Efficient Computation of Extensions for Dynamic Abstract Argumentation Frameworks: An Incremental Approach

Gianvincenzo Alfano, Sergio Greco, **Francesco Parisi**

{g.alfano, greco, fparisi}@dimes.unical.it
Department of Informatics, Modeling, Electronics and System Engineering
University of Calabria
Italy

26th International Joint Conference on Artificial Intelligence

August 19-25, 2017

Melbourne, Australia

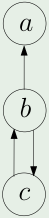
Argumentation in AI

- A general way for representing arguments and relationships (rebuttals) between them
- It allows representing dialogues, making decisions, and handling inconsistency and uncertainty

Abstract Argumentation Framework (AF) [Dung 1995]: arguments are abstract entities (no attention is paid to their internal structure) that may attack and/or be attacked by other arguments

Example (a simple AF)

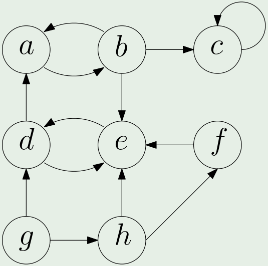
a =	Our friends will have great fun at our party on Saturday
b =	Saturday will rain (according to the weather forecasting service 1)
c =	Saturday will be sunny (according to the weather forecasting service 2)



Argumentation Semantics

- Several semantics have been proposed to identify “reasonable” sets of arguments, called *extensions*

Example (AF $\mathcal{A}_0$)



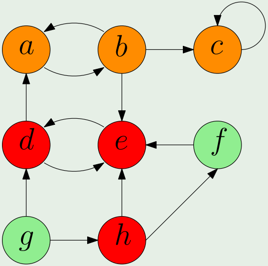
Semantic $\mathcal{S}$	Set of extensions $\mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$
complete ($\texttt{co}$)	$\{\{f, g\}, \{a, f, g\}, \{b, f, g\}\}$
preferred ($\texttt{pr}$)	$\{\{a, f, g\}, \{b, f, g\}\}$
stable ($\texttt{st}$)	$\{\{b, f, g\}\}$
grounded ($\texttt{gr}$)	$\{\{f, g\}\}$

- Argumentation semantics can be also defined in terms of *labelling*
- Function $L : A \to \{\text{IN}, \text{OUT}, \text{UN}\}$ assigns a label (accepted, rejected, undecided) to each argument

Argumentation Semantics

- Several semantics have been proposed to identify “reasonable” sets of arguments, called *extensions*

Example (AF $\mathcal{A}_0$)



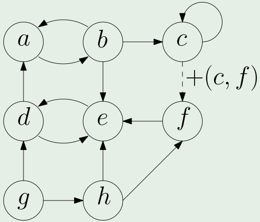
Semantic $\mathcal{S}$	Set of extensions $\mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$
complete ($\texttt{co}$)	$\{\{f,g\},\{a,f,g\},\{b,f,g\}\}$
preferred ($\texttt{pr}$)	$\{\{a,f,g\},\{b,f,g\}\}$
stable ($\texttt{st}$)	$\{\{b,f,g\}\}$
grounded ($\texttt{gr}$)	**{ {f,g} }**

- Argumentation semantics can be also defined in terms of *labelling*
- Function $L : A \rightarrow \{\text{IN}, \text{OUT}, \text{UN}\}$ assigns a label (accepted, rejected, undecided) to each argument

Dynamic Abstract Argumentation Frameworks

- Most argumentation frameworks are dynamic systems, which are often updated by adding/removing arguments/attacks.
- For each semantics, extensions/labellings change if we update the initial AF by adding/removing arguments/attacks

Example (Updated AF $\mathcal{A} = +(c, f)(\mathcal{A}_0)$)



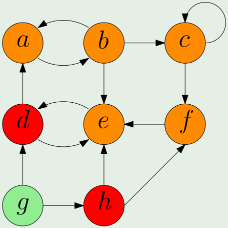
$\mathcal{S}$	$\mathcal{E}_\mathcal{S}(\mathcal{A}_0)$	$\mathcal{E}_\mathcal{S}(\mathcal{A}))$
co	$\{\{f,g\},\{a,f,g\},\{b,f,g\}\}$	?
pr	$\{\{a,f,g\},\{b,f,g\}\}$	?
st	$\{\{b,f,g\}\}$	?
gr	$\{\{f,g\}\}$	?

