Algorithm Selection Library: Format Specification

October 2, 2014

Bernd Bischl[a], Lars Kothoff[b], Marius Lindauer[c], Yuri Malitsky[b], Alexandre Frechétte[d], Holger Hoos[d], Frank Hutter[c], Pascal Kerschke[e], Kevin Leyton-Brown[d], Kevin Tierney[f], Joaquin Vanschoren[g]

[a] *University of Dortmund; Germany*
[b] *Cork Constraint Computation Centre, Ireland*
[c] *University of Freiburg, Germany*
[d] *University of British Columbia, Vancouver, Canada*
[e] *University of Münster; Germany*
[f] *University of Paderborn, Germany*
[g] *Eindhoven University of Technology, The Netherlands*

---

1. Directory / File Layout

- Every algorithm selection scenario is stored in an individual folder on the server.
- The folder contains at most one copy of the following, optional files are marked with an (*)
  - *readme.txt*: Human-readable with general information.
  - *description.txt*: Global definitions for the scenario.
  - *feature_values.arff*: Values of features used to predict solvers.
  - *feature_runstatus.arff*: Technical info about features, e.g., aborts.
  - *feature_costs.arff* (*): Costs of feature (step) calculation.
  - *algorithm_runs.arff*: Algorithm runs and performance measurements.
  - *cv.arff* (*): Cross-validation splits for evaluation. Users are allowed to (if they have good reason) to submit this file, but in general it will be generated on the repository server automatically, when the rest of the data set is submitted.

Email addresses: bischl@statistik.tu-dortmund.de (Bernd Bischl), larsko@4c.ucc.ie (Lars Kothoff), manju@cs.uni-potsdam.de (Marius Lindauer), y.malitsky@4c.ucc.ie (Yuri Malitsky), afrechet@cs.ubc.ca (Alexandre Frechétte), hoos@cs.ubc.ca (Holger Hoos), fh@informatik.uni-freiburg.de (Frank Hutter), kerschke@uni-muenster.de (Pascal Kerschke), kevinlb@cs.ubc.ca (Kevin Leyton-Brown), tierney@dsor.de (Kevin Tierney), j.vanschoren@tue.nl (Joaquin Vanschoren)

– *ground_truth.arff* (*): Information about solution of instances.
– *citation.bib* (*): Citation file for data set.

2. General Remarks

Please keep the following in mind when generating the files.

- Every file base name / column name / factor level name adheres to: ASCII characters; if whitespaces or commas are used, the name has to be quoted
- Instance IDs can also contain "/" to represent an original file path.
- We **highly** recommend using descriptive instance names rather than using numerical IDs. One possible purpose for this is that such descriptive instance names can later be used to reproduce experiments.
- All pairs of instance ID and repetition number have to be unique.
- Use "?" to denote missing values conforming to the rules of the ARFF mechanism. Each such "?" must be explained in the *readme* making it clear why the missing values occurred where they did. Also, use the "runstatus" concept, explained below, for features and algorithms. Never use anything else except "?" for missing values.
- relations of all *arff*-files should begin with the information type and end with scenario_id (see *description.txt*); e.g., if we have instance features of the 2013 SAT Competition, the relation name of the *feature_values* should be `@RELATION FEATURE_VALUES_2013-SAT-Competition`. In case of doubt, this helps to identify files which belong together

3. readme.txt

This is a human readable file meant to allow the creator to explain any and all peculiarities of how the data came into existence. Most importantly, this file is where the creators must explain the reason for any missing data in the set. In addition to this, consider the following as a starting list of things that must be mentioned in this file. But please be aware that explaining things here is no excuse to not have perfectly machine-readable information in the other files or for circumventing format specifications!

- Must describe where and how the data originated.
- Describe the problem that the algorithms are supposed to solve and how their performance is measured.
- Describe the used solvers, their configurations, versions and origins.
- Reference the feature generator.
- Try to include all relevant details of the underlying experiment.

4. description.txt

A global description file containing general information about the scenario. This is one of the few files which is not an ARFF, but for machine readability it follows specific guidelines for its generation. Each line contains very specific bits of information. Each item in the list that follows specifies the format of the corresponding line. As such, it first presents the line format followed by an explanation. The two parts are separated by a colon.

- scenario_id [string]: Name of the scenario
  This id is necessary for human bookkeeping and making sure that all other files in the directory are for the same set of experiments. In the ARFF files, this id will be present under the "@RELATION" line.

- performance_measures [comma-separated list of strings]: Name of performance metrics measured for the algorithms
  While in many cases only a single entry might be reasonable, there are times when an experiment may yield secondary objectives. As such, all these measures must be listed in decreasing order of importance, if there is such an order, the first one being the one of primary interest.

- maximize [comma-separated list of Booleans]: true / false
  For each of the objectives listed in the *performance_measures* line, this line specifies if the objective is to minimize or maximize the corresponding value. The number of entries must match number of entries in "performance_measures".

- performance_type [comma-separated list of factors]: runtime / solution_quality
  For each of the objectives listed for "performance_measures", this line specifies whether the corresponding metric is "runtime" or "solution_quality". Number of entries must match number of entries in "performance_measures".

