ECAI 2014
T. Schaub et al. (Eds.)

doi:10.3233/978-1-61499-419-0-1117

Argumentation Frameworks Features: an Initial Study

Mauro Vallati[1] and **Federico Cerutti**[2] and **Massimiliano Giacomin**[3]

Abstract. Semantics extensions are the outcome of the argumentation reasoning process: enumerating them is generally an intractable problem. For preferred semantics two efficient algorithms have been recently proposed, PrefSAT and SCC-P, with significant runtime variations. This preliminary work aims at investigating the reasons (argumentation framework features) for such variations. Remarkably, we observed that few features have a strong impact, and those exploited by the most performing algorithm are not the most relevant.

1 INTRODUCTION

Abstract argumentation is a popular approach for non-monotonic reasoning. Although it is built on just arguments — vertexes of a graph — and attacks — directed edges — it has been proved able to encompass other approaches to non-monotonic reasoning [5]. Given these simple structures, argumentation theory prescribes several semantics [1]: each semantics identifies the so-called *semantics extensions*, viz. sets of arguments that are "collectively acceptable".

In this paper we focus on the so-called *preferred semantics*: it has interesting theoretical properties [5] making it one of the most used semantics. The enumeration problem associated to this semantics is at the second level of the polynomial hierarchy [6]: this justifies the search for efficient mechanisms for solving it (*solvers*). Among others, two solvers have been recently proposed: PrefSAT [3], which translates the enumeration problem into several SAT problems, and SCC-P [4], which exploits the SCC-recursiveness schema [2]. PrefSAT is more efficient than other state-of-the-art solvers [3], but in some situations it is way less efficient than SCC-P [4].

The aim of this paper is to explain the significant runtime variation between PrefSAT and SCC-P [4] by implementing so-called *empirical performance models (EPMs)* [8]. First, we run the two solvers on a large number of instances (AFs). For each instance, the solvers' performances are recorded and a set of *instances features* is computed. Each feature is a real number that summarises a property of the instance. A predictive model is then learnt as a mapping *instance features–solver performance*. Finally, the EPMs are cross-validated.

Although EPMs are well established within AI [8], as to our knowledge this is the first study in argumentation theory (whose main concepts are summarised in Section 2). Since the critical step to building accurate EPMs is identifying a "good" set of instance features, the core of this paper is to investigate suitable features for argumentation frameworks (Section 3): as a preliminary study, we focused on features related to graph theory. Finally Section 4 summarises the main contributions and concludes the paper.

[1] University of Huddersfield, UK, email: m.vallati@hud.ac.uk
[2] University of Aberdeen, UK, email: f.cerutti@abdn.ac.uk
[3] University of Brescia, Italy, email: massimiliano.giacomin@unibs.it

2 PRELIMINARIES

An *argumentation framework* (AF) [5] is a pair $\langle \mathcal{A}, \rightarrow \rangle$, where $\mathcal{A}$ is a set of arguments (vertexes of a graph) and $\rightarrow \subseteq \mathcal{A} \times \mathcal{A}$ is a set of attacks (directed edges). $S \subseteq \mathcal{A}$ is *conflict-free* iff $\forall \mathbf{a}, \mathbf{b} \in S, \langle \mathbf{a}, \mathbf{b} \rangle \notin \rightarrow$. An argument $\mathbf{a} \in \mathcal{A}$ is *acceptable* w.r.t. $S \subseteq \mathcal{A}$ iff $\forall \mathbf{b} \in \mathcal{A} \mid \langle \mathbf{b}, \mathbf{a} \rangle \in \rightarrow, \exists \mathbf{c} \in S \mid \langle \mathbf{c}, \mathbf{b} \rangle \in \rightarrow$. $S \subseteq \mathcal{A}$ is *admissible* iff it is conflict-free and each argument of S is acceptable w.r.t. S. $S \subseteq \mathcal{A}$, admissible, is a *complete extension* iff each argument, which is acceptable w.r.t. S, belongs to S. $S \subseteq \mathcal{A}$ is a *preferred extension* iff it is a maximal (w.r.t. set inclusion) complete extension.

