Testing Credulous and Sceptical Acceptance in Small-World Networks

Stefano Bistarelli, Fabio Rossi, Francesco Santini

Dipartimento di Matematica e Informatica, Università di Perugia
[bista,rossi,francesco.santini]@dmi.unipg.it

Abstract. In this paper we test how efficiently state-of-the art solvers are capable of solving credulous and sceptical argument-acceptance for lower-order extensions. As our benchmark we consider two different random graph-models to obtain random Abstract Argumentation Frameworks with small-world characteristics: Kleinberg and Watt-Strogatz. We test two reasoners, i.e., ConArg2 and dynPARTIX, on such benchmark, by comparing their performance on NP/co-NP-complete decision problems related to argument acceptance in admissible, complete, and stable semantics.

1 Introduction

An *Abstract Argumentation Framework* (AAF) [8], or System, is simply a pair $\langle A, R\rangle$ consisting of a set A of arguments, and of a binary relation R on A, called the "attack" relation. An abstract argument is not assumed to have any specific structure but, roughly speaking, an argument is anything that may attack or be attacked by another argument. Two main styles of argumentation semantics definition can be identified in the literature: *extension-based* and *labelling-based*. In this work we exploit the extension-based approach, where a given semantics definition (related to varying degrees of scepticism or credulousness) specifies how to derive a set of extensions from an AAF, which basically consist in conflict-free subsets of A with different properties.

In this paper we are interested in the acceptance (or justification) state of arguments: intuitively an argument is regarded as accepted if it is at least once (credulously) or always (sceptically) present in all the extensions satisfying a given semantics. The kind of semantics (from le least to the most binding), and its acceptance state (e.g., credulous or sceptical) point to a strength evaluation of an argument.

In this work we move along the line activated in our previous works [5, 2, 4, 3]. Differently from these papers, here we extend [3] by considering networks with small-world topologies [15, 18]: in [3] we present a benchmark assembled with random trees, scale-free networks [1], and just random graphs [13]. The motivation is to study topologies possibly shown by advanced social debating platforms, as *DebateGraph*[1], which allow a discussion to be less rigidly structured

[1] http://debategraph.org

than a tree, as instead commonly offered in today's digital fora. Hence, in this paper we consider Kleinberg [15] and Watts-Strogatz [18] topologies.

The problems we tackle in this work correspond to the credulous acceptance in admissible, complete, and stable semantics (all NP-complete problems), and the sceptical acceptance in the stable semantics (a coNP-complete problem). We test two different solvers ConArg2 [7] and dynPARTIX [10], in order to have a comparison between them and a more informative analysis on the most efficient relation "technology against AAF topology" (i.e., Constraint Programming against Dynamic Programming).

Note that in this work we do not consider higher-order semantics, e.g., preferred or grounded [8], which can be defined from lower-order ones by selecting only the maximal or minimal ones w.r.t. set inclusion; for instance, preferred extensions are the maximal (w.r.t. set inclusion) admissible extensions. We leave their testing to future work (see Sec. 5).

2 Preliminaries

In this section we focus on the basic definitions of an AAF, and on the extension-based semantics that will be tested in our comparison (see Sec. 4).

Definition 1 (Abstract AFs). *An Abstract Argumentation Framework (AAF) is a pair $F = \langle A, R \rangle$ of a set A of arguments and a binary relation $R \subseteq A \times A$, called the attack relation. $\forall a, b \in A$, $aR\,b$ (or, $a \rightarrowtail b$) means that a attacks b. An AAF may be represented by a directed graph (an interaction graph) whose nodes are arguments and edges represent the attack relation. A set of arguments $S \subseteq A$ attacks an argument a, i.e., $S \rightarrowtail a$, if a is attacked by an argument of S, i.e., $\exists b \in S . b \rightarrowtail a$.*

The following notion of defence [8] is fundamental to AAFs.

Definition 2 (Defence). *Given an AAF, $F = \langle A, R \rangle$, an argument $a \in A$ is defended (in F) by a set $S \subseteq A$ if for each $b \in A$, such that $b \rightarrowtail a$, also $S \rightarrowtail b$ holds. Moreover, for $S \subseteq A$, we denote by S_R^+ the set $S \cup \{b \mid S \rightarrowtail b\}$.*

The "acceptability" of an argument [8], defined under different semantics, depends on the frequency of its membership to some argument subsets, called *extensions*: such semantics characterise a collective "acceptability". In Def. 3 we report only the semantics of interest in this study.