- algorithm_cutoff_time [integer]: Cut-off time in seconds
  Algorithms are terminated after this time if they did not produce a successful solution. Might be used both for "runtime" and "solution_quality". Put "?" if it was not set for experiment.

- algorithm_cutoff_memory [integer]: Cut-off memory in megabytes
  Algorithms are terminated if their memory exceeded this value. Might be used both for "runtime" and "solution_quality". Put "?" if it was not set for experiment.

- features_cutoff_time [integer]: Cut-off time in seconds
  Feature set computation is terminated if it exceeds this time. Might be used both for "runtime" and "solution_quality". Put "?" if it was not set for experiment.

- features_cutoff_memory [integer]: Cut-off memory in megabytes
  Feature set computation is terminated if memory exceeded this value.

Might be used both for "runtime" and "solution_quality". Put "?" if it was not set for experiment.

- algorithms_deterministic: [comma-separated list of strings]
  List names of all algorithms which are deterministic.

- algorithms_stochastic: [comma-separated list of strings]
  List names of all algorithms which are stochastic.

- The next lines allow to define dependencies of features to processing steps. For instance, probing features need normally a preprocessing step. Therefore probing features depend on the preprocessing step and the probing step. This is helpful for *feature_costs.arff*, as feature generators often calculate features in several steps. The first line should define the number of steps that are to be listed. Each feature has to be listed in at least one step. To be able to use a feature, all processing steps have to be used in which the feature is listed. On the other hand, if $S' \subseteq S$ processing steps are used and we have a set of features $F(S)$ listed in S, you are able to use features $F' = F(S) \backslash F(S \backslash S')$, i.e., features that are not listed in not used steps. The steps have to be sorted according to their execution sequence. Their definition looks like this:

  number_of_feature_steps: 2
  feature_step name_1: feature1, feature2, feature4
  feature_step name_2: feature3, feature4

  You are free to use any step names you like, as long as they are unique and do not contain illegal characters.

- default_steps: [comma-separated list of strings]
  List names of all features which should be used as a default.

- features_deterministic: [comma-separated list of strings]
  List names of all features processing steps which are deterministic.

- features_stochastic: [comma-separated list of strings]
  List names of all features which are stochastic.

Listing 1: Example description.txt

```
scenario_id: 2013-SAT-Competition
performance_measures: PAR10, Number_of_satisfied_clauses
maximize: false, true
performance_type: runtime, solution_quality
algorithm_cutoff_time: 900
algorithm_cutoff_memory: 6000
features_cutoff_time: 90
features_cutoff_memory: 6000
algorithms_deterministic: lingeling, clasp
algorithms_stochastic: sparrow
number_of_feature_steps: 2
```

```
12 feature_step preprocessing: number_of_variables, first_local_min_steps
13 feature_step local_search_probing: first_local_min_steps
14 default_steps: preprocessing
15 features_deterministic: number_of_variables, first_local_min_steps
16 features_stochastic: first_local_min_steps
```

5. feature_values.arff

This file contains the numerical feature values for all instances. Specifically, it is generally assumed that this is the information that will be used by implemented techniques to differentiate between different instances. Like with all subsequent files, the data stored here should follow the conventions of an ARFF file. It is therefore expected that the rows will correspond to a specific (instance, repetition) pair.

- The first column is called "instance_id" and contains the instance name represented as a string. These names must follow the guidelines of utilizing only ASCII characters.
- The second column specifies the "repetition" and contains integers, starting at 1 and increasing in increments of 1 for each instance with the same name.
- The following columns correspond to instance features. We allow numeric, integer and categorical columns, but only the names defined in *description.txt* can be used. In that file, these names were specified in fields "features_step".
- "?" means the feature value is missing because the feature calculation algorithm crashed or aborted or some other problem occurred. The explanation of such a missing value will need to be registered in *feature_runstatus.arff*.
- In the case of stochastic features, the user might opt to repeat the feature calculation (column "repetition"). The following remarks have to be respected:
  - In case of no repetitions, do not omit the column, simply put a 1 in every entry.
  - If you use repetitions, you can use a different number of repetitions for instances (if you really insist on it), but always the same number of repetitions for features in one row (the latter is enforced by the format definition).
  - If you use repetitions, but some features are deterministic, simply repeat their feature values across the repetitions.
  - Whether features are stochastic or deterministic is defined implicit over the feature steps in "features_steps_deterministic" and "features_steps_stochastic" in in *description.txt*.

Listing 2: Example feature_values.arff with three features

```
1  @RELATION FEATURE_VALUES_2013-SAT-Competition

3  @ATTRIBUTE instance_id STRING
4  @ATTRIBUTE repetition NUMERIC
5  @ATTRIBUTE number_of_variables NUMERIC
6  @ATTRIBUTE number_of_clauses NUMERIC
7  @ATTRIBUTE first_local_min_steps NUMERIC

9  @DATA
10 inst1.cnf,1,101,24,33
11 inst1.cnf,2,101,24,42
12 inst2.cnf,1,303,105,12
13 inst2.cnf,2,303,105,?
14 inst3.cnf,1,1002,337,?
15 inst3.cnf,2,1002,337,?
16 ...
```

6. feature_runstatus.arff

This file contains technical information about the feature calculation in general. Specifically, it serves to state whether the execution of feature processing steps was successful or if some crash occurred. The file is designed to allow the user to specify what kind of a crash has transpired, but the reasons for each such situation must be explained in the *readme.txt* file.

