Learning as a Tool for Algorithm Design and Beyond-Worst-Case Analysis

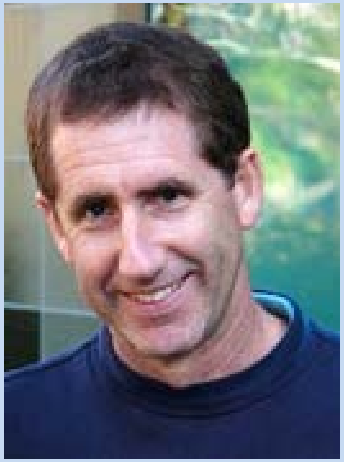
Kevin Leyton-Brown
Computer Science Department
University of British Columbia

This Talk Suveys 15 Years of Work with/by Many Collaborators, notably:

Holger Hoos
UBC

Frank Hutter
UBC

Eugene Nudelman
Stanford/Google



Yoav Shoham
Stanford

Lin Xu
UBC

[L-B, Nudelman, Shoham: CP 2002; JACM 2009]
[Nudelman, L-B, Hoos, Devkar, Shoham: CP 2004]
[Xu, Hoos, L-B: CP 2007; AAAI 2012]
[**Hutter, Xu, Hoos, L-B: CACM 2014**; AIJ 2015]

Intractability

> Problems are intractable when they "can be solved, but not fast enough for the solution to be usable"
>
> [Hopcroft, Motwani & Ullman, 2007]

- NP-complete problems are commonly said to be intractable, but the reality is more complex
- The best available methods tend to
  - offer **no interesting theoretical guarantees**
  - work **astoundingly well** in practice
  - exhibit **exponentially varying performance** (e.g., milliseconds to days) even on fixed-size problems

Motivating Question

"How hard is it to solve a given problem in practice, using the best available methods?"

Even in settings where **formal analysis seems hopeless**:

- algorithms are complex black boxes
- instance distributions are heterogeneous or richly structured

…it is possible to apply **rigorous statistical methods** to answer such questions with high levels of confidence.

EMPIRICAL HARDNESS MODELS:

Learning the Performance of Algorithms for NP-Complete Problems

[L-B, Nudelman, Shoham: CP 2002; JACM 2009]
[Hutter, Xu, Hoos, L-B, INFORMS 2006; CACM 2014; AIJ 2015]
[Hutter, Xu, Hoos, L-B: CACM 2014]

Empirical Hardness Models

- Predict how long an algorithm will take to run, given:
  - A **set of instances** D
  - For each instance $i \in D$, a vector $\boldsymbol{x}_i$ of **feature values**
  - For each instance $i \in D$, a **runtime observation** y_i
- We want a mapping $f(x) \rightarrow y$ that **accurately predicts** y_i given $\boldsymbol{x}_i$
- This is a **regression** problem
  - We've tried about a dozen different methods over the years
  - This choice can matter, but features are more important
  - Overall, we recommend **random forests of regression trees**

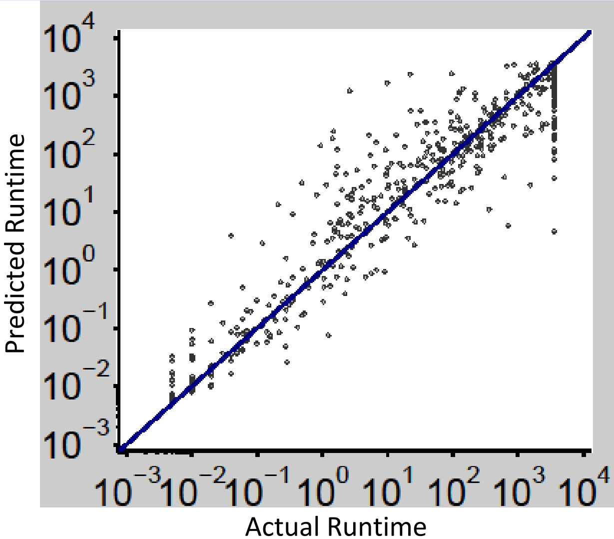
Overall View

We've found that **EHMs work consistently**, across:

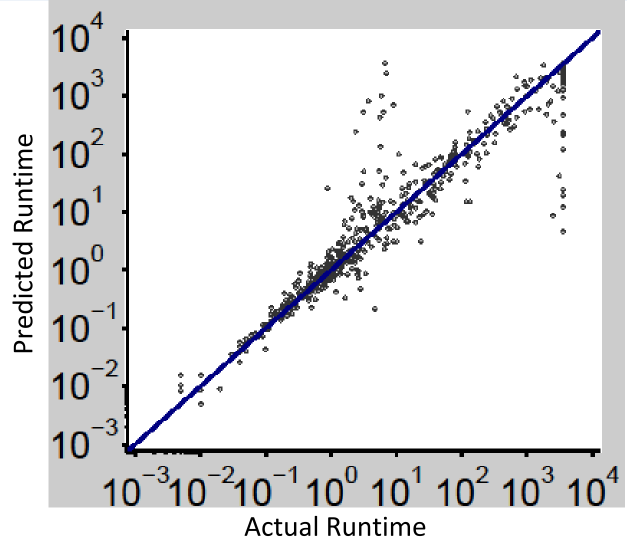
- 4 **problem domains** (with new features in each domain)
  - Satisfiability (SAT)
  - Mixed Integer Programming (MIP)
  - Travelling Salesman Problem (TSP)
  - Combinatorial Auctions
- dozens of **solvers**, including:
  - state of the art solvers in each domain
  - black-box, commercial solvers
- dozens of **instance distributions**, including:
  - major benchmarks (SAT competitions; MIPLIB; …)
  - real-world data (hardware verification, computational sustainability, …)

Examples: EHMs for SAT, MIP

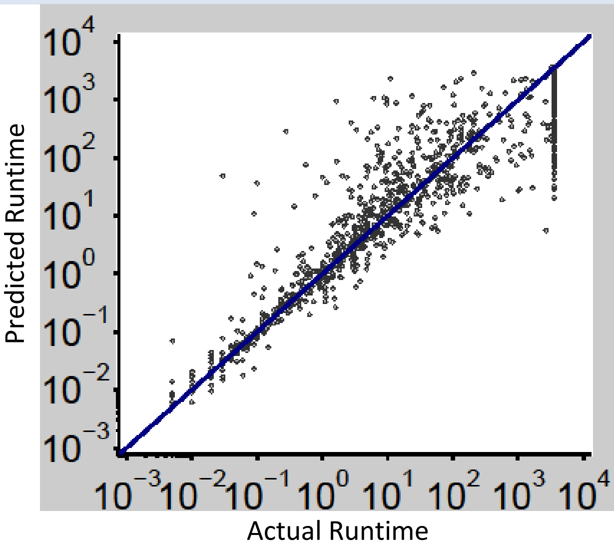
SAT Competition (Random + Handmade + Industrial) data, MINISAT solver
Random Forest (RMSE=0.47)



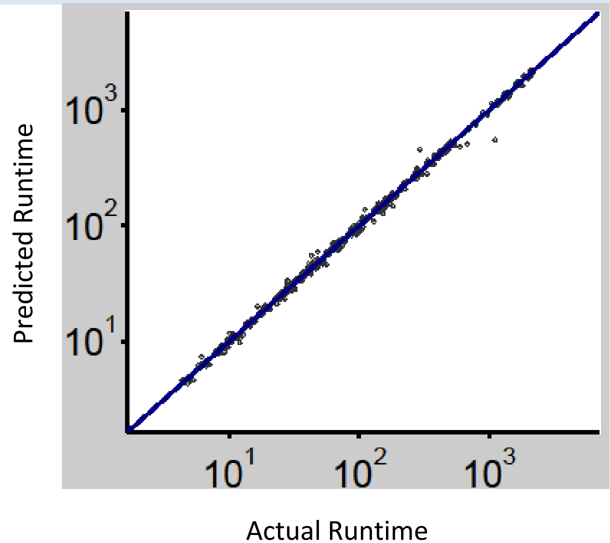
SAT: IBM hardware verification data, SPEAR solver
Random Forest (RMSE=0.38)



MIPLIB data, CPLEX 12.1 solver
Random Forest (RMSE=0.63)



Red Crested Woodpecker habitat data, CPLEX 12.1 solver
Random Forest (RMSE=0.02)