Definition 3. *Let $F = \langle A, R \rangle$ be an AAF. A set $S \subseteq A$ is conflict-free (in F), denoted $S \in cf(F)$, iff there are no $a, b \in S$, such that $(a, b), (b, a) \in R$. For $S \in cf(F)$, it holds that*

- *$S \in adm(F)$, if each $a \in S$ is defended by S;*
- *$S \in com(F)$, if $S \in adm(F)$ and for each $a \in A$ defended by S, $a \in S$ holds;*
- *$S \in stb(F)$, if for each $a \in A \backslash S$, $S \rightarrowtail a$, i.e., $S_R^+ = A$;*

Table 1: Some known complexity results for the credulous and sceptical acceptance [16, Ch. 5]: in bold, the problems we test in this study.

	adm	com	stb
Credulous acc.	**NP-c**	**NP-c**	**NP-c**
Sceptical acc.	trivial	P-c	**coNP-c**

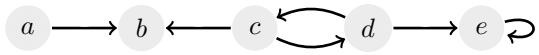

Fig. 1: A graphical example of an AAF.

We recall that for each AAF, $stb(F) \subseteq com(F) \subseteq adm(F)$ holds, and that $adm(F) \neq \emptyset$, and $com(F) \neq \emptyset$ always hold, while $stb(F) = \emptyset$ may happen instead.

Definition 4 (Acceptance state). *Given a semantics σ (e.g.,* stb*) and a framework F, an argument a is* i) *sceptically accepted iff $\forall E \in \sigma(F), a \in E$, and* ii) *credulously accepted if $\exists E \in \sigma(F), a \in E$.*

Checking the credulous/sceptical acceptance of an argument is sometimes a (time) complex problem (e.g., with the grounded semantics they are polynomial): in Tab. 1 we report in bold the complexity class of the problems we tackle in this paper, i.e., the credulous acceptance in the admissible, complete, and stable semantics (all NP-complete problems), and the sceptical acceptance in the stable semantics (a coNP-complete problem).

Consider the $F = \langle A, R \rangle$ in Fig. 1, with $A = \{a, b, c, d, e\}$ and $R = \{(a, b), (c, b), (c, d), (d, c), (d, e), (e, e)\}$. We have that $stb(F) = \{\{a, d\}\}$. The admissible extensions of F are $adm(F) = \{\emptyset, \{a\}, \{c\}, \{d\}, \{a, c\}, \{a, d\}\}$, while the complete ones are $com(F) = \{\{a\}, \{a, c\}, \{a, d\}\}$. For instance, argument a is both credulously and sceptically accepted in $stb(F)$ and $com(F)$, while it is only credulously accepted in $adm(F)$.

3 Tools and Graphs

In this section we introduce how we performed our tests, by describing the analysed tools (Sec. 3.1) and the generated AAFs (Sec. 3.2).

3.1 Tools

dynPARTIX[2] is a system based on decomposition and dynamic programming; it is motivated by the theoretical results in [10]. The underneath algorithms make use of the graph-parameter tree-width, which measures the "tree-likeness" of a graph. More specifically, tree-width is defined via the so-called

[2] `http://www.dbai.tuwien.ac.at/proj/argumentation/dynpartix/`

tree-decompositions. A tree decomposition is a mapping from an AAF to a tree where the nodes in the tree contain *bags* of arguments from the AAF. Each argument appears in at least one bag, adjacent arguments are together in at least one bag, and bags containing the same argument are connected. The benchmarks in [9] show that the run-time performance of dynPARTIX heavily depends on the tree-width of the considered graph: for example, with instances of small tree-width, dynPARTIX outperforms ASPARTIX [12], while with a high tree-width ASPARTIX still performs comparably (better on credulous acceptance, still worse on sceptical acceptance). In our tests we use the new 64-bit version (2.0) of dynPARTIX, which has recently become available.

ConArg2. ConArg[3] [7] is our solver based on the *Java Constraint Programming* solver[4] (JaCoP), a Java library that provides a *Finite Domain Constraint Programming* paradigm [17]. The tool comes with a graphical interface, which visually shows all the obtained extensions for each problem. ConArg is able to solve also the weighted and coalition-based problems presented in [6]. Moreover, it can import/export AAFs with the same text format of ASPARTIX. Recently, we have extended the tool to its second version, i.e., ConArg2 (freely downloadable from the same Web-page of ConArg), in order to improve its performance: we implemented all the models in Gecode[5], which is an open, free, and efficient C++ environment where to develop constraint-based applications. Hence, we model each semantics as a *Constraint Satisfaction Problem* (*CSP*) [17]. We have also dropped the graphical interface, having a textual output only, with the purpose to have a tool exclusively oriented to performance. So far, on classical AAFs, ConArg2 is able to find all conflict-free, admissible, complete, stable, grounded, preferred, semi-stable, and ideal extensions; moreover, it solves the credulous and sceptical acceptance of arguments given the admissible, complete, and stable semantics, and the existence of a stable extension (which is an NP-complete problem).

