Design and Results of the Fifth Answer Set Programming Competition☆

Francesco Calimeri[a], Martin Gebser[b,1], Marco Maratea[c,*], Francesco Ricca[a]

[a]*Dipartimento di Matematica e Informatica, Università della Calabria, Italy*
[b]*Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University, Finland*
[c]*Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi, Università di Genova, Italy*

Abstract

Answer Set Programming (ASP) is a well-established paradigm of declarative programming that has been developed in the field of logic programming and nonmonotonic reasoning. Advances in ASP solving technology are customarily assessed in competition events, as it happens for other closely related problem solving areas such as Boolean Satisfiability, Satisfiability Modulo Theories, Quantified Boolean Formulas, Planning, etc. This paper reports on the fifth edition of the ASP Competition by covering all aspects of the event, ranging from the new design of the competition to an in-depth analysis of the results. The paper comprises also additional analyses that were conceived for measuring the progress of the state of the art, as well as for studying aspects orthogonal to solving technology, such as the effects of modeling. A detailed picture of the progress of the state of the art in ASP solving is drawn, and the ASP Competition is located in the spectrum of related events.

Keywords: Answer Set Programming, Solver Competition, Computational Logic

☆This is an extended and revised version of (Calimeri et al., 2014a).

*Corresponding author

Email addresses: calimeri@mat.unical.it (Francesco Calimeri), martin.gebser@aalto.fi (Martin Gebser), marco@dibris.unige.it (Marco Maratea), ricca@mat.unical.it (Francesco Ricca)

[1]Also affiliated with the University of Potsdam, Germany.

1. Introduction

Answer Set Programming (ASP) (Baral, 2003; Brewka et al., 2011; Eiter et al., 1997, 2000, 2009; Gelfond and Leone, 2002; Gelfond and Lifschitz, 1988, 1991; Lifschitz, 2002; Marek and Truszczyński, 1999; Niemelä, 1999) is a well-established declarative programming approach to knowledge representation and reasoning, proposed in the area of logic programming and non-monotonic reasoning. The idea of ASP is to represent a given problem by means of a logic program whose stable models or answer sets correspond to solutions, and then to use an ASP solver for computing solutions. The availability of high-performance implementations, e.g. (Alviano et al., 2013b; Dal Palù et al., 2009; Gebser et al., 2012a, 2013; Giunchiglia et al., 2006; Janhunen et al., 2006; Leone et al., 2006; Lin and Zhao, 2004; Liu et al., 2012; Mariën et al., 2008; Simons et al., 2002), made ASP a powerful tool for developing advanced applications. Nowadays, ASP has been employed in many research areas ranging from Artificial Intelligence to Databases and Bioinformatics; moreover, it has already been used in industrial systems, e.g. (Nogueira et al., 2001; Ricca et al., 2012; Tiihonen et al., 2003).

The advances in ASP solving technology are customarily assessed in competition events (Brewka et al., 2002; Gebser et al., 2007; Denecker et al., 2009; Calimeri et al., 2014c; Alviano et al., 2013a), as it happens for other closely related problem solving areas such as Boolean Satisfiability (SAT), Satisfiability Modulo Theories (SMT), Quantified Boolean Formulas (QBF), and Planning, to mention a few. ASP Competitions are (usually) biennial events; however, the fifth edition departed from tradition and took place in 2014, affiliated with the 30th International Conference on Logic Programming (ICLP 2014), in order to join the FLoC Olympic Games at the Vienna Summer of Logic.

The Fifth ASP Competition (Calimeri et al., 2014b) featured a revised setup with novelties involving every aspect of the design from the definition of tracks to the scoring scheme. The new design aims at lowering the efforts of participating in the event, and further pushes the adoption of the recent standard language ASP-Core-2 (Calimeri et al., 2013) introduced in 2013. Indeed, in 2013, the ASP-Core-2 language was not fully supported yet by most implementations, and/or the participants did not have enough time to integrate new language features in a completely satisfactory way. Taking these considerations into account, the Fifth ASP Competition was based on the System track of the 2013 edition,[2] reusing the

[2] In the 2013 edition, the "System track" was conceived to assess ASP systems on a fixed set of problem encodings. A detailed comparison with the 2013 format is reported in Section 7.

available benchmarks but also adding novel problem encodings.

The benchmark domains were classified by the language features used in encodings (e.g. choice rules, aggregates, presence of queries), rather than by problem "complexity" considered in past events (Calimeri et al., 2012). The competition tracks were devised in accordance with the new benchmark classification and by carefully considering the increasing effort needed for the implementation of specific language features. This was intended not only to widen participation, but also to properly analyze benchmarks and solvers' performance from the perspective of the language. Concerning participants, competitors in the 2013 edition as well as newcomers were invited to participate.

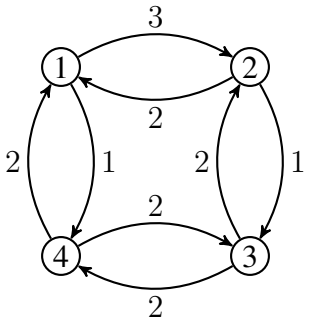
This paper provides a comprehensive report of the Fifth ASP Competition along with an in-depth analysis of results, originally published on the competition homepage (Calimeri et al., 2014b). For one, novel problem encodings were devised to furnish an extended benchmark collection and to assess the impact of modeling on the ASP solving process. For another, the most successful systems submitted in 2013 were run against new versions to evaluate the progress. Moreover, the competition setup is compared to those in other problem solving areas. As a result, this paper draws a detailed picture of the state of the art in ASP solving, and locates the ASP Competition in the spectrum of related events.

The remainder of this paper is structured as follows. Section 2 provides the reader with proper preliminaries about the ASP-Core-2 language. The competition setup is discussed in Section 3, while Sections 4 and 5 present the domains and ASP systems, respectively, taking part in the competition. Section 6 announces the competition winners and analyzes the results. In Section 7, the competition design is compared to those of previous editions as well as related events. Section 8 concludes the paper, also pointing out some recommendations for future editions.

2. The ASP-Core-2 Language: Syntax and Semantics

The input language for competition benchmarks, represented in terms of a uniform (first-order) problem encoding along with (ground) facts specifying a problem instance, follows the ASP-Core-2 standard (Calimeri et al., 2013). The syntax of ASP-Core-2 includes elements from classical first-order logic, i.e. terms, atoms, and connectives, as well as extensions like integer arithmetic, aggregates, weak constraints, and queries. These constituents provide a conceptually simple yet powerful modeling language for expressing computational problems with diverse features and complexity. For example, a data representation of the directed graph with arc costs displayed in Figure 1(a) is shown in Figure 1(b), where nodes, arcs,

(a) A directed graph



(b) Fact representation of the graph in (a)

$$
\begin{array}{lll}
arc(1,2) \leftarrow & cost(1,2,3) \leftarrow & node(1) \leftarrow \\
arc(2,1) \leftarrow & cost(2,1,2) \leftarrow & node(2) \leftarrow \\
arc(2,3) \leftarrow & cost(2,3,1) \leftarrow & node(3) \leftarrow \\
arc(3,2) \leftarrow & cost(3,2,2) \leftarrow & node(4) \leftarrow \\
arc(3,4) \leftarrow & cost(3,4,2) \leftarrow & \\
arc(4,3) \leftarrow & cost(4,3,2) \leftarrow & \\
arc(1,4) \leftarrow & cost(1,4,1) \leftarrow & \\
arc(4,1) \leftarrow & cost(4,1,2) \leftarrow &
\end{array}
$$

Figure 1: An example graph with arc costs that constitutes a TSP instance

and their associated costs are specified in terms of facts over corresponding predicates. Given such instance data, the following ASP-Core-2 program encodes the well-known Traveling Salesperson Problem (TSP):

$$
\begin{array}{rcll}
\{hc(X,Y) : arc(X,Y)\} = 1 & \leftarrow & node(X) & (1) \\
 & \leftarrow & node(Y), \#\texttt{count}\{X : hc(X,Y)\} \neq 1 & (2) \\
reach(X) & \leftarrow & \#\texttt{min}\{Y : node(Y)\} = X & (3) \\
reach(Y) & \leftarrow & reach(X), hc(X,Y) & (4) \\
 & \leftarrow & node(X), \textbf{not}\ reach(X) & (5) \\
 & :\sim & hc(X,Y), cost(X,Y,C)\,[C@1, X, Y] & (6)
\end{array}
$$

Without going into details yet, note that the rules in (1) and (2) express that, for every node, exactly one outgoing and one incoming arc must be part of a Hamiltonian cycle, i.e. a round trip that visits each node, and the selected arcs are represented by (true) atoms over the predicate hc. Then, starting from some arbitrary yet fixed node, taken to be the lexicographically smallest one in (3), further nodes reachable via selected arcs are derived by the rule in (4). Given that a round trip must visit all nodes, the so-called integrity constraint in (5) denies isolated subcycles, indicated by a node whose corresponding atom over the predicate $reach$ does not hold. In turn, every node must be reached from the starting node, so that the true atoms over the predicate hc provide a Hamiltonian cycle. The objective of finding a Hamiltonian cycle with minimum cost is formulated in terms of the weak constraint in (6), which penalizes any selected arc by its cost. The (unique) optimal round trip for the example graph in Figure 1(a) includes the arcs $(1, 4)$,

$(4,3)$, $(3,2)$, and $(2,1)$, yielding an accumulated cost of 7. However, note that the first-order ASP-Core-2 program in (1)–(6) uniformly encodes the Optimization problem TSP for any directed graph specified by facts like those in Figure 1(b).

In what follows, we review the syntax and semantics of ASP-Core-2 programs, laying the ground for the classification of competition benchmarks into tracks (see Section 3). As usual, *terms* are composed of constants, variables, and functions. Dedicated *arithmetic* terms have the form $-(t)$ or $(t \diamond u)$ with $\diamond \in \{+, -, *, /\}$. Terms and $\prec \in \{<, \leq, =, \neq, >, \geq\}$ are used to construct three kinds of *atoms*:

- *classical* atoms $p(t_1, \ldots, t_n)$ and $\neg p(t_1, \ldots, t_n)$ for a predicate name p and terms $t_1, \ldots, t_n$,
- *built-in* atoms $t \prec u$ for terms t and u, and
- *aggregate* atoms $\#\mathbf{agg}\{e_1; \ldots; e_k\} \prec u$ for $\#\mathbf{agg} \in \{\#\mathtt{count}, \#\mathtt{sum}, \#\mathtt{max}, \#\mathtt{min}\}$, where e_i $(1 \leq i \leq k)$ is an aggregate element $t_1, \ldots, t_n : l_1, \ldots, l_m$ in which $t_1, \ldots, t_n$ are terms and $l_1, \ldots, l_m$ are *naf-literals*, i.e. built-in atoms $t \prec u$ or expressions a and $\mathbf{not}\ a$ for classical atoms a.

Note that $\mathbf{not}$ stands for *default negation*, and *literals* in general include naf-literals as well as a and $\mathbf{not}\ a$ for aggregate atoms a.

ASP-Core-2 programs consist of rules, possibly accompanied by weak constraints or a query. A *rule* r is of the form $a_1 \,|\, \ldots \,|\, a_m \leftarrow b_1, \ldots, b_n$, where $a_1, \ldots, a_m$ are classical atoms for $m \geq 0$ and $b_1, \ldots, b_n$ are literals for $n \geq 0$. If $m = 1$ and $n = 0$, r is also called a *fact*, and r is a *disjunctive* rule if $m > 1$. Moreover, *choice* rules have the form $\{e_1; \ldots; e_k\} \prec u \leftarrow b_1, \ldots, b_n$, where e_i $(1 \leq i \leq k)$ is a choice element $a : l_1, \ldots, l_m$ in which a is a classical atom and $l_1, \ldots, l_m$ are naf-literals. A *weak constraint* $\coloneq\sim b_1, \ldots, b_n\,[w@l, t_1, \ldots, t_m]$ associates literals $b_1, \ldots, b_n$ with a *weight* w, a *level* l, and additional terms $t_1, \ldots, t_m$ for $m \geq 0$. Finally, $a?$ is a *query* for a classical atom a.

An atom, a rule, or an ASP-Core-2 program is *ground* if it does not contain any variables or arithmetic terms, and an *interpretation* I is a consistent set of ground classical atoms, i.e. $p(t_1, \ldots, t_n)$ and $\neg p(t_1, \ldots, t_n)$ must not jointly occur in I for any predicate name p and terms $t_1, \ldots, t_n$. The satisfaction relation w.r.t. I is defined inductively by

- $I \models a$ for a classical atom a, if $a \in I$, otherwise $I \models \mathbf{not}\ a$;
- $I \models t \prec u$, if $t \prec u$ according to the definition in (Calimeri et al., 2013, Section 2.3);

- $I \models \#\mathbf{agg}\{e_1;\ldots;e_k\} \prec u$ for $\#\mathbf{agg} \in \{\#\texttt{count}, \#\texttt{sum}, \#\texttt{max}, \#\texttt{min}\}$, if $\#\mathbf{agg}(T) \prec u$, where $T = \bigcup_{1\leq i\leq k, e_i=t_1,\ldots,t_n:l_1,\ldots,l_m}\{(t_1,\ldots,t_n) \mid I \models l_1, \ldots, I \models l_m\}$ is the (finite) set of tuples $(t_1,\ldots,t_n)$ for aggregate elements $t_1,\ldots,t_n : l_1,\ldots,l_m$ whose naf-literals $l_1,\ldots,l_m$ are satisfied w.r.t. I, and

  - $\#\texttt{count}(T) = |T|$,
  - $\#\texttt{sum}(T) = \sum_{(t_1,\ldots,t_n)\in T \text{ with integer } t_1} t_1$,
  - $\#\texttt{max}(T) = \max\{t_1 \mid (t_1,\ldots,t_n) \in T\}$, and
  - $\#\texttt{min}(T) = \min\{t_1 \mid (t_1,\ldots,t_n) \in T\}$,[3]

  while $I \models \mathbf{not}\ \#\mathbf{agg}\{e_1;\ldots;e_k\} \prec u$ otherwise;

- $I \models \{e_1;\ldots;e_k\} \prec u$, if $|\bigcup_{1\leq i\leq k, e_i=a:l_1,\ldots,l_m}\{a \in I \mid I \models l_1,\ldots, I \models l_m\}| \prec u$;

- $I \models a_1 \mid \ldots \mid a_m$ for classical atoms $a_1,\ldots,a_m$, if $\{a_1,\ldots,a_m\} \cap I \neq \emptyset$.

A rule of the form $A \leftarrow b_1,\ldots,b_n$ is satisfied w.r.t. I, if $I \models b_1,\ldots,I \models b_n$ implies $I \models A$. Moreover, I is a *model* of a ground ASP-Core-2 program P, if every rule in P is satisfied w.r.t. I. Following (Faber et al., 2004, 2011) in extending the original notion (Gelfond and Lifschitz, 1991) to aggregates, the *reduct* P^I of P w.r.t. I is obtained in two steps:

1. Delete all rules $A \leftarrow b_1,\ldots,b_i,\ldots,b_n$ from P such that $I \not\models b_i$.
2. Replace remaining choice rules $\{e_1;\ldots;a : l_1,\ldots,l_m;\ldots;e_k\} \prec u \leftarrow b_1,\ldots,b_n$ by rules $a \leftarrow b_1,\ldots,b_n,l_1,\ldots,l_m$ for choice elements $a : l_1,\ldots,l_m$ such that $a \in I$ and $I \models l_1,\ldots,I \models l_m$.

Then, a model I of P is an *answer set* of P, if I is a $\subseteq$-minimal model of P^I. That is, all rules of P have to be satisfied w.r.t. I, and the (true) atoms in I must be "derivable" from the applicable rules in P^I. A ground *query* $a?$ holds for P, if a belongs to every answer set of P. Moreover, let P^I_l denote the sum of integers w over all distinct tuples $(w,t_1,\ldots,t_m)$ such that P contains some weak constraint $:\sim b_1,\ldots,b_n\,[w@l,t_1,\ldots,t_m]$ with $I \models b_1,\ldots,I \models b_n$. An answer set I of P is

[3] By convention (Calimeri et al., 2013, Section 2.4), $\#\texttt{max}(\emptyset) < u$ and $\#\texttt{min}(\emptyset) > u$ hold for every term u, so that all aggregates $\#\mathbf{agg}(T) \prec u$ can be evaluated w.r.t. any interpretation I.

optimal, if there is no answer set J of P such that $P_l^J < P_l^I$ for an integer l and $P_{l'}^J = P_{l'}^I$ for all integers $l' > l$.

Depending on the class of programs under consideration, verifying $\subseteq$-minimality w.r.t. P^I can be tractable or computationally complex (Eiter and Gottlob, 1995). The syntactic property of *head-cycle-freeness* (Ben-Eliyahu and Dechter, 1994) allows for distinguishing such cases based on the positive dependency graph of a ground program P, having the atoms in P as nodes and arcs from head atoms $a_1, \ldots, a_m$ of $a_1 \mid \ldots \mid a_m \leftarrow b_1, \ldots, b_n$ or a in $\{e_1; \ldots; a : l_1, \ldots, l_m; \ldots; e_k\} \prec u \leftarrow b_1, \ldots, b_n$ to the classical atoms among $b_1, \ldots, b_n$ as well as $l_1, \ldots, l_m$. Then, P is *head-cycle-free* (HCF), if there is no disjunctive rule $a_1 \mid \ldots \mid a_m \leftarrow b_1, \ldots, b_n$ in P such that two or more of the head atoms $a_1, \ldots, a_m$ share some strongly connected component in the positive dependency graph of P; otherwise, P is *non-HCF*. Furthermore, P is *non-tight*, if some strongly connected component in its positive dependency graph contains an arc, and *tight* otherwise (Fages, 1994; Erdem and Lifschitz, 2003); if P is non-HCF, it is also non-tight, but not necessarily vice versa. The check whether a model is an answer set of P is tractable for HCF programs but coNP-complete for non-HCF programs. Also note that our account of head-cycle-freeness could disregard aggregate atoms, since ASP-Core-2 restrictions on their usage (see (Calimeri et al., 2013)) do not permit circular (positive) dependencies through aggregates.

The semantics of a non-ground program P, possibly including arithmetic terms, is given by the answer sets of its ground instantiation. To this end, variables in a rule or weak constraint r are distinguished into *global* variables, appearing outside of aggregate and choice elements in r, while the remaining variables are *local*. A *ground instance* of r is obtained in two steps:

1. Apply a *global substitution* σ that maps the global variables in r to ground terms yielding a rule $r\sigma$ without global variables.

2. Replace any aggregate or choice element e in $r\sigma$ by the collection of all aggregate or choice elements $e\theta$ obtainable by applying *local substitutions* θ that map the local variables in e to ground terms.

