Where Are We Now? State of the Art and Future Trends of Solvers for Hard Argumentation Problems

Federico CERUTTI\textsuperscript{a,1}, Mauro VALLATI\textsuperscript{b} and Massimiliano GIACOMIN\textsuperscript{c}

\textsuperscript{a}Cardiff University, UK
\textsuperscript{b}University of Huddersfield, UK
\textsuperscript{c}University of Brescia, Italy

Abstract. We evaluate the state of the art of solvers for hard argumentation problems—the enumeration of preferred and stable extensions—to envisage future trends based on evidence collected as part of an extensive empirical evaluation. In the last international competition on computational models of argumentation a general impression was that reduction-based systems (either SAT-based or ASP-based) are the most efficient.

Our investigation shows that this impression is not true in full generality and suggests the areas where the relatively under-developed non reduction-based systems should focus more to improve their performance. Moreover, it also highlights that the state-of-the-art solvers are very complementary and can be successfully combined in portfolios: our best per-instance portfolio is 51\% (resp. 53\%) faster than the best single solver for enumerating preferred (resp. stable) extensions.

Keywords. Abstract Argumentation, Solvers for Argumentation Problems, Portfolios methods for Argumentation

1. Introduction

An abstract argumentation framework (AF) consists of a set of arguments and a binary attack relation between them. In \cite{9} four semantics were introduced, namely grounded, preferred, complete, and stable semantics: each of them lead to a single or to multiple extensions (or no extensions in the case of stable semantics) where an extension is intuitively a set of arguments which can “survive the conflict together.” We refer the reader to \cite{2} for a detailed analysis. Moreover, for each semantics, several decision and enumeration problems have been identified. In this paper we focus on the enumeration of preferred and stable extensions because: (i) the solution to the problem of enumerating extensions implies the answer to other problems; (ii) the problems of enumerating preferred and stable extensions are among the hardest in abstract argumentation.

Research around argumentation-based technology is fast growing: for instance, three of the most cited papers (top-25) published on Artificial Intelligence Journal since 2011...
according to Scopus\(^2\) are in this field, and the last International Competition on Computational Models of Argumentation (ICCMA-15) received more submissions than the last ASP competition.

The results of ICCMA-15\(^3\) [17] suggest that (i) reduction-based systems (either SAT-based or ASP-based) are more efficient than non reduction-based: indeed the best solvers for enumerating stable and preferred extensions are either SAT-based or ASP-based; and (ii) a mixture of approaches can be fruitful: CoQuiAas—that scored first among all for each semantics considered in ICCMA-15—uses a variety of approaches.

Here, we test how general such conclusions are with a large empirical investigation focused on enumeration of stable and preferred extensions using the solvers submitted to ICCMA-15. By adopting different metrics, we identified avenues for improvement that we hope will be valuable for solvers’ authors and for the argumentation community.

Solvers indeed proved to be very complementary (i.e. a mixture of approaches can be fruitful), and we then exploit portfolio approaches in order to highlight (relative) strengths and weaknesses of solvers. As testified by experiences in other research areas in artificial intelligence, such as planning [19], SAT [21], and ASP [12], portfolios and algorithm selection techniques [14] are very useful tools for understanding the importance of solvers, evaluate the improvements, and effectively combine solvers for increasing overall performance. Existing works [6,5] either focus on algorithm selection for enumerating preferred extensions, with a very small number of solvers and of instances; or on theoretical complementariness of algorithms.

Our findings reshape one of the take-away messages from ICCMA-15, namely that reduction-based systems have higher performance than non reduction-based. This is not always the case, although it is the case that they have better coverage, and ICCMA-15 privileged coverage against speed.

Finally, the analysis of portfolio techniques—and their generalisation capabilities—highlighted that, by combining solvers, it is possible to increase the coverage of 13% (resp. 3%) and the speed of 51% (resp. 53%) against the best single solver for enumerating preferred (resp. stable) extensions.