3.2 Graphs

The justification behind using Kleinberg and Watts-Strogatz models is that several works in the Argumentation literature investigate AAFs extracted from social networks [14]. However, benchmarks collected with such tools are still not available.

To generate random graphs we adopted two different libraries. The first one is the *Java Universal Network/Graph Framework* (*JUNG*[6]), which is a Java software library for the modelling, generation, analysis and visualization of graphs. With JUNG we generate Kleinberg [15] graphs. The second library we use is *NetworkX*[7], and it consists of a Python software package for the creation, manipu-

[3] `http://www.dmi.unipg.it/conarg/`
[4] `http://www.jacop.eu`
[5] `http://www.gecode.org`
[6] `http://jung.sourceforge.net`
[7] `http://networkx.github.io`

Model/Nodes	Edges	Shortest Path	Clustering C.	Diam.	InDeg.	Cycles	Min-Max Deg.
KL/16	47	1.6	0.3	2.9	5	Yes	5-11
KL/25	74	1.9	0.19	3	5	Yes	5-11
KL/36	107	2.2	0.14	3.9	5	Yes	5-11
KL/49	146	2.4	0.11	4	5	Yes	5-11
KL/64	191	2.57	0.8	4.2	5	Yes	5-12
KL/81	242	2.7	0.07	4.8	5	Yes	5-11
WS/25	50	2.4	0.22	4.7	2	Yes	2-8
WS/50	100	3.4	0.28	6.7	2	Yes	2-8
WS/80	240	2.6	0.09	4.9	3	Yes	3-13
WS/100	400	2.4	0.12	4	4	Yes	4-15

Table 2: Analysis of the generated random-AFs: values are averaged over the generated 100 AFs in each class. The considered models are Kleinberg (KL) and Watts-Strogatz (WS).

lation, and study of the structure, dynamics, and functions of complex networks. With NetworkX we generate Watts-Strogatz [18] graphs.

The **Kleinberg** [15] graph-model adds a number of directed long-range random links to an $n \times n$ lattice network. Links have a non-uniform distribution that favours edges to close nodes over more distant ones (in the number of hops). In the implementation provided by JUNG, each node u has four local connections, one to each of its neighbours, and in addition, one or more long-range connections to some node v, where v is randomly chosen according to probability proportional to $d^{-\theta}$ where d is the lattice distance between u and v and θ is the clustering exponent. In our generation we set $\theta = 0.9$, in order to have a high clustering coefficient. Given a node, each link is directed towards its neighbours: edges are created in both directions between neighbours. Each long-distance edge has the tail in the considered node.

The **Watts-Strogatz** model [18] consists in a ring over n nodes. each node in the ring is connected with its k nearest neighbours ($k-1$ neighbours if k is odd). Then shortcuts are created by replacing some edges as follows: for each edge (u, v) in the underlying "n-ring with k nearest neighbours with probability p replace it with a new edge (u, w) with uniformly random choice of existing node w. Varying p makes it possible to interpolate between a regular lattice ($p = 0$) and an Erdős-Rényi graph ($p = 1$). In our graph generation we set $k = (n/10) - 2$ and $p = 0.1$: in this way we obtain a high clustering coefficient (the max is with $k = n/2$, i.e., a complete graph) without increasing the number of edges (attacks) too much; we also obtain a graph structure closer to a lattice (p is low). Note that, since NetworkX generates undirected Watts-Strogatz graphs, we orient each edge in one of the two directions (0.5 of probability each).

For each set of 100 networks we also collected the following parameters, shown in Tab. 2: the Average Number of Edges, the Average Shortest Path (between all the couples of nodes), the Average Clustering Coefficient (the fraction of all the possible triangles through a node), the Average Diameter, the Average InDegree, the presence of Simple Cycles (closed paths where no node appears twice, except that the first and last node are the same), and the Max and Min degree for a node over the whole class.

