TLP **14** (1): 117–135, 2014. © Cambridge University Press 2012
doi:10.1017/S1471068412000105 First published online 6 September 2012

The third open answer set programming competition

FRANCESCO CALIMERI, GIOVAMBATTISTA IANNI and FRANCESCO RICCA
Dipartimento di Matematica, Università della Calabria, Italy
(*e-mail:* {calimeri,ianni,ricca}@mat.unical.it)

submitted 24 August 2011; revised 12 January 2012; accepted 4 June 2012

Abstract

Answer Set Programming (ASP) is a well-established paradigm of declarative programming in close relationship with other declarative formalisms such as SAT Modulo Theories, Constraint Handling Rules, FO(.), PDDL and many others. Since its first informal editions, ASP systems have been compared in the now well-established ASP Competition. The Third (Open) ASP Competition, as the sequel to the ASP Competitions Series held at the University of Potsdam in Germany (2006–2007) and at the University of Leuven in Belgium in 2009, took place at the University of Calabria (Italy) in the first half of 2011. Participants competed on a pre-selected collection of benchmark problems, taken from a variety of domains as well as real world applications. The Competition ran on two tracks: the Model and Solve (M&S) Track, based on an open problem encoding, and open language, and open to any kind of system based on a declarative specification paradigm; and the System Track, run on the basis of fixed, public problem encodings, written in a standard ASP language. This paper discusses the format of the competition and the rationale behind it, then reports the results for both tracks. Comparison with the second ASP competition and state-of-the-art solutions for some of the benchmark domains is eventually discussed.

KEYWORDS: answer set programming, logic programming, declarative languages, artificial intelligence competitions

1 Introduction

Answer Set Programming (ASP) is a declarative approach to computer programming stemming roots in the area of nonmonotonic reasoning and logic programming (Gelfond and Lifschitz 1991; Marek and Truszczyński 1999; Niemelä 1999). The main advantage of ASP[1] is its high declarative nature combined with a relatively high expressive power (Dantsin *et al.* 2001). After some pioneering work (Bell *et al.* 1994; Subrahmanian *et al.* 1995), nowadays there are a number of systems that support ASP and its variants (Simons *et al.* 2002; Janhunen and Niemelä 2004; Lierler and Maratea 2004; Lin and Zhao 2004; Anger *et al.* 2005; Leone *et al.* 2006; Gebser *et al.*

[1] For introductory material on ASP, the reader might refer to (Baral 2003; Eiter *et al.* 2009).

2007; Dal Palù *et al.* 2009; Lefèvre and Nicolas 2009b). The availability of some efficient systems make ASP a powerful tool for developing advanced applications in several fields, ranging from Artificial Intelligence (Baral and Gelfond 2000; Balduccini *et al.* 2001; Baral and Uyan 2001; Franconi *et al.* 2001; Nogueira *et al.* 2001; Wasp 2003; Friedrich and Ivanchenko 2008) to Information Integration (Leone *et al.* 2005; Marileo and Bertossi 2010), Knowledge Management (Bardadym 1996; Baral 2003; Grasso *et al.* 2009), Bioinformatics (Palopoli *et al.* 2005; Dovier 2011; Gebser *et al.* 2011), and has stimulated some interest also in industry (Grasso *et al.* 2010; Ricca *et al.* 2010).

ASP systems are evaluated in the now well-established ASP Competitions, that started with two informal trials at ASP Dagstuhl meetings in 2002 and 2005. The present competition, held at the University of Calabria (Italy), is the third official edition, since the rules of the contest were formalized and implemented in the first two "official" ASP Competitions (Gebser *et al.* 2007; Denecker *et al.* 2009). Besides comparing ASP systems with each other, one of the goals of the competition is to benchmark similar systems and declarative paradigms close in spirit to ASP. To this end, the Third ASP Competition featured two tracks: the Model and Solve Competition Track (M&S Track from now on), based on an open problem encoding, open language basis, and open to any system based on a declarative specification paradigm; and the System Competition Track, based on a fixed problem encodings, written in a standard ASP language. The M&S Competition Track essentially follows the direction of the previous ASP Competition; the System Competition Track (System Track from now on) was conceived in order to compare participant systems on the basis of fixed input language and fixed conditions.

A preliminary work reporting results of the System Track appeared in Calimeri *et al.* 2011b; this paper extends that work in the following respects:

- detailed results of the System Track, which now include non-participant systems such as parallel solvers and some latecomers;
- description of the problem categories, including those appearing in the M&S Track only;
- discussion of the rationale and the rules of the M&S Track, presentation of competitors and results of the Track;
- a number of comparisons; namely, we show:
  - — how the winner of the Second ASP Competition performed on this edition's benchmarks;
  - — whenever applicable, how participants to this edition performed on the former Competition benchmarks;
  - — for systems to which this is applicable, whether and how performance changed when switching from the System Track settings to the more liberal settings of the M&S competition;
  - — for a selection of benchmarks, how the participants performed against some known state-of-the-art solutions; these ranged from specialized algorithms/systems to tailored ad-hoc solutions based on constraint programming and/or SAT.

The remainder of the paper is structured as follows: in Section 2 we discuss the Competition rationale, the subsequent format and regulations for both Tracks, and we briefly overview the standard language adopted in the System Track; Section 3 illustrates the classes of declarative languages participating in the M&S Track and the classes of evaluation techniques adopted (particularly focusing on the System Track), and presents the participants in the Competition; in Section 4 we illustrate the scoring criteria, the benchmark suite and other competition settings; Section 5 reports and discusses the actual results of the Competition; in Section 6 we report details about comparisons of participants with a number of yardstick problem solutions and/or systems; conclusions are eventually drawn in Section 7. An electronic online appendix details the above whenever appropriate.

2 Competition format

In this Section we describe the Competition format for the System and M&S Track, thoroughly discussing motivations and purposes that led to the choices made: the two tracks differ in regulations and design principles. It must be observed that the System Track resembles competitions of neighboring communities in spirit, and is indeed played on a fixed input language, with fixed input problem specifications. However, both tracks introduce specific aspects related to the ASP philosophy. As a main difference, note that competitions close to the ASP community (e.g. SAT, CASC, IPC) are run on a set of couples (i, S), for i an input instance and S a participant solver. Instead, the ASP competition has problem specifications as a further variable. ASP Competitions can be seen as played on a set of triples (i, p, S): here, i is an input instance, p a problem specification, and S a solver. Depending on track regulations, i, p and S are subject to specific constraints.

System Track Format. The regulations of the System Track were conceived taking into account two main guidelines. As a first remark, it must be observed that ASP is still missing a *standard* high-level input language, in contrast with other similar declarative paradigms.[2] It was thus important to play on the grounds of a common language, despite restriction to commonly acknowledged constructs only. As a second guideline, it has been taken into account that the outcome of the System Track should give a fairly objective measure of what one can expect when switching from one system to another, while keeping all other conditions fixed, such as the problem encoding and solver settings. In accordance with the above, the System Track was held on the basis of the following rules.

