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 (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).

Results

- 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\(^2\) were also used to see if combined regression model performs better than a slightly modified individual model
- \( mcp \) and \( par10 \) 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
- OPENML was grayed out for \( par10 \) table since this metric does not make sense for the scenario

Future Work

- Build pair regression models that use both software and instance features to see if they perform any better (currently running experiments)
- Perform feature selection (forward and backward) to find out which software features will be filtered out
- 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 trace)
- Find a way to automatically analyze more relevant pieces of source code

References

[2] Metrix++ is a tool to collect code metrics. URL: https://metrixplusplus.github.io/home.html