AI-Augmented Algorithms

How I Learned to Stop Worrying and Love Choice

Lars Kotthoff

University of Wyoming

larsko@uwyo.edu

Boulder, 16 January 2019

Outline

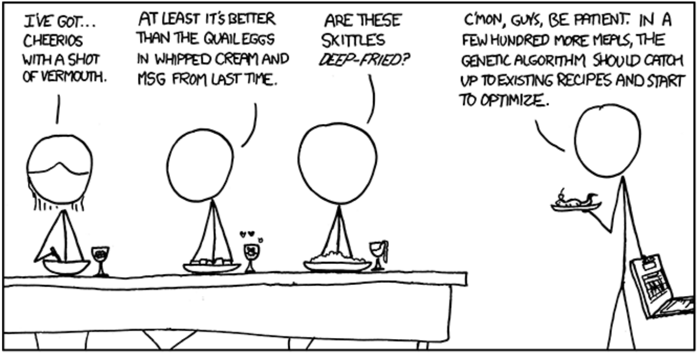
- Big Picture
- Motivation
- Choosing Algorithms
- Tuning Algorithms
- (NCAR-relevant) Applications
- Outlook and Resources

Big Picture

- advance the state of the art through meta-algorithmic techniques
- rather than inventing new things, use existing things more intelligently – automatically
- invent new things through combinations of existing things

Big Picture

- advance the state of the art through meta-algorithmic techniques
- rather than inventing new things, use existing things more intelligently – automatically
- invent new things through combinations of existing things

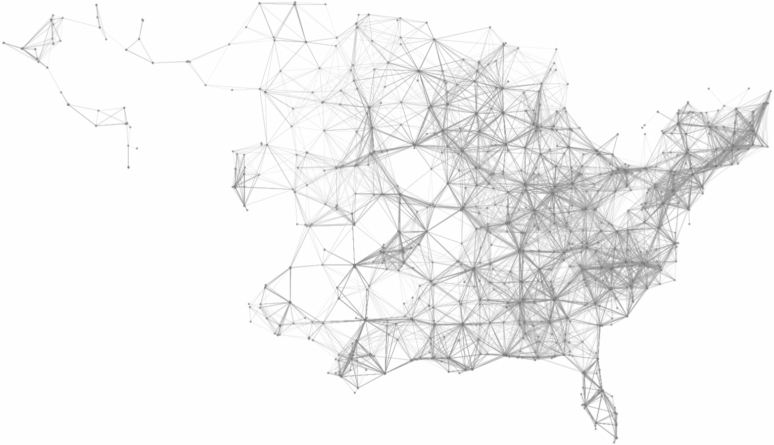


WE'VE DECIDED TO DROP THE CS DEPARTMENT FROM OUR WEEKLY DINNER PARTY HOSTING ROTATION.

https://xkcd.com/720/

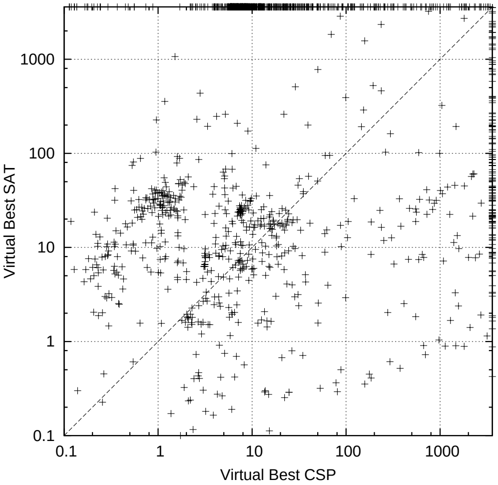
Motivation – What Difference Does It Make?

Prominent Application



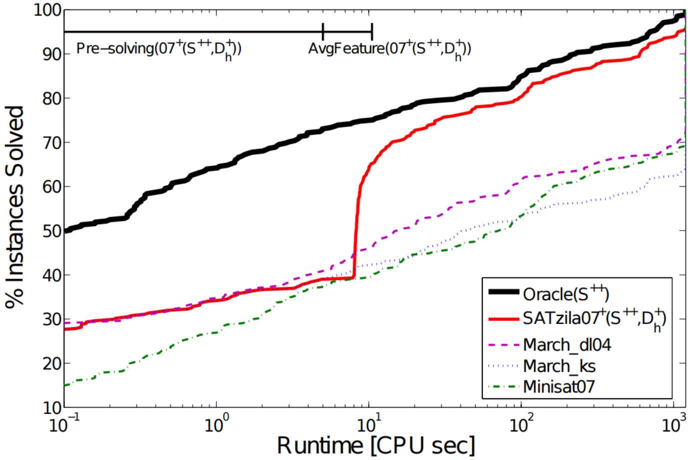
Fréchette, Alexandre, Neil Newman, Kevin Leyton-Brown. "Solving the Station Packing Problem." In Association for the Advancement of Artificial Intelligence (AAAI), 2016.