\section{2. Dung’s Argumentation Framework}

An argumentation framework [9] consists of a set of arguments and a binary attack relation between them.\(^4\)

\textbf{Definition 1.} An argumentation framework (AF) is a pair \(\Gamma = (\mathcal{A}, R)\) where \(\mathcal{A}\) is a set of arguments and \(R \subseteq \mathcal{A} \times \mathcal{A}\). We say that \(b\) attacks \(a\) iff \(\langle b, a \rangle \in R\), also denoted as \(b \rightarrow a\).

The basic properties of conflict-freeness, acceptability, and admissibility of a set of arguments are fundamental for the definition of argumentation semantics.

\textbf{Definition 2.} Given an AF \(\Gamma = (\mathcal{A}, R)\):


\(^3\)http://argumentationcompetition.org/2015/results.html

\(^4\)In this paper we consider only finite sets of arguments: see [3] for a discussion on infinite sets of arguments.
• a set \( S \subseteq \mathcal{A} \) is a conflict–free set of \( \Gamma \) if \( \forall a, b \in S \ s.t. \ a \rightarrow b \);
• an argument \( a \in \mathcal{A} \) is acceptable with respect to a set \( S \subseteq \mathcal{A} \) of \( \Gamma \) if \( \forall b \in \mathcal{A} \ s.t. \ b \rightarrow a, \exists c \in S \ s.t. \ c \rightarrow b \);
• a set \( S \subseteq \mathcal{A} \) is an admissible set of \( \Gamma \) if \( S \) is a conflict–free set of \( \Gamma \) and every element of \( S \) is acceptable with respect to \( S \) of \( \Gamma \).

An argumentation semantics \( \sigma \) prescribes for any \( AF \) \( \Gamma \) a set of extensions, namely a set of sets of arguments satisfying the conditions dictated by \( \sigma \).

**Definition 3.** Given an \( AF \) \( \Gamma = (\mathcal{A}, \mathcal{R}) \): a set \( S \subseteq \mathcal{A} \) is a:

• preferred extension of \( \Gamma \) iff \( S \) is a maximal (w.r.t. set inclusion) admissible set of \( \Gamma \);
• stable extension of \( \Gamma \) iff \( S \) is a conflict–free set of \( \Gamma \) and \( \mathcal{A} \ \backslash \ S = \{ a \in \mathcal{A} \ | \ b \rightarrow a \ and \ b \in S \} \).

### 3. Generation of Portfolios

In this section we describe the techniques we used for combining solvers into sequential portfolios. Every approach requires as input a set of solvers, a set of training \( AFs \), and measures of performance of solvers on the training set. Solvers’ performance are measured in terms of Penalised Average Runtime (PAR) score. This metric trades off coverage and runtime for successfully analysed \( AFs \): runs that do not solve the given problem get ten times the cutoff time (PAR10), other runs get the actual runtime. The PAR10 score of a solver on a set of \( AFs \) is the average of the associated scores. Although PAR10 largely emphasises the coverage, it also gives a clear indication on effective performance, thus resulting in an interesting and useful measure. This is also compatible with the ICCMA experience: ties on coverage are automatically solved on the basis of performance.

#### 3.1. Static Portfolios

Static portfolios—as the name suggests—are generated once, according to the performance of the considered solvers on training instances, and never adjusted. Static portfolios are defined by: (i) the selected solvers; (ii) the order in which solvers will be run, and (iii) the runtime allocated to each solver.

We considered two different approaches for configuring static portfolios. First, we generated static portfolios of exactly \( k \) components, \( Shared-k \). Each component solver has been allocated the same amount of CPU-time, equal to \( maxRuntime/k \) seconds. Solvers are selected and ordered according to overall PAR10 score achieved by the resulting portfolio. We considered values of \( k \) between 2 and 5. In fact, \( k = 1 \) would be equivalent to select the single solver with the best PAR10 score on training instances, which is not relevant for our investigation. For \( k > 5 \), the CPU-time assigned to each solver tends to be too short hence drastically reducing portfolio performance.

For our second static portfolio approach, named \( FDSS \), we adapted the Fast Downward Stone Soup technique [15]. We start from an empty portfolio, and iteratively add
either a new solver component, or extend the allocated CPU-time\(^5\) of a solver already added to the portfolio, depending on what maximises the increment of the PAR10 score of the portfolio. We continue until the time limit of the portfolio has been reached, or it is not possible to further improve the PAR10 score of the portfolio on the training instances.