For example, applying the global substitution $\{X \mapsto 1\}$ to the rule in (1) gives $\{hc(1, Y) : arc(1, Y)\} = 1 \leftarrow node(1)$. Mapping the local variable Y to the nodes 1, 2, 3, and 4 then leads to the ground instance $\{hc(1, 1) : arc(1, 1); hc(1, 2) : arc(1, 2); hc(1, 3) : arc(1, 3); hc(1, 4) : arc(1, 4)\} = 1 \leftarrow node(1)$. Note that applicable substitutions σ and θ are required to be *well-formed* (see (Calimeri et al., 2013)), that is, the arithmetic evaluation of arithmetic terms that

do not contain variables must be well-defined. Global substitutions σ violating this condition cannot be used to obtain $r\sigma$ from a rule r, while a respective local substitution θ does not yield an instance $e\theta$ of an aggregate or choice element e.

The *ground instantiation* of a program P, denoted by $grnd(P)$, is the collection of all ground instances of rules or weak constraints in P obtainable by applying well-formed substitutions and evaluating arithmetic terms. Then, the (optimal) answer sets of P are the (optimal) answer sets of $grnd(P)$, possibly subject to a query in P, where the answer to a non-ground query $a?$ consists of all ground instances of atom a that belong to every answer set of P. Taking the prerequisites of instantiation procedures like "intelligent grounding" (Faber et al., 2012) into account, non-ground ASP-Core-2 programs have to comply with additional requirements (Calimeri et al., 2013), such as finiteness of answer sets and safety. In a nutshell, these conditions require the availability of (positive) occurrences of variables within the bodies of rules as well as aggregate or choice elements in order to restrict the relevant substitutions and enable grounders to compute a (finite) ground program, which is typically much smaller yet equivalent to $grnd(P)$.

Reconsidering the TSP instance in Figure 1(b) along with the encoding in (1)–(6), the choice rule in (1) expresses that exactly one outgoing arc must be selected per node for representing a Hamiltonian cycle in terms of (true) atoms over the predicate hc. While the local variable Y occurs in a choice element only, X is global, so that an instance of (1) is obtained for each of the nodes 1, 2, 3, and 4 in the example graph. On the other hand, Y is global in the integrity constraint in (2), i.e. a rule with an empty head that cannot be satisfied. Hence, the $\#\texttt{count}$ atom requires that exactly one incoming arc per node is included in a Hamiltonian cycle. In (3), the lexicographically smallest node name, that is, 1 in our example, is determined by a $\#\texttt{min}$ atom, assigned to X, and the node thus marked as initially reached. The idea of the rule in (4) is to derive further nodes as reached by tracing the arcs in a Hamiltonian cycle. Note that instances of (4) yield circular positive dependencies among ground atoms over the $reach$ predicate, so that ground instantiations are non-tight (and HCF since there are no proper disjunctive rules). The requirement that all nodes must be reached is asserted by the integrity constraint in (5). In view of the $\subseteq$-minimality of an answer set w.r.t. the reduct, models containing isolated subcycles, e.g. between nodes 1 and 4 as well as 2 and 3, are thus discarded, so that answer sets correspond one-to-one to Hamiltonian cycles in a directed graph given as instance. For the example graph in Figure 1(a), there are two answer sets representing the Hamiltonian cycles $(1, 2, 3, 4, 1)$ and $(1, 4, 3, 2, 1)$. Their contained arcs are penalized by their costs (at the common level 1) via instances of the weak constraint in (6). Given

that arc costs for $(1, 2, 3, 4, 1)$ and $(1, 4, 3, 2, 1)$ sum to 8 or 7, respectively, only the answer set representing $(1, 4, 3, 2, 1)$ in terms of the atoms $hc(1, 4)$, $hc(4, 3)$, $hc(3, 2)$, and $hc(2, 1)$ is optimal.

3. Format of the Fifth ASP Competition

The competition builds on the basis of the 2013 edition, offering former and new participants the chance to submit updated and/or novel solvers, and the focus is on the System track, i.e. fixed encodings. As already discussed in Section 1, particular emphasis is placed on ASP-Core-2 language features in order to establish a uniform setting for running and comparing participant systems.

Starting from past experience, one objective was to simplify the scoring system w.r.t. the 2013 edition (see (Alviano et al., 2013a)), and to properly adjust it in the case of Optimization problems, while preserving the general idea of rewarding systems that perform well in a variety of domains and produce solutions of better quality. However, simplified scoring is just a starting point for fostering meaningful solver comparisons. Moreover, tracks are now conceived based on language features, rather than on a complexity basis. The rationale is to encourage the submission of solvers even if they are limited or specialized, respectively, to a language fragment. In addition, this track design paves the way to more detailed analyses of what (combinations of) techniques work well for particular language features. In our opinion, respective insights are, from a scientific point of view, more interesting than merely reporting track winners.

In the remainder of this section, we describe the competition design, introduce categories and tracks, and present the scoring system. Further details about the ASP Competition series and past editions thereof are provided in Section 7.

3.1. Competition Design

In view of the objective of comparing participant systems in a uniform setting, this edition of the ASP Competition did not include a Model&Solve track, but took place in the spirit of the former System track: it was open to any general-purpose solving system, provided it was able to process ASP-Core-2 programs. The general input-output format followed the 2013 edition, so that previous System track submissions should (in principle) be able to participate again. However, the 2013 edition still made exceptions and admitted problem encodings in legacy formats, while the Fifth ASP Competition insisted on ASP-Core-2 compliance. Benchmark domains, encodings, and instances used to assess participant systems were selected by the Organizing Committee; they are detailed in Section 4.

3.2. Competition Categories

The competition consists of *two categories*, depending on the computational resources allotted to each running system:

- **SP**: One processor allowed;
- **MP**: Multiple processors allowed.

While the **SP** category aims at sequential solving systems, parallelism can be exploited in the **MP** category.

3.3. Competition Tracks

Both categories of the competition are structured into *four tracks*, which are described next:

- **Track #1**: *Basic Decision*. Encodings are normal logic programs, with simple arithmetic and comparison operators.
- **Track #2**: *Advanced Decision*. Encodings exploit the full language, with queries, excepting optimization statements and non-HCF disjunction.
- **Track #3**: *Optimization*. Encodings exploit the full language with optimization statements, excepting non-HCF disjunction.
- **Track #4**: *Unrestricted*. Encodings exploit the full language.

3.4. Scoring System

The adopted scoring system balances the following factors:

- Domains are always weighted equally.
- If a system outputs an incorrect answer to some instance in a domain, this invalidates its score for the whole domain, even if all other instances are correctly solved.
- In case of Optimization problems, scoring is based on solution quality.

In general, 100 points can be earned for each domain. The final score of a system consists of the sum of scores over all domains, and ties are broken by cumulative CPU times (timeouts included).

3.5. Scoring Details

For *Decision and Query problems*, the score of a solver S on a domain P featuring N instances is calculated as

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

where N_S is the number of instances solved within the allotted time and memory limits.

For *Optimization problems*, solvers are ranked by solution quality. Let M be the number of participant systems; then, the score of a solver S for an instance I in a domain P featuring N instances is calculated as

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

where $M_S(I)$ is

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

The score $S(P)$ of a solver S for domain P consists of the sum of scores $S(P, I)$ over all N instances I featured by P. Note that, as with Decision and Query problems, $S(P)$ can range from 0 to 100.

3.6. Verification of Answers

Each benchmark domain P is equipped with a checker program C_P that takes as input both an instance I and a corresponding witness solution A, and it is such that $C_P(A, I) =$ *true* in case A is a valid witness for I w.r.t. P.

There are two possible ways to detect incorrect behavior, and subsequently disqualify system S for P:

- S produces an answer A, but A is not a correct solution for I. This case is detected by checking the outcome of $C_P(A, I)$.

- S recognizes instance I as unsatisfiable, but I actually has some witness solution. In this case, it is checked whether another system S' produced a solution A' for which $C_P(A', I)$ is true.[4]

A case of general failure (e.g. "out of memory" errors or other abrupt system failures) does not imply disqualification for a given benchmark domain.

When dealing with Optimization problems, checkers produce also the cost of the (last) witness. For one, this value is considered when calculating scores and assessing answers of systems. For another, given an instance I of an Optimization problem P, the cost of a best valid witness found by any participant is taken as the *imperfect optimum*. In case a system S marks its witness A as optimal for I and the cost of A turns out to be different from the imperfect optimum, this is regarded as incorrect behavior, and S is disqualified for P.

3.7. Solver I/O Interface

Participant systems were required to support the same input-output format as used in 2013. Input programs are compliant with the ASP-Core-2 standard, and the expected system output depends on the kind of a problem, i.e. Decision, Query, or Optimization. Details on the output format can be found in (Krennwallner, 2013).

4. Benchmark Suite

The benchmark domains used in this edition of the ASP Competition largely coincide with the ones from 2013. While respective ASP-Core-2 encodings were already available back then, most participants lacked preparation time and could not submit suitable systems. Hence, half of the systems were in 2013 run on "equivalent" encoding reformulations in legacy formats, and the Fifth ASP Competition is the first edition relying on common inputs in ASP-Core-2. Nowadays, the format is for instance supported by the grounder GRINGO-4 (Gebser et al., 2014b), which thus offers an off-the-shelf front-end for solvers operating at the propositional level.

As described in Section 3, the benchmarks in the Fifth ASP Competition are categorized into tracks based on the language features utilized by encodings. Table 1 provides an overview that groups benchmark domains in terms of language

[4]This is a traditional pragmatic choice made in previous ASP Competitions, where hard unsatisfiable instances have been used as benchmarks even though exhaustive correctness checking cannot be afforded.

features in the ASP-Core-2 encodings from 2013. That is, the 2013 encodings for *Labyrinth* and *Stable Marriage* belong to the Basic Decision track (#1), and the "D" entries in the fourth column indicate that both domains deal with Decision problems. The Advanced Decision track (#2) includes the sixteen 2013 encodings for the domains in rows from *Bottle Filling* to *Weighted-Sequence Problem*. Among them, the *Reachability* domain aims at Query answering, as indicated by the "Q" in the fourth column. The next four rows marked with "O" provide the domains in the Optimization track (#3). Finally, the last four rows give the domains in the Unrestricted track (#4), where *Abstract Dialectical Frameworks* is an Optimization problem and *Strategic Companies* deals with Query answering.

The second column of Table 1 summarizes the usage of particular language features, among the ones introduced in Section 2, for the encodings from the Fourth ASP Competition in 2013.[5] While merely normal rules and comparison operators, considered as `basic` features, are used for *Stable Marriage*, the Basic Decision encoding for *Labyrinth* induces `non-tight` ground instantiations with positive recursion among atoms (Fages, 1994; Erdem and Lifschitz, 2003). Rules including `aggregates` like `#count`, `#sum`, `#max`, and `#min` in their bodies (Faber et al., 2008) are used, e.g. in the Advanced Decision encoding for *Bottle Filling*. Moreover, the Advanced Decision encoding for *Graceful Graphs* includes `choice` rules (Simons et al., 2002), where the superscript "`#`" indicates non-trivial lower and/or upper bounds on the number of chosen atoms. Unlike that, the choice rules for *Complex Optimization* in the Unrestricted track are unbounded, and thus "`#`" is omitted in its row. Proper `disjunctions` in rule heads (Gelfond and Lifschitz, 1991; Eiter and Gottlob, 1995) are utilized, e.g. in the 2013 encoding for *Graph Colouring*. Finally, *Abstract Dialectical Frameworks* is the only Optimization problem in the Fifth ASP Competition for which more than one `level`, i.e. two levels of significance (Simons et al., 2002; Leone et al., 2006), is used in weak constraints.

In order to furnish an extended benchmark collection for this year, we devised new encoding variants for all domains but *Reachability* and *Strategic Companies*, whose 2013 encodings are relatively straightforward positive programs subject to ground queries. In particular, the encoding for *Reachability* is basic, yet categorized into the Advanced Decision track in view of queries. One motivation for providing alternative encodings was to circumvent grounding bottlenecks that

[5] Compared to the Fourth ASP Competition, we decided to drop the *Chemical Classification* domain, whose large encoding (more than 60 MB) imposed primarily a grounding bottleneck. On the other hand, we reintroduced the application-oriented *Partner Units* domain (Aschinger et al., 2011), reusing encodings and instances submitted to the Third ASP Competition in 2011.

Table 1: Benchmark suite of the Fifth ASP Competition. Language features used in the problem encodings from 2013 as well as alternative encodings devised for this year's edition are summarized in the second or third column, respectively, where the abbreviations mean: `b`(asic) features, `n`(on-tight) instances, `a`(ggregate) atoms, `c`(hoice) rules, `d`(isjunctive) rules, `l`(evels) for weak constraints, and n/a if no alternative encoding is provided. Problem encodings in the Unrestricted track **T#4** induce non-HCF programs, while disjunctive rules are not permitted in the Basic Decision track **T#1** or remain HCF in the Advanced Decision track **T#2** as well as the Optimization track **T#3**. The entries "D", "O", and "Q" in the **P** column indicate Decision, Optimization, or Query answering tasks, respectively.

Domain	**2013 Encoding**	**2014 Encoding**	**P**	
Labyrinth	`b`, `n`	`b`, `n`	D	**T#1**
Stable Marriage	`b`	`b`	D	
Bottle Filling	`a`	`a`, `c`	D	**T#2**
Graceful Graphs	`c`$^{\#}$	`c`$^{\#}$	D	
Graph Colouring*	`d`	`b`	D	
Hanoi Tower*	`d`	`b`	D	
Incremental Scheduling	`a`, `c`$^{\#}$	`a`, `c`$^{\#}$	D	
Knight Tour with Holes*	`d`, `n`	`b`, `n`	D	
Nomystery	`a`, `c`$^{\#}$	`c`$^{\#}$	D	
Partner Units	`a`, `d`, `n`	`a`, `c`	D	
Permutation Pattern Matching	`c`$^{\#}$	`c`	D	
Qualitative Spatial Reasoning	`c`$^{\#}$, `d`	`d`	D	
Reachability	`b`, `n`	n/a	Q	
Ricochet Robots	`c`$^{\#}$	`a`, `c`$^{\#}$	D	
Sokoban	`a`, `c`$^{\#}$	`c`$^{\#}$	D	
Solitaire	`c`$^{\#}$	`a`, `c`$^{\#}$	D	
Visit-all*	`a`, `c`$^{\#}$	`b`	D	
Weighted-Sequence Problem	`c`$^{\#}$	`a`, `c`	D	
Connected Still Life	`a`, `c`$^{\#}$, `n`	`a`, `c`, `n`	O	**T#3**
Crossing Minimization	`d`	`a`, `c`	O	
Maximal Clique	`d`	`b`	O	
Valves Location	`a`, `c`$^{\#}$, `n`	`a`, `c`$^{\#}$, `n`	O	
Abstract Dialectical Frameworks	`a`, `d`, `l`, `n`	`a`, `d`, `l`, `n`	O	**T#4**
Complex Optimization	`c`, `d`, `n`	`c`, `d`, `n`	D	
Minimal Diagnosis	`d`, `n`	`d`, `n`	D	
Strategic Companies	`d`, `n`	n/a	Q	

previously hampered meaningful system evaluations for some domains, and thus to extend the collection of benchmarks suitable for solver development. Moreover, evaluating participant systems on previous encodings as well as alternative variants serves to validate competition results and to gain insights regarding the impact of encodings on system performance, where deviations may help to iden-

tify more or less successful modeling approaches.

The language features utilized in the new encoding variants are indicated in the third column of Table 1. Given that the 2014 encodings for the domains marked with "*" omit advanced language features of their 2013 counterparts, in addition to *Labyrinth* and *Stable Marriage*, the Basic Decision track on new encodings comprises four more benchmark domains: *Graph Colouring*, *Hanoi Tower*, *Knight Tour with Holes*, and *Visit-all*. The restriction to basic features is primarily achieved by rewriting disjunctive or choice rules like $p(X) \,|\, q(X) \leftarrow d(X)$ and $\{p(X) : d(X)\} = 1 \leftarrow$ into normal rules such as $p(X) \leftarrow d(X), \mathbf{not}\ q(X)$ and $q(X) \leftarrow d(X), p(Y), X \neq Y$. Such rewriting has also been applied to the 2013 encoding for *Maximal Clique*, which however includes weak constraints and remains in the Optimization track. Further variation is introduced by using the idea of explanatory frame axioms (Reiter, 1991) for the planning problems *Hanoi Tower* and *Visit-all*. For two 2014 encodings with basic features only, those for *Stable Marriage* and *Knight Tour with Holes*, the size of ground instantiations is significantly smaller than before, due to the linearization of encoding parts that previously led to quadratic space consumption.

In more detail, *Knight Tour with Holes* is a specialized version of the Hamiltonian cycle problem, so that rules similar to (3)–(5) can be used for checking reachability and are included in the alternative encoding variant for this domain. The encoding from 2013, however, is based on a quadratic representation of the full transitive closure of reachable nodes in terms of rules like the following:

$$\begin{aligned} reach(X,Y) &\leftarrow hc(X,Y) && (7)\\ reach(X,Z) &\leftarrow hc(X,Y), reach(Y,Z) && (8)\\ &\leftarrow node(X), node(Y), \mathbf{not}\ reach(X,Y) && (9) \end{aligned}$$

Given n nodes, where n amounts to a square number determined by the dimension of a chessboard, a ground instantiation of the subprogram in (7)–(9) is of space complexity $\mathcal{O}(n^2)$, while the rules in (3)–(5) yield linear complexity $\mathcal{O}(n)$.

Moreover, the 2013 encoding for *Stable Marriage* includes a space-demanding integrity constraint of the form

$$\begin{aligned} \leftarrow\ & marry(M,W'), marry(M',W), M \neq M',\\ & m(M,W,P), m(M,W',P'), P > P',\\ & w(W,M,Q), w(W,M',Q'), Q \geq Q' \end{aligned} \qquad (10)$$

for expressing that the marriages of a man M and a woman W to partners W' or M', respectively, are unstable if M has a higher preference for W than for W'

and the preference of W for M is at least as high as for M'. As a consequence, a ground instantiation lists all forbidden pairs of matches between men and women explicitly, yielding space complexity $\mathcal{O}(n^4)$ when n is the number of men as well as women. To avoid an explicit enumeration of such pairs, the 2014 encoding reformulates (10) in terms of a subprogram as follows:

$$\begin{aligned}
keep(m, M, P) &\leftarrow marry(M, W), m(M, W, P) && (11)\\
keep(w, W, Q-1) &\leftarrow marry(M, W), w(W, M, Q), 1 < Q && (12)\\
keep(G, X, R-1) &\leftarrow keep(G, X, R), 1 < R && (13)\\
&\leftarrow m(M, W, P), \mathbf{not}\ keep(m, M, P),\\
&\quad\ w(W, M, Q), \mathbf{not}\ keep(w, W, Q) && (14)
\end{aligned}$$

The idea of the rules in (11)–(13) is to derive preference values such that a man M or a woman W does not want to switch to any respective partner. The integrity constraint in (14) then denies situations in which a man and a woman would like to switch to each other, which in turn means that their marriages are unstable. Provided that preference values lie in the range from 1 to n, as sufficient to distinguish n men as well as women, a ground instantiation of the subprogram in (11)–(14) is of space complexity $\mathcal{O}(n^2)$ and thus much more compact than (10).

