Getting the Best out of your Constraint Solver: Portfolios and Automated Tuning

Lars Kotthoff

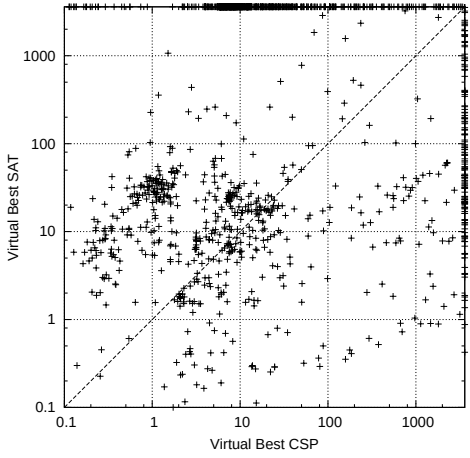
larsko@uwyo.edu

ACP Summer School, Leuven, 11 July 2023

Big Picture

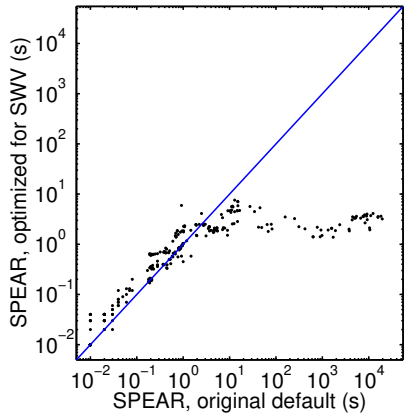
- ▷ AI and CP are difficult, choosing the "right" approach is even more difficult
- ▷ explore the space of possibilities automatically and efficiently, lowering threshold of adoption
- ▷ anybody can use state-of-the-art AI and CP

Performance Differences



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

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.

Common Theme

Meta-algorithmics to improve performance

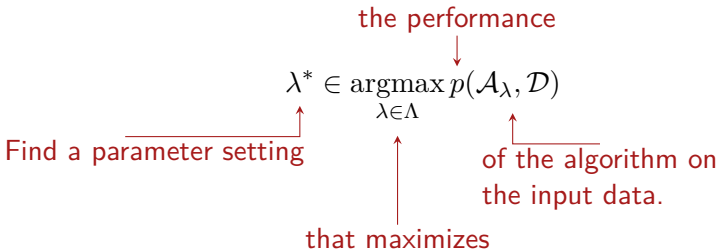
- ▷ no deep understanding of what happens required
- ▷ adapt existing approaches rather than developing new ones
- ▷ all of this happens automatically (more or less)

Algorithm Selection

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

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

Automated Parameter Tuning



Algorithm 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.

Algorithms

"algorithm" used in a very loose sense

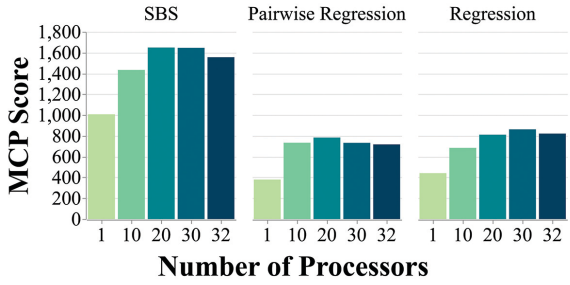
- ▷ constraint solvers
- ▷ search strategies
- ▷ modelling choices
- ▷ different types of consistency
- ▷ ...

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)

Aside: Why not simply run all algorithms in parallel?

- too many!
- waste of resources
- actually slower...



Haniye Kashgarani and Lars Kotthoff, "Is Algorithm Selection Worth It? Comparing Selecting Single Algorithms and Parallel Execution," in AAAI Workshop on Meta-Learning and MetaDL Challenge, vol. 140, Proceedings of Machine Learning Research (PMLR, 2021), 58–64.

Building an Algorithm Selection System

- most approaches rely on machine learning
- train with representative data, i.e. performance of all algorithms in portfolio on a number of instances
- evaluate performance on separate set of instances
- potentially large amount of prep work

Choosing Instances

- ▷ we want selectors that generalise, i.e. good for more than one instance
- ▷ 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
- ▷ may need hundreds or thousands of instances

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 small 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
- ...

