Metalearning & Algorithm Selection

Pavel Brazdil, Joaquin Vanschoren, Christophe Giraud-Carrier, **Lars Kotthoff**

07 September 2015

4. Algorithm selection and configuration in different domains

Lars Kotthoff
University of British Columbia

Outline

Algorithm selection in other domains

- Problem definition
- Components of algorithm selection systems
- Algorithm selection in specific domains
- Example systems: SATzilla, LLAMA
- System demonstration

The Algorithm Selection Problem

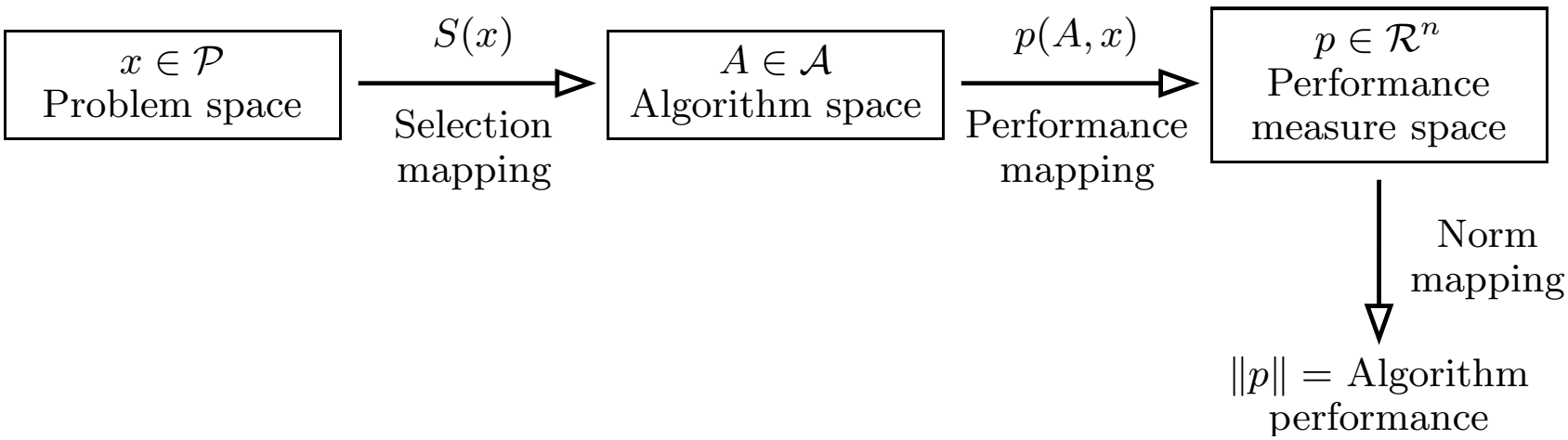
Algorithm Selection

Given a problem, choose the best algorithm to solve it.