Regarding the revised Advanced Decision encodings, significant reductions in grounding size, by about one order of magnitude in comparison to their 2013 counterparts (as indicated in Table 12), were achieved for the domains *Incremental Scheduling*, *Nomystery*, *Partner Units*, and *Weighted-Sequence Problem*. These savings are due to more compact formulations of resource restrictions.

In the *Incremental Scheduling* domain, jobs must be assigned to devices such that their executions do not overlap one another. By representing the starting times of jobs in terms of intervals (Crawford and Baker, 1994), such mutual exclusion can be expressed conveniently and more concisely than with pairwise comparisons of exact starting times. Assuming that instances of $share(J_1, J_2)$ provide jobs J_1 and J_2 to be executed on a common device, the 2013 encoding includes a subprogram as follows for picking starting times and denying overlaps:

$$\begin{aligned}
\{start(J, T) : time(T)\} = 1 &\leftarrow job(J) && (15)\\
&\leftarrow start(J_1, T), len(J_1, L), start(J_2, T_2),\\
&\quad\ share(J_1, J_2), T \leq T_2, T_2 < T + L && (16)
\end{aligned}$$

Similar to (10), the integrity constraint in (16) lists all forbidden starting times T_2 for job J_2 relative to a starting time T and the length L of job J_1 explicitly. In

view of instances with thousand and more time points, such pairwise combinations incur a significant space overhead. Hence, the 2014 encoding abstracts from exact starting times and utilizes a subprogram as follows:

$$\begin{aligned}
\mathit{geq}(J,T) &\leftarrow \mathit{job}(J), \mathit{time}(T), \mathbf{not}\ \mathit{time}(T-1) && (17)\\
\{\mathit{geq}(J,T)\} &\leftarrow \mathit{job}(J), \mathit{time}(T), \mathit{time}(T-1) && (18)\\
&\leftarrow \mathit{geq}(J,T), \mathit{time}(T-2), \mathbf{not}\ \mathit{geq}(J,T-1) && (19)\\
\mathit{start}(J,T) &\leftarrow \mathit{geq}(J,T), \mathbf{not}\ \mathit{geq}(J,T+1) && (20)\\
\mathit{before}(J_1,J_2) &\leftarrow \mathit{job}(J_1), \mathit{geq}(J_2,T), \mathit{len}(J_2,L), \mathbf{not}\ \mathit{geq}(J_1,T+L) && (21)\\
&\leftarrow \mathit{share}(J_1,J_2), \mathit{before}(J_1,J_2), \mathit{before}(J_2,J_1) && (22)
\end{aligned}$$

The rules in (17)–(19) represent an exact starting time T for a job J in terms of atoms $\mathit{geq}(J,t),\ldots,\mathit{geq}(J,T)$ for some smallest time point t. Instead of a choice as in (15), an instance of atom $\mathit{start}(J,T)$ is then derived by the rule in (20). However, the exact starting time of a job J_1 is not needed in (21) to detect that J_1 must be executed before a job J_2, finishing at time $T+L$, in view of the absence of $\mathit{geq}(J_1,T+L)$, in case J_1 and J_2 share a device. That is, the exact starting times of J_1 and J_2 are not directly compared to figure out which order is needed and to deny impossible cases by means of the integrity constraint in (22), so that a ground instantiation saves space in comparison to (16).

While the 2013 encoding for the planning problem *Nomystery*, which had been generated from PDDL (Ghallab et al., 1998; Gerevini et al., 2009), was mostly kept as is, grounding benefits from the addition of built-in atoms to discard rules specifying redundant fluents. In particular, the fuel consumption of a truck driving from a location L_1 to L_2 at time T is expressed by rules like the following:

$$\begin{aligned}
\mathit{fuel}(F-C,T) &\leftarrow \mathit{fuel}(F,T-1), \mathit{drive}(L_1,L_2,T), \mathit{cost}(L_1,L_2,C) && (23)\\
&\leftarrow \mathit{fuel}(F,T-1), \mathit{drive}(L_1,L_2,T), \mathit{cost}(L_1,L_2,C),\\
&\quad F < C && (24)
\end{aligned}$$

The integrity constraint in (24) denies plans in which $F-C$ in $\mathit{fuel}(F-C,T)$ would become negative. Instances of the rule in (23) including such atoms are nevertheless produced by common grounders, as they do not utilize integrity constraints to restrict the required substitutions. As a consequence, atoms representing negative amounts of fuel are first included in a ground instantiation and then falsified by a solver. In order to avoid such redundancy, the 2014 encoding augments the rule in (23) with the precondition $F \geq C$, so that irrelevant instances can be directly discarded during grounding.

The revised encodings for *Partner Units* and *Weighted-Sequence Problem*, both dealing with matching tasks involving budgets, are rewrites. The main principle for reducing the grounding size in comparison to their previously submitted counterparts is the usage of compact aggregates, $\#\texttt{count}$ or $\#\texttt{sum}$, respectively, instead of basic subprograms to encode resource restrictions. In the *Partner Units* domain, a cubic integrity constraint of the form

$$\leftarrow \quad \mathit{assign}(X,V), \mathit{assign}(Y,V), \mathit{assign}(Z,V), X<Y, Y<Z$$

has been reformulated as

$$\leftarrow \quad \mathit{value}(V), \#\texttt{count}\{X : \mathit{assign}(X,V)\} > 2$$

to avoid an explicit enumeration of triples for expressing that at most two elements can be assigned to a common value V. The *Weighted-Sequence Problem*, on the other hand, involves a limitation of accumulated costs for elements denoted by numbers $1,\ldots,n$. In the 2013 encoding, the accumulation and limitation of costs is addressed by a subprogram as follows:

$$\begin{aligned}
\mathit{sum}(1,C) &\leftarrow \mathit{cost}(1,C), \mathit{limit}(M), C \leq M \\
\mathit{sum}(N,C+S) &\leftarrow \mathit{cost}(N,C), \mathit{limit}(M), \mathit{sum}(N-1,S), C+S \leq M \\
\mathit{ok} &\leftarrow \mathit{sum}(n,S) \\
&\leftarrow \mathbf{not}\ \mathit{ok}
\end{aligned}$$

That is, basic rules describe the formation of total costs along n elements, whose individual costs depend on an interpretation at hand, and the limit is enforced by checking the existence of a viable outcome. Given that instances involve cost limitations to several hundreds and plenty (intermediate) sums are constructable, the above subprogram is responsible for large ground instantiations. Such a space blow-up is avoided by using a $\#\texttt{sum}$ aggregate in an integrity constraint of the form

$$\leftarrow \quad \mathit{limit}(M), \#\texttt{sum}\{C,N : \mathit{cost}(N,C)\} > M$$

included in the 2014 encoding for the *Weighted-Sequence Problem* domain.

New variants of encodings for domains in the Optimization track and the Unrestricted track are mainly obtained through local modifications and sometimes simplifications, while maintaining the underlying ideas. The only exception is the matching problem *Crossing Minimization*, dealing with the relative positions of

endpoints connected by edges. Similar to the starting times of jobs in *Incremental Scheduling*, the 2014 encoding represents positions in terms of intervals. That is, for each endpoint of some edge, a predicate geq provides all positions preceding it w.r.t. the placement described by an answer set. This allows for a compact retrieval of the relative positions of endpoints X and Y by means of a rule

$$before(X,Y) \quad \leftarrow \quad point(X), geq(Y,P), \mathbf{not}\ geq(X,P)$$

of the same pattern as (21), whereas the 2013 encoding relies on a rule of the form

$$before(X,Y) \quad \leftarrow \quad assign(X,P), assign(Y,P'), P < P'$$

whose ground instances enumerate pairs of candidate positions explicitly. Beyond reducing the size of ground instantiations (which is uncritical for *Crossing Minimization* instances used in the competition), the reformulation aims at an increased abstraction from exact placements for possibly improving search performance, especially when conflict-driven learning (Marques-Silva and Sakallah, 1999; Zhang et al., 2001) is applied.

Finally, note that the instances run in the Fifth ASP Competition have been randomly selected from the suites submitted in 2013 (or 2011 for *Partner Units*), using the concatenation of winning numbers from the EuroMillions lottery of Tuesday, 22nd April 2014, as random seed. In this way, twenty instances were picked per domain in order to assess the participant systems both on the encodings from 2013 as well as their new variants.

5. Participants

In this section, we introduce the participants of the Fifth ASP Competition: 16 systems submitted by three teams.

The *Aalto team* from Aalto University, Finland, submitted nine solvers, working by means of translations, whose detailed descriptions can be found in (Bomanson and Janhunen, 2013; Gebser et al., 2014a; Liu et al., 2012; Nguyen et al., 2011). Three systems, LP2SAT3+GLUCOSE, LP2SAT3+LINGELING, and LP2SAT3+PLINGELING-MT, rely on translation to SAT, which includes the normalization of aggregates as well as the encoding of level mappings for non-tight ground programs. The latter are expressed in terms of bit-vector logic or acyclicity checking, respectively, supported by the back-end SMT solvers of the LP2BV2+BOOLECTOR and LP2GRAPH systems. While the aforementioned systems do not support optimization and participate in the Basic and Advanced Decision

tracks (#1 and #2) only, LP2MAXSAT+CLASP, LP2MIP2, and LP2MIP2-MT, running CLASP as a Max-SAT solver or the Mixed Integer Programming solver CPLEX as back-ends, respectively, compete in the Optimization track (#3) as well. Finally, LP2NORMAL2+CLASP normalizes aggregates (of up to certain size, following the rationale that introduced structural propositions may be useful in conflict-driven learning) and runs CLASP as back-end ASP solver; LP2NORMAL2+CLASP participates in all four tracks and thus also in the Unrestricted track (#4). All systems by the Aalto team utilize GRINGO-4 for grounding, and none of them supports Query problems (*Reachability* and *Strategic Companies*). The systems LP2SAT3+PLINGELING-MT and LP2MIP2-MT exploit multi-threading and run in the **MP** category, while the other, sequential systems participate in the **SP** category.

The *Potassco team* from the University of Potsdam, Germany, submitted the sequential system CLASP (Gebser et al., 2012a), a native ASP solver for (extended) disjunctive logic programs based on conflict-driven learning, in the **SP** category and its multi-threaded version CLASP-MT (Gebser et al., 2012b) in the **MP** category. Both systems utilize GRINGO-4 for grounding and participate in all four tracks, while they do not support Query problems.

The *Wasp team* from the University of Calabria, Italy, submitted five incarnations of WASP (Alviano et al., 2013b, 2014), a native ASP solver based on conflict-driven learning, yet extended with techniques specifically designed for solving disjunctive logic programs, in the **SP** category. Unlike WASP-1, utilizing a prototype version of DLV (to cope with the ASP-Core-2 language) for grounding, WASP-2 relies on GRINGO-4 and further differs from WASP-1 in the implementation of program simplifications and deterministic inferences. Moreover, WASP-1.5 is a hybrid system combining WASP-1 and WASP-2, basically switching between them depending on whether a logic program is non-HCF or subject to a query. While WASP-1 and WASP-1.5 compete in all domains and tracks, WASP-2 does not participate in the Unrestricted track (#4). Additionally, the WASP-WMSU1-ONLY-WEAK and WASP-WPM1-ONLY-WEAK systems are specifically designed for solving Optimization problems in the Optimization track (#3) plus the *Abstract Dialectical Frameworks* domain in the Unrestricted track only.

Table 2 lists the submitted systems (in rows) along with their tracks and categories (in columns). The Advanced Decision track in column **T#2** (other tracks denoted similarly) is further subdivided to indicate in subcolumn **q** the ability of dealing with Query problems, admitted within this as well as the Unrestricted track **T#4**. The last column reports whether a system falls into the **SP** or **MP** category. After a dry-run period, in which teams could test their submissions on the competition machine, the Wasp team withdrew WASP-WMSU1-ONLY-WEAK (due

Table 2: Overview of submitted systems along with their tracks and categories

System	T#1	T#2		T#3	T#4	SP/MP
			q			
LP2BV2+BOOLECTOR	√	√				**SP**
LP2GRAPH	√	√				**SP**
LP2SAT3+GLUCOSE	√	√				**SP**
LP2SAT3+LINGELING	√	√				**SP**
LP2SAT3+PLINGELING-MT	√	√				**MP**
LP2MAXSAT+CLASP	√	√		√		**SP**
LP2MIP2	√	√		√		**SP**
LP2MIP2-MT	√	√		√		**MP**
LP2NORMAL2+CLASP	√	√		√	√	**SP**
CLASP	√	√		√	√	**SP**
CLASP-MT	√	√		√	√	**MP**
WASP-1	√	√	√	√	√	**SP**
WASP-1.5	√	√	√	√	√	**SP**
WASP-2	√	√		√		**SP**
WASP-WMSU1-ONLY-WEAK				√	(√)	**SP**
WASP-WPM1-ONLY-WEAK				√	(√)	**SP**

to a bug that appeared after the submission deadline but before the competition started), so that 15 systems remained in the competition, 12 in the **SP** and 3 in the **MP** category.

Regarding ASP solving approaches, similar to past competition editions, we can identify two main lines:

- "native" systems, which exploit techniques purposely conceived for dealing with logic programs under the stable models semantics, and
- "translation-based" systems, which at some stage of the evaluation process produce an intermediate specification in some different formalism that is then fed to a corresponding solver.

Native systems include the ones submitted by the Potassco and Wasp teams as well as LP2NORMAL2+CLASP, while the remaining systems by the Aalto team utilize translations.

It is also worth mentioning that, in order to assess improvements in system implementation, we reran a selection of the systems submitted to the Fourth ASP Competition in 2013. Section 6.2 compares their performance to current versions.

6. Competition Results

The competition was run on a Debian Linux server (64bit kernel), equipped with Intel Xeon X5365 processors and 16GB RAM. Time and memory per run were limited to 600 seconds and 6GB, respectively. Participant systems could use up to 8 cores in the **MP** category, while the execution was limited to a single core in the **SP** category. The performance of systems was measured using the *pyrunlim* tool (Alviano, 2014).

We start by presenting performance results for the systems introduced in Section 5 on benchmarks consisting of 2013 problem encodings and twenty randomly selected instances per domain. This also includes the announcement of winners in tracks and categories. Then, Section 6.2 compares current systems to previous versions submitted in 2013 to give an account of the progress. The impact of modeling on the ASP solving process is assessed in Section 6.3 by comparing the performance on alternative encoding variants extending the benchmark collection.

6.1. Results of the Fifth ASP Competition

In the following, we report the official results of the Fifth ASP Competition track by track, and finally announce the category winners. We first present results for the **SP** category, and then we turn to the **MP** category.

Tables 3–6 show the results for each track of the **SP** category. The tables are organized as follows: the first column provides systems, the second their scores, and the third cumulative CPU times. Systems are listed by scores in decreasing order with ties broken by CPU times, so that table positions correspond to the ranks of systems. For readability and since the ranks of systems are preserved, the reported CPU times account only for runs rewarded with positive scores, thus excluding 600 seconds penalty per failed solving attempt.

The results for Track #1 are given in Table 3. This track includes only two domains, namely *Labyrinth* and *Stable Marriage*. The system CLASP performs best by solving all but three of the forty instances, followed by LP2NORMAL2+CLASP and WASP-1.5. Another incarnation of WASP, i.e. WASP-2, as well as LP2GRAPH have the same score as WASP-1.5, but with higher cumulative CPU times.

Table 4 shows the results for Track #2, consisting of sixteen domains (cf. Table 1). Here LP2NORMAL2+CLASP is the best performing system, solving two instances more than CLASP, which ranks second, and fifteen instances more than LP2MAXSAT+CLASP, ranking third (closely followed by LP2SAT3+LINGELING). Notably, translation-based systems turn out to be competitive in this track, coming quite close to native solvers.

Table 3: Overall results for Track #1 (Basic Decision)

System	Score	CPU Time
CLASP	185	5532.6
LP2NORMAL2+CLASP	165	4742.8
WASP-1.5	160	3918.7
WASP-2	160	3949.0
LP2GRAPH	160	6324.8
LP2MAXSAT+CLASP	140	5871.5
LP2SAT3+GLUCOSE	130	4809.0
WASP-1	110	3755.9
LP2SAT3+LINGELING	95	3661.4
LP2BV2+BOOLECTOR	15	1510.4
LP2MIP2	0	–

Table 4: Overall results for Track #2 (Advanced Decision)

System	Score	CPU Time
LP2NORMAL2+CLASP	870	13749.4
CLASP	860	14904.0
LP2MAXSAT+CLASP	795	18186.7
LP2SAT3+LINGELING	790	20001.9
WASP-1.5	740	14622.4
LP2GRAPH	735	9593.4
LP2SAT3+GLUCOSE	735	10277.6
LP2BV2+BOOLECTOR	670	15167.8
WASP-2	660	11940.5
WASP-1	605	17511.2
LP2MIP2	140	5662.8

The results for Track #3, including four Optimization problems, are presented in Table 5, where system scores, determined by relative solution quality (cf. Section 3.5), are for readability rounded to nearest integers; this also applies to the scores in Tables 6–8, 11, and 12. The best performing systems are CLASP, WASP-1, and WASP-2 as well as WASP-1.5 with identical scores and CPU times, given that WASP-1.5 always resorts to WASP-2 here. Somewhat unfortunately, the wrapper scripts of LP2NORMAL2+CLASP, LP2MAXSAT+CLASP, and LP2MIP2 failed to return their current best witnesses in case of timeouts, so that the scores draw an incom-

Table 5: Overall results for Track #3 (Optimization)

System	Score	CPU Time
CLASP	322	21018.7
WASP-1	224	32453.5
WASP-2	186	28802.3
WASP-1.5	186	28802.3
LP2NORMAL2+CLASP	125	4667.0
LP2MAXSAT+CLASP	115	2529.6
LP2MIP2	110	523.8
WASP-WPM1-ONLY-WEAK	46	12000.0

Table 6: Overall results for Track #4 (Unrestricted)

System	Score	CPU Time
CLASP	285	2307.7
LP2NORMAL2+CLASP	280	3478.5
WASP-1	101	7492.4
WASP-1.5	101	7541.8
WASP-WPM1-ONLY-WEAK	25	451.4

plete picture of the potential optimization capabilities of these systems.

Finally, Table 6 provides the results for Track #4, containing non-HCF instances of the Decision problems *Complex Optimization* and *Minimal Diagnosis*, the Query problem *Strategic Companies*, and the Optimization problem *Abstract Dialectical Frameworks*. Note that WASP-WPM1-ONLY-WEAK competed only in the latter domain, scored like the Optimization problems in Track #3, and that no instance of *Strategic Companies* was solved by any system. In the three effectively remaining domains, CLASP performs best, closely followed by LP2NORMAL2+CLASP, which has some edge over WASP-1. Since WASP-1.5 always resorts to WASP-1 for non-HCF programs, it comes just a short time margin behind.