3.2. Per-instance Portfolios

Per-instance portfolios rely on instance features for configuring an instance-specific portfolio. For each \(AF\) a vector of features is computed; each feature is a real number that summarises a potentially important aspect of the considered \(AF\). Similar instances should have similar feature vectors, and, on this basis, portfolios are configured using empirical performance models [13].

In this investigation we consider the largest set of features available for \(AFs\) [6]. Such set includes 50 features, extracted by exploiting the representation of \(AFs\) both as directed (loss-less) or undirected (lossy) graphs. Features are extracted by considering aspects such as the size of graphs, the presence of connected components, the presence of auto-loops, etc. The features extraction process is usually quick (less than 2 CPU-time seconds on average) and is done by exploiting a wrapper written in Python.

3.2.1. Classification-based approach

The classification-based (hereinafter Classify) approach exploits the technique introduced in [6]. It trains a random decision forest classification model to perform algorithm selection. It classifies a given \(AF\) into a single category which corresponds to the single solver predicted to be the fastest. The difference between solvers’ performance is ignored: all the available CPU-time is then allocated to the selected solver.

3.2.2. Regression-based approaches

For regression-based approaches, deciding which solver to execute and its runtime depends on the empirical hardness models learned from the available training data, in particular a M5-Rules [11] model generated for each solver. When executed on a fresh \(AF\), the predictive model estimates the CPU-time required by each solver to successfully terminate.

We exploit the regression-based model in two different ways. First, for performing algorithm selection (hereinafter 1-Regression): given the predicted runtime of each solver, the solver predicted to be the fastest is selected and it has allocated all the available CPU-time. However, such use of the models do not fully exploit the available predicted runtimes. Therefore, we designed a different way for using the regression-based approach, referred to as M-regression. As in 1-Regression, we initially select the solver predicted to be the fastest, but we allocate only its predicted CPU-time (increased by 10%). If the selected solver is not able to successfully analyse the given \(AF\) in the allocated time, it is stopped and no longer available to be selected, and the process iterates by selecting a different solver. The M-regression approach stops when either a solver has successfully analysed the \(AF\), or the runtime budget has been exhausted.

With regards to existing well-known portfolio-based solver approaches, it is worthy to remark that SATZilla [21] is a regression-based approach similar to the 1-regression

\(^5\) A granularity of 5 CPU-time seconds is considered.
we introduced. However, since it was developed for competition purposes, SATZilla also exploits pre and backup solvers. These are undoubtedly useful for improving coverage, but not when the main point is to evaluate to which extent solvers composition/selection can improve results, as in our investigation.

4. Experimental Analysis of ICCMA-15 Solvers

We randomly generated 2,000 AFs based on four different graph models: Barabasi-Albert [1], Erdős-Rényi [10], Watts-Strogatz [20] and graphs featuring a large number of stable extensions (hereinafter StableM).

Erdős-Rényi graphs [10] are generated by randomly selecting attacks between arguments according to a uniform distribution. While Erdős-Rényi was the predominant model used for randomly generated experiments, [4] investigated also other graph structures such as scale-free and small-world networks. As discussed by Barabasi and Albert [1], a common property of many large networks is that the node connectivities follow a scale-free power-law distribution. This is generally the case when: (i) networks expand continuously by the addition of new nodes, and (ii) new nodes attach preferentially to sites that are already well connected. Moreover, Watts and Strogatz [20] show that many biological, technological and social networks are neither completely regular nor completely random, but something in the between. They thus explored simple models of networks that can be tuned through this middle ground: regular networks rewired to introduce increasing amounts of disorder. These systems can be highly clustered, like regular lattices, yet have small characteristic path lengths, like random graphs, and they are named small-world networks by analogy with the small-world phenomenon. The AFs have been generated by using an improved version of AFBenchGen [7]. It is worthy to emphasise that Watts-Strogatz and Barabasi-Albert produce undirected graphs: in this work, differently from [4], each edge of the undirected graph is then associated with a direction following a probability distribution, that can be provided as input to AFBenchGen. Finally, the fourth set has been generated using the code provided in Probo [8] by the organisers of ICCMA-15.6

In order to identify challenging frameworks—i.e., neither trivial nor too complex to be successfully analysed in the given CPU-time—AFs for each set have been selected using the protocol introduced in the 2014 edition of the International Planning Competition [18]. This protocol lead to the selection of AFs with a number of arguments between 250 and 650, and number of attacks between (approximately) 400 and 180,000.