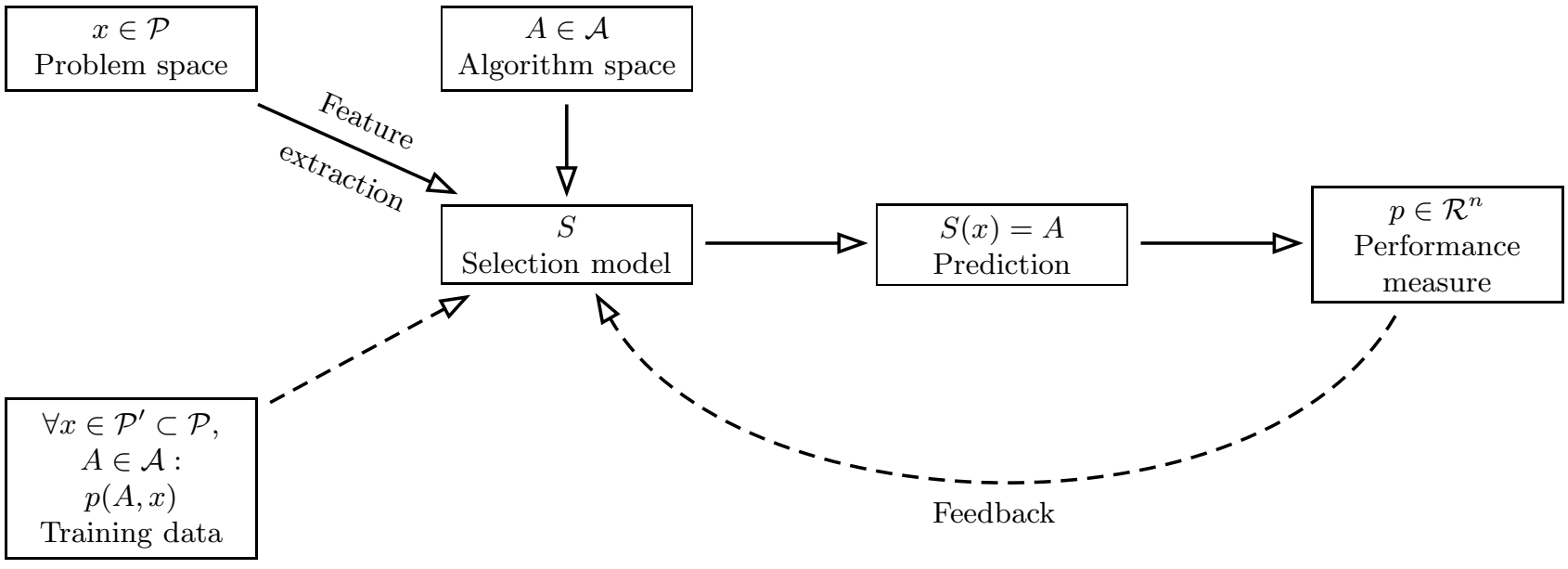
Definition

Definition goes back to Rice (1976):

> *"The objective is to determine $S(x)$ [the mapping of problems to algorithms] so as to have high algorithm performance."*



Contemporary Model



Algorithm Selection Steps

- characterise problem space with features
- learn performance models (usually statistical models induced with machine learning)
- use models to choose best algorithm

Key Components of an Algorithm Selection System

- feature extraction
- performance model
- prediction-based selector

optional:

- presolver
- secondary/hierarchical models and predictors (e.g. for feature extraction time)

Portfolios

- instead of a single algorithm, use several complementary algorithms
- idea from Economics – minimise risk by spreading it out across several securities [Huberman, Lukose, and Hogg (1997)]
- same for computational problems – minimise risk of algorithm performing poorly [Gomes and Selman (2001)]
- in practice often constructed from competition winners

Features

Problem Features

- relate properties of problem instances to performance
- relatively cheap to compute
- specified by domain expert
- syntactic – analyse instance description
- probing – run algorithm for short time (similar to landmarking)

Syntactic Features

- number of variables, number of clauses/constraints/...
- ratios
- order of variables/values
- clause/constraints–variable graph or variable graph:
  - node degrees
  - connectivity
  - clustering coefficient
  - ...
- ...

Probing Features

- number of nodes/propagations within time limit
- estimate of search space size
- tightness of problem/constraints
- ...

Performance Models

Types of Performance Models

- models for individual algorithms
- models for entire portfolios
- models that are somewhere in between

Models for individual Algorithms [e.g. Xu et al. (2008) (SATzilla 2009)]

- predict the performance for each algorithm separately
- standard regression problem
- combine the predictions to choose the best one
- for example: predict the runtime for each algorithm, choose the one with the lowest runtime

advantages:

- ▷ flexibility – no need to relearn when adding/removing algorithms
- ▷ predictions themselves do not need to be very accurate as long as they provide the correct ordering

disadvantages:

- ▷ cost – need separate model for each algorithm
- ▷ cannot take relational knowledge such as “algorithm A is always faster than algorithm B” into account

Models for entire Portfolios [e.g. Gent et al. (2010)]

