stable semantics.

[4]Note that this is not necessarily true for DS-st but is indeed true for DC-st due to the reasoning in the previous footnote.

Algorithm 2 GR-DS algorithm for skeptical reasoning wrt. $\sigma \in \{\mathsf{pr}, \mathsf{st}, \mathsf{sst}, \mathsf{stg}, \mathsf{id}\}$

```
Input:   $F = (A, R)$, $a \in A$
Output:  TRUE iff $a$ is contained in all $\sigma$-extensions.
GR-DS($F, a$)
1: $S = \mathsf{grounded}(F)$
2: if $a \in S$ then return TRUE
3: return FALSE
```

differ only in one aspect. For both algorithms, upon input F and a, we first determine the grounded extension of F (with $\mathsf{grounded}(F)$ in line 1 of both Algorithm 1 and 2). This can be done in polynomial time using, e. g., the algorithm from [30]. For GR-DC we answer with FALSE if the argument a is attacked by the grounded extension. In all other cases (so if a is in the grounded extension or not connected to it at all) we answer with TRUE. For GR-DS we answer with TRUE only if a is in the grounded extension. In all other cases we answer with FALSE.

5 Bounded search

While using grounded semantics as a heuristic for solving problems in `Prob` can be practically successful (see Section 6), there are still some cases where Algorithms GR-DC and GR-DS provide an uninformed answer. In particular, the algorithm GR-DC defaults to the answer TRUE and GR-DC defaults to the answer FALSE for arguments that are neither contained in nor attacked by the grounded extension. Addressing these cases is the focus of the algorithms developed in this section.

The core of our approach lies in (two variants of) an algorithm for heuristically determining whether an argument a is contained in an admissible set (recall that a set S is admissible if it is conflict-free and defends all its elements). The general algorithm is an implementation of an exhaustive search algorithm that tries to extend a given set (initialised with a set just containing the query argument a) by arguments that defend arguments from the set that are not yet defended. This search algorithm is given a bound on the runtime and therefore is not guaranteed to find an admissible set if one exists. The two variants we will discuss differ in the type of bound that is given. In Section 5.1 we discuss an algorithm that bounds the depth of the search and in Section 5.2 we discuss a variant that bounds the number of calls of the search routine.

5.1 Depth-bounded search

The general depth-bounded search algorithm DB-SEARCH is shown in Algorithm 3, which is a variant of the standard DPLL-search algorithm [4], where the search direction is influenced by the attack directions. This algorithm has

Algorithm 3 DB-SEARCH algorithm for verifying whether a given subset can be extended to an admissible set

```
Input:    $F = (A, R)$, $S \subseteq A$, $n \in \mathbb{N} \cup \{\infty\}$
Output:   TRUE iff there is admissible $S'$ with $S \subseteq S'$.
DB-SEARCH$(F, S, n)$
 1: if $S$ is admissible then
 2:     return TRUE
 3: if $n \leq 0$ then
 4:     return FALSE
 5: for $b \in S_F^- \setminus S_F^+$ do
 6:     if $b_F^- \setminus (S_F^- \cup S_F^+) = \emptyset$ then
 7:         return FALSE
 8:     for $c \in b_F^- \setminus (S_F^- \cup S_F^+)$ do
 9:         if DB-SEARCH$(F, S \cup \{c\}, n-1)$ then
10:             return TRUE
11: return FALSE
```

three parameters $F = (A, R)$, $S \subseteq A$, $n \in \mathbb{N} \cup \{\infty\}$ and is supposed to return TRUE if and only if there is an admissible set S' with $S \subseteq S'$. The third parameter n gives a bound on the search depth that may impair the correctness of the algorithm. More precisely, if the search depth is not ∞, it may happen that the algorithm does not find an admissible set, even if it exists.

The algorithm DB-SEARCH works on input F, a, and n as follows. If S is admissible (which can be checked in polynomial time), we are already finished and terminate with TRUE (lines 1–2). If we have reached the bound of the search, we terminate with FALSE (lines 3–4). Otherwise (lines 5–10), for each argument b that is attacking some argument in S but not attacked back[5] (i. e., those arguments in $S_F^- \setminus S_F^+$), we check if there is a potential defender c, i. e., an argument that could be added to S without violating conflict-freeness of S and that attacks b. For each such candidate, we recursively check whether the set $S \cup \{c\}$ can be extended to an admissible set. If the search fails, we return FALSE (line 11).

We summarise the formal properties of DB-SEARCH(F, S, n) in the following result, which is given without proof but should be clear from the discussion above.

Proposition 1. *Let $F = (A, R)$, $S \subseteq A$, and $n \in \mathbb{N} \cup \{\infty\}$.*

1. *If DB-SEARCH(F, S, n) = TRUE then there is an admissible set S' with $S \subseteq S'$.*
2. *If DB-SEARCH(F, S, n) = FALSE and $n = \infty$ then there is no admissible set S' with $S \subseteq S'$.*

[5]Since S is not admissible, either such a b must exist or S is not conflict-free. In the latter case, the algorithm right away returns FALSE in line 11.