Performance Differences



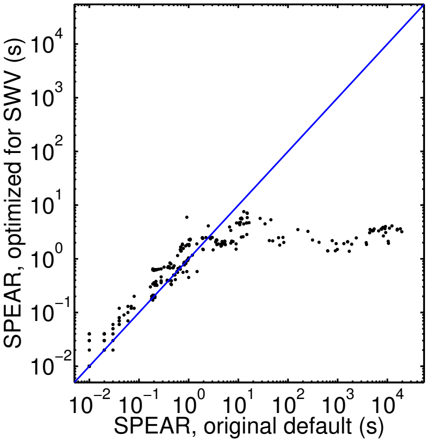
Hurley, Barry, Lars Kotthoff, Yuri Malitsky, and Barry O'Sullivan. "Proteus: A Hierarchical Portfolio of Solvers and Transformations." In CPAIOR, 2014.

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.

Common Theme

Performance models of black-box processes

- also called surrogate models
- substitute expensive underlying process with cheap approximate model
- build approximate model using machine learning techniques based on results of evaluations of the underlying process
- no knowledge of what the underlying process is required (but can be helpful)
- may facilitate better understanding of the underlying process through interrogation of the model

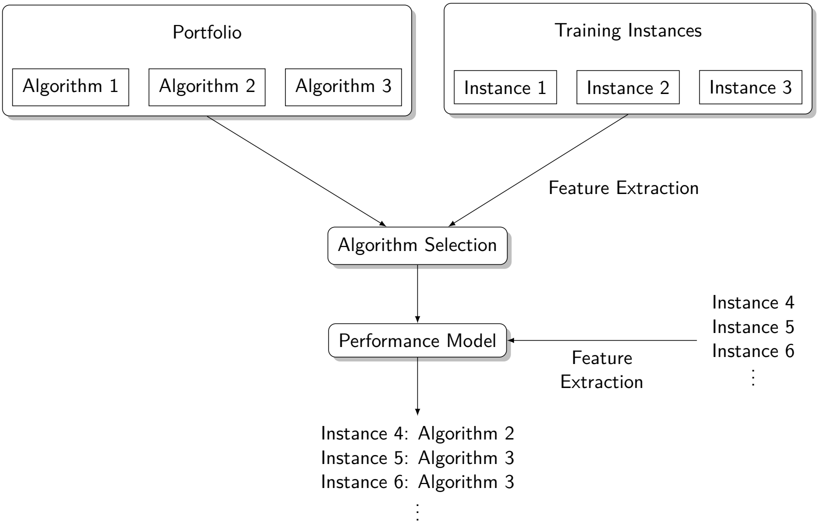
Choosing Algorithms

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.

Algorithm Selection



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 or other algorithms known to have good performance

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

- algorithms
- heuristics
- machine learning models
- software systems
- machines
- ...

Parallel Portfolios

Why not simply run all algorithms in parallel?

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

Building an Algorithm Selection System

- requires algorithms with complementary performance
- 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