(1) The Track was open to systems able to parse input written in a fixed language format, called ASP-Core.

[2] These range from the Satisfiability Modulo Theories SMT-LIB format (smt-lib-web 2011), the Planning Domain Definition Language (PDDL) (Gerevini and Long 2005), the TPTP format used in the CASC Automated Theorem Proving System Competitions (CADE-ATP 2011), to the Constraint Handling Rules (CHR) family (CHR 2004).

(2) For each benchmark problem, the organizers chose a fixed ASP-Core specification: each system had to use this specification compulsorily for solving the problem at hand.
(3) Syntactic special-purpose solving techniques, e.g. recognizing a problem from file names, predicates name etc., were forbidden.

The detailed rules and the definition of "syntactic technique" are reported in the online Appendix B.

The language ASP-Core. ASP-Core is a rule-based language its syntax stemming from plain Datalog and Prolog: it is a conservative extension to the nonground case of the *Core* language adopted in the First ASP Competition; it complies with the core language draft specified at LPNMR 2004 (ASP-Draft 2004), and refers to the language specified in the seminal paper (Gelfond and Lifschitz 1991). Its reduced set of constructs is nowadays common for ASP parsers and can be supported by any existing system with very minor implementation effort.[3] ASP-Core features disjunction in the rule heads, both strong and negation-as-failure (NAF) negation in rule bodies, as well as nonground rules. A detailed overview of ASP-Core is reported in the online Appendix A;the full ASP-Core language specification can be found in (Calimeri *et al.* 2011a).

Model and Solve Track Format. The regulations of the M&S Track take into account the experience coming from the previous ASP Competitions. As driving principles in the design of the M&S Track regulations we can list: encouraging the development of new expressive declarative constructs and/or new modeling paradigms; fostering the exchange of ideas between communities in close relationships with ASP; and, stimulating the development of new *ad-hoc* solving methods, and refined problem specifications and heuristics, on a per benchmark domain basis. In the light of the above, the M&S Track was held under the following rules:

(1) the competition organizers made a set of problem specifications public, together with a set of test instances, these latter expressed in a common instance input format;
(2) for each problem, teams were allowed to submit a specific solution bundle, based on a solver (or a combination of solvers) of choice, and a problem encoding;
(3) any submitted solution bundle was required to be mainly based on a declarative specification language.

3 Participants

In this section we present the participants in the competition, categorized by the adopted modeling paradigms and their evaluation techniques.

[3] During competition activities, we also developed a larger language proposal, called ASP-RfC (Request for Comments), including aggregates and other widely used, but not yet standardized, language features.

3.1 System Track

The participants in the System Track were only ASP-Based systems. The traditional approach to ASP program evaluation follows an instance processing work-flow composed of a grounding module, generating a propositional theory, coupled with a subsequent propositional solver module. There have been other attempts deviating from this customary approach (Dal Palù *et al.* 2009; Lefèvre and Nicolas 2009a, 2009b); nonetheless all the participants adopted the canonical "ground & solve" strategy. In order to deal with nonvariable-free programs, all solvers eventually relied on the grounder `Gringo` (Gebser *et al.* 2007). In detail, the System Track had 11 official participants and five noncompeting systems. These can be classified according to the employed evaluation strategy as follows:

Native ASP: featuring custom propositional search techniques that are based on backtracking algorithms tailored for dealing with logic programs. To this class belong: `clasp` (Gebser *et al.* 2009), `claspD` (Drescher *et al.* 2008), `claspfolio` (Gebser *et al.* 2011), `Aclasp` (Aclasp 2011), SMODELS (Simons *et al.* 2002), IDP (Wittocx *et al.* 2008), and the noncompeting `clasp-mt` (Ellguth *et al.* 2009). `clasp` features techniques from the area of boolean constraint solving, and its primary algorithm relies on conflict-driven nogood learning. `claspD` is an extension of `clasp` that is able to solve unrestricted disjunctive logic programs, while `claspfolio` exploits machine-learning techniques in order to choose the best-suited configuration of `clasp` to process the given input program; `Aclasp` (non-participant system) is a variant of clasp employing a different restart strategy; `clasp-mt` is a multi-threaded version of `clasp`. IDP is a finite model generator for extended first-order logic theories. Finally, SMODELS, one of the first robust ASP systems that have been made available to the community, was included in the competition for comparison purposes, given its historical importance.

SAT-Based: employing translation techniques, – e.g. completion (Fages 1994), loop formulas (Lee and Lifschitz 2003; Lin and Zhao 2004), nonclausal constraints (Lierler 2008) – to enforce correspondence between answer sets and satisfying assignments of SAT formulas so that state-of-the-art SAT solvers can be used for computing answer sets. To this class belong: CMODELS (Lierler and Maratea 2004), SUP (Lierler 2008) and three variants of LP2SAT (Janhunen 2006): (LP2GMINISAT, LP2LMINISAT and LP2MINISAT). In detail, CMODELS can handle disjunctive logic programs and exploits a SAT solver as a search engine for enumerating models, and also verifying model minimality whenever needed; SUP makes use of nonclausal constraints, and can be seen as a combination of the computational ideas behind CMODELS and SMODELS; the LP2SAT family of solvers, where the trailing *g* and *l* account for the presence of variants of the basic strategy, employed MINISAT (Eén and Sörensson 2003).

Difference Logic-based: exploiting a translation (Janhunen *et al.* 2009) from ASP propositional programs to Difference Logic (DL) theories (Nieuwenhuis and Oliveras 2005) to perform the computation of answer sets via Satisfiability Modulo Theories (Nieuwenhuis *et al.* 2006) solvers. To this class belongs LP2DIFFZ3 and its three noncompeting variants, namely: LP2DIFFGZ3, LP2DIFFLZ3, and LP2DIFFLGZ3. The

LP2DIFF solver family (Janhunen *et al.* 2009) translates ground ASP programs into the QF_IDL dialect (difference logic over integers) of the SMT library (smt-lib-web 2011); the trailing *g*, *l* and *lg* letters account for different variants of the basic translation technique. The LP2DIFF family had *Z3* (de Moura and Bjørner 2008), as underlying SMT solver.