Algorithm 4 DB-DC algorithm for credulous reasoning wrt. $\sigma \in \{\mathsf{co}, \mathsf{st}, \mathsf{sst}, \mathsf{stg}\}$

```
Input:    $F = (A, R)$, $a \in A$, $n \in \mathbb{N} \cup \infty$
Output:   TRUE iff $a$ is contained in a $\sigma$-extension.
DB-DC($F, a, N$)
1: $S = \mathsf{grounded}(F)$
2: if $a \in S$ then return TRUE
3: if $a \in S_F^+$ then return FALSE
4: return DB-SEARCH($F, S \cup \{a\}, n$)
```

Algorithm 5 DB-DS algorithm for skeptical reasoning wrt. $\sigma \in \{\mathsf{pr}, \mathsf{st}, \mathsf{sst}, \mathsf{stg}, \mathsf{id}\}$

```
Input:    $F = (A, R)$, $a \in A$, $n \in \mathbb{N} \cup \infty$
Output:   TRUE iff $a$ is contained in all $\sigma$-extensions.
DB-DS($F, a, N$)
1: $S = \mathsf{grounded}(F)$
2: if $a \in S$ then return TRUE
3: if $a \in S_F^+$ then return FALSE
4: if not DB-SEARCH($F, S \cup \{a\}, n$) then return FALSE
5: for $b \in a_F^- \setminus S_F^+$ do
6:     if DB-SEARCH($F, S \cup \{b\}, n$) then return FALSE
7: return TRUE
```

So note that the algorithm DB-SEARCH is, in general, an *incomplete algorithm.* If the algorithm returns TRUE, we can be sure that this is the correct answer. If the algorithm returns FALSE, the answer may be correct or not (only for $n = \infty$ it is guaranteed that the answer is correct).

We embed the algorithm DB-SEARCH in the algorithm DB-DC for solving problems in $\texttt{Prob}_C$ as shown in Algorithm 4. In lines 1–3 we first check whether the query argument is contained in or attacked by the grounded extension and return the corresponding answer (see Section 4). For the remaining cases, we check whether we can find an admissible set containing $S \cup \{a\}$ using DB-SEARCH. Therefore, this algorithm uses admissibility as a heuristic for all semantics to solve problems in $\texttt{Prob}_C$ (while it is generally true, that if an argument a is contained in an admissible set, it is necessarily also contained in a complete and preferred extension, this is not generally true for semi-stable, stable, and stage semantics; but note that any such extension must necessarily also contain the grounded extension, which is why we start from $S \cup \{a\}$ with S being the grounded extension).

As for skeptical reasoning, we embed the algorithm DB-SEARCH in the algorithm DB-DS as shown in Algorithm 5. Lines 1–3 are as for DB-DC and cover the cases where grounded semantics makes an informed decision. In order to decide skeptical acceptance for the remaining cases, we employ the following heuristic:

Algorithm 6 IB-SEARCH algorithm for verifying whether a given subset can be extended to an admissible set

```
Input:   $F = (A, R)$, $S \subseteq A$
Output:  TRUE iff there is admissible $S'$ with $S \subseteq S'$.
Global:  $n \in \mathbb{N} \cup \{\infty\}$
IB-SEARCH($F, S$)
 1: if $S$ is admissible then
 2:     return TRUE
 3: $n := n - 1$
 4: if $n \leq 0$ then
 5:     return FALSE
 6: for $b \in S_F^- \setminus S_F^+$ do
 7:     if $b_F^- \setminus (S_F^- \cup S_F^+) = \emptyset$ then
 8:         return FALSE
 9:     for $c \in b_F^- \setminus (S_F^- \cup S_F^+)$ do
10:         if IB-SEARCH($F, S \cup \{c\}$) then
11:             return TRUE
12: return FALSE
```

if the argument a is contained in an admissible set (which again must contain the grounded extension) and no attacker b of a is contained in an admissible set (that contains the grounded extension and is not attacked by it), then a is skeptically accepted wrt. $\sigma \in \{\mathsf{pr}, \mathsf{st}, \mathsf{sst}, \mathsf{stg}, \mathsf{id}\}$. Note that this heuristic actually captures a necessary (but not sufficient) condition for skeptical acceptance under preferred and ideal semantics. Lines 4–7 of Algorithm 5 implement this approach.

5.2 Iteration-bounded search

We present now an alternative implementation of a bounded search algorithm. Instead of bounding the search depth we now bound the number of calls to the search function.

The general iteration-bounded search algorithm IB-SEARCH is shown in Algorithm 6 and is structurally similar to the algorithm DB-SEARCH from above. This algorithm has two parameters $F = (A, R)$ and $S \subseteq A$ and accesses an additional parameter $n \in \mathbb{N} \cup \{\infty\}$ that is shared among multiple instances of IB-SEARCH. The algorithm is supposed to return TRUE if and only if there is an admissible set S' with $S \subseteq S'$. The parameter n gives a bound on the number of iterations that may impair the correctness of algorithm. As before, if the iteration bound is not ∞, it may happen that the algorithm does not find an admissible set, even if it exists.

