Algorithm Selection and Portfolios
(and Algorithm Configuration)

Lars Kotthoff

University of British Columbia
larsko@cs.ubc.ca

ACP Summer School, Cork, June 20-24 2016

Outline

- ▷ Motivation
- ▷ Algorithm Configuration
- ▷ Algorithm Configuration Exercises (after break)
- ▷ Algorithm Selection and Portfolios (tomorrow morning)

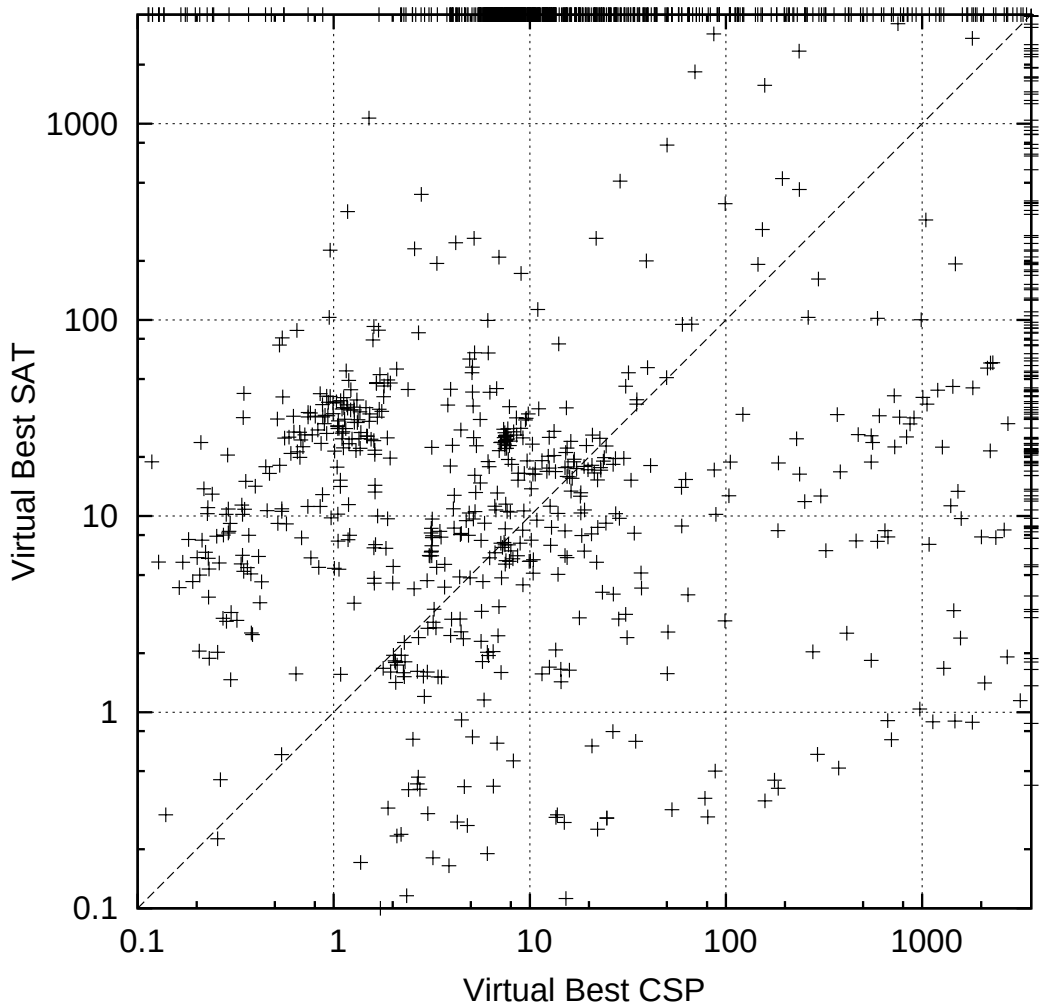
Big Picture

- advance the state of the art through meta-algorithmic techniques
- rather than inventing new things, use existing things better

Prominent Application

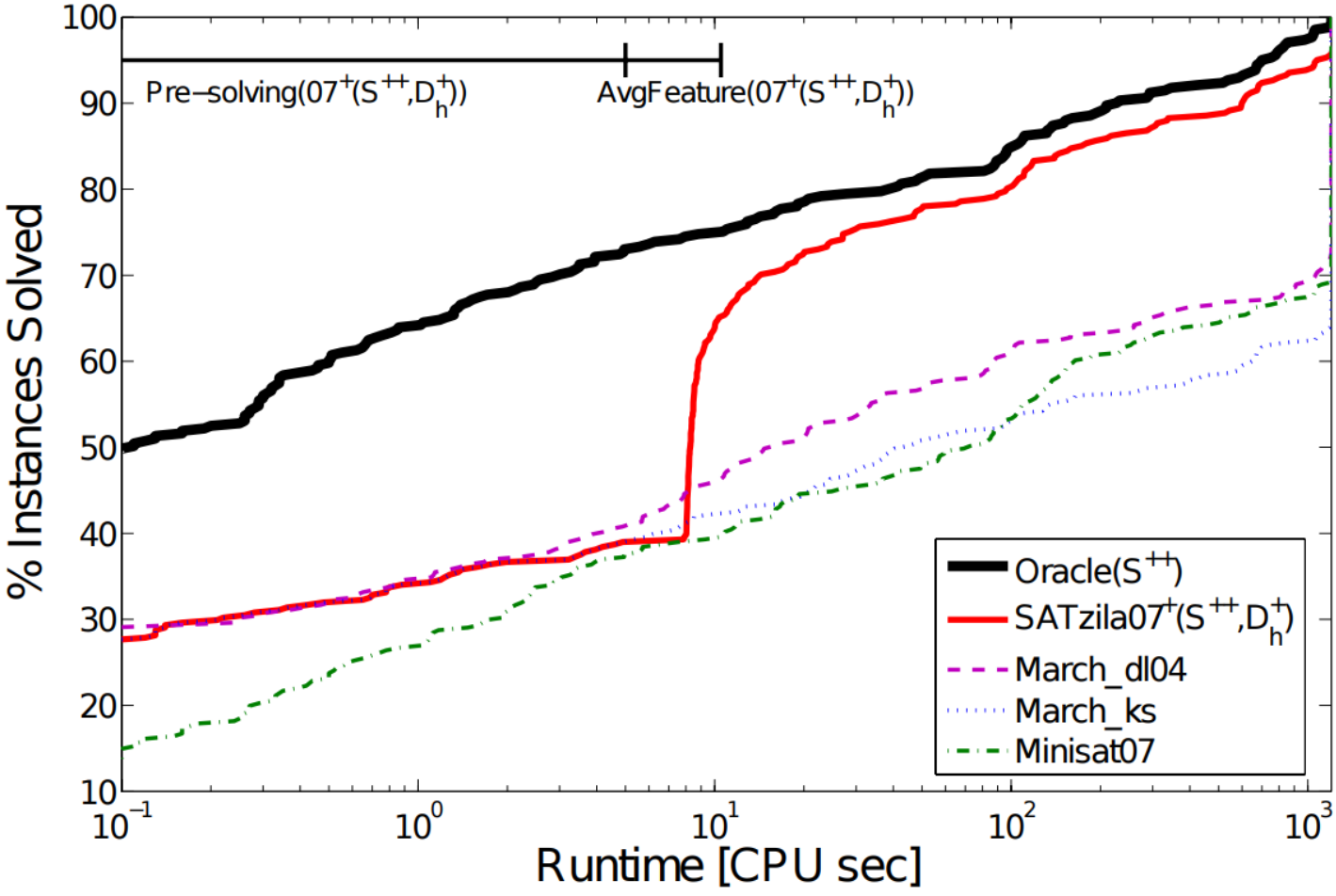
Fréchette, Alexandre, Neil Newman, Kevin Leyton-Brown. "Solving the Station Packing Problem." In Association for the Advancement of Artificial Intelligence (AAAI), 2016.

Performance Differences



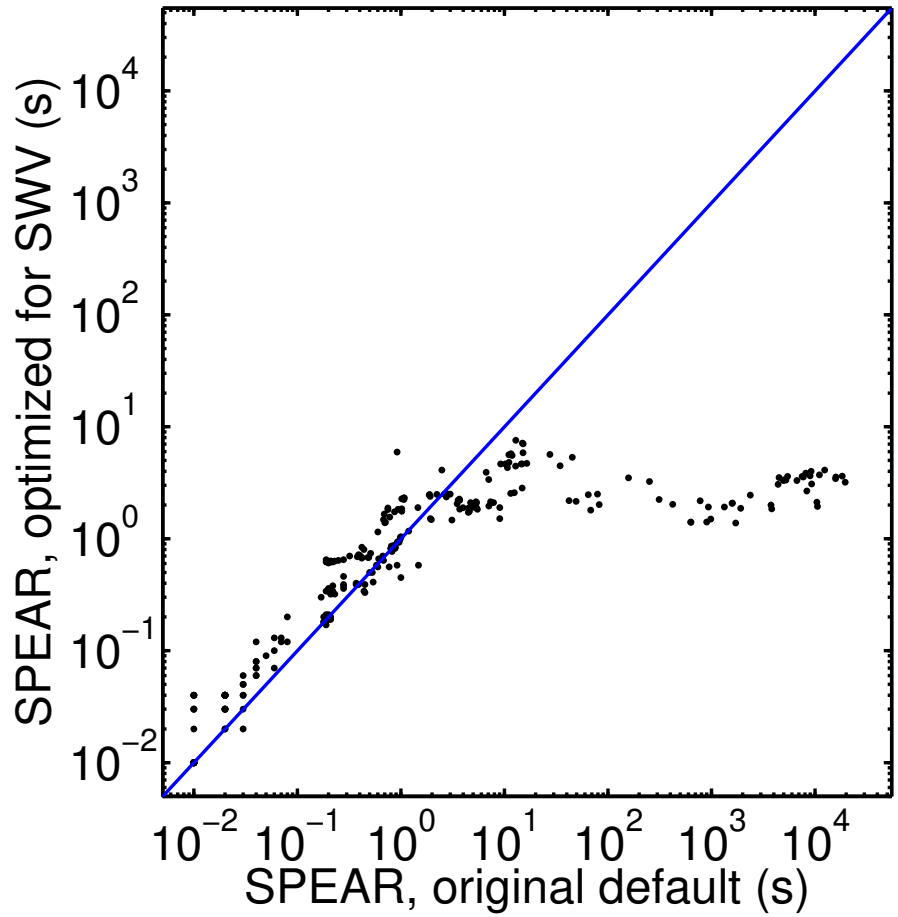
Hurley, Barry, Lars Kotthoff, Yuri Malitsky, and Barry O'Sullivan. "Proteus: A Hierarchical Portfolio of Solvers and Transformations." In CPAIOR, 2014.

Leveraging the Differences



Xu, Lin, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "SATzilla: Portfolio-Based Algorithm Selection for SAT." J. Artif. Intell. Res. (JAIR) 32 (2008): 565–606.