We assume that feature processing steps are able to generate features if its execution terminates successfully. If a step terminates because of another reason, e.g., timeout or memout, all features are not available which depend on this processing step as defined in *description.txt*.

- The first column is the "instance_id" and contains the instance name in string format. These names must follow the guidelines of utilizing only ASCII characters.
- The second column specifies the "repetition" and contains integers, starting at 1 and increasing in increments of 1 for each instance with the same name.
- The two columns, "instance_id" and "repetition", must represent exactly the same data as presented in *feature_values.arff*.
- All remaining columns correspond to feature processing steps. Here, the same column names must be adhered to as was defined in *description.txt*. The entries are categoric values specifying the termination condition for each step. The supported range of these values is:
  - "ok" - The step was executed normally.

  - "timeout" - Time ran out before this step terminated.
  - "memout" - Feature computation ran out of memory causing a crash.
  - "presolved" - Instance was solved during feature computation.
  - "crash" - The program failed to execute.
  - "other" - Some other critical problem occurred while this feature was computed.

- If feature calculation is aborted ("crash" or "other"), you must explain the reasons in the *readme.txt*.

- If at least one used steps has the state "presolved", the instance was solved during feature set calculation.

Listing 3: Example feature_runstatus.arff with two feature processing steps

```
1  @RELATION FEATURE_RUNSTATUS_2013-SAT-Competition

3  @ATTRIBUTE instance_id STRING
4  @ATTRIBUTE repetition NUMERIC
5  @ATTRIBUTE preprocessing {ok, timeout, memout, presolved, crash, other}
6  @ATTRIBUTE local_search_probing {ok, timeout, memout, presolved, crash, other}

8  @DATA
9  inst1.cnf,1,ok,ok
10 inst1.cnf,2,ok,ok
11 inst2.cnf,1,ok,ok
12 inst2.cnf,2,ok,timeout
13 inst3.cnf,1,ok,memout
14 inst3.cnf,2,ok,presolved
15 ...
```

7. feature_costs.arff

Although usually a minor overhead, feature computation is rarely free. In many cases this cost is in the form of time, but no such requirement is made. Instead, this file specifies the cost of computing each feature processing step.

We strongly recommend to provide information about feature costs. However, you are allowed to omit this file if your scenario does not entail feature costs or you have absolutely no knowledge of them. Please note that certain analysis, e.g., runtime improvement against best single algorithm, are not possible, if feature costs are not provided.

- The first column is the "instance_id" and contains the instance name represented as a string. These names must follow the guidelines of utilizing only ASCII characters

- The second column is an integer specifying the "repetition", starting at 1 and increasing in increments of 1 per each instance repetition.
- The "instance_id" and "repetition" columns must represent exactly the same data as in *feature_values.arff*.
- All columns correspond to the costs of each feature processing steps. The names of these steps must be exactly the same as those specified in *description.txt* in the feature step section. Entries are numerical, they specify the cost of the processing step for that instance / repetition. We recommend to specify a feature processing step for each feature, if the costs for each feature is known. If only the computation cost for all features is known, specify exactly one processing step.
- Put "?" as an entry if the feature computation was not successful due to cut-off or unusual abort.

Listing 4: Example feature_costs.arff

```
1  @RELATION FEATURE_COSTS_2013-SAT-Competition

3  @ATTRIBUTE instance_id STRING
4  @ATTRIBUTE repetition NUMERIC
5  @ATTRIBUTE preprocessing NUMERIC
6  @ATTRIBUTE local_search_probing NUMERIC

8  @DATA
9  inst1.cnf,1,0.1,10.0
10 inst1.cnf,2,0.1,12.0
11 inst2.cnf,1,0.2,4.0
12 inst2.cnf,2,0.2,?
13 inst3.cnf,1,1.1,?
14 inst3.cnf,2,1.1,?
15 ...
```

8. algorithm_runs.arff

This file contains information on all algorithms evaluated on the instances. This includes, but not limited to, their performance values and runstatus. In practice, certain algorithms may be evaluated multiple times, while others can be performed just once. For example, this is the case for stochastic and deterministic solvers, respectively. To cover both scenarios, each row of this file specifies a single experiment. Please note that different algorithms are allowed to have different numbers of repetitions. Similarly, the number of repetitions used for stochastic algorithms does not have to coincide with the number of repetitions used for stochastic features.

