The Seventh Answer Set Programming Competition: Design and Results

MARTIN GEBSER

Institute for Computer Science, University of Potsdam, Germany (e-mail: gebser@cs.uni-potsdam.de)

MARCO MARATEA

DIBRIS, University of Genova, Italy (e-mail: marco@dibris.unige.it)

FRANCESCO RICCA

Dipartimento di Matematica e Informatica, Università della Calabria, Italy (e-mail: ricca@mat.unical.it)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

Answer Set Programming (ASP) is a prominent knowledge representation language with roots in logic programming and non-monotonic reasoning. Biennial ASP competitions are organized in order to furnish challenging benchmark collections and assess the advancement of the state of the art in ASP solving. In this paper, we report on the design and results of the Seventh ASP Competition, jointly organized by the University of Calabria (Italy), the University of Genova (Italy), and the University of Potsdam (Germany), in affiliation with the 14th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR 2017). (Under consideration foracceptance in TPLP).

KEYWORDS: Answer Set Programming; Competition

1 Introduction

Answer Set Programming (ASP) is a prominent knowledge representation language with roots in logic programming and non-monotonic reasoning (Baral 2003; Brewka et al. 2011; Eiter et al. 2009; Gelfond and Leone 2002; Lifschitz 2002; Marek and Truszczyński 1999; Niemelä 1999). The goal of the ASP Competition series is to promote advancements in ASP methods, collect challenging benchmarks, and assess the state of the art in ASP solving (see, e.g., (Alviano et al. 2015; Alviano et al. 2017; Bruynooghe et al. 2015; Gebser et al. 2015; Lefèvre et al. 2017; Maratea et al. 2015; Marple and Gupta 2014; Calimeri et al. 2017) for recent ASP systems, and (Gebser et al. 2018) for a recent survey). Following a nowadays customary practice of publishing results of AI-based competitions in archival journals, where they are expected to remain available and can be used as references, the results of ASP competitions have been hosted in prominent journals of the area (see, (Calimeri et al. 2014; Calimeri et al. 2016; Gebser et al. 2017b)). Continuing the tradition, this paper reports on the design and results of the Seventh ASP Com-

M. Gebser, M. Maratea and F. Ricca

petition,¹ which was jointly organized by the University of Calabria (Italy), the University of Genova (Italy), and the University of Potsdam (Germany), in affiliation with the 14th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR 2017).²

The Seventh ASP Competition is conceived along the lines of the System track of previous competition editions (Calimeri et al. 2016; Lierler et al. 2016; Gebser et al. 2016; Gebser et al. 2017b), with the following characteristics: (*i*) benchmarks adhere to the ASP-Core-2 standard modeling language,³ (*ii*) sub-tracks are based on language features utilized in problem encodings (e.g., aggregates, choice or disjunctive rules, queries, and weak constraints), and (*iii*) problem instances are classified and selected according to their expected hardness. Both single and multiprocessor categories are available in the competition, where solvers in the first category run on a single CPU (core), while they can take advantage of multiple processors (cores) in the second category. In addition to the basic competition design, which has also been addressed in a preliminary version of this report (Gebser et al. 2017a), we detail the revised benchmark selection process as well as the results of the event, which were orally presented during LPNMR 2017 in Hanasaari, Espoo, Finland.

The rest of this paper is organized as follows. Section 2 introduces the format of the Seventh ASP Competition. In Section 3, we describe new problem domains contributed to this competition edition as well as the revised benchmark selection process for picking instances to run in the competition. The participant systems of the competition are then surveyed in Section 4. In Section 5, we then present the results of the Seventh ASP Competition along with the winning systems of competition categories. Section 6 concludes the paper with final remarks.

2 Competition Format

This section gives an overview of competition categories, sub-tracks, and scoring scheme(s), which are similar to the previous ASP Competition edition. One addition though concerns the system ranking of Optimization problems, where a ranking by the number of instances solved "optimally" complements the relative scoring scheme based on solution quality used previously.

Categories. The competition includes two categories, depending on the computational resources provided to participant systems: **SP**, where one processor (core) is available, and **MP**, where multiple processors (cores) can be utilized. While the **SP** category aims at sequential solving systems, **MP** allows for exploiting parallelism.

Sub-tracks. Both categories are structured into the following four sub-tracks, based on the ASP-Core-2 language features utilized in problem encodings:

- **Sub-track #1** (*Basic Decision*): Encodings consisting of non-disjunctive and non-choice rules (also called normal rules) with classical and built-in atoms only.
- **Sub-track #2** (*Advanced Decision*): Encodings exploiting the language fragment allowing for aggregates, choice as well as disjunctive rules, and queries, yet excepting weak constraints and non-head-cycle-free (non-HCF) disjunction.

¹ http://aspcomp2017.dibris.unige.it ² http://lpnmr2017.aalto.fi

³ http://www.mat.unical.it/aspcomp2013/ASPStandardization/



(c) Basic Decision encoding of Hamiltonian cycles.

```
1 cycle(X,Y) :- edge(X,Y), edge(X,Z), Y != Z, not cycle(X,Z).
2 reach(Y) :- cycle(1,Y).
3 reach(Y) :- cycle(X,Y), reach(X).
4 :- node(Y), not reach(Y).
```

(d) Advanced Decision encoding of Hamiltonian cycles.

1 {cycle(X,Y) : edge(X,Y) } = 1 :- node(X).
2 reach(Y) :- cycle(1,Y).
3 reach(Y) :- cycle(X,Y), reach(X).
4 :- node(Y), not reach(Y).

(e) Unrestricted encoding of Hamiltonian cycles.

(f) Weak constraint for Hamiltonian cycle optimization.

1 :~ cycle(X,Y), cost(X,Y,C). [C,X,Y]

Fig. 1: An example graph with edge costs, its fact representation, and corresponding encodings.

- **Sub-track #3** (*Optimization*): Encodings extending the aforementioned language fragment by weak constraints, while still excepting non-HCF disjunction.
- **Sub-track #4** (*Unrestricted*): Encodings exploiting the full language and, in particular, non-HCF disjunction.

A problem domain, i.e., an encoding together with a collection of instances, belongs to the first sub-track its problem encoding is compatible with.

Example 1

To illustrate the sub-tracks and respective language features, consider the directed graph displayed in Figure 1(a) and the corresponding fact representation given in Figure 1(b). Facts over the predicate node/1 specify the nodes of the graph, those over edge/2 provide the edges, and cost/3 associates each edge with its cost. The idea in the following is to encode the well-known Traveling Salesperson problem, which is about finding a Hamiltonian cycle, i.e., a round trip visiting each node exactly once, such that the sum of edge costs is minimal. Note that the example graph in Figure 1(a) includes precisely two outgoing edges per node, and for simplicity the encodings in Figures 1(c)–(e) build on this property, while accommodating an arbitrary number of outgoing edges would also be possible with appropriate modifications.

The first encoding in Figure 1(c) complies with the language fragment of Sub-track #1, as it does not make use of aggregates, choice or disjunctive rules, queries, and weak constraints. Note that terms starting with an uppercase letter, such as X, Y, and Z, stand for universally quantified first-order variables, Y = Z is a built-in atom, and not denotes the (default) negation connective. Given this, the rule in line 1 expresses that exactly one of the two outgoing edges per node must belong to a Hamiltonian cycle, represented by atoms over the predicate cycle/2 within a stable model (Lifschitz 2008). Starting from the distinguished node 1, the least fixpoint of the rules in lines 2 and 3 provides the nodes reachable from 1 via the edges of a putative Hamiltonian cycle. The so-called integrity constraint, i.e., a rule with an empty head that is interpreted as false, in line 4 then asserts that all nodes must be reachable from the starting node 1, which guarantees that stable models coincide with Hamiltonian cycles. While edge costs are not considered so far, the encoding in Figure 1(c) can be used to decide whether a Hamiltonian cycle exists for a given graph (with precisely two outgoing edges per node).

The second encoding in Figure 1(d) includes a choice rule in line 1, thus making use of language features permitted in Sub-track #2, but incompatible with Sub-track #1. The instance of this choice rule obtained for the node 1, {cycle(1,2); cycle(1,4)} = 1., again expresses that exactly one outgoing edge of node 1 must be included in a Hamiltonian cycle, and respective rule instances apply to the other nodes of the example graph in Figure 1(a). Notably, the choice rule adapts to an arbitrary number of outgoing edges, and the assumption that there are precisely two per node could be dropped when using the encoding in Figure 1(d).

The rules in lines 1 and 2 of the third encoding in Figure 1(e) are disjunctive, and rule instances as follows are obtained together with line 3:

```
cycle(1,2) | cycle(1,4).
cycle(2,1) | cycle(2,3) :- reach(2).
cycle(3,2) | cycle(3,4) :- reach(3).
cycle(4,1) | cycle(4,3) :- reach(4).
reach(1) :- cycle(2,1). reach(3) :- cycle(2,3).
reach(1) :- cycle(4,1). reach(3) :- cycle(4,3).
reach(2) :- cycle(1,2). reach(2) :- cycle(3,2).
reach(4) :- cycle(1,4). reach(4) :- cycle(3,4).
```

Observe that reach(3) occurs in the body of a disjunctive rule with cycle(3,2) and cycle(3,4) in the head. These atoms further imply reach(2) or reach(4), respectively, which lead on two disjunctive rules, one containing cycle(2,3) in the head and the other cycle(4,3). As the latter two atoms also occur in the body of rules with reach(3) in the head, we have that all of the mentioned atoms recursively depend on each other. Since cycle(3,2)

and cycle(3,4) jointly constitute the head of a disjunctive rule, this means that rule instances obtained from the encoding in Figure 1(e) are non-HCF (Ben-Eliyahu and Dechter 1994) and thus fall into a syntactic class of logic programs able to express problems at the second level of the polynomial hierarchy (Eiter and Gottlob 1995). Hence, the encoding in Figure 1(e) makes use of a language feature permitted in Sub-track #4 only.

Given that either of the encodings in Figures 1(c)-(e) yields stable models corresponding to Hamiltonian cycles, the weak constraint in Figure 1(f) can be added to each of them to express the objective of finding a Hamiltonian cycle whose sum of edge costs is minimal. In case of the encodings in Figures 1(c) and 1(d), the addition of the weak constraint leads to a reclassification into Sub-track #3, since the focus is shifted from a Decision to an Optimization problem. For the encoding in Figure 1(e), Sub-track #4 still matches when adding the weak constraint, as non-HCF disjunction is excluded in the other sub-tracks.

Scoring Scheme. The applied scoring schemes are based on the following considerations:

- All domains are weighted equally.
- If a system outputs an incorrect answer to some instance in a domain, this invalidates its score for the domain, even if other instances are solved correctly.

In general, 100 points can be earned in each problem domain. The total score of a system is the sum of points over all domains.

For *Decision* problems and *Query answering* tasks, the score S(D) of a system S in a domain D featuring N instances is calculated as

$$S(D) = \frac{N_S * 100}{N}$$

where N_S is the number of instances successfully solved within the time and memory limits of 20 minutes wall-clock time and 12GB RAM per run.

For *Optimization* problems, we employ two alternative scoring schemes. The first one, which has also been used in the previous competition edition, performs a relative ranking of systems by solution quality, following the approach of the MANCOOSI International Solver Competition.⁴ Given M participant systems, the score S(D, I) of a system S for an instance I in a domain D featuring N instances is calculated as

$$S(D,I) = \frac{M_S(I) * 100}{M * N}$$

where $M_S(I)$ is

- 0, if S did neither produce a solution nor report unsatisfiability; or otherwise
- the number of participant systems that did not produce any strictly better solution than S, where a confirmed optimum solution is considered strictly better than an unconfirmed one.

The score $S_1(D)$ of system S in domain D is then taken as the sum of scores S(D, I) over the N instances I in D.

The second scoring scheme considers the number of instances solved "optimally", i.e., a confirmed optimum solution or unsatisfiability is reported. Hence, the score $S_2(D)$ of a system S in a domain D is defined as S(D) above, with N_S being the number of instances solved optimally. This second scoring scheme (inspired by the MaxSAT Competition) gives more importance to solvers that can actually solve instances to the optimum, but it does not consider "non-optimal" solutions. The two measures provide alternative perspectives on the performance of participants solving optimization problems.

Note that, as with Decision problems and Query answering tasks, $S_1(D)$ and $S_2(D)$ range from 0 to 100 in each domain. $S_1(D)$ focuses on the best solutions found by participant systems, while $S_2(D)$ on completed runs.

In each category and respective sub-tracks, the participant systems are ranked by their sums of scores over all domains, in decreasing order. In case of a draw in terms of the sum of scores, the sums of runtimes over all instances are taken into account as a tie-breaking criterion.

3 Benchmark Suite and Selection

The benchmark suite of the Seventh ASP Competition includes 36 domains, where 28 stem from the previous competition edition (Gebser et al. 2017b), and 8 domains, as well as additional instances for the *Graph Colouring* problem, were newly submitted. We first describe the eight new domains and then detail the instance selection process based on empirical hardness.