3.2 M&S Track

The M&S Competition Track was held on an open problem encoding, open language basis. Thus participants adopted several different declarative paradigms, which roughly belong to the following families of languages: *ASP-based*, adopting ASP (Gelfond and Lifschitz 1991) (and variants) as modeling language; *FO(.)-based*, employing FO(ID) (Denecker and Ternovska 2008); *CLP-based*, using logic programming as declarative middle-ware language for reasoning on constraint satisfaction problems (Jaffar and Lassez 1987); and, *Planning-based*, adopting PDDL (Planning Domain Definition Language) as modeling language (PDDL 3.1 2008), respectively. In detail, six teams participated to the M&S Track:

Potassco: The Potassco team from the University of Potsdam, Germany (Gebser *et al.* 2007) submitted a heterogenous ASP-based solution bundle. Depending on the benchmark problem, Potassco employed `Gringo` (Gebser *et al.* 2007) coupled with either `clasp` (Gebser *et al.* 2009) or `claspD` (Drescher *et al.* 2008), and `Clingcon`, which is an answer set solver for constraint logic programs, built upon the `Clingo` system and the CSP solver Gecode (GECODE 2011), embedding and extending `Gringo` for grounding.

`Aclasp`: The team exploited the same ASP-based solutions provided by the Potassco team, and participated only in a number of selected problem domains. The solver of choice was `Aclasp` (Aclasp 2011), a modified version of `clasp` that features a different restart-strategy, depending on the average decision-level on which conflicts occurred. The grounder of choice was `Gringo`.

IDP: The IDP (Wittocx *et al.* 2008) team, from the Knowledge Representation and Reasoning (KRR) research group of K.U. Leuven, Belgium, proposed FO(.)-based solutions. In particular, problem solutions were formulated in the *FO*(.) input language, and the problem instances were solved by MiniSatID (Mariën *et al.* 2008) on top of the the grounder Gidl (Wittocx *et al.* 2008). A preprocessing script was used to rewrite ASP instances into FO(.) structures.

EZCSP: EZCSP is an Eastman Kodak Company and University of Kentucky joint team. The team is interested in evaluating and comparing ASP and hybrid languages on challenging industrial-sized domains. EZCSP (Balduccini 2009c) is also the name of both the CLP-based modeling language, featuring a lightweight integration between ASP and Constraint Programming (CP), and the solver employed by this team. The EZCSP system supports the free combination of different ASP and CP solvers which can be selected as sub-solvers, according to the features of the target domain. In particular, the team exploited the following ASP solvers depending on the benchmark at hand: `clasp`, `iClingo` and ASPM (Balduccini 2009a). Moreover, in cases where CP constraints were used, the team selected B-Prolog as solver.

BPSolver: This team adopted a CLP-based modeling language and exploited the B-Prolog system (Zhou 2011) for implementing solutions. BPSolver employed either pure Prolog, tabling techniques (Chen and Warren 1996), or CLP(FD) (Hentenryck 1989) depending on the problem at hand. In particular, apart from a few problems that required only plain Prolog, all the provided solutions were based on either CLP(FD) or tabling.

Fast Downward: This is an international multiple research institutions joint team that proposed some planning-based solutions. Benchmark domains were statically modeled as planning domains, and problem instances were automatically translated from ASP to PDDL. The submitted solutions were based on Fast Downward (Helmert 2006), a planning system developed in the automated planning community. The team restricted its participation to benchmarks that could be easily seen as planning problems and exploited a number of different configurations/heuristics of Fast Downward.

4 Competition settings

We now briefly describe the scoring methodology of choice, the selected benchmarks and other practical settings in what the competition was run. A detailed description of general settings, the scoring criteria and the selected benchmark suite can be, respectively, found in the online Appendices B, C, and D of the paper.

Scoring System. The scoring framework is a refinement of the one adopted in the first and second ASP Competitions. In these former editions, scoring rules were mainly based on a weighted sum of the number of instances solved within a given time-bound; in this edition, the scoring framework has been extended by awarding additional points to systems performing well in terms of evaluation time. For search and query problems, each system on benchmark problem P was awarded the score $S(P) = S_{solve}(P) + S_{time}(P)$. S_{solve} and S_{time} could range from 0 to 50 each: while S_{solve} is linearly dependent on the number of instances solved in the allotted time, S_{time} contains a logarithmic dependence on participants' running times, thus making less significant time differences in the same order of magnitude. As for optimization problems, the S_{solve} quota was replaced with a scoring formula taking into account also the solution quality, in particular, the closer to the optimal cost, the better exponentially.

Benchmark suite. There were a total of 35 selected benchmark domains, mainly classified according to the computational complexity of the related problem, in *Polynomial*, *NP*, and *Beyond-NP* ones, where this latter category was split into Σ_2^P and *Optimization*. The benchmark suite included planning domains, temporal and spatial scheduling problems, combinatory puzzles, a few graph problems, and a number of applicative domains taken from the database, information extraction and molecular biology field. According to their type, problems were also classified into *Search*, *Query* and *Optimization* ones.

Software and Hardware Settings. The Competition took place on servers featuring a 4-core Intel Xeon CPU X3430 running at 2.4 Ghz, with 4 GiB of physical RAM. All the systems where benchmarked with just one out of four processors enabled, with the exception of the parallel solver `clasp-mt`, and were allowed to use up to 3 GiB of user memory. The allowed execution time for each problem's instance was set at 600 seconds.

5 Results and discussion

The final competition results are reported in Figures 1 and 2, for System and M&S Track, respectively. The detailed results for each considered benchmark problem, and cactus plots detailing the number of instances solved and the corresponding time, on a per participant basis, are reported in the online Appendix F (note that timed out instances are not drawn). Full competition figures, detailed on a per instance basis, together with executable packages and declarative specifications submitted by participants, are available on the competition web site (Calimeri *et al.* 2010).

5.1 SystemTrack results

Polynomial Problems. Grounding modules are mainly assessed while dealing with problems from this category, with the notable exception of two problems, for which, although known to be solvable in polynomial time, we chose their natural declarative encoding, making use of disjunction. In the case of these last two problems, the "combined" ability of grounder and propositional solver modules was tested. The aim was to measure whether, and to what extent, a participant system could be able to converge on a polynomial evaluation strategy when fed with such a natural encoding. All the participant systems employed `Gringo` (v.3.0.3) as the grounding module: however, we noticed some systematic performance differences, owing to the different command line options fed to `Gringo` by participants. The winner of the category is `clasp`, with 213 points, as shown in Figure 1. Interestingly, Figure 5(a) of the online Appendix F shows a sharp difference between a group of easy and hard instances: notably, these latter enforced a bigger memory footprint when evaluated; indeed, it is worth mentioning that the main cause of failure in this category was out of memory, rather than time-out. Instances were in fact relatively large, usually.

NP Problems. The results of this category show how `claspfolio` (609 points) slightly outperformed `clasp` and IDP (597 points), these latter having a slightly better time score (227 versus 224 of `claspfolio`).

Beyond-NP Problems. Only the two systems `claspD` and CMODELS were able to deal with the two problems in this category, with `claspD` solving and gaining points on both problems, and CMODELS behaving well on MINIMALDIAGNOSIS only.