- ▷ predict the best algorithm in the portfolio
- ▷ standard classification problem
- ▷ alternatively: cluster and assign best algorithms to clusters [e.g. Kadioglu et al. (2010)]

optional (but important):

- ▷ attach a "weight" during learning (e.g. the difference between best and worst solver) to bias model towards the "important" instances
- ▷ special loss metric

advantages:

- ▷ cheap – just a single model to train and run

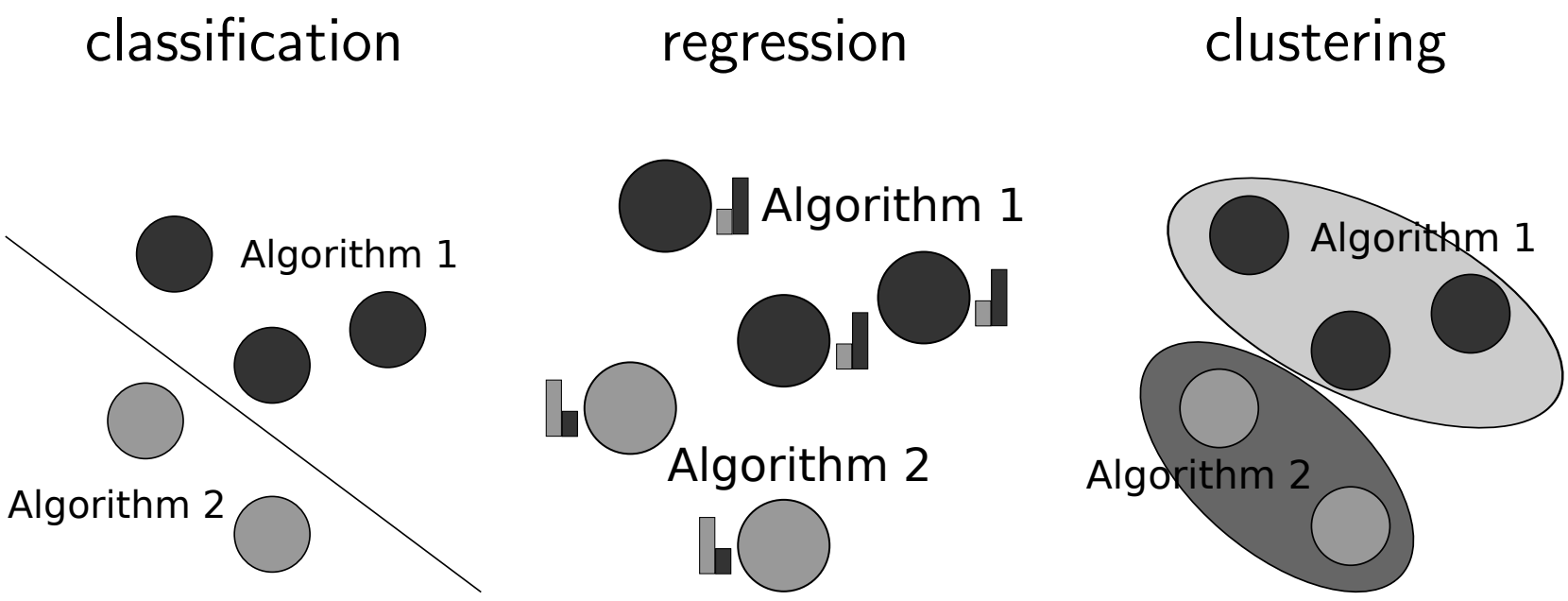
disadvantages:

- ▷ inflexible – need to relearn if portfolio changes

Hybrid Models

- get the best of both worlds
- for example: consider pairs of algorithms to take relations into account [Xu et al. (2011) (SATzilla 2012)]
- for each pair of algorithms, learn model that predicts which one is faster
- or use meta-learning techniques such as stacking [Kotthoff (2012)]