The algorithm IB-SEARCH works on input F and a as follows. If S is admissible, we are already finished and terminate with TRUE (lines 1–2). We then decrement the global parameter n and check if we have reached the bound of the search (lines 3–5). Otherwise (lines 6–12), we proceed exactly as for DB-

Algorithm 7 IB-DC algorithm for credulous reasoning wrt. $\sigma \in \{\mathsf{co}, \mathsf{st}, \mathsf{sst}, \mathsf{stg}\}$

```
Input:   $F = (A, R)$, $a \in A$, $N \in \mathbb{N} \cup \infty$
Output:  TRUE iff $a$ is contained in a $\sigma$-extension.
IB-DC($F, a, N$)
1: $S = \mathsf{grounded}(F)$
2: if $a \in S$ then return TRUE
3: if $a \in S_F^+$ then return FALSE
4: set global $n = N$
5: return IB-SEARCH($F, S \cup \{a\}$)
```

Algorithm 8 IB-DS algorithm for skeptical reasoning wrt. $\sigma \in \{\mathsf{pr}, \mathsf{st}, \mathsf{sst}, \mathsf{stg}, \mathsf{id}\}$

```
Input:   $F = (A, R)$, $a \in A$, $N \in \mathbb{N} \cup \infty$
Output:  TRUE iff $a$ is contained in all $\sigma$-extensions.
IB-DS($F, a, N$)
1: $S = \mathsf{grounded}(F)$
2: if $a \in S$ then return TRUE
3: if $a \in S_F^+$ then return FALSE
4: set global $n = N/2$
5: if not IB-SEARCH($F, S \cup \{a\}$) then return FALSE
6: for $b \in a_F^- \setminus S_F^+$ do
7:     set global $n = N/(2 * |a_F^-|)$
8:     if IB-SEARCH($F, S \cup \{b\}$) then return FALSE
   return TRUE
```

SEARCH and consider defenders of undefended arguments. The design of the algorithm ensures that IB-SEARCH can be called only a maximum number of n times (it terminates earlier if an admissible set has been found).

We summarise the formal properties of IB-SEARCH(F, S) in the following result, which is given without proof but should be clear from the discussion above.

Proposition 2. *Let $F = (A, R)$, $S \subseteq A$, and $n \in \mathbb{N} \cup \{\infty\}$.*

1. *If IB-SEARCH(F, S) =* TRUE *then there is an admissible set S' with $S \subseteq S'$.*

2. *If IB-SEARCH(F, S) =* FALSE *and $n = \infty$ then there is no admissible set S' with $S \subseteq S'$.*

As with DB-SEARCH, the algorithm IB-SEARCH is an *incomplete algorithm.* If it returns TRUE, we can be sure that this is the correct answer. If the algorithm returns FALSE, the answer may be correct or not (only for $n = \infty$ it is guaranteed that the answer is correct).

We embed the algorithm IB-SEARCH in the algorithm IB-DC for solving problems in $\texttt{Prob}_C$ as shown in Algorithm 7, which is almost exactly the same as DB-DC (Algorithm 4). The main difference is line 4, where we set the global parameter n.

As for skeptical reasoning, we embed the algorithm IB-SEARCH in the algorithm IB-DS as shown in Algorithm 8, cf. also DB-DS (Algorithm 5). Here, we distribute the allocated number of iterations N over the different calls for the subproblems. In particular, for deciding whether the query argument a is contained in an admissible set we allot a maximum number of $N/2$ iterations (line 4). To check whether any of the attackers of a is contained in an admissible set, we allot a maximum number of $N/(2 * |a_F^-|)$ iterations each. Note that the maximum total number of iterations is indeed N.[6]

6 Experiments

Algorithms DB-DC, DB-DS, IB-DC, and IB-DS are each parametrised with a bound parameter. In a first experiment (Section 6.1) we experimentally test the behaviour of these algorithms for different values of n/N. In a second experiment (Section 6.2), we compare the performance of DB-DC, DB-DS, IB-DC, and IB-DS (with the best parameters found before) with GR-DC, GR-DS and other approaches from the literature.

For all experiments, we use the ICCMA'23 main data set[7], which contains 308–329 (depending on the problem and the semantics) *verified* instances, i. e., pairs of an argumentation framework and a query argument, where the true answer to the corresponding computational problems is known. Algorithms GR-DC and GR-DS have been implemented in the solver HARPER++[8] and algorithms DB-DC, DB-DS, IB-DC, and IB-DS have been implemented in the solver FARGO[9]. Raw results of all experiments and the used scripts can be found online.[10]

6.1 Parameter selection

Given the theoretical discussion in Section 5, it should be clear that for larger values of n/N the algorithms DB-SEARCH and IB-SEARCH are more likely to give a correct answer, but will also require more time to run. With the following experiments, we seek to find the optimal value for n/N, such that the algorithms give a maximal number of correct answers within a given time limit. For the latter, we use the same time limit as was used in the approximate track of ICCMA'23, namely 60 seconds. For reasons of simplicity, we only evaluate DB-DC and IB-DC on stable semantics and use the obtained optimal values for all semantics and the skeptical variants as well in the next section.