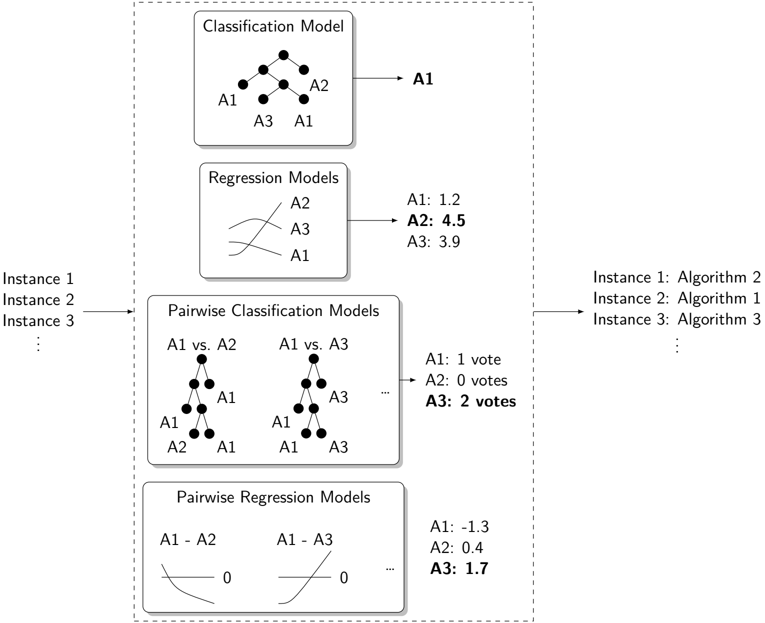
Key Components of an Algorithm Selection System

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

optional:

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

Types of Performance Models



Tuning Algorithms

Algorithm Configuration

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

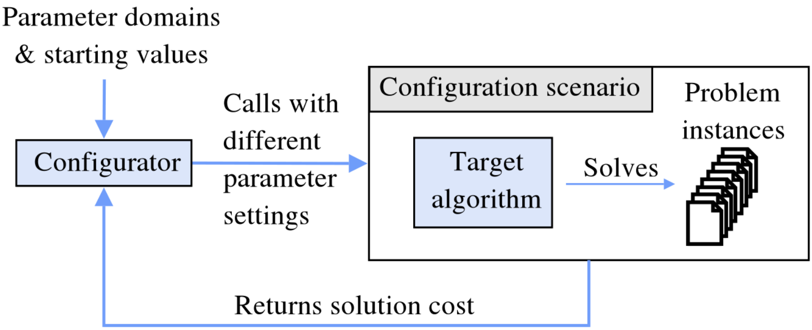
Parameters?

- anything you can change that makes sense to change
- e.g. search heuristic, optimization level, computational resolution
- not random seed, whether to enable debugging, etc.
- some will affect performance, others will have no effect at all

Automated Algorithm Configuration

- no background knowledge on parameters or algorithm – black-box process
- as little manual intervention as possible
  - failures are handled appropriately
  - resources are not wasted
  - can run unattended on large-scale compute infrastructure

Algorithm Configuration



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

General Approach

- evaluate algorithm as black-box function
- observe effect of parameters without knowing the inner workings, build surrogate model based on this data
- decide where to evaluate next, based on surrogate model
- repeat

When are we done?

- most approaches incomplete, i.e. do not exhaustively explore parameter space
- 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?

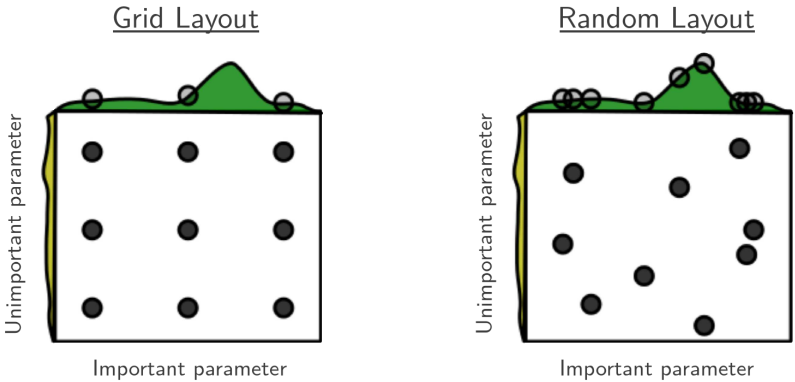
Time Budget

How much time/how many function evaluations?