Performance Models Overview



Predictions

Types of Predictions/Algorithm Selectors

- best algorithm [e.g. Gomes and Selman (2001)]
- n best algorithms ranked [e.g. Kanda et al. (2012)]
- allocation of resources to n algorithms [e.g. Kadioglu et al. (2011)]
- change the currently running algorithm? [e.g. Borrett, Tsang, and Walsh (1996)]

Time of Prediction

before problem is being solved [e.g. O'Mahony et al. (2008)]

- ▷ select algorithm(s) once
- ▷ no recourse if predictions are bad

while problem is being solved [e.g. Stergiou (2009)]

- ▷ continuously monitor problem features and/or performance
- ▷ can remedy bad initial choice or react to changing problem

both [e.g. Pulina and Tacchella (2009)]

Types of Machine Learning

Lots!

- depends on the type of prediction to make (e.g. label – classification, number – regression)
- in general, all kinds of machine learning applicable
- good results in practice with methods that use some kind of meta-learning, e.g. random forests

. . . and the rest

Combination with Automatic Configuration

- configure algorithms for different parts of the problem space [Kadioglu et al. (2010) (ISAC)]
- construct algorithm portfolios with complementary configurations [Xu, Hoos, and Leyton-Brown (2010) (Hydra)]
- configure selection model for best performance [Lindauer et al. (2015) (Autofolio)]

Evaluation Measures

- penalized average runtime (PAR) – runtime of selected algorithm, timeout value times factor if timeout
- number of instances solved within timeout
- misclassification penalty – how much additional time needed to solve instances because of wrong predictions
- compare to single best solver and virtual best

Application Domains

- AI Planning [Garbajosa, Rosa, and Fuentetaja (2014)]
- Answer Set Programming [Hoos, Lindauer, and Schaub (2014)]
- Mixed Integer Programming [Xu et al. (2011)]
- Software Design [Simon et al. (2013)]
- Travelling Salesperson [Kotthoff et al. (2015)]

Some concrete examples

SATzilla [Xu et al. (2008)]

- 7 SAT solvers, 4811 problem instances
- syntactic (33) and probing features (15)
- ridge regression to predict log runtime for each solver, choose the solver with the best predicted performance
- later version uses random forests to predict better algorithm for each pair, aggregation through simple voting scheme
- pre-solving, feature computation time prediction, hierarchical model
- won several competitions

Proteus [Hurley et al. (2014)]

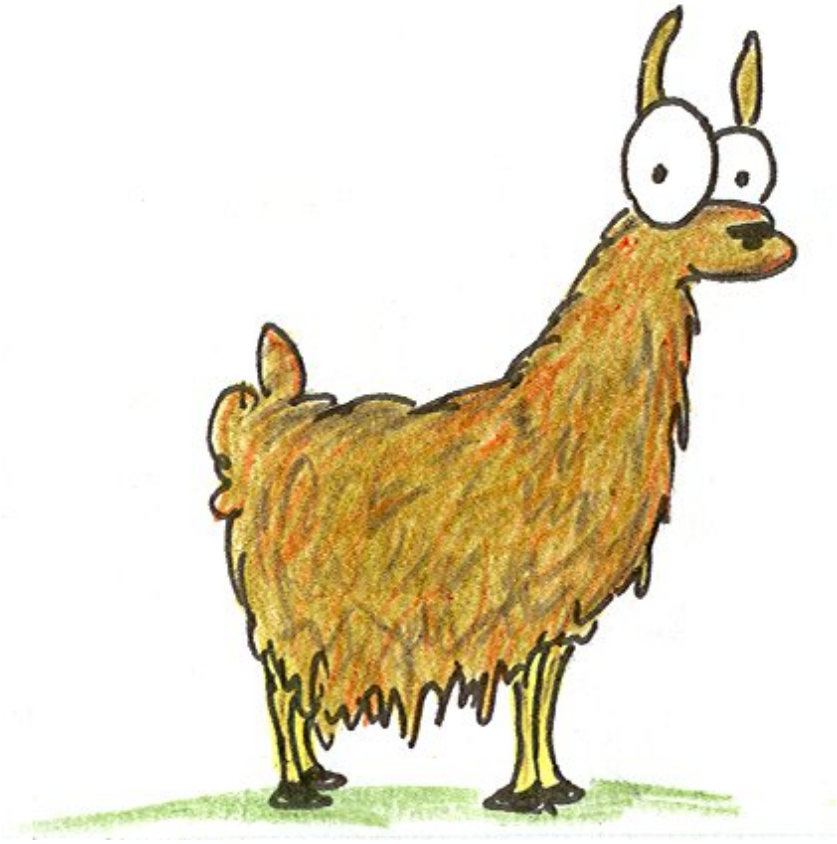
- 4 CP solvers, 3 encodings CP to SAT, 6 SAT solvers, 1493 constraint problems
- 36 CSP features, 54 SAT features
- hierarchical tree of performance models – solve as CP or SAT, if SAT, which encoding, which solver
- considered different types of performance models

