ME-ASP: A Multi-Engine Solver for Answer Set Programming

Marco Maratea[1], Luca Pulina[2] and Francesco Ricca[3]

[1] DIBRIS, University of Genova
[2] POLCOMING, University of Sassari
[3] Dep. of Matematichs, University of Calabria

CILC2012: Rome, Italy, June 6-7th 2012

Motivation

- The number of real-life ASP applications is significantly increasing; thus, efficient ASP systems are needed
- It is well-known that, on empirically hard problems, there is rarely a “global” best algorithm
- Instead, different algorithms perform well on different problem domains/instances (Rice, 1976)
- This fact can be taken as an advantage, by exploiting machine learning techniques

Context

Machine learning techniques, for solving empirically “hard” (ASP) problems, range over:

- multi-engine approach
  chooses among its engines/solvers the one which is more likely to yield optimal results (completely offline)
- portfolio approach
  (multi-engine $+$) allows for changing online the engine employed (or, to change engine’s configuration)
- algorithm configuration and parameter tuning
  finds parameter settings (or *configurations*) of an engine for which the empirical performance on a given set of problem instances is “optimized”
- (Balduccini, 2011)
  selects offline a heuristic ordering to be used in an engine when solving other programs from the same domain

Some of these approaches proved efficient for solving SAT, QBF and MIP problems.

Useful components

In these contexts, some ingredients are always considered:

- a set of “features” that represent several aspects of a problem
- one or more engines on top of which building the approach
- a training set on which learning the decision policy (with a classifier)
- a test set on which the approach is evaluated

ASP seems particularly suited for a multi-engine approach:

- many mature ASP systems, featuring different techniques
- many representative ASP domains and instances (thanks to competitions), on which train and test the approach

Our contribution

A multi-engine approach for Answer Set Programming (ME-ASP), obtained with:

1. design of (syntactic) features that are both significant for classifying the ASP instances and cheap-to-compute
2. selection of ASP solvers that are representative of the state of the art
3. design of training sets (based on uniquely solved instances), and choice of “meaningful” test sets

ASP: Syntax and Semantic

- **ASP**:
  - purely declarative programming paradigm
  - **idea:** write a logic program s.t. its answer sets represent solutions
  - captures all problems at the second level of the polynomial hierarchy
  - exploited in AI and real-world applications

- An ASP program Π is made of rules:

  $$a_1 \vee \ldots \vee a_n \text{ :- } b_1, \ldots, b_k, \text{not } b_{k+1}, \ldots, \text{not } b_m$$

- Answer Set Semantics:
  - consider the ground instantiation $P = ground(\pi)$
  - find a minimal model of the Gelfond-Lifschitz reduct of P

Domains and instances considered

All *NP* and *Beyond NP* instances of the 3rd ASP Competition

Problem	**Class**	**#Instances**
DisjunctiveScheduling	*NP*	10
GraphColouring	*NP*	60
HanoiTower	*NP*	59
KnightTour	*NP*	10
MazeGeneration	*NP*	50
Labyrinth	*NP*	261
MultiContextSystemQuerying	*NP*	73
Numberlink	*NP*	150
PackingProblem	*NP*	50
SokobanDecision	*NP*	50
Solitaire	*NP*	25
WeightAssignmentTree	*NP*	62
MinimalDiagnosis	*Beyond NP*	551
StrategicCompanies	*Beyond NP*	51
Total		1462

grounded with GRINGO (can ground 1425 out of 1462 in less than 600s).

Features selection in ME-ASP (I)

We started by considering a very wide set of features:

- problem size: e.g. number of rules and atoms, ...
- balance: e.g. fraction of unary, binary, ternary rules; ratio of positive and negative atoms in rules, ...
- proximity to horn: fraction of horn rules; occurrences of atoms in horn rules, ...
- ASP peculiar: e.g. number of true facts, disjunctive facts; fraction of normal rule, of constraints; head sizes; occurrences of atoms in heads, loops ...

For features implying distributions, we considered the five numbers (min, 25% percentile, median, 75% percentile, max).