No Features

- ▷ use deep learning on problem representation
- ▷ no need for expert-defined features
- ▷ competitive in some areas

Andrea Loreggia et al., 'Deep Learning for Algorithm Portfolios', in Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016 (AAAI Press, 2016), 1280–86, http://dl.acm.org/citation.cfm?id=3015812.3016001.

Maxime Gasse et al., 'Exact Combinatorial Optimization with Graph Convolutional Neural Networks', in 33rd International Conference on Neural Information Processing Systems (Curran Associates Inc., 2019).

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
- ▷ often only need a handful of features in practice
- ▷ in the end, whatever works best

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

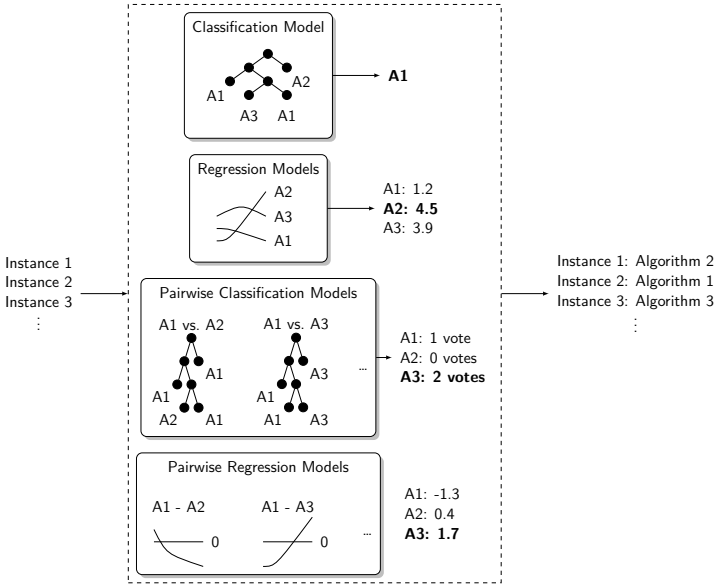
- ▷ get the best of both worlds
- ▷ for example: consider pairs of algorithms to take relations into account
- ▷ for each pair of algorithms, learn model that predicts which one is faster

---

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.

Types of Performance Models



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.

Kashgarani, Haniye, and Lars Kotthoff. "Automatic Parallel Portfolio Selection." 2023.

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

Example System – SATzilla

- portfolio of 7 SAT solvers, trained on 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, selection of algorithms to include in portfolio based on overall performance
- 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.

Example System – Learn2Branch

- ▷ branching variable selection in SCIP MIP solver
- ▷ Graph Neural Network on graph representation of problem to learn branching policy

---

Maxime Gasse et al., 'Exact Combinatorial Optimization with Graph Convolutional Neural Networks', in 33rd International Conference on Neural Information Processing Systems (Curran Associates Inc., 2019).

Solver Features

- ▷ automatically analyze source code of solvers, extract e.g. number of lines of code, cyclomatic complexity, connectivity of Abstract Syntax Tree
- ▷ use together with problem instance features to make performance predictions
- ▷ only single model for all algorithms needed

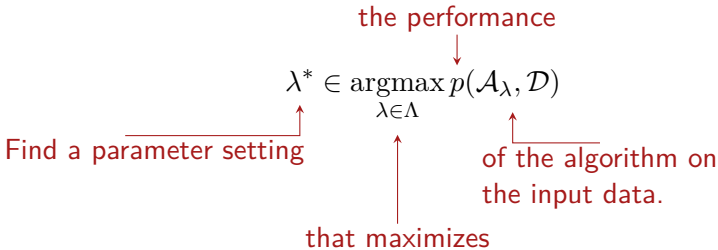
Damir Pulatov et al., "Opening the Black Box: Automated Software Analysis for Algorithm Selection," First Conference on Automated Machine Learning (2022).