System	**Overall**			**P**			**NP**			**Bnd-NP**		
	Score	**Inst**	**Time**	**Score**	**Inst**	**Time**	**Score**	**Inst**	**Time**	**Score**	**Inst**	**Time**
claspd	**861**	560	301	206	145	61	552	355	197	**103**	60	43
claspfolio	**818**	535	283	209	150	59	**609**	385	224	0	0	0
clasp	**810**	520	290	**213**	150	63	597	370	227	0	0	0
idp	**781**	500	281	184	130	54	597	370	227	0	0	0
aclasp	**780**	510	270	191	135	56	589	375	214	0	0	0
claspmt	**766**	485	281	137	100	37	629	385	244	0	0	0
cmodels	**766**	510	256	184	130	54	510	335	175	72	45	27
lp2diffz3	**572**	405	167	178	135	43	394	270	124	0	0	0
sup	**541**	380	161	195	140	55	346	240	106	0	0	0
lp2sat2gminisat	**495**	365	130	185	140	45	310	225	85	0	0	0
lp2diffgz3	**495**	360	135	178	135	43	317	225	92	0	0	0
lp2sat2minisat	**481**	355	126	179	135	44	302	220	82	0	0	0
lp2sat2lminisat	**472**	350	122	171	130	41	301	220	81	0	0	0
lp2difflz3	**471**	345	126	179	135	44	292	210	82	0	0	0
lp2difflgz3	**470**	345	125	178	135	43	292	210	82	0	0	0
smodels	**449**	295	154	180	130	50	269	165	104	0	0	0

Fig. 1. SystemTrack - Results by categories.

System	**Overall**			**P**			**NP**			**Bnd-NP**			**Opt**		
	Score	**Inst**	**Time**	**Score**	**Inst**	**Time**	**Score**	**Inst**	**Time**	**Score**	**Inst**	**Time**	**Score**	**Inst / Opt**	**Time**
potassco	**2413**	1432	981	**497**	290	207	**1463**	848	615	**110**	60	50	**343**	234	109
bpsolver	**1970***	1111*	859*	459	253	206	1218*	654*	564*	86	43	43	207	161	46
aclasp	**1935**	1140	795	404	240	164	1206	680	526	0	0	0	325	220	105
ezcsp	**1760**	993	767	320	173	147	1406	786	620	0	0	0	34	34	0
idp	**1436**	918	518	175	117	58	1121	680	441	0	0	0	140	121	19
fastdownward	**432***	254*	178*	181	100	81	170*	107*	63*	0	0	0	81	47	34

Fig. 2. M&STrack - Results by categories.

Overall Results. Figure 1 shows `claspD` as the overall winner, with 861 points: 560 points were awarded for the instance score, corresponding to a total of 112 instances solved out of 200. `claspfolio` and `clasp` follow with a respective grand total of 818 and 810. It is worth noting that `claspD` is the only system, together with CMODELS, capable of dealing with the two Beyond-NP problems included in the benchmark suite, this giving to `claspD` a clear advantage in terms of score.

Noncompeting Systems. After the official competition run, we additionally ran five noncompeting systems, including: the parallel system `clasp-mt`, and a number of solvers which did not meet the final official deadline. We included here also those systems in order to give a wider picture of the state-of-the-art in ASP solving.

Noncompeting systems are reported in italics in Figure 1, and their behavior is plotted in Figures 5–8 of the online Appendix F, with the winner of the System Track used as a yardstick. Note that noncompeting executables were mostly variants of the executables presented by the Potassco and the Aalto teams. None of them performed clearly better than the "official" participating versions. The best sequential noncompeting system was `Aclasp`, with a score of 780 points, which would have reached the fifth absolute position in the final classification; see Figure 1.

A special mention goes to the parallel system `clasp-mt`, the only system that ran on a machine with four CPUs enabled. `clasp-mt` is a comparatively young system and (although it was disqualified from some domains[4]) it is clearly the best performer in *NP* totalizing 629 points in this category corresponding to 29 points more than the best sequential system (`claspfolio`).

This result confirms the importance of investing in parallel solving techniques for exploiting the nowadays-diffused parallel hardware.

5.2 M&S Track results

Polynomial Problems. The winner in the category (see Figure 2) is the Potassco team. It is worth mentioning that the runner-up BPsolver, which for this category presented solutions based on predicate tabling, was the absolute winner in three out of the seven problem domains. This suggests that top-down evaluation techniques might pay off, especially for polynomial problems.

NP Problems. The category shows the Potassco team as winner with 1,463 points, closely followed by EZCSP (1,406 points), which was notably the best performing team on ReverseFolding and Packing. Also, BPsolver was by far the fastest system in six domains out of nineteen, although its performance was fluctuating (e.g. in Solitaire and GraphColouring) and it was disqualified in a couple of domains.[5] The very good performance of idp on GraphColouring is also worth mentioning.

Beyond-NP Problems. Only two teams submitted solutions for the two problems in this category, with the Potassco team being the clear winner in both domains.

Optimization Problems. In this category the two clasp-based solution bundles (Potassco and `Aclasp`) outperformed the rest of participants, with idp being the first nonclasp-based system in the category. As in the NP category, BPsolver was the best team in a couple of domains.

Overall Results. Figure 2 shows the Potassco solution bundle as the clear winner of the M&S Track (online Appendix F, Figure F 2). The results detailed per benchmark (again, the online Appendix F, Figure F 4) show that all the teams were best performers in one ore more domains with very encouraging results.

[4] The *overall score* for a problem P is set to zero if the system produces an incorrect answer for some instance P. See the online Appendix C for more details.

[5] For the sake of scientific comparison, Figure F 4 reports also scores obtained by bpsolver on a late submission for HanoiTowers, fixing the faulty solution submitted within the deadline. Grand totals include this latter score.

A special remark must be made concerning the *fastdownward* team, coming from the planning community. It competed in only a few number of domains, mostly corresponding to planning problems, totalizing 433 points. Given the small number of problems which *fastdownward* participated in, a clear comparison cannot be drawn: however, it can be noted that *fastdownward* performed quite well on HydraulicLeaking and HydraulicPlanning. In other domains, the relatively low performance can be explained considering that some of the problems where specified more in the form of knowledge representation problems, with some common restrictions specific to ASP.