- Should we recompute the semantics of updated AFs from scratch?

Dynamic Abstract Argumentation Frameworks

- Most argumentation frameworks are dynamic systems, which are often updated by adding/removing arguments/attacks.
- For each semantics, extensions/labellings change if we update the initial AF by adding/removing arguments/attacks

Example (Updated AF $\mathcal{A} = +(c, f)(\mathcal{A}_0)$)



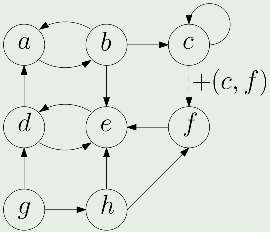
$\mathcal{S}$	$\mathcal{E}_\mathcal{S}(\mathcal{A}_0)$	$\mathcal{E}_\mathcal{S}(\mathcal{A}))$
co	$\{\{f, g\}, \{a, f, g\}, \{b, f, g\}\}$	$\{\{g\}, \{a, g\}, \{b, f, g\}\}$
pr	$\{\{a, f, g\}, \{b, f, g\}\}$	$\{\{a, g\}, \{b, f, g\}\}$
st	$\{\{b, f, g\}\}$	$\{\{b, f, g\}\ \}$
gr	$\{\{f, g\}\}$	$\{\ \{\mathbf{g}\}\}$

- **Should we recompute the semantics of updated AFs from scratch?**

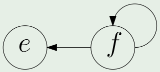
Reduced AF

- We show that for four well-known semantics (i.e., *grounded*, *complete*, *preferred*, and *stable*) an extension of the updated AF can be efficiently computed by looking only at a small part of the AF, called the *Reduced* AF, which is “influenced by” the update operation

Example (From the updated AF to the Reduced AF)



$\Longrightarrow$ Reduced AF:



- Once computed an extension for the reduced AF, it can be combined with the initial extension of the given AF to get an extension of the updated AF

Incremental Algorithm

1. We formally define the *Reduced AF*
   - Sub-AF consisting of the arguments whose status could change after an update
   - It depends on both the update and the initial extension E_0 (and thus the semantics)
2. We present an incremental algorithm for recomputing an extension of an updated AF for the *grounded*, *complete*, *preferred*, and *stable* semantics
   - It calls a non-incremental solver to compute an extension of the reduced AF
   - It obtains the final extension by merging a portion of the initial extension with that computed for the reduced AF.
3. A thorough experimental analysis showing the effectiveness of our approach for all the four semantics
   - Our technique outperforms the computation from scratch of the best solvers by two orders of magnitude

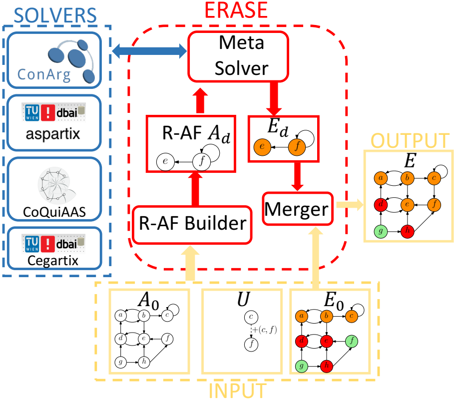
Outline

Overview of the approach

Given an initial AF $\mathcal{A}_0$, an extension E_0, and an update $u = \pm(a, b)$

Three main steps/modules:

1) Identify a sub-AF $\mathcal{A}_d = \langle A_d, \Sigma_d \rangle$, called *reduced* AF (R-AF) on the basis of the updates in U and additional information provided by the initial extension E_0
2) Compute an $\mathcal{S}$-extension E_d of the reduced AF $\mathcal{A}_d$ by using an external (non-incremental) solver
3) Merge E_d with the portion $(E_0 \setminus A_d)$ of the initial extension that does not change



Architecture of ERASE, our system for Efficiently Recomputing Argumentation SEmantics.