Features selection in ME-ASP (II)

Further considerations:

- the time for computing features is integral part of the ME-ASP solving time
- it is thus important that features are computed fast
- an alternative program for computing features (CLASPRE)
  - computes quite costly features (e.g. related to loops)
  - can compute in 600s all its 74 features of 573 out of 850 *NP* ground instances

Our design, implemented with a JAVA program

- relies on 52 cheap-to-compute features
- can compute all features of 849 *NP* ground instances
- can compute all features of 1371 out of 1425 *NP+Beyond NP* ground instances having distribution of the
  - programs size: 14KB, 1.8MB, 6.1MB, 19MB, 8.1GB
  - CPU times: 0.24s, 1.74s, 2.40s, 4.37s, 541.92s

Solvers selection in ME-ASP

Solvers considered

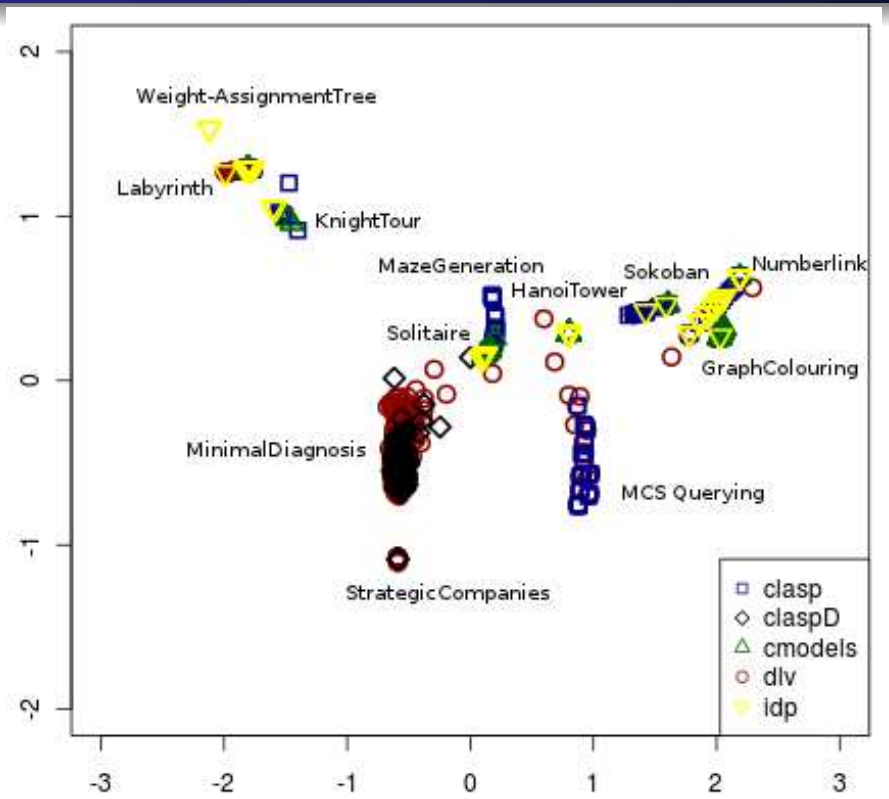
All solvers that entered the 3rd ASP Competition, and DLV.

Solver	Solved	Unique	Solver	Solved	Unique
CLASP	445	26	LP2DIFFZ3	307	–
CMODELS	333	6	LP2SAT2GMINISAT	328	–
DLV	241	37	LP2SAT2LGMINISAT	322	–
IDP	419	15	LP2SAT2LMINISAT	324	–
LP2DIFFGZ3	254	–	LP2SAT2MINISAT	336	–
LP2DIFFLGZ3	242	–	SMODELS	134	–
LP2DIFFLZ3	248	–	SUP	311	1

Solvers considered for the *Beyond NP* class

All solvers that can solve such problems: CMODELS, DLV, CLASPD.

Dataset distribution and training sets



Two-dimensional space projection of the whole dataset.

Our model is based on, and training sets are composed of, uniquely solved instances:

- TS1: 320, without taking into account the instances evaluated in the competition
- TS2: 77, considering only 5 (randomly-chosen) domains

Classification methods employed in ME-ASP

Classifier: matches features with engines. Applied to

- a training set to learn an algorithm selection strategy
- a test set to implement the learned strategy

1. **Aggregation Pheromone density based pattern Classification** (APC): based on the ants colony behavior and distributed adaptive organization in nature
2. **Decision rules** (FURIA): based on a set of "if-then-elsif" constructs to induce decision rules
3. **Decision trees** (J48): based on a tree structure to induce decision trees
4. **Nearest-neighbor** (NN): based on the idea of yielding the label of the training instance which is closer to the given test instance
5. **Support Vector Machine** (SVM): based on an optimal linear hyperplane such that the expected classification error for (unseen) test patterns is minimized