- Each row corresponds to the measurement of 1 algorithm on one instance / repetition.
- The first column is the "instance_id" and contains the instance name as a string.
- The second column is an integer that states the "repetition", starting at 1 and increasing in increments of 1 per each instance repetition. The number of repetitions can vary between the algorithms, e.g., the performance of deterministic algorithms are only measured once and of stochastic ones more than once. However, the combination of algorithm and number of repetition should be the same for all instances.
- "instance_id" must contain exactly the same elements as the corresponding column in *feature_values.arff*.
- The third column is called "algorithm", a string value specifying the name of the evaluated algorithm. Only the names defined in *description.txt* may be used; those listed under the fields "algorithms_deterministic" and "algorithms_stochastic".
- All but the last subsequent columns are numerical, containing the algorithm performance measurements.
- The last column is called "runstatus" and contains whether the algorithm terminated normally or the reason for an abort.
  - "ok" - The algorithm finished properly.
  - "timeout" - Time ran out prior to completion.
  - "memout" - The algorithm required more memory than permitted.
  - "not_applicable" - This algorithm can't be run on this instance.
  - "crash" - The program failed to execute.
  - "other" - Something really unexpected happened.
- If the run was aborted ("crash" or "other"), you have to explain the reasons in the *readme.txt*.
- "?" are not permitted in this file. In case of scenarios with runtime as performance type, the runtime is always known regardless of the runstatus. In case of scenarios with solution quality as performance type, missing values have to be imputed somehow. However, this imputation is domain-specific and hence, it has to be provided by the scenario designer.

Listing 5: Example algorithm_runs.arff with four algorithms

```
1 @RELATION ALGORITHM_RUNS_2013-SAT-Competition

3 @ATTRIBUTE instance_id STRING
```

```
4  @ATTRIBUTE repetition NUMERIC
5  @ATTRIBUTE algorithm STRING
6  @ATTRIBUTE PAR10 NUMERIC
7  @ATTRIBUTE Number_of_satisfied_clauses NUMERIC
8  @ATTRIBUTE runstatus {ok, timeout, memout, not_applicable, crash, other}

10 inst1.cnf, 1, lingeling, 5.0, 24, ok
11 inst1.cnf, 1, clasp, 5.8, 24, ok
12 inst1.cnf, 1, sparrow, 900, 20, timeout
13 inst1.cnf, 2, sparrow, 1.7, 24, ok
14 inst2.cnf, 1, lingeling, 588.1, 105, ok
15 inst2.cnf, 1, clasp, 50, 0, memout
16 inst2.cnf, 1, sparrow, 501, 105, ok
17 inst2.cnf, 2, sparrow, 900, 101, timeout
18 ...
```

9. cv.arff

This file contains information how to split the set of instances according to a cross-validation scheme to assess the performance of an algorithm selector in an unbiased and standard fashion. The file is provided to guarantee that all algorithm selectors are evaluated on the same splits. We support repeated cross-validation, where the process of a single cross-validation is simply repeated. These repetitions are a standard technique for smaller data sets or in cases where other sources of instability might be present in the data.

- The first column is the "instance_id" and contains the instance name as a string. The column must contain exactly the same elements as the corresponding column in *feature_values.arff* at least once.
- The second column is an integer that states the "repetition", starting at 1 and increasing in increments of 1 per each instance repetition. Note that this is for repeated cross-validation and has no connection to repeated algorithm runs or repeated feature evaluations. The column is always present, for a simple cross-validation it is always 1.
- The third column is an integer that states the "fold" of the cross-validation, starting at 1.

Listing 6: Example cv.arff for two times a 2-fold cross validation

```
1  @RELATION CV_2013-SAT-Competition

3  @ATTRIBUTE instance_id STRING
4  @ATTRIBUTE repetition NUMERIC
5  @ATTRIBUTE fold NUMERIC

7  inst1.cnf, 1, 1
```

```
inst2.cnf, 1, 1
inst3.cnf, 1, 2
inst4.cnf, 1, 2
inst1.cnf, 2, 1
inst2.cnf, 2, 2
inst3.cnf, 2, 2
inst4.cnf, 2, 1
...
```

10. ground_truth.arff

This is an optional file meant to keep the known information about the instance. This information can be used both as a sanity check to make sure the solvers return correct solutions, but can also be used for cases where a new evaluation metric depends on the optimal solution. Each row specifies exactly one instance, and no duplicates are allowed.

- Rows correspond to instances.
- First column is called "instance_id" and contains a string representing the instance name.
- The "instance_id" column has exactly the same data as in *feature_values.arff*.
- The other columns provide information about ground truths of each instance, e.g.,
  - Satisfiable or Unsatisfiable for SAT instances.
  - Optimal quality of optimization problems.
  - Any other known optimal value of a metric.
- Put "?" if you do not know the ground truth for a particular instance.
- Omit this file if you do not have this information for any instance.

Listing 7: Example ground_truth.arff

```
@RELATION GROUND_TRUTH_2013-SAT-Competition

@ATTRIBUTE instance_id STRING
@ATTRIBUTE SATUNSAT {SAT,UNSAT}
@ATTRIBUTE OPTIMAL_VALUE NUMERIC

@DATA
inst1.cnf, SAT, 24
inst2.cnf, UNSAT, 105
...
```

11. citation.bib

Please provide a bibtex file containing the references that should be cited when using this data

References

[1] Ansótegui, C., Malitsky, Y., Sellmann, M., 2014. Maxsat by improved instance-specific algorithm configuration, in: AAAI.

