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

Federico CERUTTI [a,1], Mauro VALLATI [b] and Massimiliano GIACOMIN [c]
[a] *Cardiff University, UK*
[b] *University of Huddersfield, UK*
[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 [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 [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

[1]Corresponding Author: Federico Cerutti, Cardiff University, School of Computer Science & Informatics, CF24 3AA, Cardiff, UK; E-mail: CeruttiF@cardiff.ac.uk.

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.

2. Dung's Argumentation Framework

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

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

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

Definition 2. *Given an AF* $\Gamma = \langle \mathscr{A}, \mathscr{R} \rangle$*:*

[2]`http://www.journals.elsevier.com/artificial-intelligence/most-cited-articles`, accessed on 10th June 2016.

[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 $S \subseteq \mathscr{A}$ is a* conflict–free *set of* Γ *if* $\nexists\, \boldsymbol{a}, \boldsymbol{b} \in S$ *s.t.* $\boldsymbol{a} \to \boldsymbol{b}$*;*
- *an argument* $\boldsymbol{a} \in \mathscr{A}$ *is* acceptable *with respect to a set* $S \subseteq \mathscr{A}$ *of* Γ *if* $\forall \boldsymbol{b} \in \mathscr{A}$ *s.t.* $\boldsymbol{b} \to \boldsymbol{a}$, $\exists\, \boldsymbol{c} \in S$ *s.t.* $\boldsymbol{c} \to \boldsymbol{b}$*;*
- *a set* $S \subseteq \mathscr{A}$ *is an* admissible set *of* Γ *if S is a conflict–free set of* Γ *and every element of S is acceptable with respect to S of* Γ.

An argumentation semantics σ prescribes for any *AF* Γ a set of *extensions*, namely a set of sets of arguments satisfying the conditions dictated by σ.

Definition 3. *Given an AF* $\Gamma = \langle \mathscr{A}, \mathscr{R} \rangle$*: a set* $S \subseteq \mathscr{A}$ *is a:*

- preferred extension *of* Γ *iff S is a maximal (w.r.t. set inclusion) admissible set of* Γ*;*
- stable extension *of* Γ *iff S is a conflict–free set of* Γ *and* $\mathscr{A} \setminus S = \{\boldsymbol{a} \in \mathscr{A} \mid \boldsymbol{b} \to \boldsymbol{a}$ *and* $\boldsymbol{b} \in S\}$.

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 *AF*s, 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 *AF*s: 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 *AF*s 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 *AF*s [6]. Such set includes 50 features, extracted by exploiting the representation of *AF*s 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 *AF*s based on four different graph models: Barabasi-Albert [1], Erdös-Rényi [10], Watts-Strogatz [20] and graphs featuring a large number of stable extensions (hereinafter StableM).

Erdös-Rényi graphs [10] are generated by randomly selecting attacks between arguments according to a uniform distribution. While Erdös-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 *AF*s 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—*AF*s 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 *AF*s with a number of arguments between 250 and 650, and number of attacks between (approximately) 400 and 180,000.

The set of *AF*s has been divided into training and testing sets. For each graph model, we randomly selected 200 *AF*s for training, and the remaining 300 for testing. Therefore, out of the 2,000 *AF*s 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

[6]http://argumentationcompetition.org/2015/results.html

Table 1. PAR10 score and coverage (cov.)—percentage of *AF*s 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 *AF*s, and on testing sets including *AF*s 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.

EE-PR

	All			**Barabasi-Albert**		**Erdös-Rényi**		**StableM**		**Watts-Strogatz**	
Solver	PAR10	Cov.	F.t	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.
Cegartix	**1350.4**	**79.1**	229	1662.6	74.2	1266.6	81.0	**1439.2**	**77.0**	1028.6	84.2
ArgSemSAT	1916.2	69.1	35	3532.3	41.9	433.7	94.2	2530.9	58.7	1171.1	81.5
LabSATSolver	2050.3	66.8	9	3430.7	43.5	261.3	96.5	2869.5	53.0	1657.5	73.9
prefMaxSAT	2057.2	66.8	273	3482.1	42.9	444.0	94.2	3625.2	40.3	**697.5**	**89.4**
DIAMOND	2417.0	61.0	1	3447.8	43.2	1366.7	79.0	2831.8	53.7	2026.0	68.0
ASPARTIX-D	2728.6	56.1	4	4101.5	32.6	3067.8	51.6	2068.8	66.7	1630.3	74.3
ASPARTIX-V	2772.2	55.2	21	3646.6	40.3	3292.6	47.1	2340.7	62.0	1772.4	71.9
CoQuiAas	3026.4	50.5	78	3736.1	38.4	2873.4	53.5	2836.4	53.3	2645.1	57.1
ASGL	3477.3	43.2	1	4809.7	20.3	96.1	**100.0**	4475.4	26.0	4585.5	25.4
Conarg	3696.3	39.3	158	1128.7	81.6	2813.9	55.8	4934.6	18.3	6000.0	0.0
ArgTools	3906.2	35.2	**322**	3694.4	39.0	**45.2**	**100.0**	6000.0	0.0	6000.0	0.0
GRIS	4543.7	24.4	174	**254.6**	**96.1**	6000.0	0.0	6000.0	0.0	6000.0	0.0