Do try this at home

Systems

- SATzilla
  `http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/`
- LLAMA
  `https://bitbucket.org/lkotthoff/llama`
- Claspfolio
  `http://www.cs.uni-potsdam.de/claspfolio/`

LLAMA



LLAMA – Getting Started

```
library(llama)
data(satsolvers)

folds = cvFolds(satsolvers)

learner = makeLearner("classif.JRip")
model = classify(learner, folds)

mean(parscores(folds, model))
> 5611.417
mean(parscores(satsolvers, vbs))
> 4645.169
mean(parscores(satsolvers, singleBest))
> 5779.526
```

LLAMA – Under the Hood

```
model$models[[1]]$learner.model
JRIP rules:
===========

(dyn_log_propags <= 3.169925) and (log_ranges >= 3.321928) and (sqrt_avg_domsiz
(log_values <= 8.366322) and (log_values >= 8.366322) and (dyn_log_nodes <= 6.0
(dyn_log_propags >= 23.653658) and (log_values <= 9.321928) => target=march_rw
(dyn_log_avg_weight <= 5.149405) and (percent_global <= 0.079239) and (sqrt_avg
(percent_global <= 0.048745) and (dyn_log_avg_weight <= 5.931328) and (sqrt_avg
(sqrt_avg_domsize <= 2.475884) and (log_constraints >= 11.920353) and (dyn_log_
(dyn_log_avg_weight <= 4.766713) and (log_constraints >= 8.686501) and (dyn_log
(dyn_log_avg_weight <= 5.422233) and (log_lists <= 6.72792) and (dyn_log_nodes
(percent_global >= 6.1207) and (dyn_log_nodes >= 5.285402) and (percent_avg_con
(log_values >= 10.643856) and (log_bits <= -1) and (dyn_log_avg_weight >= 7.939
(log_lists >= 8.214319) and (percent_global <= 0.022831) and (dyn_log_nodes >=
(log_lists >= 8.214319) and (log_constraints <= 12.005975) => target=cryptomini
(log_constraints <= 7.491853) and (log_values >= 10.643856) and (dyn_log_avg_we
 => target=clasp (1651.0/1161.0)

Number of Rules : 14
```

LLAMA – Other Models

```
rflearner = makeLearner("classif.randomForest")
rfmodel = classify(rflearner, folds)
mean(parscores(folds, rfmodel))
> 5598.554

rplearner = makeLearner("classif.rpart")
rpmodel = classify(rplearner, folds)
mean(parscores(folds, rpmodel))
> 5561.635

# this will take a long time...
rfrlearner = makeLearner("regr.randomForest")
rfrmodel = regression(rfrlearner, folds)
mean(parscores(folds, rfrmodel))
> 5689.075

rfpmodel = classifyPairs(rflearner, folds)
mean(parscores(folds, rfpmodel))
> 5945.246
```

Further Information

- COSEAL group `https://code.google.com/p/coseal/`
- Data format and benchmark library `http://aslib.net`
- Lars Kotthoff. "Algorithm Selection for Combinatorial Search Problems: A Survey". In: *AI Magazine* 35.3 (2014), pp. 48–60
- Kate A. Smith-Miles. "Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection". In: *ACM Comput. Surv.* 41 (Dec. 2008), 6:1–6:25

Further Information

Comments? Suggestions? Corrections?
Let me know!