ME-ASP architecture

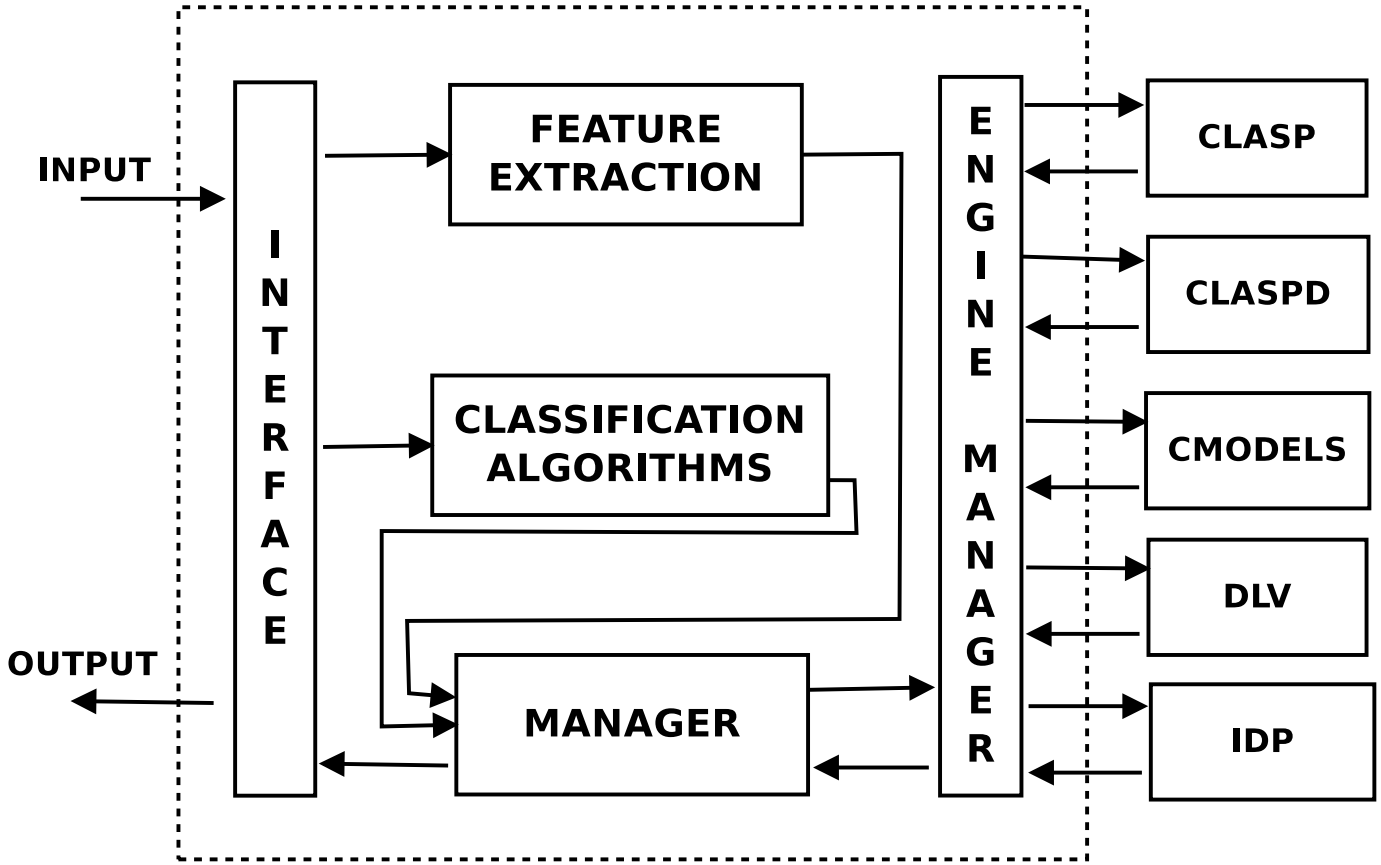


Figure: The architecture of ME-ASP. The dotted box denotes the whole system and, inside it, each solid box represents its modules. Arrows denote functional connections between modules.

Experimental set up

Hardware

- cluster comprised of 50 Intel Xeon E5420 blades equipped with 64 bit GNU Scientific Linux 5.5
- 2GB of memory limit
- 600s of CPU time limit

Training and test sets

- TS1 and TS2
- ground instances submitted (1371) or evaluated (88) at the 3rd ASP competition

Solvers considered

- ME-ASP(X), where X is a classification method
- ME-ASP engines, as submitted to the competition
- CLASPFOLIO, chooses the best CLASP configuration
- SOTA, the ideal solver that always chooses the best engine

Experiment on efficiency I

Solver	*NP*		*Beyond NP*	
	#Solved	Time	#Solved	Time
CLASP	60	5132.45	–	–
CLASPD	–	–	13	2344.00
CMODELS	56	5092.43	9	2079.79
DLV	37	1682.76	15	1359.71
IDP	61	5010.79	–	–
ME-ASP (APC)	63	5531.68	15	3286.28
ME-ASP (FURIA)	63	5244.73	15	3187.73
ME-ASP (J48)	68	5873.25	15	3187.73
ME-ASP (NN)	66	4854.78	15	3187.31
ME-ASP (SVM)	60	4830.70	15	2308.60
CLASPFOLIO	62	4824.06	–	–
CLASPFOLIO(TS1)	59	3709.69	–	–
SOTA	71	5403.54	15	1221.01

Table: Training set: TS1. Test set: grounded instances evaluated at the 3rd ASP competition.