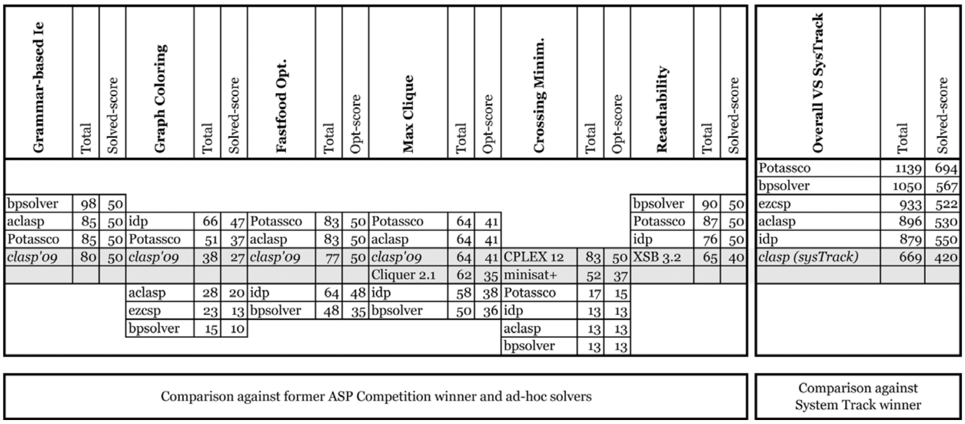
The solutions proposed by all participants were very heterogenous, ranging from purely declarative to the usage of Prolog in a nearly procedural style. Among the lessons learned, it is worth observing the fact that purely declarative solutions very often paid off in terms of efficiency, and outperformed comparatively more tweaked approaches to problem solving.

6 Further analysis

In this section a number of additional analyses are reported, with the aim of giving a clearer and more complete picture of the state of the art in ASP solving and declarative programming. In particular: we compared the winning solution bundles[6] submitted to the former ASP Competition with the updated ones submitted also to the Third ASP Competition, so that two years of advances in the state of the art are outlined; we measured the distance in performance among ASP-based solutions and some specialized *ad-hoc* solutions available in the literature, for some specific benchmark problems considered in the current competition; and, eventually, we assessed the impact of fine-tuning of systems/solutions by comparing specialized executables and problem encodings submitted to the M&S Track with a good-performing default-setting ASP system of the System Track. The outcomes are summarized in Figure 3. A grey strip highlights the lines corresponding to "yardstick" systems which competitors have been compared to. *Total* scores are computed according to the rules of this Competition; *Solved-score*, which roughly corresponds to the score computed according to past Competition rules, is obtained by subtracting the score corresponding to the time quota from the Total score introduced in this competition (see the online Appendix C for more insights). The results are discussed in detail in the following.

The State-of-the-art after Two-years of Improvements. In order to assess possible improvements over former participants in the Second ASP Competition, we selected some significant problems appearing both in the Third and Second edition of the Competition with same specification. This set of problems counts a polynomial problem (Grammar-based Information Extraction), an NP problem (GraphColoring), and two optimization problems (FastFoodOptimization and

[6] As "solution bundle" we mean here the combination of ad-hoc tuned solver binaries together with *ad-hoc* encodings, as they were submitted in the second ASP competion.



Grammar-based Ie	Total	Solved-score	**Graph Coloring**	Total	Solved-score	**Fastfood Opt.**	Total	Opt-score	**Max Clique**	Total	Opt-score	**Crossing Minim.**	Total	Opt-score	**Reachability**	Total	Solved-score
bpsolver	98	50													bpsolver	90	50
aclasp	85	50	idp	66	47	Potassco	83	50	Potassco	64	41				Potassco	87	50
Potassco	85	50	Potassco	51	37	aclasp	83	50	aclasp	64	41				idp	76	50
clasp'09	80	50	*clasp'09*	38	27	*clasp'09*	77	50	*clasp'09*	64	41	CPLEX 12	83	50	XSB 3.2	65	40
									Cliquer 2.1	62	35	minisat+	52	37			
			aclasp	28	20	idp	64	48	idp	58	38	Potassco	17	15			
			ezcsp	23	13	bpsolver	48	35	bpsolver	50	36	idp	13	13			
			bpsolver	15	10							aclasp	13	13			
												bpsolver	13	13			

Comparison against former ASP Competition winner and ad-hoc solvers

Overall VS SysTrack	Total	Solved-score
Potassco	1139	694
bpsolver	1050	567
ezcsp	933	522
aclasp	896	530
idp	879	550
clasp (sysTrack)	669	420

Comparison against System Track winner

Fig. 3. Comparisons with state-of-the-art solutions and other yardsticks.

MaximalClique). On these benchmarks, we ran the solution bundles submitted by the winners of the Second ASP Competition (the Potassco team) alongside all the solution bundles of the current participants to the M&S Track. The instance families used for this test were composed of both the instances used in the Second ASP Competition and in the current edition.

The state of the art in the last two years has been clearly pushed forward, as witnessed by the results reported in the four leftmost sections of Figure 3 (corresponding to the above-mentioned problems). Indeed, the new solution bundles based on `clasp` (indicated by *Potassco* in Figure 3) outperformed in all considered benchmarks the ones (indicated by *clasp'09* in Figure 3) submitted to the Second ASP competition. Note also that, other current solution bundles (i.e. bpsolver, `Aclasp`, IDP) were often able to outperform *clasp'09*, and are generally comparable to *Potassco* even beating it on the Grammar-based Information Extraction and GraphColoring problems.

Participants versus *Ad-hoc* Solutions. Participants in the Competition were based on declarative formalisms, and were essentially conceived as "general-purpose" solvers. For a selection of benchmark domains, we compared participants in the M&S Track with specialized *ad-hoc* solutions, not necessarily based on declarative specifications, with the aim of figuring out what a user might expect to pay in order to enjoy the flexibility of a declarative system.

Maximal Clique. MaximalClique is a graph problem with a long standing history of research towards efficient evaluation algorithms. It was one of the problems investigated in the early Second DIMACS Implementation Challenge (Johnson and Trick 1996), and research continued on the topic later on (Gibbons *et al.* 1996; Bomze *et al.* 1999; Gutin 2004). In the Competition, the problem was specified as finding the maximum cardinality clique on a given graph (Calimeri *et al.* 2010) and most of the instances were taken from BHOSLIB (Xu 2004).

We compared participants in the M&S Track with Cliquer (Niskanen 2003). Cliquer is an up-to-date implementation of an exact branch-and-bound algorithm, which is expected to perform well while dealing with several classes of graphs including sparse, random graphs and graphs with certain combinatorial properties (Östergård 2002).

Cliquer is based on an exact algorithm: in this respect we found it the more natural choice for comparison with participants in the ASP Competition, all of which are based on exhaustive search algorithms. A comparison with other existing approximate algorithms (Boppana and Halldórsson 1992; Feige 2005) would have been expectedly unbalanced in favor of these latter. Our findings show that, in the setting of the competition, *Potassco* and `Aclasp` have comparable performance with Cliquer, while IDP performed quite close to them.

Crossing Minimization in layered graphs. Minimizing crossings in layered graphs is an important problem having relevant impact, e.g. in the context of VLSI layout optimization. The problem has been studied thoroughly, and valuable algorithms have been proposed for solving it, among which (Jünger *et al.* 1997; Healy and Kuusik 1999; Mutzel 2000) and (Gange *et al.* 2010). The instances considered for the Competition were taken from the graphviz repository (Graphviz 2011).