Irrelevant updates (1/2)

- Updates preserving a given initial extension/labelling
- Cases for which E_0 is still an extension of the updated AF after a *positive* update

update $+(a,b)$		$L_0(b)$ IN	$L_0(b)$ UN	$L_0(b)$ OUT
$L_0(a)$	IN			co, pr, st, gr
	UN		co, gr	co, pr, gr
	OUT	co, **pr**, st	co, gr	co, pr, st, gr

Example (For the update $+(c,f)$ the initial preferred extension $E_0 = \{b, f, g\}$ is preserved, as $L_0(c) = \text{OUT}$ and $L_0(f) = \text{IN}$)

initial labelling: updated labelling:

Irrelevant updates (2/2)

- Similar result for *negative* updates
- Cases for which E_0 is still an extension of the updated AF after a *negative* update

update $-(a, b)$		$L_0(b)$ IN	$L_0(b)$ UN	$L_0(b)$ OUT
$L_0(a)$	IN	NA	NA	
	UN	NA		co, pr, gr
	OUT	co, pr, st, gr	co, pr, gr	co, pr, st, gr

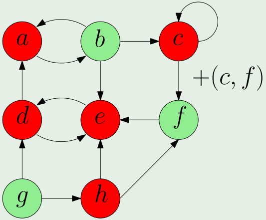
- In these cases we do not need to recompute the semantics of the updated AF: just return the initial extension

Influenced set: Intuition

- $\mathcal{I}(u, \mathcal{A}_0, E_0))$ denotes the *influenced set* of $u = \pm(a, b)$ w.r.t. $\mathcal{A}_0$ and E_0

1) $\mathcal{I}(u, \mathcal{A}_0, E_0) = \emptyset$ if u is irrelevant w.r.t. E_0 and the considered semantics.
2) The status of an argument can change only if it is reachable from b: $\mathcal{I}(u, \mathcal{A}_0, E_0) \subseteq Reach_{\mathcal{A}}(b)$
3) If argument z is not reachable from b and $z \in E_0$, then also the status of the arguments attacked by z cannot change: their status remain OUT

Example (Set of arguments influenced by an update operation)



Update $+(c, f)$ is irrelevant w.r.t. the preferred extension $E_0 = \{b, f, g\}$

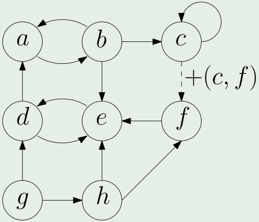
$\Rightarrow \mathcal{I}(+(c, f), \mathcal{A}_0, \{b, f, g\}) = \emptyset$

Influenced set: Intuition

- $\mathcal{I}(u, \mathcal{A}_0, E_0))$ denotes the *influenced set* of $u = \pm(a, b)$ w.r.t. $\mathcal{A}_0$ and E_0

1) $\mathcal{I}(u, \mathcal{A}_0, E_0) = \emptyset$ if u is irrelevant w.r.t. E_0 and the considered semantics.
2) The status of an argument can change only if it is reachable from b: $\mathcal{I}(u, \mathcal{A}_0, E_0) \subseteq Reach_{\mathcal{A}}(b)$
3) If argument z is not reachable from b and $z \in E_0$, then also the status of the arguments attacked by z cannot change: their status remain OUT

Example (Set of arguments influenced by an update operation)



$$\mathcal{I}(+(c, f), \mathcal{A}_0, E_0) \subseteq Reach_{\mathcal{A}}(f) = \{e, d, a, b, c\}$$

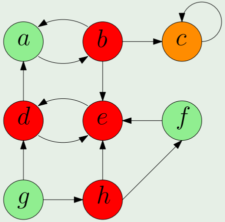
$$\Rightarrow g, h \notin \mathcal{I}(+(c, f), \mathcal{A}_0, E_0)$$

Influenced set: Intuition

- $\mathcal{I}(u, \mathcal{A}_0, E_0))$ denotes the *influenced set* of $u = \pm(a, b)$ w.r.t. $\mathcal{A}_0$ and E_0

1) $\mathcal{I}(u, \mathcal{A}_0, E_0) = \emptyset$ if u is irrelevant w.r.t. E_0 and the considered semantics.
2) The status of an argument can change only if it is reachable from b: $\mathcal{I}(u, \mathcal{A}_0, E_0) \subseteq Reach_{\mathcal{A}}(b)$
3) If argument z is not reachable from b and $z \in E_0$, then also the status of the arguments attacked by z cannot change: their status remain OUT

Example (Set of arguments influenced by an update operation)



$d \notin \mathcal{I}(+(d, f), \mathcal{A}_0, E_0)$ since it is attacked by $g \in E_0$ and g is not reachable from f.