We are now ready to announce the winners in the **SP** category, according to the accumulated results given in Table 7. Thus, the first place goes to CLASP by the Potassco team, the second place to LP2NORMAL2+CLASP by the Aalto team, and the third place to WASP-1.5 by the Wasp team. That is, native systems are ahead of translation-based ones in the category ranking, which is partly explained by their versatility: while the winner systems competed in all four tracks, the translation-based systems by the Aalto team could only participate in Tracks #1

Table 7: Overall results for the **SP** category

System	Score	CPU Time
CLASP	1652	43763.0
LP2NORMAL2+CLASP	1440	26637.7
WASP-1.5	1187	54885.2
LP2MAXSAT+CLASP	1050	26587.8
WASP-1	1040	61213.0
WASP-2	1006	44691.8
LP2GRAPH	895	15918.2
LP2SAT3+LINGELING	885	23663.2
LP2SAT3+GLUCOSE	865	15086.6
LP2BV2+BOOLECTOR	685	16678.2
LP2MIP2	250	6186.6
WASP-WPM1-ONLY-WEAK	71	12451.4

Table 8: Overall results for the **MP** category

System	Score	CPU time
CLASP-MT	1770	45440.9
LP2SAT3+PLINGELING-MT	995	18563.0
LP2MIP2-MT	315	8757.1

and #2 plus, in case of LP2MAXSAT+CLASP and LP2MIP2, the Optimization problems in Track #3.

Similarly, Table 8 summarizes performance results for the **MP** category, in which CLASP-MT by the Potassco team is followed by LP2SAT3+PLINGELING-MT and LP2MIP2-MT by the Aalto team. The native system CLASP-MT participated and achieved first places in all four tracks, LP2SAT3+PLINGELING-MT was more successful than LP2MIP2-MT among translation-based systems in Tracks #1 and #2, while LP2MIP2-MT became second (behind CLASP-MT) in Track #3 dealing with Optimization problems; detailed results for each domain are given in Table 16. Albeit CLASP-MT was disqualified in *Abstract Dialectical Frameworks* due to incorrect parallel optimization w.r.t. non-HCF programs, all three multi-threaded systems have a clear edge over their respective sequential counterparts and could thus take advantage of parallelism to solve more instances.

Having announced the winners, in the following we give more detailed views of the performance of participant systems, focusing again on the **SP** category. To

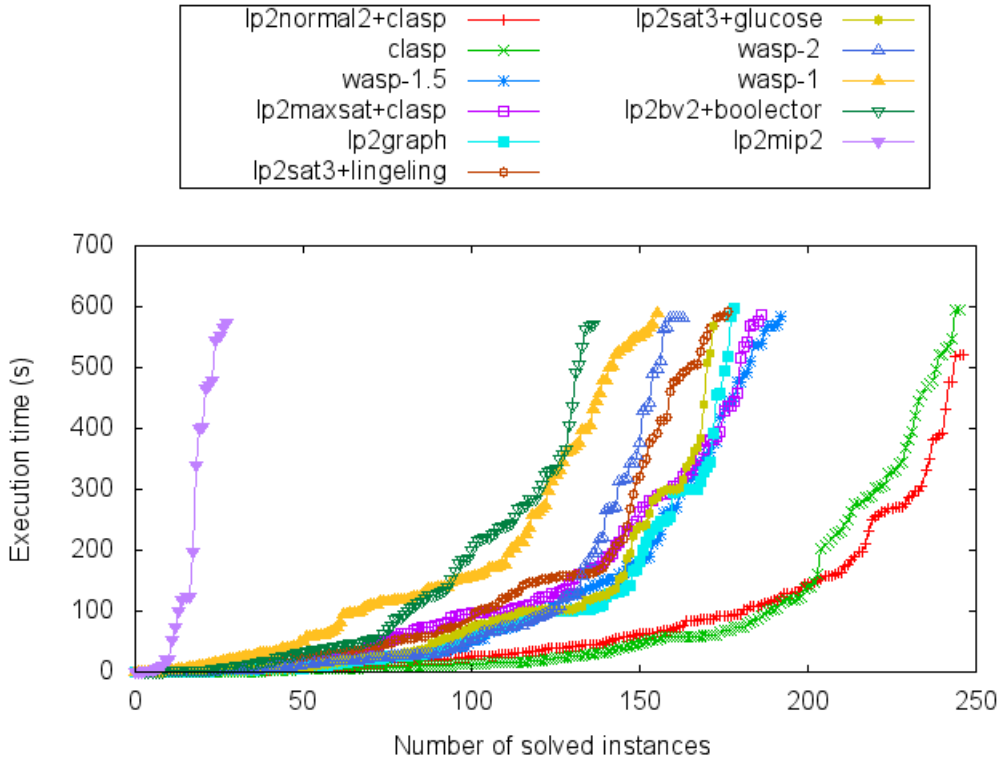


Figure 2: Cactus plot for Decision problems

begin with, Figure 2 displays a cactus plot for Decision problems (marked with "D" in Table 1). For each system, the x-axis gives the number of instances solved within a respective execution time on the y-axis. The performance curves exhibit three patterns: CLASP and LP2NORMAL2+CLASP, based on the same solver, are close to each other on the right hand side, and LP2NORMAL2+CLASP turns out to be quite effective in still solving time-consuming instances; the majority of systems gathers in the center block led by WASP-1.5, and span over a range of about 60 instances with gradual performance differences; LP2MIP2 remains on the left hand side of the plot, taking into account that its back-end solver CPLEX is less geared towards purely propositional settings than other solvers.

The curves in Figure 3 plot the score acquisition of systems for Optimization problems (marked with "O" in Table 1), where system runs are ordered by the achieved positive scores, and the sum of scores for the number of instances on the x-axis is given on the y-axis. In the first place, the plot indicates that CLASP had more success in producing high-quality witnesses than the four incarnations

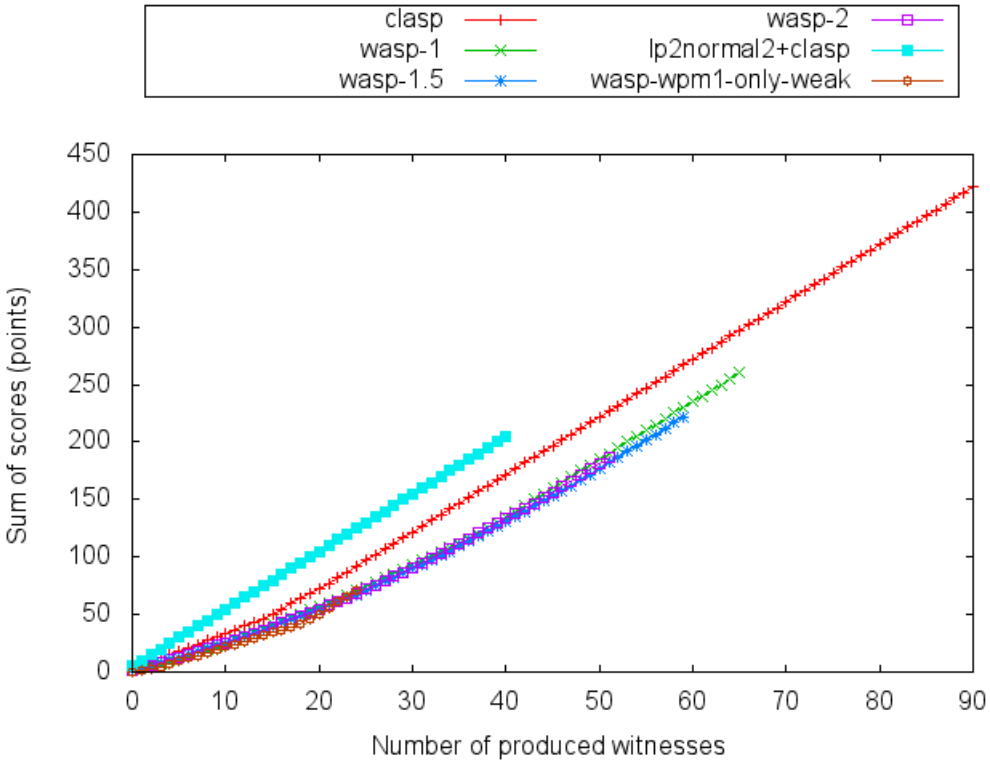


Figure 3: Score acquisition plot for Optimization problems

of WASP whose curves are underneath. As mentioned above, due to malfunctioning wrapper scripts, the systems LP2NORMAL2+CLASP, LP2MAXSAT+CLASP, and LP2MIP2 failed to return their current best witnesses in case of timeouts, but produced confirmed optimum solutions only. Since the resulting curves are indistinguishable, LP2MAXSAT+CLASP and LP2MIP2, which solved fewer instances and are thus dominated by LP2NORMAL2+CLASP, are not displayed in Figure 3. While LP2NORMAL2+CLASP returned optimal witnesses only, the other five systems gained (positive) scores in considerably more runs, and CLASP earns roughly double the sum of scores using the same search back-end as LP2NORMAL2+CLASP.

Table 9 gives an impression of the memory consumption of participant systems by providing the minimum, 25th percentile, median, 75th percentile, and maximum amounts of memory used in runs rewarded with positive scores along with respective average values and standard deviations. Apart from WASP-WPM1-ONLY-WEAK, which dealt with Optimization problems in five domains only, the median and average values indicate that CLASP and LP2GRAPH, whose support for

Table 9: Memory consumption (MB)

System	min	25%	50%	75%	max	avg	stdev
CLASP	<1.0	19.1	38.1	203.5	4039.6	257.2	539.4
LP2BV2+BOOLECTOR	12.9	90.7	157.6	559.9	2701.3	484.9	656.6
LP2GRAPH	5.5	33.5	82.2	202.6	502.9	131.5	122.6
LP2MAXSAT+CLASP	<1.0	30.0	59.9	367.1	5998.2	584.8	1282.7
LP2MIP2	15.5	29.7	117.3	397.1	3823.2	329.3	579.4
LP2NORMAL2+CLASP	<1.0	28.9	77.0	326.1	5770.5	410.6	915.2
LP2SAT3+GLUCOSE	6.2	33.7	92.1	391.1	6009.0	586.3	1248.6
LP2SAT3+LINGELING	4.4	22.1	59.1	306.6	5728.1	359.2	711.0
WASP-1	<1.0	35.6	146.9	619.6	5625.1	835.2	1407.9
WASP-1.5	<1.0	84.4	364.3	1235.3	5538.7	1027.4	1426.6
WASP-2	<1.0	90.9	336.7	1205.2	5185.6	840.5	1202.0
WASP-WPM1-ONLY-WEAK	<1.0	15.9	28.9	72.8	474.9	62.9	90.2
CLASP-MT	<1.0	57.9	163.8	616.0	5156.2	509.8	825.9
LP2MIP2-MT	16.4	46.2	71.7	216.1	1153.3	159.7	210.9
LP2SAT3+PLINGELING-MT	<1.0	23.4	95.1	541.8	5979.2	604.8	1137.6

acyclicity checking admits a compact representation of non-tight programs, consume the least memory among systems in the **SP** category. The memory overhead incurred by the normalization of aggregates and/or translation to the formalisms of other back-end solvers can be observed in the second and fourth to eighth row. The three incarnations of WASP below consume relatively much memory on some specific instances (as witnessed by the high standard deviations), while still being comparable with other systems in terms of maximum allocation. Also note that the selection process implemented in WASP-1.5 requires more memory than WASP-1 and WASP-2, which is because the input instance is provided as unique stream that is buffered to allow for an efficient solver selection. Regarding the three systems in the **MP** category, CLASP-MT and LP2SAT3+PLINGELING-MT, which are based on conflict-driven learning, demand significantly more memory than their sequential counterparts. Unlike that, the multi-threaded version LP2MIP2-MT of LP2MIP2 scores on more instances, while consuming less memory on average.

Finally, Tables 14–16 at the end of this paper report detailed results per solver and domain. Notably, the higher score of LP2NORMAL2+CLASP than CLASP in Track #2 is due to the domains *Hanoi Tower* and *Visit-all*. Since their encodings do not involve substantial aggregates, we attribute these performance differences to the usage of different search parameters rather than normalization.

6.2. Comparison to Previous Competition Edition

In the following, we compare the performance of the most successful 2013 submission by each team, rerun under the same conditions as this year's entries in order to obtain comparable results, to "updated" versions in the Fifth ASP Competition. We begin with WASP-1 by the Wasp team, which resembles a corresponding system submitted in 2013, and then proceed to systems by the Aalto and Potassco teams.

Wasp team. The system WASP-1 is a bug-fix version of the 2013 submission DLV+WASP (Alviano et al., 2013b). Hence, we concentrate on the comparison between WASP-1, its successor WASP-2, and the hybrid system WASP-1.5. Regarding the Decision problems in Tracks #1 and #2, WASP-2 solves 10 or 27, respectively, more instances than WASP-1, and 16 instances of the Query problem *Reachability* are solved likewise by WASP-1 and WASP-1.5. The only domain in which WASP-1 solves (three) more instances of a Decision problem than WASP-1.5 and WASP-2 is *Qualitative Spatial Reasoning*. This outlier can be explained by the naive handling of disjunction implemented in WASP-2. Indeed, the encoding for *Qualitative Spatial Reasoning* involves long disjunctions, so that the straightforward application of shifting (Ben-Eliyahu and Dechter, 1994) causes a space blow-up resulting in five memory outs. Unlike that, WASP-1 handles disjunctions by means of dedicated data structures. In Track #3 dealing with Optimization problems, WASP-1 still has advantages over the more recent WASP-2 system. However, when disregarding Track #4 in which WASP-2 did not participate, it remains ahead of WASP-1 by a score difference of 67. This gap increases to 147 points when considering WASP-1.5, which resorts to WASP-1 for Query answering in the *Reachability* domain and to deal with non-HCF programs in Track #4. Notably, the overhead of switching between WASP-1 and WASP-2 stays negligible, as WASP-1.5 achieves the same score as the system it picks for each domain. Hence, by utilizing WASP-2 in domains where it is applicable, WASP-1.5 significantly improves on WASP-1.

Aalto team. The system LP2SAT (Janhunen and Niemelä, 2011), relying on eager translation of logic programs to SAT and using the PRECOSAT solver as search back-end, has been the best performing 2013 submission by the Aalto team. Among this year's entries, LP2SAT3+GLUCOSE and LP2SAT3+LINGELING are based on the same approach. In order to use common inputs, we compare the previous and this year's systems on Basic Decision encodings only, taking into account that LP2SAT processes a legacy format that differs from ASP-Core-2 on advanced constructs such as aggregates. Given that the Basic Decision track on 2013 encodings

Table 10: Comparison between the 2013 submission LP2SAT, LP2SAT3+GLUCOSE, and LP2SAT3+LINGELING

	LP2SAT		LP2SAT3+GLUCOSE		LP2SAT3+LINGELING	
Domain	**Score**	**CPU Time**	**Score**	**CPU Time**	**Score**	**CPU Time**
Labyrinth	0	–	35	1668.9	30	1600.7
Stable Marriage	5	170.9	95	3140.1	65	2060.7
Graph Colouring	60	4334.2	80	2597.5	90	2306.2
Hanoi Tower	95	96.8	100	47.9	100	105.7
Knight Tour with Holes	0	–	0	–	0	–
Labyrinth	0	–	40	1586.8	25	1783.0
Stable Marriage	95	508.3	100	442.9	100	452.9
Visit-all	95	3165.7	60	1296.6	65	570.2

merely includes two domains, we here also consider the six alternative encodings with basic features only, and Table 10 shows respective scores and cumulative CPU times for solved instances. Regardless of which encoding is used, LP2SAT, LP2SAT3+GLUCOSE, and LP2SAT3+LINGELING have difficulties with instances of *Labyrinth* and *Knight Tour with Holes*, since translation to SAT is particularly intricate for the non-tight programs in these domains. However, LP2SAT3+GLUCOSE and LP2SAT3+LINGELING improve on LP2SAT by solving from 5 to 8 instances of *Labyrinth*, depending on which encoding is used, whereas LP2SAT times out on all non-tight instances. Another gap is observed on the 2013 encoding for *Stable Marriage*, while the differences on 2014 encodings are smaller. Nevertheless, LP2SAT3+GLUCOSE and LP2SAT3+LINGELING have some advantages in *Graph Colouring*, but LP2SAT solves more instances of *Visit-all*. In general, the performance differences between LP2SAT, on the one hand, and LP2SAT3+GLUCOSE as well as LP2SAT3+LINGELING, on the other hand, are owed to re-engineered translation tools and the use of other SAT solvers as search back-ends. Considering the overall picture, the solving approach based on translation to SAT has been clearly advanced since the Fourth ASP Competition in 2013.

Potassco team. The prototype system CLASPD2 (Gebser et al., 2013) has been the overall winner in the Fourth ASP Competition and evolved into CLASP as submitted this year. Table 11 contrasts both systems on 2013 encodings for all but the domains aiming at Query answering (*Reachability* and *Strategic Companies*), where Decision problems are addressed in the upper part and Optimization problems in the lower part. With either kind of problem, the relative performance of CLASPD2 and CLASP is quite mixed. That is, CLASPD2 solves two or three, respectively, more instances than CLASP in *Hanoi Tower*, *Nomystery*, and *Sokoban*, and

Table 11: Comparison between the 2013 submission CLASPD2 and CLASP

	CLASPD2		CLASP	
Domain	**Score**	**CPU Time**	**Score**	**CPU Time**
Bottle Filling	100	124.3	100	170.4
Complex Optimization	80	527.4	85	553.6
Graceful Graphs	25	756.8	40	756.4
Graph Colouring	40	769.9	45	834.1
Hanoi Tower	100	1527.4	85	1498.3
Incremental Scheduling	0	–	0	–
Knight Tour with Holes	0	–	0	–
Labyrinth	80	1672.6	90	3396.0
Minimal Diagnosis	100	286.3	100	199.1
Nomystery	50	985.2	40	621.8
Partner Units	25	506.1	20	290.6
Permutation Pattern Matching	70	1247.8	70	1556.6
Qualitative Spatial Reasoning	100	1276.8	100	2124.4
Ricochet Robots	100	933.6	100	2464.4
Sokoban	45	855.9	35	1076.9
Solitaire	75	669.2	85	112.6
Stable Marriage	95	1828.8	95	2136.6
Visit-all	60	570.6	55	1846.2
Weighted-Sequence Problem	75	938.1	85	1551.3
Abstract Dialectical Frameworks	100	1115.3	100	1555.0
Connected Still Life	82	10200.7	72	9613.3
Crossing Minimization	75	9028.7	96	4466.4
Maximal Clique	60	10708.3	100	938.3
Valves Location	97	10481.3	54	6000.7

it also achieves higher scores for the Optimization problems *Connected Still Life* and *Valves Location*. Opposite behavior is in turn observed in *Graceful Graphs*, *Labyrinth*, *Solitaire*, and *Weighted-Sequence Problem* as well as *Crossing Minimization* and *Maximal Clique*. In total, the accumulated results amount to two more solved instances of Decision problems and a score difference of 8 for Optimization problems, both in favor of CLASP. Albeit CLASP is equipped with a refined implementation and fine-tuned search parameters, the 2013 submission CLASPD2 thus remains close to its successor on the competition benchmarks.