[6]This distribution of iterations over the sub-problems has been empirically determined and other distributions may be better for other data sets.
[7]http://argumentationcompetition.org/2023/benchmarks.html
[8]`https://github.com/aig-hagen/taas-harperpp`
[9]`https://github.com/aig-hagen/taas-fargo`
[10]`http://mthimm.de/misc/heuristic_solvers_2024_mt.zip`

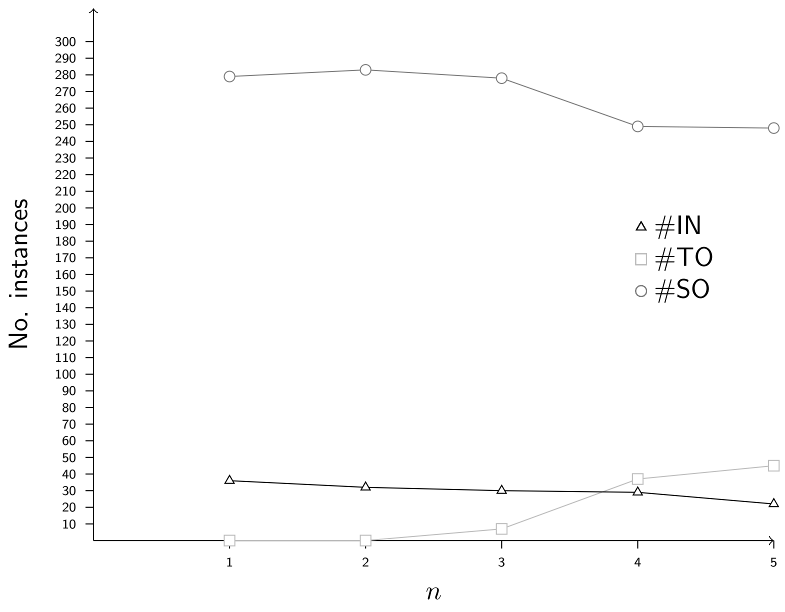

Figure 1: Relationship between the number of timeouts (#TO), the number of incorrectly solved instances (#IN), and the number of correctly solved instances within the time limit (#SO) for varying values of n when using algorithm DB-DC for stable semantics.

We first consider DB-DC for stable semantics. Figure 1 shows the relationship between the number of timeouts (#TO) and the number of incorrectly solved instances (#IN) for varying values of n. As expected, for increasing values of n, the number of timeouts increases while the number of incorrectly solved instances decreases. The figure also shows the number of correctly solved instances within the time limit (#SO), which has its maximum at $n = 2$, which will be used in the experiments in Section 6.2.

We now consider IB-DC for stable semantics. Note first that, in contrast to the search *depth*, the value of N here is much more dependent on the size of the argumentation framework. In particular, preliminary tests showed that using a constant parameter N for frameworks of all sizes gives quite poor results. We therefore parametrise the used value N for each call to IB-DC via $N = \hat{N}\sqrt{|A|}$ (where A is the set of arguments in the framework $F = (A, R)$ under consideration) and evaluated different values for $\hat{N}$. Correspondingly, Figure 2 shows the relationship between the number of timeouts (#TO), the number of incorrectly solved instances (#IN), and the number of correctly solved instances within the time limit (#SO) for varying values of $\hat{N}$. Here, #SO has its maximum value at $\hat{N} = 3000$, which will be used in the experiments in Section 6.2.

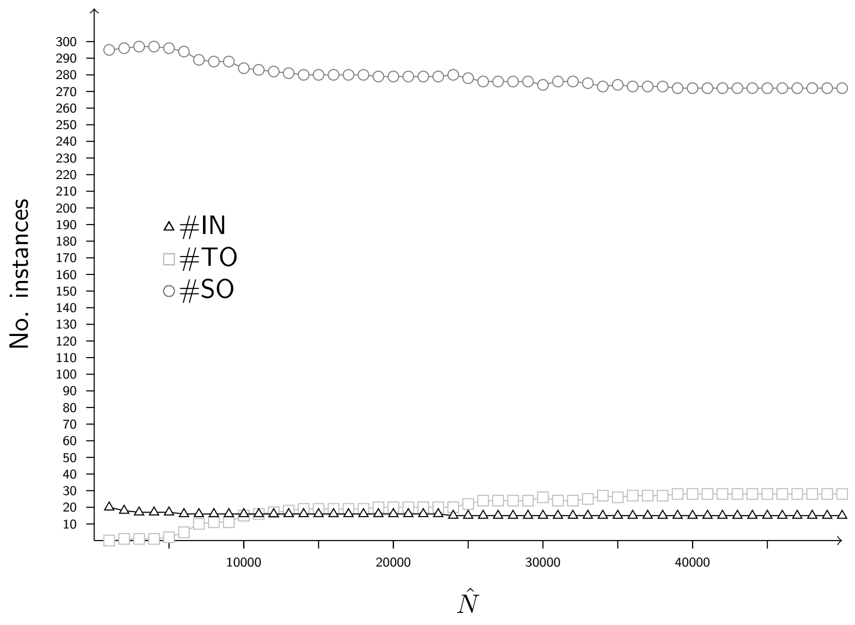

Figure 2: Relationship between the number of timeouts (#TO), the number of incorrectly solved instances (#IN), and the number of correctly solved instances within the time limit (#SO) for varying values of $\hat{N}$.

6.2 Performance evaluation

We now experimentally evaluate the performance of our approaches with existing systems, in terms of both *accuracy*, i. e., the number of correctly solved instances within the time limit (which is again set to 60 seconds), and *runtime performance*, i. e., the average runtime on all instances solved within the time limit.

