Strong Refinements for Hard Problems in Argumentation Dynamics

Andreas Niskanen Matti Järvisalo

HIIT, Department of Computer Science, University of Helsinki, Finland

ECAI 2020

Motivation

Argumentation in AI

- Active and vibrant area of modern AI research
- Central KR formalism for reasoning in abstract argumentation: *argumentation frameworks (AFs)* [Dung, 1995]
- Recent interest in dynamic aspects of AFs [Doutre and Mailly, 2018]

Computational Problems Arising from Dynamics of AFs

Several variants and AF semantics give rise to optimization problems with complexity beyond NP [Wallner et al., 2017, Niskanen et al., 2016, Niskanen et al., 2019]

Contributions

What?

Improve the scalability of state-of-the-art practical algorithms for optimization problems arising from AF dynamics

- Current approaches based on declaratively employing *maximum satisfiability (MaxSAT)* solvers [Wallner et al., 2017, Niskanen et al., 2019]
- Focus on second-level complete variants of problems, algorithms based on *counterexample-guided abstraction refinement (CEGAR)*

How?

Design strong refinements using recent results on the persistence of extensions under adding and removing attacks in an AF [Rienstra et al., 2015]

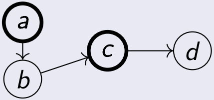
- Allowing for significantly reducing the number of CEGAR iterations by ruling out larger sets of solution candidates
- **Noticeable empirical runtime improvements** and scalability to larger instance sizes

Abstract Argumentation Frameworks

Argumentation Framework (AF)

A directed graph $F = (A, R)$, where

- A is the set of **arguments**
- $R \subseteq A \times A$ is the **attack relation**
    - $a \rightarrow b$ means argument a attacks argument b



Semantics of AFs

Define sets of jointly accepted arguments called **extensions**

- Required to be **conflict-free** (independent sets)
- Additional desired properties
    - self-defense: **admissible** sets
    - self-defense + subset-maximality: **preferred** extensions

Computational Problems

Focus: second-level complete variants of

- **extension enforcement** [Wallner et al., 2017]
- status enforcement [Niskanen et al., 2016]
- argumentation framework synthesis [Niskanen et al., 2019]

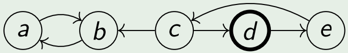
Improving the scalability of state-of-the-art MaxSAT-based CEGAR algorithms by designing and applying **strong refinements**

Computational Problems

Strict Extension Enforcement under Preferred Semantics

- **Given:** an AF $F = (A, R)$, set $T \subseteq A$
- **Task:** find an AF $F' = (A, R')$ where T is a preferred extension while minimizing the number of changes between R and R'
- **Complexity:** Σ_2^P-complete [Wallner et al., 2017]

Example

$F =$ 

$T = \{d\}$

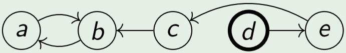
Currently: unique preferred extension is $\{a\}$ ✗

Computational Problems

Strict Extension Enforcement under Preferred Semantics

- **Given:** an AF $F = (A, R)$, set $T \subseteq A$
- **Task:** find an AF $F' = (A, R')$ where T is a preferred extension while minimizing the number of changes between R and R'
- **Complexity:** Σ_2^P-complete [Wallner et al., 2017]

Example

$F =$  $T = \{d\}$

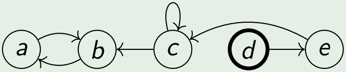
Remove attack $c \to d$: $\{d\}$ is admissible but not preferred ✗

Computational Problems

Strict Extension Enforcement under Preferred Semantics

- **Given:** an AF $F = (A, R)$, set $T \subseteq A$
- **Task:** find an AF $F' = (A, R')$ where T is a preferred extension while minimizing the number of changes between R and R'
- **Complexity:** Σ_2^P-complete [Wallner et al., 2017]

Example

$F =$ 

$T = \{d\}$

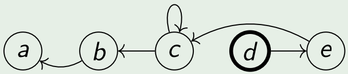
Add attack $c \to c$: $\{d\}$ is complete but not preferred ✗

Computational Problems

Strict Extension Enforcement under Preferred Semantics

- **Given:** an AF $F = (A, R)$, set $T \subseteq A$
- **Task:** find an AF $F' = (A, R')$ where T is a preferred extension while minimizing the number of changes between R and R'
- **Complexity:** Σ_2^{P}-complete [Wallner et al., 2017]

Example

$F =$ 

$T = \{d\}$

Remove attack $a \to b$: $\{d\}$ is preferred ✓

MaxSAT-based CEGAR Algorithms

CEGAR for Strict Extension Enforcement under Preferred Semantics

Given: $F = (A, R)$, $T \subseteq A$, changes to the original attack structure R are encoded using variables $r_{a,b}$ for each $a, b \in A$. Iteratively:

1. **Abstraction**: using a MaxSAT solver call, strictly enforce T to be a complete extension
   - obtain candidate solution AF $F' = (A, R')$ from the optimal truth assignment on $r_{a,b}$ variables
2. **Counterexample**: using a SAT solver call, check whether there is an admissible set in F' that is a superset of T
   - if none exists, T is preferred in F' which is an optimal solution
3. **Refinement**: exclude the candidate attack structure via the clause

$$\bigvee_{(a,b)\in(A\times A)\setminus R'} r_{a,b} \vee \bigvee_{(a,b)\in R'} \neg r_{a,b}$$

Strong Refinements

Idea: instead of excluding only the current solution AF, use the counterexample to rule out more non-solution AFs
Observation: since the counterexample is an extension that invalidates the solution, all candidate solutions with the extension are non-solutions
Goal: characterize changes to the attack structure that do not affect the existence of the counterexample extension for a shorter refinement clause