6.3. Impact of Modeling

We further investigate revised encodings furnishing an alternative benchmark collection. To this end, Table 12 provides grounding parameters, obtained by running GRINGO-4 under the same time and memory limits as participant systems,

Table 12: Comparison between 2013 and alternative encodings. For each domain where an alternative encoding is provided, the upper row gives results for the respective 2013 encoding, and the lower row refers to its novel variant. The columns before system scores report the number of ground instantiations obtainable with GRINGO-4 within time and memory limits along with the average CPU time, memory consumption, and output size of GRINGO-4.

Domain	**Ground Instantiations**	**Grounding Time (s)**	**Grounding Memory (MB)**	**Grounding Size (MB)**	CLASP	LP2BV2+BOOLECTOR	LP2GRAPH	LP2MAXSAT+CLASP	LP2MIP2	LP2NORMAL2+CLASP	LP2SAT3+GLUCOSE	LP2SAT3+LINGELING	WASP-1	WASP-1.5	WASP-2
Abstract Dialectical Frameworks	20	2.3	10.3	289553.0	100	–	–	–	–	80	–	–	36	36	–
	20	2.3	10.1	281718.0	94	–	–	–	–	75	–	–	36	36	–
Bottle Filling	20	3.7	136.8	20295921.8	100	55	5	50	0	100	5	55	100	100	100
	20	3.2	175.4	16622528.8	100	55	5	55	0	100	5	30	95	100	100
Complex Optimization	20	3.9	132.7	11558131.2	85	–	–	–	–	100	–	–	0	0	–
	20	3.9	132.8	11531030.5	80	–	–	–	–	100	–	–	0	0	–
Connected Still Life	20	0.0	0.0	384243.7	72	–	–	15	10	20	–	–	96	86	86
	20	0.0	0.0	356416.0	92	–	–	15	5	20	–	–	19	53	53
Crossing Minimization	20	0.0	0.0	312055.4	96	–	–	30	0	70	–	–	59	41	41
	20	0.0	0.0	67974.5	99	–	–	35	85	55	–	–	60	57	57
Graceful Graphs	20	0.1	2.9	2354707.8	40	35	40	50	0	40	40	45	5	20	20
	20	0.0	1.7	829697.1	35	15	25	25	0	35	25	25	0	20	20
Graph Colouring	20	0.0	0.0	202227.8	45	25	45	50	30	45	45	50	25	35	35
	20	0.0	0.0	182446.8	85	65	80	85	15	85	80	90	25	60	60
Hanoi Tower	20	0.1	6.3	2627765.1	85	85	100	100	0	100	100	85	40	70	70
	20	0.0	0.8	1249704.0	100	100	100	100	0	100	100	100	75	100	100
Incremental Scheduling	9	141.3	4189.7	1495237500.0	0	0	0	0	0	0	0	0	0	0	0
	15	11.6	823.8	118731077.4	65	20	25	25	20	65	25	25	35	65	65
Knight Tour with Holes	15	229.8	2349.4	8014717407.0	0	0	0	0	0	0	0	0	0	0	0
	20	6.1	24.3	67876423.6	5	0	5	0	90	10	0	0	0	5	5
Labyrinth	20	0.2	11.0	3771757.2	90	15	65	40	0	70	35	30	45	60	60
	20	0.2	11.0	3804473.9	90	15	70	25	0	75	40	25	40	60	60
Maximal Clique	20	0.4	13.3	2582856.1	100	–	–	65	100	15	–	–	69	55	55
	20	0.4	10.5	1128405.5	100	–	–	65	100	20	–	–	69	54	54
Minimal Diagnosis	20	3.0	105.6	21853053.6	100	–	–	–	–	100	–	–	65	65	–
	20	1.6	41.7	13523566.9	100	–	–	–	–	100	–	–	100	100	–
Nomystery	20	17.1	77.6	287019297.5	40	45	45	45	5	40	45	40	25	35	35
	20	0.8	19.6	17639442.9	35	45	40	50	15	45	40	50	35	40	40
Partner Units	20	63.4	7.8	1359146887.0	20	0	15	20	0	20	15	15	5	15	15
	20	0.2	9.6	182766999.0	65	20	20	40	0	45	20	30	15	60	60
Permutation Pattern Matching	19	78.9	10.0	3224788374.0	70	40	65	65	35	65	65	65	55	65	65
	20	71.6	11.3	1839055915.0	75	15	75	75	40	75	70	75	70	75	75
Qualitative Spatial Reasoning	20	2.9	135.3	44982026.1	100	0	50	50	10	100	50	50	90	75	75
	20	2.9	134.1	48253514.6	100	0	50	50	20	100	50	50	95	70	70
Ricochet Robots	20	0.0	0.3	12098392.7	100	100	100	100	0	100	100	100	30	35	35
	20	0.0	0.7	845186.5	100	100	100	95	0	100	100	100	0	35	35
Sokoban	20	3.9	94.8	33764103.7	35	35	30	35	0	30	30	25	5	30	30
	20	3.9	94.8	45082889.3	35	35	30	35	0	30	30	25	5	30	30
Solitaire	20	0.0	0.0	2265558.1	85	80	80	80	25	80	80	85	75	85	85
	20	0.0	0.0	286293.1	95	95	95	95	25	100	95	95	95	95	95
Stable Marriage	20	15.7	8.9	293695544.8	95	0	95	100	0	95	95	65	65	100	100
	20	1.1	8.2	41459902.7	100	100	100	100	100	100	100	100	100	100	100
Valves Location	20	8.5	531.5	66926254.4	54	–	–	5	0	20	–	–	–	5	5
	20	8.1	433.0	65056582.2	97	–	–	5	5	24	–	–	0	10	10
Visit-all	20	0.1	4.6	1126664.0	55	100	65	60	35	65	65	90	25	30	30
	20	0.4	3.7	8585873.7	70	85	60	60	85	65	60	65	30	60	60
Weighted-Sequence Problem	20	0.6	5.3	10495461.3	85	70	95	90	0	85	95	85	45	65	65
	20	0.0	0.0	760871.8	100	100	100	100	75	100	100	100	70	100	100

along with the scores of systems in the **SP** category (except for WASP-WPM1-ONLY-WEAK, which did not participate in the majority of domains) on previous and novel encoding variants, given in the upper or lower row, respectively, per domain.

As described in Section 4, one objective of revising previous encodings was to reduce grounding bottlenecks that affected all participant systems and hampered a meaningful comparison. Crucial size reductions were achieved with the new encodings for *Incremental Scheduling*, *Knight Tour with Holes*, *Nomystery*, *Partner Units*, *Stable Marriage*, and *Weighted-Sequence Problem*. In particular, no system solved any instance of *Incremental Scheduling* or *Knight Tour with Holes* on their 2013 encodings, rendering system comparisons void in these cases. Unlike that, some systems earn scores on the corresponding revised encodings that aim at linear rather than quadratic space consumption w.r.t. instance data. Interestingly, LP2MIP2 could benefit most from the revised encoding for *Knight Tour with Holes* and solved 18 out of 20 highly combinatorial instances, even though its translation to MIP is less effective for other domains; e.g. LP2MIP2 solved no instance of *Hanoi Tower* or *Ricochet Robots* regardless of the encoding used. With the exception of *Nomystery*, where system scores remain roughly at the same level with the previous and the new encoding variant, savings in grounding generally paid off.

The second aspect of investigating alternatives was to gain some impression of the impact of encodings on search performance. In this regard, the revised encodings turn out to boost several systems in the domains *Crossing Minimization* (cf. timeouts in the detailed results in Tables 14–19), *Graph Colouring*, *Hanoi Tower*, *Minimal Diagnosis*, and *Solitaire*. However, the opposite effect also applies in some cases, as observed on the scores of WASP variants in the *Connected Still Life* domain and declines for all systems in *Graceful Graphs*. Furthermore, different systems may react non-uniformly like in *Visit-all*, where LP2MIP2 and the WASP variants perform better on the revised encoding, while LP2BV2+BOOLECTOR and LP2SAT3+LINGELING behave the other way round.

Apart from objective measures such as grounding size, the effect of encoding variants on system performance is difficult to predict, and different options shed some light on whether benchmarks are inherently hard or influenced by modeling. Arguably, more challenging instances would be desirable for domains in which some encoding variant enables a participant system to complete all its runs in time, as it happens in half of the domains listed in Table 12.

The impact of modeling on performance is summarized in Figure 4, which contrasts overall scores, displayed in decreasing order, of systems in the **SP** category on alternative encoding variants with corresponding scores on 2013 encodings (as listed in Table 7). Apart from the prototype system WASP-WPM1-ONLY-

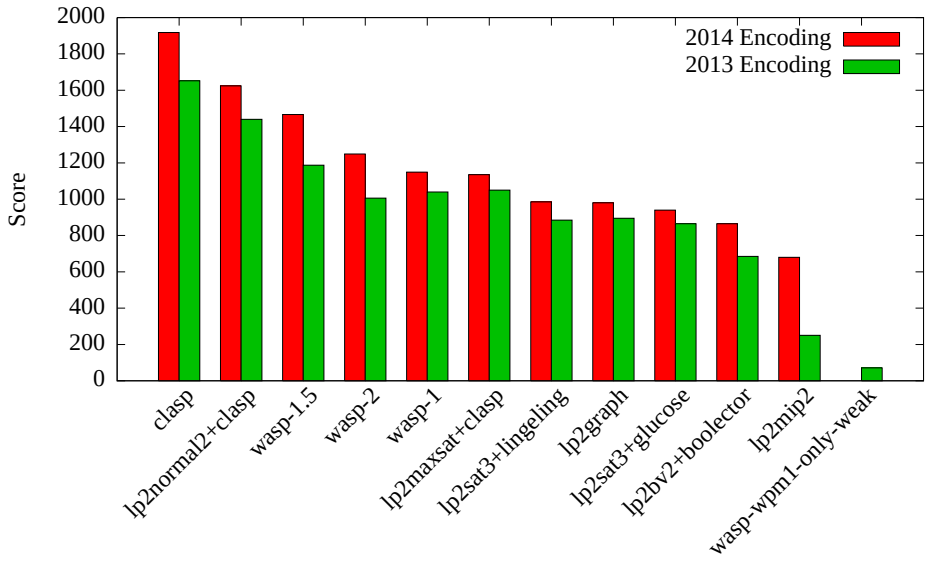


Figure 4: Comparison of system scores achieved on new encoding variants

WEAK, which failed on the new encodings for Optimization problems, all systems benefit more or less substantially from rewrites. However, the first three places remain the same, and gradual differences to the official ranking in the **SP** category concern systems whose performance is close on both previous and novel encoding variants. That is, relative performance is largely unaffected by modeling aspects, but primarily owed to solving technology and versatility, while encoding improvements can be highly effective in general.

More detailed results of running systems in the **SP** category on alternative encodings can be found in Tables 17–19 at the end of this paper.

7. Comparison to Previous ASP and Other Competitions

In this section, we compare the format of the Fifth ASP Competition with past editions, discussing the differences with respect to several aspects, such as tracks or scoring systems. Furthermore, it is worth noting that ASP has a close relationship to other modeling paradigms and languages, such as SAT, SMT, QBF, and more, all of which aim at solving demanding AI problems. In the following, we locate this edition of the ASP Competition in the spectrum of related events.

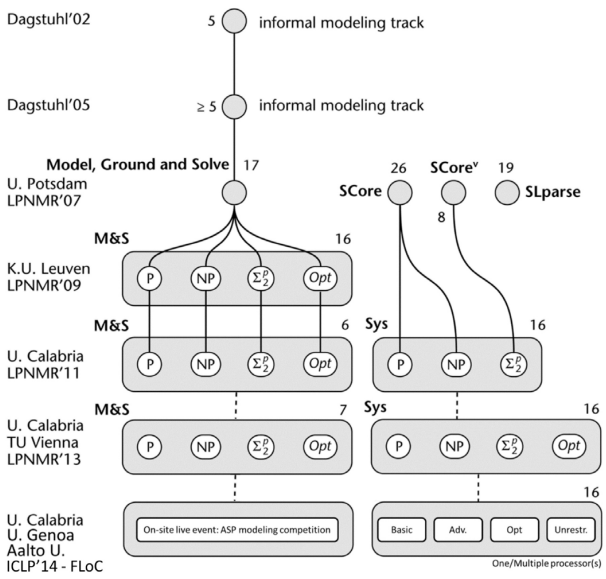


Figure 5: Evolution of the ASP Competition series (numbers denote participant counts in corresponding tracks)

7.1. The ASP Competition Series

In September 2002, participants to the Dagstuhl Seminar on Nonmonotonic Reasoning, Answer Set Programming and Constraints (Brewka et al., 2002) agreed that standardization was a key issue for the development of ASP; hence, they decided to establish an infrastructure for benchmarking ASP solvers, as already in use in the neighboring fields of SAT and Constraint Programming. A first informal competition took place during the workshop, featuring the five systems DLV (Leone et al., 2006), SMODELS (Simons et al., 2002), ASSAT (Lin and Zhao, 2004), CMODELS (Giunchiglia et al., 2006), and ASPPS (East and Truszczyński, 2001) from TU Vienna/Univ. of Calabria, Helsinki UT, Hong Kong UST, Univ. of Texas at Austin, and Univ. of Kentucky. Since then, and after a second informal edition in 2005, the ASP Competition series has been established as a reference event for the community. It takes usually place biennially, and results are officially announced at the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR).

Interestingly, the competition format was not definitely established from the beginning; rather, as research went on and the state of the art was pushed forward, it has been adjusted in order to foster advancements and grant benefits to the

community. However, it is worth noting that language standardization has always been a crucial issue. A progress schema of the ASP Competition series is shown in Figure 5. For further details, we refer the reader to (Calimeri et al., 2012).

In the 2011 and 2013 editions, the format consisted of two different tracks, adopting the distinction between a Model&Solve and a System track. Both tracks featured a selected suite of domains, chosen by means of an open *Call for Problems* stage. This edition omitted a benchmark submission phase and relied on available encodings and instances (mainly from 2013) in view of the short preparation period.

The System track was essentially similar to the format of the current edition. It was conceived with the aim of fostering language standardization, and let the participants compete on given encodings under fixed conditions, e.g. excluding custom problem encodings and domain-tailored evaluation strategies. However, in previous editions, the track was subdivided on the basis of problem complexity, while this edition makes a more fine-grained distinction based on language features; the rationale for this change has been discussed in Section 3.

Unlike the System track, the Model&Solve track in previous editions was instead left open to any (bundle of) solver systems loosely based on a declarative specification language: no constraints were imposed on encodings apart from the requirement that the language had to be declarative. The aims were to encourage the development of new expressive declarative constructs and/or new modeling paradigms, to foster the exchange of ideas between communities in close relationship to ASP, and to stimulate the development of new ad-hoc solving methods. Model&Solve track competitors received textual problem descriptions from a variety of domains and had several months to produce problem encodings together with one or multiple working systems of choice, which could be configured on a per domain basis.

It is worth discussing why the Model&Solve track has been detached from the system competition, and turned into an informal on-site event, summarized at (Calimeri et al., 2014b), in the spirit of the Prolog Programming Contest. Although its goals were ambitious and definitely relevant, it requires a substantial amount of work to organize and, even more, participate in such a track. First of all, fine-tuning a system, or a bundle of systems, and an encoding for an individual domain is a hard task; this, in spite of the fact that contestants have always been encouraged to exchange their solutions freely, tends to give larger teams a clear advantage over others. In addition, the participation from neighboring communities was rather limited, probably due to corresponding genuine competitions and the non-negligible effort of participating. Compared to this, it is easier to com-

pete in an event similar to the System track by submitting a solver system with appropriate support for the respective input language.

Eventually, the scoring scheme has been significantly changed for the sake of simplicity. Indeed, the scoring has been refined throughout editions. At the beginning, it was mainly based on a weighted sum of the number of instances solved within given time and memory limits; in the 2011 and 2013 editions, the scoring scheme has been extended by awarding additional points to systems performing well in terms of elapsed time. For a Decision or Query problem P, each system gained a score $S(P) = S_{solve}(P) + S_{time}(P)$, where S_{solve} and S_{time} ranged from 0 to 50 each: while S_{solve} was linearly dependent on the number of solved instances, S_{time} involved a logarithmic dependence on participants' running times, thus rendering time differences at a common order of magnitude less significant. For Optimization problems, the S_{solve} part was replaced with a scoring formula taking objective values associated with solutions into account, where distances from an (imperfect) optimum were penalized exponentially.

We performed a number of ex-post analyses on the results of the 2011 and 2013 editions, experimenting with different scoring schemes. At first, we found that time quota did not make much difference and accordingly use $S_{solve}(P)$ only (simply denoted $S(P)$ in Section 3.5), which as before linearly reflects the number of solved instances of a Decision or Query problem P. Furthermore, we pondered that absolute objective values are not adequate for scoring solutions for Optimization problems, given that they depend heavily on domains and are susceptible to perturbations; rather than that, relative rankings can draw a more reliable picture. Hence, for an Optimization problem P, we determine $S_{solve}(P)$ by accumulating ranks by solution quality for instances of P. Interestingly, if employed in the latest editions, the winners would have been the same. The new scoring scheme is also closer to those of competitions in neighboring fields, which are considered next.

7.2. Related Competitions

In the following, we briefly overview other competitions in neighboring fields and relate them to the ASP Competition series. Note that we do not aim at contrasting formalisms or respective systems, but are rather interested in the similarities and differences of competition setups. Comparisons of systems from different fields can be found, e.g. in (Aschinger et al., 2011; Calimeri et al., 2014c; Dovier et al., 2007; Mancini et al., 2008).

We start by listing related competitions (in no particular order) along with acronyms, years of latest editions, and recent references, if available:

- IJCAR ATP System Competition – CASC 2014 (CASC, 2014; Sutcliffe, 2014)
- Confluence Competition – CoCo 2014 (CoCo, 2014)
- Configurable SAT Solver Challenge – CSSC 2014 (CSSC, 2014)
- Hardware Model Checking Competition – HWMCC 2014 (HWMCC, 2014)
- OWL Reasoner Evaluation – ORE 2014 (ORE, 2014; Bail et al., 2013)
- QBF Gallery – QBF 2014 (QBF, 2014)
- SAT Competition – SAT-COMP 2014 (SAT, 2014; Järvisalo et al., 2012)
- Satisfiability Modulo Theories Solver Competition – SMT-COMP 2014 (SMT, 2014; Barrett et al., 2013)
- Competition on Software Verification – SV-COMP 2014 (SV-COMP, 2014; Beyer, 2014)
- Syntax-Guided Synthesis Competition – SyGuS-COMP 2014 (SyGuS-COMP, 2014)
- Synthesis Competition – SYNTCOMP 2014 (SYNTCOMP, 2014)
- Termination Competition – termCOMP 2014 (TermCOMP, 2014)
- Max-SAT Evaluation – Max-SAT 2014 (Max-SAT, 2014)
- Mancoosi International Solver Competition – MISC 2012 (MISC, 2012)
- Pseudo-Boolean Competition – PB 2012 (PB, 2012; Järvisalo et al., 2012)
- International CSP Solver Competition – CSC 2009 (CSC, 2009); MiniZinc Challenge 2014 (MiniZinc, 2014; Stuckey et al., 2014)
- International Planning Competition – IPC 2014 (IPC, 2014; Coles et al., 2012)

A quick glance at competition characteristics is given in Table 13, displaying whether a standard input format, optimization tasks, non-ground problem descriptions, and industrial/real-world domains are featured; the last column reports the number of participant systems in latest editions. Notably, almost all competitions rely on a standard language, with the exception of SV-COMP in view of its applied nature. Considerable efforts in dealing with industrial/real-world problems are apparent too. The latter frequently demand for optimization capabilities, central in one third of the competitions, as well as non-ground representations, featured by half of the competitions. Both of these concepts are jointly addressed in the CSC, IPC, and ASP Competition series.