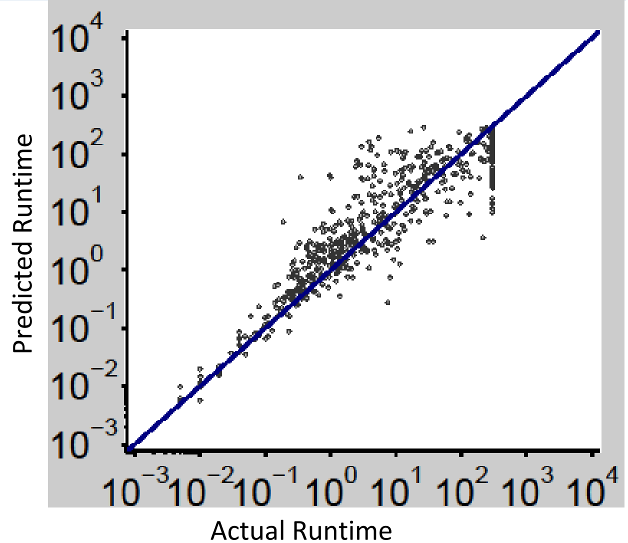
Modeling Algorithm Families

- So far we've considered single, black box algorithms
- What about parameterized **algorithm families**?

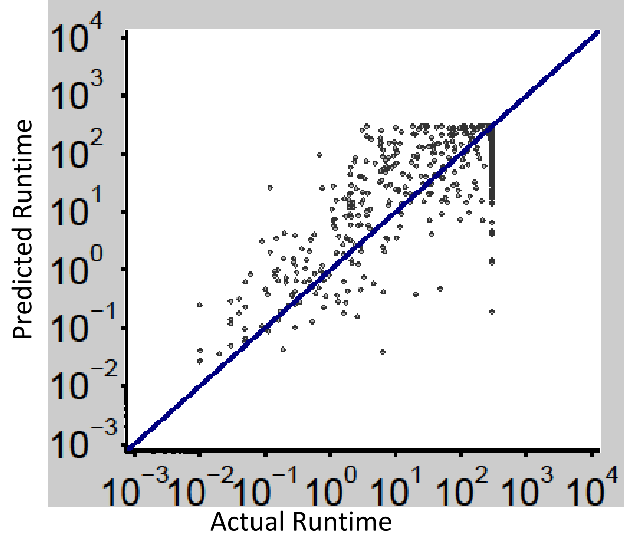
- **Models can be extended to the *sets* of algorithms** described by solvers with parameters that are:
  - continuous or discrete
  - ordinal or categorical
  - potentially conditional on the values of other parameters

- We call full parameter instantiations (i.e., runnable algorithms) **configurations**

SAT: IBM hw verification data, SPEAR
Random Forest (RMSE=0.43)



MIP: MIPLIB data, CPLEX 12.1 solver
Random Forest (RMSE=0.55)

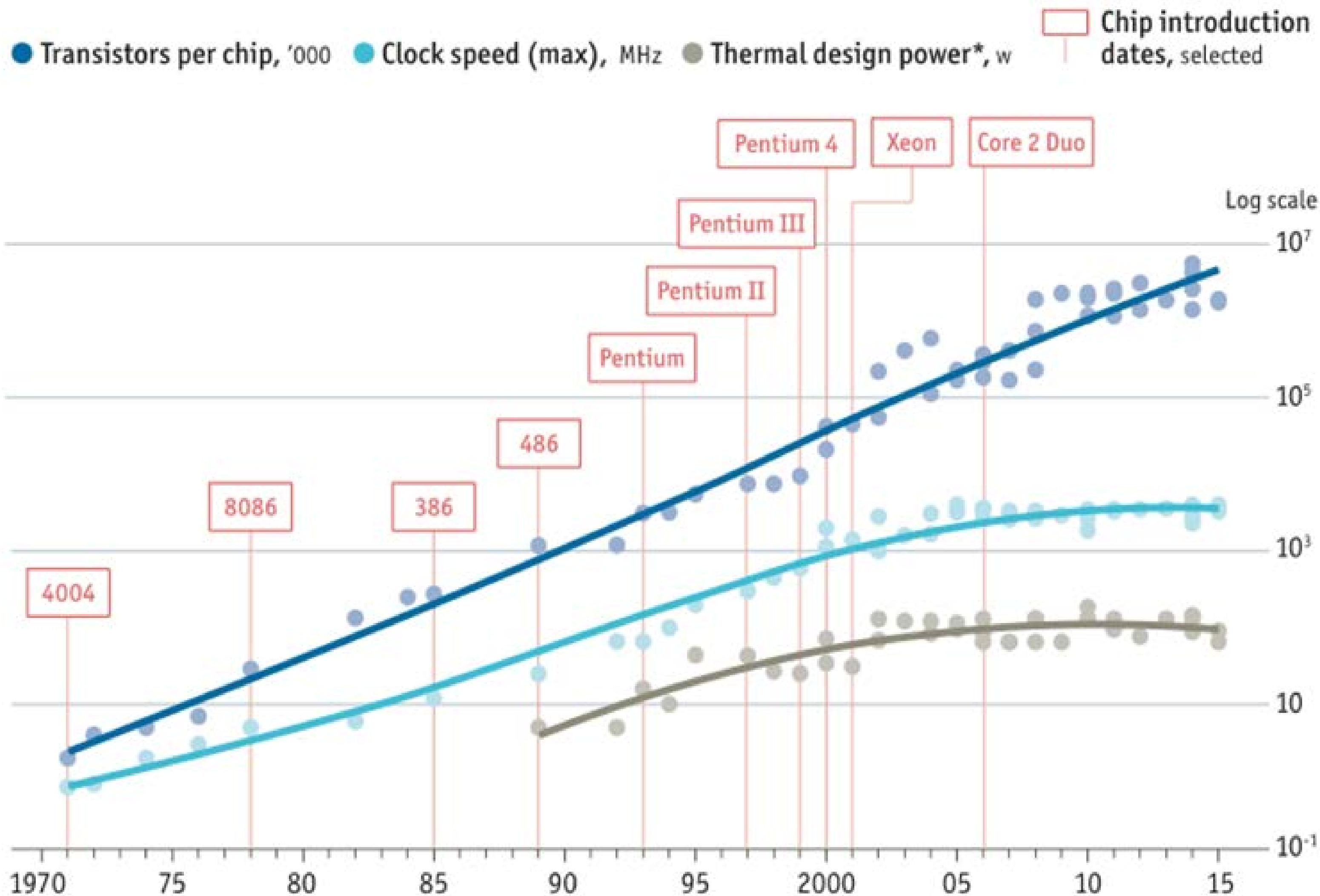


ALGORITHM DESIGN: CONFIGURATION

[Hutter, Hoos, L-B, LION 2011]
[Hutter, Hamadi, Hoos, L-B, CP 2006]
[Hutter, Hoos, L-B, Murphy, GECCO 2009]
[Hutter, Bartz-Beielstein, Hoos, L-B, Murphy, LION 2010]
[Hutter, Hoos, L-B, CPAIOR 2010; AMAI 2010]

Recent, enormous increases in compute power

Approaches that might have seemed crazy in 2000 can ***make a lot of sense in 2016…***



Sources: Intel; press reports; Bob Colwell; Linley Group; IB Consulting; *The Economist*

*Maximum safe power consumption

Deep Optimization

Machine learning

- **Classical approach**
  - Features based on expert insight
  - Model family selected by hand
  - Manual tuning of hyperparameters
- **Deep learning**
  - Very highly parameterized models, using expert knowledge to identify appropriate invariances and model biases (e.g., convolutional structure)
    - "deep": many layers of nodes, each depending on the last
  - Use lots of data (plus "dropout" regularization) to avoid overfitting
  - Computationally intensive search replaces human design