- too much $\rightharpoonup$ wasted resources
- too little $\rightharpoonup$ suboptimal result
- use statistical tests
- evaluate on parts of the instance set
- for runtime: adaptive capping
- in general: whatever resources you can reasonably invest

Grid and Random Search

- evaluate certain points in parameter space



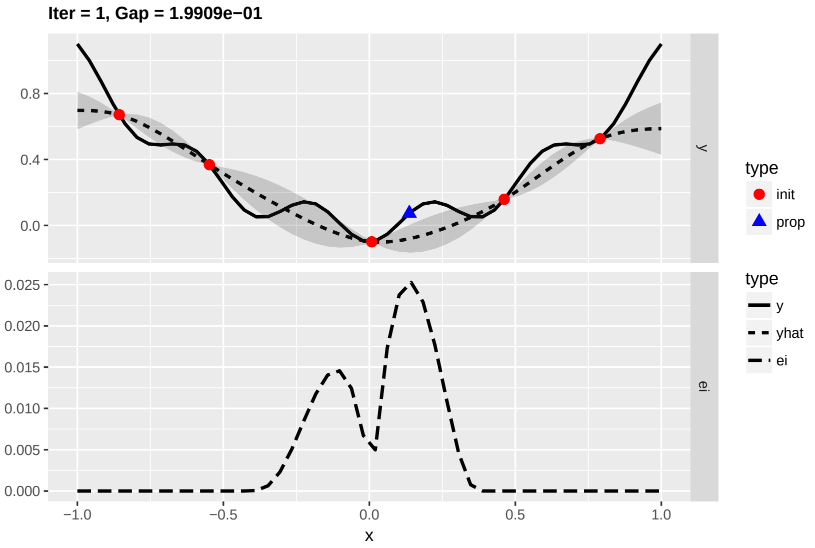
Bergstra, James, and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." J. Mach. Learn. Res. 13, no. 1 (February 2012): 281–305.

Model-Based Search

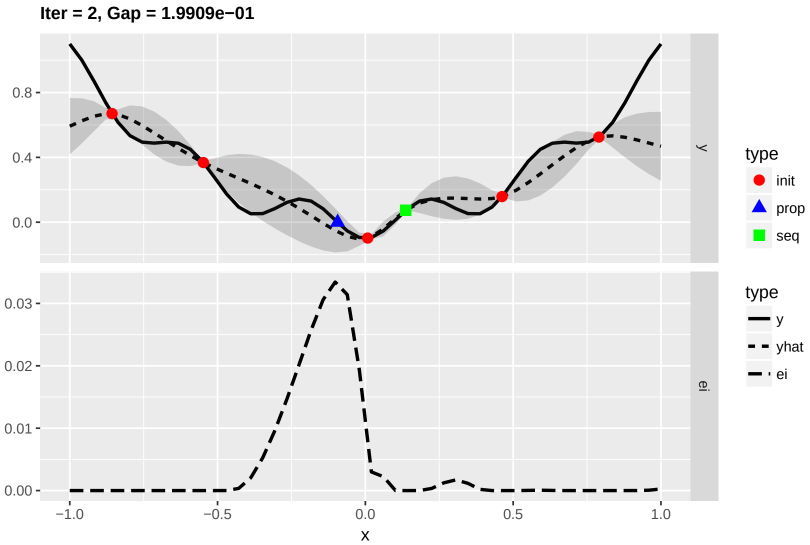
- evaluate small number of configurations
- build model of parameter-performance surface based on the results
- use model to predict where to evaluate next
- repeat
- allows targeted exploration of new configurations
- can take instance features into account like algorithm selection