The set of AFs has been divided into training and testing sets. For each graph model, we randomly selected 200 AFs for training, and the remaining 300 for testing. Therefore, out of the 2,000 AFs generated, 800 have been used for training purposes, while the remaining 1,200 have been used for testing and comparing the performance of trained approaches.

We considered all the solvers that took part in the EE-PR and EE-ST tracks of ICCMA-15 [17], respectively 15 and 11 systems. For the sake of clarity and conciseness, we removed from the analysis single solvers that did not successfully analyse at least one AF or which were always outperformed by another solver. The interested reader

6http://argumentationcompetition.org/2015/results.html
Table 1. PAR10 score and coverage (cov.)—percentage of AFs successfully analysed—of the considered basic solvers for solving the preferred enumeration (upper table) and stable enumeration (lower table) problems on the complete testing set (All) of 1,200 AFs, and on testing sets including AFs generated by specific graph models. Solvers are ordered according to PAR10 on the All testing set. F.t column indicates the number of times a solver has been the fastest among considered. Best results in bold.

<table>
<thead>
<tr>
<th>Solver</th>
<th>PAR10</th>
<th>Cov.</th>
<th>F.t</th>
<th>PAR10</th>
<th>Cov.</th>
<th>PAR10</th>
<th>Cov.</th>
<th>PAR10</th>
<th>Cov.</th>
<th>PAR10</th>
<th>Cov.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cegartix</td>
<td>1350.4</td>
<td>79.1</td>
<td>229</td>
<td>1662.6</td>
<td>74.2</td>
<td>1266.6</td>
<td>81.0</td>
<td>1439.2</td>
<td>77.0</td>
<td>1028.6</td>
<td>84.2</td>
</tr>
<tr>
<td>ArgSemSAT</td>
<td>1916.2</td>
<td>69.1</td>
<td>35</td>
<td>3401.5</td>
<td>51.3</td>
<td>3067.8</td>
<td>51.6</td>
<td>2085.8</td>
<td>58.7</td>
<td>1630.3</td>
<td>74.3</td>
</tr>
<tr>
<td>LabSATSolver</td>
<td>2050.3</td>
<td>66.8</td>
<td>9</td>
<td>3430.7</td>
<td>43.5</td>
<td>261.3</td>
<td>96.5</td>
<td>2869.5</td>
<td>53.0</td>
<td>1657.5</td>
<td>73.9</td>
</tr>
<tr>
<td>prefBarSat</td>
<td>2417.0</td>
<td>61.0</td>
<td>1</td>
<td>3447.8</td>
<td>43.2</td>
<td>1366.7</td>
<td>79.0</td>
<td>2831.8</td>
<td>53.1</td>
<td>2026.0</td>
<td>68.0</td>
</tr>
<tr>
<td>DIAMOND</td>
<td>2728.6</td>
<td>56.1</td>
<td>4</td>
<td>4010.5</td>
<td>32.6</td>
<td>3067.8</td>
<td>51.6</td>
<td>2085.8</td>
<td>58.7</td>
<td>1630.3</td>
<td>74.3</td>
</tr>
<tr>
<td>ASPARTIX-D</td>
<td>2772.2</td>
<td>55.2</td>
<td>21</td>
<td>3646.6</td>
<td>40.3</td>
<td>3292.6</td>
<td>47.1</td>
<td>2340.7</td>
<td>62.0</td>
<td>1772.4</td>
<td>71.9</td>
</tr>
<tr>
<td>CoQuiAas</td>
<td>3026.4</td>
<td>50.5</td>
<td>78</td>
<td>3736.1</td>
<td>38.4</td>
<td>2873.4</td>
<td>53.5</td>
<td>2836.4</td>
<td>53.3</td>
<td>2645.1</td>
<td>57.1</td>
</tr>
<tr>
<td>ASGL</td>
<td>3477.3</td>
<td>43.2</td>
<td>1</td>
<td>4809.7</td>
<td>20.3</td>
<td>96.1</td>
<td>100.0</td>
<td>4475.4</td>
<td>26.0</td>
<td>4585.5</td>
<td>24.4</td>
</tr>
<tr>
<td>Conarg</td>
<td>3696.3</td>
<td>39.3</td>
<td>158</td>
<td>1128.7</td>
<td>81.6</td>
<td>2813.9</td>
<td>55.8</td>
<td>4934.6</td>
<td>18.3</td>
<td>6000.0</td>
<td>0.0</td>
</tr>
<tr>
<td>ArgTools</td>
<td>3906.2</td>
<td>35.2</td>
<td>322</td>
<td>3694.4</td>
<td>39.0</td>
<td>45.2</td>
<td>100.0</td>
<td>6000.0</td>
<td>0.0</td>
<td>6000.0</td>
<td>0.0</td>
</tr>
<tr>
<td>GRIS</td>
<td>4543.7</td>
<td>24.4</td>
<td>174</td>
<td>254.6</td>
<td>96.1</td>
<td>6000.0</td>
<td>100.0</td>
<td>6000.0</td>
<td>100.0</td>
<td>6000.0</td>
<td>100.0</td>
</tr>
</tbody>
</table>