We consider the following implementations of algorithms from the current work:

- Harper++: This solver implements the algorithms GR-DC and GR-DS from Section 4.
- Fargo2-DB-2: This solver implements the algorithms DB-DC and DB-DS from Section 5.1 with $n = 2$ (see previous section).
- Fargo2-IB-3000: This solver implements the algorithms IB-DC and IB-DS from Section 5.2 with $N = 3000\sqrt{|A|}$ for each $F = (A, R)$ (see previous section).

We also consider the earlier version of Fargo that participated in the approximation track of ICCMA'23 and only differs from the above version in the value used for the iteration bound:

- Fargo1-500: This solver implements the algorithms IB-DC and IB-DS from Section 5.2 with $N = 500|A|$ for each $F = (A, R)$.

In addition, we also consider all further systems that participated in the approximation track of ICCMA'23:

- ARIPOTER-Degrees: This solver works like Harper++ for instances where the query argument a is contained in or attacked by the grounded extension. In other cases, the solver answers True iff $|\{a\}^+| \geq k|\{a\}^-|$ where k is some meta parameter [12].

- ARIPOTER-HCAT: This solver works like Harper++ for instances where the query argument a is contained in or attacked by the grounded extension. In other cases, the solver answers True iff the *h-categorizer* [3] value of a is larger than a given threshold that is given as a meta parameter [13]

- AFGCNv2: This solver uses a graph convolutional neural network (with 4 layers and 128 neurons per layer) that has been trained on benchmarks from previous ICCMA competitions [28].

We refer to the set of the last three solvers also as the state-of-the-art solvers (sota-solvers).

Finally, we also consider the ICCMA'23 version[11] of μ-toksia [29], which is a state-of-the-art *complete* solver for the same abstract argumentation problems. Note that μ-toksia is expected to always provide the correct answer, but will more often time out under the stricter time limit of 60 seconds.

Tables 1–9 show the results of the experimental evaluation. Here, the value #SO refers to the number of instances correctly solved within the time limit, #TO to the number of instances with a timeout, #IN to the number of incorrectly solved instances within the time limit, #ERR to the number instances that resulted in some error (e. g., memory issues), CRT to the cumulative runtime (in milliseconds) on all correctly solved instances, and AVG to the average runtime (in milliseconds) over all instances that were solved within the time limit (not necessarily correct) by all solvers in that track. For each track, solvers are ordered by their ranking in that track, which is given by their values #SO and (in cases of ties wrt. #SO) CRT.

Let us first discuss the results for credulous reasoning (Tables 1–4). The first thing to note is that the first place for all four semantics is taken by one of our approaches. Moreover, for all semantics based on admissibility (co, st, and sst), the solver Fargo2-IB-3000 wins that track. Moreover, only that solver features a single timeout, all other of our solvers could solve all instances in time, in contrast to the sota-solvers, which all have 23 or more timeouts. This observation on the runtime performance is also reflected on the average runtime for instances solved within the time limit. Here, all our solvers are almost up to two orders of magnitude faster than the sota-solvers. As for accuracy, Fargo2-IB-3000 improves upon the runner-up from the sota-solvers (ARIPOTER-HCAT) for co,

[11]`https://bitbucket.org/andreasniskanen/mu-toksia/`

DC-co ($n = 316$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 197$)
1	Fargo2-IB-3000	290	1	25	0	204097.03	38.69
2	μ-toksia	290	26	0	0	821087.19	102.61
3	Fargo1-500	286	0	30	0	119927.58	40.87
4	Fargo2-DB-2	264	0	52	0	100807.85	37.67
5	Harper++	221	0	95	0	81106.80	26.24
6	ARIPOTER-HCAT	211	47	58	0	862004.68	3046.57
7	AFGCNv2	192	23	64	36	947482.26	3648.85
8	ARIPOTER-Degrees	181	35	100	0	633082.10	1658.33

Table 1: Results for DC-co on the ICCMA'23 data set.

DC-st ($n = 315$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 198$)
1	Fargo2-IB-3000	297	1	17	0	644064.12	38.51
2	Fargo1-500	293	0	22	0	225191.61	40.77
3	μ-toksia	291	24	0	0	708770.37	96.13
4	Fargo2-DB-2	283	0	32	0	186448.21	37.90
5	ARIPOTER-HCAT	219	46	50	0	1249174.73	3029.09
6	ARIPOTER-Degrees	196	35	84	0	910502.90	1645.83
7	AFGCNv2	192	23	64	36	1005081.15	3638.10
8	Harper++	187	0	128	0	52966.34	26.28

Table 2: Results for DC-st on the ICCMA'23 data set.

st, and sst by up to almost 30% more solved instances (297 solved instances for Fargo2-IB-3000 compared to 219 for ARIPOTER-HCAT on DC-st). The only exception is credulous reasoning with stage semantics, which is not surprising as stage extensions are not necessarily admissible and the heuristic used for Fargo is not generally plausible. For that track, the solver Harper++ outperforms all other solvers. It should also be noted that the complete solver μ-toksia behaves significantly better than one might expect. In particular, μ-toksia outperforms all sota-solvers and always scores second or third place. Our new approaches only marginally improve upon μ-toksia in terms of number of solved instances, but one can also observe a significant gap in the average runtime (for example 38.51ms for Fargo2-IB-3000 compared to 96.13ms for μ-toksia on DC-st).