Algorithm Selection literature summary

Last update 04 August 2015

click headings to sort
click citations to expand

citation	domain	features	predict what	predict how	predict when	portfolio	year
Langley 1983b, Langley 1983a	search	past performance	algorithm	hand-crafted and learned rules	offline and online	dynamic	1983
Carbonell et al. 1991	planning	problem domain features, search statistics	control rules	explanation-based rule construction	online	dynamic	1991
Gratch and DeJong 1992	planning	problem domain features, search statistics	control rules	probabilistic rule construction	online	dynamic	1992
Smith and Setliff 1992	software design	features of abstract representation	algorithms and data structures	simulated annealing	offline	static	1992
Aha 1992	machine learning	instance features	algorithm	learned rules	offline	static	1992
Brodley 1993	machine learning	instance and algorithm features	algorithm	hand-crafted rules	offline	static	1993
Kamel et al. 1993	differential equations	past performance, instance features	algorithm	hand-crafted rules	offline	static	1993
Minton 1993b, Minton 1993a, Minton 1996	constraints	runtime performance	algorithm	hand-crafted and learned rules	offline	dynamic	1993
Cahill 1994	software design	instance features	algorithms and data structures	frame-based knowledge base	offline	static	1994
Tsang et al. 1995	constraints	instance features	-	-	-	static	1995
Brewer 1995	software design	runtime performance	algorithms, data structures and their parameters	statistical model	offline	static	1995
Weerawarana et al. 1996, Joshi et al. 1996	differential equations	instance features	runtime performance	Bayesian belief propagation, neural nets	offline	static	1996
Borrett et al. 1996	constraints	search statistics	switch algorithm?	hand-crafted rules	online	static, static order	1996
Allen and Minton 1996	SAT, constraints	probing	runtime performance	hand-crafted rules	online	static	1996
Sakkout et al. 1996	constraints	search statistics	switch algorithm?	hand-crafted rules	online	static	1996
Huberman et al. 1997	graph colouring	past performance	resource allocation	statistical model	offline	static	1997
Gomes and Selman 1997b, Gomes and Selman 1997a	constraints	problem size and past performance	algorithm	statistical model	offline	static	1997
Cook and Varnell 1997	parallel search	probing	set of search strategies	decision trees, Bayesian classifier, nearest neighbour, neural net	online	static	1997
Fink 1997, Fink 1998	planning	past performance	resource allocation	statistical model, regression	offline	static	1997
Lobjois and Lemaître 1998	branch and bound	probing	runtime performance	hand-crafted rules	online	static	1998
Caseau et al. 1999	vehicle routing problem	runtime performance	algorithm	genetic algorithms	offline	static	1999
Howe et al. 1999	planning	instance features	resource allocation	linear regression	offline	static	1999
Terashima-Marín et al. 1999	scheduling	instance and search features	algorithm	genetic algorithms	offline	dynamic	1999
Wilson et al. 2000	software design	instance features	data structures	nearest neighbour	offline	static	2000
Beck and Fox 2000	job shop scheduling	instance feature changes during search	algorithm scheduling policy	hand-crafted rules	online	static	2000
Brazdil and Soares 2000	classification	past performance	ranking	distribution model	offline	static	2000

https://larskotthoff.github.io/assurvey/

This just in!

ICON challenge on Algorithm Selection

- first challenge on algorithm selection just finished
- description and results at `http://challenge.icon-fet.eu/`

References

James E. Borrett, Edward P. K. Tsang, and Natasha R. Walsh. "Adaptive Constraint Satisfaction: The Quickest First Principle". In: *ECAI*. 1996, pp. 160–164.