Benchmark library – ASlib

- ▷ `https://github.com/coseal/aslib_data`
- ▷ SAT, CSP, QBF, ASP, MAXSAT, OR
- ▷ includes data used frequently in the literature that you should evaluate your approach on
- ▷ `http://aslib.net`

Bischl, Bernd, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, et al. "ASlib: A Benchmark Library for Algorithm Selection." Artificial Intelligence Journal (AIJ), no. 237 (2016): 41–58.

Automated Parameter Tuning



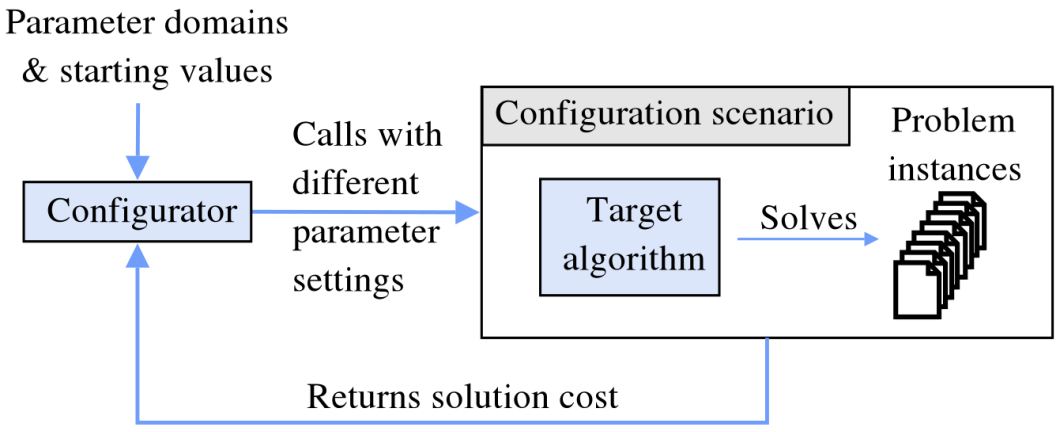
Parameters?

- ▷ anything you can change that makes sense to change
- ▷ e.g. search heuristic, variable ordering, type of global constraint decomposition
- ▷ not random seed, whether to enable debugging, etc.
- ▷ 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)...

Algorithm Configuration



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, KUL, UWyo...
- conditional dependencies – e.g. if A is 1, B can't be 2, C is only active if A is >2

→ 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
- ▷ decide where to evaluate next
- ▷ balance diversification/exploration and intensification/exploitation

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

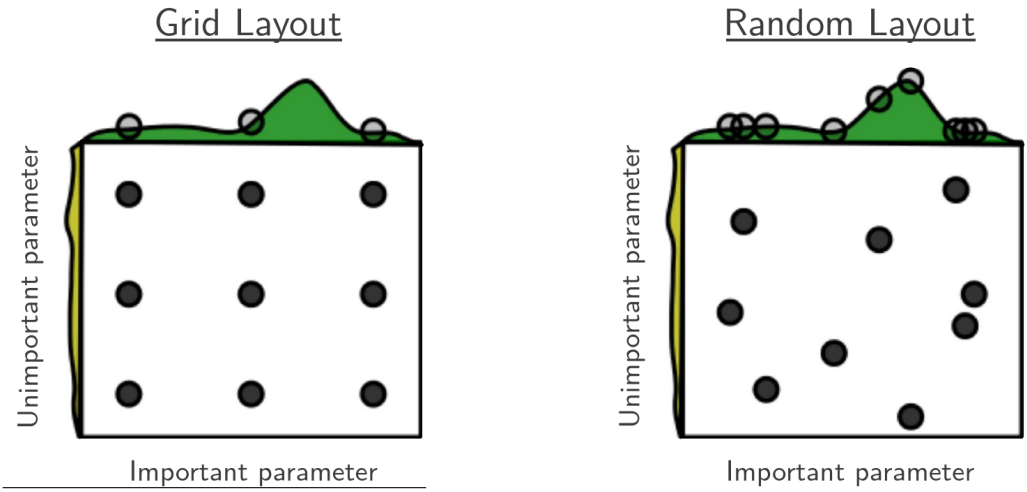
→ How do we know when to stop?