Thus the arguments that can be reached only using d cannot belong to $\mathcal{I}(+(c, f), \mathcal{A}_0, E_0)$.

$\Rightarrow$ **The influenced set is** $\mathcal{I}(+(c, f), \mathcal{A}_0, E_0) = \{f, e\}$

Influenced set: Definition

- $\mathcal{I}(\pm(a,b),\mathcal{A}_0,E_0)$ is the set of arguments that can be reached from b without using any intermediate argument y whose status is known to be OUT because it is determined by an argument $z \in E_0$ which is not reachable from b

Definition (Influenced set)

Let $\mathcal{A} = \langle A, \Sigma\rangle$ be an AF, $u = \pm(a,b)$ an update, E an extension of $\mathcal{A}$ under a given semantics $\mathcal{S}$, and let

- $\mathcal{I}_0(u,\mathcal{A},E) = \begin{cases} \emptyset & \text{if } u \text{ is irrelevant w.r.t. } E \text{ and } \mathcal{S} \text{ or} \\ & \exists (z,b) \in \Sigma \text{ s.t. } z \in E \wedge z \notin Reach_{\mathcal{A}}(b); \\ \{b\} & \text{otherwise}; \end{cases}$

- $\mathcal{I}_{i+1}(u,\mathcal{A},E) = \mathcal{I}_i(u,\mathcal{A},E) \cup \{y \mid \exists (x,y) \in \Sigma \text{ s.t. } x \in \mathcal{I}_i(u,\mathcal{A},E) \wedge \nexists (z,y) \in \Sigma \text{ s.t. } z \in E \wedge z \notin Reach_{\mathcal{A}}(b)\}$.

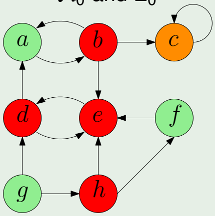
The influenced set of u w.r.t. $\mathcal{A}$ and E is $\mathcal{I}(u,\mathcal{A},E) = \mathcal{I}_n(u,\mathcal{A},E)$ such that $\mathcal{I}_n(u,\mathcal{A},E) = \mathcal{I}_{n+1}(u,\mathcal{A},E)$. □

Reduced AF

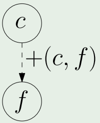
- Given an AF $\mathcal{A}_0$, an extension E_0, and an update $u = \pm(a, b)$, an extension for the updated AF is recomputed for a small part of the updated AF, called *reduced AF* and denoted $\mathcal{R}(u, \mathcal{A}_0, E_0)$
- $\mathcal{R}(u, \mathcal{A}_0, E_0)$ consists of the subgraph of $u(\mathcal{A}_0)$ induced by $\mathcal{I}(u, \mathcal{A}_0, E_0)$
- plus additional nodes/edges representing the "external context":
  1) if there is in $u(\mathcal{A}_0)$ an edge from a node $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ to a node $b \in \mathcal{I}(u, \mathcal{A}_0, E_0)$, we add edge (a, b) if the status of a is IN,
  2) if there is in $u(\mathcal{A}_0)$ an edge from a node $e \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ to a node $c \in \mathcal{I}(u, \mathcal{A}_0, E_0)$ such that e in UN, we add edge (c, c) to $\mathcal{R}_{gr}(u, \mathcal{A}_0, E_0)$

Example (Reduced AF)