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

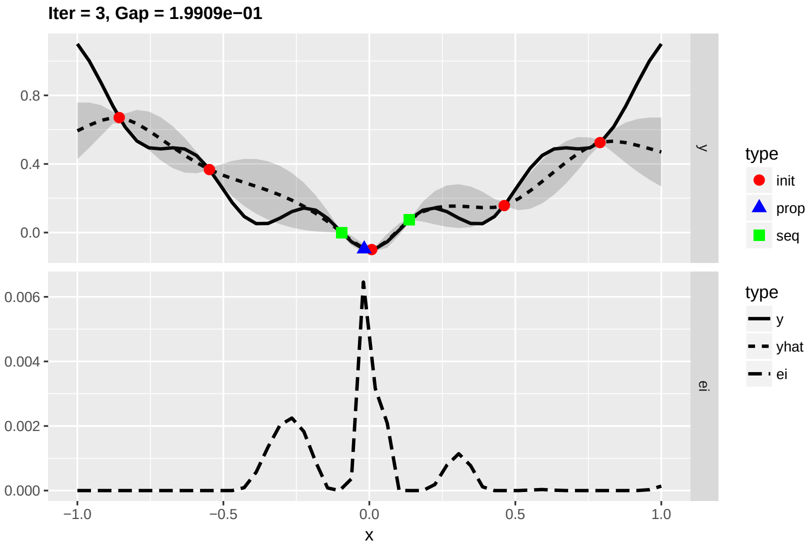
Model-Based Search Example



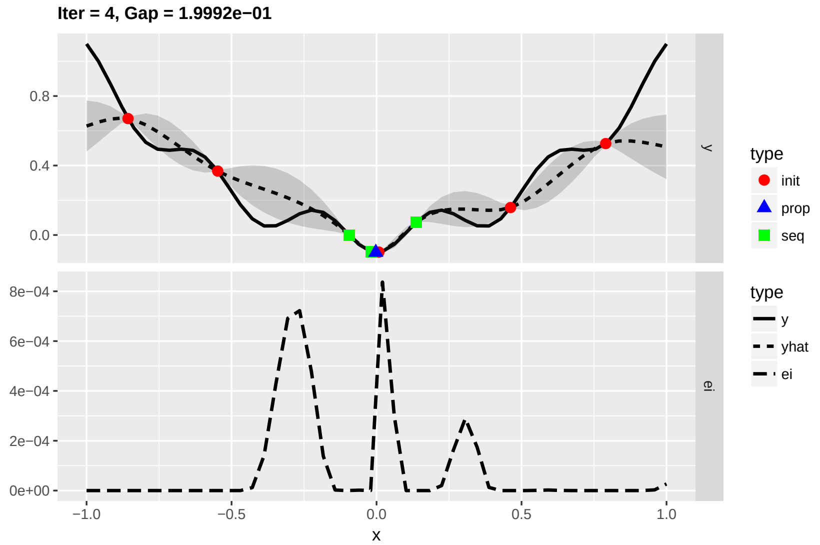
Model-Based Search Example



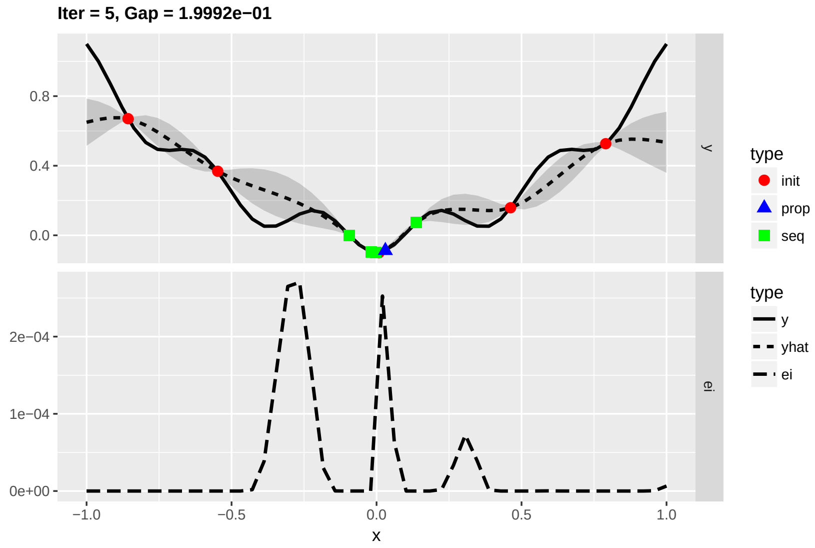
Model-Based Search Example



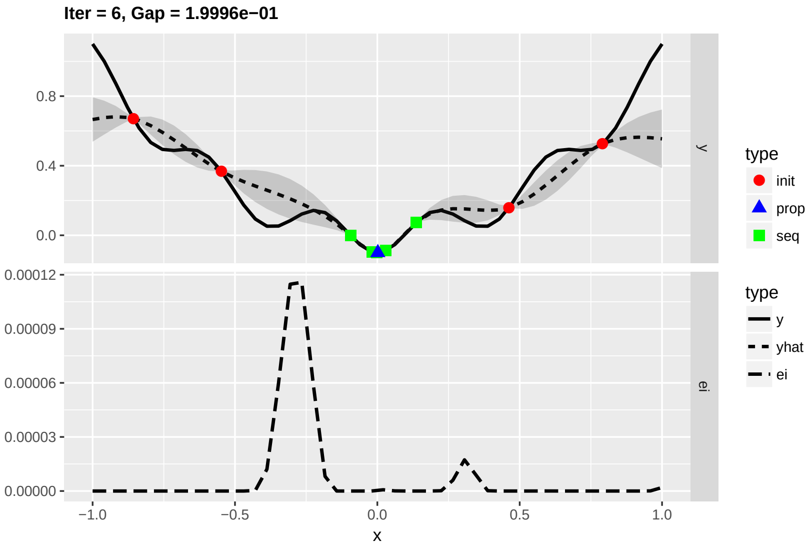
Model-Based Search Example



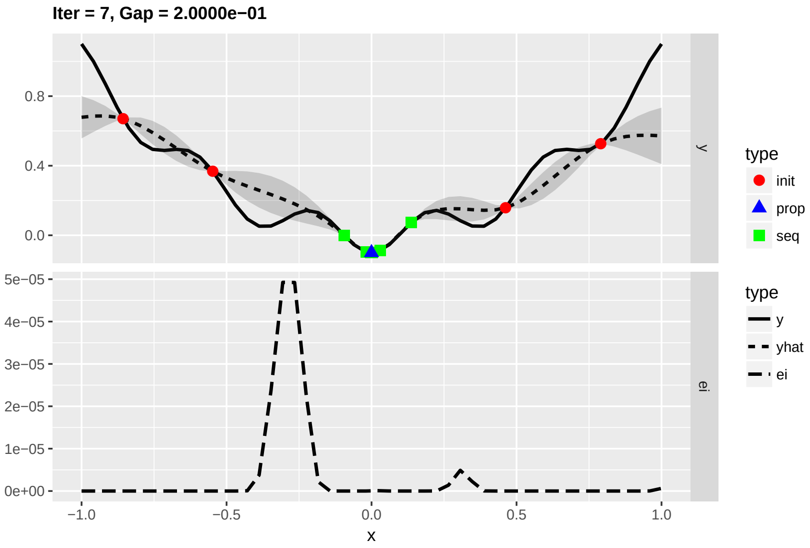
Model-Based Search Example



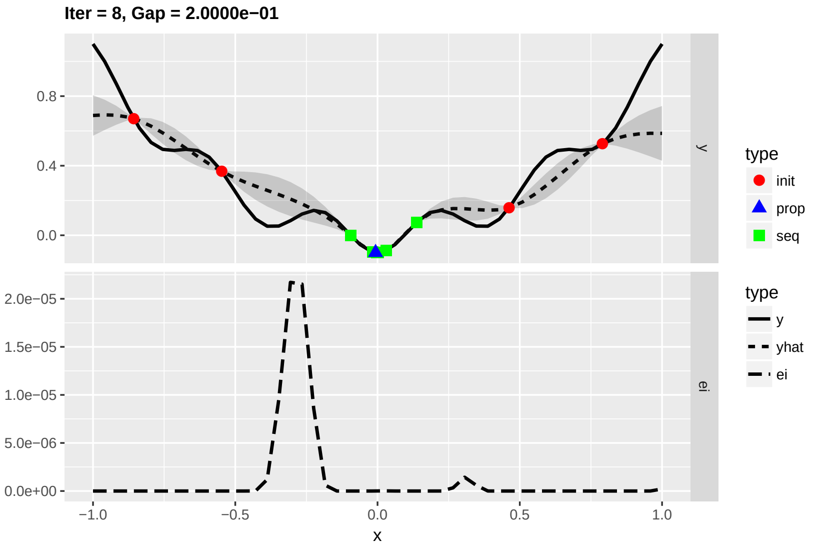
Model-Based Search Example



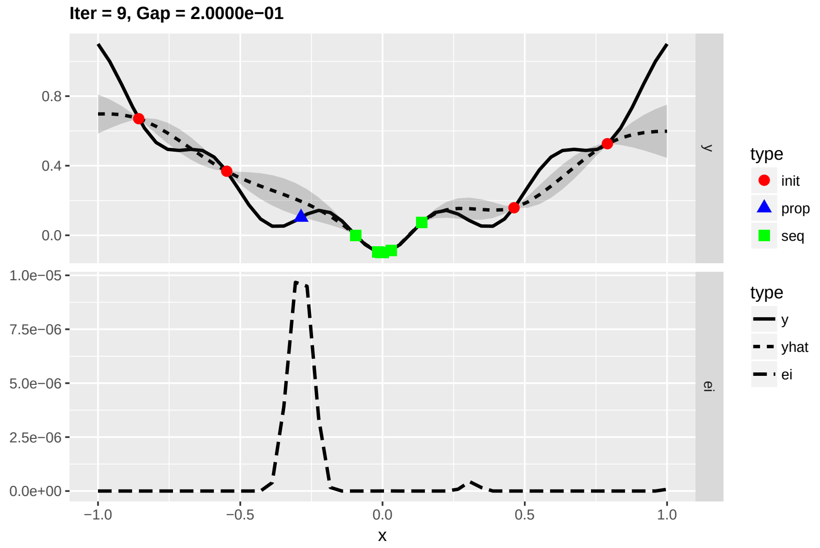
Model-Based Search Example



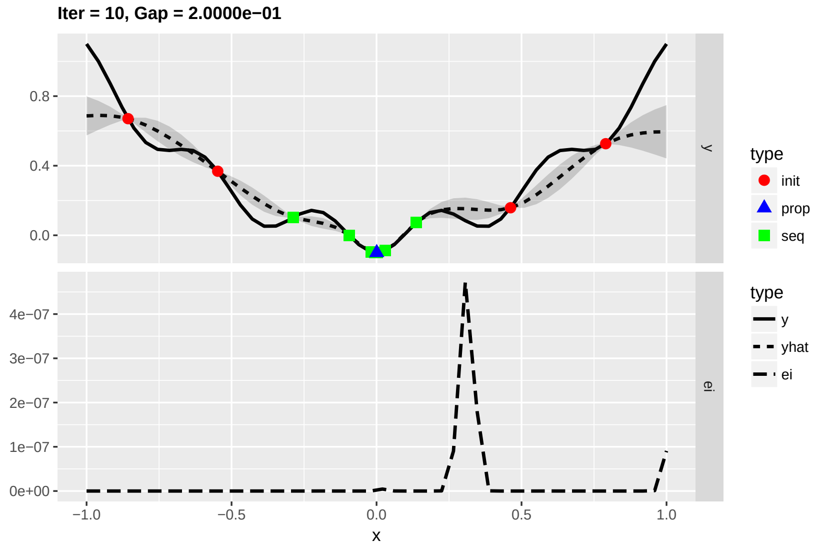