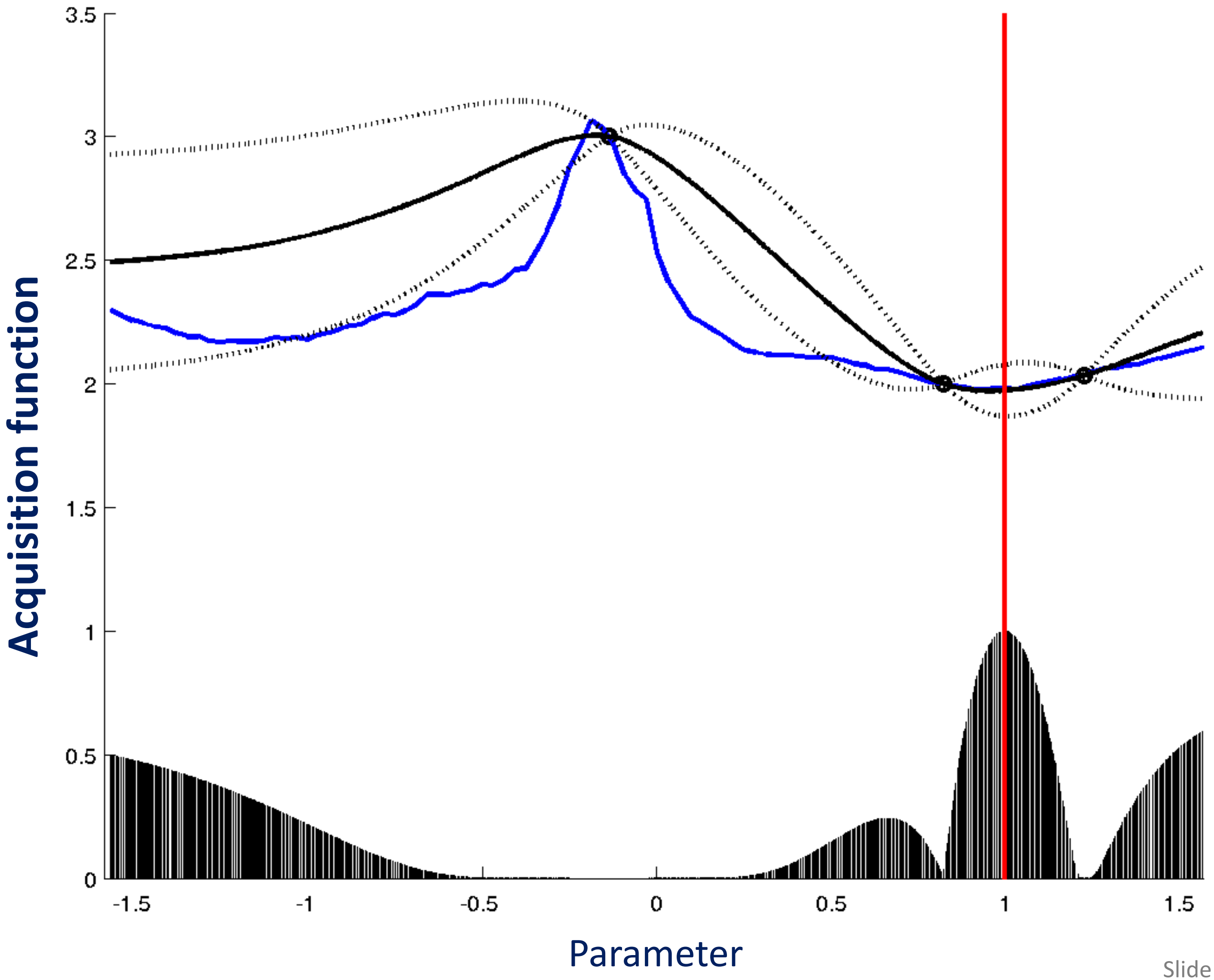
Discrete Optimization

- **Classical approach**
  - Expert designs a heuristic algorithm
  - Iteratively conducts small experiments to improve the design
- **Deep optimization**
  - Very highly parameterized algorithms express a combinatorial space of heuristic design choices that make sense to an expert
    - "deep": many layers of parameters, each depending on the last
  - Use lots of data to characterize the distribution of interest
  - Computationally intensive search replaces human design

Algorithm Configuration

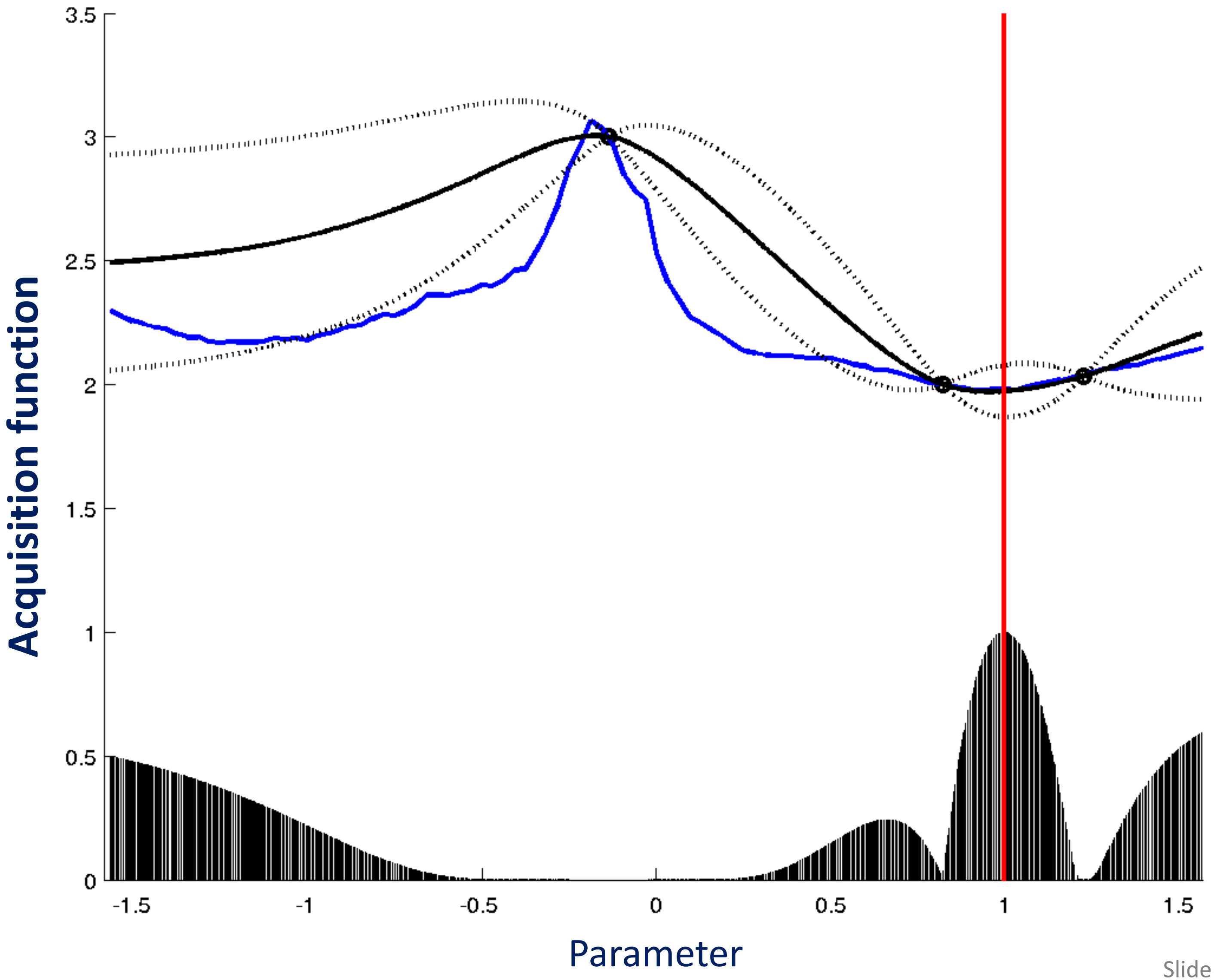
- Our input: **parameters** encoding each design choice considered by the author of our heuristic algorithm
- Our task: the stochastic optimization problem of **finding a parameter configuration** with good performance.
- An interesting **black-box function optimization** problem
  - design dimensions can be continuous; ordinal; categorical
  - extra design dimension: which instance do I test?
  - objective function to be minimized is the same as the cost of evaluating a given point
  - censored sampling: long runs can be terminated
- Best current methods for solving this problem are **based on EHMs**