3.1 New Domains

The eight new domains of this ASP Competition edition can be roughly characterized as closely related to machine learning (*Bayesian Network Learning, Markov Network Learning,* and *Supertree Construction*), personnel scheduling (*Crew Allocation* and *Resource Allocation*), or combinatorial problem solving (*Paracoherent Answer Sets, Random Disjunctive ASP,* and *Traveling Salesperson*), respectively. While *Traveling Salesperson* constitutes a classical optimization problem in computer science, the five domains stemming from machine learning and personnel scheduling are application-oriented, and the contribution of such practically relevant benchmarks to the ASP Competition is particularly encouraged (Gebser et al. 2017b). Moreover, the *Paracoherent Answer Sets* and *Random Disjunctive ASP* domains contribute to Sub-track #4, which was sparsely populated in recent ASP Competition editions, and beyond theoretical interest these benchmarks are relevant to logic program debugging and industrial solvers development. The following paragraphs provide more detailed background information for each of the new domains.

Bayesian Network Learning. Bayesian networks are directed acyclic graphs representing (in)dependence relations between variables in multivariate data analysis. Learning the structure of Bayesian networks, i.e., selecting arcs such that the resulting graph fits given data best, is a combinatorial optimization problem amenable to constraint-based solving methods like the one proposed in (Cussens 2011). In fact, data sets from the literature serve as instances in this domain, while a problem encoding in ASP-Core-2 expresses optimal Bayesian networks, given by directed acyclic graphs whose associated cost is minimal.

Crew Allocation. This scheduling problem, which has also been addressed by related constraintbased solving methods (Guerinik and Caneghem 1995), deals with allocating crew members to flights such that the amount of personnel with certain capabilities (e.g., role on board and spoken language) as well as off-times between flights are sufficient. Moreover, instances with different numbers of flights and available personnel restrict the amount of personnel that may be allocated to flights in such a way that no schedule is feasible under the given restrictions.

Markov Network Learning. As with Bayesian networks, the learning problem for Markov networks (Janhunen et al. 2017) aims at the optimization of graphs representing the dependence structure between variables in statistical inference. In this domain, the graphs of interest are undirected and required to be chordal, while associated scores express marginal likelihood with respect to given data. Problem instances of varying hardness are obtained by taking samples of different size and density from literature data sets.

Resource Allocation. This scheduling problem deals with allocating the activities of business processes to human resources such that role requirements and temporal relations between activities are met (Havur et al. 2016). Moreover, the total makespan of schedules is subject to an upper bound as well as optimization. The hardness of instances in this domain varies with respect to the number of activities, temporal relations, available resources, and upper bounds.

Supertree Construction. The goal of the supertree construction problem (Koponen et al. 2015) is to combine the leaves of several given phylogenetic subtrees into a single tree fitting the given subtrees as closely as possible. That is, optimization aims at preserving the structure of subtrees, where the introduction of intermediate nodes between direct neighbors is tolerated, while the avoidance of such intermediate nodes is an optimization target as well. Instances of varying hardness are obtained by mutating projections of binary trees with different numbers of leaves.

Traveling Salesperson. The well-known traveling salesperson problem (Applegate et al. 2007) is to find a round trip through a (directed) graph that is optimal in terms of the accumulated edge costs. Instances in this domain are twofold by stemming from the TSPLIB repository⁵ or being randomly generated to increase the variety in the ASP Competition, respectively.

Paracoherent Answer Sets. Given an incoherent logic program P, i.e., a program P without answer sets, a paracoherent (or semi-stable) answer set corresponds to a gap-minimal answer set of the epistemic transformation of P (Inoue and Sakama 1996; Amendola et al. 2016). The instances in this domain, used in (Amendola et al. 2017; Amendola et al. 2018) to evaluate genuine implementations of paracoherent ASP, are obtained by grounding and transforming incoherent programs from previous editions of the ASP Competition. In particular, weak constraints single out answer sets of a transformed program containing a minimal number of atoms that are actually underivable from the original program.

Random Disjunctive ASP. The disjunctive logic programs in this domain (Amendola et al. 2017) express random 2QBF formulas, given as conjunctions of terms in disjunctive normal form, by an extension of the Eiter-Gottlob encoding (Eiter and Gottlob 1995). Parameters controlling the random generation of 2QBF formulas (e.g., number of variables and number of conjunctions) are set so that instances lie close to the phase transition region, while having an expected average solving time below the competition timeout of 20 minutes per run.

M. Gebser, M. Maratea and F. Ricca

Domain	P		Eas	sy		Mee	liun	1		Ha	rd			Too l	naro	ł	J
Graph Colouring	D	1	(1)	3 (5)	2 (2)	4	(21)	2	(3)	5((16)	0	(0)	3	(3)	S
Knight Tour with Holes	D	2	(5)	3 (4)	4 (4)	0	(0)	4	(9)	0	(0)	0	(0)	7((302)	-i-
Labyrinth	D	4	(45)	0 (0)	5(72)	0	(0)	7	(83)	0	(0)	0	(0)	4	(8)	tra
Stable Marriage	D	0	(0)	0 (0)	3 (3)	0	(0)	6	(15)	1	(1)	0	(0)	10	(55)	ck
Visit-all	D	8	(14)	0 (0)	5 (5)	0	(0)	7	(40)	0	(0)	0	(0)	0	(0)	#1
Combined Configuration	D	1	(1)	0 (0)	1 (1)	0	(0)	12	(44)	0	(0)	0	(0)	6	(34)	
Consistent Query Answering	Q	0	(0)	0 (0)	0 (0)	0	(0)	0	(0)	0	(0)	0	(0)	20((120)	
Crew Allocation	D	0	(0)	4(10)	0 (0)	6	(11)	0	(0)	6((10)	0	(0)	4	(6)	
Graceful Graphs	D	3	(3)	0 (0)	4 (4)) 1	(1)	4	(28)	2	(2)	0	(0)	6	(21)	7.0
Incremental Scheduling	D	2	(11)	2 (6)	3(47)	20	(11)	3	(37)	2((10)	0	(0)	6	(76)	Ľ,
Nomystery	D	4	(4)	0 (0)	4 (5)	0	(0)	4	(10)	0	(0)	0	(0)	8	(32)	Ŧ
Partner Units	D	3	(9)	1 (1)	4(34)	0	(0)	3	(15)	1	(1)	0	(0)	8	(32)	acl
Permutation Pattern Matching	D	2	(16)	2(32)	2(14)	20	(58)	0	(0)	5((14)	0	(0)	7	(20)	K #
Qualitative Spatial Reasoning	D	5	(35)	4(35)	4(34)	20	(19)	3	(7)	2	(2)	0	(0)	0	(0)	ы
Reachability	Q	0	(0)	0 (0) 1	0(30)	10	(30)	0	(0)	0	(0)	0	(0)	0	(0)	
Ricochet Robots	D	2	(2)	0 (0)	7(18)	0	(0)	4(181)	0	(0)	0	(0)	7	(38)	
Sokoban	D	2	(77)	2(10)	2(84)	2	(8)	5(114)	2((12)	0	(0)	5((620)	
Bayesian Network Learning	0	4	(4)	0 (0)	4 (8)	0	(0)	8	(19)	0	(0)	4	(20)	0	(6)	
Connected Still Life	0	0	(0)	0 (0)	5 (5)	0	(0)	10	(70)	0	(0)	5	(45)	0	(0)	
Crossing Minimization	0	1	(1)	0 (0)	1 (1)	0	(0)	17	(80)	0	(0)	1	(1)	0	(0)	
Markov Network Learning	0	0	(0)	0 (0)	0 (0)	0	(0)	0	(0)	0	(0)	10	(10)	10	(50)	
Maximal Clique	0	0	(0)	0 (0)	0 (0)	0	(0)	10	(41)	0	(0)	10	(94)	0	(1)	\mathbf{S}
MaxSAT	0	0	(0)	0 (0)	4 (4)	0	(0)	0	(0)	0	(0)	0	(0)	16	(50)	Þ
Resource Allocation	0	-	(3)	- (0)	- (3)) —	(0)	-	(0)	-	(0)	-	(0)	-	(0)	Ira
Steiner Tree	0	0	(0)	0 (0)	0 (0)	0	(0)	1	(1)	0	(0)	16	(45)	3	(3)	ck
Supertree Construction	0	0	(0)	0 (0)	0 (0)	0	(0)	6	(30)	0	(0)	14	(30)	0	(0)	选
System Synthesis	0	0	(0)	0 (0)	0 (0)	0	(0)	8	(16)	0	(0)	8	(80)	4	(4)	
Traveling Salesperson	0	0	(0)	0 (0)	2 (2)	0	(0)	3	(3)	0	(0)	12	(60)	3	(3)	
Valves Location	0	6	(10)	0 (0)	2 (2)	0	(0)	7	(29)	0	(0)	5((244)	0	(23)	
Video Streaming	0	11	(16)	0 (0)	0 (0)	0	(0)	0	(0)	0	(0)	8	(22)	1	(1)	
Abstract Dialectical Frameworks	0	4	(18)	0 (0)	8(20)	0	(0)	6((122)	0	(0)	2	(2)	0	(0)	70
Complex Optimization	D	0	(0)	0 (0)	0 (0)	0	(0)	20	(78)	0	(0)	0	(0)	0	(0)	ŭ
Minimal Diagnosis	D	7((158)	2(55)	3 (9)	2	(8)	4	(4)	1	(1)	0	(0)	0	(0)	Ŧ
Paracoherent Answer Sets	0	0	(0)	0 (0)	1 (1)	0	(0)	12([112)	0	(0)	0	(0)	7	(43)	acl
Random Disjunctive ASP	D	0	(0)	0 (0)	0 (0)	0	(0)	5	(48)	13((73)	0	(0)	2	(2)	#
Strategic Companies	Q	0	(0)	0 (0)	0 (0)	0	(0)	0	(0)	0	(0)	0	(0)	20	(37)	+

Table 1: Problem domains of benchmarks for the Seventh ASP Competition, where entries in the **P** column indicate *Decision* ("D"), *Optimization* ("O"), or *Query answering* ("Q") tasks. The remaining columns provide numbers of instances per empirical hardness class, distinguishing *satisfiable* and *unsatisfiable* instances classified as **Easy**, **Medium**, or **Hard**, while **Too hard** instances are divided into those known to be *satisfiable* and others whose satisfiability is *unknown*. For each hardness class and satisfiability status, the number in front of parentheses stands for the selected instances out of the respective available instances whose number is given in parentheses.

3.2 Benchmark Selection

Table 1 gives an overview of all problem domains, grouped by their respective sub-tracks, of the Seventh ASP Competition, where the names of new domains are highlighted in boldface. The second column provides the computational task addressed in a domain, distinguishing Decision

("D") and Optimization ("O") problems as well as Query answering ("Q"). Further columns categorize the instances in each domain by their empirical hardness, where hardness classes are based on the performance of the same reference systems, i.e., CLASP, LP2NORMAL2+CLASP, and WASP-1.5, as in the previous ASP Competition edition (Gebser et al. 2017b):⁶

- Easy: Instances completed by at least one reference system in more than 20 seconds and by all reference systems in less than 2 minutes solving time.
- **Medium**: Instances completed by at least one reference system in more than 2 minutes and by all reference systems in less than 20 minutes (the competition timeout) solving time.
- Hard: Instances completed by at least one reference system in less than 40 minutes, while also at least one (not necessarily the same) reference system did not finish solving in 20 minutes.
- **Too hard**: Instances such that none of the reference systems finished solving in 40 minutes.

For each of these hardness classes, numbers of available instances per problem domain are shown within parentheses in Table 1, further distinguishing satisfiable and unsatisfiable instances, whose respective numbers are given first or second, respectively. In case of instances classified as "too hard", however, no reference system could report unsatisfiability, and thus the numbers of instances listed second refer to an unknown satisfiability status. Note that there are likewise no "too hard" instances of Decision problems or Query answering domains known as satisfiable, so that the respective numbers are zero. For example, the *Sokoban* domain features satisfiable as well as unsatisfiable instances for each hardness class apart from the "too hard" one, where 0 instances are known as satisfiable and 620 have an unknown satisfiability status. Unlike that, "too hard" instances of Optimization problems are frequently known to be satisfiable, in which case none of the reference systems was able to confirm an optimum solution within 40 minutes. Moreover, we discarded any instance of an Optimization problem that was reported to be unsatisfiable, so that the respective numbers given second are zero for the first three hardness classes. This applies, e.g., to instances in the *Bayesian Network Learning* domain, including 4, 8, 19, and 20 satisfiable instances that are "easy", "medium", "hard", or "too hard", respectively, while the satisfiability status of further 6 "too hard" instances is unknown. Finally, the numbers in front of parentheses in Table 1 report how many instances were (randomly) selected per hardness class and satisfiability status, and the selection process is described in the rest of this section.