[2] Ansótegui, C., Sellmann, M., Tierney, K., 2009. A Gender-Based genetic algorithm for the automatic configuration of algorithms, in: CP, pp. 142–157.

[3] Arbelaez, A., Hamadi, Y., Sebag, M., 2009. Online heuristic selection in constraint programming, in: International Symposium on Combinatorial Search (SoCS).

[4] Argelich, J., Berre, D.L., Lynce, I., Marques-Silva, J., Rapicault, P., 2010. Solving linux upgradeability problems using boolean optimization, in: International Workshop on Logics for Component Configuration (LoCoCo), pp. 11–22.

[5] Argelich, J., Li, C., Manyà, F., Planes, J., 2012. Seventh MaxSAT Evaluation. `http://www.maxsat.udl.cat/12/`.

[6] Asn, R., Nieuwenhuis, R., 2010. `http://www.lsi.upc.edu/~rasin/timetabling.html`.

[7] Babic, D., Hutter, F., 2008. Spear theorem prover. Solver description, International SAT Competition.

[8] Baral, C., 2003. Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press.

[9] Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T., 2010. in: Bartz-Beielstein, T., Chiarandini, M., Paquete, L., Preuss, M. (Eds.), Empirical Methods for the Analysis of Optimization Algorithms. Springer. chapter F-race and iterated F-race: An overview.

[10] Bischl, B., 2014. mlr: Machine Learning in R. R package version 1.2. `https://github.com/berndbischl/mlr`.

[11] Bischl, B., Kotthoff, L., Lindauer, M., Malitsky, Y., Frechétte, A., Hoos, H., Hutter, F., Kerschke, P., Leyton-Brown, K., Vanschoren, J., 2014. Algorithm Selection Format Specification. Technical Report. Available at `http://www.aslib.net/`.

[12] Bischl, B., Lang, M., Mersmann, O., Rahnenfuehrer, J., Weihs, C., 2012a. Computing on high performance clusters with R: Packages BatchJobs and BatchExperiments. Technical Report. TU Dortmund. URL: http://sfb876.tu-dortmund.de/PublicPublicationFiles/bischl_etal_2012a.pdf.

[13] Bischl, B., Mersmann, O., Trautmann, H., Preuss, M., 2012b. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning, in: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, pp. 313–320. doi:10.1145/2330163.2330209.

[14] Bortfeldt, A., Forster, F., 2012. A tree search procedure for the container pre-marshalling problem. European Journal of Operational Research 217, 531–540.

[15] Bregman, D.R., 2009. The sat solver mxc, version 0.99 (2009 sat competition version). Solver description, International SAT Competition.

[16] Cai, S., Su, K., 2012. Configuration checking with aspiration in local search for SAT, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, pp. 434–440.

[17] Caserta, M., Voß, S., 2009. A corridor method-based algorithm for the pre-marshalling problem, in: Giacobini, M. (Ed.), Applications of Evolutionary Computing, Springer. pp. 788–797.

[18] Chen, J., 2011. Phase selection heuristics for satisfiability solvers. CoRR abs/1106.1372.

[19] Cicirello, V.A., Smith, S.F., 2005. The max k-armed bandit: A new model of exploration applied to search heuristic selection, in: Proceedings of the 20th National Conference on Artificial Intelligence, AAAI Press. pp. 1355–1361.

[20] Cook, D.J., Varnell, R.C., 1997. Maximizing the benefits of parallel search using machine learning, in: Proceedings of the 14th National Conference on Artificial Intelligence, AAAI Press. pp. 559–564.

[21] Crawford, J.M., Baker, A.B., 1994. Experimental results on the application of satisfiability algorithms to scheduling problems, in: In Proceedings of the Twelfth National Conference on Artificial Intelligence, pp. 1092–1097.

[22] Davis, M., Logemann, G., Loveland, D., 1962. A machine program for theorem proving. Communications of the ACM 5, 394–397.

[23] Demmel, J., Dongarra, J., Eijkhout, V., Fuentes, E., Petitet, A., Vuduc, R., Whaley, R.C., Yelick, K., 2005. Self-Adapting linear algebra algorithms and software. Proceedings of the IEEE 93, 293–312.

[24] Fawcett, C., Vallati, M., Hutter, F., Hoffmann, J., Hoos, H., Leyton-Brown, K., 2014. Improved features for runtime prediction of domain-independent planners, in: Proceedings of ICAPS 2014. To appear.

[25] Gagliolo, M., Zhumatiy, V., Schmidhuber, J., 2004. Adaptive online time allocation to search algorithms, in: Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D. (Eds.), Machine Learning: ECML 2004. Springer. volume 3201 of *Lecture Notes in Computer Science*, pp. 134–143. doi:10.1007/978-3-540-30115-8_15.

[26] Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Schneider, M., 2011a. Potassco: The Potsdam answer set solving collection. AI Communications 24, 107–124.

[27] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., 2012a. Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers.

[28] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., Schneider, M.T., Ziller, S., 2011b. A portfolio solver for answer set programming: preliminary report, in: 11th International Conference on Logic Programming and Nonmonotonic Reasoning, Springer. pp. 352–357.