$\mathcal{A}_0$ and E_0



$u = +(c, f)$



Sub-AF induced by $\mathcal{I}(u, \mathcal{A}_0, E_0)$

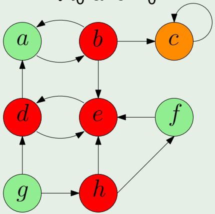
Reduced AF

Reduced AF

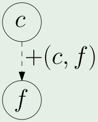
- Given an AF $\mathcal{A}_0$, an extension E_0, and an update $u = \pm(a, b)$, an extension for the updated AF is recomputed for a small part of the updated AF, called *reduced AF* and denoted $\mathcal{R}(u, \mathcal{A}_0, E_0)$
- $\mathcal{R}(u, \mathcal{A}_0, E_0)$ consists of the subgraph of $u(\mathcal{A}_0)$ induced by $\mathcal{I}(u, \mathcal{A}_0, E_0)$
- plus additional nodes/edges representing the "external context":
  1) if there is in $u(\mathcal{A}_0)$ an edge from a node $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ to a node $b \in \mathcal{I}(u, \mathcal{A}_0, E_0)$, we add edge (a, b) if the status of a is IN,
  2) if there is in $u(\mathcal{A}_0)$ an edge from a node $e \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ to a node $c \in \mathcal{I}(u, \mathcal{A}_0, E_0)$ such that e in UN, we add edge (c, c) to $\mathcal{R}_{\text{gr}}(u, \mathcal{A}_0, E_0)$

Example (Reduced AF)

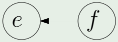
$\mathcal{A}_0$ and E_0



$u = +(c, f)$



Sub-AF induced by $\mathcal{I}(u, \mathcal{A}_0, E_0)$

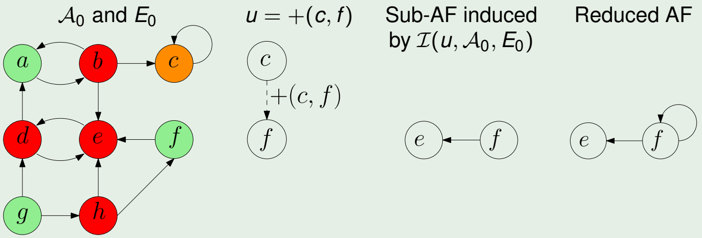


Reduced AF

Reduced AF

- Given an AF $\mathcal{A}_0$, an extension E_0, and an update $u = \pm(a, b)$, an extension for the updated AF is recomputed for a small part of the updated AF, called *reduced AF* and denoted $\mathcal{R}(u, \mathcal{A}_0, E_0)$
- $\mathcal{R}(u, \mathcal{A}_0, E_0)$ consists of the subgraph of $u(\mathcal{A}_0)$ induced by $\mathcal{I}(u, \mathcal{A}_0, E_0)$
- plus additional nodes/edges representing the "external context":
  1) if there is in $u(\mathcal{A}_0)$ an edge from a node $a \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ to a node $b \in \mathcal{I}(u, \mathcal{A}_0, E_0)$, we add edge (a, b) if the status of a is IN,
  2) if there is in $u(\mathcal{A}_0)$ an edge from a node $e \notin \mathcal{I}(u, \mathcal{A}_0, E_0)$ to a node $c \in \mathcal{I}(u, \mathcal{A}_0, E_0)$ such that e in UN, we add edge (c, c) to $\mathcal{R}_{\mathtt{gr}}(u, \mathcal{A}_0, E_0)$

Example (Reduced AF)



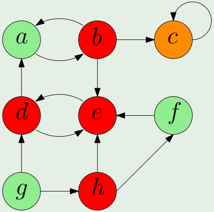
Using extensions of the reduced AF

Theorem (Merging extensions)

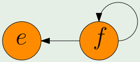
Let $\mathcal{A}_0$ be an AF, and $\mathcal{A} = u(\mathcal{A}_0)$ be the AF resulting from performing update $u = \pm(a, b)$ on $\mathcal{A}_0$. Let $E_0 \in \mathcal{E}_\mathcal{S}(\mathcal{A}_0)$ be an extension for $\mathcal{A}_0$ under a semantics $\mathcal{S} \in \{\mathtt{co}, \mathtt{pr}, \mathtt{st}, \mathtt{gr}\}$. Then, if $\mathcal{E}_\mathcal{S}(\mathcal{R}(u, \mathcal{A}_0, E_0))$ is not empty, then there is an extension $E \in \mathcal{E}_\mathcal{S}(\mathcal{A})$ for the updated AF $\mathcal{A}$ such that
$E = (E_0 \setminus \mathcal{I}(u, \mathcal{A}_0, E_0)) \cup E_d$ where
E_d is an S-extension for reduced AF $\mathcal{R}(u, \mathcal{A}_0, E_0)$.