Given the numbers of satisfiable, unsatisfiable, or unknown in case of "too hard" instances per hardness class, our benchmark selection process aims at picking 20 instances in each problem domain such that the four hardness classes are balanced, while another share of instances is added freely. Perfect balancing would then consist of picking four instances per hardness class and another four instances freely in order to guarantee that each hardness class contributes 20% of the instances in a domain. Since in most domains the instances are not evenly distributed, it is not possible though to insist on at least four instances per hardness class, and rather we have to compensate for underpopulated classes at which the number of available instances is smaller.

The input to our balancing scheme includes a collection C of classes, where each class is identified with the set of its contained instances. The first step of balancing then determines the non-empty classes from which instances can be picked:

⁶ This choice of reference systems allows us to reuse the runtime results for previous domains gathered in exhaustive experiments on all available instances that took about 212 CPU days on the competition platform. Instances that do not belong to any of the listed hardness classes are in the majority of cases "very easy" and the remaining ones "non-groundable", and we exclude such (uninformative regarding the system ranking) instances from the benchmark suite.

$$classes = \{ x \in C \mid x \neq \emptyset \}.$$

The number of non-empty classes is used to calculate how many instances should ideally be picked per class, where the calculation makes use of the parameters n = 20 and m = 1, standing for the total number of instances to select per domain and the fraction of instances to pick freely, respectively:

$$target = \lfloor n/(|classes| + m) \rfloor.$$
(1)

To account for underpopulated classes, in the next step we calculate the gap between the intended number of and the available instances in each class:

$$gap(x) = \begin{cases} 0 \text{ if } x \in C \setminus classes\\ target - |x| \text{ if } x \in classes. \end{cases}$$

Example 2

In the *Graceful Graphs* domain, the "easy", "medium", "hard", and "too hard" classes contain 3, 5, 30, or 21 instances, respectively, when not (yet) distinguishing between satisfiable and unsatisfiable instances. Since all four hardness classes are non-empty, we obtain *classes* = {"easy", "medium", "hard", "too hard"}. The calculation of instances to pick per class yields *target* = $\lfloor 20/(4+1) \rfloor = 4$, so that we aim at 4 instances per hardness class. Along with the number of instances available in each class, we then get gap ("easy") = 4 - 3 = 1, gap ("medium") = 4 - 5 = -1, gap ("hard") = 4 - 30 = -26, and gap ("too hard") = 4 - 21 = -17. Note that a positive number expresses underpopulation of a class relative to the intended number of instances, while negative numbers indicate capacities to compensate for such underpopulation.

Our next objective is to compensate for underpopulated classes by increasing the number of instances to pick from other classes in a fair way. Regarding hardness classes, our compensation scheme relies on the direct successor relation \prec given by "easy" \prec "medium", "medium" \prec "hard", and "hard" \prec "too hard". We denote the strict total order obtained as the transitive closure of \prec by <, and its inverse relation by >. Moreover, we let \circ below stand for either < or > to specify calculations that are performed symmetrically, such as determining the number of easier or harder instances available to compensate for the (potential) underpopulation of a class:

$$available(x)^{\circ} = \sum_{x' \circ x} gap(x').$$

The possibility of compensation in favor of easier or harder instances is then determined as follows:

 $compensate(x)^{\circ} = \min\{(|gap(x)| + gap(x))/2, (|available(x)^{\circ}| - available(x)^{\circ})/2\}.$

The calculation is such that a positive gap, standing for the underpopulation of a class, is a prerequisite for obtaining a non-zero outcome, and the availability of easier or harder instances to compensate with is required in addition. Given the compensation possibilities, the following calculations decide about how many easier or harder instances, respectively, are to be picked to resolve an underpopulation, where the distribution should preferably be even and tie-breaking in favor of harder instances is used as secondary criterion if the number of instances to compensate for is odd:⁷

⁷ Given that $distribute(x)^{<}$ (or $distribute(x)^{<}$) is limited by $compensate(x)^{<}$ (or $compensate(x)^{>}$), the superscripts ">" and "<" refer to easier or harder instances, respectively, to be picked in addition. This reading is chosen for a convenient notation in the specification of classes whose numbers of instances are to be increased for compensation.

 $distribute(x)^{>} = \min\{compensate(x)^{<}, \max\{|gap(x)| - compensate(x)^{>}, \lfloor |gap(x)|/2 \rfloor\}\}$ $distribute(x)^{<} = \min\{compensate(x)^{>}, |gap(x)| - distribute(x)^{>}\}.$

It remains to choose classes whose numbers of instances are to be increased for compensation, where we aim to distribute instances to closest classes with compensation capacities. The following inductive calculation scheme accumulates instances to distribute according to this objective:

$$accumulate(x)^{\circ} = \begin{cases} 0 \text{ if } \{x' \in C \mid x' \circ x\} = \emptyset \\ accumulate(x')^{\circ} + distribute(x')^{\circ} - increase(x')^{\circ} \text{ if } x' \circ x \\ \text{ and } x' \prec x \text{ or } x \prec x' \end{cases}$$
$$increase(x)^{<} = \min\{accumulate(x)^{<}, (|gap(x)| - gap(x))/2\}$$
$$increase(x)^{>} = \min\{accumulate(x)^{>}, (|gap(x)| - gap(x))/2 - increase(x)^{<}\}.$$

In a nutshell, $accumulate(x)^{<}$ and $accumulate(x)^{>}$ express how many easier or harder instances, respectively, ought to be distributed up to a class x, and $increase(x)^{<}$ and $increase(x)^{>}$ stand for corresponding increases of the number of instances to be picked from x. The instances to increase with are then added to the original number of instances to pick from a class as follows:

$$select(x) = |x| - (|gap(x)| - gap(x))/2 + increase(x)^{<} + increase(x)^{>}.$$

Example 3

Given gap(``easy'') = 1, gap(``medium'') = -1, gap(``hard'') = -26, and gap(``too hard'') = -17 from Example 2 for the *Graceful Graphs* domain, we obtain the following numbers indicating the availability of easier instances: $available(\text{``easy''})^{<} = 0$, $available(\text{``medium''})^{<} = 1$, $available(\text{``hard''})^{<} = 1 + (-1) = 0$, and $available(\text{``too hard''})^{<} = 1 + (-1) + (-26) = -26$. Likewise, the available harder instances are expressed by $available(\text{``too hard''})^{>} = 0$, $available(\text{``hard''})^{>} = -17$, $available(\text{``medium''})^{>} = (-17) + (-26) = -43$, and $available(\text{``easy''})^{>} = (-17) + (-26) + (-1) = -44$. Again note that positive numbers like $available(\text{``medium''})^{<} = 1$ represent a (cumulative) underpopulation, while negative numbers such as $available(\text{``easy''})^{>} = -44$ indicate compensation capacities.

Considering "easy" instances, we further calculate $compensate(\text{``easy''})^{<} = \min\{(|1|+1)/2, (|0|-0)/2\} = 0$ and $compensate(\text{``easy''})^{>} = \min\{(|1|+1)/2, (|-44|-(-44))/2\} = 1$. This tells us that we can add one harder instance to compensate for the underpopulation of the "easy" class, while $compensate(x)^{\circ} = compensate(\text{``easy''})^{<} = 0$ for the other classes $x \in \{\text{``medium'', ``hard'', ``too hard''\}}$ and $\circ \in \{<,>\}$. Given that instances to distribute are limited by compensation possibilities, which are non-zero at underpopulated classes only, it is sufficient to concentrate on "easy" instances in the *Graceful Graphs* domain. This yields $distribute(\text{``easy''})^{>} = \min\{0, \max\{1-1, 0\}\} = 0$ and $distribute(\text{``easy''})^{<} = \min\{1, 1-0\} = 1$, so that one harder instance is to be picked more.

The calculation of instance number increases to compensate for underpopulated classes then starts with $accumulate("easy")^{<} = 0$, $increase("easy")^{<} = \min\{0, (|1| - 1)/2\} = 0$, $accumulate("medium")^{<} = 0 + 1 - 0 = 1$, $increase("medium")^{<} = \min\{1, (|-1| - (-1))/2\} = 1$, and $accumulate("hard")^{<} = 1 + 0 - 1 = 0$. That is, the instance to distribute from the underpopulated "easy" class to some harder class increases the number of "medium" instances, while we obtain $increase("hard")^{<} = accumulate("too hard")^{<} = increase("too hard")^{<} = 0$ as well as $accumulate(x)^{>} = increase(x)^{>} = 0$ for all $x \in \{\text{"easy", "medium", "hard", "too hard"}\}$. The final numbers of instances to pick per hardness

class in the *Graceful Graphs* domain are thus determined by select("easy") = 3 - (|1| - 1)/2 + 0 + 0 = 3, select("medium") = 5 - (|-1| - (-1))/2 + 1 + 0 = 5, select("hard") = 30 - (|-26| - (-26))/2 + 0 + 0 = 4, and select("too hard") = 21 - (|-17| - (-17))/2 + 0 + 0 = 4. Note that 16 instances are to be selected from particular hardness classes in total, sparing the four instances to be picked freely, and also that our balancing scheme takes care of exchanging an "easy" for a "medium" instance.

After determining the numbers of instances to pick per hardness class, we also aim to balance between satisfiable and unsatisfiable instances within the same class. In fact, the above balancing scheme is general enough to be reused for this purpose by letting $C = \{\text{satisfiable}(x), \text{unsatisfiable}(x)\}$ consist of the subclasses of satisfiable or unsatisfiable instances, respectively, in a hardness class x that includes at least one instance known to be satisfiable or unsatisfiable.⁸ Moreover, the parameters n and m used in (1) are fixed to n = select(x) and m = 0, which reflect that the satisfiability status should be balanced among all instances to be picked from x without allocating an additional share of instances to pick freely. For the strict total order on the subclasses in C, we use satisfiable $(x) \prec$ unsatisfiable(x), let < denote the transitive closure of \prec , and > its inverse relation.

Example 4

Reconsidering the *Graceful Graphs* domain, we obtain the following number of instances to pick based on their satisfiability status: select(satisfiable("easy")) = 3, select(unsatisfiable("medium")) = 3, select(unsatisfiable("medium")) = 1, select(satisfiable("hard")) = 2, and select(unsatisfiable("hard")) = 2. Note that select(satisfiable("hard")) + select(unsatisfiable(x)) = select(x) for $x \in \{"easy", "hard"\}$, while select(satisfiable("medium")) + select(unsatisfiable("medium")) = 3 + 1 = 4 < 5 = select("medium"). The latter is due to rounding in $target = \lfloor 5/2 \rfloor = 2$, and then compensating for the underpopulated unsatisfiable instances by increasing the number of satisfiable "medium" instances to pick by one.

For instances of Decision problems or Query answering domains, we have that secondary balancing based on the satisfiability status is generally void for "too hard" instances, of which none are known to be satisfiable or unsatisfiable. In case of Optimization problems, where we discard instances known as unsatisfiable, select(satisfiable(x)) = select(x) holds for $x \in \{\text{"easy"},$ "medium", "hard"}, while our balancing scheme favors "too hard" instances known as satisfiable over those with an unknown satisfiability status. This approach makes sure that "too hard" instances to be picked possess solutions, yet confirming an optimum is hard, and instances with an unknown satisfiability status can still be contained among those that are picked freely.

Example 5

Regarding the Optimization problem in the *Valves Location* domain, we obtain *select*(satisfiable("too hard")) = select("too hard") = 4, given that the 23 instances whose satisfiability status is unknown are not considered for balancing.

The described twofold balancing scheme, first considering the hardness of instances and then the satisfiability status of instances of similar hardness, is implemented by an ASP-Core-2 encoding that consists of two parts: a deterministic program part (having a unique answer set) takes care

⁸ Otherwise, all subclasses to pick instances from are empty, which would lead to division by zero in (1).

of determining the numbers select(x) from the runtime results of reference systems, and a nondeterministic part similar to the selection program used in the previous ASP Competition edition (Gebser et al. 2017b) encodes the choice of 20 instances per domain such that lower bounds given by the calculated numbers select(x) are met. In comparison to the previous competition edition, we updated the deterministic part of the benchmark selection encoding by implementing the balancing scheme described above, which is more general than before and not fixed to a particular number of classes (regarding hardness or satisfiability status) to balance. The instance selection was then performed by running the ASP solver CLASP with the options -rand-freq, -sign-def, and -seed for guaranteeing reproducible randomization, using the concatenation of winning numbers in the EuroMillions lottery of 2nd May 2017 as the random seed. This process led to the numbers of instances picked per domain, hardness class, and satisfiability status listed in Table 1.