[29] Gebser, M., Kaufmann, B., Schaub, T., 2012b. Multi-threaded ASP solving with clasp. Theory and Practice of Logic Programming 12, 525–545.

[30] Gent, I., Jefferson, C., Kotthoff, L., Miguel, I., Moore, N., Nightingale, P., Petrie, K., 2010. Learning when to use lazy learning in constraint solving, in: 19th European Conference on Artificial Intelligence, IOS Press. pp. 873–878.

[31] Gomes, C.P., Selman, B., 2001. Algorithm portfolios. Artificial Intelligence 126, 43–62.

[32] Grasso, G., Iiritano, S., Leone, N., Lio, V., Ricca, F., Scalise, F., 2010. An ASP-based system for team-building in the Gioia-Tauro seaport, in: Carro, M., Peña, R. (Eds.), Proceedings of the Twelfth International Symposium on Practical Aspects of Declarative Languages, Springer. pp. 40–42.

[33] Guerri, A., Milano, M., 2004. Learning techniques for automatic algorithm portfolio selection, in: Proceedings of the 16th Eureopean Conference on Artificial Intelligence, IOS Press. pp. 475–479.

[34] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., 2009. The WEKA data mining software: An update. SIGKDD Explor. Newsl. 11, 10–18.

[35] Helmert, M., Röger, G., Karpas, E., 2011. Fast downward stone soup: A baseline for building planner portfolios, in: ICAPS-2011 Workshop on Planning and Learning (PAL), pp. 28–35.

[36] Hoos, H., Lindauer, M., Schaub, T., 2014a. claspfolio 2: Advances in algorithm selection for answer set programming. Under review at the 30th International Conference on Logic Programming.

[37] Hoos, H.H., Kaminski, R., Lindauer, M., Schaub, T., 2014b. aspeed: Solver scheduling via answer set programming. Theory and Practice of Logic Programming , 1–26.

[38] Hoos, H.H., Stützle, T., 2004. Stochastic local search: Foundations & applications. Elsevier.

[39] Hothorn, T., Hornik, K., Zeileis, A., 2006. Unbiased recursive partitioning: A conditional inference framework. Journal of Computational and Graphical Statistics 15, 651–674.

[40] Hurley, B., Kotthoff, L., Malitsky, Y., O'Sullivan, B., 2014. Proteus: A hierarchical portfolio of solvers and transformations, in: Proceedings of the 11th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems.

[41] Hutter, F., Babić, D., Hoos, H.H., Hu, A.J., 2007. Boosting Verification by Automatic Tuning of Decision Procedures, in: Formal Methods in Computer Aided Design, IEEE Computer Society, Washington, DC, USA. pp. 27–34.

[42] Hutter, F., Hamadi, Y., Hoos, H.H., Leyton-Brown, K., 2006. Performance prediction and automated tuning of randomized and parametric algorithms, in: Proceedings of the 12th International Conference on Principles and Practice of Constraint Programming, pp. 213–228.

[43] Hutter, F., Hoos, H.H., Leyton-Brown, K., 2010. Automated configuration of mixed integer programming solvers, in: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 186–202.

[44] Hutter, F., Hoos, H.H., Leyton-Brown, K., 2011. Sequential model-based optimization for general algorithm configuration, in: Learning and Intelligent Optimization, pp. 507–523.

[45] Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T., 2009. ParamILS: an automatic algorithm configuration framework. J. Artif. Int. Res. 36, 267–306.

[46] Hutter, F., nez, M.L.I., Fawcett, C., Lindauer, M., Hoos, H., Leyton-Brown, K., Stützle, T., 2014a. Aclib: a benchmark library for algorithm configuration, in: Learning and Intelligent Optimization. To appear.

[47] Hutter, F., Xu, L., Hoos, H.H., Leyton-Brown, K., 2014b. Algorithm runtime prediction: Methods & evaluation. Artificial Intelligence 206, 79–111.

[48] Ishebabi, H., Mahr, P., Bobda, C., Gebser, M., Schaub, T., 2009. Answer set vs integer linear programming for automatic synthesis of multiprocessor systems from real-time parallel programs. Journal of Reconfigurable Computing. URL: http://www.hindawi.com/journals/ijrc/2009/863630.html. Article ID 863630.

[49] Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M., 2011. Algorithm selection and scheduling, in: Lee, J. (Ed.), Principles and Practice of Constraint Programming. Springer. volume 6876 of *Lecture Notes in Computer Science*, pp. 454–469. doi:10.1007/978-3-642-23786-7_35.

[50] Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K., 2010. ISAC Instance-Specific Algorithm Configuration, in: 19th European Conference on Artificial Intelligence, IOS Press. pp. 751–756.

[51] Karatzoglou, A., Smola, A., Hornik, K., Zeileis, A., 2004. kernlab – an S4 package for kernel methods in R. Journal of Statistical Software 11, 1–20. URL: http://www.jstatsoft.org/v11/i09/.

[52] Kautz, H., Selman, B., 1999. Unifying sat-based and graph-based planning, in: Proceedings of the 16th International Joint Conference on Artifical Intelligence, Morgan Kaufmann. pp. 318–325.