is referred to [16] for detailed descriptions of the solvers. Hereinafter, we will refer to such systems as basic solvers, regardless of the approach they exploit for solving argumentation-related problems.

Experiments have been run on a cluster with computing nodes equipped with 2.5 Ghz Intel Core 2 Quad Processors, 4 GB of RAM and Linux operating system. A cutoff of 600 seconds was imposed to compute the extensions—either preferred or stable—for each AF. For each solver we recorded the overall result: success (if it solved the considered problem), crashed, timed-out or ran out of memory.

In ICCMA, solvers have been evaluated by considering only coverage (in case of ties the overall runtime on solved instances). Here we also evaluate solvers’ performance by considering the PAR10 score.

4.1. Hypothesis 1: Reduction-based Solvers Constantly Outperform Others

Table 1 shows the results of this analysis in terms of coverage, PAR10 scores, and number of instances on which a given solver has been the fastest. We considered runtimes below 1 CPU-time second as equally fast.

Each basic solver for the EE-PR problem has at least one instance on which it is the fastest. We note that, when considering performance achieved on the whole testing set (All) by solvers, there can be a significant discrepancy between results shown in the coverage and fastest columns. One would expect that the higher the coverage, the larger
the possibility of a solver to be the fastest. Interestingly, we observed that some of the
solvers with low coverage tend to be fast on the (few) instances they are able to analyse.
For instance, ArgTools (a non reduction-based system) achieves low overall coverage,
but it is the best solver for handling AFs of the Erdős-Rényi set. This contradicts the
hypothesis—endorsed by ICCMA-15 results—that reduction-based systems constantly
outperform others.

The best basic solver for solving the EE-PR problem on the StableM set of AFs
is Cegartix, which is able to solve 77.0% of the instances. This is approximately 10% more than the coverage of the second best solver on such set, ASPARTIX-D. The
prefMaxSAT solver has shown the best performance on the Watts-Strogatz AFs. From
an (empirical) complexity perspective, we observe that the set with the lowest average
coverage is the Barabasi-Albert set of AFs. This is possibly due to the very large number
(up to few thousands, in some cases) of preferred extensions of such testing frameworks.
Conversely, the Erdős-Rényi set is the less complex for the considered basic solvers
when solving the EE-PR problem. Moreover we can derive that even though there is usu-
ally a basic solver with best coverage performance on each testing set, such solver is not
always the fastest.