Performance Improvements



Hutter, Frank, Domagoj Babic, Holger H. Hoos, and Alan J. Hu. "Boosting Verification by Automatic Tuning of Decision Procedures." In FMCAD '07: Proceedings of the Formal Methods in Computer Aided Design, 27–34. Washington, DC, USA: IEEE Computer Society, 2007.

Algorithm Configuration

Algorithm Configuration

Given a (set of) problem(s), find the best parameter configuration.

Parameter Configurations?

- anything you can change
- e.g. search heuristic, variable ordering, type of global constraint decomposition
- some will affect performance, others will have no effect at all

Examples

- Spear SAT solver, 26 parameters
- CPLEX MIP solver, 76 parameters
- WEKA machine learning package, 3 feature search methods, 8 feature evaluators, 39 classifiers (of which 12 can be combined with other classifiers)...

Automated Algorithm Configuration

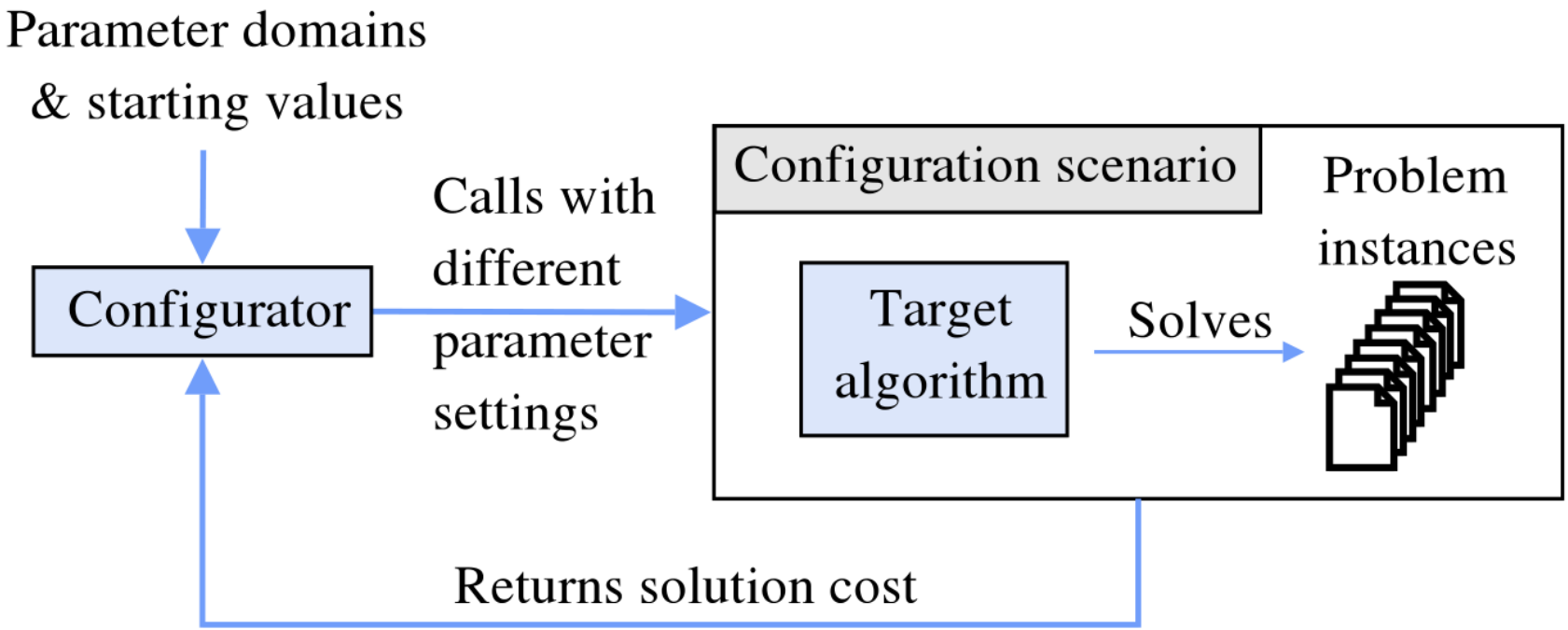
- ▷ no background knowledge on parameters
- ▷ algorithm treated as a "black box"
- ▷ as little manual intervention as possible

Motivation

"Unlike our human subjects, [the system] experimented with a wide variety of combinations of heuristics. Our human subjects rarely had the inclination or patience to try many alternatives, and on at least one occasion incorrectly evaluated alternatives that they did try [...]."

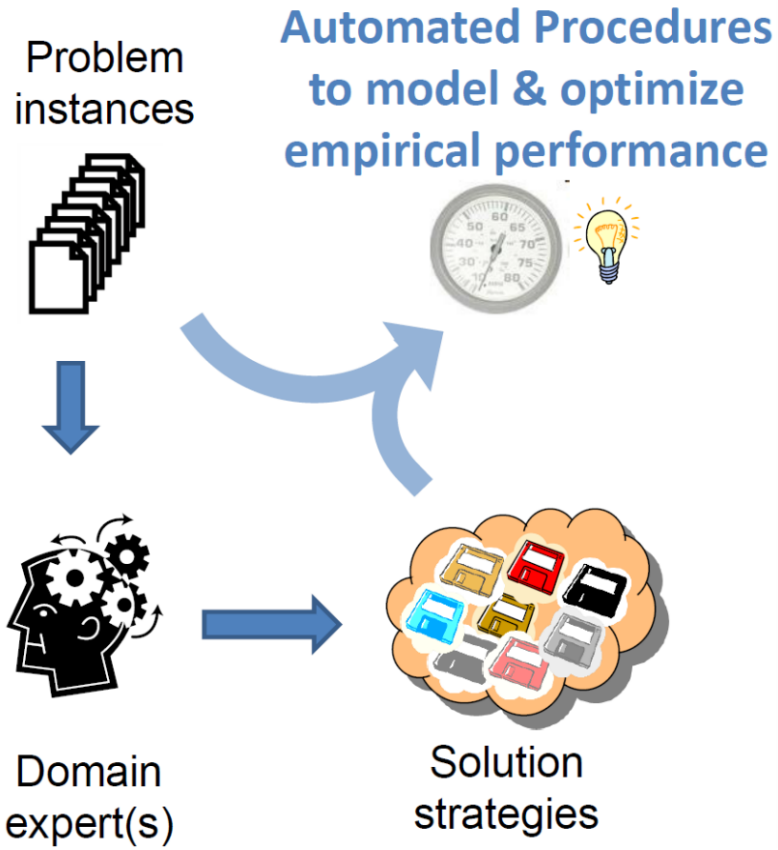
Minton, Steven. "Automatically Configuring Constraint Satisfaction Programs: A Case Study." Constraints 1 (1996): 7–43.

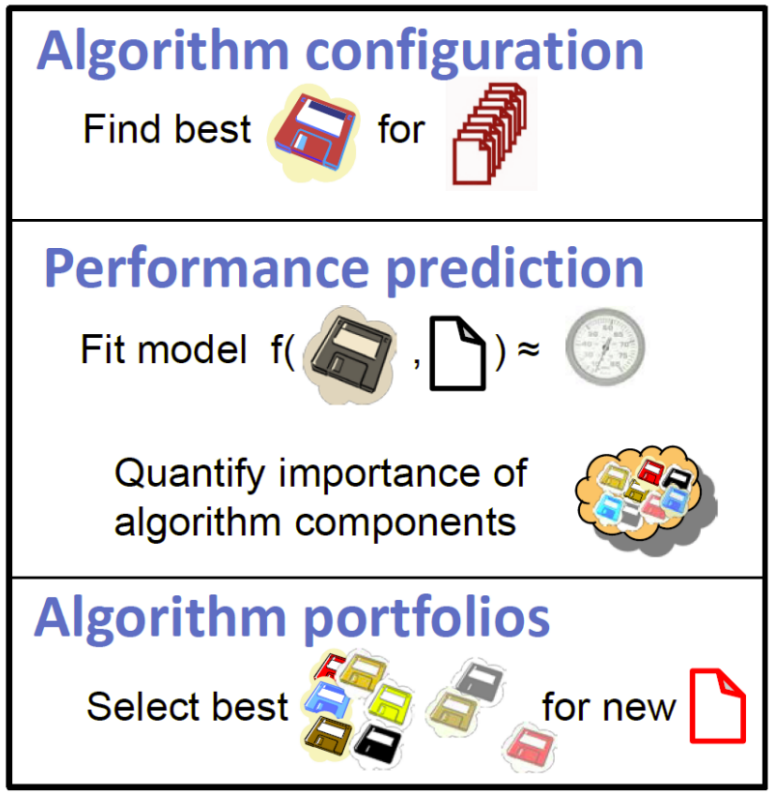
Algorithm Configuration



Frank Hutter and Marius Lindauer, "Algorithm Configuration: A Hands on Tutorial", AAAI 2016

In Context





Frank Hutter and Marius Lindauer, "Algorithm Configuration: A Hands on Tutorial", AAAI 2016

Parameter Spaces

- ▷ numeric – 1, 2, 3...
- ▷ ordinal – a, b, c...
- ▷ categoric – ACP, UBC, UCC...
- ▷ conditional dependencies – e.g. if A is 1, B can't be 2, C is only active if A is >2

$\rightarrow$ not every tool suitable for every type of parameter

General Approach

- evaluate algorithm as black box function
- observe effect of parameters without knowing the inner workings
- balance diversification/exploration and intensification/exploitation

Choosing Instances

- we want configurations that generalise, i.e. good for more than one instance
- similar problem to machine learning – want models that generalise
- split instances into training set (which we configure on) and test set (which we only evaluate performance on)
- need to balance easy/hard instances in both sets

When are we done?

- ▷ most approaches incomplete
- ▷ cannot prove optimality, not guaranteed to find optimal solution (with finite time)
- ▷ performance highly dependent on configuration space

→ How do we know when to stop?