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 instance set
- ▷ for runtime: adaptive capping

Grid and Random Search



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

Local Search

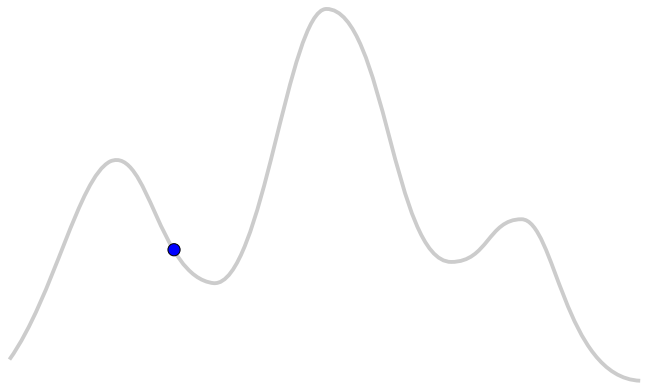
- ▷ start with random configuration
- ▷ change a single parameter (local search step)
- ▷ if better, keep the change, else revert
- ▷ repeat
- ▷ optional (but important): restart with new random configurations

---

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.

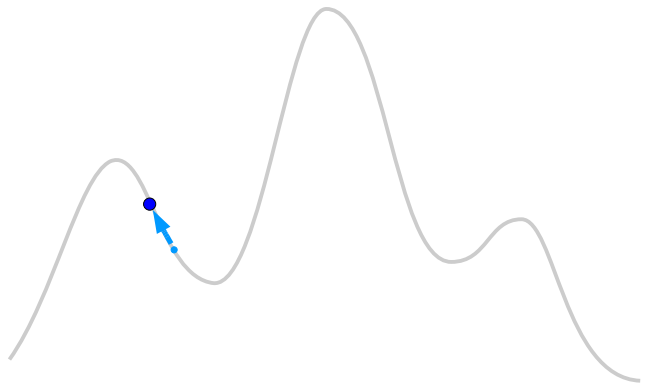
Local Search Example

Local Search Example