As a final remark, we note that we had to exclude the *Resource Allocation* domain from the main competition in view of an insufficient number of instances belonging to the hardness classes under consideration. In fact, the majority of instances turned out to be "very easy" relative to an optimized encoding devised in the phase of checking/establishing the ASP-Core-2 compliance of initial submissions by benchmark authors. This does not mean that the problem of *Resource Allocation* as such would be trivial or uninteresting, but rather time constraints on running the main competition did unfortunately not permit to extend and then reassess the collection of instances.

4 Participant Systems

Fourteen systems, registered by three teams, participate in the System track of the Seventh ASP Competition. The majority of systems runs in the **SP** category, while two (indicated by the suffix "-MT" below) exploit parallelism in the **MP** category. In the following, we survey the registered teams and systems.

Aalto. The team from Aalto University submitted nine systems that utilize normalization (Bomanson et al. 2014; Bomanson et al. 2016) and translation (Bogaerts et al. 2016; Bomanson et al. 2016; Gebser et al. 2014; Janhunen and Niemelä 2011; Liu et al. 2012) means. Two systems, LP2sAT+LINGELING and LP2sAT+PLINGELING-MT, perform translation to SAT and use LINGELING or PLINGELING, respectively, as back-end solver. Similarly, LP2MIP and LP2MIP-MT rely on translation to Mixed Integer Programming along with a single- or multi-threaded variant of CPLEX for solving. The LP2ACYCASP, LP2ACYCPB, and LP2ACYCSAT systems incorporate translations based on acyclicity checking, supported by CLASP run as ASP, Pseudo-Boolean, or SAT solver, as well as the GRAPHSAT solver in case of SAT with acyclicity checking. Moreover, LP2NORMAL+LP2STS takes advantage of the SAT-TO-SAT framework to decompose complex computations into several SAT solving tasks. Unlike that, LP2NORMAL+CLASP confines preprocessing to the (selective) normalization of aggregates and weak constraints before running CLASP as ASP solver. Beyond syntactic differences between target formalisms, the main particularities of the available translations concern space complexity and the supported language features. Regarding space, the translation to SAT utilized by LP2sAT+LINGELING and LP2sAT+PLINGELING-MT comes along with a logarithmic overhead in case of non-tight logic programs that involve positive recursion (Fages 1994), while the other translations are guaranteed to remain linear. Considering language features, the systems by the Aalto team do not support queries, and the back-end solver CLASP of LP2ACYCASP, LP2ACYCPB, and LP2NORMAL+CLASP provides a native implementation of aggregates, which the

M. Gebser, M. Maratea and F. Ricca

other systems treat by normalization within preprocessing. Optimization problems are supported by all systems but LP2SAT+LINGELING, LP2SAT+PLINGELING-MT, and LP2NORMAL+LP2STS, while only LP2NORMAL+LP2STS and LP2NORMAL+CLASP are capable of handling non-HCF disjunction.

ME-ASP. The ME-ASP team from the University of Genova, the University of Sassari, and the University of Calabria submitted the multi-engine ASP system ME-ASP2, which is an updated version of ME-ASP (Maratea et al. 2012; Maratea et al. 2014; Maratea et al. 2015), the winner system in the Regular track of the Sixth ASP Competition. Like its predecessor version, ME-ASP2 investigates features of an input program to select its back-end among a pool of ASP grounders and solvers. Basically, ME-ASP2 applies algorithm selection techniques before each stage of the answer set computation, with the goal of selecting the most promising computation strategy overall. As regards grounders, ME-ASP2 can pick either DLV or GRINGO, while the available solvers include a selection of those submitted to the Sixth ASP Competition as well as the latest version of CLASP. The first selection (basically corresponding to the selection of the grounder) is based on features of non-ground programs and was obtained by implementing the result of the application of the PART decision list algorithm, whereas the choice of a solver is based on the multinomial classification algorithm k-Nearest Neighbors, used to train a model on features of ground programs extracted (whenever required) from the output generated by the grounder (for more details, see (Maratea et al. 2015)).

UNICAL. The team from the University of Calabria submitted four systems utilizing the recent IDLV grounder (Calimeri et al. 2017), developed as a redesign of (the grounder component of) DLV going along with the addition of new features. Moreover, back-ends for solving are selected from a variety of existing ASP solvers. In particular, IDLV-CLASP-DLV makes use of DLV (Leone et al. 2006; Maratea et al. 2008) for instances featuring a ground query; otherwise, it consists of the combination of the grounder IDLV with CLASP executed with the option --configuration=trendy. The IDLV+-CLASP-DLV system is a variant of the previous system that uses a heuristic-guided rewriting technique (Calimeri et al. 2018) relying on hyper-tree decomposition, which aims to automatically replace long rules with sets of smaller ones that are possibly evaluated more efficiently. IDLV+-WASP-DLV is obtained by using WASP in place of CLASP. In more detail, wasp is executed with the options --shrinking-strategy=progression --shrinking-budget=10 --trim-core --enable-disjcores, which configure WASP to use two techniques tailored for Optimization problems. Inspired by ME-ASP, IDLV+S (Fuscà et al. 2017) integrates IDLV+ with an automatic selector to choose between WASP and CLASP on a per instance basis. To this end, IDLV+s implements classification, by means of the well-known support vector machine technique. A more detailed description of the IDLV+S system is provided in (Calimeri et al. 2019).

5 Results

This section presents the results of the Seventh ASP Competition. We first announce the winners in the **SP** category and analyze their performance, and then proceed overviewing results in the **MP** category. Finally, we analyze the results more in details outlining some of the outcomes.



Fig. 2: Results of the **SP** category with computation S_1 for Optimization problems.

5.1 Results in the SP Category

Figures 2 and 3 summarize the results of the **SP** category, by showing the scores of the various systems, where Figure 2 utilizes function S_1 for computing the score of Optimization problems, while Figure 3 utilizes function S_2 . To sum up, considering Figure 2, the first three places go to the systems:

- 1. IDLV+S, by the UNICAL team, with 2665 points;
- 2. IDLV+-CLASP-DLV, by the UNICAL team, with 2655,9 points;
- 3. IDLV-CLASP-DLV, by the UNICAL team, with 2634 points.

Also, ME-ASP is quite close in performance, earning 2560 points in total.

Thus, the first three places are taken by versions of the IDLV system pursuing the approaches outlined in Section 4. The fourth place, with very positive results, is instead taken by ME-ASP which pursues a portfolio approach, and was the winner of the last competition.

Going into the details of sub-tracks, the three top-performing systems overall take the first places as well:

- Sub-track #1 (Basic Decision): IDLV-CLASP-DLV and IDLV+-CLASP-DLV with 400 points;
- Sub-track #2 (Advanced Decision): IDLV+-CLASP-DLV with 805 points;
- Sub-track #3 (Optimization): IDLV+-CLASP-DLV with 1015,9 points;
- Sub-track #4 (Unrestricted): IDLV+S with 450 points.

Considering Figure 3, which employs function S_2 for computing scores of Optimization problems, the situation is slightly different, i.e., ME-ASP now gets the third place. To sum up, the first three places go to the systems:

- 1. IDLV+S, by the UNICAL team, with 2330 points;
- 2. IDLV+-CLASP-DLV, by the UNICAL team, with 2200 points;
- 3. ME-ASP, by the ME-ASP team, with 2185 points.

IDLV+-CLASP-DLV is now fourth with the same score of 2185 points but higher cumulative CPU time: the difference is in Sub-track #3, where relative results are different with respect to



Fig. 3: Results of the SP category with computation S_2 for Optimization problems.



Fig. 4: Cactus plot of solver performances in the SP category.

using S_1 , and with the new score computation ME-ASP earns 55 points more than IDLV+-CLASP-DLV. In general, employing S_2 function for computing scores of Optimization problems leads to lower scores: indeed, S_2 is more restrictive than S_1 given that only optimal results are considered.

An overall view of the performances of all participant systems on all benchmarks is shown in the cactus plot of Figure 4. Detailed results are reported in Appendix A.

Official results in Figures 2 and 3 are complemented by the data showed in Figures 5 and 6. Figure 5 contains, for each solver, the number of (optimally) solved instances in each reasoning problem of the competition, i.e., Decision, Optimization and Query (denoted Dec, Opt, and Query in the figure, respectively). From the figure, we can see that IDLV+-CLASP-DLV and IDLV-CLASP-DLV are the solvers that perform best on Decision problems, while IDLV+-S is the best on Optimization problems. For what concern Query answering, the first four solvers perform equally



Fig. 5: Number of (optimally) solved instances in the SP category by task.



Fig. 6: Percentage of solved instances in the SP category.

well on them. Figure 6, instead, reports, for each solver, the percentage of solved instances (resp. score) in the various sub-tracks out of the total number of (optimally) solved instances (resp. global score), i.e., what is the "contribution" of tasks in each sub-track to the results of the respective system.

5.2 Results in the MP category

Figure 7 shows results about the **MP** category. The bottom part of the figure reports the scores acquired by the two participant systems, which cumulatively are: for

- 1. LP2SAT+PLINGELING-MT, by the Aalto team, 715 points;
- 2. LP2MIP-MT, by the Aalto team, 635 points.

Looking into details of the sub-tracks, we can note that LP2SAT+PLINGELING-MT is better than LP2MIP-MT on Sub-track #1 and much better on Sub-track #2, while on Sub-track #3,

17



Fig. 7: Results of the MP category.

where LP2SAT+PLINGELING-MT does not compete, LP2MIP-MT earns a consistent number of points, but not enough to globally reach the score of LP2SAT+PLINGELING-MT in the first two sub-tracks.

The top part of Figure 7, instead, complements the results by showing the "contribution" of solved instances in each sub-track out of the score of the respective system.

5.3 Analysis of the results

There are some general observations that can be drawn out of the results presented in this section. First, the best overall solver implements algorithm selection techniques, and continues the "tradition" of the efficiency of portfolio-based solvers in ASP competitions, given that CLASP-FOLIO (Gebser et al. 2011) and ME-ASP (Maratea et al. 2014) were the overall winners of the 2011 and 2015 competitions, respectively. At the same time, the result outlines the importance of the introduction of new evaluation techniques and implementations. Indeed, although IDLV+-S applies a strategy similar to the one of ME-ASP, IDLV+-S exploited a new component (i.e., the grounder (Calimeri et al. 2016)) that was not present in ME-ASP (which is based on solvers from the previous competition). Second, the approach implemented by LP2NORMAL using CLASP confirms its very good behavior in all sub-tracks, and thus overall. Third, specific istantiations of the translation-based approach perform particularly well in some sub-tracks: this is the case for the LP2ACYCASP solver using CLASP in Sub-track #3, especially when considering scoring scheme S_1 , but also for LP2MIP, that compiles to a general purpose solver, in the same sub-track, especially when considering scoring scheme S_2 (even if to a less extent). As far as the comparison between solvers in the MP category and their counter-part in the SP category is concerned, we can see that globally the score of LP2SAT+PLINGELING-MT and LP2SAT+PLINGELING is the same, with the small advantage of LP2SAT+PLINGELING-MT in Sub-track #1 being compensated in Sub-track #2. Instead, LP2MIP+MT earns a consistent number of points more than LP2MIP,

especially in Sub-track #1 and #3. In general, more specific research is probably needed on ASP solvers exploiting multi-threading to take real advantage from this setting.

6 Conclusion and Final Remarks

We have presented design and results of the Seventh ASP Competition, with particular focus on new problem domains, revised benchmark selection process, systems registered for the event, and results.

In the following, we draw some recommendations for future editions. These resemble the ones of the past event: for some of them some steps have been already made in this seventh's event, but they may be considered, with the aim of widening the number of participant systems and application domains that can be analyzed, starting from the next (Eighth) ASP competition that will take place in 2019 in affiliation with the 15th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR 2019), in Philadelphia, US:

- We also tried to re-introduce a Model&Solve track at the competition. But, given the short call for contributions and the low number of expressions of interest received, we decided not to run the track. Despite this, we still think that a (restricted form of a) Model&Solve track should be re-introduced in the ASP competition series.
- Our aim with the re-introduction of a Model&Solve track was at solving domains involving, e.g., discrete as well as continuous dynamics (Balduccini et al. 2017), so that extensions like Constraint Answer Set Programming (Mellarkod et al. 2008) and incremental ASP solving (Gebser et al. 2008) may be exploited. The mentioned extensions could be added as tracks of the competition, but for CASP the first step that would be needed is a standardization of its language.
- Given that still basically all participant systems rely on grounding, the availability of more grounders is crucial. In this event the I-DLV grounder came into play, but there is also the need for more diverse techniques. This may also help improving portfolio solvers, by exploiting machine learning techniques at non-ground level (for a preliminary investigation, see (Maratea et al. 2013; Maratea et al. 2015)).
- Portfolio solvers showed good performance in the editions where they participated. However, no such system in the various editions has exploited a parallel portfolio approach. Exploring such techniques in conjunction could be an interesting topic of future research for further improving the efficiency.
- Another option for attracting (young) researchers from neighboring areas to the development of ASP solvers may be a track dedicated to modifications of a common reference system, in the spirit of the Minisat hack track of the SAT Competition series. This would lower the entrance barrier by keeping the effort of a participation affordable, even for small teams.