Almost all competitions include tracks, also called divisions or categories. The way tracks are defined depends on the competition at hand, but three main criteria can be identified: language features of inputs, the kind of problem domains, and reasoning tasks to be accomplished. For instance, SAT-COMP distinguishes

Table 13: Overview of related competitions

Comp	Std	Opt	Non-ground	Industrial	Participants
CASC 2014	yes	no	yes	no	31 (+3 demo)
CoCo 2014	yes	no	yes	no	7
CSSC 2014	yes	no	no	yes	22
HWMCC 2014	yes	no	no	yes	34
ORE 2014	yes	no	yes	yes	11
QBF 2014	yes	no	no	yes	16
SAT-COMP 2014	yes	no	no	yes	227
SMT-COMP 2014	yes	no	yes	yes	21
SV-COMP 2014	no	no	yes	yes	15
SyGuS-COMP 2014	yes	no	yes	no	5
SYNTCOMP 2014	yes	no	no	yes	22
termCOMP 2014	yes	no	no	yes	9
Max-SAT 2014	yes	yes	no	yes	46
MISC 2012	yes	yes	no	yes	4
PB 2012	yes	yes	no	yes	18
CSC 2009	yes	yes	yes	yes	14
IPC 2014	yes	yes	yes	yes	87
ASPCOMP 2014	*yes*	*yes*	*yes*	*yes*	*16*

application, combinatorial, and random instances, and the kind of benchmarks has a major impact on participant and winner systems. Interestingly, none of the considered competitions conceives tracks on an intentional complexity basis.

For scoring, almost all competitions rely on the number of solved instances, sometimes with particularities like bonus/malus points in case of errors, and ties are usually broken by taking resources spent into account. For instance, SMT-COMP adopts a rather complex scheme, based on correctness, number of queries answered, and running times. In case of Optimization problems, the quality of solutions matters, such as plan quality in IPC. The simplified scoring scheme adopted in this edition of the ASP Competition is in line with related competitions and pursues similar objectives. In particular, the relative ranking of systems by solution quality has been inspired by MISC, which deals with optimization in the context of software package management.

Many communities (if not all) maintain benchmark repositories, often well-known, well-studied, and classified in terms of dimensions like language features or hardness. Even though the Asparagus platform (Asparagus, 2009) gathers a broad collection of benchmarks for ASP systems, it however appears less central to the ASP community than corresponding initiatives in neighboring fields. Moreover, almost all competitions make use of benchmarks from previous editions and are additionally accompanied by a call for benchmark submissions (which had to be omitted in this edition of the ASP Competition). Notably, benchmarks submitted to SV-COMP are reviewed and possibly revised by organizers and participants in order to improve the benchmark collection; our encoding variants partly go into that direction as well. A subset of the available instances to be fed to participant systems is then typically determined randomly, sometimes applying eligibility criteria, e.g. based on estimated hardness.

Finally, we would like to acknowledge the MINISAT (Eén and Sörensson, 2003) "hack" track, featured by SAT-COMP since its 2009 edition. This track promotes ideas that can be integrated into a common base SAT solver with relatively modest development effort. Such a possibility lowers the threshold for entering the competition, especially for students or small research teams, and is one among several factors for the comparably high participation in SAT-COMP. If a corresponding baseline were available for ASP solving as well, it might encourage more teams to participate in future editions of the ASP Competition.

8. Conclusions

The Fifth ASP Competition has been jointly organized by the University of Calabria (Italy), Aalto University (Finland), and the University of Genova (Italy), affiliated with the 30th International Conference on Logic Programming (ICLP 2014). The main goals were to evaluate the progress of the state of the art in ASP solving and to further push the adoption of the standard language ASP-Core-2. This paper contributes a detailed account of the competition design, participants, and results, and locates the ASP Competition in the spectrum of related events.

The results were orally presented at ICLP 2014 in Vienna, part of the Federated Logic Conference at the Vienna Summer of Logic, where the winners were announced and awarded in a FLoC Olympic Games ceremony on Monday, 21st July 2014. The system CLASP is the overall winner in the **SP** category, and its multi-threaded version CLASP-MT won the **MP** category. Different from previous competition editions, the Model&Solve track was an informal event held on-site: the team consisting of Mario Alviano (University of Calabria), Carmine Dodaro

(University of Calabria), and Wolfgang Faber (University of Huddersfield) won the first place in this newly established ASP Modeling Competition.

Organizing and running the Fifth ASP Competition required a significant effort, and during this journey we faced several issues. It was impossible to settle all of them in this edition of the ASP Competition, and in the following we share central aspects deserving further consideration in the future.

Benchmark collection. It would be a service to the community to create a common repository of domains and instances (say, ASP-LIB), similar to what is done in neighboring fields, e.g. SAT-LIB (SAT-LIB, 2014), SMT-LIB (SMT-LIB, 2014), and QBF-LIB (QBF-LIB, 2014). Current benchmark sources include Asparagus and competition homepages, but none of them provides a comprehensive and maintained benchmark collection. The lack of a standard language had been an obstacle for the general acceptance of a repository in the past, which is now abolished by the ASP-Core-2 standard.

Domain and instance selection. There is a clear need for more domains stemming from real-world applications. Similarly, the current benchmarks do not comprehensively cover ASP modeling concepts; e.g. only two out of seventeen Decision problems in Tracks #1 and #2 take significant advantage of non-tightness (cf. Table 1). The next call for benchmarks should, in our opinion, explicitly target real-world domains as well as modeling concepts. We also feel that the classification by language features was useful in the analysis of competition results and should be maintained. Regarding instance selection, generators for random or crafted instances should always be made publicly available for reference and reusability, while methods to focus on "meaningful" instances only, e.g. (Hoos et al., 2013), should be taken into account as well.

Track design. ASP Competitions are inherently multifaceted and thus complex events, but the need to compare systems and evaluate the progress in the field is of course still central. In this regard, tracks for non-ground (i.e. as of now) and ground (via a common format) inputs could be considered. Moreover, additional tasks such as cautious and brave reasoning could give rise to new tracks.

Scoring scheme. The traditional scoring scheme of ASP Competitions accumulates the results of multiple tracks to determine an overall winner per category. Although track winners are also awarded separately, newcomer and specialized systems that cannot compete in all tracks are penalized by global rankings, which may discourage their participation. An option to be considered in the future is

whether to keep tracks separate, in analogy to SAT Competitions where first places are awarded in each track, while there is no overall competition winner.

Output format. The diversity of ASP Competitions, featuring Decision, Optimization, and Query answering tasks, led to several output formats that solvers must comply with (Krennwallner, 2013). Multiple formats result in a significant burden to both organizers and developers. We thus encourage future simplifications, e.g. by reducing the number of exit codes or making the usage of admitted ones more uniform and independent from specific reasoning tasks.

Modeling competition. The Model&Solve track of past competition editions has been replaced with an on-site event, the ASP Modeling Competition (Calimeri et al., 2014b), in the spirit of the Prolog Programming Contest. This event attracted a significant number of participants, even though it was not widely advertised, as well as the interest of the community, thus calling for future continuation.

Acknowledgments. The organizers of the Fifth ASP Competition would like to thank the ICLP and FLoC officials for the co-location of the event, and especially Thomas Krennwallner, both as chair of the FLoC Olympic Games and co-chair of the Fourth ASP Competition, for his support and feedback in both competition editions. Similarly, we thank Giovambattista (Gb) Ianni, as co-chair of the 2013 edition. We are grateful to the Department of Mathematics and Computer Science at the University of Calabria for providing the computational resources to run the competition. Last but not least, we thank all participants, who worked hard on their systems and made this competition possible (and successful)!

References

Alviano, M., 2014. The pyrunlim tool. URL `https://github.com/alviano/python/`

Alviano, M., Calimeri, F., Charwat, G., Dao-Tran, M., Dodaro, C., Ianni, G., Krennwallner, T., Kronegger, M., Oetsch, J., Pfandler, A., Pührer, J., Redl, C., Ricca, F., Schneider, P., Schwengerer, M., Spendier, L., Wallner, J., Xiao, G., 2013a. The fourth answer set programming competition: Preliminary report. In: Cabalar, P., Son, T. (Eds.), Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013). Vol. 8148 of Lecture Notes in Computer Science. Springer, pp. 42–53.

Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F., 2013b. WASP: A native ASP solver based on constraint learning. In: Cabalar, P., Son, T. (Eds.), Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013). Vol. 8148 of Lecture Notes in Computer Science. Springer, pp. 54–66.

Alviano, M., Dodaro, C., Ricca, F., 2014. Preliminary report on WASP 2.0. In: Konieczny, S., Tompits, H. (Eds.), Proceedings of the Fifteenth International Workshop on Non-Monotonic Reasoning (NMR 2014). TU Vienna, pp. 68–72.

Aschinger, M., Drescher, C., Friedrich, G., Gottlob, G., Jeavons, P., Ryabokon, A., Thorstensen, E., 2011. Optimization methods for the partner units problem. In: Achterberg, T., Beck, C. (Eds.), Proceedings of the Eighth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2011). Vol. 6697 of Lecture Notes in Computer Science. Springer, pp. 4–19.

Asparagus, 2009. Asparagus - a web-based benchmarking environment for answer set programming. URL `http://asparagus.cs.uni-potsdam.de/`

Bail, S., Glimm, B., Jiménez-Ruiz, E., Matentzoglu, N., Parsia, B., Steigmiller, A. (Eds.), 2013. Proceedings of the Third International Workshop on OWL Reasoner Evaluation (ORE 2014). Vol. 1207 of CEUR Workshop Proceedings.

Baral, C., 2003. Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press.

Barrett, C., Deters, M., de Moura, L., Oliveras, A., Stump, A., 2013. 6 years of SMT-COMP. Journal of Automated Reasoning 50 (3), 243–277.

Ben-Eliyahu, R., Dechter, R., 1994. Propositional semantics for disjunctive logic programs. Annals of Mathematics and Artificial Intelligence 12 (1-2), 53–87.

Beyer, D., 2014. Status report on software verification - (competition summary SV-COMP 2014). In: Ábrahám, E., Havelund, K. (Eds.), Proceedings of the Twentieth International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2014). Vol. 8413 of Lecture Notes in Computer Science. Springer, pp. 373–388.

Bomanson, J., Janhunen, T., 2013. Normalizing cardinality rules using merging and sorting constructions. In: Cabalar, P., Son, T. (Eds.), Proceedings of the

Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2013). Vol. 8148 of Lecture Notes in Computer Science. Springer, pp. 187–199.

Brewka, G., Eiter, T., Truszczyński, M., 2011. Answer set programming at a glance. Communications of the ACM 54 (12), 92–103.

Brewka, G., Niemelä, I., Schaub, T., Truszczyński, M., 2002. Workshop on Nonmonotonic Reasoning, Answer Set Programming and Constraints. URL http://www.dagstuhl.de/02381/

Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T., 2013. ASP-Core-2 - input language format. URL https://www.mat.unical.it/aspcomp2013/ASPStandardization/

Calimeri, F., Gebser, M., Maratea, M., Ricca, F., 2014a. The design of the fifth answer set programming competition. In: Leuschel, M., Schrijvers, T. (Eds.), Proceedings of the Thirtieth International Conference on Logic Programming (ICLP 2014). Theory and Practice of Logic Programming 14 (4-5), Online Supplement.

Calimeri, F., Gebser, M., Maratea, M., Ricca, F., 2014b. The fifth answer set programming competition (ASPCOMP 2014). URL http://www.mat.unical.it/aspcomp2014/

Calimeri, F., Ianni, G., Krennwallner, T., Ricca, F., 2012. The answer set programming competition. AI Magazine 33 (4), 114–118.

Calimeri, F., Ianni, G., Ricca, F., 2014c. The third open answer set programming competition. Theory and Practice of Logic Programming 14 (1), 117–135.

CASC, 2014. The seventh IJCAR ATP system competition (CASC-J7 2014). URL http://www.cs.miami.edu/~tptp/CASC/J7/

CoCo, 2014. The third confluence competition (CoCo 2014). URL http://coco.nue.riec.tohoku.ac.jp/2014/

Coles, A., Coles, A., García Olaya, A., Jiménez Celorrio, S., Linares López, C., Sanner, S., Yoon, S., 2012. A survey of the seventh international planning competition. AI Magazine 33 (1), 83–88.

Crawford, J., Baker, A., 1994. Experimental results on the application of satisfiability algorithms to scheduling problems. In: Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI 1994). AAAI Press, pp. 1092–1097.

CSC, 2009. The fourth international CSP solver competition (CSC 2009). URL http://www.cril.univ-artois.fr/CSC09/

CSSC, 2014. The configurable SAT solver challenge (CSSC 2014). URL http://aclib.net/cssc2014/

Dal Palù, A., Dovier, A., Pontelli, E., Rossi, G., 2009. GASP: Answer set programming with lazy grounding. Fundamenta Informaticae 96 (3), 297–322.

Denecker, M., Vennekens, J., Bond, S., Gebser, M., Truszczyński, M., 2009. The second answer set programming competition. In: Erdem, E., Lin, F., Schaub, T. (Eds.), Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2009). Vol. 5753 of Lecture Notes in Computer Science. Springer, pp. 637–654.

Dovier, A., Formisano, A., Pontelli, E., 2007. An experimental comparison of constraint logic programming and answer set programming. In: Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI 2007). AAAI Press, pp. 1622–1625.

East, D., Truszczyński, M., 2001. aspps - an implementation of answer-set programming with propositional schemata. In: Eiter, T., Faber, W., Truszczyński, M. (Eds.), Proceedings of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2001). Vol. 2173 of Lecture Notes in Computer Science. Springer, pp. 402–405.

Eén, N., Sörensson, N., 2003. An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (Eds.), Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT 2003). Vol. 2919 of Lecture Notes in Computer Science. Springer, pp. 502–518.

Eiter, T., Faber, W., Leone, N., Pfeifer, G., 2000. Declarative problem-solving using the DLV system. In: Minker, J. (Ed.), Logic-Based Artificial Intelligence. Vol. 597 of International Series in Engineering and Computer Science. Springer, pp. 79–103.

Eiter, T., 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., Gottlob, G., Mannila, H., 1997. Disjunctive Datalog. ACM Transactions on Database Systems 22 (3), 364–418.

Eiter, T., Ianni, G., Krennwallner, T., 2009. Answer set programming: A primer. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M., Schmidt, R. (Eds.), Proceedings of the Fifth International Reasoning Web Summer School (RW 2009). Vol. 5689 of Lecture Notes in Computer Science. Springer, pp. 40–110.

Erdem, E., Lifschitz, V., 2003. Tight logic programs. Theory and Practice of Logic Programming 3 (4-5), 499–518.

Faber, W., Leone, N., Perri, S., 2012. The intelligent grounder of DLV. In: Erdem, E., Lee, J., Lierler, Y., Pearce, D. (Eds.), Correct Reasoning: Essays on Logic-Based AI in Honour of Vladimir Lifschitz. Vol. 7265 of Lecture Notes in Computer Science. Springer, pp. 247–264.

Faber, W., Leone, N., Pfeifer, G., 2004. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In: Alferes, J., Leite, J. (Eds.), Proceedings of the Ninth European Conference on Logics in Artificial Intelligence (JELIA 2004). Vol. 3229 of Lecture Notes in Computer Science. Springer, pp. 200–212.

Faber, W., Leone, N., Pfeifer, G., 2011. Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence 175 (1), 278–298.

Faber, W., Pfeifer, G., Leone, N., Dell'Armi, T., Ielpa, G., 2008. Design and implementation of aggregate functions in the DLV system. Theory and Practice of Logic Programming 8 (5-6), 545–580.

Fages, F., 1994. Consistency of Clark's completion and the existence of stable models. Journal of Methods of Logic in Computer Science 1, 51–60.

Gebser, M., Janhunen, T., Rintanen, J., 2014a. Answer set programming as SAT modulo acyclicity. In: Schaub, T., Friedrich, G., O'Sullivan, B. (Eds.), Proceedings of the Twenty-First European Conference on Artificial Intelligence (ECAI

2014). Vol. 263 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 351–356.

Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., 2014b. Clingo = ASP + control: Preliminary report. In: Leuschel, M., Schrijvers, T. (Eds.), Proceedings of the Thirtieth International Conference on Logic Programming (ICLP 2014). Theory and Practice of Logic Programming 14 (4-5), Online Supplement.

Gebser, M., Kaufmann, B., Schaub, T., 2012a. Conflict-driven answer set solving: From theory to practice. Artificial Intelligence 187-188, 52–89.

Gebser, M., Kaufmann, B., Schaub, T., 2012b. Multi-threaded ASP solving with clasp. Theory and Practice of Logic Programming 12 (4-5), 525–545.

Gebser, M., Kaufmann, B., Schaub, T., 2013. Advanced conflict-driven disjunctive answer set solving. In: Rossi, F. (Ed.), Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013). IJCAI/AAAI Press, pp. 912–918.

Gebser, M., Liu, L., Namasivayam, G., Neumann, A., Schaub, T., Truszczyński, M., 2007. The first answer set programming system competition. In: Baral, C., Brewka, G., Schlipf, J. (Eds.), Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2007). Vol. 4483 of Lecture Notes in Computer Science. Springer, pp. 3–17.

Gelfond, M., Leone, N., 2002. Logic programming and knowledge representation - the A-Prolog perspective. Artificial Intelligence 138 (1-2), 3–38.

Gelfond, M., Lifschitz, V., 1988. The stable model semantics for logic programming. In: Kowalski, R., Bowen, K. (Eds.), Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP 1988). MIT Press, pp. 1070–1080.

Gelfond, M., Lifschitz, V., 1991. Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385.

Gerevini, A., Haslum, P., Long, D., Saetti, A., Dimopoulos, Y., 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. Artificial Intelligence 173 (5-6), 619–668.

Ghallab, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D., 1998. PDDL - the planning domain definition language. URL http://www.cs.yale.edu/homes/dvm/

Giunchiglia, E., Lierler, Y., Maratea, M., 2006. Answer set programming based on propositional satisfiability. Journal of Automated Reasoning 36 (4), 345–377.