[53] Kerschke, P., Preuss, M., Hernandez, C., Schuetze, O., Sun, J.Q., Grimme, C., Rudolph, G., Bischl, B., Trautmann, H., 2014. Cell mapping techniques for exploratory landscape analysis, in: EVOLVE : A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation. To appear.

[54] Kotthoff, L., 2013. LLAMA: Leveraging Learning to Automatically Manage Algorithms. Technical Report arXiv:1306.1031. arXiv. http://arxiv.org/abs/1306.1031.

[55] Kotthoff, L., 2014. Algorithm selection for combinatorial search problems: A survey. AI Magazine. To appear.

[56] Kotthoff, L., Gent, I.P., Miguel, I., 2012. An evaluation of machine learning in algorithm selection for search problems. AI Communications 25, 257–270.

[57] Lagoudakis, M., Littman, M., 2000. Algorithm selection using reinforcement learning, in: ICML, pp. 511–518.

[58] Lagoudakis, M., Littman, M., 2001. Learning to select branching rules in the DPLL procedure for satisfiability, in: SAT, pp. 344–359.

[59] Lee, Y., Hsu, N., 2007. An optimization model for the container premarshalling problem. Computers & Operations Research 34, 3295–3313.

[60] Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., Shoham, Y., 2003. A portfolio approach to algorithm selection, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Morgan Kaufmann. pp. 1542–1543.

[61] Liaw, A., Wiener, M., 2002. Classification and regression by randomforest. R News 2, 18–22. URL: http://CRAN.R-project.org/doc/Rnews/.

[62] Malitsky, Y., Mehta, D., O'Sullivan, B., 2013a. Evolving instance specific algorithm configuration, in: The Sixth Annual Symposium on Combinatorial Search.

[63] Malitsky, Y., O'Sullivan, B., Previti, A., Marques-Silva, J., 2014. A portfolio approach to enumerating minimal correction subsets for satisfiability problems, in: CPAIOR.

[64] Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M., 2011. Non-model-based algorithm portfolios for SAT, in: Sakallah, K.A., Simon, L. (Eds.), 14th International Conference on Theory and Applications of Satisfiability Testing, Springer. pp. 369–370.

[65] Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M., 2013b. Algorithm portfolios based on cost-sensitive hierarchical clustering, in: IJCAI.

[66] Maratea, M., Pulina, L., Ricca, F., 2013. A multi-engine approach to answer-set programming. Theory and Practice of Logic Programming , 1–28.

[67] Mersmann, O., Bischl, B., Trautmann, H., Wagner, M., Bossek, J., Neumann, F., 2013. A novel feature-based approach to characterize algorithm performance for the traveling salesperson problem. Annals of Mathematics and Artificial Intelligence , 1–32doi:10.1007/s10472-013-9341-2.

[68] Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., 2014. e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. URL: http://CRAN.R-project.org/package=e1071. r package version 1.6-3.

[69] Milborrow, S., 2014. earth: Multivariate Adaptive Regression Spline Models. URL: http://CRAN.R-project.org/package=earth. r package version 3.2-7.

[70] Nikolić, M., Marić, F., Janičić, P., 2009. Instance-based selection of policies for SAT solvers, in: Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, Springer. pp. 326–340.

[71] Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., Barry, M., 2001. An A-prolog decision support system for the space shuttle, in: Ramakrishnan, I. (Ed.), Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages, Springer. pp. 169–183.

[72] Nudelman, E., Leyton-Brown, K., Andrew, G., Gomes, C., McFadden, J., Selman, B., Shoham, Y., 2003. Satzilla 0.9. Solver description, International SAT Competition.

[73] Nudelman, E., Leyton-Brown, K., Hoos, H.H., Devkar, A., Shoham, Y., 2004. Understanding random SAT: beyond the Clauses-to-Variables ratio, in: Wallace, M. (Ed.), Principles and Practice of Constraint Programming CP 2004, Springer. pp. 438–452.

[74] O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., O'Sullivan, B., 2008. Using case-based reasoning in an algorithm portfolio for constraint solving, in: Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science.

[75] Park, J., 2002. Using weighted max-sat engines to solve mpe, in: AAAI, pp. 682–687.

[76] Prasad, M.R., Biere, A., Gupta, A., 2005. A survey of recent advances in sat-based formal verification. International Journal on Software Tools for Technology Transfer 7, 156–173.

[77] Pulina, L., Tacchella, A., 2007. A multi-engine solver for quantified boolean formulas, in: Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming, Springer. pp. 574–589.

[78] Pulina, L., Tacchella, A., 2009. A self-adaptive multi-engine solver for quantified boolean formulas. Constraints 14, 80–116.

[79] Purdom, P.W., Le Berre, D., Simon, L., 2005. A parsimony tree for the SAT2002 competition. Annals of Mathematics and Artificial Intelligence 43, 343–365.

[80] R Core Team, 2014. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria. URL: `http://www.R-project.org/`.

[81] Rice, J.R., 1976. The algorithm selection problem. Advances in Computers 15, 65–118.