Acknowledgments. The organizers of the Seventh ASP Competition would like to thank the LPNMR 2017 officials for the co-location of the event. We also acknowledge the Department of Mathematics and Computer Science at the University of Calabria for supplying the computational resources to run the competition. Finally, we thank all solver and benchmark contributors, and participants, who worked hard to make this competition possible.

References

- ALVIANO, M., CALIMERI, F., DODARO, C., FUSCÀ, D., LEONE, N., PERRI, S., RICCA, F., VELTRI, P., AND ZANGARI, J. 2017. The ASP system DLV2. In Proceedings of the Fourteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'17), M. Balduccini and T. Janhunen, Eds. Lecture Notes in AI (LNAI), vol. 10377. Springer-Verlag, 215–221.
- ALVIANO, M., DODARO, C., LEONE, N., AND RICCA, F. 2015. Advances in WASP. In Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LP-NMR'15), F. Calimeri, G. Ianni, and M. Truszczyński, Eds. Lecture Notes in Computer Science, vol. 9345. Springer-Verlag, 40–54.
- AMENDOLA, G., DODARO, C., FABER, W., LEONE, N., AND RICCA, F. 2017. On the computation of paracoherent answer sets. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (AAAI'17), S. P. Singh and S. Markovitch, Eds. AAAI Press, 1034–1040.
- AMENDOLA, G., DODARO, C., FABER, W., AND RICCA, F. 2018. Externally supported models for efficient computation of paracoherent answer sets. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI'18)*, S. A. McIlraith and K. Q. Weinberger, Eds. AAAI Press, 1720–1727.
- AMENDOLA, G., EITER, T., FINK, M., LEONE, N., AND MOURA, J. 2016. Semi-equilibrium models for paracoherent answer set programs. *Artificial Intelligence 234*, 219–271.
- AMENDOLA, G., RICCA, F., AND TRUSZCZYŃSKI, M. 2017. Generating hard random Boolean formulas and disjunctive logic programs. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI'17)*, C. Sierra, Ed. ijcai.org, 532–538.
- APPLEGATE, D., BIXBY, R., CHVÁTAL, V., AND COOK, W. 2007. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.
- BALDUCCINI, M., MAGAZZENI, D., MARATEA, M., AND LEBLANC, E. 2017. CASP solutions for planning in hybrid domains. *Theory and Practice of Logic Programming* 17, 4, 591–633.
- BARAL, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.
- BEN-ELIYAHU, R. AND DECHTER, R. 1994. Propositional semantics for disjunctive logic programs. Annals of Mathematics and Artificial Intelligence 12, 53–87.
- BOGAERTS, B., JANHUNEN, T., AND TASHARROFI, S. 2016. Stable-unstable semantics: Beyond NP with normal logic programs. *Theory and Practice of Logic Programming 16*, 5-6, 570–586.
- BOMANSON, J., GEBSER, M., AND JANHUNEN, T. 2014. Improving the normalization of weight rules in answer set programs. In *Proceedings of the Fourteenth European Conference on Logics in Artificial Intelligence (JELIA'14)*, E. Fermé and J. Leite, Eds. Lecture Notes in Artificial Intelligence, vol. 8761. Springer-Verlag, 166–180.
- BOMANSON, J., GEBSER, M., AND JANHUNEN, T. 2016. Rewriting optimization statements in answerset programs. In *Technical Communications of the Thirty-second International Conference on Logic Programming (ICLP'16)*, M. Carro and A. King, Eds. Open Access Series in Informatics, vol. 52. Schloss Dagstuhl, 5:1–5:15.
- BOMANSON, J., GEBSER, M., JANHUNEN, T., KAUFMANN, B., AND SCHAUB, T. 2016. Answer set programming modulo acyclicity. *Fundamenta Informaticae* 147, 1, 63–91.
- BREWKA, G., EITER, T., AND TRUSZCZYŃSKI, M. 2011. Answer set programming at a glance. Communications of the ACM 54, 12, 92–103.
- BRUYNOOGHE, M., BLOCKEEL, H., BOGAERTS, B., DE CAT, B., DE POOTER, S., JANSEN, J., LABARRE, A., RAMON, J., DENECKER, M., AND VERWER, S. 2015. Predicate logic as a modeling language: Modeling and solving some machine learning and data mining problems with IDP3. *Theory and Practice of Logic Programming 15*, 6, 783–817.
- CALIMERI, F., DODARO, C., FUSCÀ, D., PERRI, S., AND ZANGARI, J. 2019. Efficiently coupling the I-DLV grounder with ASP solvers. *Theory and Practice of Logic Programming. To appear.*
- CALIMERI, F., FUSCÀ, D., PERRI, S., AND ZANGARI, J. 2016. I-DLV: The new intelligent grounder of dlv. In Proceedings of AI*IA 2016: Advances in Artificial Intelligence Fifteenth International

Conference of the Italian Association for Artificial Intelligence, G. Adorni, S. Cagnoni, M. Gori, and M. Maratea, Eds. Lecture Notes in Computer Science, vol. 10037. Springer, 192–207.

- CALIMERI, F., FUSCÀ, D., PERRI, S., AND ZANGARI, J. 2017. I-DLV: The new intelligent grounder of DLV. *Intelligenza Artificiale 11*, 1, 5–20.
- CALIMERI, F., FUSCÀ, D., PERRI, S., AND ZANGARI, J. 2018. Optimizing answer set computation via heuristic-based decomposition. In *Proceedings of the Twentieth International Symposium on Practical Aspects of Declarative Languages (PADL'18)*, F. Calimeri, K. W. Hamlen, and N. Leone, Eds. Lecture Notes in Computer Science, vol. 10702. Springer, 135–151.
- CALIMERI, F., GEBSER, M., MARATEA, M., AND RICCA, F. 2016. Design and results of the fifth answer set programming competition. *Artificial Intelligence 231*, 151–181.
- CALIMERI, F., IANNI, G., AND RICCA, F. 2014. The third open answer set programming competition. *Theory and Practice of Logic Programming* 14, 1, 117–135.
- CUSSENS, J. 2011. Bayesian network learning with cutting planes. In Proceedings of the Twenty-seventh International Conference on Uncertainty in Artificial Intelligence (UAI'11), F. Cozman and A. Pfeffer, Eds. AUAI Press, 153–160.
- EITER, T. AND GOTTLOB, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence 15*, 3-4, 289–323.
- EITER, T., IANNI, G., AND KRENNWALLNER, T. 2009. Answer Set Programming: A Primer. In *Reasoning Web. Semantic Technologies for Information Systems, 5th International Summer School Tutorial Lectures.* Brixen-Bressanone, Italy, 40–110.
- FAGES, F. 1994. Consistency of Clark's Completion and Existence of Stable Models. *Journal of Methods of Logic in Computer Science 1*, 1, 51–60.
- FUSCÀ, D., CALIMERI, F., ZANGARI, J., AND PERRI, S. 2017. I-DLV+MS: Preliminary report on an automatic ASP solver selector. In *Proceedings of the Twenty-fourth RCRA International Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA'17)*, M. Maratea and I. Serina, Eds. CEUR Workshop Proceedings, vol. 2011. CEUR-WS.org, 26–32.
- GEBSER, M., JANHUNEN, T., AND RINTANEN, J. 2014. Answer set programming as SAT modulo acyclicity. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence (ECAI'14)*, T. Schaub, G. Friedrich, and B. O'Sullivan, Eds. Frontiers in Artificial Intelligence and Applications, vol. 263. IOS Press, 351–356.
- GEBSER, M., KAMINSKI, R., KAUFMANN, B., OSTROWSKI, M., SCHAUB, T., AND THIELE, S. 2008. Engineering an incremental ASP solver. In *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, M. Garcia de la Banda and E. Pontelli, Eds. Lecture Notes in Computer Science, vol. 5366. Springer-Verlag, 190–205.
- GEBSER, M., KAMINSKI, R., KAUFMANN, B., ROMERO, J., AND SCHAUB, T. 2015. Progress in clasp series 3. In Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), F. Calimeri, G. Ianni, and M. Truszczyński, Eds. Lecture Notes in Computer Science, vol. 9345. Springer-Verlag, 368–383.
- GEBSER, M., KAMINSKI, R., KAUFMANN, B., SCHAUB, T., SCHNEIDER, M. T., AND ZILLER, S. 2011. A portfolio solver for answer set programming: Preliminary report. In *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*. Lecture Notes in Computer Science, vol. 6645. Springer, Vancouver, Canada, 352–357.
- GEBSER, M., LEONE, N., MARATEA, M., PERRI, S., RICCA, F., AND SCHAUB, T. 2018. Evaluation techniques and systems for answer set programming: a survey. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI 2018)*, J. Lang, Ed. ijcai.org, 5450–5456.
- GEBSER, M., MARATEA, M., AND RICCA, F. 2016. What's hot in the answer set programming competition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, D. Schuurmans and M. P. Wellman, Eds. AAAI Press, 4327–4329.
- GEBSER, M., MARATEA, M., AND RICCA, F. 2017a. The design of the seventh answer set programming competition. In *Proceedings of the Fourteenth International Conference on Logic Programming* and Nonmonotonic Reasoning (LPNMR'17), M. Balduccini and T. Janhunen, Eds. Lecture Notes in AI (LNAI), vol. 10377. Springer-Verlag, 3–9.

- GEBSER, M., MARATEA, M., AND RICCA, F. 2017b. The sixth answer set programming competition. *Journal of Artificial Intelligence Research* 60, 41–95.
- GELFOND, M. AND LEONE, N. 2002. Logic Programming and Knowledge Representation the A-Prolog perspective. Artificial Intelligence 138, 1–2, 3–38.
- GUERINIK, N. AND CANEGHEM, M. V. 1995. Solving crew scheduling problems by constraint programming. In Proceedings of the First International Conference on Principles and Practice of Constraint Programming (CP'95), U. Montanari and F. Rossi, Eds. Lecture Notes in Computer Science, vol. 976. Springer, 481–498.
- HAVUR, G., CABANILLAS, C., MENDLING, J., AND POLLERES, A. 2016. Resource allocation with dependencies in business process management systems. In *Proceedings of the Business Process Management Forum (BPM'16)*, M. L. Rosa, P. Loos, and O. Pastor, Eds. Lecture Notes in Business Information Processing, vol. 260. Springer, 3–19.
- INOUE, K. AND SAKAMA, C. 1996. A Fixpoint Characterization of Abductive Logic Programs. Journal of Logic Programming 27, 2, 107–136.
- JANHUNEN, T., GEBSER, M., RINTANEN, J., NYMAN, H., PENSAR, J., AND CORANDER, J. 2017. Learning discrete decomposable graphical models via constraint optimization. *Statistics and Computing* 27, 1, 115–130.
- JANHUNEN, T. AND NIEMELÄ, I. 2011. Compact translations of non-disjunctive answer set programs to propositional clauses. In *Proceedings of the Symposium on Constructive Mathematics and Computer Science in Honour of Michael Gelfonds 65th Anniversary*. Lecture Notes in Computer Science, vol. 6565. Springer, 111–130.
- KOPONEN, L., OIKARINEN, E., JANHUNEN, T., AND SÄILÄ, L. 2015. Optimizing phylogenetic supertrees using answer set programming. *Theory and Practice of Logic Programming* 15, 4-5, 604–619.
- LEFÈVRE, C., BÉATRIX, C., STÉPHAN, I., AND GARCIA, L. 2017. ASPeRiX, a first-order forward chaining approach for answer set computing. *Theory and Practice of Logic Programming* 17, 3, 266–310.
- LEONE, N., PFEIFER, G., FABER, W., EITER, T., GOTTLOB, G., PERRI, S., AND SCARCELLO, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Log.* 7, 3, 499–562.
- LIERLER, Y., MARATEA, M., AND RICCA, F. 2016. Systems, engineering environments, and competitions. *AI Magazine 37*, 3, 45–52.
- LIFSCHITZ, V. 2002. Answer Set Programming and Plan Generation. Artificial Intelligence 138, 39-54.
- LIFSCHITZ, V. 2008. Twelve definitions of a stable model. In *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, M. Garcia de la Banda and E. Pontelli, Eds. Lecture Notes in Computer Science, vol. 5366. Springer-Verlag, 37–51.
- LIU, G., JANHUNEN, T., AND NIEMELÄ, I. 2012. Answer set programming via mixed integer programming. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, G. Brewka, T. Eiter, and S. A. McIlraith, Eds. AAAI Press, 32–42.
- MARATEA, M., PULINA, L., AND RICCA, F. 2012. The multi-engine ASP solver me-asp. In *Proceedings* of the 13th European Conference on Logics in Artificial Intelligence (JELIA 2012), L. F. del Cerro, A. Herzig, and J. Mengin, Eds. Lecture Notes in Computer Science, vol. 7519. Springer, 484–487.
- MARATEA, M., PULINA, L., AND RICCA, F. 2013. Automated selection of grounding algorithm in answer set programming. In Advances in Artificial Intelligence Proceedings of the 13th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2013), M. Baldoni, C. Baroglio, G. Boella, and R. Micalizio, Eds. Lecture Notes in Computer Science, vol. 8249. Springer, 73–84.
- MARATEA, M., PULINA, L., AND RICCA, F. 2014. A multi-engine approach to answer-set programming. *Theory and Practice of Logic Programming 14*, 6, 841–868.
- MARATEA, M., PULINA, L., AND RICCA, F. 2015. Multi-level algorithm selection for ASP. In Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15), F. Calimeri, G. Ianni, and M. Truszczyński, Eds. Lecture Notes in Computer Science, vol. 9345. Springer-Verlag, 439–445.
- MARATEA, M., RICCA, F., FABER, W., AND LEONE, N. 2008. Look-back techniques and heuristics in dlv: Implementation, evaluation and comparison to qbf solvers. *Journal of Algorithms in Cognition, Informatics and Logics 63*, 1–3, 70–89.