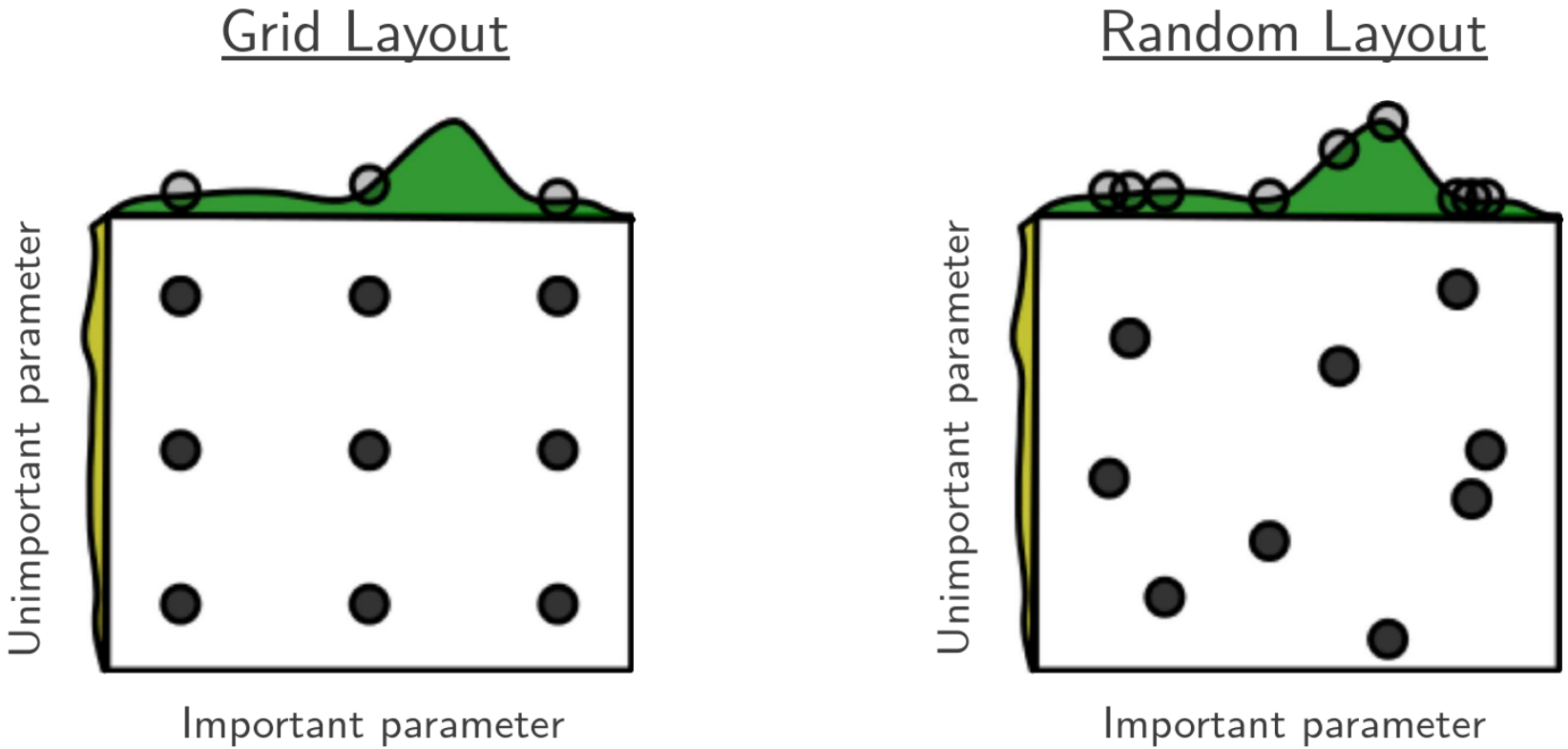
Time Budget

How much time/how many function evaluations?

- ▷ too much $\rightarrow$ wasted resources
- ▷ too little $\rightarrow$ suboptimal result
- ▷ use statistical tests
- ▷ evaluate on parts of the data
- ▷ for runtime: adaptive capping

Grid and Random Search

- evaluate certain points in parameter space



Bergstra, James, and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." J. Mach. Learn. Res. 13, no. 1 (February 2012): 281–305.

Population-Based Methods

- e.g. Racing and Genetic Algorithms
- start with population of random configurations
- eliminate "weak" individuals
- generate new population from "strong" individuals
- iterate

Birattari, Mauro, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. "F-Race and Iterated F-Race: An Overview." In Experimental Methods for the Analysis of Optimization Algorithms, 311–36. 2010.

Ansótegui, Carlos, Meinolf Sellmann, and Kevin Tierney. "A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms." In CP, 142–57, 2009.

Local Search

- start with random configuration
- change a single parameter (local search step)
- if better, keep the change, else revert
- repeat
- optional (but important): restart with new random configurations

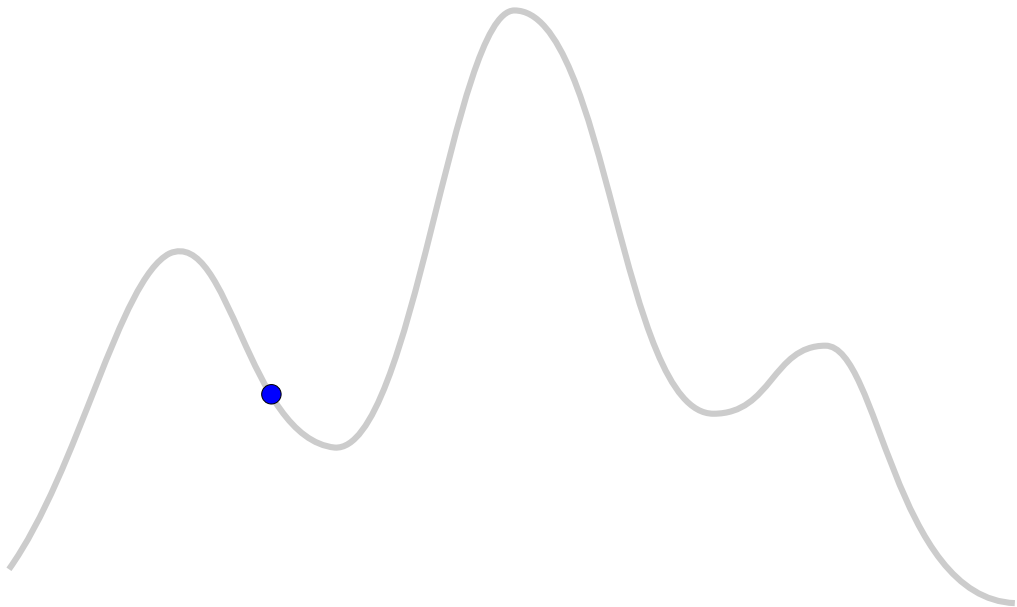
---

Hutter, Frank, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. "ParamILS: An Automatic Algorithm Configuration Framework." J. Artif. Int. Res. 36, no. 1 (2009): 267–306.