Hoos, H., Kaufmann, B., Schaub, T., Schneider, M., 2013. Robust benchmark set selection for Boolean constraint solvers. In: Nicosia, G., Pardalos, P. (Eds.), Proceedings of the Seventh International Conference on Learning and Intelligent Optimization (LION 2013). Vol. 7997 of Lecture Notes in Computer Science. Springer, pp. 138–152.

HWMCC, 2014. The seventh hardware model checking competition (HWMCC 2014). URL http://fmv.jku.at/hwmcc14cav/

IPC, 2014. The eighth international planning competition (IPC 2014). URL http://ipc.icaps-conference.org/

Janhunen, T., Niemelä, I., 2011. Compact translations of non-disjunctive answer set programs to propositional clauses. In: Balduccini, M., Son, T. (Eds.), Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays Dedicated to Michael Gelfond on the Occasion of His 65th Birthday. Vol. 6565 of Lecture Notes in Computer Science. Springer, pp. 111–130.

Janhunen, T., Niemelä, I., Seipel, D., Simons, P., You, J., 2006. Unfolding partiality and disjunctions in stable model semantics. ACM Transactions on Computational Logic 7 (1), 1–37.

Järvisalo, M., Le Berre, D., Roussel, O., Simon, L., 2012. The international SAT solver competitions. AI Magazine 33 (1), 89–92.

Krennwallner, T., 2013. Answer set solver output. URL https://www.mat.unical.it/aspcomp2013/files/aspoutput.txt

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F., 2006. The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic 7 (3), 499–562.

Lifschitz, V., 2002. Answer set programming and plan generation. Artificial Intelligence 138 (1-2), 39–54.

Lin, F., Zhao, Y., 2004. ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence 157 (1-2), 115–137.

Liu, G., Janhunen, T., Niemelä, I., 2012. Answer set programming via mixed integer programming. In: Brewka, G., Eiter, T., McIlraith, S. (Eds.), Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2012). AAAI Press, pp. 32–42.

Mancini, T., Micaletto, D., Patrizi, F., Cadoli, M., 2008. Evaluating ASP and commercial solvers on the CSPLib. Constraints 13 (4), 407–436.

Marek, V., Truszczyński, M., 1999. Stable models and an alternative logic programming paradigm. In: Apt, K., Marek, V., Truszczyński, M., Warren, D. (Eds.), The Logic Programming Paradigm: A 25-Year Perspective. Springer, pp. 375–398.

Mariën, M., Wittocx, J., Denecker, M., Bruynooghe, M., 2008. SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In: Kleine Büning, H., Zhao, X. (Eds.), Proceedings of the Eleventh International Conference on Theory and Applications of Satisfiability Testing (SAT 2008). Vol. 4996 of Lecture Notes in Computer Science. Springer, pp. 211–224.

Max-SAT, 2014. The ninth Max-SAT evaluation (Max-SAT 2014). URL `http://maxsat.ia.udl.cat/`

MiniZinc, 2014. The MiniZinc challenge 2014. URL `http://www.minizinc.org/challenge2014/challenge.html`

MISC, 2012. The Mancoosi international solver competition (MISC 2012). URL `http://www.mancoosi.org/misc/`

Nguyen, M., Janhunen, T., Niemelä, I., 2011. Translating answer-set programs into bit-vector logic. In: Tompits, H., Abreu, S., Oetsch, J., Pührer, J., Seipel, D., Umeda, M., Wolf, A. (Eds.), Proceedings of the Nineteenth International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2011) and the Twenty-Fifth Workshop on Logic Programming (WLP 2011). Vol. 7773 of Lecture Notes in Computer Science. Springer, pp. 105–116.

Niemelä, I., 1999. Logic programming with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence 25 (3-4), 241–273.

Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., Barry, M., 2001. An A-Prolog decision support system for the space shuttle. In: Ramakrishnan, I. (Ed.), Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages (PADL 2001). Vol. 1990 of Lecture Notes in Computer Science. Springer, pp. 169–183.

ORE, 2014. The OWL reasoner evaluation (ORE 2014). URL http://www.easychair.org/smart-program/VSL2014/ORE-index.html

PB, 2012. The Pseudo-Boolean competition 2012 (PB 2012). URL http://www.cril.univ-artois.fr/PB12/

QBF, 2014. The QBF gallery 2014. URL http://qbf.satisfiability.org/gallery/

QBF-LIB, 2014. The Quantified Boolean Formulas satisfiability library. URL http://www.qbflib.org/

Reiter, R., 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In: Lifschitz, V. (Ed.), Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, pp. 359–380.

Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N., 2012. Team-building with answer set programming in the Gioia-Tauro seaport. Theory and Practice of Logic Programming 12 (3), 361–381.

SAT, 2014. The international SAT competitions (SAT-COMP 2014). URL http://www.satcompetition.org/

SAT-LIB, 2014. The satisfiability library. URL http://www.satlib.org/

Marques-Silva, J., Sakallah, K., 1999. GRASP: A search algorithm for propositional satisfiability. IEEE Transaction on Computers 48 (5), 506–521.

Simons, P., Niemelä, I., Soininen, T., 2002. Extending and implementing the stable model semantics. Artificial Intelligence 138 (1-2), 181–234.

SMT, 2014. The Satisfiability Modulo Theories solver competition (SMT-COMP 2014). URL http://smtcomp.sourceforge.net/2014/

SMT-LIB, 2014. The Satisfiability Modulo Theories library. URL http://smt-lib.org/

Stuckey, P., Feydy, T., Schutt, A., Tack, G., Fischer, J., 2014. The MiniZinc challenge 2008-2013. AI Magazine 35 (2), 55–60.

Sutcliffe, G., 2014. The CADE-24 automated theorem proving system competition - CASC-24. AI Communications 27 (4), 405–416.

SV-COMP, 2014. The third international competition on software verification (SV-COMP 2014). URL http://sv-comp.sosy-lab.org/2014/

SyGuS-COMP, 2014. The syntax-guided synthesis competition (SyGuS-COMP 2014). URL http://sygus.org/

SYNTCOMP, 2014. The synthesis competition (SYNTCOMP 2014). URL http://www.syntcomp.org/

TermCOMP, 2014. The termination competition (termCOMP 2014). URL http://www.termination-portal.org/wiki/Termination_Competition_2014

Tiihonen, J., Soininen, T., Niemelä, I., Sulonen, R., 2003. A practical tool for mass-customising configurable products. In: Folkeson, A., Gralen, K., Norell, M., Sellgren, U. (Eds.), Proceedings of the Fourteenth International Conference on Engineering Design (ICED 2003).

Zhang, L., Madigan, C., Moskewicz, M., Malik, S., 2001. Efficient conflict driven learning in a Boolean satisfiability solver. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD 2001). ACM Press, pp. 279–285.

Table 14: Detailed results of the Fifth ASP Competition (1/3). The benchmark domains are subdivided by tracks, and the entries "D", "O", and "Q" in the **P** column indicate Decision, Optimization, or Query answering tasks, respectively. Further columns provide the scores, cumulative CPU times for runs rewarded with positive scores, and numbers of time- and memory outs for the systems CLASP, LP2BV2+BOOLECTOR, LP2GRAPH, LP2MAXSAT+CLASP, and LP2MIP2. Give-ups of LP2BV2+BOOLECTOR on two *Graph Colouring* instances and LP2MAXSAT+CLASP on one *Visit-all* instance are neither scored nor counted as time- or memory outs. The last five rows accumulate results for tracks and over all domains in which a system participated.

		clasp				**lp2bv2+boolector**				**lp2graph**				**lp2maxsat+clasp**				**lp2mip2**			
Domain	**P**	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO
Labyrinth	D	90.0	3396.0	2	0	15.0	1510.4	17	0	65.0	3201.9	7	0	40.0	2254.0	12	0	0.0	–	20	0
Stable Marriage	D	95.0	2136.6	1	0	0.0	–	0	20	95.0	3122.9	1	0	100.0	3617.5	0	0	0.0	–	0	20
Bottle Filling	D	100.0	170.4	0	0	55.0	2184.1	0	9	5.0	457.8	10	9	50.0	3100.8	1	9	0.0	–	11	9
Graceful Graphs	D	40.0	756.4	12	0	35.0	1253.2	13	0	40.0	509.9	12	0	50.0	1068.1	10	0	0.0	–	20	0
Graph Colouring	D	45.0	834.1	11	0	25.0	25.2	13	0	45.0	814.2	11	0	50.0	1055.0	10	0	30.0	1342.4	14	0
Hanoi Tower	D	85.0	1498.3	3	0	85.0	1022.0	3	0	100.0	347.5	0	0	100.0	935.8	0	0	0.0	–	20	0
Incremental Scheduling	D	0.0	–	5	15	0.0	–	3	17	0.0	–	3	17	0.0	–	3	17	0.0	–	3	17
Knight Tour with Holes	D	0.0	–	0	20	0.0	–	0	20	0.0	–	0	20	0.0	–	0	20	0.0	–	0	20
Nomystery	D	40.0	621.8	10	2	45.0	1363.8	3	8	45.0	834.2	6	5	45.0	726.6	7	4	5.0	196.6	10	9
Partner Units	D	20.0	290.6	16	0	0.0	–	1	19	15.0	729.6	1	16	20.0	1133.4	0	16	0.0	–	1	19
Permutation Pattern Matching	D	70.0	1556.6	1	5	40.0	735.8	0	12	65.0	1258.4	0	7	65.0	1589.5	0	7	35.0	1134.6	1	12
Qualitative Spatial Reasoning	D	100.0	2124.4	0	0	0.0	–	19	1	50.0	688.0	10	0	50.0	716.7	10	0	10.0	1140.9	18	0
Reachability	Q	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Ricochet Robots	D	100.0	2464.4	0	0	100.0	952.4	0	0	100.0	749.5	0	0	100.0	2267.5	0	0	0.0	–	20	0
Sokoban	D	35.0	1076.9	13	0	35.0	852.6	6	7	30.0	739.5	13	1	35.0	1043.9	12	1	0.0	–	18	2
Solitaire	D	85.0	112.6	3	0	80.0	1026.0	4	0	80.0	263.7	4	0	80.0	733.6	4	0	25.0	1287.6	15	0
Visit-all	D	55.0	1846.2	9	0	100.0	2819.7	0	0	65.0	724.0	7	0	60.0	922.9	7	0	35.0	560.7	13	0
Weighted-Sequence Problem	D	85.0	1551.3	3	0	70.0	2933.0	6	0	95.0	1477.1	1	0	90.0	2892.9	2	0	0.0	–	20	0
Connected Still Life	O	72.5	9613.3	16	0	–	–	–	–	–	–	–	–	15.0	3.3	17	0	10.0	424.8	18	0
Crossing Minimization	O	95.6	4466.4	6	0	–	–	–	–	–	–	–	–	30.0	637.2	14	0	0.0	–	20	0
Maximal Clique	O	100.0	938.3	0	0	–	–	–	–	–	–	–	–	65.0	1732.2	7	0	100.0	99.0	0	0
Valves Location	O	53.8	6000.7	19	0	–	–	–	–	–	–	–	–	5.0	156.9	6	13	0.0	–	0	20
Abstract Dialectical Frameworks	O	100.0	1555.0	0	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Complex Optimization	D	85.0	553.6	3	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Minimal Diagnosis	D	100.0	199.1	0	0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Strategic Companies	Q	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Track #1		185.0	5532.6	3	0	15.0	1510.4	17	20	160.0	6324.8	8	0	140.0	5871.5	12	0	0.0	0.0	20	20
Track #2		860.0	14904.0	86	42	670.0	15167.8	71	93	735.0	9593.4	78	75	795.0	18186.7	66	74	140.0	5662.8	184	88
Track #3		321.9	21018.7	41	0	0.0	0.0	0	0	0.0	0.0	0	0	115.0	2529.6	44	13	110.0	523.8	38	20
Track #4		285.0	2307.7	3	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
Overall		**1651.9**	**43763.0**	**133**	**42**	**685.0**	**16678.2**	**88**	**113**	**895.0**	**15918.2**	**86**	**75**	**1050.0**	**26587.8**	**122**	**87**	**250.0**	**6186.6**	**242**	**128**

Table 15: Detailed results of the Fifth ASP Competition (2/3). The benchmark domains are subdivided by tracks, and the entries "D", "O", and "Q" in the **P** column indicate Decision, Optimization, or Query answering tasks, respectively. Further columns provide the scores, cumulative CPU times for runs rewarded with positive scores, and numbers of time- and memory outs for the systems LP2NORMAL2+CLASP, LP2SAT3+GLUCOSE, LP2SAT3+LINGELING, WASP-1, and WASP-1.5. Due to technical difficulties with handling the encoding, WASP-1 was disqualified in *Valves Location*. The last five rows accumulate results for tracks and over all domains in which a system participated.

		lp2normal2+clasp				**lp2sat3+glucose**				**lp2sat3+lingeling**				**wasp-1**				**wasp-1.5**			
Domain	**P**	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO
Labyrinth	D	70.0	1831.5	6	0	35.0	1668.9	13	0	30.0	1600.7	14	0	45.0	2098.3	11	0	60.0	2754.6	1	7
Stable Marriage	D	95.0	2911.3	1	0	95.0	3140.1	1	0	65.0	2060.7	7	0	65.0	1657.6	0	7	100.0	1164.1	0	0
Bottle Filling	D	100.0	270.7	0	0	5.0	439.3	10	9	55.0	4622.3	0	9	100.0	1700.0	0	0	100.0	167.3	0	0
Graceful Graphs	D	40.0	1356.5	12	0	40.0	497.9	12	0	45.0	2038.5	11	0	5.0	559.1	19	0	20.0	569.7	16	0
Graph Colouring	D	45.0	608.9	11	0	45.0	811.9	11	0	50.0	873.7	10	0	25.0	202.1	15	0	35.0	409.3	13	0
Hanoi Tower	D	100.0	1307.0	0	0	100.0	316.5	0	0	85.0	794.5	3	0	40.0	2670.4	12	0	70.0	1221.5	6	0
Incremental Scheduling	D	0.0	–	3	17	0.0	–	3	17	0.0	–	3	17	0.0	–	5	15	0.0	–	3	17
Knight Tour with Holes	D	0.0	–	0	20	0.0	–	0	20	0.0	–	0	20	0.0	–	18	2	0.0	–	0	20
Nomystery	D	40.0	85.3	8	4	45.0	877.9	6	5	40.0	605.8	8	4	25.0	1501.2	15	0	35.0	1032.0	5	8
Partner Units	D	20.0	905.2	0	16	15.0	828.3	1	16	15.0	1236.9	1	16	5.0	125.5	3	16	15.0	994.7	2	15
Permutation Pattern Matching	D	65.0	1542.3	0	7	65.0	1265.1	0	7	65.0	1061.5	0	7	55.0	1053.0	1	8	65.0	611.3	1	6
Qualitative Spatial Reasoning	D	100.0	1701.7	0	0	50.0	684.0	10	0	50.0	926.0	10	0	90.0	2276.1	2	0	75.0	1992.9	0	5
Reachability	Q	–	–	–	–	–	–	–	–	–	–	–	–	80.0	2182.1	0	4	80.0	2408.0	0	4
Ricochet Robots	D	100.0	1255.5	0	0	100.0	891.8	0	0	100.0	945.0	0	0	30.0	1859.3	14	0	35.0	1144.3	13	0
Sokoban	D	30.0	581.4	14	0	30.0	831.6	13	1	25.0	245.5	14	1	5.0	60.0	19	0	30.0	742.5	12	2
Solitaire	D	80.0	627.9	4	0	80.0	298.1	4	0	85.0	561.8	3	0	75.0	685.5	5	0	85.0	609.3	3	0
Visit-all	D	65.0	918.6	7	0	65.0	783.7	7	0	90.0	2997.0	2	0	25.0	28.9	15	0	30.0	20.1	14	0
Weighted-Sequence Problem	D	85.0	2588.4	3	0	95.0	1751.5	1	0	85.0	3093.4	3	0	45.0	2608.0	11	0	65.0	2699.5	7	0
Connected Still Life	O	20.0	151.0	16	0	–	–	–	–	–	–	–	–	96.3	10202.5	17	0	85.6	9601.2	17	0
Crossing Minimization	O	70.0	1931.6	6	0	–	–	–	–	–	–	–	–	58.8	10800.0	20	0	40.6	7200.0	20	0
Maximal Clique	O	15.0	1344.0	17	0	–	–	–	–	–	–	–	–	69.4	11451.0	17	0	55.0	12000.0	20	0
Valves Location	O	20.0	1240.4	18	0	–	–	–	–	–	–	–	–	Disq.	Disq.	Disq.	Disq.	5.0	1.1	13	6
Abstract Dialectical Frameworks	O	80.0	1617.9	4	0	–	–	–	–	–	–	–	–	35.6	2699.2	15	0	35.6	2710.5	15	0
Complex Optimization	D	100.0	1340.7	0	0	–	–	–	–	–	–	–	–	0.0	–	20	0	0.0	–	20	0
Minimal Diagnosis	D	100.0	519.9	0	0	–	–	–	–	–	–	–	–	65.0	4793.2	7	0	65.0	4831.3	7	0
Strategic Companies	Q	–	–	–	–	–	–	–	–	–	–	–	–	0.0	–	0	20	0.0	–	0	20
Track #1		165.0	4742.8	7	0	130.0	4809.0	14	0	95.0	3661.4	21	0	110.0	3755.9	11	7	160.0	3918.7	1	7
Track #2		870.0	13749.4	62	64	735.0	10277.6	78	75	790.0	20001.9	68	74	605.0	17511.2	154	45	740.0	14622.4	95	77
Track #3		125.0	4667.0	57	0	0.0	0.0	0	0	0.0	0.0	0	0	224.5	32453.5	54	0	186.2	28802.3	70	6
Track #4		280.0	3478.5	4	0	0.0	0.0	0	0	0.0	0.0	0	0	100.6	7492.4	42	20	100.6	7541.8	42	20
Overall		**1440.0**	**26637.7**	**130**	**64**	**865.0**	**15086.6**	**92**	**75**	**885.0**	**23663.2**	**89**	**74**	**1040.1**	**61213.0**	**261**	**72**	**1186.8**	**54885.2**	**208**	**110**

Table 16: Detailed results of the Fifth ASP Competition (3/3). The benchmark domains are subdivided by tracks, and the entries "D", "O", and "Q" in the **P** column indicate Decision, Optimization, or Query answering tasks, respectively. Further columns provide the scores, cumulative CPU times for runs rewarded with positive scores, and numbers of time- and memory outs for the systems WASP-2, WASP-WPM1-ONLY-WEAK, CLASP-MT, LP2MIP2-MT, and LP2SAT3+PLINGELING-MT. Due to incorrect parallel optimization w.r.t. non-HCF programs, CLASP-MT was disqualified in *Abstract Dialectical Frameworks*. The last five rows accumulate results for tracks and over all domains in which a system participated.