Example (Merging an initial extension with that of the reduced AF)

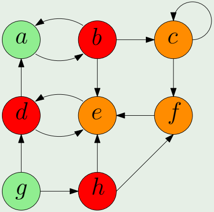
$E_0 \in \mathcal{E}_{\mathtt{pr}}(\mathcal{A}_0)$



$E_d \in \mathcal{E}_{\mathtt{pr}}(\mathcal{R}(u, \mathcal{A}_0, E_0))$



$E \in \mathcal{E}_{\mathtt{pr}}(u(\mathcal{A}_0))$



Incremental Algorithm

Algorithm Incr-Alg($\mathcal{A}_0, u, \mathcal{S}, E_0$, Solver$_{\mathcal{S}}$)

Input: AF $\mathcal{A}_0 = \langle A_0, \Sigma_0 \rangle$, update $u = \pm(a, b)$,
semantics $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}\}$, extension $E_0 \in \mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$,
function Solver$_{\mathcal{S}}(\mathcal{A})$ returning an $\mathcal{S}$-extension for AF $\mathcal{A}$ if it exists, $\perp$ otherwise;
Output: An $\mathcal{S}$-extension $E \in \mathcal{E}_{\mathcal{S}}(u(\mathcal{A}_0))$ if it exists, $\perp$ otherwise;

```
S = I(u, A_0, E_0); // Compute the influenced set
if (S = ∅) then
    return E_0; // If the influenced set is empty, return the initial extension E_0
A_d = R(u, A_0, E_0); // Otherwise, compute the reduced AF
Let E_d = Solver_S(A_d); // Compute an extension for the reduced AF using an external solver
if (E_d ≠ ⊥) then
    return E = (E_0 \ S) ∪ E_d; // Merge E_0 with extension E_d of the reduced AF
else
    return Solver_S(u(A_0)); // If an extension for the reduced AF doesn't exist (it can happen
    for stable semantics only), compute an extension from scratch
```

Theorem (The algorithm is sound and complete)

Let $\mathcal{A}_0$ be an AF, $u = \pm(a, b)$, and $E_0 \in \mathcal{E}_{\mathcal{S}}(\mathcal{A}_0)$ an extension for $\mathcal{A}_0$ under $\mathcal{S} \in \{\texttt{co}, \texttt{pr}, \texttt{st}, \texttt{gr}\}$. If Solver$_{\mathcal{S}}$ is sound and complete then the algorithm computes $E \in \mathcal{E}_{\mathcal{S}}(u(\mathcal{A}_0))$ if $\mathcal{E}_{\mathcal{S}}(u(\mathcal{A}_0)) \neq \emptyset$, otherwise it returns $\perp$.

Outline

Datasets and algorithms

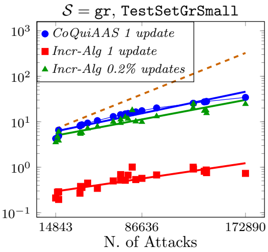
Datasets: ICCMA'15 benchmarks

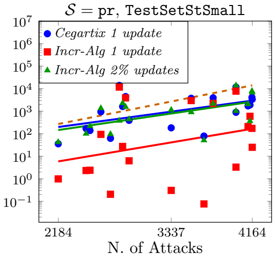
- `TestSetGr` consists of AFs with a very large grounded extension and many arguments in general
- `TestSetSt` consists of AFs with many complete/preferred/stable extensions
- `TestSetSCC` consists of AFs with a rich structure of strongly connected components
- for each of these test sets, three classes of AFs of different sizes: `Small`, `Medium`, and `Large`.