Visualizing Sequential Model-Based Optimization



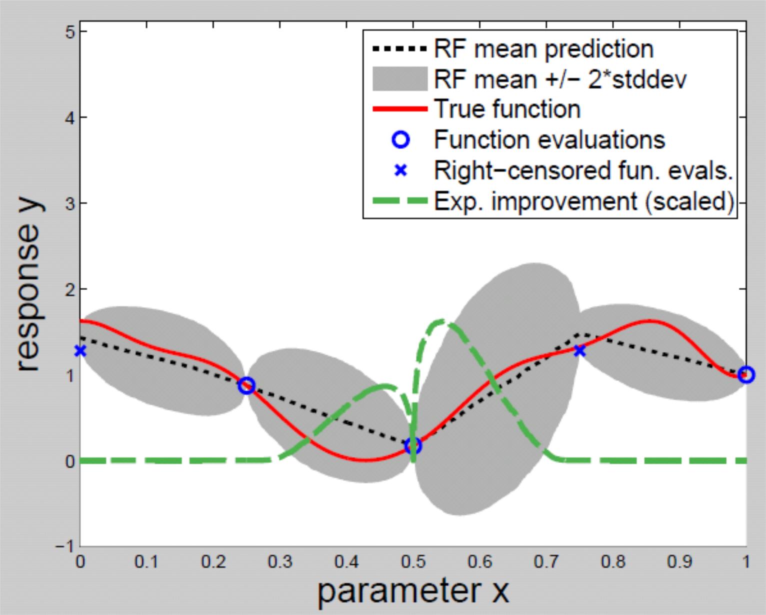
Slide credit: Frank Hutter

Visualizing Sequential Model-Based Optimization



Slide credit: Frank Hutter

Sequential Model-based Algorithm Configuration (SMAC)



Initialize with a single run for the default configuration

repeat

  Learn a random forest model $m : \Theta \times \Pi \rightarrow \mathbb{R}$ from data so far

  Marginalize out instance features: $f(\theta) = \mathbb{E}_{\pi}[m(\theta, \pi)]$

  Find θ that maximizes expected improvement in $f(\theta)$ over incumbent

  Compare θ to the incumbent, updating if it's better.

until *time budget exhausted*

Applications of Algorithm Configuration

Mixed integer programming

Spectrum repacking

Scheduling and Resource Allocation

Game Optimization

Supply Chain Planning & Optimization

Exam Timetabling

Wins in Competitions

SAT: since 2009

IPC: since 2011

ASP: since 2011

Timetabling: 2007

SMT: 2007

Academic Applications by Others

Protein Folding

Game Theory: Kidney Exchange

Computer GO

Linear algebra subroutines

Evolutionary Algorithms

Machine Learning: Classification

ALGORITHM DESIGN: PORTFOLIOS

[Nudelman, L-B, Andrew, Gomes, McFadden, Selman, Shoham, 2003]
[Nudelman, L-B, Hoos, Devkar, Shoham, CP 2004]
[Xu, Hutter, Hoos, L-B, JAIR 2008]
[L-B, Nudelman, Andrew, McFadden, Shoham, IJCAI 2003; CP 2003]
[L-B, Nudelman, Shoham; JACM 2009]
[Xu, Hoos, L-B, AAAI 2010; Xu, Hutter, Hoos, L-B, workshop 2011]
[Lindauer, Hoos, L-B, Schaub, AIJ 2016]

Is Algorithm Configuration Enough?

- There's not (yet) a "best" SAT solver
  - different solvers perform well on **different instances**
  - performance differences between them are **typically very large**
- The effectiveness of EHMs suggests a straightforward solution
  - given a new problem instance, **predict the runtime** of each SAT solvers from an **algorithm portfolio**
  - run the one **predicted to be fastest**
- **SATzilla**: a portfolio-based algorithm selector for SAT (2003-present)

Algorithm Selection

- Since proposing it, we've improved the approach to:
  - allow **randomized and incomplete** algorithms as component solvers
  - include **presolvers** that run for a short, fixed time
  - optimize for complex **scoring functions** beyond runtime
  - automate the **construction of the selector** given data
    - e.g., pre-solver selection; component solver selection
    - again, "deep optimization"
- We can also improve by moving to a different ML framework
  - **cost-sensitive classification** directly selects best-performing solver
  - doesn't need to predict runtime
- Or, just run all algorithms in the portfolio together **in parallel**

Success of SATzilla

- 2003 **SAT Competition**
  - placed second and third in several categories
- 2007 and 2009 **SAT Competitions**
  - winning five medals each time
- 2012 **SAT Challenge**
  - eligible to enter four categories
  - placed first, first, first, second
- Then, portfolios **banned** from competitions ☺

- SATzilla's success demonstrates the effectiveness of **automated, statistical methods** for combining solvers
  - including "uncompetitive" solvers with poor average performance
- Our approach is **entirely general**
  - likely to work well for other problems with high runtime variation
  - caveat: each domain needs instance features

Hydra: Automatic Portfolio Synthesis

- So far we've assumed that we start out with a manageable set of **relatively uncorrelated solvers**
  - what if all we start out with is a **huge, deep parameter space**?
    - top level parameter may encode for which of many different solvers to use
  - want a "**deep optimization**" approach that works entirely automatically

- Hydra: augment an additional portfolio *P* by targeting instances **on which *P* performs poorly**
- Give SMAC a dynamic performance metric:
  - performance of alg *s* when *s* outperforms *P;* performance of *P* otherwise
  - Intuitively: *s* scored for **marginal contribution** to *P*

ALGORITHM DESIGN:

A Case Study on Spectrum Repacking

[Frechette, Newman, L-B, AAAI 2016; ongoing work]

FCC's "Incentive Auction"

FCC's complex incentive auction could net more than $30 billion

Mike Snider, USA TODAY 7:19 a.m. EDT March 30, 2016

(Photo: MANDEL NGAN, AFP/Getty Images)

f 33 CONNECT | TWEET | in 5 LINKEDIN | 1 COMMENT | EMAIL | MORE

The most sophisticated and complex spectrum auction ever conducted by the Federal Communications Commission is officially underway.

When the entire process comes to an end more than three years from now, big wireless carriers that provide most of our smartphone access should have more bandwidth to delivery services to mobile-hungry consumers.

TV broadcasters by Tuesday night must have made official their intentions to accept the FCC's opening price for the rights to the spectrum they currently use for digital TV broadcasts. Once the agency knows how much spectrum can be made available in this "reverse auction," then, in a few months, the FCC will open up the bidding in the "forward auction" in which companies such as AT&T and Verizon can bid onthe reallocated spectrum in each of 400-plus localities.

MORE STORIES

Thanks to all those who helped make this work possible!

Student leads on the project: **Neil Newman** , **Alexandre Fréchette** 

Further students who made contributions to software:
Nick Arnosti; Emily Chen; Ricky Chen;
Paul Cernek; Guillaume Saulnier Comte; Alim Virani

Others (then) at UBC:

- Chris Cameron
- **Holger Hoos**
- **Frank Hutter**
- Ashiqur Khudabukhsh
- Steve Ramage
- James Wright
- Lin Xu

Auctionomics:

- Ulrich Gall
- Jon Levin
- **Paul Milgrom**
- **Ilya Segal**
- Karen Wrege

FCC & associates:

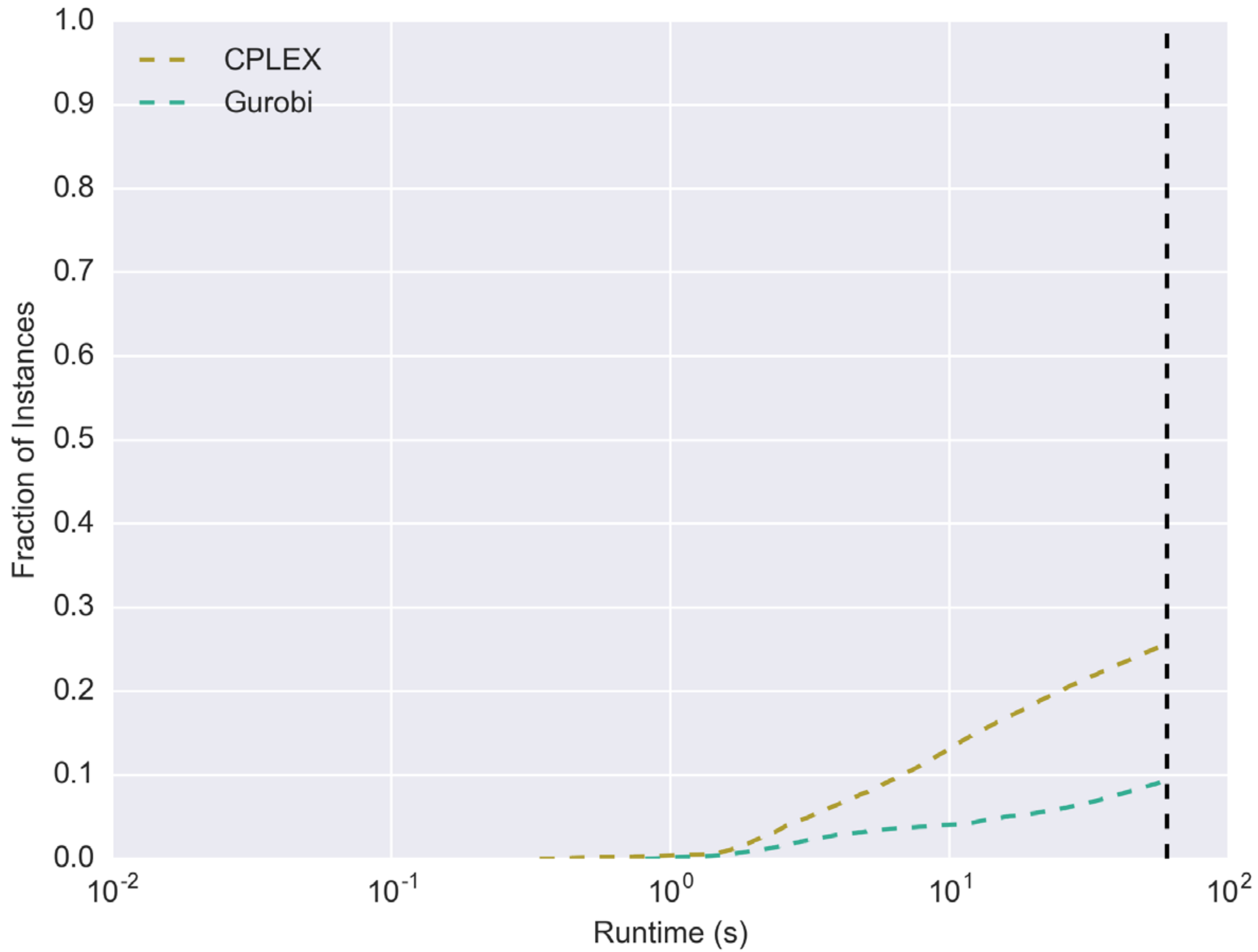
- **Melissa Dunford**
- Gary Epstein
- Karla Hoffman
- **Sasha Javid**
- Evan Kwerel
- Rory Molinari
- Brett Tarnutzer
- Venkat Veeramneni

Funding from: Auctionomics; Compute Canada; NSERC Discovery; NSERC E.W.R. Steacie

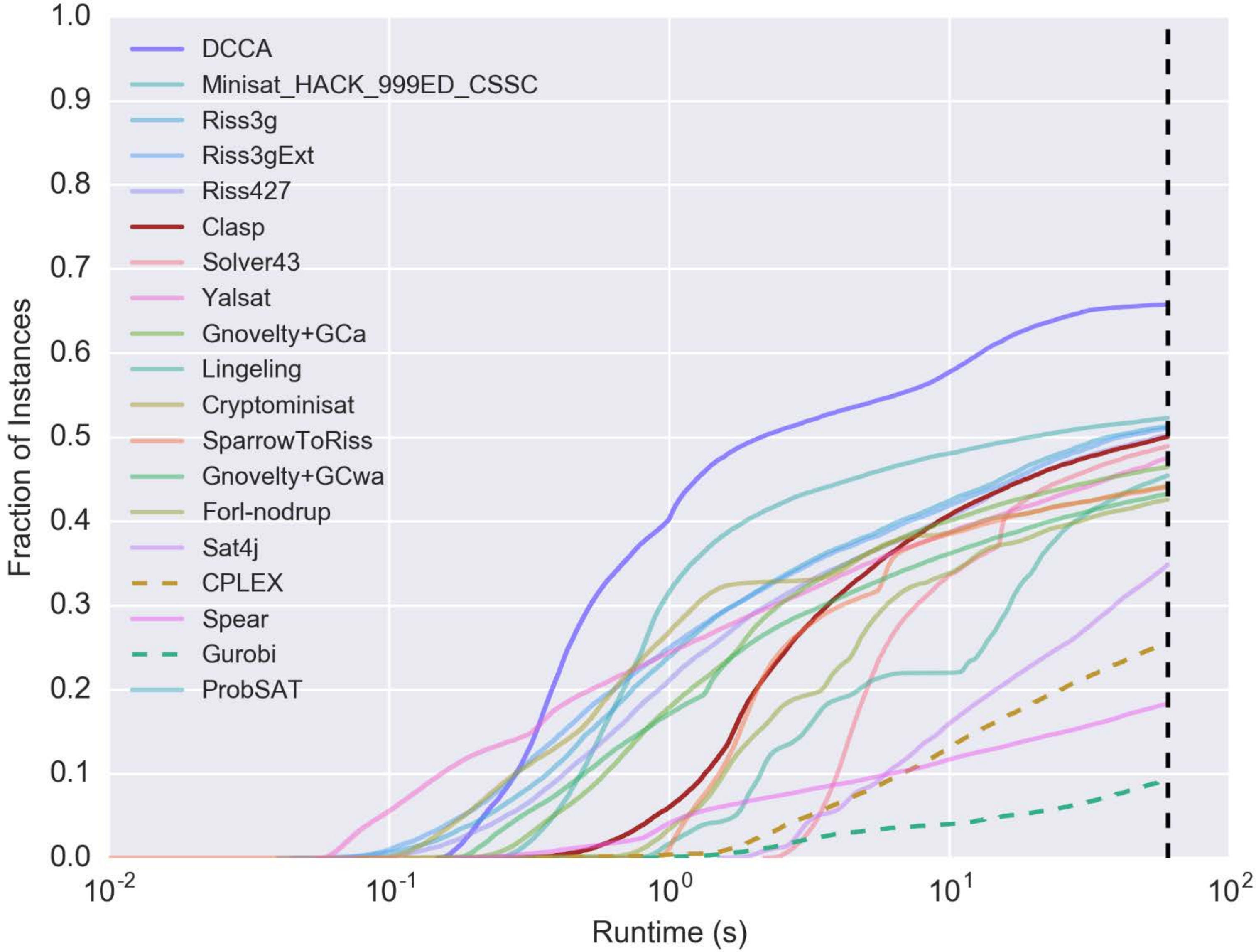
Building (& Evaluating) a Feasibility Tester

- **Data** generated Nov 2015 – Feb 2016 using
  - the FCC's Nov 2015 interference constraints
  - the FCC's "smoothed ladder" simulator
  - varying **simulation assumptions**:
    - how much spectrum is cleared: 126 MHz; 108 MHz; 84 MHz
    - which stations opt to participate
    - these stations' valuations
    - the timeout given to SATFC in the simulation (1; 5; 10; 60 min)
- 128 **auctions** $\Rightarrow$ 1.4 M **instances**
  - 6,128 – 17,764 instances per auction
    - all not solvable by directly augmenting the previous solution
    - about 20% of the problems encountered in full simulations
  - split auctions 102/26 into training/test sets
- Our goal: solve problems within a **one-minute cutoff**