EE-ST

	All			**Barabasi-Albert**		**Erdös-Rényi**		**StableM**		**Watts-Strogatz**	
Solver	PAR10	Cov.	F.t	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.
ArgTools	440.7	94.5	245	1328.6	78.4	47.4	**100.0**	**144.1**	**100.0**	**230.5**	**100.0**
LabSATSolver	641.6	90.0	352	396.2	93.9	**22.7**	**100.0**	1497.6	76.0	684.9	90.7
ASPARTIX-D	829.7	87.1	**395**	412.2	93.5	1194.4	81.6	1187.2	81.0	535.0	93.0
CoQuiAas	1477.2	76.2	372	1453.3	76.5	1485.1	76.5	1879.0	69.3	1106.5	83.3
DIAMOND	1555.4	75.2	42	2527.1	58.7	692.2	89.7	1887.2	69.7	1127.1	83.7
ArgSemSAT	1826.6	70.5	70	4019.0	33.5	408.9	94.5	1970.0	68.0	900.8	87.0
Conarg	1976.4	67.8	292	**261.4**	**96.1**	33.6	**100.0**	3742.1	38.3	4010.0	35.3
ASGL	2647.6	57.3	11	2737.4	56.1	85.2	**100.0**	3723.8	38.7	4152.8	33.7

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 *AF*s of the Erdös-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 *AF*s 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 *AF*s. From an (empirical) complexity perspective, we observe that the set with the lowest average coverage is the Barabasi-Albert set of *AF*s. 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 Erdös-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 usually 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 Erdös-Rényi set, but four of the considered *basic solvers* successfully analyse each of the 300 *AF*s 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 *AF*s 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 substantially 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, ordered 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, ordered 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**.

Table 2. Coverage (Cov.) and PAR10 of the systems considered in this study for solving the EE-PR problem (left part) and the EE-ST problem (right part) on the complete set of 1,200 testing *AF*s. VBS indicates the performance of the virtual best solver. Systems are ordered according to PAR10.

EE-PR			**EE-ST**		
System	**Cov.**	**PAR10**	**System**	**Cov.**	**PAR10**
VBS	91.4	562.9	*VBS*	100.0	39.3
Classify	89.7	665.2	*1-Regression*	97.4	206.9
1-Regression	88.6	734.7	*Classify*	97.1	217.5
M-Regression	82.8	1068.3	*Shared-2*	97.7	262.3
FDSS	80.0	1311.4	*M-Regression*	94.7	378.4
Cegartix	79.1	1350.4	*Shared-3*	94.0	420.1
Shared-2	73.2	1678.0	**ArgTools**	94.5	440.7
Shared-3	69.4	1892.0	**LabSATSolver**	90.0	641.6
ArgSemSAT	69.1	1916.2	*FDSS*	89.4	677.4
LabSATSolver	66.8	2050.3	**ASPARTIX-D**	87.1	829.7
prefMaxSAT	66.8	2057.2	*Shared-5*	86.3	867.4
Shared-4	65.7	2105.5	*Shared-4*	86.0	873.8
Shared-5	63.3	2240.3	**CoQuiAas**	76.2	1477.2
DIAMOND	61.0	2417.0	**DIAMOND**	75.2	1555.4
ASPARTIX-D	56.1	2728.6	**ArgSemSAT**	70.5	1826.6
ASPARTIX-V	55.2	2772.2	**Conarg**	67.8	1976.4
CoQuiAas	50.5	3026.4	**ASGL**	57.3	2647.6
ASGL	43.2	3477.3			
Conarg	39.3	3696.3			
ArgTools	35.2	3906.2			
GRIS	24.4	4543.7			

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 *AF* s, 200 *AF*s 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 outperfom 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.

	EE-PR		**EE-ST**	
System	Class.	M-Reg.	Class.	M-Reg.
ArgSemSAT	0	253	0	212
ArgTools	**311**	305	138	428
ASGL	6	36	0	35
ASPARTIX-D	2	80	**305**	409
ASPARTIX-V	1	99		
Cegartix	221	**403**		
Conarg	157	122	231	337
CoQuiAas	43	44	288	193
DIAMOND	0	65	33	138
GRIS	153	278		
LabSATSolver	13	208	228	**548**
prefMaxSAT	297	301		

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 *AF*s 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 outperfom 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 *AF*s of that structure (leave-one-set-out scenario). Systems are ordered according to results shown in Table 2. Best results in bold.