Model-Based Search Example



Model-Based Search Example



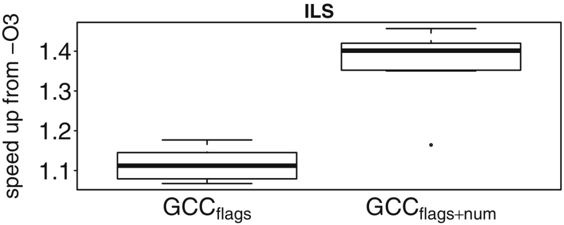
Model-Based Search Example



Selected Applications

Compiler Parameter Tuning

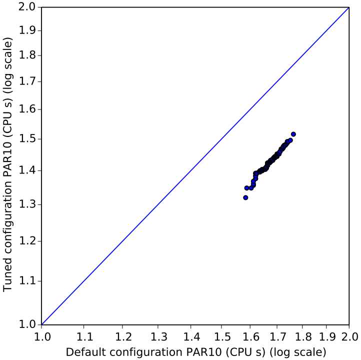
- pre-defined optimization levels offer not much flexibility
- improvements possible by tuning full compiler parameter space
- tuned compute-intensive AI algorithms
- up to 40% runtime improvement over gcc -O2/-O3



Pérez Cáceres, Leslie, Federico Pagnozzi, Alberto Franzin, and Thomas Stützle. "Automatic Configuration of GCC Using Irace." In Artificial Evolution, edited by Evelyne Lutton, Pierrick Legrand, Pierre Parrend, Nicolas Monmarché, and Marc Schoenauer, 202–16. Cham: Springer International Publishing, 2018.

Compiler Parameter Tuning

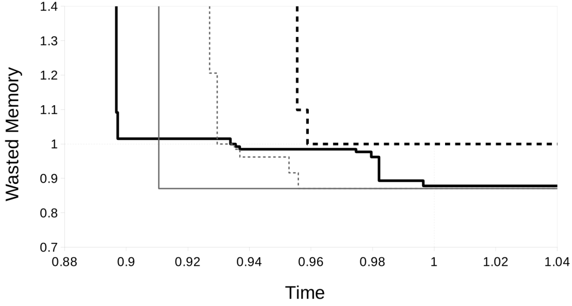
- not only for C/C++
- JavaScript (JavaScriptCode, V8) optimized for standard benchmarks
- up to 35% runtime improvement



Fawcett, Chris, Lars Kotthoff, and Holger H. Hoos. "Hot-Rodding the Browser Engine: Automatic Configuration of JavaScript Compilers." CoRR abs/1707.04245 (2017). http://arxiv.org/abs/1707.04245.

