Utilizing Software Features for Algorithm Selection

Damir Pulatov, Lars Kotthoff
Department of Computer Science, University of Wyoming
dpulatov@uwyo.edu, larsko@uwyo.edu

Motivation

- **Goal** – investigate whether algorithm selection can be improved if we utilize algorithm features along with instance features
- This iteration of the project uses static algorithm (software) features collected automatically
- Advantage – the number of performance models is constant no matter how many algorithms are used in scenario

Setup

- Run Experiments
- Source Code
- Feature Collection Tool
- Performance Measure
- Train Model
- Algorithm Properties
- Algorithm Selection

- Executed experiments on seven ASlib scenarios (SAT11-INDU, OPENML-WEKA-2017, etc) as well as scenarios not currently in ASlib (SAT-2018)
- Created SAT18-EXP scenario using main track results with 400 benchmark instances from SAT 2018 competition. Converted data into ASlib format using scripts from COSEAL's aslib-spec repository
- SAT18-EXP has all solvers that participated in the competition, except for variat since it was written in Rust (software metrics tool does not work)
- Obtained SAT18-EXP's instance features with SATzilla's feature collection tool
- Modified scenarios due to lack of source code: ambiguity in solvers, lack of ability to take into the account parameter settings, and repeated runs
- Automatically collected algorithmic features for solvers written in C++, Java, and x86 such as cyclomatic complexity (average and total), maxindent complexity (average and total), lines of code (average and total), size in bytes (average and total), and number of files
- Collected algorithmic features by selecting more relevant pieces of code (e.g., ignored code responsible for parallelism and certificate generation whenever possible)

Setup (cont.)

- Trained all models on Teton High-Performance Computing cluster
- Combined software and instance features by constructing a \( n \times m \) dataframe, where \( n \) is the number of instances times number of solvers, and \( m \) is the number of instance and software features
- Utilized server scripts from aslib-r for tuning hyperparameters for individual models. Tuning for combined models was done similarly (e.g., nested cross-validation and so on)
- Combined model is a Random Forest regression model that utilizes both instance and software features
- Individual model is the standard model that uses instance features only
- Models with pair regression method available in LLAMA were also used to see if combined regression model performs better than a slightly modified individual model
- \( \text{mcp} \) and \( \text{par10} \) gaps show the normalized fraction of the gap closed by different methods
- A value of 0 corresponds to the single best solver and a value of 1 to the virtual best. Negative values indicate performance worse than the single best solver

Results

<table>
<thead>
<tr>
<th>Scenario</th>
<th>( \text{mcp} ) gap</th>
<th>( \text{par10} ) gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSP-LION2015</td>
<td>-0.17 to -0.11</td>
<td>-0.42 to -0.22</td>
</tr>
<tr>
<td>SAT11-EXP</td>
<td>0.59 to 0.61</td>
<td>0.29 to 0.32</td>
</tr>
<tr>
<td>SAT11-RAND</td>
<td>0.58 to 0.59</td>
<td>0.46 to 0.48</td>
</tr>
<tr>
<td>SAT11-INDU</td>
<td>0.41 to 0.39</td>
<td>0.13 to 0.05</td>
</tr>
<tr>
<td>SAT11-HAND</td>
<td>0.58 to 0.67</td>
<td>0.24 to 0.28</td>
</tr>
<tr>
<td>SAT03-1E-INDU</td>
<td>0.46 to 0.44</td>
<td>0.13 to 0.12</td>
</tr>
<tr>
<td>OPENML-WEKA-2015</td>
<td>-0.42 to 0.22</td>
<td>-0.51 to 0.49</td>
</tr>
<tr>
<td>GRAPHS-2015</td>
<td>0.46 to 0.62</td>
<td>0.12 to 0.15</td>
</tr>
</tbody>
</table>

Future Work

- Investigate better static algorithmic features (a lot of minisat hacked solvers have very similar values)
- Take into the account data structures and Object-Oriented properties
- Collect dynamic algorithmic features that characterize only the parts of software that were executed during runtime (stack traces)
- Find a way to automatically analyze more relevant pieces of source code related to computation (e.g., ignore code used for GUIs and so on)
- Add feature costs for algorithm properties

Summary

- Building algorithm selection models with current static features produces mixed and inconsistent results
- Some scenarios (OPENML) are improved, some stay about same (SAT01-1E-INDU), and others worsen (SAT11-RAND)
- Performing pair regression with instance features gives a much larger improvement on some scenarios compared to combined regression model

References

2. Metrics-z is a tool to collect and analyze code metrics. URL: https://metrics-z.github.io/home.html