- MAREK, V. W. AND TRUSZCZYŃSKI, M. 1999. Stable Models and an Alternative Logic Programming Paradigm. In *The Logic Programming Paradigm A 25-Year Perspective*, K. R. Apt, V. W. Marek, M. Truszczyński, and D. S. Warren, Eds. Springer Verlag, 375–398.
- MARPLE, K. AND GUPTA, G. 2014. Dynamic consistency checking in goal-directed answer set programming. *Theory and Practice of Logic Programming 14*, 4-5, 415–427.
- MELLARKOD, V., GELFOND, M., AND ZHANG, Y. 2008. Integrating answer set programming and constraint logic programming. *Annals of Mathematics and Artificial Intelligence 53*, 1-4, 251–287.
- NIEMELÄ, I. 1999. Logic Programming with Stable Model Semantics as Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence* 25, 3–4, 241–273.

Appendix A Detailed Results

We report in this appendix the detailed results aggregated by solver. In particular, Figures A 1-A 4 report for each solver and for each domain the score with computation S_1 for Optimization problems (ScoreASP2015), with computation score S_2 for Optimization problems (ScoreSolved), the sum of the execution times for all instances (Sum(Time)), the average memory usage on solved instances (Avg(Mem)), the number of solved instances (#Sol), the number of timed out executions (#TO), the number of execution terminated because the solver exceeded the memory limit (#MO) and the number of execution with abnormal execution (#OE), this last counting the instances that could not be solved by a solver, thus including output errors, abnormal terminations, give-ups as well as instances that cannot be solved by a solver when it did not participate to a domain. An "*" near to a score of 0 indicates that the solver was disqualified from a domain because it terminated normally but produced a wrong witness in some instance of the domain.

_	w	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ຊ	0	0	0	0	0	20	0	0	0	0	0	0	읉
	윺	0	_	0	0	0	_	0	0	0	_	_		0	0	_	0	0	0	0	_	0	0			0	0	_	_	0	0	0	0			0	-
	N M																																				
	5	0	S	S	0	10	0	4	1	∞	m	9	S	6	ŝ	10	-	0	6	10	0	0	9	0	9	4	10	9	9	0	13	20	17	4	~	0	5
	Ħ	0	S	5	0	0	2	19	6	12	5	4	5	1	S	9	61	0	11	9	0	0	4	0	9	19	0	4	4	0	7	0	3	9	<u> </u>	0	99
	\$¥																							-													4
	em)	7,75	6,76	5,12	2,04	8,41	9,14	5,90	3,92	1,06	8,62	5,69	5,38	3,65	3,20	4,94	2,93	7,62	6,05	5,58	9,53	1,25	1,88	4,08	7,05	7,62	7,97	3,81	9,86	6,97	2,68	8,08	1,00	1,69	9,28	3,98	0,0
	g(N	9	9	61	9	3	842	9	80	5	9	103	15	24	5	8	33	30	22	3	36	79	~	450		353	40	605			5	63	16	5	Ű	80	30.39
ŝ	¥ (80	4	3	33	5	6	1	1	2	6	ŝ	-	0	S	6	0	2	2	7	5	80	5	-	-	5	6	4	0	ŝ	3	5	7	m	2	ŝ	0
+nlp	ine	73,5	38,8	71,6	20,1	40,2	24,8	28,8	54,7	73,4	45,7	06,0	8,1	84,7	81,2	19,4	72,1	72,4	64,5	72,2	71,0	28,3	36,3	91,6	54,1	71,1	36,1	17,8	87,1	26,3	96,6	07,8	47,4	84,6	35,1	13,4	50,0
	Ē		65	67	34	129	45	66	12	114	5	8	19	126	8	149	23	7	121	121	4	17	107	2	136	78	129	162	76	~	166	240	204	55	8	53	2924
	d S	8	75	75	8	22	8	8	95	3	58	2	52	55	75	ß	95	8	55	ß	8	8	2	0	ß	8	50	2	2	0	35	0	15	8	5	8	ŝ
	olve	-			-		-											1			-	1														-	23
	core																																				
	5 5	00	73	00	00	73	00	00	00	8	8	8	8	00	64	73	91	00	00	8	00	00	8	8	8	00	00	8	82	00	73	36	45	8	16	00	69
	P201	100,	97,	75,	100,	77,	100,	80,	100,	60,	85,	70,	75,	55,	98,	67,	95,	100,	55,	50,	100,	100,	70,	ò	3	80,	50,	70,	96,	°,	82,	61,	75,	Ą,	95,	100,	26
	reAS																																				
_	Sco																																				
	ä								0		0				0									2			0		0	2						0	4
	ΝQ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	٩	0	0	0
	Ŧ	0	5	5	0	13	0	3	6	8	m	9	ŝ	6	ŝ	20	6	0	6	10	0	0	6	0	9	4	80	2	9	0	13	20	17	4	~	1	20
	H H	0	5	5	0	7	0	7	.1	.2	Ŀ.	4	S	e.	ŝ	0	1	0	1	0	0	0	1	0	0	9	.2	80	4	0	7	0	e	9	m	6	2
	#Sa	0	-	5	~			-	5	-	-	0	-	2	_		2		0	6	~	8	-	10	-		8	~	-	_			N	-		-	44
	lem)	19,10	15,46	32,86	05,33	19,85	29,04	12,94	18,85	59,03	13,86	33,4(21,49	14,27	50,7	10,75	59,57	17,03	18,6(15,69	71,68	96,18	17,77	03,76	52,13	35,91	29,28	59,53	34,23	16,94	71,15	36,45	15,47	99,06	28,80	54,56	21,00
2	vg(⊵			6	a		84					10	5	2	러	ŝ	H	ŝ	2	÷	3	7		450		35	1	60				9	à	픤		- '	290
p-dsi	A (s	88	76	90	55	87	93	59	58	6	59	44	07	07	66	18	64	48	83	53	86	76	28	54	8	38	23	88	05	99	80	32	50	6	2	90	8
+-clc	Ĩ	146,	521,	749,	509,	854,	649,	556,	786,	470,	732,	071,	478,	654,	744,	010,	963,	233,	216,	158,	342,	719,	162,	865,	672,	902,	696,	591,	668,	221,	727,	010,	442,	542,	621,	370,	068,
idh	Sum		9	9	ŝ	17	4	9	11	1	ŝ	~	9	12	~	24	11		12	12		1	13	7	8	7	11	13	~		16	24	20	۵ ا	~	5	325
	be	100	75	75	100	35	100	85	55	60	85	70	75	55	75	0	55	100	55	50	100	100	55	٥	23	80	60	6	20	0	35	0	15	8	65	95	200
	Solv																																				2
	Scor																																				
	15	00'0	7,73	5,00	00'0	3,18	00'0	5,00	9,55	00'(5,00	00'0	00'9	00'9	3,64	3,18	5,82	00'0	00'9	00'0	00'0	00'0	00'9	00'0	00'(00'0	00'0	00'0	5,82	00'0	1,55	L,36	5,91	60,t	5,91	5,00	658
	SP2	10	6	7	100	73	10	80	6	9	õõ	7	2	5	6	73	7	10(5	5	10	10(5		2	8	61	6	6		õ	7	7	õ	6	6	7
	oreA																																				
-	ESc	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
	â			0	_	_		_	0		_			_	_		_		0		~	0		~		_	0	_	_	2	0		_			_	4
	ON#								-				ľ														-										
	5	0	5	5	0	13	0	3	9	8	m	5	S	6	ŝ	20	6	0	6	10	0	0	6	0	80	4	8	2	9	0	12	20	17	4	~	1	216
	#	20	15	15	20	7	20	17	11	12	17	15	15	11	15	0	11	20	11	10	13	20	11	0	12	16	12	18	14	0	8	0	3	19	с П	19	37
	S# (80	1	1	5	88	0	00	\$5	80.	65	80	69	00	4	4	12	4	09	80	5	4	5	6	5	1	80	33		4	11	65	68	ŝ	6	1	0 4
	Mem	18,6	15,3	585,7	103,6	19,8	129,1	12,5	18,8	69,1	13,8	02.1,4	220,6	238,6	149,3	540,9	158,4	311,5	218,6	115,5	246,4	793,6	17,7	503,4	62,3	178,6	129,2	047,€	34,8	16,9	64,4	507,8	145,8	198,4	28,6	55,6	294,0
1lr	Avg(-			õ					Ŧ				- 1					7(4		÷		9									33.
asp-c	r (ər),64	3,23	5,83	3,35	,11	5,18	9,39	9,21	1,31	5,53	3,66	7,45	9,72	1,09	0,14	-,99	5,95	6,17	,01	0,24	1,67	6,08	5,01	1,63	5,34	9,02	,56	5,35	L,53	6,25	0,43	00(9	2,03	00(2	1,64	00
flv-cl	ŗ,	150	6523	6745	3448	7885	4616	6549	1789	1474	5725	6828	6477	2679	8751	4010	1971	235	2146	2162	6170	1704	3166	2786	2733	7365	1789	3372	7475	221	6296	4010	0446	5162	8617	4954	6442
ic	Sun					-			1	-				-		2	1		1	-			-		-		1	-			1	2	2				32
	lved	10	75	75	100	35	10	85	55	99	8	75	52	55	35	0	55	100	55	3	65	100	55	ି	8	8	60	6	8	0	40	0	15	8	59	95	2185
	reSo																																				
	Sco																							_				_									
	015	00'00	97,73	75,00	00'00	78,18	00'00	35,00	70,91	50,00	35,00	75,00	75,00	55,00	98,64	78,15	76,82	00'00	55,00	50,00	55,00	00'00	55,00	0,0	0,00	30,00	50,00	90,06	96,82	0,00	20'62	71,36	75,00	97,27	5,91	95,00	2636
	ASP2	ä	0,		ä	1	ä	~			Ű	1		υ,	0,	`		10	υ,	Ξ,	Ű	10	.,		Ĩ	~	ę	0,	0,			`	1			01	
	core																																				
-	s	sy				Η	Η			Η						Η		H	H	Η	pr		۲									۲	Η	Η	Η		tal
		ewor	ß				Bu								~						ching	Ding				s											₽
		rame	arnir	tion	ç		swer		c			Bu	s		rning						Mat	ea sor	ASP			r Set					c						
		ical F	rkLe	gura	zatic	ife	y An		zatic			edul	HOL		k Leã			sis			ttern	ial Rt	tive.			JSWe				nies	uctic	s	erso				
		alecti	etwo	Confi	otimi	Still	Quer	Ition	nimi.	aphs	uring	Sch	with		WOL	anb.		BIDO		ts	n Pat	Spat	sjunc		bots	nt Ar		riage	a,	mpa.	Contr	thesi	desp	tion	ming		
		iă	ž	p p	ő	ed.	int,	ocâ	S Mi	ő	e	enta	lo	무	Net		F	I Di	tery	r Uni	tatio	ative	m Di	bility	et Rc	here.	u.	Marı	Tre	gicCo	ree (Syn	ng Sé	Loca	strea		
	E	t	1	Ĕ.	a।	5	121		2	_ = 1	0	1 Z '		- C	~	1.27.1	1.000			1.11						_											
	omain	ostract	ayesia n	mbine	mplex	nnect	on siste	ew Al	ossing	acefu	aph C	crem	ight	byrin	arkov	aximi	axSA	inima	shuuc.	irtne	rmu	ualité	opue	achi	访	racc	kob	able	eine	rate	pert	sten	aveli	lves	deo	sit-al	

Fig. A 1: Detailed results for IDLV-CLASP-DLV, IDLV+-CLASP-DLV, IDLV+-S.