Local Search Example

graphics by Holger Hoos

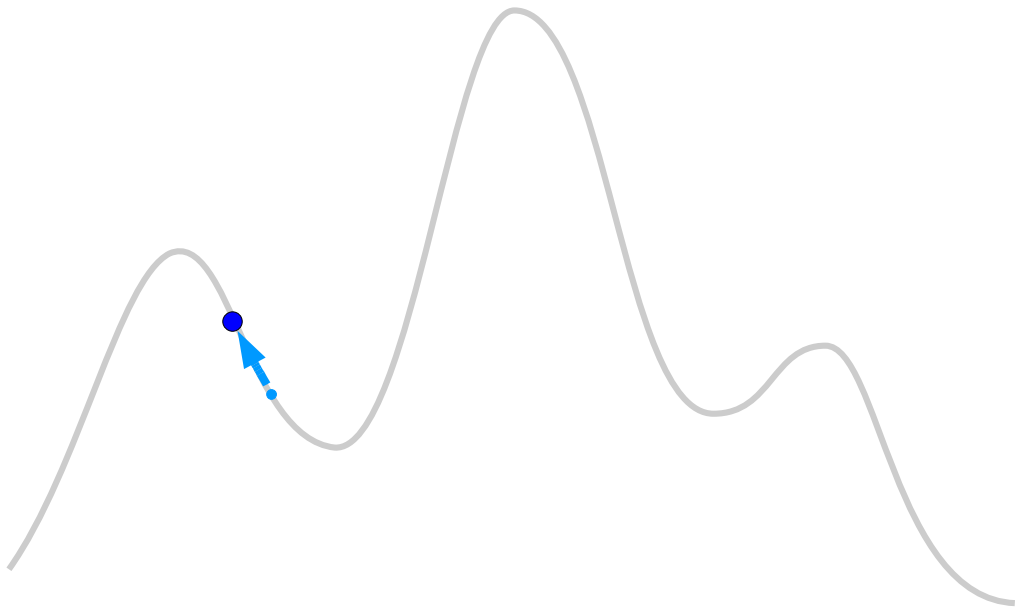
Local Search Example



Initialisation

graphics by Holger Hoos

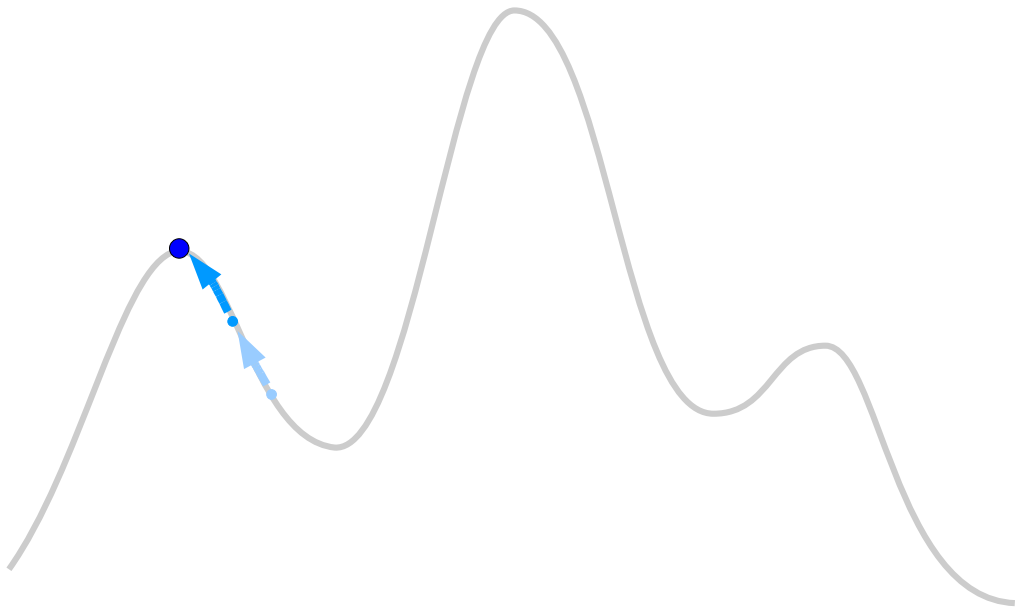
Local Search Example



Local Search

graphics by Holger Hoos

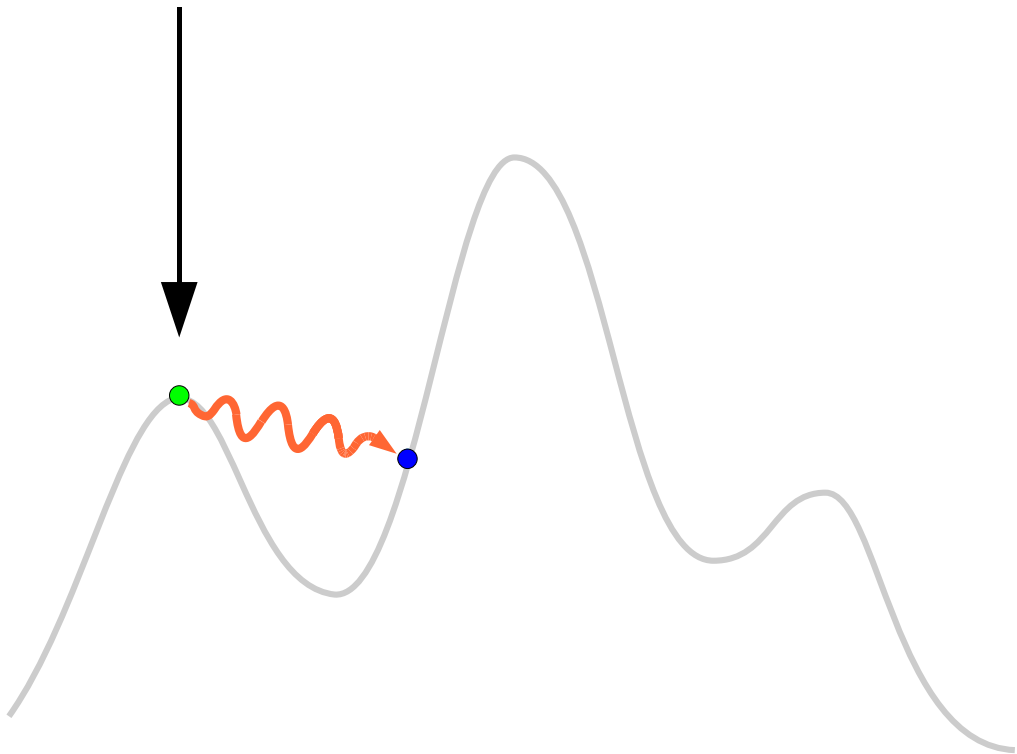
Local Search Example



Local Search

graphics by Holger Hoos

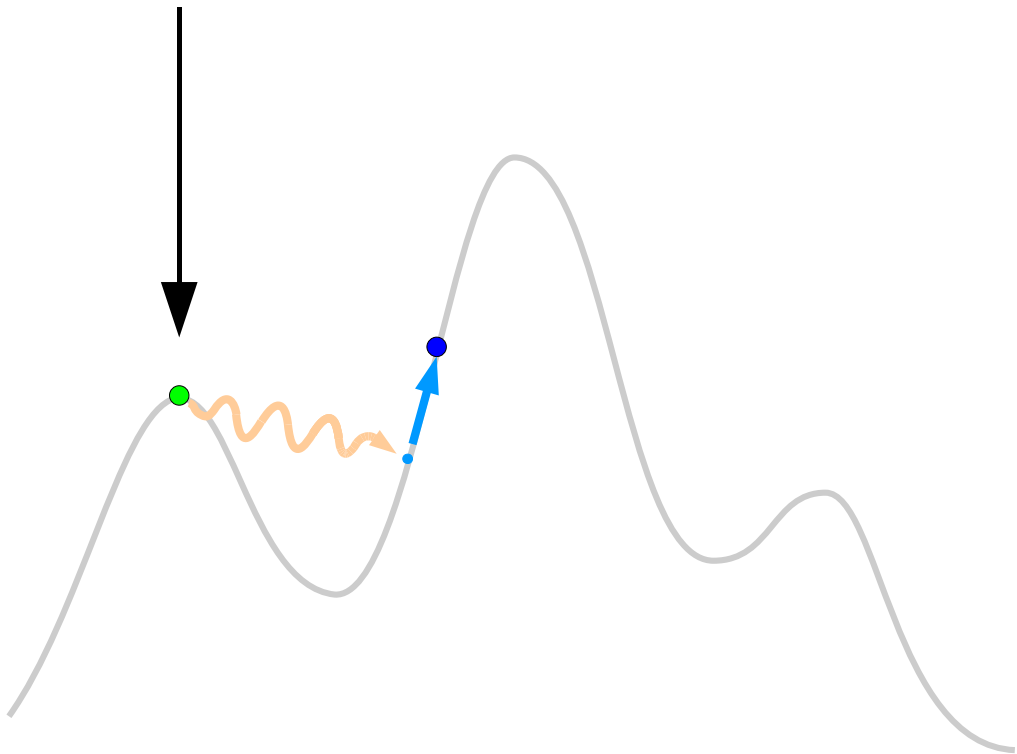
Local Search Example



Perturbation

graphics by Holger Hoos

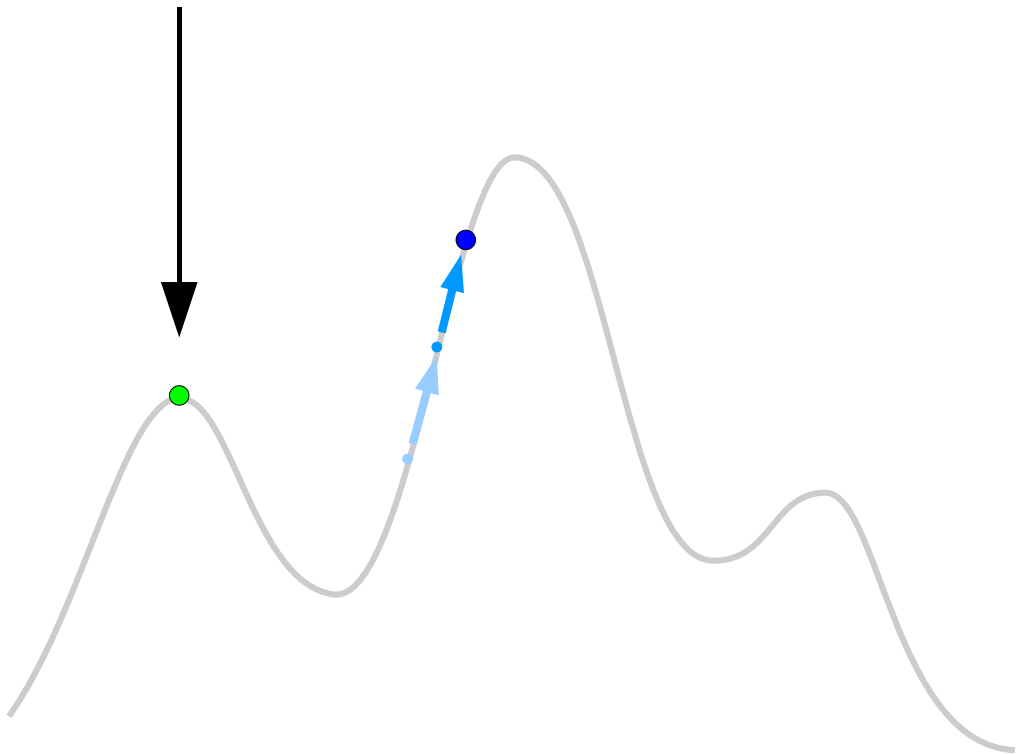
Local Search Example



Local Search

graphics by Holger Hoos

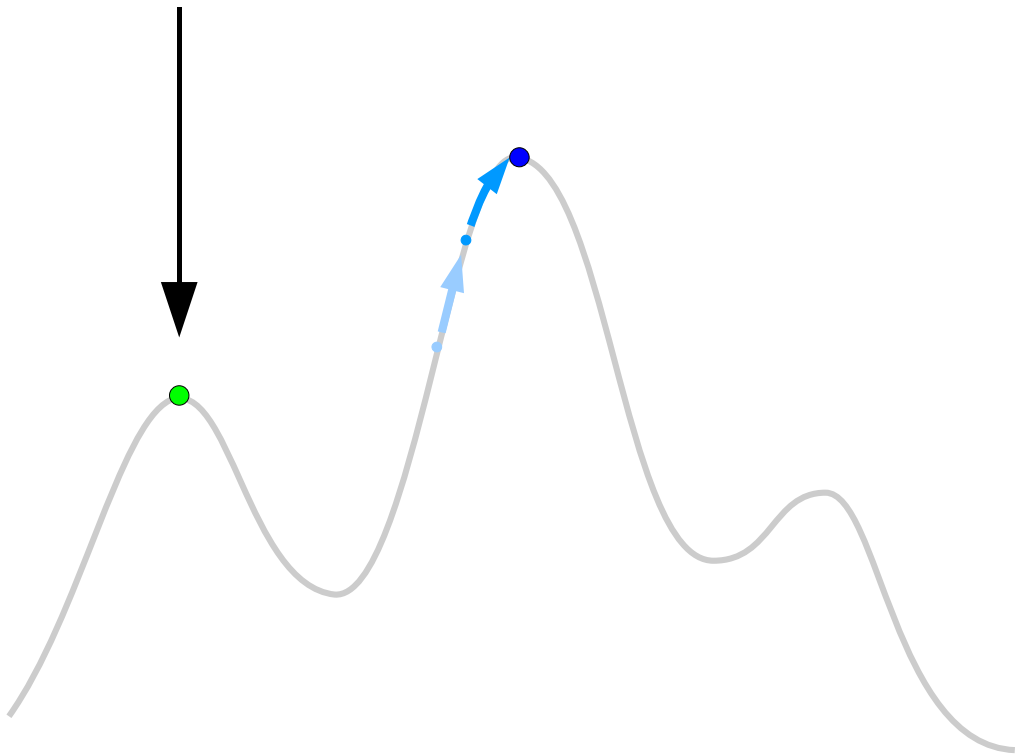
Local Search Example



Local Search

graphics by Holger Hoos

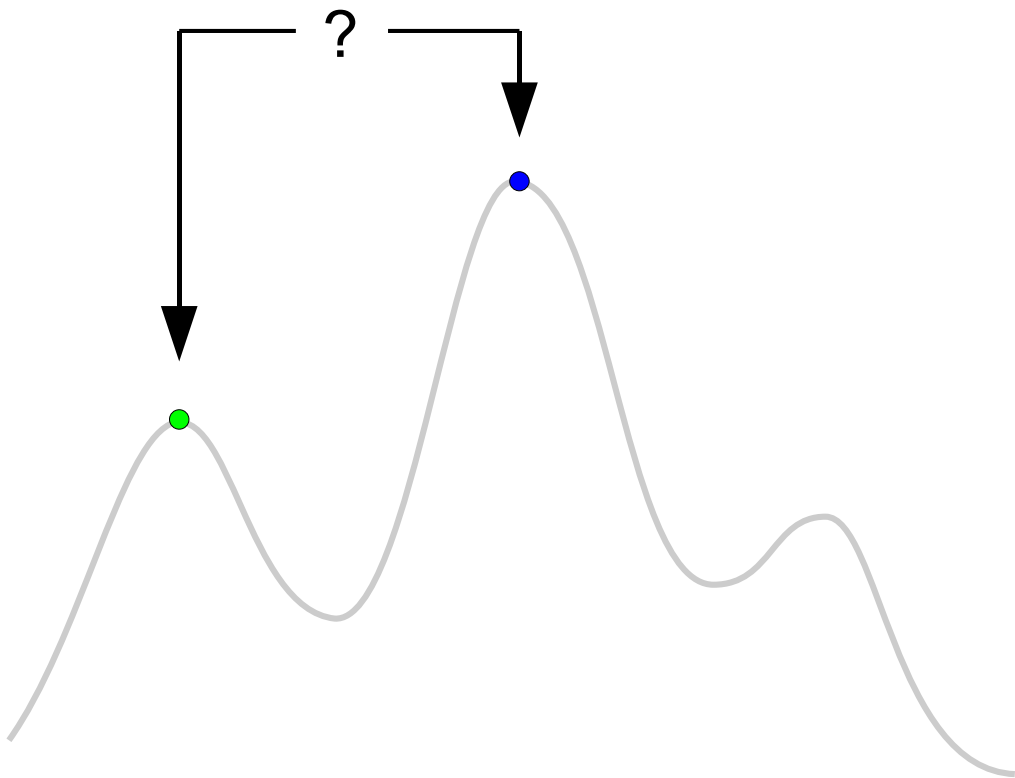
Local Search Example



Local Search

graphics by Holger Hoos

Local Search Example



Selection (using Acceptance Criterion)