Two efficient algorithms for preferred extensions enumeration have been proposed: PrefSAT [3] and SCC-P [4]. PrefSAT solves the SAT problem equivalent to find a complete extension in an AF, and finds the preferred extensions through hill-climbing. The already found extensions are excluded by subsequent search steps. SCC-P implements the SCC-recursiveness schema [2]. First, the extensions of the frameworks restricted to the initial (i.e. not receiving any attack) strongly connected components (SCCs) are computed and combined together. Then each SCC which is attacked only from initial SCCs is considered: the extensions of such a SCC are locally computed and merged with those already obtained. Then the subsequent SCCs are considered until no remaining SCCs are left to process. The schema is recursive: for each SCC (1) all arguments attacked by the extension selected in the previous SCCs are suppressed; (2) the procedure is recursively applied to the remaining part of the SCC. The base of the recursion is reached when there is only one SCC: in this case a solver similar to PrefSAT is called. Unsurprisingly, SCC-P proved to be more efficient than PrefSAT on AFs with a significant number of SCCs [4]. In the other cases, PrefSAT is much more efficient than the other state-of-the-art solvers [3, 4].

3 ANALYSIS

PrefSAT, SCC-P and the features extraction have been run on a cluster with computing nodes equipped with 2.5 Ghz Intel Core 2 Quad Processors™, 8 GB of RAM. A cutoff of 15 minutes was imposed to compute the preferred extensions for each AF. For each solver we recorded the overall result: success (found all the preferred extensions), crashed, timed-out or ran out of memory. Unsuccessful runs were assigned a runtime equal to the cutoff.

The 10,000 AFs are generated using a parametric random approach allowing to select (probabilistically — average, standard deviation) the density of attacks for each SCC; and how many arguments (probabilistically) in each SCC attack how many arguments (probabilistically) in how many (probabilistically) other SCCs. The number of arguments ranges between 10 and 40,000. We exploited a 10-fold cross-validation approach on a uniform random permutation of our instances — a standard method where nine slices are used for training and the tenth for testing. EPMs are built using WEKA [7].

Features. We considered 26 features, belonging to 5 classes: *graph size* (5), *degree* (4), *SCC* (5), *graph structure* (5), *times* (7).
Graph size features: number of vertices, number of edges, ratios vertices–edges and inverse, and graph density (NT – non trivial).
Degree features (overall NT): average, standard deviation, maximum, minimum degree values across the nodes in the graph.
SCC features (overall NT): number of SCCs, average, standard deviation, maximum and minimum size.
Graph structure: presence of auto-loops, number of isolated vertices (NT), flow hierarchy (NT) and results of test on Eulerian (NT) and aperiodic structure of the graph (NT).
Finally, the needed *CPU-time* for extracting each NT feature has been considered. It is worth mention that usually the overall features extraction process takes less than a CPU-time second per instance apart from the aperiodicity of the graph (up to 20 seconds). A feature is trivial if its extraction requires less than 0.001 seconds.
Prediction and Feature Importance. We considered EPMs for both classification and regression approaches. Classification approaches classify the AF into a single category: this indicates the algorithm which is predicted to be the fastest. Regression techniques model the behaviour of each algorithm for predicting its runtime on a new AF. Since solvers runtimes vary from 0.01 to 900 CPU seconds, we trained our regression models to predict log-runtime rather than absolute runtime: this have been effective in similar circumstances [8]. We first assessed the performance of various classification and regression models, and we observed that random forests performed best in classification, and M5-Rules in regression.

Table 1 shows the results for prediction and regression (RMSE) using the best performing models, with 10-fold cross validation on a uniform random permutation of our full set of 10,000 AFs. Results are shown by generating EPMs considering the single best feature (B1), the best two and three of them (B2, B3), all the features but SCC-related ones (A-S) and all the extracted features (*All*). Best features are selected according to a greedy forward search in the space of features: starting from an empty subset, it iteratively adds a new feature to the considered subset. It stops when the addition of any attribute results in the decrease of the evaluation. The evaluation of a subset of features is done by considering the predictive ability of each feature along with the degree of redundancy between them.

In Table 1, the three best features for the classification task are: (1) the density of the AF, (2) the minimum degree value and (3) the number of SCCs. Remarkably, (1) allows to achieve, even while used alone, significant classification performances. From [4], SCC-related information are believed to be very informative, thus without SCC features we expect a noticeable decrease of the EPMs performance. Interestingly, considering the number of SCCs in the 3-best features introduces some noise that worsen the performance. For the regression task, the three best features for predicting PrefSAT runtimes are the same exploited for classification, while for SCC-P are: (1) the density, (2) the number of SCCs and (3) the size of the largest SCC. According to the results, SCC-P performances are somehow easier to predict than PrefSAT ones. Differently from classification, for the regression task exploiting the three best features allows significantly better runtime prediction than the best single feature.