As for the EE-ST problem, the results in Table 1 show another interesting scenario.
ArgTools is able to achieve the best PAR10 and coverage performance on two of the four
considered sets, namely StableM and Watts-Strogatz. LabSATSolver obtained the best
PAR10 score on the Erdős-Rényi set, but four of the considered basic solvers successfully
analyse each of the 300 AFs in such a set. The winner of the EE-ST track of ICCMA-15,
ASPARTIX-D, has been the fastest solver on 395 of the testing frameworks, but it did
never excel in any of the 4 considered subsets. It seems that the AFs of the StableM set
are (empirically) the most complex to solve for the considered systems.

4.2. Hypothesis 2: Basic Solvers Show Complementary Performance

Table 1 indicate that there is not a basic solver that is always the best selection on the
vast majority of the testing frameworks. This is evidence that the basic solvers are sub-
stantially complementary, thus supporting the claim that a mixture of approaches can be
fruitful, and justifying the search for improvements via portfolios.

5. Experimental Analysis of Portfolios

First of all, we generated the Virtual Best Solver (VBS) as the (virtual) oracle which
always select the best solver (as to PAR10) for the given framework and problem. This
provides the upper bound of performance achievable by combining considered solvers.

For the preferred semantics, the solvers included in the Shared-5 portfolio, or-
dered following their execution order, are: Cegartix, ArgSemSAT, prefMaxSAT,
LabSATSolver and DIAMOND. Smaller static portfolios include subsets of those 5
solvers, not necessarily in that order. FDSS static portfolio includes ArgSemSAT and
GRIS, only.

For the stable semantics, the solvers included in the Shared-5 portfolio, or-
dered following their execution order, are: LabSATSolver, ArgTools, ASPARTIX-D,
CoQuiAas and DIAMOND. Smaller portfolios include subsets of the listed solvers, not
necessarily in that order. The FDSS portfolio includes LabSATSolver and ASPARTIX-D.
We also generated the three per-instance (per-problem) portfolios that exploit predictive models in order to map the features of the given AF to a solver selection or combination: Classify, 1-Regression, and M-Regression. Classify and 1-Regression select a single solver by relying, respectively, on classification and regression techniques. M-regression iteratively selects the next solver to run, and allocates its CPU-time, by considering the predicted runtime of the available solvers for the given framework and problem, increased by 10% in order to mitigate the impact of negligible prediction mistakes.

We trained all the portfolio approaches using our training set of 800 AFs, 200 AFs from each set. The runtime cutoff once again was 600 CPU-time seconds. Table 2 shows the coverage and PAR10 scores of all portfolios, basic solvers and the VBS on the testing frameworks.

5.1. Hypothesis 3: Static Portfolios are more Efficient than Basic Solvers

Results for the static portfolios vary between stable and preferred semantics. When dealing with the EE-PR problem, the FDSS approach is the only technique which is able to outperform the best basic solver. Shared-2 and Shared-3 achieve performance close to those of the best basic solver, while Shared-4 and Shared-5 are undistinguishable from average basic solvers. FDSS portfolio performs better than Shared-k static portfolios because it includes GRIS. ArgSemSAT has good coverage, and GRIS excels on the
Table 3. Number of times each solver has been selected by the Classify (Class.) or M-Regression (M-Reg.) approaches for solving EE-PR (left part) and EE-ST (right part) problems on the testing frameworks. Basic solvers are alphabetically ordered. Highest numbers in bold. Empty cells indicate that the corresponding solver is not able to handle the considered problem.

<table>
<thead>
<tr>
<th>System</th>
<th>EE-PR</th>
<th></th>
<th></th>
<th>EE-ST</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ArgSemSAT</td>
<td>0</td>
<td>253</td>
<td>0</td>
<td>212</td>
<td></td>
</tr>
<tr>
<td>ArgTools</td>
<td>311</td>
<td>305</td>
<td>138</td>
<td>428</td>
<td></td>
</tr>
<tr>
<td>ANGL</td>
<td>6</td>
<td>36</td>
<td>0</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>ASPARTIX-D</td>
<td>2</td>
<td>80</td>
<td></td>
<td>305</td>
<td>409</td>
</tr>
<tr>
<td>ASPARTIX-V</td>
<td>1</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cegartix</td>
<td>221</td>
<td>403</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conarg</td>
<td>157</td>
<td>122</td>
<td>231</td>
<td>337</td>
<td></td>
</tr>
<tr>
<td>CoQuiAas</td>
<td>43</td>
<td>44</td>
<td>288</td>
<td>193</td>
<td></td>
</tr>
<tr>
<td>DIAMOND</td>
<td>0</td>
<td>65</td>
<td>33</td>
<td>138</td>
<td></td>
</tr>
<tr>
<td>GRIS</td>
<td>153</td>
<td>278</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LabSATSolver</td>
<td>13</td>
<td>208</td>
<td>228</td>
<td>548</td>
<td></td>
</tr>
<tr>
<td>PrefMaxSAT</td>
<td>297</td>
<td>301</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Barabasi-Albert set (Table 1), while Shared-k portfolios do not include any solver able to efficiently solve the EE-PR problem on the Barabasi-Albert set.

