The Multi-engine ASP Solver ME-ASP: Progress Report

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

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

NMR2014: Vienna, Austria, July 17-19th 2014

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 (ME-ASP)

  chooses among its engines/solvers the one which is more likely to yield optimal results

- portfolio approach (CLASPFOLIO ver. 1)

  (multi-engine $+$) allows for changing online the engine employed (or, to change engine's configuration)

- algorithm configuration/scheduling, parameter tuning (ASPeeD)

  finds parameter settings (or, *configurations*) of an engine for which the empirical performance on a given set of problem instances is "optimized", and/or computes an ordering on the engines to be run (*schedule*)

- (Balduccini, 2011) (DORS)

  selects offline a heuristic ordering to be used in an engine when solving other programs from the same domain

Useful components

In these contexts, some ingredients are often 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
- a test set on which the approach is evaluated

ASP seems to be good venue for applying a multi-engine approach:

- many ASP systems, featuring different techniques
- many ASP domains and instances (thanks to competitions), on which training and testing the approach

ME-ASP: Review (I)

A multi-engine approach for Answer Set Programming (ME-ASP) that:

1. extracts syntactic cheap-to-compute features on a *training set* of instances
2. selects a pool of ASP solvers that are representative of the state of the art
3. learns a decision policy with a classifier, based on the features computed and the performances of the selected solvers on the training instances
4. applies the policy to instances in a *test set*

ME-ASP: Review (II)

ME-ASP (MPR, 2014a) shows good performance in a setting in which

- training and test sets are composed of ground programs taken from the *NP* and *Beyond NP* domains of the 3rd ASP Competition
- off-line features are computed, e.g. number of rules and atoms, ratio of horn rules and constraints
- CLASP, IDP, DLV, CMODELS are selected as ASP solvers
- the policy is learned with classification algorithms, e.g. decision rules, decision trees, nearest-neighbor

Enhancement to the basic, off-line approach: (MPR, 2014b)

- adaptation of the policy when it fails to give good prediction

ASP: Syntax and Semantics

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

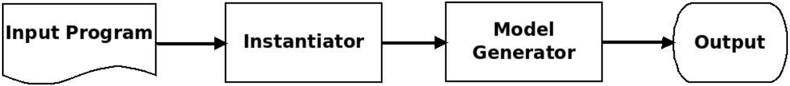
- An ASP program Π is made of rules:

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

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

Current contribution

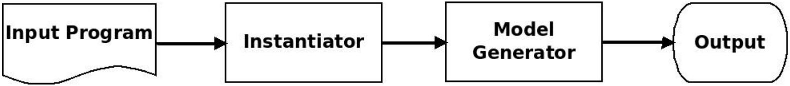
Evaluation of ASP programs



We have seen automated **algorithm selection** techniques for improving the performance of ASP systems, with focus on the multi-engine approach

Current contribution

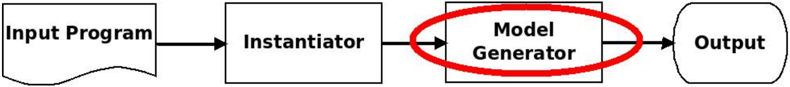
Evaluation of ASP programs



We have seen automated **algorithm selection** techniques for improving the performance of ASP systems, with focus on the multi-engine approach

Current contribution

Evaluation of ASP programs

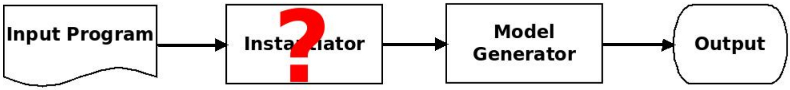


We have seen automated **algorithm selection** techniques for improving the performance of ASP systems, with focus on the multi-engine approach

Up to now, all applications are “confined” to the **Model Generator**

Current contribution

Evaluation of ASP programs



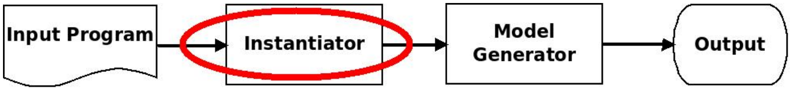
We have seen automated **algorithm selection** techniques for improving the performance of ASP systems, with focus on the multi-engine approach

Up to now, all applications are “confined” to the **Model Generator**

What about the **Instantiator**?