In addition to the results discussed in Table 1, it is worth mentioning that: (I) for AFs with few arguments ("small") the classification process mainly depends on the ratio vertices/edges, while for "bigger" AFs it mainly depends on the average degree value; (II) considering only SCC-related features drops the classification accuracy — proportion of instances correctly classified — to 62.3%; (III) the EPM precision — the proportion of true positives within a class — is similar between the two algorithms; (IV) considering regression EPM for SCC-P, 88.9% of the predictions are within a factor of 1 of the observed runtimes with the full feature set (*All*), vs. 82.4% and 60.4% when using respectively B3 and B1 features; (V) for PrefSAT the corresponding values are: 77.5%, 75.9% and 61.0%; (VI) not considering aperiodicity of the AF — the most expensive feature to extract — the regression performance does not change, while classification accuracy decreases by 0.2%.

Table 1. Evaluation of classification and regression EPMs for different features subsets. B1, B2, B3 indicates the best 1, 2 and 3 features according to a greedy forward search; A-S indicates all the features but SCC-related ones; *All* indicates all the extracted features

Classification (Higher is better)					
	B1	B2	B3	A-S	*All*
Accuracy	83.5%	82.6%	80.9%	83.5%	**84.4%**
Precision PrefSAT	84.0%	83.3%	80.3%	83.3%	**84.2%**
Precision SCC-P	83.1%	82.0%	81.4%	83.7%	**84.6%**
Regression - RMSE log (Lower is better)					
	B1	B2	B3	A-S	*All*
PrefSAT	1.39	1.31	0.93	**0.89**	**0.89**
SCC-P	1.36	0.80	0.78	0.76	**0.75**

4 CONCLUSIONS

Exploiting EPMs for predicting solvers' performance is known to lead to practical improvements [8] and to provide useful insights that can be exploited for further solvers improvements. In this work we have investigated the use of EPMs for two efficient solvers that enumerate the preferred extensions of argumentation frameworks, identifying a set of features and analysing their effectiveness for both classification and regression predicting tasks.

Table 1 shows that a small and easy to compute set of features leads to good EPMs. Remarkably, although one of the considered solvers relies on SCC-decomposition of the AF, SCC related features are very informative only in conjunction with other features.

Future work includes extending this preliminary work on features collection, by encoding AFs in different structures and by considering probing features; considering other state-of-the-art solvers; and engineering a portfolio approach.

REFERENCES

[1] Pietro Baroni, Martin Caminada, and Massimiliano Giacomin, 'An introduction to argumentation semantics', *Knowledge Engineering Review*, **26**(4), 365–410, (2011).
[2] Pietro Baroni, Massimiliano Giacomin, and Giovanni Guida, 'SCC-recursiveness: a general schema for argumentation semantics', *Artificial Intelligence*, **168**(1-2), 165–210, (2005).
[3] Federico Cerutti, Paul E. Dunne, Massimiliano Giacomin, and Mauro Vallati, 'Computing preferred extensions in abstract argumentation: A sat-based approach', in *TAFA*, pp. 176–193, (2013).
[4] Federico Cerutti, Massimiliano Giacomin, Mauro Vallati, and Marina Zanella, 'A SCC recursive meta-algorithm for computing preferred labellings in abstract argumentation', in *KR*, (2014). to appear.
[5] Phan M. Dung, 'On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games', *Artificial Intelligence*, **77**(2), 321–357, (1995).
[6] Paul E. Dunne and Michael Wooldridge, 'Complexity of abstract argumentation', in *Argumentation in AI*, eds., I Rahwan and G Simari, chapter 5, 85–104, Springer-Verlag, (2009).
[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, 'The WEKA data mining software: An update', *SIGKDD Explorations*, **11**(1), 10–18, (2009).
[8] Frank Hutter, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown, 'Algorithm runtime prediction: Methods & evaluation', *Artificial Intelligence*, **206**(0), 79 – 111, (2014).