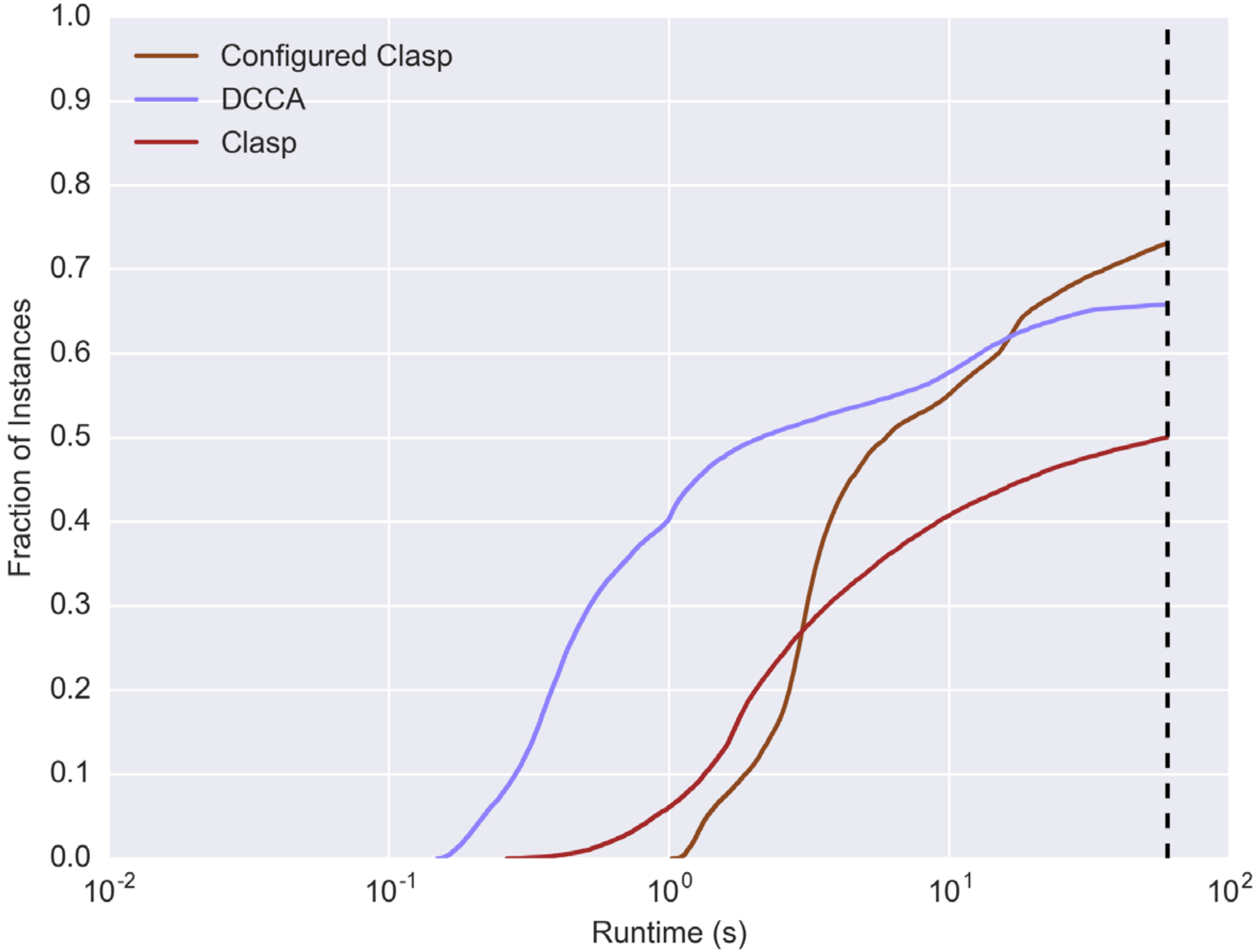
Feasibility Testing via MIP Encoding



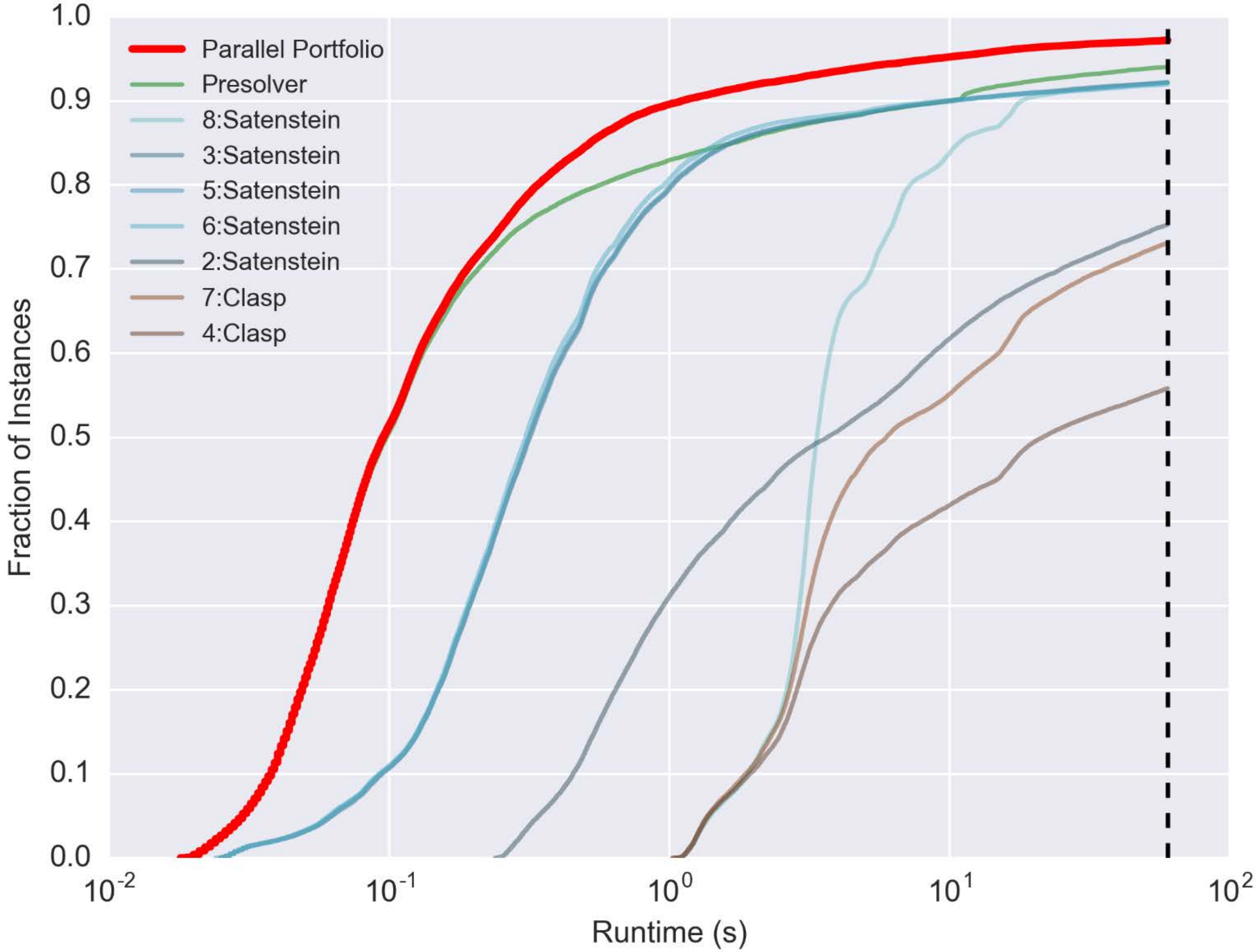
Feasibility Testing via SAT Encoding



Best Configured Solver



Performance of the Algorithm Portfolio



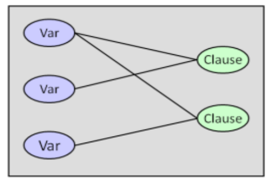
BEYOND WORST-CASE COMPLEXITY:

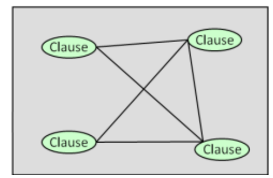
A Case Study on Characterizing SAT Solver Performance On Uniform Random 3-SAT: Beyond the Clauses-to-Variables Ratio

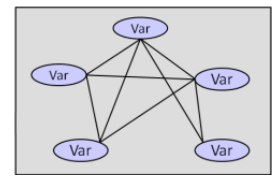
[L-B, Nudelman, Shoham: CP 2002; JACM 2009]
[Nudelman, L-B, Hoos, Devkar, Shoham: CP 2004]
[Xu, Hoos, L-B: CP 2007; AAAI 2012]
[Hutter, Xu, Hoos, L-B: CACM 2014]

SAT Instance Features

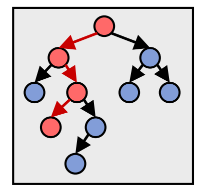
- Problem **Size** (clauses, variables, clauses/variables, …)
- **Syntactic** properties (e.g., positive/negative clause ratio)
- Statistics of various **constraint graphs**
  - factor graph
  - clause–clause graph
  - variable–variable graph