	۳	20	0	0	20	0	20	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	20	20	0	13	0	0	0	20	0	0	0	18	0	0	5
	0 Å	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	12	0	0	0	0	7	0	16	0	0	0	14	0	0	0	0	54
	2#	0	5	9	0	13	0	4	80	7	7	9	9	12	8	0	6	0	L1	9	0	6	0	0	5	0	5	0	8	0	5	5	19	2	8	7	22
	1 #T0	0	15	7	0	7	0	16	12	13	13	2	14	80	12	0	=	0	8	2	80	1	0	0	00	0	6	4	12	0	5	1	4	0	12	13	t3 2:
) #Sc	26	80	0t	0	35	22	10		14	56	0	16	12	13	75	20	55	90	10	32	12	63	52	9	53	1	73	. 62	32	52	37	52	57	12	76	00 24
0	(Mem	13,5	24,0	3321,4	42,5	33,5	216,8	11,6	19,8	58,1	22,2	2672,0	579,5	974,7	578,1	593,7	303,7	159,6	2113,0	377,5	7789,8	2027,5	0,2	1572,2	209,1	6301,6	401,0	2266,7	74,7	9	393,6	1824,3	694,6	1677,6	45,1	350,7	7753,0
+clas	Avg	2	6	2	6	0	4	2	8	2	2	00	0	9	0	0	m	4	2	m	਼ਿ	9	0	9	80	9	ŝ	1 6	ņ	4	80	5 1	3	e,	9	2	0
cycpt	(Time	35,4	5897,8	1689,8	11,1	3612,1	253,8	3868,6	0582,0	2036,8	2119,3	3967,4	3626,2	5435,9	2121,2	1012,7	1364,2	118,6	5405,6	3314,7	2619,6	5177,9	0,0	1655,6	3583,4	2093,7	5,999,7	5239,5	2085,4	1,8	3163,2	5497,1	919,2	8613,1	0188,3	528,7	5841,0
lp2(Sum	0	2	5 1/	0	5 18		~	10	5 12	5	1	0	16	1	5	5	0	1	1	0	51	0	0	ñ	0	5	0	1		5	2	0 19	0	10	5 10	5 345
	Solved		7	3		3		~	9	9	9	ß	~	4	9		Ś		4	20	4	ι. Ω			4		4	2	9		5		2	-	9	9	121
	Score!																																				
	2015	00'0	94,09	35,00	00'0	58,18	00'0	30,00	76,36	55,00	55,00	50,00	70,00	40,00	87,73	70,91	75,00	00'0	40'00	50,00	40,00	55,00	00'0	0,00	40,00	0,00	45,00	20,00	30,91	00'0	47,73	5,00	32,73	0,00	87,27	55,00	1526
	eASP2					- /				-		-								-		-														-	
_	Scor	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	8
	0# 0	0 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0 2	0 2	0	7 1	0	9	0	0	0	0	0	0	0	0	5 15
	W#	0	4	2	0	8		m	6	2	4	~	5	80	2	0	6	0	80	6	0	6	0	0		0	6	0	9	0	4	0	9	4	9	3	2
	1 #10	0	. 9	5	0	7 1	0	2	न	e.	. 9	m	5	2	8	0	न	0	2	-	- 00	न	0	0	m	0	न	4	4	0	9	0 2	4	. 9	4	7	5 20
) #So	11	36 1	82 1	δ	6,	90	88	1	1 10	1	1	0	13	1 96	6.	1	33	1 1	6	5	0	0	ŝ	55	2	1	55	1	2	9	09	99	1	1	1 1	00 30
0	(Mem	14,0	20,8	794,8	42,7	22,7	218,0	10,9	17,4	60,0	13,0	2672,0	201,4	257,7	577,9	599,1	278,9	158,8	250,9	109,2	7794,1	1324,1	0,0	1572,2	58,6	5297,2	128,0	262,6	31,5	6,2	70,1	622,6	559,6	557,4	24,3	105,4	7736,0
p+clas) Avg	τ.	80	33	5	00	9	8	69	6,	33	00	2	32	\$2	E	80	9	.3	-	0		0	5	5	1	ស្ល	1 1	0.0	99	2	5	98	0	09	92	3
rcycas,	(Time	35,7	5482,5	7211,3	10,8	7197,8	261,4	5654,9	2061,6	3865,7	5783,5	9051,5	5426,1	1445,8	4760,3	4010,2	1419,9	119,4	1628,1	1623,1	3047,5	3002,6	0'0	1942,7	2109,3	2260,5	1756,8	5457,7	7566,5	1,1	5918,7	4011,3	919,8	5522,4	7717,6	5683,5	2670,0
Ip20	Sum	0	0	5	0	5		5	5	5 10	0	5	5	1	0	5 0	5	0	1	5	0	5	0	0	5	0	5	0	0		1	0 2	1	0	0	2	5 29
	Solvec		~	7		3		~	ŝ	9	~	°	-	9	6		ŝ		9	Lo.	4	Lo.			۹ ٩		ŝ	2	7		m		2	~	7	8	152
	Score																																				
	2015	0,00	97,73	75,00	0,00	68,64	0,00	85,00	71,36	65,00	80,00	65,00	75,00	60,00	00'00	78,18	75,91	0,00	60,00	55,00	40,00	55,00	00'0	0,00	65,00	0,00	55,00	20,00	95,91	0,00	87,73	77,27	80,00	94,55	96,82	85,00	1964
	reASP														1																						
_	E Sco	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	₽ 0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7
	¥ ₩	5	4	L1	0	10	0	5	-	13	12	=	~	7	50	6	-	0	10	12	0	2	-	0	2	6	0	13	4	0	15	03	19	5	6	7	66
	ol #T0	15	16	9	20	10	11	ŝ	19	2 2	80	6	13	13	0	11	19	20	10	00	50	18	6	0	5	11	6	2 2	9	0	5	0	4	15	11	13	62 28
	n) #S	561	5,9	72	395	15	\$75	59	35	86	35	05	02	561	.15	315	55	84	.23	44	5	015	805	92	05	35	18	.25	85	955	25	615	65	99	7,4	59	00 3
>	rg(Mer	199,3	17	2162	229,8	113,7	4506,8	86	22.7	299	69,2	1335	388	448,4	1025	1026,8	478,3	1508	546	506	244	2834,9	75,8	4504	241	3534,2	403	1983	415,7	16,9	236,5	635,6	1821,3	383	24	481	43191
lp-dsp.	e) A	,38	,08	,62	98	,24	,26 1	59	,40	,23	,06	59,	,45	98	,76	60,	,70	00,	,78	58,	,65	,47	,17	,63	.93	,05	68	.95	.91	Ę	,56	,74	,73	,86	88	,22	00
dlv+-w	m(Tim	6348	6213	16055	2600	13355	3810	19825	1242	17784	16245	15698	9391	9624	24008	14620	2834	486	14385	17701	317	6069	14581	2848	15772	11284	12895	20502	17629	223	18357	24010	19475	7499	10804	9312	03820
	ed Su	75	80	45	8	50	55	22	95	35	40	45	65	65	0	55	95	8	50	4	8	6	45	0	50	55	22	35	30	0	22	0	20	75	55	65	10 4
	eSolve				-													-			-																18
	Scor	3	7	0	0	5	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	T.	0	5	9	8	0	5	0	1
	P2015	75,8	87,2	45,0	100,0	75,4	55,0	25,0	100,0	35,0	40,0	45,0	65,0	65,0	40,0	70,0	96,3	100,0	50,0	40,0	100,0	90'06	45,0	0,0	50,0	55,0	50,0	35,0	55,9	0,0	55,4	61,3	88,1	85,0	65,4	65,0	211
	oreAS																																				
-	Sc	sy.																			50																tal
		newoi	ning	c			ering								Bu						atchin	ioning	4			ets											Ę
		al Frai	k Leari	uratio	ation	fe	Answ		ation			duling	Holes		Learn			s			ern M	alReas	ive AS			swer S				ies	ction		rson				
		alectic	etwor	Config	pt im iz,	Still Li	Query	ation	inimiz,	aphs	uring	I Sche	with		twork	ique		agnosi		its	n Patt	Spatié	sjuncti		obots	nt Ans		riage	a	mpan	Contru	thesis	alespe	ntion	ming		
	ain	act Di	sia n N	bined (o xalc	ected	istent	Alloca	ing M	eful Gr	h Colo	menta	t Tour	inth	ov Ne	mal Cl	SAT	nal Di	ystery	ner Uni	utatio	tative	om Di:	Jabilit	het Rc	ohere	ban	e Mari	er Tre	egicCo	rtree (m Syn	iling Si	ss Loca	o Strea	all	
		- ∓	نۇ ا	H,	É	18	IS I	l ≷	l S	act	l de	18	1	ā	l ¥	axir	axs	Ē	Ê	١£	١E	iai i	ġ	ac	lö	rac	- Ko	able	ein	raț,	l a	ste	ave	Š	je	sit-	

Fig. A 2: Detailed results for IDLV-WASP-DLV, LP2ACYCASP+CLASP, LP2ACYCPB+CLASP.

75 6487 75 6487 95 63325 98 3325 93 2342 95 632 93 2325 93 24 93 24 93 25 93 25 95 95 95 95 95 95 95 95 95 95 95 95 95
75 6465 75 6787 95 3955 95 3955 95 3325 95 2346 95 2323 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 2232 95 232 95 232 95 25 75 75 75 75
75 678 95 395: 95 395: 80 832i 80 832i 90 24i 75 692i 95 223i 95 223i 95 223i 96 933i 97 935 98 65 99 1255 60 1255 75 75
3953 3953 8326 8326 933 993 993 993 993 1255 1255
8328 24(692 993 993 993 993 1255 1255
4 2 8 8 8 9 9 9
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2555
9933 679 1255 6 760
80 6793 60 12555 75 7601
0 12555 5 760
75 760
100/ 0/
20 U
0 20
934,57 U 888,66 0
- I i
23142,17
0 23142,17
U U,UU 0 24011,34 1 0 0,000 0 23142.17 1
0 0 0 0,000 0 24011,34 1 0 0 0 0.000 0 23142.17
1 0 0 0 0.000 0 24011,34 1 0 24011,34 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
8 1.21 U U U U U 0,001 0 24011,34 1
- 10,114 40 10 40 10 10 10 10 10 10 10 10 10 10 10 10 10
in training the second of the second se

Fig. A 3: Detailed results for LP2ACYCSAT+CLASP, LP2MIP, LP2NORMAL+CLASP.