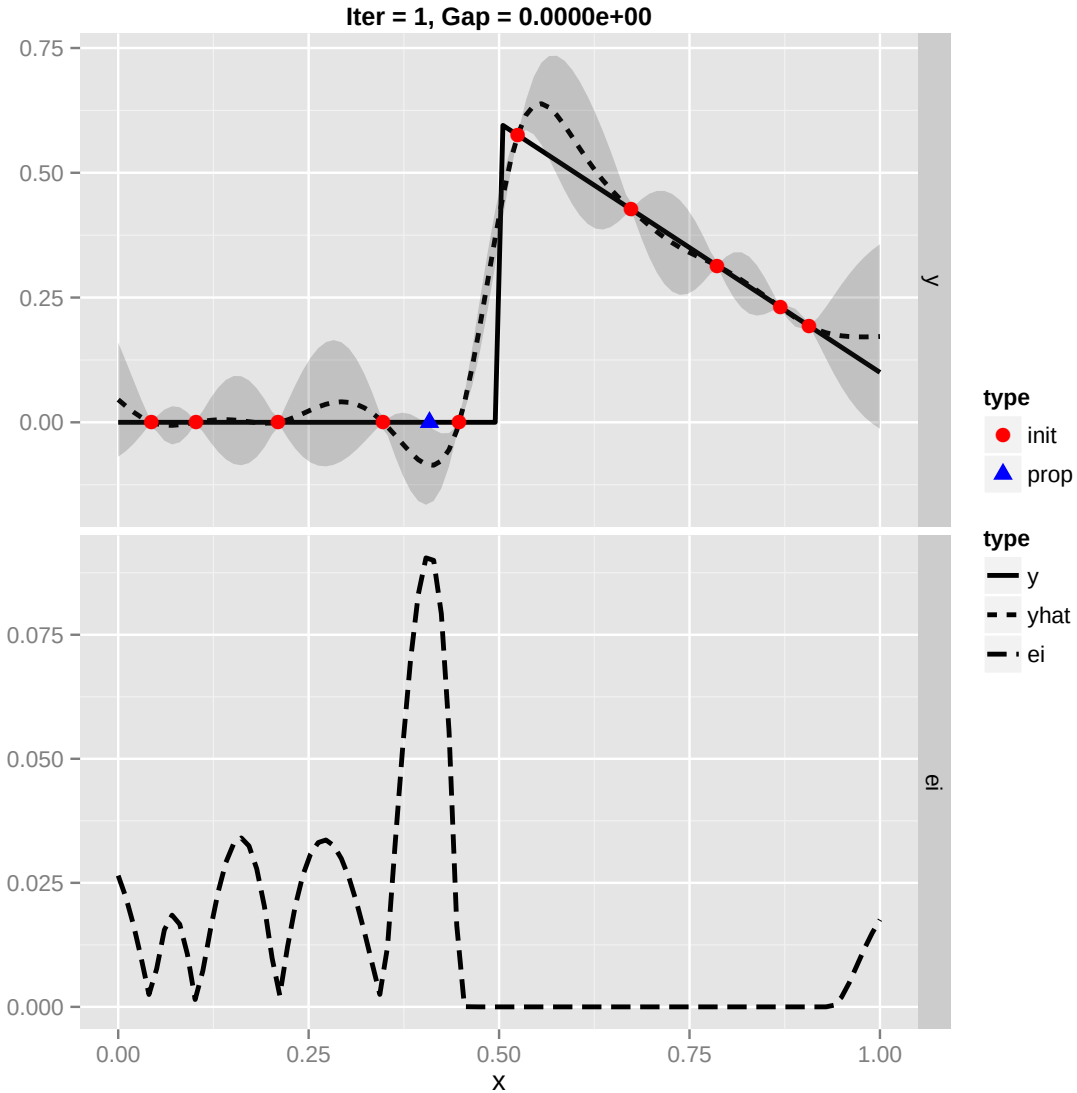
graphics by Holger Hoos

Model-Based Search

- ▷ build model of parameter-response surface
- ▷ allows targeted exploration of new configurations
- ▷ (can take instance features into account like algorithm selection)

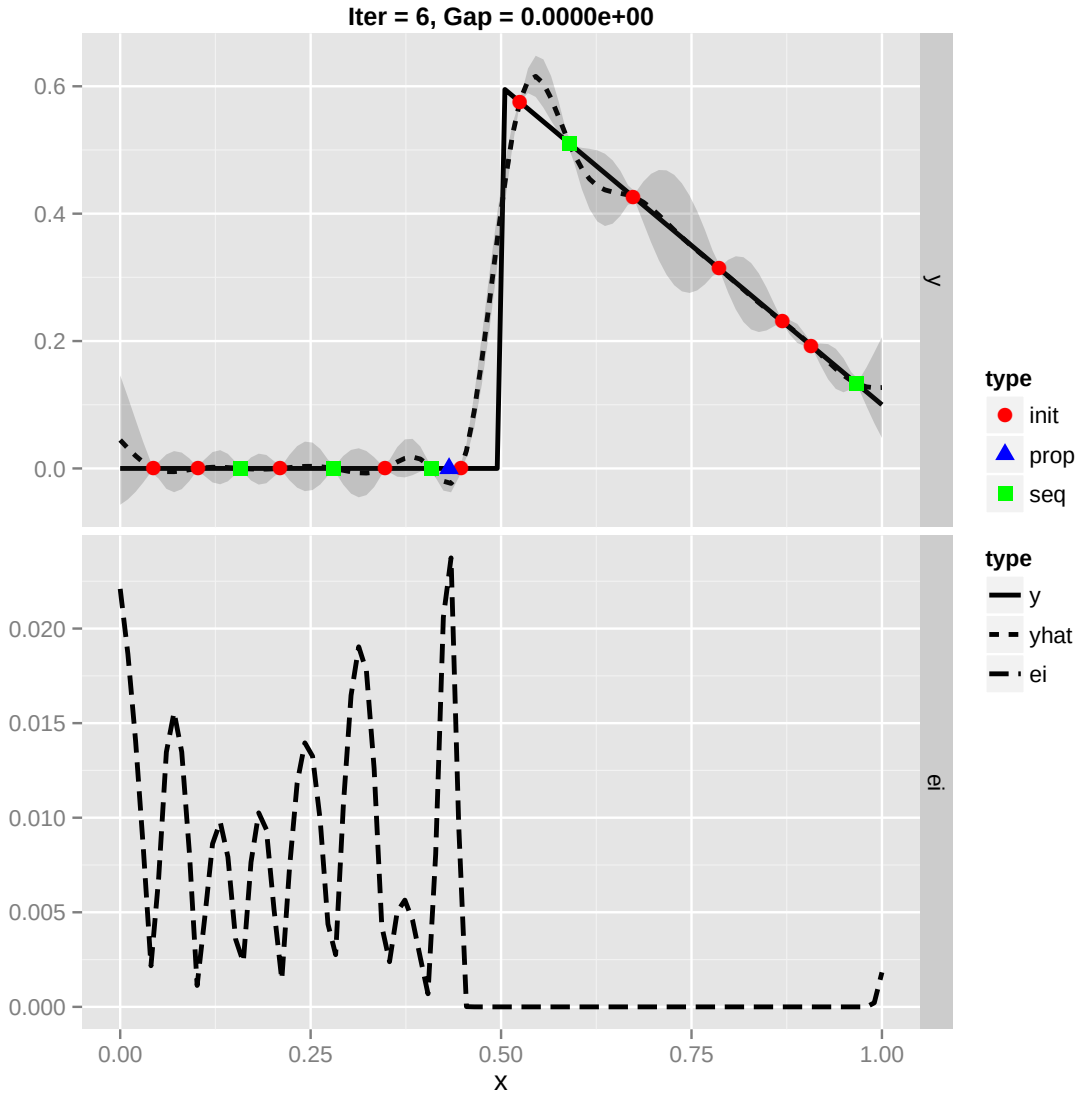
Hutter, Frank, Holger H. Hoos, and Kevin Leyton-Brown. "Sequential Model-Based Optimization for General Algorithm Configuration." In LION 5, 507–23, 2011.

Model-Based Search Example



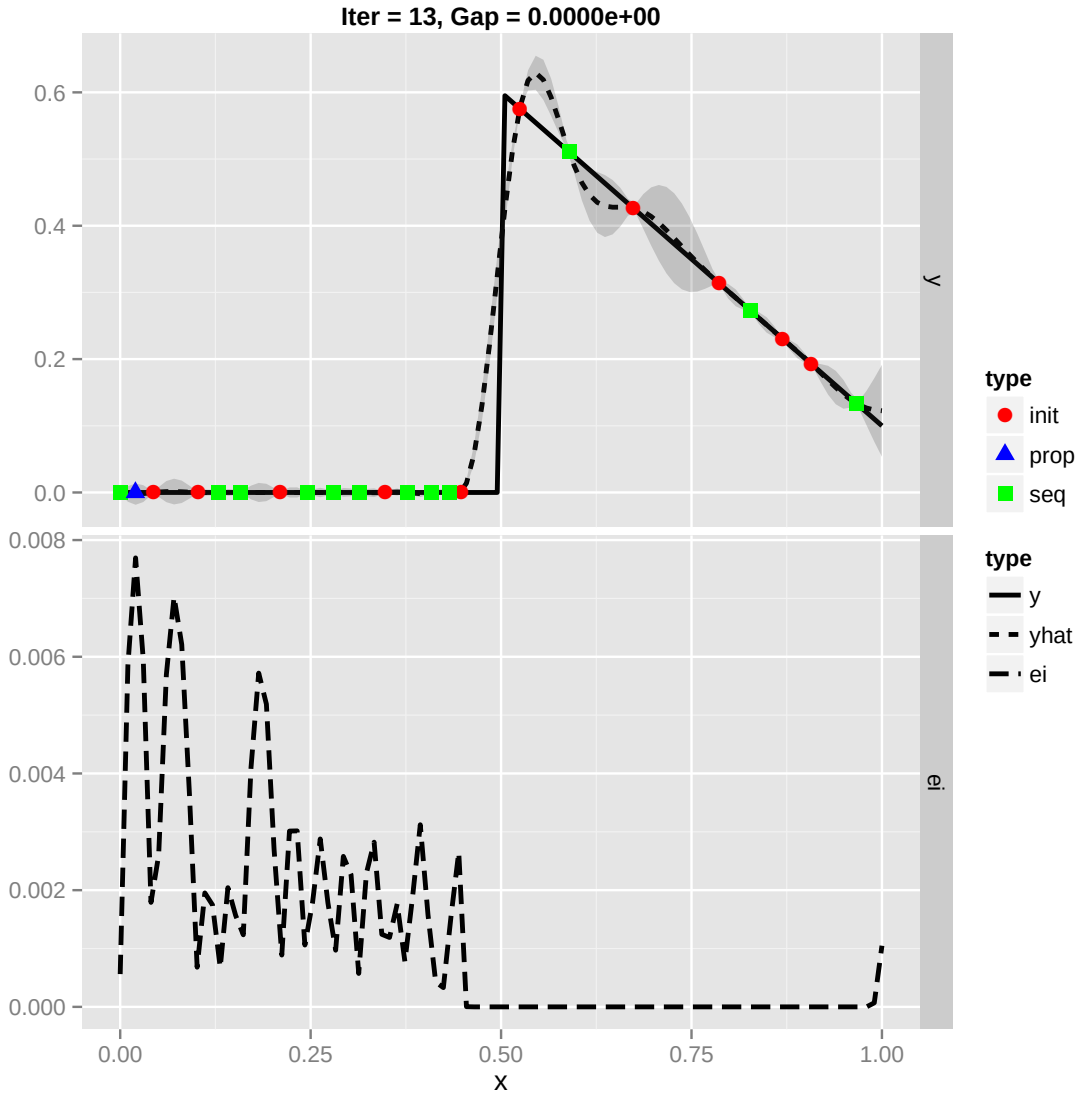
graphics by Bernd Bischl with `mlrMBO` R package