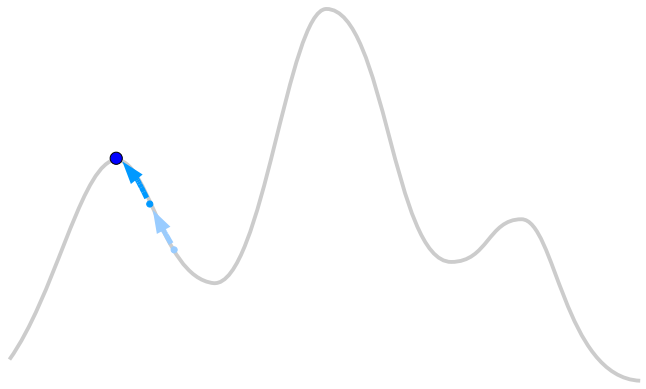
Initialisation

Local Search Example



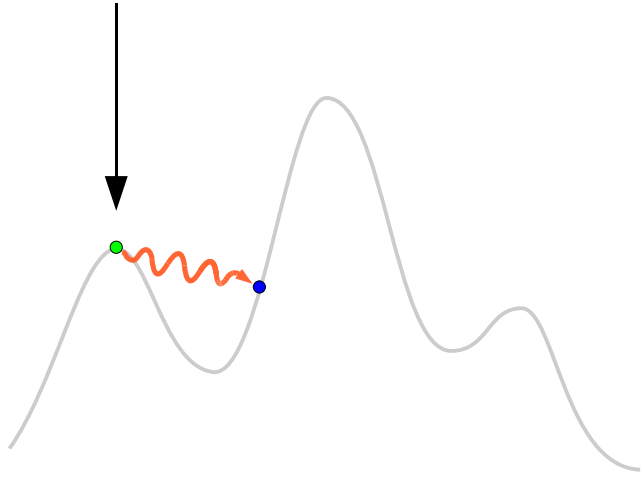
Local Search

Local Search Example



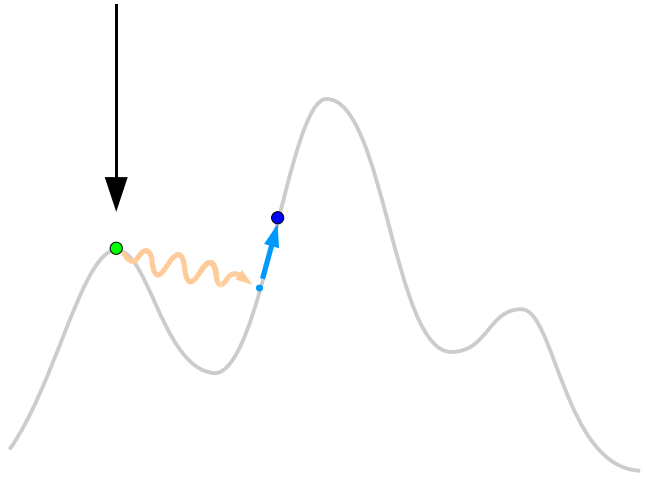
Local Search

Local Search Example



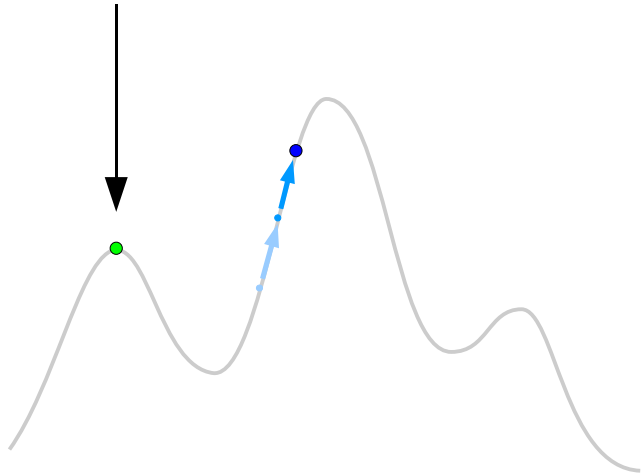
Perturbation

Local Search Example



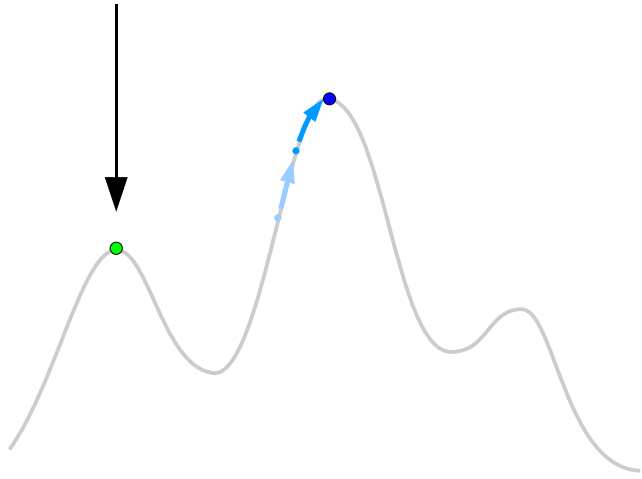
Local Search

Local Search Example



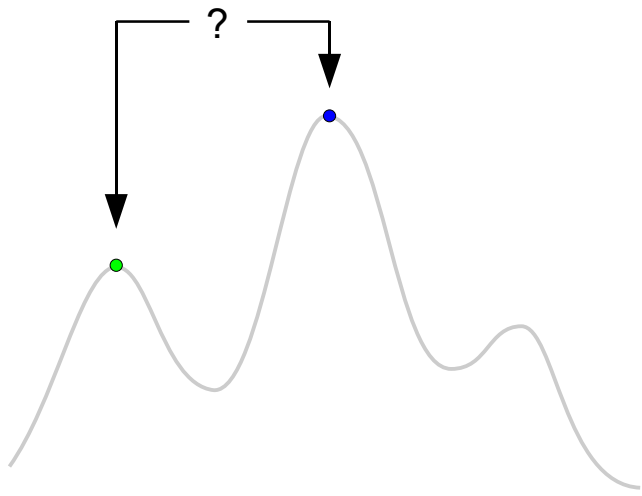
Local Search

Local Search Example



Local Search

Local Search Example

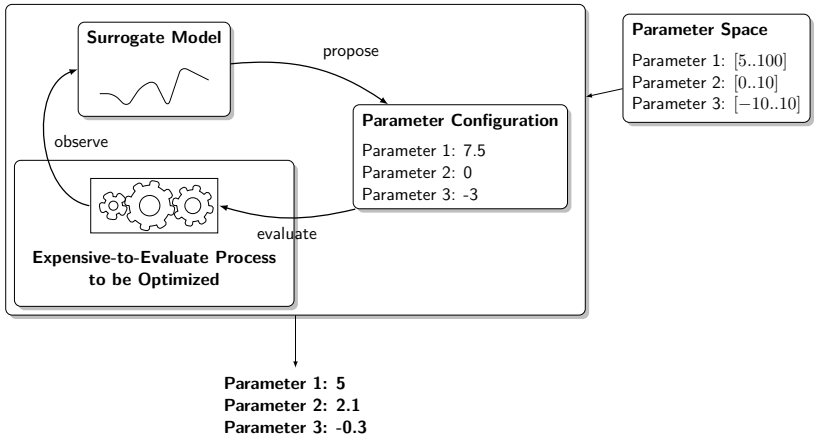


Selection (using Acceptance Criterion)