Experiment on efficiency II

Solver	*NP*		*Beyond NP*	
	#Solved	Time	#Solved	Time
CLASP	445	47096.14	–	–
CLASPD	–	–	433	52029.74
CMODELS	333	40357.30	270	38654.29
DLV	241	21678.46	364	9150.47
IDP	419	37582.47	–	–
ME-ASP (APC)	497	55334.15	516	60537.67
ME-ASP (FURIA)	480	48563.26	518	60009.23
ME-ASP (J48)	490	49564.19	510	59922.86
ME-ASP (NN)	490	46780.31	518	55043.39
ME-ASP (SVM)	445	40917.70	518	52553.84
CLASPFOLIO	431	41874.53	–	–
CLASPFOLIO(TS1)	441	41130.54	–	–
SOTA	516	39857.76	520	24300.82

Table: Training set: TS1. Test set: grounded instances submitted to the 3rd ASP competition.

Experiment on robustness

Solver	*NP*		*Beyond NP*	
	#Solved	Time	#Solved	Time
CLASP	445	47096.14	–	–
CLASPD	–	–	433	52029.74
CMODELS	333	40357.30	270	38654.29
DLV	241	21678.46	364	9150.47
IDP	419	37582.47	–	–
ME-ASP (APC)	491	53875.41	444	57555.34
ME-ASP (FURIA)	450	50495.50	365	10483.81
ME-ASP (J48)	450	53272.70	366	10486.43
ME-ASP (NN)	484	52191.49	364	10550.01
ME-ASP (SVM)	383	36786.04	364	10543.00
CLASPFOLIO	431	41874.53	–	–
SOTA	516	39857.76	520	24300.82

Table: Training set: TS2. Test set: grounded instances submitted to the 3rd ASP competition.

Further results and considerations

Solver	#Solved	
	EVAL	SUBM
CLASPD	65	835
ME-ASP (APC)	78	1013
ME-ASP (J48)	83	1000
SOTA	86	1036

Table: Training set: TS1.

Last experiment has a very hard setting, considering e.g. CLASPFOLIO paper (Gebser et al., 2011) uses a training set of 3096 instances and a test (sub)set of 377 instances.

Some hints on classifiers performance

Classifier	Accuracy	
	MOD1	MOD2
APC	96.58%	89.83%
FURIA	94.09%	83.39%
J48	93.12%	79.46%
NN	92.81%	80.71%
SVM	94.38%	82.32%

Table: Accuracy of the trained models of ME-ASP.

Conclusion

We have developed the first multi-engine ASP solver, by

- designing a set of cheap-to-compute syntactic features
- choosing a set of state-of-the-art ASP solvers
- designing test sets based on uniquely solved instances on which instances are trained
- classifying (ground) instances from the 3rd ASP Competition to learn selection strategies

Our analysis shows that, overall, ME-ASP solves (many) more problems than both its engines and CLASPFOLIO.

Other/more details at:
`https://www.mat.unical.it/~ricca/me-asp/`

Future work

We plan to

- evaluate DORS (Balduccini, 2011)
- provide more insights on the “model learned”, e.g. classification rules and decision trees
- design a “self-adaptive” extension of ME-ASP with the ability to adapt its model when it fails to give good predictions

Some references

(Rice, 1976) J.R. Rice. The algorithm selection problem. Advances in Computers. 1976

(Balduccini, 2011) M. Balduccini. Learning and using domain-specific heuristics in ASP solvers. AI Communications 24(2):147-164, 2011

(Gebser et. al, 2011) A Portfolio Solver for Answer Set Programming: Preliminary Report. LPNMR 2011: 352-357, 2011

List of features in ME-ASP

Number of:

- Rules, Atoms, A/R, $(A/R)^2$, $(A/R)^3$, R/A, $(R/A)^2$, $(R/A)^3$
- true, disjunctive, normal and horn facts
- unary, binary, ternary, horn, disjunctive, normal, constraint rules
- “number is the last two groups, normalized by R, to the second and the third”

Solvers call with ME-ASP(J48) on the first experiment

Solver	#Calls
CLASP	35
CLASPD	10
CMODELS	8
DLV	10
IDP	25