EE-PR

	Barabasi-Albert		**Erdös-Rényi**		**StableM**		**Watts-Strogatz**	
System	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10
Classify	**78.9**	**1321.4**	**88.6**	**745.0**	74.4	1574.3	**89.5**	**677.8**
1-Regression	76.3	1479.0	63.0	2255.2	76.5	1453.9	83.0	1079.9
M-Regression	70.4	1828.4	67.3	2039.7	77.0	1434.7	79.6	1267.6
FDSS	69.1	1916.2	80.9	1245.5	**79.1**	**1341.9**	78.6	1380.0
Shared-2	73.2	1678.0	73.2	1678.0	74.2	1620.4	73.2	1678.0
Shared-3	69.4	1892.0	67.3	2007.9	69.5	1896.7	69.4	1892.0
Shared-4	65.7	2106.2	65.7	2101.1	65.7	2108.1	65.7	2103.9
Shared-5	63.3	2240.9	63.4	2235.8	63.3	2242.9	63.3	2242.9

EE-ST

	Barabasi-Albert		**Erdös-Rényi**		**StableM**		**Watts-Strogatz**	
System	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10	Cov.	PAR10
1-Regression	88.6	756.9	92.6	508.7	**98.6**	**149.9**	81.6	1153.0
Classify	93.0	470.4	92.4	519.6	91.2	575.6	93.4	439.3
Shared-2	**97.7**	**262.3**	97.3	285.2	97.7	220.9	**97.7**	**262.3**
M-Regression	96.2	297.4	**96.4**	**282.2**	95.6	334.9	90.3	636.5
Shared-3	94.0	420.1	94.0	435.5	94.0	420.1	94.0	476.6
FDSS	89.4	677.4	87.1	829.7	89.4	677.4	88.7	714.7
Shared-4	85.9	878.2	86.0	887.5	86.0	873.8	86.8	833.8
Shared-5	86.3	867.4	86.3	870.8	86.3	862.3	84.3	973.4

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 *AF*s, 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 *AF*s, 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 scarse 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 *AF*s 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 message 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 *AF*s, 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 differentlystructured *AF*s, 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

[1] A. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):11, 1999.

[2] P. Baroni, M. Caminada, and M. Giacomin. An introduction to argumentation semantics. *Knowl. Eng. Rev.*, 26(4):365–410, 2011.

[3] P. Baroni, F. Cerutti, P. E. Dunne, and M. Giacomin. Automata for Infinite Argumentation Structures. *Artif. Intell.*, 203(0):104–150, 2013.

[4] S. Bistarelli, F. Rossi, and F. Santini. Benchmarking Hard Problems in Random Abstract AFs: The Stable Semantics. In *Proc. of COMMA*, pages 153–160, 2014.

[5] R. Brochenin, T. Linsbichler, M. Maratea, J. P. Wallner, and S. Woltran. *TAFA 2015*, chapter Abstract Solvers for Dung's Argumentation Frameworks, pages 40–58. 2015.

[6] F. Cerutti, M. Giacomin, and M. Vallati. Algorithm selection for preferred extensions enumeration. In *Proc. of COMMA*, pages 221–232, 2014.

[7] F. Cerutti, M. Giacomin, and M. Vallati. Generating challenging benchmark AFs. In *Proc. of COMMA*, pages 457–458, 2014.

[8] F. Cerutti, N. Oren, H. Strass, M. Thimm, and M. Vallati. A benchmark framework for a computational argumentation competition. In *Proc. of COMMA*, pages 459–460, 2014.

[9] 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.

[10] P. Erdös and A. Rényi. On random graphs. I. *Publ. Math-Debrecen*, 6:290–297, 1959.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explor.*, 11(1):10–18, 2009.

[12] H. Hoos, M. Lindauer, and T. Schaub. claspfolio 2: Advances in algorithm selection for answer set programming. *Theor. Pract. Log. Prog.*, 14(Special Issue 4-5):569–585, 2014.

[13] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods & evaluation. *Artif. Intell.*, 206:79–111, 2014.

[14] J. R. Rice. The algorithm selection problem. *Adv. Comput.*, 15:65–118, 1976.

[15] J. Seipp, M. Braun, J. Garimort, and M. Helmert. Learning portfolios of automatically tuned planners. In *Proc. of ICAPS*, pages 369–372, 2012.

[16] M. Thimm and S. Villata. System descriptions of the first international competition on computational models of argumentation (ICCMA'15). *arXiv preprint arXiv:1510.05373*, 2015.

[17] M. Thimm, S. Villata, F. Cerutti, N. Oren, H. Strass, and M. Vallati. Summary Report of The First International Competition on Computational Models of Argumentation. *AI Mag.*, 2016.

[18] M. Vallati, L. Chrpa, M. Grzes, T. L. McCluskey, M. Roberts, and S. Sanner. The 2014 international planning competition: Progress and trends. *AI Mag.*, 2015.

[19] M. Vallati, L. Chrpa, and D. E Kitchin. Portfolio-based planning: State of the art, common practice and open challenges. *AI Commun.*, 2015.

[20] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[21] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *J. Artif. Intell. Res.*, pages 565–606, 2008.