Alberto Garbajosa, Toms de la Rosa, and Raquel Fuentetaja. "Planning with ensembles of classifiers". In: *ECAI*. 2014, pp. 1007–1008.

Ian P. Gent et al. "Learning When to Use Lazy Learning in Constraint Solving". In: *19th European Conference on Artificial Intelligence*. Aug. 2010, pp. 873–878.

Carla P. Gomes and Bart Selman. "Algorithm Portfolios". In: *Artificial Intelligence* 126.1-2 (2001), pp. 43–62.

References


Holger Hoos, Marius Lindauer, and Torsten Schaub. "claspfolio 2: Advances in Algorithm Selection for Answer Set Programming". In: *TPLP* 14.4-5 (2014), pp. 569–585.


Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. "An Economics Approach to Hard Computational Problems". In: *Science* 275.5296 (1997), pp. 51–54.


Barry Hurley et al. "Proteus: A Hierarchical Portfolio of Solvers and Transformations". In: *CPAIOR*. May 2014.


Serdar Kadioglu et al. "Algorithm Selection and Scheduling". In: *17th International Conference on Principles and Practice of Constraint Programming*. 2011, pp. 454–469.

References

Serdar Kadioglu et al. "ISAC Instance-Specific Algorithm Configuration". In: *Proceeding of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010, pp. 751–756.

Jorge Kanda et al. "A Meta-Learning Approach to Select Meta-Heuristics for the Traveling Salesman Problem Using MLP-Based Label Ranking". In: *19th International Conference on Neural Information Processing*. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 488–495.

Lars Kotthoff. "Algorithm Selection for Combinatorial Search Problems: A Survey". In: *AI Magazine* 35.3 (2014), pp. 48–60.

References


Lars Kotthoff. "Hybrid Regression-Classification Models for Algorithm Selection". In: *20th European Conference on Artificial Intelligence*. Aug. 2012, pp. 480–485.


Lars Kotthoff et al. "Improving the State of the Art in Inexact TSP Solving using Per-Instance Algorithm Selection". In: *LION 9*. 2015.


Kevin Leyton-Brown et al. "A Portfolio Approach to Algorithm Selection". In: *In IJCAI-03*. 2003, pp. 1542–1543.


Marius Lindauer et al. "AutoFolio: Algorithm Configuration for Algorithm Selection". In: *Proceedings of the Twenty-Ninth AAAI Workshops on Artificial Intelligence*. Jan. 2015.

References


Eoin O'Mahony et al. "Using Case-based Reasoning in an Algorithm Portfolio for Constraint Solving". In: *Proceedings of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*. Jan. 2008.


Luca Pulina and Armando Tacchella. "A self-adaptive multi-engine solver for quantified Boolean formulas". In: *Constraints* 14.1 (2009), pp. 80–116.


John R. Rice. "The Algorithm Selection Problem". In: *Advances in Computers* 15 (1976), pp. 65–118.


Douglas Simon et al. "Automatic Construction of Inlining Heuristics Using Machine Learning". In: *IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. CGO '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 1–12.

References


Kate A. Smith-Miles. "Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection". In: *ACM Comput. Surv.* 41 (Dec. 2008), 6:1–6:25.


Kostas Stergiou. "Heuristics for Dynamically Adapting Propagation in Constraint Satisfaction Problems". In: *AI Commun.* 22.3 (2009), pp. 125–141.


Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. "Hydra: Automatically Configuring Algorithms for Portfolio-Based Selection". In: *Twenty-Fourth Conference of the Association for the Advancement of Artificial Intelligence*. 2010, pp. 210–216.


Lin Xu et al. "Evaluating Component Solver Contributions to Portfolio-Based Algorithm Selectors". In: *International Conference on Theory and Applications of Satisfiability Testing*. 2012, pp. 228–241.

References

Lin Xu et al. “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*. 2011, pp. 16–30.

Lin Xu et al. “SATzilla: Portfolio-based Algorithm Selection for SAT”. In: *J. Artif. Intell. Res.* 32 (2008), pp. 565–606.