Persistence of Extensions

Given an AF $F = (A, R)$ and $E \in \sigma(F)$ under $\sigma \in \{adm, stb\}$, **if we**

- **add an attack** (a, b) to F with the source a already attacked by E, or the target b outside E,
- **remove an attack** (a, b) from F where the source a is not in E, or the target b is not attacked by E,

E **is still an extension** in the AF. [Rienstra et al., 2015, Niskanen et al., 2020]

Strong Refinements

Recall the refinement clause for a non-solution AF $F' = (A, R')$:

$$\underbrace{\bigvee_{(a,b)\in(A\times A)\setminus R'} r_{a,b}}_{\text{add an attack}} \vee \underbrace{\bigvee_{(a,b)\in R'} \neg r_{a,b}}_{\text{remove an attack}}$$

Using result on persistence of extensions, obtain a **shorter clause** by excluding literals which have no effect on counterexample extension

- prune search space of potential attack structures more efficiently

Implementation and Benchmark Setup

Pakota and AFSynth

- State-of-the-art implementations for extension and status enforcement and AF synthesis reimplemented via PySAT [Ignatiev et al., 2018]
- Available in open source via `https://bitbucket.org/andreasniskanen/{pakota|afsynth}`

Benchmark Setup

- Per-instance 1800-second time limit and 64-GB memory limit
- Benchmark instances:
  - $>$ 1000 extension enforcement instances and status enforcement instances generated based on ICCMA'19 AFs
  - 400 AF synthesis instances generated using a random model

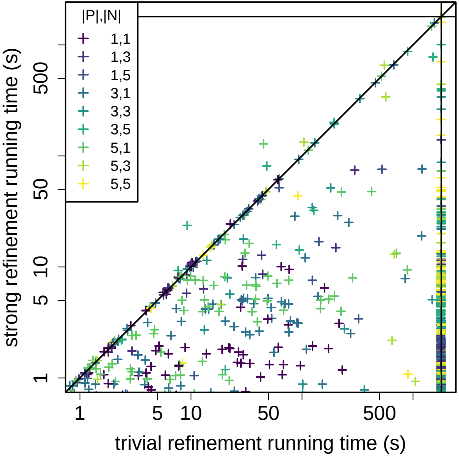
Experimental Evaluation

Mean running times (with timeouts as 1800 s) and number of timeouts (out of 221 instances): strict extension enforcement under preferred

	Refinement type			
$\|T\|/\|A\|$	**trivial**		**strong**	
0.025	1023.32	(121)	**798.11**	**(94)**
0.05	830.51	(95)	**666.93**	**(78)**
0.075	748.53	(87)	**671.96**	**(79)**
0.1	717.16	(82)	**676.62**	**(81)**
0.2	463.21	(54)	**433.36**	**(51)**
0.3	325.47	(38)	**301.14**	**(34)**

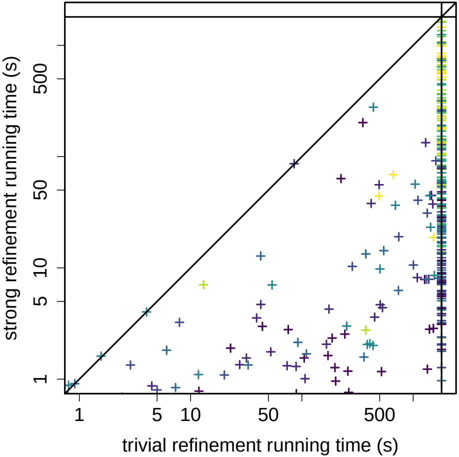
Experimental Evaluation

Trivial vs. strong refinement:
Credulous status enforcement under admissible



Experimental Evaluation

Trivial vs. strong refinement:
AF synthesis under preferred



Conclusions

Paper Summary

- **Strong refinements** for second-level MaxSAT-based CEGAR algorithms for problems arising from AF dynamics
  - Applicable to extension and status enforcement, AF synthesis
  - Based on recent theoretical results on the persistence of an extension under changes to the attack structure
- **Empirical evaluation:** our approach significantly scales up the current state-of-the-art approaches to the computational problems

Future Outlook

Strong refinements for other second-level hard problems over AFs?

- extension enforcement under semi-stable semantics? [Wallner et al., 2017]

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

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

Ignatiev, A., Morgado, A., and Marques-Silva, J. (2018).
Pysat: A Python toolkit for prototyping with SAT oracles.
In *Proc. SAT*, volume 10929 of *LNCS*, pages 428–437. Springer.

Niskanen, A., Neugebauer, D., Järvisalo, M., and Rothe, J. (2020).
Deciding acceptance in incomplete argumentation frameworks.
In *Proc. AAAI*, pages 2942–2949. AAAI Press.

Niskanen, A., Wallner, J. P., and Järvisalo, M. (2016).
Optimal status enforcement in abstract argumentation.
In *Proc. IJCAI*, pages 1216–1222. IJCAI/AAAI Press.

Niskanen, A., Wallner, J. P., and Järvisalo, M. (2019).
Synthesizing argumentation frameworks from examples.
J. Artif. Intell. Res., 66:503–554.

Rienstra, T., Sakama, C., and van der Torre, L. W. N. (2015).
Persistence and monotony properties of argumentation semantics.
In *Proc. TAFA*, volume 9524 of *LNCS*, pages 211–225. Springer.

Wallner, J. P., Niskanen, A., and Järvisalo, M. (2017).
Complexity results and algorithms for extension enforcement in abstract argumentation.
J. Artif. Intell. Res., 60:1–40.