		wasp-2				**wasp-wpm1-only-weak**				**clasp-mt**				**lp2mip2-mt**				**lp2sat3+plingeling-mt**			
Domain	**P**	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO
Labyrinth	D	60.0	2787.5	1	7	–	–	–	–	95.0	512.5	1	0	0.0	–	20	0	30.0	1659.3	14	0
Stable Marriage	D	100.0	1161.5	0	0	–	–	–	–	100.0	1427.3	0	0	0.0	–	0	20	100.0	2570.9	0	0
Bottle Filling	D	100.0	169.7	0	0	–	–	–	–	100.0	124.7	0	0	0.0	–	0	20	55.0	4023.4	0	9
Graceful Graphs	D	20.0	552.4	16	0	–	–	–	–	60.0	1117.3	8	0	0.0	–	20	0	55.0	1172.0	9	0
Graph Colouring	D	35.0	393.0	13	0	–	–	–	–	95.0	1691.3	1	0	75.0	3852.3	5	0	65.0	1681.4	7	0
Hanoi Tower	D	70.0	1203.3	6	0	–	–	–	–	100.0	919.2	0	0	0.0	–	20	0	100.0	613.2	0	0
Incremental Scheduling	D	0.0	–	3	17	–	–	–	–	0.0	–	3	17	0.0	–	3	17	0.0	–	3	17
Knight Tour with Holes	D	0.0	–	0	20	–	–	–	–	0.0	–	0	20	0.0	–	0	20	0.0	–	0	20
Nomystery	D	35.0	663.5	5	8	–	–	–	–	40.0	620.1	7	5	5.0	393.5	6	13	45.0	809.6	4	7
Partner Units	D	15.0	1205.3	2	15	–	–	–	–	25.0	488.2	15	0	0.0	–	1	19	10.0	234.9	0	18
Permutation Pattern Matching	D	65.0	538.9	1	6	–	–	–	–	70.0	1040.8	1	5	35.0	1133.2	1	12	65.0	552.4	0	7
Qualitative Spatial Reasoning	D	75.0	2171.4	0	5	–	–	–	–	95.0	1518.5	1	0	0.0	–	14	6	50.0	482.0	10	0
Reachability	Q	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Ricochet Robots	D	35.0	1157.5	13	0	–	–	–	–	100.0	744.3	0	0	0.0	–	20	0	100.0	400.7	0	0
Sokoban	D	30.0	598.7	12	2	–	–	–	–	50.0	1205.5	9	1	0.0	–	15	5	40.0	813.8	11	1
Solitaire	D	85.0	609.4	3	0	–	–	–	–	85.0	18.0	3	0	50.0	1801.5	10	0	85.0	319.0	3	0
Visit-all	D	30.0	17.6	14	0	–	–	–	–	95.0	2470.8	1	0	40.0	1193.4	12	0	100.0	1260.1	0	0
Weighted-Sequence Problem	D	65.0	2659.8	7	0	–	–	–	–	80.0	1230.3	2	2	0.0	–	11	9	95.0	1970.3	1	0
Connected Still Life	O	85.6	9601.2	17	0	45.6	12000.0	20	0	100.0	9648.1	16	0	10.0	287.5	18	0	–	–	–	–
Crossing Minimization	O	40.6	7200.0	20	0	0.0	–	20	0	100.0	6142.4	7	0	0.0	–	20	0	–	–	–	–
Maximal Clique	O	55.0	12000.0	20	0	0.0	–	20	0	90.0	4544.9	4	0	100.0	95.7	0	0	–	–	–	–
Valves Location	O	5.0	1.1	13	6	Disq.		Disq.	Disq.	90.0	9323.6	17	0	0.0	–	0	20	–	–	–	–
Abstract Dialectical Frameworks	O	–	–	–	–	25.0	451.4	15	0	Disq.		Disq.	Disq.	–	–	–	–	–	–	–	–
Complex Optimization	D	–	–	–	–	–	–	–	–	100.0	461.7	0	0	–	–	–	–	–	–	–	–
Minimal Diagnosis	D	–	–	–	–	–	–	–	–	100.0	191.4	0	0	–	–	–	–	–	–	–	–
Strategic Companies	Q	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Track #1		160.0	3949.0	1	7	0.0	0.0	0	0	195.0	1939.8	1	0	0.0	0.0	20	20	130.0	4230.2	14	0
Track #2		660.0	11940.5	95	73	0.0	0.0	0	0	995.0	13189.0	51	50	205.0	8373.9	138	121	865.0	14332.8	48	79
Track #3		186.2	28802.3	70	6	45.6	12000.0	60	0	380.0	29659.0	44	0	110.0	383.2	38	20	0.0	0.0	0	0
Track #4		0.0	0.0	0	0	25.0	451.4	15	0	200.0	653.1	0	0	0.0	0.0	0	0	0.0	0.0	0	0
Overall		**1006.2**	**44691.8**	**166**	**86**	**70.6**	**12451.4**	**75**	**0**	**1770.0**	**45440.9**	**96**	**50**	**315.0**	**8757.1**	**196**	**161**	**995.0**	**18563.0**	**62**	**79**

Table 17: Detailed results for systems in the **SP** category on alternative encodings (1/3). The benchmark domains are subdivided by tracks, and the entries "D" and "O" in the **P** column indicate Decision or Optimization tasks, respectively. Further columns provide the scores, cumulative CPU times for runs rewarded with positive scores, and numbers of time- and memory outs for the systems CLASP, LP2BV2+BOOLECTOR, LP2GRAPH, and LP2MAXSAT+CLASP. Give-ups of LP2BV2+BOOLECTOR and LP2MAXSAT+CLASP on one *Visit-all* instance each are neither scored nor counted as time- or memory outs. The last five rows accumulate results for tracks and over all domains in which a system participated.

		clasp				**lp2bv2+boolector**				**lp2graph**				**lp2maxsat+clasp**			
Domain	**P**	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO
Graph Colouring	D	85.0	2039.6	3	0	65.0	1771.0	7	0	80.0	2513.4	4	0	85.0	1676.3	3	0
Hanoi Tower	D	100.0	648.8	0	0	100.0	199.5	0	0	100.0	51.0	0	0	100.0	252.9	0	0
Knight Tour with Holes	D	5.0	9.1	19	0	0.0	–	20	0	5.0	41.1	19	0	0.0	–	20	0
Labyrinth	D	90.0	2005.7	2	0	15.0	1291.8	17	0	70.0	3191.4	6	0	25.0	1144.4	15	0
Stable Marriage	D	100.0	106.7	0	0	100.0	1674.5	0	0	100.0	418.1	0	0	100.0	415.2	0	0
Visit-all	D	70.0	700.2	6	0	85.0	2896.1	2	0	60.0	1420.0	8	0	60.0	451.3	7	0
Bottle Filling	D	100.0	94.8	0	0	55.0	1891.1	0	9	5.0	562.0	10	9	55.0	2623.7	0	9
Graceful Graphs	D	35.0	823.8	13	0	15.0	825.5	17	0	25.0	445.0	15	0	25.0	756.1	15	0
Incremental Scheduling	D	65.0	639.1	2	5	20.0	5.0	6	10	25.0	118.9	6	9	25.0	92.0	6	9
Nomystery	D	35.0	43.0	13	0	45.0	1166.1	10	1	40.0	306.9	12	0	50.0	981.3	10	0
Partner Units	D	65.0	1009.7	7	0	20.0	607.1	16	0	20.0	936.0	16	0	40.0	1283.7	12	0
Permutation Pattern Matching	D	75.0	529.9	0	5	15.0	52.5	6	11	75.0	1943.0	0	5	75.0	1547.8	0	5
Qualitative Spatial Reasoning	D	100.0	1528.1	0	0	0.0	–	20	0	50.0	494.3	10	0	50.0	543.7	10	0
Ricochet Robots	D	100.0	2468.0	0	0	100.0	928.5	0	0	100.0	987.8	0	0	95.0	2958.7	1	0
Sokoban	D	35.0	946.6	13	0	35.0	791.6	6	7	30.0	650.7	13	1	35.0	826.2	12	1
Solitaire	D	95.0	597.0	1	0	95.0	164.1	1	0	95.0	74.4	1	0	95.0	151.8	1	0
Weighted-Sequence Problem	D	100.0	259.4	0	0	100.0	620.5	0	0	100.0	902.7	0	0	100.0	980.3	0	0
Connected Still Life	O	92.5	9406.2	15	0	–	–	–	–	–	–	–	–	15.0	4.2	17	0
Crossing Minimization	O	99.4	608.7	1	0	–	–	–	–	–	–	–	–	35.0	690.7	13	0
Maximal Clique	O	100.0	210.5	0	0	–	–	–	–	–	–	–	–	65.0	1718.1	7	0
Valves Location	O	96.9	11400.7	19	0	–	–	–	–	–	–	–	–	5.0	85.7	6	13
Abstract Dialectical Frameworks	O	94.4	1891.1	2	0	–	–	–	–	–	–	–	–	–	–	–	–
Complex Optimization	D	80.0	939.4	4	0	–	–	–	–	–	–	–	–	–	–	–	–
Minimal Diagnosis	D	100.0	106.7	0	0	–	–	–	–	–	–	–	–	–	–	–	–
Track #1		450.0	5510.1	30	0	365.0	7832.9	46	0	415.0	7635.0	37	0	370.0	3940.1	45	0
Track #2		805.0	8939.4	49	10	500.0	7052.0	82	38	565.0	7421.7	83	24	645.0	12745.3	67	24
Track #3		388.8	21626.1	35	0	0.0	0.0	0	0	0.0	0.0	0	0	120.0	2498.7	43	13
Track #4		274.4	2937.2	6	0	0.0	0.0	0	0	0.0	0.0	0	0	0.0	0.0	0	0
Overall		**1918.2**	**39012.8**	**120**	**10**	**865.0**	**14884.9**	**128**	**38**	**980.0**	**15056.7**	**120**	**24**	**1135.0**	**19184.1**	**155**	**37**

Table 18: Detailed results for systems in the **SP** category on alternative encodings (2/3). The benchmark domains are subdivided by tracks, and the entries "D" and "O" in the **P** column indicate Decision or Optimization tasks, respectively. Further columns provide the scores, cumulative CPU times for runs rewarded with positive scores, and numbers of time- and memory outs for the systems LP2MIP2, LP2NORMAL2+CLASP, LP2SAT3+GLUCOSE, and LP2SAT3+LINGELING. The last five rows accumulate results for tracks and over all domains in which a system participated.

		lp2mip2				**lp2normal2+clasp**				**lp2sat3+glucose**				**lp2sat3+lingeling**			
Domain	**P**	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO
Graph Colouring	D	15.0	266.4	17	0	85.0	1558.7	3	0	80.0	2597.6	4	0	90.0	2306.2	2	0
Hanoi Tower	D	0.0	–	20	0	100.0	266.1	0	0	100.0	47.9	0	0	100.0	105.7	0	0
Knight Tour with Holes	D	90.0	1346.6	2	0	10.0	246.3	18	0	0.0	–	20	0	0.0	–	20	0
Labyrinth	D	0.0	–	20	0	75.0	3542.9	5	0	40.0	1586.8	12	0	25.0	1783.0	15	0
Stable Marriage	D	100.0	753.1	0	0	100.0	403.4	0	0	100.0	442.9	0	0	100.0	452.9	0	0
Visit-all	D	85.0	1570.1	3	0	65.0	218.5	7	0	60.0	1296.6	8	0	65.0	570.2	7	0
Bottle Filling	D	0.0	–	11	9	100.0	172.2	0	0	5.0	537.5	10	9	30.0	2089.8	5	9
Graceful Graphs	D	0.0	–	20	0	35.0	1012.1	13	0	25.0	442.0	15	0	25.0	777.3	15	0
Incremental Scheduling	D	20.0	2.2	6	10	65.0	1346.1	2	5	25.0	121.3	6	9	25.0	198.2	6	9
Nomystery	D	15.0	309.4	16	1	45.0	839.1	11	0	40.0	315.9	12	0	50.0	1267.9	10	0
Partner Units	D	0.0	–	20	0	45.0	890.9	11	0	20.0	898.5	16	0	30.0	1212.5	14	0
Permutation Pattern Matching	D	40.0	716.2	4	8	75.0	2006.2	0	5	70.0	1446.0	1	5	75.0	869.8	0	5
Qualitative Spatial Reasoning	D	20.0	1665.8	16	0	100.0	1159.7	0	0	50.0	493.1	10	0	50.0	754.7	10	0
Ricochet Robots	D	0.0	–	20	0	100.0	2307.7	0	0	100.0	974.1	0	0	100.0	828.7	0	0
Sokoban	D	0.0	–	18	2	30.0	455.5	14	0	30.0	640.9	13	1	25.0	211.2	14	1
Solitaire	D	25.0	1198.4	15	0	100.0	586.4	0	0	95.0	73.4	1	0	95.0	157.8	1	0
Weighted-Sequence Problem	D	75.0	2875.2	5	0	100.0	81.5	0	0	100.0	898.3	0	0	100.0	713.2	0	0
Connected Still Life	O	5.0	496.5	19	0	20.0	111.0	16	0	–	–	–	–	–	–	–	–
Crossing Minimization	O	85.0	1761.3	3	0	55.0	1628.3	9	0	–	–	–	–	–	–	–	–
Maximal Clique	O	100.0	96.8	0	0	20.0	1483.0	16	0	–	–	–	–	–	–	–	–
Valves Location	O	5.0	182.8	1	18	24.4	1811.7	18	0	–	–	–	–	–	–	–	–
Abstract Dialectical Frameworks	O	–	–	–	–	75.0	1133.1	5	0	–	–	–	–	–	–	–	–
Complex Optimization	D	–	–	–	–	100.0	1108.3	0	0	–	–	–	–	–	–	–	–
Minimal Diagnosis	D	–	–	–	–	100.0	225.8	0	0	–	–	–	–	–	–	–	–
Track #1		290.0	3936.2	62	0	435.0	6235.9	33	0	380.0	5971.8	44	0	380.0	5218.0	44	0
Track #2		195.0	6767.2	151	30	795.0	10857.4	51	10	560.0	6841.0	84	24	605.0	9081.1	75	24
Track #3		195.0	2537.4	23	18	119.4	5034.0	59	0	0.0	0.0	0	0	0.0	0.0	0	0
Track #4		0.0	0.0	0	0	275.0	2467.2	5	0	0.0	0.0	0	0	0.0	0.0	0	0
Overall		**680.0**	**13240.8**	**236**	**48**	**1624.4**	**24594.5**	**148**	**10**	**940.0**	**12812.8**	**128**	**24**	**985.0**	**14299.1**	**119**	**24**

Table 19: Detailed results for systems in the **SP** category on alternative encodings (3/3). The benchmark domains are subdivided by tracks, and the entries "D" and "O" in the **P** column indicate Decision or Optimization tasks, respectively. Further columns provide the scores, cumulative CPU times for runs rewarded with positive scores, and numbers of time- and memory outs for the systems WASP-1, WASP-1.5, WASP-2, and WASP-WPM1-ONLY-WEAK. The last five rows accumulate results for tracks and over all domains in which a system participated.

		wasp-1				**wasp-1.5**				**wasp-2**				**wasp-wpm1-only-weak**			
Domain	**P**	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO	Score	Time	TO	MO
Graph Colouring	D	25.0	480.9	15	0	60.0	1908.0	8	0	60.0	1949.8	8	0	–	–	–	–
Hanoi Tower	D	75.0	790.9	5	0	100.0	523.7	0	0	100.0	544.9	0	0	–	–	–	–
Knight Tour with Holes	D	0.0	–	20	0	5.0	99.6	19	0	5.0	101.2	19	0	–	–	–	–
Labyrinth	D	40.0	1448.2	12	0	60.0	1843.7	2	6	60.0	1829.1	2	6	–	–	–	–
Stable Marriage	D	100.0	615.4	0	0	100.0	137.5	0	0	100.0	155.4	0	0	–	–	–	–
Visit-all	D	30.0	36.7	14	0	60.0	658.0	8	0	60.0	514.4	8	0	–	–	–	–
Bottle Filling	D	95.0	1001.0	0	1	100.0	116.7	0	0	100.0	114.2	0	0	–	–	–	–
Graceful Graphs	D	0.0	–	20	0	20.0	483.2	16	0	20.0	483.9	16	0	–	–	–	–
Incremental Scheduling	D	35.0	372.0	11	2	65.0	1324.5	2	5	65.0	1323.2	2	5	–	–	–	–
Nomystery	D	35.0	684.3	13	0	40.0	410.9	11	1	40.0	410.6	11	1	–	–	–	–
Partner Units	D	15.0	308.7	17	0	60.0	1878.8	8	0	60.0	1879.8	8	0	–	–	–	–
Permutation Pattern Matching	D	70.0	505.7	0	6	75.0	570.2	0	5	75.0	571.2	0	5	–	–	–	–
Qualitative Spatial Reasoning	D	95.0	2408.3	1	0	70.0	841.6	1	5	70.0	840.3	1	5	–	–	–	–
Ricochet Robots	D	0.0	–	20	0	35.0	1250.3	13	0	35.0	1248.9	13	0	–	–	–	–
Sokoban	D	5.0	58.2	19	0	30.0	448.0	12	2	30.0	448.6	13	1	–	–	–	–
Solitaire	D	95.0	377.8	1	0	95.0	38.8	1	0	95.0	39.3	1	0	–	–	–	–
Weighted-Sequence Problem	D	70.0	1214.9	6	0	100.0	325.0	0	0	100.0	325.3	0	0	–	–	–	–
Connected Still Life	O	18.8	604.2	17	0	53.1	4802.8	17	0	53.1	4802.7	17	0	0.0	–	20	0
Crossing Minimization	O	60.0	12000.0	20	0	56.9	12000.0	20	0	56.9	12000.0	20	0	0.0	–	20	0
Maximal Clique	O	68.8	11458.9	17	0	54.4	12000.0	20	0	54.4	12000.0	20	0	0.0	–	20	0
Valves Location	O	0.0	–	20	0	10.0	561.4	9	9	10.0	559.7	9	9	0.0	–	20	0
Abstract Dialectical Frameworks	O	36.3	2468.1	15	0	36.3	2468.1	15	0	–	–	–	–	0.0	–	20	0
Complex Optimization	D	0.0	–	20	0	0.0	–	20	0	–	–	–	–	–	–	–	–
Minimal Diagnosis	D	100.0	925.1	0	0	100.0	926.6	0	0	–	–	–	–	–	–	–	–
Track #1		270.0	3372.1	66	0	385.0	5170.5	37	6	385.0	5094.8	37	6	0.0	0.0	0	0
Track #2		515.0	6930.9	108	9	690.0	7688.0	64	18	690.0	7685.3	65	17	0.0	0.0	0	0
Track #3		147.6	24063.1	74	0	174.4	29364.2	66	9	174.4	29362.4	66	9	0.0	0.0	80	0
Track #4		136.3	3393.2	35	0	136.3	3394.7	35	0	0.0	0.0	0	0	0.0	0.0	20	0
Overall		**1068.9**	**37759.3**	**283**	**9**	**1385.7**	**45617.4**	**202**	**33**	**1249.4**	**42142.5**	**168**	**32**	**0.0**	**0.0**	**100**	**0**