Current contribution

Evaluation of ASP programs



We have seen automated **algorithm selection** techniques for improving the performance of ASP systems, with focus on the multi-engine approach

Up to now, the applications are "confined" to the **Model Generator**

Current contribution

A first step toward the exploitation of automated algorithm selection techniques to the **Instantiator** (or, **grounder**).

Features and problems

Features:

- Problem size, balance and proximity to Horn features
- Presence of queries
- Maximum Strongly Connected Components size
- Features indicating if the program is recursive, tight, stratified
- ...

Features computed on instances of the *P* and *NP* domains **submitted** to the 3rd ASP Competition (**evaluated** instances have been discarded, and are used to test our solution).

Classification and model

- PART algorithm to automatically build an If-then-else decision list
  - Supervised classification algorithm
  - Patterns are the feature vectors
  - Classes (labels) are the grounders

Resulting model with DLV-G and GRINGO3 grounders

- DLV-G is usually preferable when
  - Dealing with queries
  - Program contain rules having large bodies (e.g., ≥ 4 literals) and the program has a simple structure (few components)
- GRINGO is usually preferable when
  - Recursive programs with many components
  - Most of the rules have a short body

Classification and model

- PART algorithm to automatically build an If-then-else decision list
  - Supervised classification algorithm
  - Patterns are the feature vectors
  - Classes (labels) are the grounders

Resulting model with DLV-G and GRINGO3 grounders

- DLV-G is usually preferable when
  - Dealing with queries
  - Program contain rules having large bodies (e.g., ≥ 4 literals) and the program has a simple structure (few components)
- GRINGO is usually preferable when
  - Recursive programs with many components
  - Most of the rules have a short body

Automated grounder SELECTOR

- A feature extractor for non-ground programs
- A decision-making module implementing the model

Solver	Grounder	*P*		*NP*		Total	
		#	Time	#	Time	#	Time
CLASP	DLV-G	48	128.26	93	62.65	141	84.99
	GRINGO	35	97.21	72	32.69	107	53.80
	SELECTOR	48	70.94	95	59.64	143	63.43
CMODELS	DLV-G	46	130.47	86	82.42	132	99.16
	GRINGO	32	116.29	67	58.44	99	77.14
	SELECTOR	46	70.60	87	80.72	133	77.22
DLV	DLV-G	41	129.17	59	71.39	100	95.08
	GRINGO	31	107.89	37	28.53	68	64.71
	SELECTOR	41	71.26	59	69.50	100	70.22
IDP	DLV-G	43	136.54	92	71.13	135	91.96
	GRINGO	32	140.57	72	46.97	104	75.77
	SELECTOR	43	74.16	94	70.19	137	71.43

Automated grounder SELECTOR

- A feature extractor for non-ground programs
- A decision-making module implementing the model

Solver	Grounder	P		NP		Total	
		#	Time	#	Time	#	Time
CLASP	DLV-G	48	128.26	93	62.65	141	84.99
	GRINGO	35	97.21	72	32.69	107	53.80
	SELECTOR	48	70.94	95	59.64	143	63.43
CMODELS	DLV-G	46	130.47	86	82.42	132	99.16
	GRINGO	32	116.29	67	58.44	99	77.14
	SELECTOR	46	70.60	87	80.72	133	77.22
DLV	DLV-G	41	129.17	59	71.39	100	95.08
	GRINGO	31	107.89	37	28.53	68	64.71
	SELECTOR	41	71.26	59	69.50	100	70.22
IDP	DLV-G	43	136.54	92	71.13	135	91.96
	GRINGO	32	140.57	72	46.97	104	75.77
	SELECTOR	43	74.16	94	70.19	137	71.43

Conclusion & Future Work

Both grounders and solvers are crucial for the performance of an ASP system

- a first step toward the exploitation of automated selection techniques to the grounding component

Our grounder selector improves the evaluation performance **independently from the solver** associated

Future Work

- Validate the results on domains of latest competitions
- Selector able to predict the best grounder+solver pair among a set of possible combinations

Other/more details at:

`https://www.mat.unical.it/~ricca/me-asp/`

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

(MPR, 2014a) A Multi-engine Approach to Answer Set Programming. TPLP. To appear in 2014

(MPR, 2014b) Multi-engine ASP Solving with Policy Adaptation. JCL. To appear in 2014