Let us consider now the results for skeptical reasoning (Tables 5-9). We first note that the complete solver μ-toksia wins the track for DS-st, which is a remarkable result given the strict time constraints. For the remaining tracks, our approaches outperform all other solvers (and μ-toksia is still better than the sota-solvers, while having a significantly larger average runtime than our approaches). Observations regarding runtime are as above, but overall the field is a bit more condensed than for credulous reasoning, i. e., the difference in the number of correctly solved instances between the best and worst solver is 98 for DC-sst and only 60 for DS-sst.

In general, all results also show that the additional effort the Fargo solver invests for arguments that are neither contained in nor attacked by the grounded extension is worthwhile. For example, while Harper++ only solved 221 in-

DC-sst ($n = 302$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 198$)
1	Fargo2-IB-3000	286	0	16	0	233139.82	38.40
2	μ-toksia	286	16	0	0	691922.17	103.41
3	Fargo1-500	282	0	20	0	123814.60	41.45
4	Fargo2-DB-2	272	0	30	0	138702.85	37.79
5	ARIPOTER-HCAT	217	36	49	0	1018422.10	3089.13
6	Harper++	199	0	103	0	55187.21	26.71
7	AFGCNv2	194	23	62	23	1022674.68	3669.37
8	ARIPOTER-Degrees	188	26	88	0	800302.43	1683.19711

Table 3: Results for DC-sst for on the ICCMA'23 data set.

DC-stg ($n = 299$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 191$)
1	Harper++	269	0	30	0	75898.55	27.00
2	μ-toksia	265	34	0	0	680462.88	486.30
3	ARIPOTER-Degrees	240	34	25	0	1007898.58	1721.04
4	ARIPOTER-HCAT	230	45	24	0	1264308.16	3158.27
5	Fargo2-IB-3000	204	1	94	0	232963.75	38.91
6	Fargo1-500	200	0	99	0	112317.53	40.93
7	Fargo2-DB-2	190	0	109	0	126506.61	38.21
8	AFGCNv2	165	23	82	29	717816.26	3704.86

Table 4: Results for DC-stg on the ICCMA'23 data set.

stances correctly for DC-co (see Table 1), Fargo2-IB-3000 could solve 290 instances correctly. This is also true, albeit a bit less pronounced, for skeptical reasoning, e. g., Fargo2-DB-2 could solve 8 more instances than Harper++ for DS-sst (see Table 7). It can also be observed that the IB-SEARCH approach (Fargo2-IB-3000) performs better for credulous reasoning tasks, while DB-SEARCH (Fargo2-DB-2) performs better for skeptical reasoning with stable, semi-stable, and stage semantics and IB-SEARCH (Fargo2-IB-3000) performs better for skeptical reasoning with preferred and ideal semantics. However, the reasons for this is not so apparent and the differences are also mostly marginal.

7 Summary and conclusion

We presented heuristic algorithms for solving credulous and skeptical reasoning problems wrt. complete, stable, semi-stable, preferred, stage, and ideal semantics. Our algorithms were based on using grounded semantics as a proxy and using bounded search in two variants. Our experimental evaluation showed that our approaches outperform the state of the art in most cases.

A key insight of our work is that simple domain-dependent heuristics can be better suited than sophisticated but general approaches based on modern machine learning-techniques. Our evaluation showed that our approaches outperformed existing approaches, in particular those based on neural network technology, both in terms of accuracy and runtime. Moreover, this evaluation did not even take into account the resources, in terms of time and energy, required

DS-pr ($n = 306$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 188$)
1	FARGO1-500	305	0	1	0	200407.69	39.81
2	FARGO2-IB-3000	304	1	1	0	417060.15	39.19
3	HARPER++	303	0	3	0	90675.86	25.79
4	FARGO2-DB-2	292	0	14	0	194895.14	36.87
5	μ-toksia	272	34	0	0	782212.95	495.60
6	ARIPOTER-Degrees	268	35	3	0	1236232.01	1644.31
7	ARIPOTER-HCAT	257	46	3	0	1617031.39	3069.07
8	AFGCNv2	238	23	9	36	1214603.06	3301.24

Table 5: Results for DS-pr on the ICCMA'23 data set.

DS-st ($n = 307$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 189$)
1	μ-toksia	276	31	0	0	779818.32	154.72
2	FARGO2-DB-2	204	0	103	0	164862.70	37.28
3	FARGO1-500	199	0	108	0	111863.40	39.87
4	FARGO2-IB-3000	198	1	108	0	147113.30	39.22
5	HARPER++	196	0	111	0	59889.24	26.16
6	ARIPOTER-Degrees	180	35	92	0	961229.40	1637.07
7	ARIPOTER-HCAT	167	46	94	0	1133795.09	3053.44
8	AFGCNv2	158	23	90	36	774614.10	3310.42

Table 6: Results for DS-st on the ICCMA'23 data set.

for training such approaches. So while using modern neural network technology may be attractive due to its success, its applicability to a new problem has to be thoroughly tested first. Our work also shows that many computational hard problems in abstract argumentation can be solved fast and accurately in many cases. So using our heuristic approaches to guide complete solvers, may be a worthwhile avenue for future work, cf. [20]. Another avenue for future work is to apply our ideas to other formalisms and problems. The general paradigm of our approach is that by using the right search methodology, a positive answer to a problem can often be found quickly and if the effort (e. g., the search depth) required to find a solution is too large, then we can often safely return a negative answer. Problems in other formalisms could be heuristically solved in a similar manner. For example, problems in *Answer Set Programming* [19] are formalised as logic programs and usually follow a guess-and-check modelling approach, i. e., the general structure of solutions (answer sets) are first defined and constraints define valid solutions. Solvers such as clingo [18] then, basically, guess solutions iteratively and verify their correctness. By further relying on domain-dependent heuristics [9] one can apply our approach in this context and avoid exhaustive search. Answers to queries such as asking whether a certain atom is contained in some answer set or all answer sets (so variants of our DC and DS queries) could then be answered similarly in a probably reliable manner.