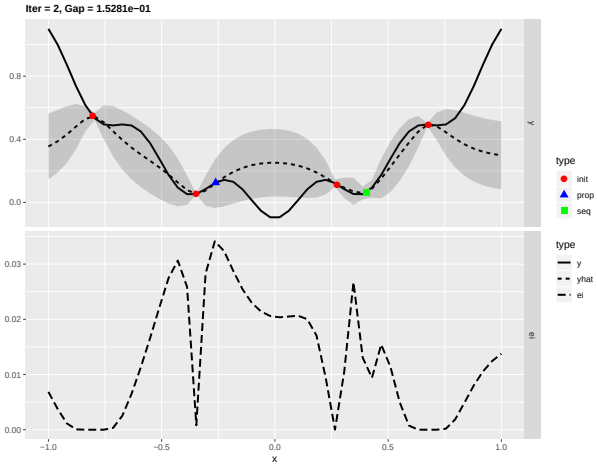
Model-Based Optimization

- ▷ evaluate small number of initial (random) configurations
- ▷ build surrogate model of parameter-performance surface based on this
- ▷ use model to predict where to evaluate next
- ▷ repeat, stop when resources exhausted or desired solution quality achieved
- ▷ allows targeted exploration of promising configurations

Model-Based Optimization

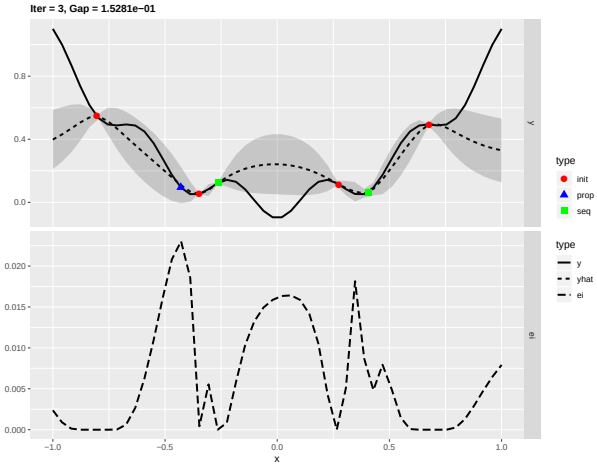


Model-Based Optimization Example



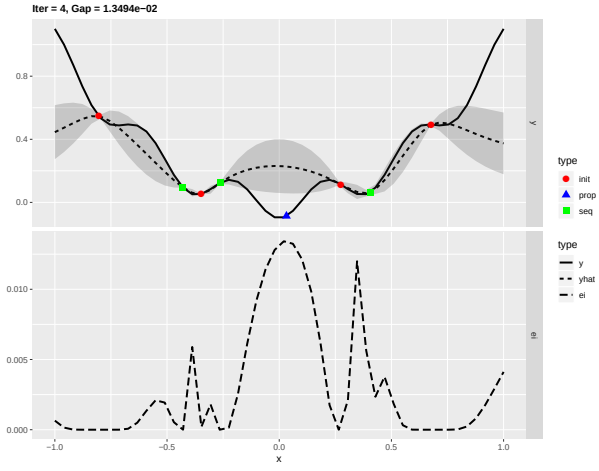
Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. http://arxiv.org/abs/1703.03373.