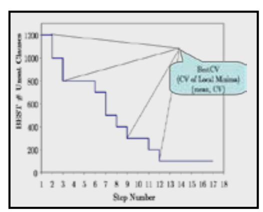


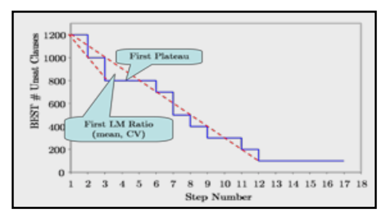


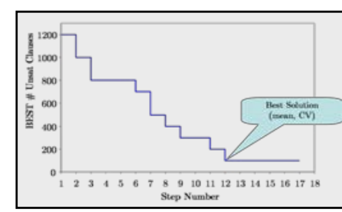
- Knuth's **search space size** estimate



- Cumulative number of **unit propagations** at different depths (SATz heuristic)
- **Local search probing**



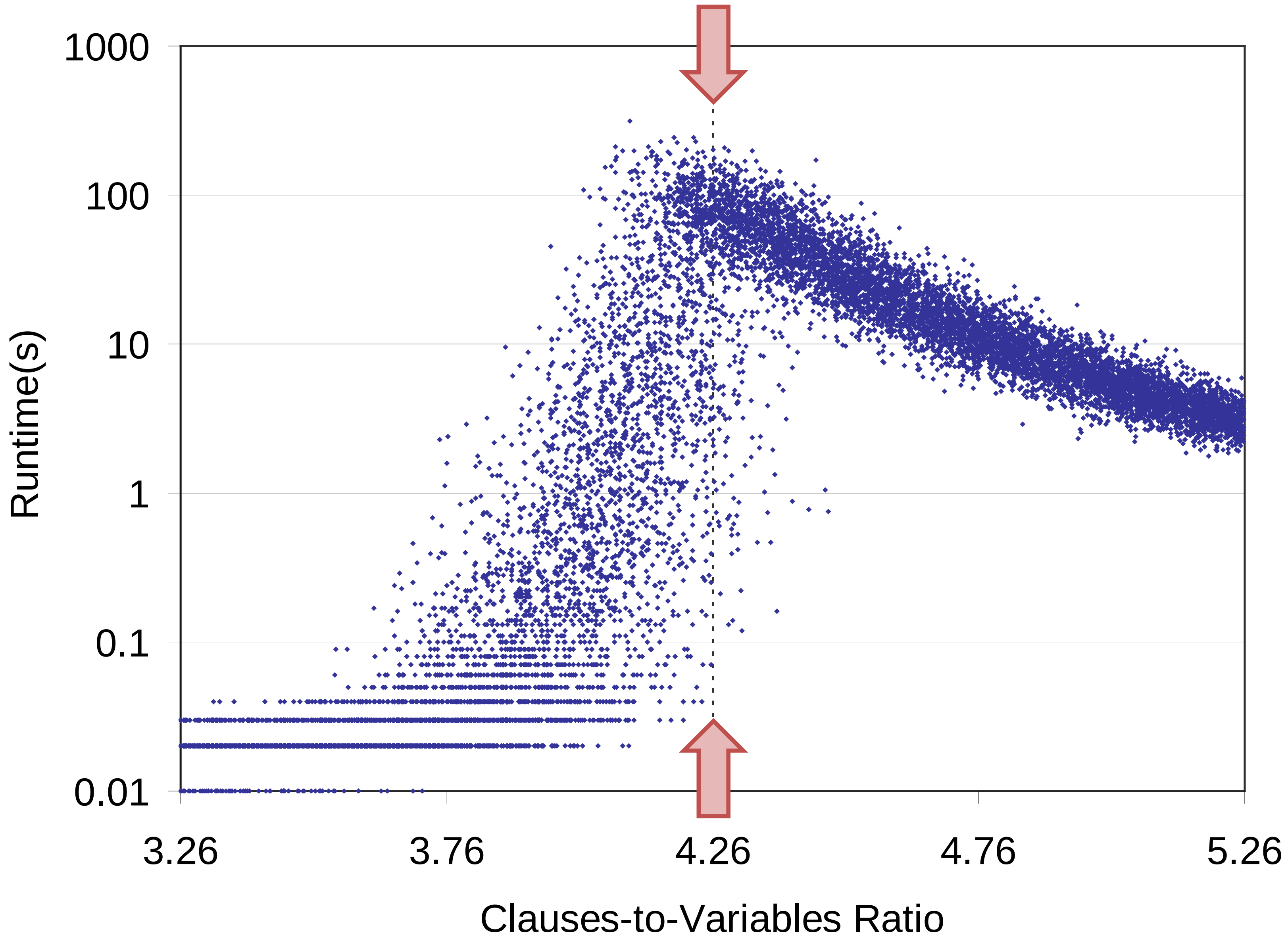




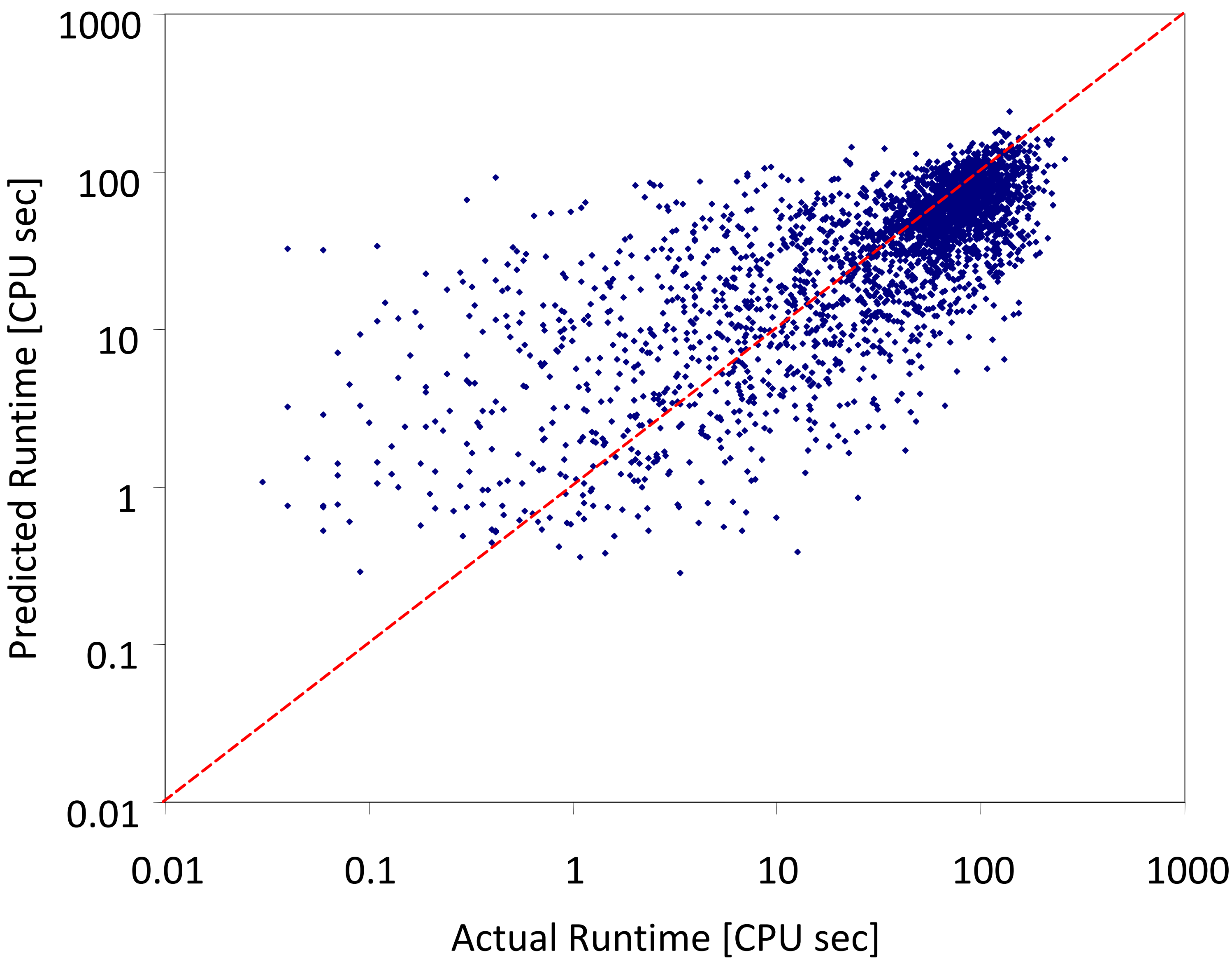
- **Linear programming** relaxation

$$\begin{aligned} \text{maximize:} \quad & \sum_{k\in C}\left(\sum_{i\in L,\, i\in k} v_i + \sum_{j\in \overline{L},\, i\in k}(1-v_j)\right) \\ \text{subject to:} \quad & \sum_{i\in k,\, i\in L} v_i + \sum_{j\in k,\, j\in \overline{L}}(1-v_j) \geq 1 \qquad \forall k\in C \\ & v_i \in \{0,1\} \qquad \forall i \end{aligned}$$