[82] van Rijn, J.N., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., Winter, P., Wiswedel, B., Berthold, M.R., Vanschoren, J., 2013. OpenML: A collaborative science platform, in: Machine Learning and Knowledge Discovery in Databases, Springer. pp. 645–649.

[83] Roberts, M., Howe, A.E., 2007. Learned models of performance for many planners, in: ICAPS 2007 Workshop on AI Planning and Learning.

[84] Roussel, O., 2011. Description of ppfolio. Solver description, SAT competition 2011.

[85] Safarpour, S., Mangassarian, H., Veneris, A., Li?ton, M., Sakallah, K., 2007. Improved design debugging using maximum satisability, in: FMCAD, IEEE Computer Society, pp. 13–19.

[86] Samulowitz, H., Memisevic, R., 2007. Learning to solve QBF, in: Proceedings of the 22nd National Conference on Artificial Intelligence, AAAI Press. pp. 255–260.

[87] Silverthorn, B., Miikkulainen, R., 2010. Latent class models for algorithm portfolio methods., in: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, pp. 167–172.

[88] Smith-Miles, K.A., 2008. Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Comput. Surv. 41, 6:1–6:25.

[89] Soininen, T., Niemelä, I., 1999. Developing a declarative rule language for applications in product configuration, in: Gupta, G. (Ed.), Proceedings of the First International Workshop on Practical Aspects of Declarative Languages, Springer. pp. 305–319.

[90] Stahlbock, R., Voß, S., 2008. Operations research at container terminals: a literature update. OR Spectrum 30, 1–52. doi:10.1007/s00291-007-0100-9.

[91] Stergiou, K., 2009. Heuristics for dynamically adapting propagation in constraint satisfaction problems. AI Commun. 22, 125–141.

[92] Streeter, M.J., Golovin, D., Smith, S.F., 2007a. Combining multiple heuristics online, in: Proceedings of the 22nd National Conference on Artificial Intelligence, AAAI Press. pp. 1197–1203.

[93] Streeter, M.J., Golovin, D., Smith, S.F., 2007b. Restart schedules for ensembles of problem instances, in: Proceedings of the 22nd National Conference on Artificial Intelligence, AAAI Press. pp. 1204–1210.

[94] Strickland, D., Barnes, E., Sokol, J., 2005. Optimal protein structure alignment using maximum cliques, in: Oper. Res., pp. 389–402.

[95] Therneau, T., Atkinson, B., Ripley, B., 2014. rpart: Recursive Partitioning and Regression Trees. URL: http://CRAN.R-project.org/package=rpart. r package version 4.1-8.

[96] Thornton, C., Hutter, F., Hoos, H., Leyton-Brown, K., 2013. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 847–855.

[97] Tierney, K., 2014. An Algorithm Selection Benchmark of the Container Pre-Marshalling Problem. Technical Report Working Paper #1402. Decision Support & Optimization Lab, University of Paderborn.

[98] Tierney, K., Pacino, D., Voß, S., 2014. Solving the Pre-Marshalling Problem to Optimality with A* and IDA*. Technical Report Working Paper #1401. Decision Support & Optimization Lab, University of Paderborn.

[99] Vallati, M., Fawcett, C., Gerevini, A., Hoos, H.H., Saetti, A., 2013. Automatic generation of efficient domain-optimized planners from generic parametrized planners, in: International Symposium on Combinatorial Search (SoCS).

[100] Van Gelder, A., 2008. Another look at graph coloring via propositional satisfiability. Discrete Applied Mathematics 156, 230–243.

[101] Vanschoren, J., Blockeel, H., Pfahringer, B., Holmes, G., 2012. Experiment databases. A new way to share, organize and learn from experiments. Machine Learning 87, 127–158.

[102] Vasquez, M., Hao, J.K., 2001. A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite, in: Comput. Optim. Appl., pp. 137–157.

[103] Xu, H., Rutenbar, R., Sakallah, K., 2003. Sub-sat: A formulation for relaxed boolean satisability with applications in routing, in: IEEE Trans. Comput. Aided Des. Integrated Circ. Syst., pp. 814–820.

[104] Xu, L., Hoos, H.H., Leyton-Brown, K., 2007. Hierarchical hardness models for SAT, in: 13th International Conference on Principles and Practice of Constraint Programming, Springer. pp. 696–711.

[105] Xu, L., Hoos, H.H., Leyton-Brown, K., 2010. Hydra: Automatically configuring algorithms for Portfolio-Based selection, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI Press. pp. 210–216.

[106] Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K., 2008. SATzilla: portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research 32, 565–606.

[107] Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K., 2011. Hydra-MIP: automated algorithm configuration and selection for mixed integer programming, in: RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion at the International Joint Conference on Artificial Intelligence (IJCAI).

[108] Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K., 2012a. Evaluating component solver contributions to Portfolio-Based algorithm selectors, in: International Conference on Theory and Applications of Satisfiability Testing, Springer. pp. 228–241.

[109] Xu, L., Hutter, F., Shen, J., Hoos, H.H., Leyton-Brown, K., 2012b. Satzilla2012: Improved algorithm selection based on cost-sensitive classification models. Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions , 57–58.