Model-Based Optimization Example



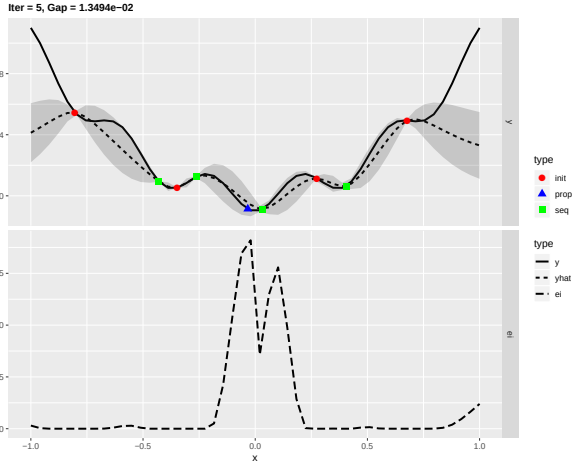
Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. `http://arxiv.org/abs/1703.03373`.

Model-Based Optimization Example



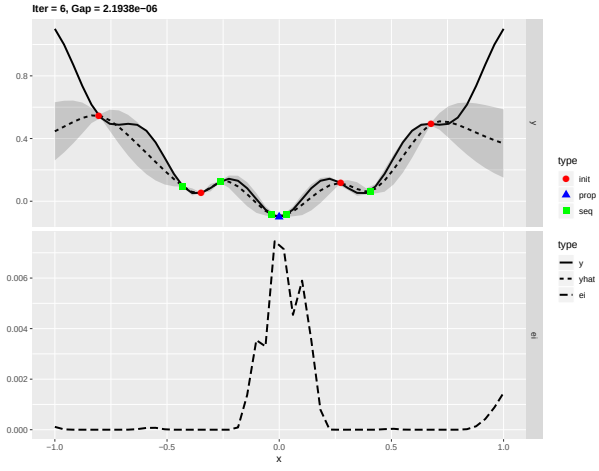
Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. `http://arxiv.org/abs/1703.03373`.

Model-Based Optimization Example



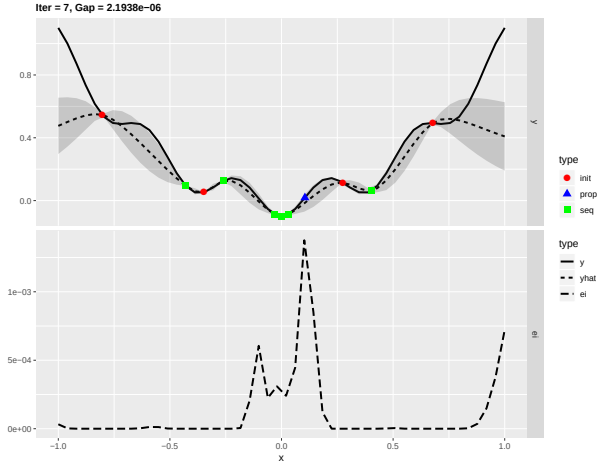
Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. `http://arxiv.org/abs/1703.03373`.

Model-Based Optimization Example



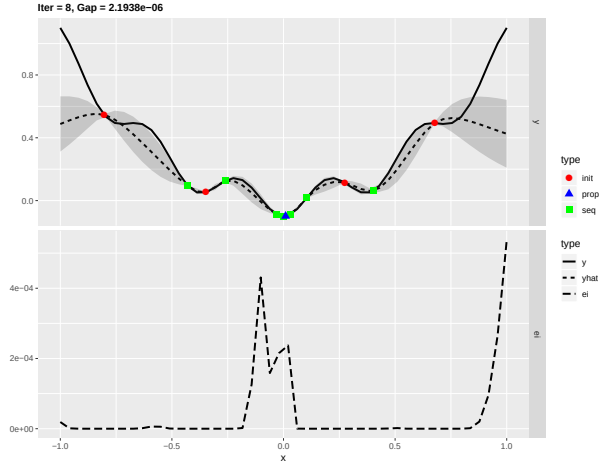
Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. http://arxiv.org/abs/1703.03373.