Conversely, the right part of Table 2 shows that on the EE-ST problem, both Shared-2 and Shared-3 are able to achieve better performance than any basic solver, and the FDSS portfolio. Shared portfolios performance are boosted by the inclusion of ArgTools, which is able to achieve the best performance on three of the considered benchmark set structures, and CoQuiAas—that is the second best basic solver in terms of number of AFs quickly analysed. Moreover, the EE-ST problems are usually quickly solved by the basic solvers, therefore 2 or 3 solvers can be easily executed within the 600 CPU-time seconds limit. When more than three solvers are combined by the Shared approach—i.e. the CPU-time allocated to each basic solver is less than 200 seconds—performance drops.

5.2. Hypothesis 4: Per-Instance Portfolios are more Efficient than Static Portfolios

When considering per-instance portfolios, Table 2 indicates that they are all able to outperform the best basic solver on the considered testing frameworks. This comes as no surprise, since per-instance approaches should be able to select the most promising—ideally, the fastest—algorithm for solving the considered problem on the given AF. For both EE-PR and EE-ST problems, the performances of Classify and 1-Regression are very similar, but the M-Regression approach performance is always worse. Such results indicate that: (i) the 50 features considered are informative for both EE-PR and EE-ST problems, and allow to effectively select solvers; (ii) classification and regression predictive models have similar performance when used for selecting a single solver to run; and (iii) the regression predictive model tends to underestimate the CPU-time needed by algorithms for solving the considered problem on the given AF.

Table 3 shows the number of times each basic solver has been executed by either the Classify or the M-Regression portfolio. 1-Regression executed solvers are not shown,
Table 4. Coverage (Cov.) and PAR10 of the systems considered in this study on the complete testing set, when trained on a training set not containing AFs of that structure (leave-one-set-out scenario). Systems are ordered according to results shown in Table 2. Best results in bold.

<table>
<thead>
<tr>
<th>System</th>
<th>Barabasi-Albert</th>
<th>Erdos-Renyi</th>
<th>StableM</th>
<th>Watts-Strogatz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cov.</td>
<td>PAR10</td>
<td>Cov.</td>
<td>PAR10</td>
</tr>
<tr>
<td>Classify</td>
<td>78.9</td>
<td>1321.4</td>
<td>88.6</td>
<td>745.0</td>
</tr>
<tr>
<td>M-Regression</td>
<td>70.4</td>
<td>1828.4</td>
<td>67.3</td>
<td>2039.7</td>
</tr>
<tr>
<td>FDSS</td>
<td>69.1</td>
<td>1916.2</td>
<td>80.9</td>
<td>1245.5</td>
</tr>
<tr>
<td>Shared-2</td>
<td>75.2</td>
<td>1678.0</td>
<td>73.2</td>
<td>1678.0</td>
</tr>
<tr>
<td>Shared-3</td>
<td>69.4</td>
<td>1892.0</td>
<td>67.3</td>
<td>2007.9</td>
</tr>
<tr>
<td>Shared-4</td>
<td>65.7</td>
<td>2106.2</td>
<td>65.7</td>
<td>2101.1</td>
</tr>
<tr>
<td>Shared-5</td>
<td>63.3</td>
<td>2240.9</td>
<td>63.4</td>
<td>2235.8</td>
</tr>
</tbody>
</table>

because they are a subset of the M-regression selections. Table 3 shows some remarkable differences in the algorithm selected by the classification and regression approaches, and also those included in the static portfolios. For instance, Classify never selects ArgSemSAT, while it is largely exploited by M-regression, and included in static portfolios generated for solving EE-PR problems. This is because ArgSemSAT, and a few other basic solvers, has rarely been the fastest: therefore the classification approach—which only focuses on the best solver—ignores its performance. On the contrary, solvers like ArgTools (EE-PR) and ASPARTIX-D (EE-ST) are usually the fastest, and are often selected by both Classify and M-Regression approaches.