Model-Based Search Example



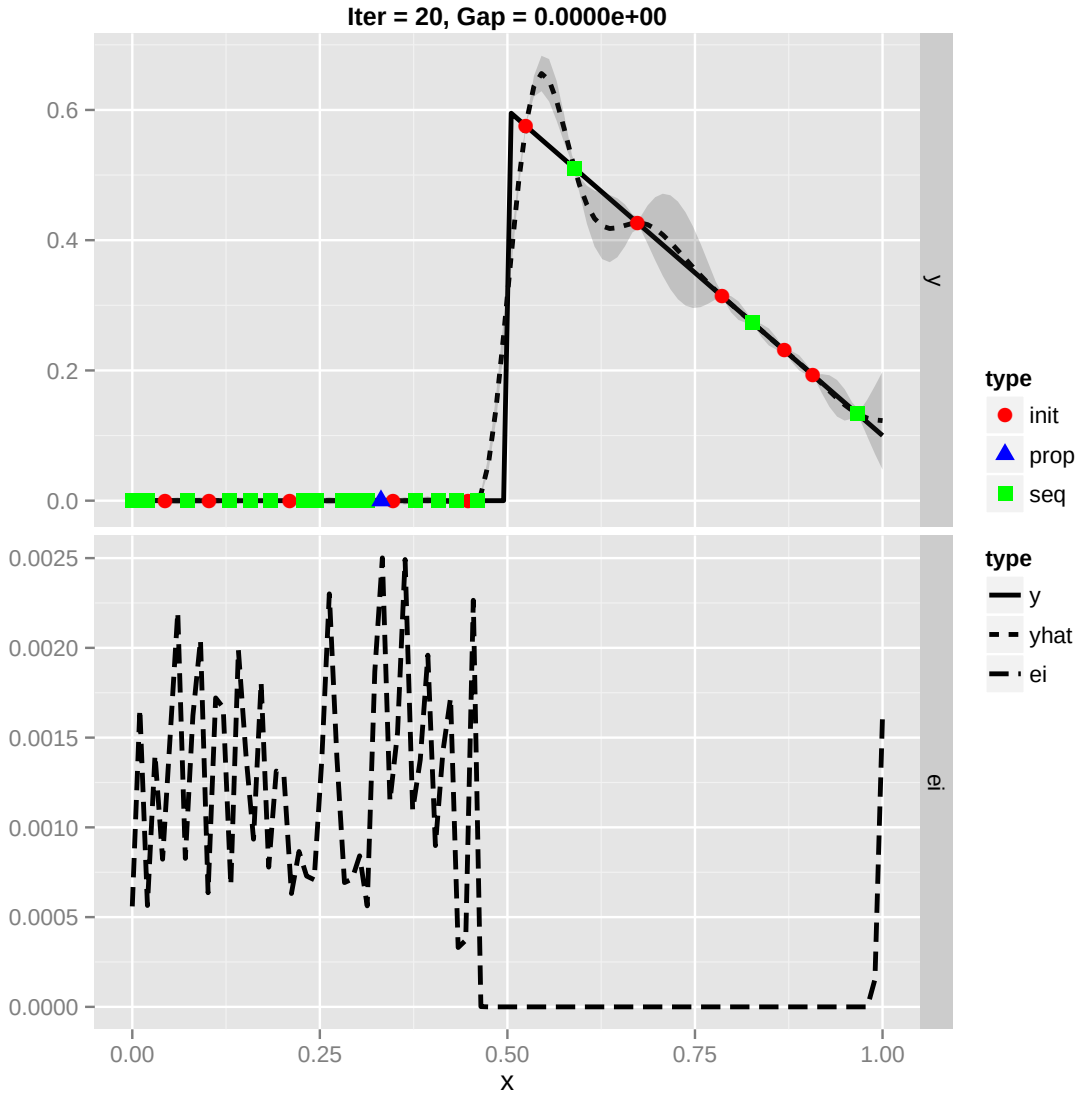
graphics by Bernd Bischl with `mlrMBO` R package

Model-Based Search Example



graphics by Bernd Bischl with `mlrMBO` R package

Model-Based Search Example



graphics by Bernd Bischl with `mlrMBO` R package

Practical Considerations

- poorly-specified parameter spaces
- incorrect results with some configurations
- crashes
- instances to configure on?
- do not configure random seeds!

Overtuning

- similar to overfitting in machine learning
- performance improves on training instances, but not on test instances
- configuration is too “tailored”, e.g. specific to satisfiable instances

Analysis of Results

- ablation analysis
  `http://www.cs.ubc.ca/labs/beta/Projects/Ablation/`
- functional ANOVA `http://www.automl.org/fanova.html`

Tools and Resources

HPOlib	`http://www.automl.org/hpolib.html`
iRace	`http://iridia.ulb.ac.be/irace/`
mlrMBO	`https://github.com/mlr-org/mlrMBO`
SMAC	`http://www.cs.ubc.ca/labs/beta/Projects/SMAC/`
Spearmint	`https://github.com/HIPS/Spearmint`
TPE	`https://jaberg.github.io/hyperopt/`
Auto-WEKA	`http://www.cs.ubc.ca/labs/beta/Projects/autoweka/`
Auto-sklearn	`https://github.com/automl/auto-sklearn`

Configurable SAT Solver Challenge

- `http://aclib.net/cssc2014/`
- idea: make solvers as configurable as possible, let the machine do the rest
- avoids spurious results because of different configurations
- mitigates impact of instance bias

Algorithm Configuration Exercises

- install Numberjack (if you haven't already)
- install SMAC
- get

  `http://www.cs.ubc.ca/~larsko/numberjack-tuning.tar.gz`

params.pcs	parameter space definition
wrapper.py	wrapper that takes parameter definitions from SMAC and outputs result for SMAC
NQueens.py	actual Numberjack model
instances-*	problem instance (sizes) to run on
scenario.txt	SMAC configuration file

- configure!
  - `smac --scenario scenario.txt`

When it's working...

- check correctness in wrapper
- find harder problem instances
- define additional parameters
- anything else you can think of!

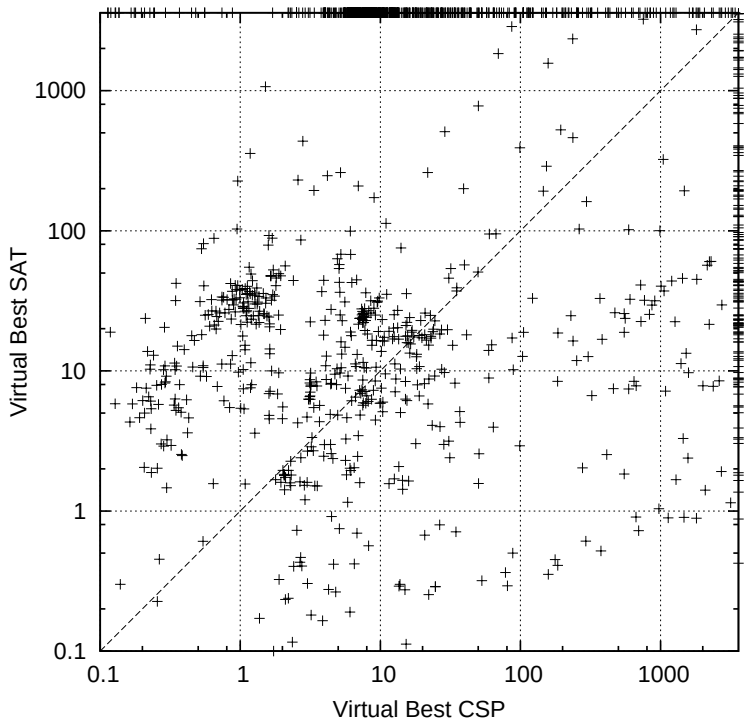
Algorithm Selection

Motivation

Remember overtuning? What if we could leverage this?

Motivation

- ▷ different configurations (algorithms) good on different instances
- ▷ have a *portfolio* of different configurations/algorithms and a selector

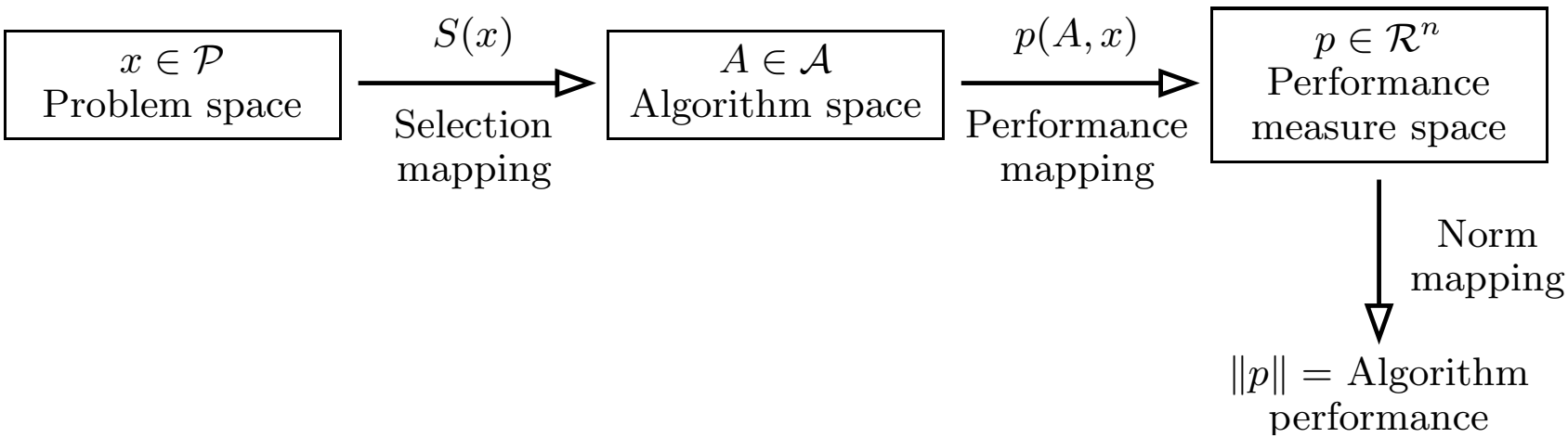


Hurley, Barry, Lars Kotthoff, Yuri Malitsky, and Barry O'Sullivan. "Proteus: A Hierarchical Portfolio of Solvers and Transformations." In CPAIOR, 2014.