Acknowledgements. The research reported here was partially supported by the Deutsche Forschungsgemeinschaft (grant 550735820).

DS-sst ($n = 298$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 189$)
1	Fargo2-DB-2	293	0	5	0	195941.83	36.97
2	Fargo1-500	288	0	10	0	156697.30	39.54
3	Fargo2-IB-3000	288	0	10	0	345588.51	38.93
4	Harper++	285	0	13	0	73404.50	26.00
5	μ-toksia	266	32	0	0	525384.43	403.52
6	ARIPOTER-Degrees	255	31	12	0	1037413.71	1657.98
7	ARIPOTER-HCAT	245	42	11	0	1344020.22	3091.45
8	AFGCNv2	233	23	14	28	1180091.90	3311.43

Table 7: Results for DS-sst on the ICCMA'23 data set.

DS-stg ($n = 301$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 188$)
1	Fargo2-DB-2	296	0	5	0	217205.34	36.96
2	Fargo1-500	291	0	10	0	186996.18	39.70
3	Fargo2-IB-3000	290	1	10	0	385304.64	39.34
4	Harper++	288	0	13	0	84181.83	26.14
5	μ-toksia	265	36	0	0	703607.28	495.57
6	ARIPOTER-Degrees	255	34	12	0	1058817.06	1658.93
7	ARIPOTER-HCAT	245	45	11	0	1397425.50	3094.37
8	AFGCNv2	232	23	14	32	1218799.84	3310.96

Table 8: Results for DS-stg on the ICCMA'23 data set.

References

[1] Leila Amgoud and Jonathan Ben-Naim. Ranking-based semantics for argumentation frameworks. In Weiru Liu, V S Subrahmanian, and Jef Wijsen, editors, *Proceedings of the 7th International Conference on Scalable Uncertainty Management (SUM 2013)*, volume 8078 of *Lecture Notes In Artificial Intelligence*, Washington, DC, USA, September 2013.

[2] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. Abstract argumentation frameworks and their semantics. In Pietro Baroni, Dov Gabbay, Massimiliano Giacomin, and Leendert van der Torre, editors, *Handbook of Formal Argumentation*, volume 1, pages 159–236. College Publications, 2018.

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

[4] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[5] Stefano Bistarelli and Francesco Santini. Weighted argumentation. In Dov Gabbay, Massimiliano Giacomin, Guillermo R. Simari, and Matthias Thimm, editors, *Handbook of Formal Argumentation*, volume 2, chapter 6. College Publications, August 2021.

DS-id ($n = 312$)							
Rank	Solver	#SO	#TO	#IN	#ERR	CRT	AVG ($n' = 195$)
1	Fargo1-500	311	0	1	0	183758.10	39.31
2	Fargo2-IB-3000	311	0	1	0	417544.93	38.42
3	Harper++	308	0	4	0	85941.22	25.80
4	Fargo2-DB-2	298	0	14	0	182693.28	36.60
5	μ-toksia	281	31	0	0	801138.52	396.08
6	ARIPOTER-Degrees	274	34	4	0	1260207.63	1626.98
7	AFGCNv2	263	10	4	35	1240996.96	2819.14
8	ARIPOTER-HCAT	262	46	4	0	1578006.87	3004.74

Table 9: Results for DS-id on the ICCMA'23 data set.

[6] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. Combinatorial optimization and reasoning with graph neural networks. *J. Mach. Learn. Res.*, 24:130:1–130:61, 2023.

[7] Federico Cerutti, Sarah A. Gaggl, Matthias Thimm, and Johannes P. Wallner. Foundations of implementations for formal argumentation. In Pietro Baroni, Dov Gabbay, Massimiliano Giacomin, and Leendert van der Torre, editors, *Handbook of Formal Argumentation*, chapter 15. College Publications, February 2018. Also appears in IfCoLog Journal of Logics and their Applications 4(8):2623–2706, October 2017.

[8] Federico Cerutti, Matthias Thimm, and Mauro Vallati. An experimental analysis on the similarity of argumentation semantics. *Argument & Computation*, 11(3):269–304, October 2020.

[9] Richard Comploi-Taupe, Gerhard Friedrich, Konstantin Schekotihin, and Antonius Weinzierl. Domain-specific heuristics in answer set programming: A declarative non-monotonic approach. *J. Artif. Intell. Res.*, 76:59–114, 2023.

[10] Dennis Craandijk and Floris Bex. Deep learning for abstract argumentation semantics. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1667–1673. ijcai.org, 2020.

[11] Jérôme Delobelle, Jean-Guy Mailly, and Julien Rossit. Revisiting approximate reasoning based on grounded semantics. In Zied Bouraoui and Srdjan Vesic, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty - 17th European Conference, ECSQARU 2023, Arras, France, September 19-22, 2023, Proceedings*, volume 14294 of *Lecture Notes in Computer Science*, pages 71–83. Springer, 2023.