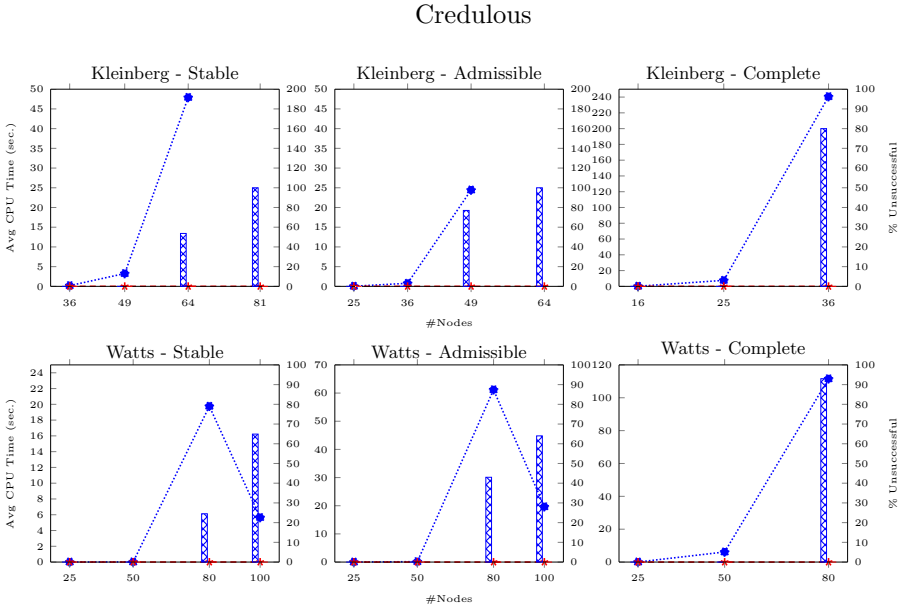

Fig. 2: Avg. time dynPARTIX ( ), ConArg2 ( ), % Unsuc. instances dynPARTIX ( ).

4 Tests and Discussion

Performance results have been collected on an Intel(R) Core(TM) i7 CPU 970 @3.20GHz (6 core, 2 threads per core), and 16GB of RAM. For both the tools, the output has been redirected to */dev/null*, and the standard error to file. To test dynPARTIX we adopted its 64-bit version, by calling commands like `./dynpartix -f barabasi_graph -s admissible --skept d`. Flag `-f` specifies the input file, `-s` the considered semantics (admissible in this case), and `--skept` sets argument with id d to be checked for sceptical acceptance. We could flag `-n semi`, i.e., the semi-normalised tree-decomposition (more performant than the default normalised one, as stated by the authors of dynPARTIX) only for the admissible semantics, because it is not currently implemented for the other two semantics. We set a timeout of 300 seconds to interrupt the search of each of the two tools.

In Fig. 2 and Fig. 3 we show the tests collected on the whole database presented in Tab. 2, for the credulous and sceptical acceptance respectively. For each of the reported number of nodes (on the x axis), the left y axis reports the CPU time averaged over 100 different instances of AAFs, and 10% random arguments chosen on that class. Acceptance and non-acceptance have been forced to be equally distributed within such random sample (5% each). On the right y axis we also report the percentage of instances that are not solved within the timeout ("% Unsuccessful").

Sceptical

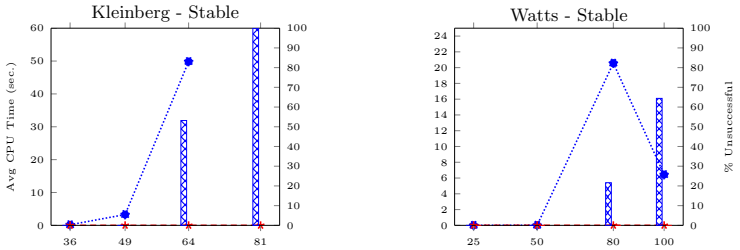

Fig. 3: Avg. time dynPARTIX (···●···), ConArg2 (-·*·-), % Unsuc. instances dynPARTIX (▯▯).

As we can appreciate from Fig. 2 and Fig. 3, we can state that constraint propagation in ConArg2 works very well with credulous/sceptical acceptance of arguments: most of the problems are solved almost instantaneously, while dynPARTIX is not able to return an answer within the timeout. Note that the performance of dynPARTIX on sceptical acceptance are proven to be better (with low tree-width graphs) or comparable (with high tree-width graphs) to the performance of ASPARTIX, while ASPARTIX works definitely better with high tree-width and credulous acceptance [9].