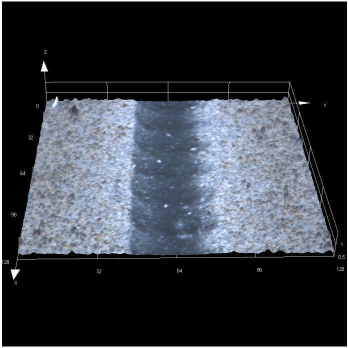
“Deep” Parameter Tuning

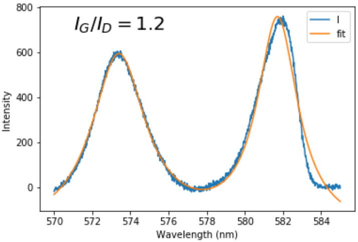
- automatically identify non-exposed parameters and allow them to be tuned (e.g. magic constants)
- tuned dlmalloc library, specialized for e.g. awk, flex, sed
- runtime improvements of up to 12%, decrease in memory consumption of up to 21%

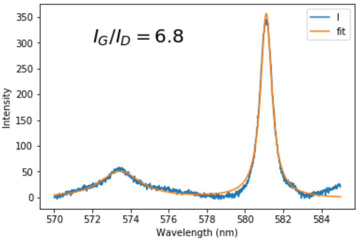


Wu, Fan, Westley Weimer, Mark Harman, Yue Jia, and Jens Krinke. “Deep Parameter Optimisation.” In Conference on Genetic and Evolutionary Computation, 1375–82. GECCO ’15. New York, NY, USA: ACM, 2015. https://doi.org/10.1145/2739480.2754648.

Beyond Software





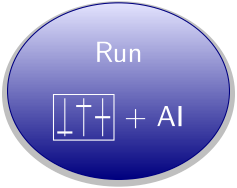


Outlook

Quo Vadis, Software Engineering?

Run

Quo Vadis, Software Engineering?



Hoos, Holger H. "Programming by Optimization." Communications of the Association for Computing Machinery (CACM) 55, no. 2 (February 2012): 70–80. https://doi.org/10.1145/2076450.2076469.

(Much) More Information

Comments? Suggestions? Corrections?
Let me know!

Algorithm Selection Literature Summary

Last update 21 November 2018

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	CSP	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	CSP	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	CSP	search statistics	switch algorithm?	hand-crafted rules	online	static, static order	1996
Allen and Minton 1996	SAT, CSP	probing	runtime performance	hand-crafted rules	online	static	1996
Sakkout et al. 1996	CSP	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	CSP	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	CSP	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	CSP, mixed integer programming	past performance	algorithm	statistical model	offline	dynamic	2001
Epstein and Freuder 2001, Epstein et al. 2002, Epstein et al. 2005, Epstein and Petrovic 2011	CSP	variable characteristics	algorithm	weights, hand-crafted rules	offline and online	dynamic	2001
Lagoudakis and Littman 2001	DPLL branching rules	instance features	remaining cost for each sub-problem	MDP	online	static	2001
Nareyek 2001	optimization	search statistics	expected utility of algorithm	reinforcement learning	offline and online	static	2001
Horvitz et al. 2001	CSP	instance and instance generator features, search statistics	runtime performance, restart parameters	Bayesian model	offline and online	static	2001

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

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

Tools and Resources

LLAMA	`https://bitbucket.org/lkotthoff/llama`
SATzilla	`http://www.cs.ubc.ca/labs/beta/Projects/SATzilla/`
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/`
autofolio	`https://bitbucket.org/mlindauer/autofolio/`
Auto-WEKA	`http://www.cs.ubc.ca/labs/beta/Projects/autoweka/`
Auto-sklearn	`https://github.com/automl/auto-sklearn`

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`

Don't set parameters prematurely, embrace choice!

I'm hiring!

Several funded graduate positions available.