[12] Jérôme Delobelle, Jean-Guy Mailly, and Julien Rossit. ARIPOTER-Degrees: ARgumentatIon apPrOximaTE Reasoning using In/Out Degrees of Arguments. In *Fifth International Competition on Computational Models of Argumentation (ICCMA'23)*, 2023.

[13] Jérôme Delobelle, Jean-Guy Mailly, and Julien Rossit. ARIPOTER-HCAT: ARgumentatIon apPrOximaTE Reasoning using the h-Categorizer semantics. In *Fifth International Competition on Computational Models of Argumentation (ICCMA'23)*, 2023.

[14] Phan Minh Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2):321–358, 1995.

[15] Phan Minh Dung, Paolo Mancarella, and Francesca Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10):642–674, 2007.

[16] P. E. Dunne, A. Hunter, P. McBurney, S. Parsons, and M. Wooldridge. Weighted argument systems: Basic definitions, algorithms, and complexity results. *Artificial Intelligence*, 175(2):457–486, 2011.

[17] Wolfgang Dvořák and Paul E. Dunne. Computational problems in formal argumentation and their complexity. In Pietro Baroni, Dov Gabbay, Massimiliano Giacomin, and Leendert van der Torre, editors, *Handbook of Formal Argumentation*, chapter 14. College Publications, February 2018.

[18] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[19] Michael Gelfond and Nicola Leone. Logic programming and knowledge representation - The A-Prolog perspective. *Artificial Intelligence*, 138(1-2):3–38, June 2002.

[20] Sandra Hoffmann, Isabelle Kuhlmann, and Matthias Thimm. Enhancing abstract argumentation solvers with machine learning-guided heuristics: A feasibility study. In *Proceedings of the 1st International Conference on Recent Advances in Robust Argumentation Machines (RATIO'24)*, June 2024.

[21] Anthony Hunter, Sylwia Polberg, Nico Potyka, Tjitze Rienstra, and Matthias Thimm. Probabilistic argumentation: A survey. In Dov Gabbay, Massimiliano Giacomin, Guillermo R. Simari, and Matthias Thimm, editors, *Handbook of Formal Argumentation*, volume 2, chapter 7. College Publications, August 2021.

[22] Matti Järvisalo, Tuomo Lehtonen, and Andreas Niskanen, editors. *Solver and Benchmark Descriptions of ICCMA 2023: 5th International Competition on Computational Models of Argumentation*, Department of Computer Science Series of Publications B. University of Helsinki, 2023.

[23] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[24] Isabelle Kuhlmann and Matthias Thimm. Using graph convolutional networks for approximate reasoning with abstract argumentation frameworks: A feasibility study. In Nahla Ben Amor, Benjamin Quost, and Martin Theobald, editors, *Proceedings of the 13th International Conference on Scalable Uncertainty Management (SUM'19)*, volume 11940, pages 24–37. Springer International Publishing, December 2019.

[25] Isabelle Kuhlmann, Thorsten Wujek, and Matthias Thimm. On the impact of data selection when applying machine learning in abstract argumentation. In *Proceedings of the 9th International Conference on Computational Models of Argument (COMMA'22)*, September 2022.

[26] H. Li, N. Oren, and T. J. Norman. Probabilistic argumentation frameworks. In *Proceedings of the First International Workshop on the Theory and Applications of Formal Argumentation (TAFA'11)*, 2011.

[27] Lars Malmqvist. *Approximate Solutions to Abstract Argumentation Problems Using Graph Neural Networks*. PhD thesis, University of York, 2022.

[28] Lars Malmqvist. AFGCN v2: A GCN-based Approximate Solver. In *Fifth International Competition on Computational Models of Argumentation (ICCMA'23)*, 2023.

[29] Andreas Niskanen and Matti Järvisalo. mu-toksia: An efficient abstract argumentation reasoner. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)*, 2020.

[30] Samer Nofal, Katie Atkinson, and Paul E. Dunne. Computing grounded extensions of abstract argumentation frameworks. *The Computer Journal*, 64(1):54–63, 2021.

[31] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[32] Kenneth Skiba, Tjitze Rienstra, Matthias Thimm, Jesse Heyninck, and Gabriele Kern-Isberner. Ranking extensions in abstract argumentation. In Zhi-Hua Zhou, editor, *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*, pages 2047–2053, August 2021.

[33] Matthias Thimm. Fargo-limited v1.1.1. In *Fifth International Competition on Computational Models of Argumentation (ICCMA'23)*, September 2023.

[34] Matthias Thimm. Harper++ v1.1.1. In *Fifth International Competition on Computational Models of Argumentation (ICCMA'23)*, September 2023.

[35] Matthias Thimm. Optimisation and approximation in abstract argumentation: The case of stable semantics. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI'24)*, August 2024.

[36] Matthias Thimm and Serena Villata. The first international competition on computational models of argumentation: Results and analysis. *Artificial Intelligence*, 252:267–294, August 2017.

[37] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[38] Zhun Yang, Adam Ishay, and Joohyung Lee. Neurasp: Embracing neural networks into answer set programming. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 1755–1762. ijcai.org, 2020.