We ran the two *ad-hoc* solutions proposed in (Gange *et al.* 2010) over the M&S Track instance family and compared results with outcomes of participants in the M&S competition. The two solutions were, respectively, based on a translation to SAT and to Mixed Integer Programming. The former was run using MiniSat+ (Eén and Sörensson 2006) as solver, while the latter used CPlex 12.0 (CPLEX 2011). Both solvers were run using default settings.

The comparison shows a big gap between participants in the competition and the two yardsticks, which both perform much better.

Reachability. This polynomial problem is a distinctive representative of problems that can be naturally specified using recursion in plain logic programming, lending itself to comparison with other logic programming-based systems.

We compared the outcomes of participants in the M&S Track with XSB 3.2 (XSB 2011), one of the reference systems of the OpenRuleBench initiative. OpenRuleBench (Fodor *et al.* 2011) is aimed at comparing rule-based systems both from the Deductive Database area, Prolog-based and oriented to RDF triples.

All the three participants submitting a solution to the Reachability problem outperformed XSB, especially in terms of time performance. However, it must be stated that the scoring system of the ASP Competition purposely does not exclude loading and indexing steps when measuring execution times, differently from the setting of OpenRuleBench; furthermore, XSB was run with its *off-the-shelf* configuration, except for attribute indexing and tabling appropriately enabled. Indeed, BPsolver depends here on the same technology as XSB (tabling and top down), but fine-tuned to the problem.

The Effects of Fine-tuning. Recall that the setting of the System Track prevented all participants from developing domain-dependent solutions and, on the contrary, the M&S Track allowed the submission of fine-tuned encodings and the static selection of systems parameters/heuristics.

In order to assess the impact of fine-tuning, we selected the `clasp` version as yardstick that participated in the System Track, labeled *clasp (sysTrack)* in Figure 3; then we ran it over P and NP problems which were in common between the System

Track and the M&S Track problem suites.[7] *clasp (sysTrack)* was run using the fixed ASP-Core encodings and the settings of the System Track, but over the different (and larger) instance sets coming from the M&S Track. These choices ensured a comparison more targeted to the assessment of the impact of tuning: indeed, (1) the *clasp (sysTrack)* executable is the "naked"[8] solver of choice in almost all the solution bundles submitted by the Potassco team, the winner of the M&S Track; (2) *clasp (sysTrack)* is the winner of the System Track in the *P* category, and runner-up in the *NP* category (in both it performed better than the overall winner `claspD`); (3) there is no significant difference between the solutions presented by the Potassco team in both the System and the M&S Track for *BeyondNP* problems.

The obtained results are reported in Figure 3, rightmost side, and, as expected, confirm the importance of fine-tuning and customized encodings. Indeed, "tuned" solution bundles outperformed the fixed configuration of *clasp (sysTrack)*. This clearly indicates the need for further developing new optimization techniques and self-tuning methods (e.g. on the line of Gebser *et al.* 2011), and to further extend the basic standard language, in order to make efficient ASP solutions within reach of users that are not expert in the system's internals and/or specific features.

7 Concluding remarks

Much effort has been spent in the last 20 years by the ASP community, and outstanding results have been achieved since the first seminal papers; ASP and ASP system can be nowadays profitably exploited in many application settings, not only thanks to the declarative and expressive power of the formalism, but also thanks to continuously improving performances. Even a two-year horizon indicates that things are getting better and better. In addition, it is interesting to note that, despite the difference between the specifically tailored solutions and the one with factory settings being significant, the current state-of-the-art ASP implementations are offering a good experience to application developers, given the nice declarative approach of the formalism and the mature, robust, and currently well-performing available systems.

Nevertheless, there is still much room for improvements. For instance, the comparison with some *ad-hoc* solvers confirmed that the performance is not a tout-court weak point anymore, but the gap with respect to some others suggests that they might be further improved. The main issue, however, still remains the lack of a sufficiently broad and standard language.

Acknowledgements

The authors feel honored by the invitation of the Università della Calabria as host institution on the part of the ASP Competition Steering Committee. The authors

[7] The HydraulicPlanning benchmark has been excluded since its specification was different in the M&S Track.
[8] In the sense that it does not feature any parameter tuning technique.

thank all the members of the Computer Science Group at the Dipartimento di Matematica of Università della Calabria for their invaluable collaboration, which made this event possible, and especially: Mario Alviano, Onofrio Febbraro, Maria Carmela Santoro and Marco Sirianni. Authors thank Nicola Leone, the Director of the Dipartimento di Matematica of Università della Calabria, who provided them with all the means, in the form of human and technical resources, and animated earlier discussions carried out together. Special thanks go to all the members of the ASP and CLP communities which authored problem domains, and to participating teams, whose continuous feedback significantly helped in improving competition rules and benchmark specifications. The authors thank Graeme Gange and Peter Stuckey for providing *ad-hoc* solutions for the CrossingMinimization problem originally benchmarked in (Gange *et al.* 2010). A special mention goes to Jim Delgrande and Wolfgang Faber, for their support as LPNMR-11 conference chairs and editors of the proceedings featuring the preliminary report of the Competition. The authors wish also to thank the anonymous reviewers for their fruitful comments and suggestions, that significantly helped to improve the work.

References

Aclasp 2011. Aclasp Sources. `https://potassco.svn.sourceforge.net/\breaksvnroot/potassco/branches/adaptive-restarts/adapt-to-avgdl`.

Anger, C., Gebser, M., Linke, T., Neumann, A. and Schaub, T. 2005. The nomore++ approach to answer set solving. In *LPAR 2005*. LNCS 3835, 95–109.

ASP-Draft 2004. Core language for ASP solver competitions. Steering committee meeting at LPNMR 2004. `https://www.mat.unical.it/aspcomp2011/files/Corelang2004.pdf`.

Balduccini, M. 2009a. A general method to solve complex problems by combining multiple answer set programs. In *ASPOCP Workshop at ICLP 2009)*.

Balduccini, M. 2009c. Representing constraint satisfaction problems in answer set programming. In *ASPOCP Workshop at ICLP 2009*.

Balduccini, M., Gelfond, M., Watson, R. and Nogeira, M. 2001. The USA-Advisor: A case study in answer set planning. In *LPNMR 2001*. LNCS 2173, 439–442.

Bancilhon, F., Maier, D., Sagiv, Y. and Ullman, J. D. 1986. Magic sets and other strange ways to implement logic programs. In *PODS 1986*. 1–15.

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

Baral, C. and Gelfond, M. 2000. Reasoning agents in dynamic domains. *Logic-Based Artificial Intelligence*, 257–279.

Baral, C. and Uyan, C. 2001. Declarative specification and solution of combinatorial auctions using logic programming. In *LPNMR 2001*. LNAI 2173, 186–199.

Bardadym, V. A. 1996. Computer-aided school and university Timetabling: The new wave. In *Practice and Theory of Automated Timetabling*. LNCS, vol. 1153. 22–45.