Algorithm Selection

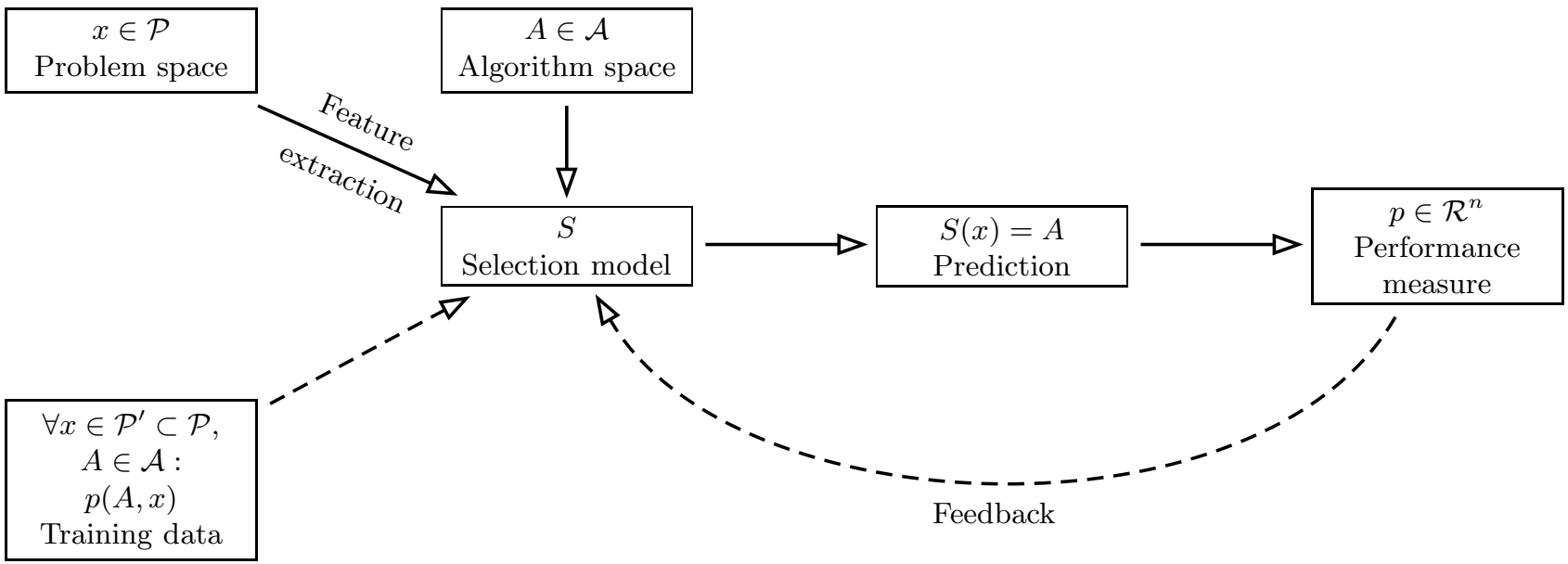
Given a problem, choose the best algorithm to solve it.

Original Model



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

Contemporary Model



Portfolios

- instead of a single algorithm, use several complementary algorithms
- idea from Economics – minimise risk by spreading it out across several securities
- same for computational problems – minimise risk of algorithm performing poorly
- in practice often constructed from competition winners

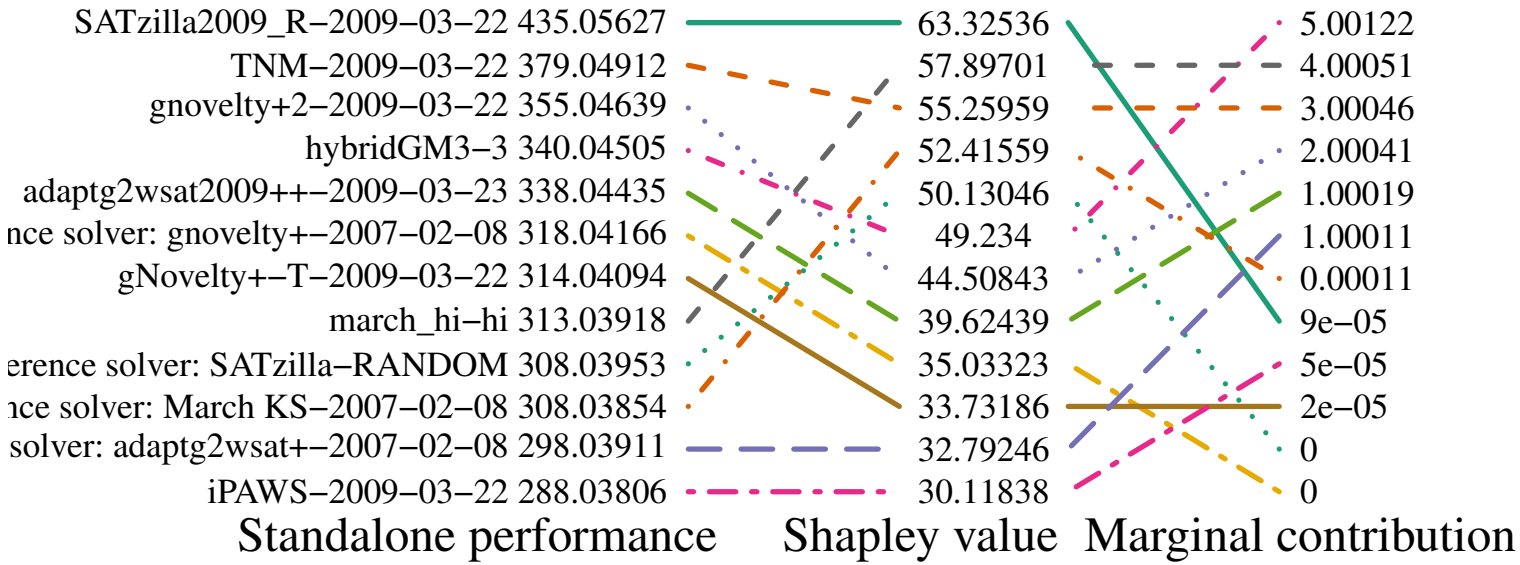
Huberman, Bernardo A., Rajan M. Lukose, and Tad Hogg. "An Economics Approach to Hard Computational Problems." Science 275, no. 5296 (1997): 51–54. doi:10.1126/science.275.5296.51.

Evaluation of Portfolios

- single best algorithm
  - algorithm with the best performance across all instances
  - lower bound for performance of portfolio – hopefully we are better!
- virtual best algorithm
  - choose the best algorithm for each instance
  - corresponds to oracle predictor or overhead-free parallel portfolio
  - upper bound on portfolio performance (assuming robust performance measurements) – note difference to configuration

Contributions of Algorithms to Portfolios

- ▷ stand-alone performance – ignore that there is a portfolio
- ▷ marginal contribution – how much does the algorithm add to the portfolio containing all other algorithms?
- ▷ Shapley value – how much does the algorithm add on average to all possible portfolios?



Fréchette, Alexandre, Lars Kotthoff, Talal Rahwan, Holger H. Hoos, Kevin Leyton-Brown, and Tomasz P. Michalak. "Using the Shapley Value to Analyze Algorithm Portfolios." In 30th AAAI Conference on Artificial Intelligence, 2016.

Parallel Porfolios

Why not simply run all algorithms in parallel?

- not enough resources may be available / waste of resources
- algorithms may be parallelized themselves
- memory contention
- ...

Key Components of an Algorithm Selection System

- ▷ feature extraction
- ▷ performance model
- ▷ prediction-based selector/scheduler

optional:

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

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
- dynamic – instance changes while algorithm is running

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
- ...

Dynamic Features

- change of variable domains
- number of constraint propagations
- number of failures a clause participated in
- ...

What Features do we need in Practice?

- ▷ trade-off between complex features and complex models
- ▷ in practice, very simple features (e.g. problem size) can perform well

Types of Performance Models

- models for entire portfolios
- models for individual algorithms
- models that are somewhere in between (e.g. pairs of algorithms)

Models for Entire Portfolios

- ▷ predict the best algorithm in the portfolio
- ▷ alternatively: cluster and assign best algorithms to clusters

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

Gent, Ian P., Christopher A. Jefferson, Lars Kotthoff, Ian Miguel, Neil Moore, Peter Nightingale, and Karen E. Petrie. "Learning When to Use Lazy Learning in Constraint Solving." In 19th European Conference on Artificial Intelligence, 873–78, 2010.

Kadioglu, Serdar, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. "ISAC – Instance-Specific Algorithm Configuration." In 19th European Conference on Artificial Intelligence, 751–56, 2010.

Models for Individual Algorithms

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

---

Xu, Lin, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "SATzilla: Portfolio-Based Algorithm Selection for SAT." J. Artif. Intell. Res. (JAIR) 32 (2008): 565–606.

Hybrid Models

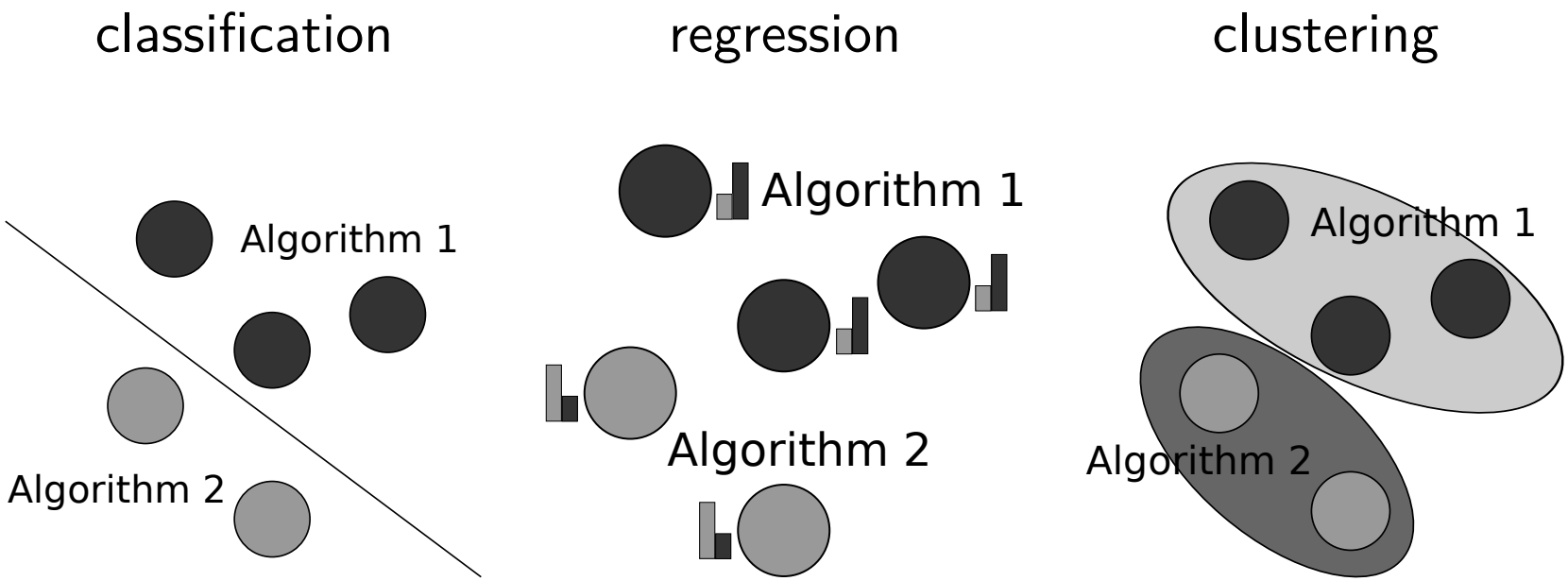
- ▷ get the best of both worlds
- ▷ for example: consider pairs of algorithms to take relations into account
- ▷ for each pair of algorithms, learn model that predicts which one is faster

---

Xu, Lin, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "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), 16–30, 2011.