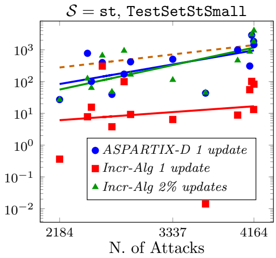
Methodology: the average run time of our algorithm to compute an $\mathcal{S}$-extension was compared with the average run time of the best ICCMA solver to compute an $\mathcal{S}$-extension for $u(\mathcal{A}_0)$ from scratch

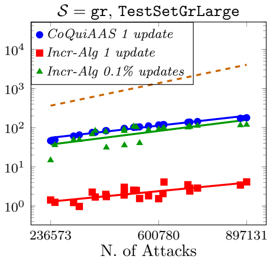
- As Solver$_\mathcal{S}$ for computing an $\mathcal{S}$-extension for the reduced AF we used the solver that won the ICCMA'15 competition for the task $\mathcal{S}$-SE
- *CoQuiAAS* [Lagniez et al. 2015] for $\mathcal{S} = \texttt{co}$ and $\mathcal{S} = \texttt{gr}$
- *Cegartix* [Dvorák et al. 2014] for $\mathcal{S} = \texttt{pr}$
- *ASPARTIX-D* [Gaggl and Manthey 2015] for $\mathcal{S} = \texttt{st}$.

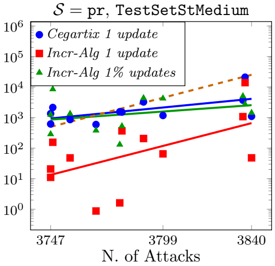
Experimental Results

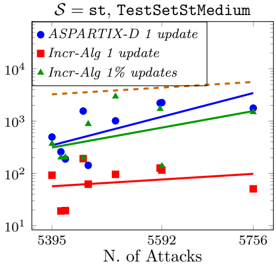












Results

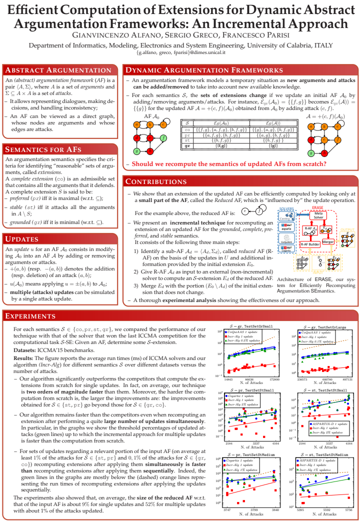
- The size of the reduced AF w.r.t. that of the input AF is about 9% for single updates and 52% for multiple updates with about 1% of the attacks updated.
- Two orders of magnitude faster than the best ICCMA solvers for single updates on average.
- The harder the computation from scratch, the larger the improvements
- Faster even when performing updates simultaneously (green lines) — include the time needed to reduce the application of multiple updates to single attack update
- For sets of updates regarding a relevant portion of the input AF, recomputing extensions after applying them simultaneously is faster than recomputing extensions after applying them sequentially (dashed orange lines)

Outline

Conclusions and Future Work

- Our technique enables any non-incremental algorithm to be used as an incremental one for computing some extension of dynamic AFs
- The technique can be used for general (multiple) updates
- We identified a tighter portion of the updated AF to be examined for recomputing the semantics
- Our algorithm exploits the initial extension of an AF for computing an extension of the updated AF
- The experiments showed that the incremental computation outperforms that of the base (non-incremental) computation
- Future work #1: applying the technique to other argumentation semantics (good results for ideal semantics, using ConArg [Bistarelli et al. 2016])
- Future work #2: enumerating all the extensions and deciding credulous/sceptical acceptance

see you at the poster!



Thank you!

... questions?

Selected References

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

Bei Shui Liao, Li Jin, Robert C. Koons.
Dynamics of argumentation systems: A division-based method.
Artif. Intell., 175(11), 1790–1814, (2011).

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

Jean-Marie Lagniez, Emmanuel Lonca, and Jean-Guy Mailly.
CoQuiAAS: A constraint-based quick abstract argumentation solver.
In *ICTAI*, pages 928–935, 2015.

Wolfgang Dvorák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran.
Complexity-sensitive decision procedures for abstract argumentation.
AI, 206:53–78, 2014.

Sarah Alice Gaggl and Norbert Manthey.
ASPARTIX-D ready for the competition, 2015.

Stefano Bistarelli, Fabio Rossi, and Francesco Santini.
ConArg: A tool for classical and weighted argumentation.
In *COMMA*, 2016.

Sergio Greco, Francesco Parisi.
Efficient Computation of Deterministic Extensions for Dynamic Abstract Argumentation Frameworks.
In *ECAI*, 2016.