Bell, C., Nerode, A., Ng, R. T. and Subrahmanian, V. 1994. Mixed integer programming methods for computing nonmonotonic deductive databases. *JACM 41*, 1178–1215.

Ben-Eliyahu-Zohary, R. and Palopoli, L. 1997. Reasoning with minimal models: Efficient algorithms and applications. *AI 96*, 421–449.

Bomze, I. M., Budinich, M., Pardalos, P. M. and Pelillo, M. 1999. The maximum clique problem. In *Handbook of Combinatorial Optimization*. Kluwer Academic Publishers, Boston, MA, 1–74.

Boppana, R. and Halldórsson, M. M. 1992. Approximating maximum independent sets by excluding subgraphs. *BIT 32*, 180–196.

CADE-ATP 2011. The CADE ATP System Competition. http://www.cs.miami.edu/~tptp/CASC.

Cadoli, M., Eiter, T. and Gottlob, G. 1997. Default logic as a query language. *IEEE TKDE 9,* 3, 448–463.

Calimeri, F., Ianni, G., Ricca, F. et al. 2010. The third answer set programming competition homepage. http://www.mat.unical.it/aspcomp2011/.

Calimeri, F., Ianni, G. and Ricca, F. 2011. 3rd ASP Competition, file and language formats. http://www.mat.unical.it/aspcomp2011/files/LanguageSpecifications.pdf.

Calimeri, F., Ianni, G., Ricca, F. et al. 2011. The third answer set programming competition: Preliminary report of the system competition track. In *LPNMR 2011*. LNCS 6645, 388–403.

Chen, W. and Warren, D. S. 1996. Tabled evaluation with delaying for general logic programs. *Journal of The ACM 43*, 20–74.

CHR 2004. Constraint handling rules. Since 2004. http://dtai.cs.kuleuven.be/CHR/.

CPLEX 2011. IBM ILOG CPLEX.
http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

Dal Palù, A., Dovier, A., Pontelli, E. and Rossi, G. 2009. GASP: Answer set programming with lazy grounding. *FI 96,* 3, 297–322.

Dantsin, E., Eiter, T., Gottlob, G. and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys 33,* 3, 374–425.

Dao-Tran, M., Eiter, T., Fink, M. and Krennwallner, T. 2010. Distributed nonmonotonic multi-context systems. In *KR 2010*. AAAI Press, 60–70.

de Moura, L. M. and Bjørner, N. 2008. Z3: An efficient SMT solver. In *TACAS*. LNCS 4963, 337–340.

Denecker, M. and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM TOCL 9*, 14:1–14:52.

Denecker, M., Vennekens, J., Bond, S., Gebser, M. and Truszczynski, M. 2009. The second answer set programming competition. In *LPNMR 2009*. LNCS 5753, 637–654.

Dovier, A. 2011. Recent constraint/logic programming based advances in the solution of the protein folding problem. *Intelligenza Artificiale 5,* 1, 113–117.

Drescher, C., Gebser, M., Grote, T., Kaufmann, B., König, A., Ostrowski, M. and Schaub, T. 2008. Conflict-driven disjunctive answer set solving. In *KR 2008*. AAAI Press, 422–432.

Eén, N. and Sörensson, N. 2003. An extensible SAT-solver. In *SAT*. LNCS 2919, 502–518.

Eén, N. and Sörensson, N. 2006. Translating Pseudo-Boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation 2*, 1–26.

Eiter, T., Ianni, G. and Krennwallner, T. 2009. Answer set programming: A primer. In *Reasoning Web*. LNCS 5689, 40–110.

Ellguth, E., Gebser, M., Gusowski, M., Kaminski, R., Kaufmann, B., Liske, S., Schaub, T., Schneidenbach, L. and Schnor, B. 2009. A simple distributed conflict-driven answer set solver. In *LPNMR 2009*. LNAI 5753, 490–495.

Faber, W., Leone, N. and Pfeifer, G. 2004. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *JELIA 2004*. LNAI 3229, 200–212.

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

Falkner, A., Haselböck, A. and Schenner, G. 2010. Modeling technical product configuration problems. In *ECAI 2010 Workshop on Configuration*. 40–46.

Feige, U. 2005. Approximating maximum clique by removing subgraphs. *SIAM Journal on Discrete Mathematics 18*, 219–225.

Fodor, P., Liang, S. and Kifer, M. 2011. OpenRuleBench: Report 2011. *Computing*, 1–14.

Franconi, E., Palma, A. L., Leone, N. et al. 2001. Census data repair: A challenging application of disjunctive logic programming. In *LPAR 2001*. LNCS 2250, 561–578.

Friedrich, G. and Ivanchenko, V. 2008. *Diagnosis from First Principles for Workflow Executions*. Technical rep. ISBI research group, Alpen-Adria-Universität Klagenfurt. `http://proserver3-iwas.uni-klu.ac.at/download\_area/Technical-Reports/technical\_report\_2008\_02.pdf`.

Gange, G., Stuckey, P. J. and Marriott, K. 2010. Optimal k-level planarization and crossing minimization. In *Graph Drawing*. LNCS, vol. 6502. 238–249.

Garcia-Molina, H., Ullman, J. D. and Widom, J. 2000. *Database System Implementation*. Prentice Hall, Upper Saddle River, New Jersey.

Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., Schneider, M. T. and Ziller, S. 2011. A portfolio solver for answer set programming: Preliminary report. In *LPNMR 2011*. LNCS 6645, 352–357.

Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T. and Schneider, M. since 2007. Potassco, the Potsdam answer set solving collection - homepage. `http://potassco.sourceforge.net/`.

Gebser, M., Kaufmann, B., Neumann, A. and Schaub, T. 2007. Conflict-driven answer set solving. In *IJCAI 2007*, 386–392.

Gebser, M., Kaufmann, B. and Schaub, T. 2009. The conflict-driven answer set solver *clasp*: Progress report. In *LPNMR 2009*. LNCS 5753, 509–514.

Gebser, M., Liu, L., Namasivayam, G., Neumann, A., Schaub, T. and Truszczyński, M. 2007. The first answer set programming system competition. In *LPNMR 2007*. LNCS 4483, 3–17.

Gebser, M., Schaub, T. and Thiele, S. GrinGo : A new grounder for answer set programming. In *LPNMR 2007*. LNCS 4483, 266–271.

Gebser, M., Schaub, T., Thiele, S. and Veber, P. 2011. Detecting inconsistencies in large biological networks with answer set programming. *TPLP 11*, 323–360.

GECODE 2011. GECODE - An open, free, efficient constraint solving toolkit - Homepage. `http://www.gecode.org/`.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *NGC 9*, 365–385.

Gerevini, A. and Long, D. 2005. *Plan Constraints and Preferences in PDDL3 - The Language of the Fifth International Planning Competition*. Technical rep. `http://cs-www.cs.yale.edu/homes/dvm/papers/pddl-ipc5.pdf`.