5 Conclusion

In the paper we have compared two reasoners (dynPARTIX and ConArg2). The main goal has been to study how efficiently state-of-the-art reasoners behave on hard problems related to credulous and sceptical acceptance in lower-order semantics, i.e., admissible, complete, and stable.

In the future we will implement in ConArg2 all the other hard problems related to higher-order semantics [16, Ch. 5]; in particular, credulous/sceptical acceptance in preferred (NP-c/Π_2^P-c), semi-stable (Σ_2^P-c/Π_2^P-c), and stage semantics (Σ_2^P-c/Π_2^P-c), with the purpose to compare our tool with dynPARTIX again, but also with CEGARTIX[8] [11], since it computes sceptical acceptance for the preferred semantics, and both sceptical and credulous acceptance for semi-stable and stage semantics. Moreover, in order to have an engine as more comprehensive as possible, we plan to solve other hard problems (not currently implemented in any other solver) related to the preferred semantics, e.g., its verification (co-NP-complete), and non-emptiness (NP-complete).

To solve higher-order semantics, we will need to work on branch-and-bound search itself, with the result to better manage maximality/minimality of set inclusion directly at the search level; the reason is that it is usually not possible to express such requirements as constraints.

[8] `http://www.dbai.tuwien.ac.at/proj/argumentation/cegartix/`

References

1. A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
2. S. Bistarelli, F. Rossi, and F. Santini. Benchmarking hard problems in random abstract AFs: The stable semantics. In *Computational Models of Argument - Proceedings of COMMA*, volume 266 of *FAIA*, pages 153–160. IOS Press, 2014.
3. S. Bistarelli, F. Rossi, and F. Santini. Efficient solution for credulous/sceptical acceptance in lower-order Dung's semantics. In *26th International Conference on Tools with Artificial Intelligence*, ICTAI, pages 800–804. IEEE Computer Society, 2014.
4. S. Bistarelli, F. Rossi, and F. Santini. Enumerating extensions on random abstract-afs with argtools, aspartix, conarg2, and dung-o-matic. In *Computational Logic in Multi-Agent Systems - 15th International Workshop, CLIMA XV*, volume 8624 of *LNCS*, pages 70–86. Springer, 2014.
5. S. Bistarelli, F. Rossi, and F. Santini. A first comparison of abstract argumentation reasoning-tools. In *ECAI 2014 - 21st European Conference on Artificial Intelligence*, volume 263 of *FAIA*, pages 969–970. IOS Press, 2014.
6. S. Bistarelli and F. Santini. A common computational framework for semiring-based argumentation systems. In *ECAI 2010 - 19th European Conference on Artificial Intelligence*, volume 215 of *FAIA*, pages 131–136. IOS Press, 2010.
7. S. Bistarelli and F. Santini. Conarg: A constraint-based computational framework for argumentation systems. In *23rd International Conference on Tools with Artificial Intelligence*, ICTAI, pages 605–612. IEEE Computer Society, 2011.
8. P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–357, 1995.
9. W. Dvořák, M. Morak, C. Nopp, and S. Woltran. dynpartix- A dynamic programming reasoner for abstract argumentation. In *Applications of Declarative Programming and Knowledge Management*, pages 259–268. Springer, 2013.
10. W. Dvorák, R. Pichler, and S. Woltran. Towards fixed-parameter tractable algorithms for abstract argumentation. *Artif. Intell.*, 186:1–37, 2012.
11. W. Dvořák, M. Järvisalo, J. P. Wallner, and S. Woltran. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.*, 206:53–78, Jan. 2014.
12. U. Egly, S. A. Gaggl, and S. Woltran. Answer-set programming encodings for argumentation frameworks. *Argument & Computation*, 1(2):147–177, 2010.
13. P. Erdős and A. Rényi. On the evolution of random graphs. *Bulletin of the International Statistical Institute*, 38(4):343–347, 1961.
14. S. Gabbriellini and P. Torroni. Arguments in social networks. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1119–1120. IFAAMAS, 2013.
15. C. Martel and V. Nguyen. Analyzing Kleinberg's (and other) small-world models. In *Proceedings of the ACM symposium on Principles of distributed computing*, PODC, pages 179–188. ACM, 2004.
16. I. Rahwan and G. R. Simari. *Argumentation in Artificial Intelligence*. Springer Publishing Company, Incorporated, 1st edition, 2009.
17. F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., 2006.
18. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.