moto moto <th col<="" th=""><th></th><th>3O# 0</th><th>0 5</th><th>0</th><th>0 0</th><th>0</th><th>0 0</th><th>0</th><th>0</th><th>0 0</th><th>0 0</th><th>0 0</th><th>0</th><th>0</th><th>0</th><th>0 0</th><th>0 0</th><th>0</th><th>0</th><th>0 0</th><th>0</th><th>0</th><th>0</th><th>0 0</th><th>0 20</th><th>0 0</th><th>5 0</th><th>1 0</th><th>0 0</th><th>0 0</th><th>0 20</th><th>0</th><th>0 0</th><th>0 0</th><th>0</th><th>0 0</th><th>0</th></th>	<th></th> <th>3O# 0</th> <th>0 5</th> <th>0</th> <th>0 0</th> <th>0</th> <th>0 0</th> <th>0</th> <th>0</th> <th>0 0</th> <th>0 0</th> <th>0 0</th> <th>0</th> <th>0</th> <th>0</th> <th>0 0</th> <th>0 0</th> <th>0</th> <th>0</th> <th>0 0</th> <th>0</th> <th>0</th> <th>0</th> <th>0 0</th> <th>0 20</th> <th>0 0</th> <th>5 0</th> <th>1 0</th> <th>0 0</th> <th>0 0</th> <th>0 20</th> <th>0</th> <th>0 0</th> <th>0 0</th> <th>0</th> <th>0 0</th> <th>0</th>		3O# 0	0 5	0	0 0	0	0 0	0	0	0 0	0 0	0 0	0	0	0	0 0	0 0	0	0	0 0	0	0	0	0 0	0 20	0 0	5 0	1 0	0 0	0 0	0 20	0	0 0	0 0	0	0 0	0
Interpretation Description Example region Example r		OW# C	0	5	5	0	5	0	5	2	7	4	~	~	4	7	17	00	0	11	6	-	0	4	0	6	0	8	12	11	0	13	20	17	5	9	e	
Intervalution Secondary Se	-	sol #T0	15	15	15	20	15	20	15	18	13	16	۲ ۲	13	16	13	8	12	20	6	11	19	20	16	0	11	15	11	80	6	0	~	0	m	15	14	17	
metro processes metro processes <t< td=""><td>ŀ</td><th>em) #5</th><td>2,11</td><td>7,06</td><td>0,08</td><td>9,87</td><td>3,17</td><td>9,33</td><td>4,03</td><td>6,16</td><td>1,75</td><td>5,41</td><td>9,89</td><td>6,96</td><td>3,35</td><td>5,07</td><td>3,43</td><td>1,74</td><td>2,43</td><td>2,83</td><td>8,23</td><td>8,07</td><td>9,15</td><td>4,99</td><td>0,41</td><td>3,87</td><td>8,96</td><td>4,51</td><td>2,68</td><td>7,70</td><td>4,43</td><td>5,40</td><td>2,43</td><td>8,28</td><td>4,47</td><td>7,60</td><td>3,92</td></t<>	ŀ	em) #5	2,11	7,06	0,08	9,87	3,17	9,33	4,03	6,16	1,75	5,41	9,89	6,96	3,35	5,07	3,43	1,74	2,43	2,83	8,23	8,07	9,15	4,99	0,41	3,87	8,96	4,51	2,68	7,70	4,43	5,40	2,43	8,28	4,47	7,60	3,92	
	a	Avg(M	4 2	9 3	7 93	2	2	1 981	~	5 2	1	5 2	5 251	12	4 21	9 48	31	21	3 25	8 56	1	5 152	66	5 2	1 571	7 10	2 592	5 84	169	3	4 2	4	7 21	2	7 52	5 4	8	
	me-as	m(Time)	72,5-	6826,59	7450,9	1217,0	8290,4	4650,9:	9084,4	3494,6	10880,6	9030,6	9745,4	8652,7	8252,6	11189,19	22918,9	9956,6	142,1	14273,31	12076,1	3654,4	3605,4	10684,6	3601,7	14190,1	2883,2	11384,20	19502,20	13537,6	224,1	16649,4	24011,5	20457,2	6541,5	8371,6	5914,80	
Instructure Instructure/instructure Instructure Instructure <td>Ī</td> <th>olved Su</th> <td>75</td> <td>75</td> <td>75</td> <td>100</td> <td>75</td> <td>100</td> <td>75</td> <td>06</td> <td>65</td> <td>80</td> <td>53</td> <td>65</td> <td>80</td> <td>65</td> <td>15</td> <td>60</td> <td>100</td> <td>45</td> <td>55</td> <td>95</td> <td>100</td> <td>80</td> <td>0</td> <td>55</td> <td>75</td> <td>55</td> <td>40</td> <td>45</td> <td>0</td> <td>35</td> <td>0</td> <td>15</td> <td>75</td> <td>70</td> <td>85</td>	Ī	olved Su	75	75	75	100	75	100	75	06	65	80	53	65	80	65	15	60	100	45	55	95	100	80	0	55	75	55	40	45	0	35	0	15	75	70	85	
International conditional condi		ScoreSc				_																			0													
Antical and control (ADDIT) Defermine (ADDIT)<		SP2015	75,00	82,73	75,00	100,00	97,27	100,00	75,00	95,91	65,00	80,00	65,00	65,00	80,00	89,55	86,36	75,45	100,00	45,00	55,00	95,00	100,00	80,00	00'0	55,00	75,00	55,00	40,00	62,27	0,00	92,73	16'06	40,45	93,64	89,55	85,00	
And the second of and the second of any of the second of		ScoreA																																				
Interpretation $p_{200mol log231ets35}$ $p_{200mol log231ets35}$ $p_{200mol log231ets35}$ $p_{200mol log314}$ $p_{200mol log3144}$ <		#OE	20	20	0	20	20	20	0	20	0	0	°	0	0	20	20	20	20	0	0	0	0	20	20	0	13	0	0	. 19	20	20	20	20	10	20	0	
And the stand And the		OW#	0	0	0	0	0		0		0	Ĩ	5	0	0	0	0	0	0	0	0	1	0	0	0	0	7	0	16	-	0	0	0	0	2			
Instructure		#T0	0	0	18	0	0	0	ŝ	0	6	4	Î	17	13	0	0	0	0	6	13	0	6	0	0	9	0	3	0	0	0	0	0	0	0	0	0	
Antimulation Accommoning/2cti-strast Accommonind/2cti-strast Accommoning/2cti-strast </td <td></td> <th>#Sol</th> <td>9</td> <td>0</td> <td>80</td> <td>0</td> <td>3</td> <td></td> <td>15</td> <td>2</td> <td>11</td> <td>16</td> <td></td> <td>5</td> <td>1</td> <td>8</td> <td>8</td> <td>4</td> <td>0</td> <td>3 11</td> <td>4</td> <td>00</td> <td>E</td> <td>0</td> <td>4</td> <td>1 14</td> <td>2 2</td> <td>7 15</td> <td>4</td> <td>5</td> <td>2</td> <td>0</td> <td>4</td> <td></td> <td>0</td> <td></td> <td>202</td>		#Sol	9	0	80	0	3		15	2	11	16		5	1	8	8	4	0	3 11	4	00	E	0	4	1 14	2 2	7 15	4	5	2	0	4		0		202	
Answers Optimination Optimization	bu,	vg(Mem)	14,05	18,0	1815,3,	43,18	3,8	219,2	32,6	0,5.	89,47	27,15	11956,0	628,6	405,9.	577,9,	456,54	346,2.	160,0	337,5:	125,04	7796,04	1244,1.	0'0	1572,2	69,65	6294,9	203,7	12269,5-	803,1.	5,9.	40,4	2876,0-	1502,8.	7290,54	681,2	148.1	
Ip/2007 <	at+lingeli.	Time) A	35,14	5,28	761,86	11,58	0,27	249,97	334,19	0,04	988,28	724,25	546,00	586,42	449,95	399,47	415,01	164,53	139,49	045,16	956,14	594,82	384,44	00'0	687,58	660,58	031,75	729,55	654,04	514,61	1,74	21,16	055,48	604,33	487,44	420,06	317.46	
Interpretation Interpreaction Interpretation Interpr	Ip2si	of Sum(0	0	10 21	0	0	0	75 7	0	55 115	80 8	5	15 20	35 18/	0	0	0	0	55 12(35 16	40 2	55 13	0	0 4(70 10(0 2(65 10	20 64	0	0	0	0 1(0	2	0	00	
Instruction		oreSolve																																				
Internation Internation <thinternation< th=""> <thinternation< th=""></thinternation<></thinternation<>		015 Sc	00'0	00'0	00'0.	00'0	00'0	00'0	5,00	00'0	5,00	30,00	5,00	5,00	15,00	00'0	00'0	00'0	00'0	22,00	15,00	10,00	55,00	00'0	00'0	00'04	00'0	55,00	00'0;	00'0	00'0	00'0	00'0	00'0	00'0	00'0	00.00	
Instrument (a) Deziment (a) Deziment (a) Deziment (a) Deziment (a) Deziment (a) Deziment (a) Dezimation Dezimation <thdezimation< th=""></thdezimation<>		coreASP2										Ĩ																,									19	
Anticipation Construction (2) Secretioned (2) Secretio	7	#OE S	6	9	0	0	80	20	0	12	0	0	0	0	0	0	0	m	20	0	0	0	0	0	20	0	0	0	0	m	20	-	0	2	-	11	0	
gloomnol statistical frameworks gloomnol (spatial)	ľ	* OW#	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	12	0	0	0	0	7	0	16	1	0	0	14	2	12	0	C	
Incommentant Incommentant<	ļ	#T0	11	14	20	19	12	0	4	8	14	~	0	17	20	20	20	17	0	17	12	0	00	6	0	13	0	13	4	16	0	19	9	16	~	6	6	
Antimical Control And Contrel And Contrel And Control And Contrel And Contrel And Contrel A	-) #Sol	4 0	7 0	0 6.	1	2 2	0 6	16	0	8 6	5 13	1	5	0 6	3 0	3 0	0 2	8	5 3	8	12 8	12	11 0	.6 C	6 7	5 13	1 7	(5 C	3 0	8 C	10	0 6;	5 0	17	0	11	
Alternation ScoreASPOILS ScoreASPOILS </td <td>ts+sts</td> <th>\vg(Mem</th> <td>134,6</td> <td>107,5</td> <td>2110,7</td> <td>129,7</td> <td>57,8</td> <td>216,9</td> <td>28,8</td> <td>51,4</td> <td>82,9</td> <td>37,2</td> <td>12111,1</td> <td>371,4</td> <td>316,1</td> <td>705,7</td> <td>456,4</td> <td>580,6</td> <td>694,3</td> <td>456,6</td> <td>183,3</td> <td>7789,9</td> <td>5476,6</td> <td>50,3</td> <td>1572,2</td> <td>179,6</td> <td>6284,1</td> <td>351,7</td> <td>12211,3</td> <td>1315,3</td> <td>5,7</td> <td>167,5</td> <td>10507,4</td> <td>2795,5</td> <td>10720,8</td> <td>2390,1</td> <td>306.6</td>	ts+sts	\vg(Mem	134,6	107,5	2110,7	129,7	57,8	216,9	28,8	51,4	82,9	37,2	12111,1	371,4	316,1	705,7	456,4	580,6	694,3	456,6	183,3	7789,9	5476,6	50,3	1572,2	179,6	6284,1	351,7	12211,3	1315,3	5,7	167,5	10507,4	2795,5	10720,8	2390,1	306.6	
Instruction	mal+lp2s	(Time) A	072,20	187,79	010,70	277,47	172,70	251,09	862,65	925,64	836,60	822,94	132,55	462,64	08, 600	19,600	011,96	401,27	482,40	153,61	563,03	462,25	714,84	491,48	788,74	821,52	099,16	647,30	935,16	665,24	1,73	875,08	510,10	380,14	199,92	428,65	745.58	
alt ScoreASP2015 ScoreASP2015 rate Date ctrcal Ferneworks 0.00 0.00 bined Configuration 0.00 0.00 bisted Configuration 0.00 0.00 sitent Configuration 0.00 0.00 sitent Configuration 0.00 0.00 sitent Configuration 0.00 0.00 sitent Configuration 0.00 0.00 neneral Scheduling 0.00 0.00 fragmes 0.00 0.00 <td>Ip2non</td> <th>ed Sum(</th> <td>0 17</td> <td>0 21</td> <td>0 24</td> <td>5 23</td> <td>0 17</td> <td>0</td> <td>80 6</td> <td>0 12</td> <td>30 18</td> <td>65 12</td> <td>5 6</td> <td>15 20</td> <td>0 24</td> <td>0 24</td> <td>0 24</td> <td>0</td> <td>*0</td> <td>15 22</td> <td>40 19</td> <td>40 2</td> <td>60 10</td> <td>55 13</td> <td>0 4</td> <td>35 18</td> <td>65 2</td> <td>35 19</td> <td>0 7</td> <td>0 19</td> <td>0</td> <td>0 23</td> <td>0 8</td> <td>0 20</td> <td>0</td> <td>0 11</td> <td>55 21</td>	Ip2non	ed Sum(0 17	0 21	0 24	5 23	0 17	0	80 6	0 12	30 18	65 12	5 6	15 20	0 24	0 24	0 24	0	*0	15 22	40 19	40 2	60 10	55 13	0 4	35 18	65 2	35 19	0 7	0 19	0	0 23	0 8	0 20	0	0 11	55 21	
ain Scons.52015 S		oreSolve																																				
alin scorebar alin scorebar fract Date ctrical Fram eworks bined Configuration bined Configuration beeted Still Ute stream Careford stream Careford stream Careford stream Careford fraction the Colouring Minimutation the Colouring Minimutation the Colouring Minimutation the Colouring and Chework Learning Minimutation the Colouring and Chework Learning fraction and Chework Learning and Chework Learning fraction and Chework Learning and Chework C	ŀ	2015 St	00'0	0,00	0,00	5,00	0,00	00'0	80,00	0,00	30,00	65,00	5,00	15,00	0,00	00'0	0,00	0,91	*0	15,00	40,00	40,00	60,00	55,00	00'0	35,00	65,00	35,00	0,00	00'0	0,00	00'0	00'0	0,00	00'0	0,00	55.00	
aliant and the second s		scoreASP:																						- '														
and and match and an entropy of the anti- phenological phenological anti- phenological phenological anti- meteral Statistical anti- meteral Statistical anti- anti- stratical anti- stratical anti- stratical anti- anti	1	S	eworks	ß				Bui														ching	guir				s											
international and a sector of the sector of			al Frame	Learnin	Ination	tion	e	Answer		tion			Juling	Holes		earning						ern Matu	Reason	ve ASP			wer Set:				es	tion		son				
and the second s			ectica	etwork	Configu	ot im iza	Still Life	Query /	tion	nimiza	aphs	uring	I Sched	with H		twork L	anb		gnosis		ts	n Patte	Spatial	sjunctiv		bots	nt Ansv		iage		mpanit	ontruc	thesis	le sper:	tion	ming		
			- E			-			12	18	1.5	ιō	41	13		ē	15		÷۳	1 >	Ē	1.2	υ	1÷	£	l H	5		2	ð	18	12	1 5	ŝ	18			

Fig. A 4: Detailed results for LP2NORMAL+LP2STS+STS, LP2SAT+LINGELING, ME-ASP.