Gibbons, L. E., Hearn, D. W., Pardalos, P. M. and Ramana, M. V. 1996. Continuous characterizations of the maximum clique problem. *Mathematics of Operations Research 22*, 754–768.

Graphviz 2011. Graphviz - Graph Visualization Software. `http://www.graphviz.org/`.

Grasso, G., Iiritano, S., Leone, N. and Ricca, F. 2009. Some DLV applications for knowledge management. In *LPNMR 2009*. LNCS 5753, 591–597.

Grasso, G., Leone, N., Manna, M. and Ricca, F. 2010. ASP at Work: Spin-off and Applications of the DLV System. *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning,* Lecture Notes in AI (LNAI), vol. 6565. Springer-Verlag, Berlin.

Gusfield, D. and Irving, R. W. 1989. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA.

Gutin, G. 2004. Independent sets and cliques. *Handbook of Graph Theory*, Boca Raton, Florida, 389.

Healy, P. and Kuusik, A. 1999. The vertex-exchange graph: A new concept for multi-level crossing minimisation. In *Graph Drawing*. LNCS 1731, 205–216.

Helmert, M. 2006. The fast downward planning system. *JAIR 26*, 191–246.

Hentenryck, P. V. 1989. *Constraint Satisfaction in Logic Programming*, Logic Programming series I–XXVI, MIT Press, Cambridge, MA, 224 pp.

Jaffar, J. and Lassez, J. 1987. Constraint logic programming. In *POPL 1987*. 111–119.

Janhunen, T. 2006. Some (in)translatability results for normal logic programs and propositional theories. *Journal of Applied Non-Classical Logics 16,* 1–2, 35–86.

Janhunen, T. and Niemelä, I. 2004. Gnt - A solver for disjunctive logic programs. In *LPNMR 2004*. LNAI 2923, 331–335.

Janhunen, T., Niemelä, I. and Sevalnev, M. 2009. Computing stable models via reductions to difference logic. In *LPNMR 2009*. LNCS 5753, 142–154.

Johnson, D. and Trick, M. 1996. *Cliques, coloring, and satisfiability: Second DIMACS implementation challenge, 11-13, 1993*. DIMACS series in discrete mathematics and theoretical computer science. American Mathematical Society, Boston, MA.

Jünger, M., Lee, E. K., Mutzel, P. and Odenthal, T. 1997. A polyhedral approach to the multi-layer crossing minimization problem. In *International Symposium on Graph Drawing*. LNCS 1353, 13–24.

Lee, J. and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In *ICLP 2003*. 451–465.

Lefèvre, C. and Nicolas, P. 2009a. A first order forward chaining approach for answer set computing. In *LPNMR 2009*. LNCS 5753, 196–208.

Lefèvre, C. and Nicolas, P. 2009b. The first version of a new asp solver: Asperix. In *LPNMR 2009*. LNCS 5753, 522–527.

Leone, N., Greco, G., Ianni, G., Lio, V., Terracina, G., Eiter, T., Faber, W., Fink, M., Gottlob, G., Rosati, R., Lembo, D., Lenzerini, M., Ruzzi, M., Kalka, E., Nowicki, B. and Staniszkis, W. 2005. The INFOMIX system for advanced integration of incomplete and inconsistent data. In *SIGMOD 2005*. 915–917.

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

Lierler, Y. 2008. Abstract answer set solvers. In *ICLP 2008*. LNCS 5366, 377–391.

Lierler, Y. and Maratea, M. 2004. Cmodels-2: SAT-based answer set solver enhanced to non-tight programs. In *LPNMR 2004*. LNAI 2923, 346–350.

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

Marek, V. W. and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm – A 25-Year Perspective*, K. R. Apt, V. W. Marek, M. Truszczyński and D. S. Warren, Eds. Springer Verlag, 375–398.

Mariën, M., Wittocx, J., Denecker, M. and Bruynooghe, M. 2008. SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In *SAT 2008*. LNCS 4996, 211–224.

Marileo, M. C. and Bertossi, L. E. 2010. The consistency extractor system: Answer set programs for consistent query answering in databases. *DKE 69,* 6, 545–572.

Mutzel, P. 2000. An alternative method to crossing minimization on hierarchical graphs. *SIAM Journal on Optimization 11*, 1065–1080.

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

Nieuwenhuis, R. and Oliveras, A. 2005. DPLL(T) with exhaustive theory propagation and its application to difference logic. In *CAV 2005*. LNCS 3576, 321–334.

Nieuwenhuis, R., Oliveras, A. and Tinelli, C. 2006. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of The ACM 53*, 937–977.

Niskanen, S. since 2003. Cliquer Homepage. `http://users.tkk.fi/pat/cliquer.html`.

Nogueira, M., Balduccini, M., Gelfond, M., Watson, R. and Barry, M. 2001 An A-Prolog decision support system for the space shuttle. In *PADL 2001*. LNCS 1990, 169–183.

Östergård, P. R. J. 2002. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics 120,* 1–3, 197–207.

Palopoli, L., Rombo, S. E. and Terracina, G. 2005. Flexible pattern discovery with (extended) disjunctive logic programming. In *ISMIS 2005*. 504–513.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley, Boston, MA.

PDDL 3.1 2008. Changes in PDDL 3.1. `http://ipc.informatik.uni-freiburg.de/PddlExtension`.

Ricca, F., Dimasi, A., Grasso, G., Ielpa, S. M., Iiritano, S., Manna, M. and Leone, N. 2010. A logic-based system for e-Tourism. *FI 105,* 1–2, 35–55.

Simons, P., Niemelä, I. and Soininen, T. 2002. Extending and implementing the stable model semantics. *AI 138*, 181–234.

smt-lib-web. 2011. The satisfiability modulo theories library. `http://www.smtlib.org/`.

Subrahmanian, V., Nau, D. and Vago, C. 1995. WFS + Branch and Bound = Stable Models. *IEEE TKDE 7,* 3 (June), 362–377.

WASP. Wasp showcase. Since 2003. `http://www.kr.tuwien.ac.at/research/projects/WASP/showcase.html`.

Wittocx, J., Mariën, M. and Denecker, M. 2008. GidL: A grounder for FO+. In *NMR 2008.*

Wittocx, J., Mariën, M. and Denecker, M. 2008. The idp system: A model expansion system for an extension of classical logic. In *Logic and Search (LaSh 2008)*. 153–165.

XSB. 2011. The Home of XSB. `http://xsb.sourceforge.net/`.

Xu, K. since 2004. BHOSLIB: Benchmarks with hidden optimum solutions for graph problems. `http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm`.

Zhou, N.-F. 2011. The language features and architecture of B-Prolog. arXiv:1103.0812v1. Submitted to *TPLP Special Issue on Prolog Systems.*