Finally, by looking at Table 2, it can be noted that the largest performance improvement can be achieved when exploiting portfolio approaches for solving the problem of enumerating preferred extensions of an AF: the use of portfolio-based techniques allows to solve up to 10.6% more instances than the best basic solver, Cegartix. Such margin is reduced to 2.9% when solving the EE-ST problem. This is due to the higher empirical complexity of the EE-PR problem, and to the higher complementarity between basic solvers able to handle the EE-PR problem.
5.3. Post-Hoc Analysis: Generalisation of Performance

To assess the ability of our portfolios on testing instances that are dissimilar from instances used for training we generated four different new training sets as follows: starting by the original training set composed by 800 AFs, we removed all the frameworks corresponding to one set at a time, and randomly oversampled frameworks from the remaining three sets—in order to have again approximately 800 frameworks for training. We then tested our portfolios on the complete testing set of 1,200 AFs, so that performance can be compared with those of basic solvers (Table 2). This can be seen as a leave-one-out scenario. The results of such generalisation analysis are shown in Table 4.

Unsurprisingly, static portfolios—particularly Shared-k—show the best generalisation performance: their behaviour does not change much with the new training sets. On the other hand, per-instance approaches do not show good generalisation capabilities: their performance varies significantly when the training set is not fully representative of the testing instances. This is true for both EE-PR and EE-ST problems, despite the fact that gaps are smaller in the EE-ST case, although it is true that also the performance of basic solvers on EE-ST tends to be closer.

Remarkably, Classify (covering up to 89.7%, cf. Table 2) is very sensible to the absence of Barabasi-Albert (−10.8%, cf. Table 4) or StableM (−15.3%, cf. Table 4) frameworks from the training set for EE-PR, while regression-based approaches show scarce generalisation abilities when the Erdös-Rényi frameworks are removed from the training set. On the contrary, Classify is very generalisable on the EE-ST set, and the 1-Regression method is very sensitive when Watts-Strogatz AFs are removed. M-Regression is more generalisable than 1-Regression when dealing with the EE-ST problem: this indicates that when testing instances are dissimilar from training ones, the exploitation of more than one solver can be fruitful.

6. Conclusion

We exploit the ICCMA-15 legacy by combining state-of-the-art solvers, able to handle EE-PR and EE-ST problems, using—for the first time in this research area—portfolio-based techniques. In particular, we tested static and per-instance portfolios, exploiting the largest available set of argumentation features [6]. We remark this is the first comprehensive experimental analysis on the performance of different portfolio-based methods, in the argumentation area.

The results of our extensive empirical analysis showed that: (i) the claim that reduction-based solvers always outperform non-reduction-based systems—one of the takeaway messages from ICCMA-15—is not always the case; (ii) the solvers at the state of the art show a high level of complementarity (specially those able to deal with EE-PR problems), thus they are suitable to be combined in portfolios; (iii) portfolio systems generally outperform basic solvers; (iv) if the training instances are representative of testing AFs, the existing set of features is informative for selecting most suitable solvers; (v) classification-based portfolios show good generalisation performance; (vi) static portfolios are usually the approaches which are less sensitive to different training sets.

As part of future research, we are interested in further investigating the generalisation capabilities of portfolios performance by considering significantly differently-
structured AFs, including complex frameworks generated by real-world scenarios. We will also extend the portfolio methods considering SATZilla [21] like approaches, or more sophisticated model-based techniques. Finally, we are interested in testing portfolio methods also in other complex argumentation problems.

Acknowledgements

The authors would like to acknowledge the use of the University of Huddersfield Queensgate Grid in carrying out this work.

References