Kotthoff, Lars. "Hybrid Regression-Classification Models for Algorithm Selection." In 20th European Conference on Artificial Intelligence, 480–85, 2012.

Overview – Machine Learning



Types of Predictions/Algorithm Selectors

- ▷ best algorithm
- ▷ n best algorithms ranked
- ▷ allocation of resources to n algorithms
- ▷ change the currently running algorithm?

Kotthoff, Lars. "Ranking Algorithms by Performance." In LION 8, 2014.

Kadioglu, Serdar, Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. "Algorithm Selection and Scheduling." In 17th International Conference on Principles and Practice of Constraint Programming, 454–69, 2011.

Stergiou, Kostas. "Heuristics for Dynamically Adapting Propagation in Constraint Satisfaction Problems." AI Commun. 22, no. 3 (2009): 125–41.

Time of Prediction

before problem is being solved

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

while problem is being solved

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

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

Example System – SATzilla

- 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

Xu, Lin, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "SATzilla: Portfolio-Based Algorithm Selection for SAT." J. Artif. Intell. Res. (JAIR) 32 (2008): 565–606.

Putting Configuration and Selection together – Hydra

- ▷ find best configuration
- ▷ find configuration that complements existing configuration best
- ▷ iterate
- ▷ stop when no further improvement

Xu, Lin, 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 (AAAI-10), 210–16, 2010.

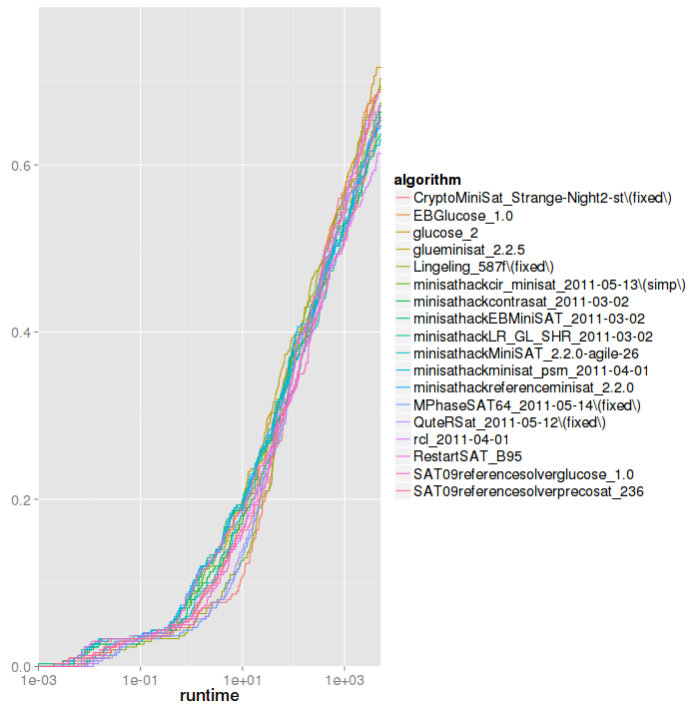
Benchmark library – ASlib

- `https://github.com/coseal/aslib_data`
- currently 17 data sets/scenarios with more in preparation
- SAT, CSP, QBF, ASP, MAXSAT, OR
- includes data used frequently in the literature that you may want to evaluate your approach on
- more scenarios in the pipeline
- `http://aslib.net`

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. "ASlib: A Benchmark Library for Algorithm Selection." To appear in Artificial Intelligence Journal.

EDA – Overview of algorithm performance

- runstatus
- performance plots
- (cumulative) density plots
- scatter plot for pairs of algorithms
- algorithm performance correlation



EDA – Selector performance comparison

- (basic) classification, regression, clustering selectors
- different machine learning techniques
- comparison to virtual best and single best

algo	model	succ	par10	mcp
baseline	vbs	0.937	400.179	0.000
baseline	singleBest	**0.859**	**880.547**	**58.884**
baseline	singleBestByPar	**0.859**	**880.547**	**58.884**
baseline	singleBestBySuccesses	**0.859**	**880.547**	**58.884**
classif	randomForest	0.885	712.222	31.433
classif	ksvm	0.889	691.993	32.039
classif	rpart	**0.895**	**654.968**	**28.429**
cluster	XMeans	**0.888**	**701.844**	**37.686**
regr	randomForest	**0.920**	**509.119**	**16.047**
regr	lm	**0.902**	**615.093**	**26.084**
regr	rpart	**0.901**	**624.923**	31.722

Algorithm Selection Challenge

- `http://challenge.icon-fet.eu/`
- ran on ASlib data
- all submissions and evaluation available

Tools

autofolio	`https://bitbucket.org/mlindauer/autofolio/`
LLAMA	`https://bitbucket.org/lkotthoff/llama`
SATzilla	`http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/`

(Much) More Information

Comments? Suggestions? Corrections?
Let me know!

Algorithm Selection literature summary

Last update 08 December 2015

click headings to sort
click citations to expand

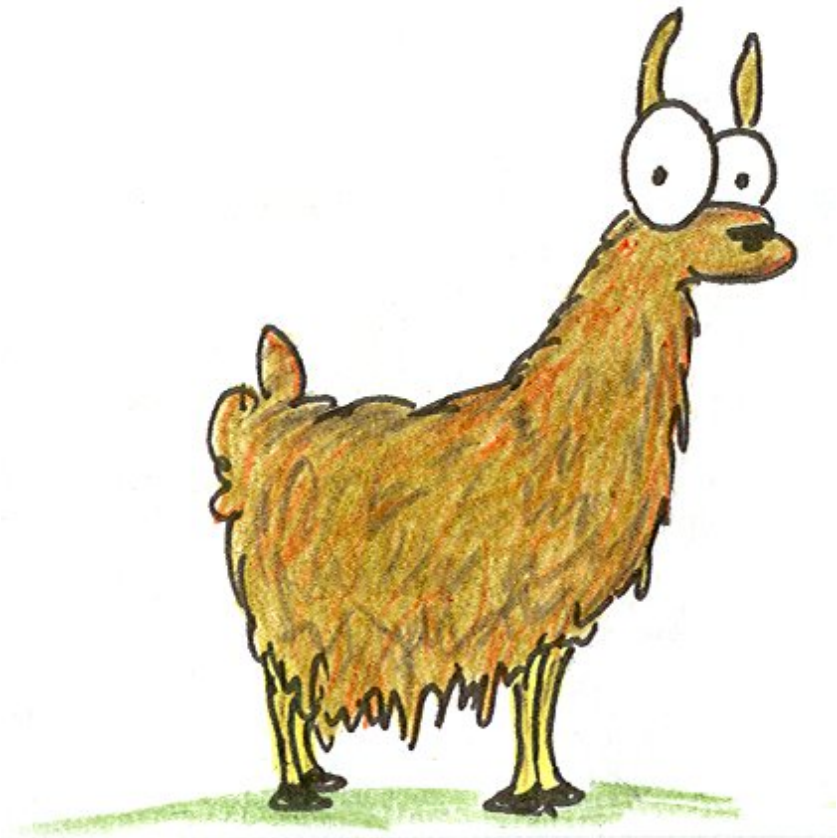
Fork me on GitHub

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
Lagoudakis and Littman 2000	order selection, sorting	instance features	remaining cost for each sub-problem	MDP	online	static	2000
Sillito 2000	constraints	probing	cost of solving problem	statistical model	offline	static	2000
Pfahringer et al. 2000	classification	instance features, probing	algorithm	9 different classifiers	offline	static	2000
Fukunaga 2000	TSP	past performance	resource allocation	performance simulation for different allocations	offline	static	2000
Soares and Brazdil 2000	machine learning	instance features	ranking	nearest neighbour	offline	static	2000
Gomes and Selman 2001	constraints, mixed integer programming	past performance	algorithm	statistical model	offline	dynamic	2001

`http://larskotthoff.github.io/assurvey/`

Kotthoff, Lars. "Algorithm Selection for Combinatorial Search Problems: A Survey." AI Magazine 35, no. 3 (2014): 48–60.

Demo(ish) – LLAMA



LLAMA

- R package
- access to most machine learning methods in R
- implements the most common approaches from the literature and a few extra ones
- open source

30 second tutorial – LLAMA and ASlib

```
library(llama)
library(aslib)

scenario = parseASScenario("QBF-2011/")
data = convertToLlamaCVFolds(scenario)

learner = makeLearner("classif.randomForest")
model = classify(learner, data)

mean(parscores(data, model))
> 9774.813
mean(parscores(data, vbs))
> 8337.099
mean(parscores(data, singleBest))
> 15228.53
```

Getting Started – LLAMA

```
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
```

Example Data

- 19 SAT solvers
- 2433 instances
- 36 features

Under the Hood

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

(dyn_log_propags <= 3.169925) and (log_ranges >=
    3.321928) and (sqrt_avg_domsize >= 3.162278) and
    (sqrt_avg_domsize <= 3.162278) => target=riss
    (12.0/0.0)
(log_values <= 8.366322) and (log_values >= 8.366322)
    and (dyn_log_nodes <= 6.066089) =>
    target=MPhaseSAT64 (17.0/6.0)
(dyn_log_propags >= 23.653658) and (log_values <=
    9.321928) => target=march_rw (253.0/108.0)
(dyn_log_avg_weight <= 5.149405) and (percent_global <=
    0.079239) and (sqrt_avg_domsize >= 3.872983) and
    (dyn_log_stdev_weight >= 3.961314) and
    (dyn_log_nodes <= 8.005625) => target=glucose
    (33.0/3.0)
(percent_global <= 0.048745) and (dyn_log_avg_weight <=
    5.931328) and (sqrt_avg_domsize >= 3.872983) and
    (dyn_log_nodes >= 5) and (dyn_log_propags <=
    14.391042) => target=glucose (42.0/10.0)
```

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
```

Summary

Algorithm Selection choose the best *algorithm* for solving a problem

Algorithm Configuration choose the best *parameter configuration* for solving a problem with an algorithm

- ▷ mature research areas
- ▷ can combine configuration and selection
- ▷ effective tools are available

COSEAL

Interested? Join the COSEAL group!

http://coseal.net