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

Marco Maratea\textsuperscript{1}, Luca Pulina\textsuperscript{2} and Francesco Ricca\textsuperscript{3}

\textsuperscript{1} DIBRIS, University of Genova  
\textsuperscript{2} POLCOMING, University of Sassari  
\textsuperscript{3} Dep. of Mathematics 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.
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
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 (MPR, 2014a) shows good performance in a setting in which

- training and test sets are composed of ground programs taken from the \textit{NP} and \textit{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:
- 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 $\Pi$ is made of rules:
$$a_1 \lor \ldots \lor a_n \leftarrow b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m$$

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

We have seen automated \textit{algorithm selection} techniques for improving the performance of ASP systems, with focus on the multi-engine approach.
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**.
Evaluation of ASP programs

We have seen automated \textit{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).
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., \( \geq 4 \) 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., \( \geq 4 \) 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

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

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

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