Model-Based Optimization Example



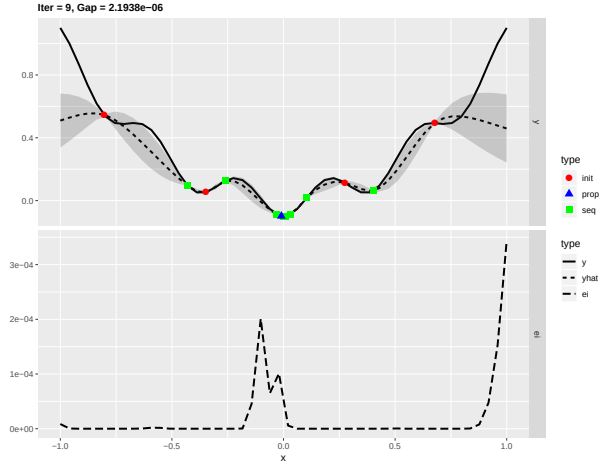
Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. `http://arxiv.org/abs/1703.03373`.

Model-Based Optimization Example



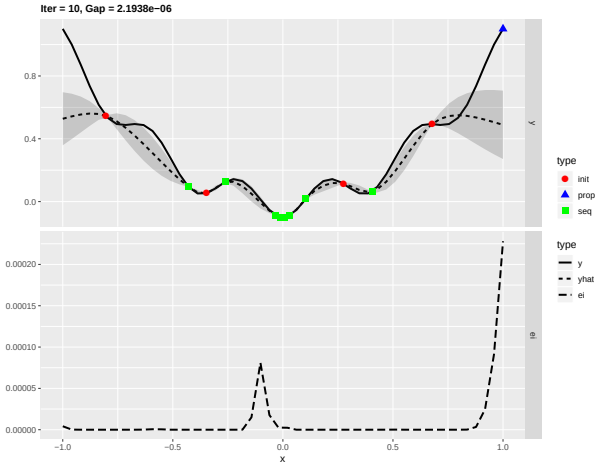
Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. http://arxiv.org/abs/1703.03373.

Model-Based Optimization Example



Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. `http://arxiv.org/abs/1703.03373`.

Model-Based Optimization Example



Bischl, Bernd, Jakob Richter, Jakob Bossek, Daniel Horn, Janek Thomas, and Michel Lang. "MlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," March 9, 2017. `http://arxiv.org/abs/1703.03373`.

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 instances that have a solution
- ▷ can be leveraged with algorithm selection to create complementary portfolios

---

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.

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
- ▷ COnfiguration and SElection of ALgorithms group COSEAL
  `http://www.coseal.net` – annual meetings, 2024 in Dresden

Resources

- SMAC – `https://github.com/automl/SMAC3`
- autofolio – `https://github.com/automl/AutoFolio`
- Pascal Kerschke et al., 'Automated Algorithm Selection: Survey and Perspectives', Evolutionary Computation 27, no. 1 (2019): 3–45, `https://doi.org/10.1162/evco_a_00242`.
- Katharina Eggensperger, Marius Lindauer, and Frank Hutter, 'Pitfalls and Best Practices in Algorithm Configuration', Journal of Artificial Intelligence Research 64 (2019), `https://doi.org/10.1613/jair.1.11420`.

Questions?

Microsoft

NSF

COLLEGE OF ENGINEERING & APPLIED SCIENCE

UNIVERSITY OF WYOMING

Automated Tuning Exercises

https://tinyurl.com/acp23-tuning

Now go for it...

- try more evaluations
- try a different constraint problem
- expand the parameter space and/or try a different solver
- handle noisy runtime evaluations
- consider the uncertainty of predictions when deciding where to evaluate next
- use the `BayesOpt` package
- ...