Example: Uniform-Random 3-SAT at Phase Transition



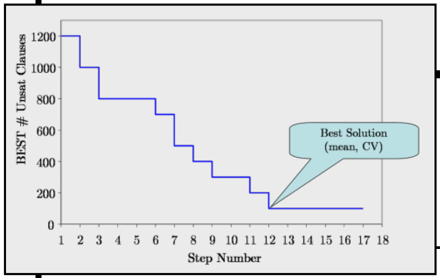
Fixed Ratio Prediction (Kcnfs)



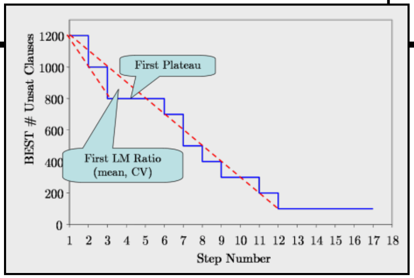
Feature Importance – Fixed Ratio

Variable	Cost of Omission
SapsBestSolMean^2	100
SapsBestSolMean · MeanDPLLDepth	74
GsatBestSolCV · MeanDPLLDepth	21
VCGClauseMean · GsatFirstLMRatioMean	9

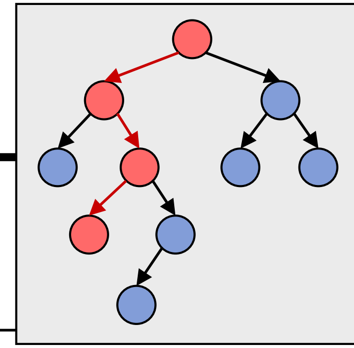
Feature Importance – Fixed Ratio



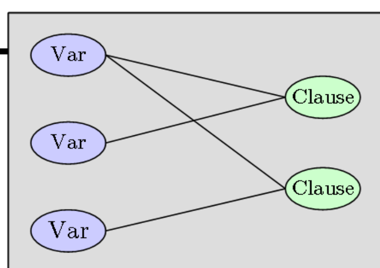
Variable	Cost of Omission
SapsBestSolMean^2	100
SapsBestSolMean · MeanDPLLDepth	74
GsatBestSolCV · MeanDPLLDepth	21
VCGClauseMean · **GsatFirstLMRatioMean**	9



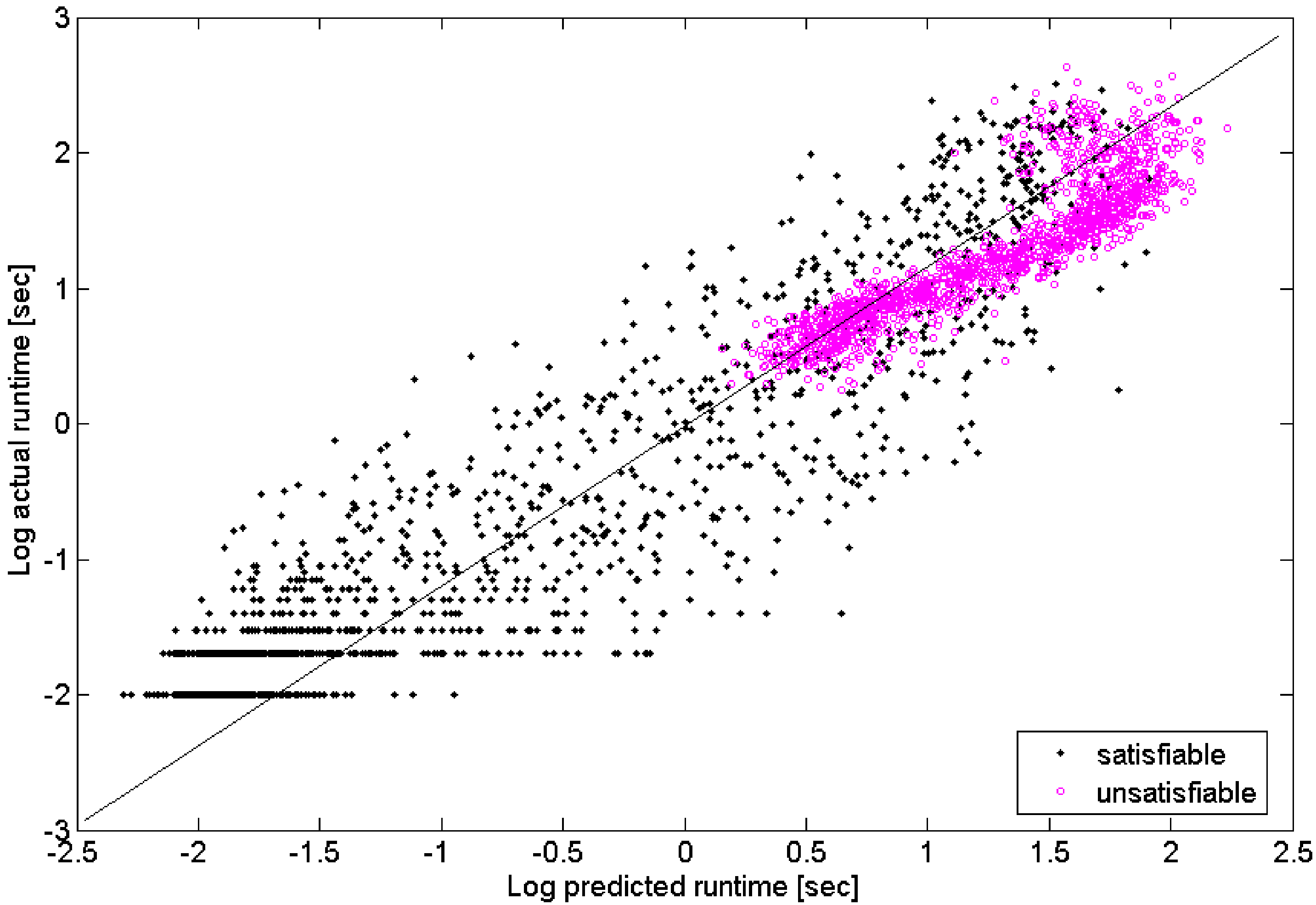
Feature Importance – Fixed Ratio



Variable	Cost of Omission
SapsBestSolMean^2	100
SapsBestSolMean · **MeanDPLLDepth**	74
GsatBestSolCV · **MeanDPLLDepth**	21
VCGClauseMean · GsatFirstLMRatioMean	9



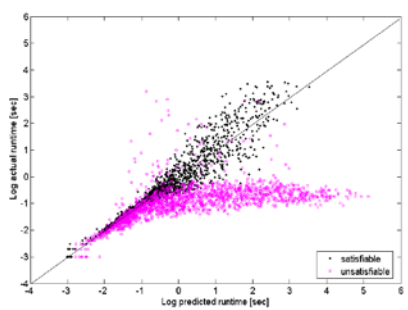
Uniform-Random 3-SAT, Variable Ratio



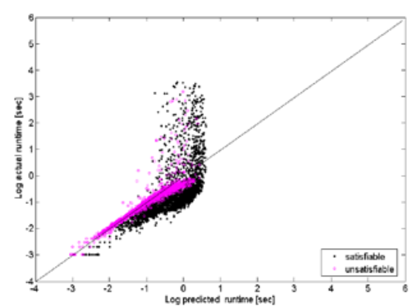
Predicted vs. Actual Log Runtime, SATZ on Uniform Random 3SAT, variable ratio

Hierarchical Hardness Models

- Conditioning on satisfiability of the instance: clauses/variables unimportant; **single-feature models become sufficient**
  - Satisfiable: **local search probing**
  - Unsatisfiable: **search space size**
- **Hierarchical hardness model** [Xu, Hoos, Leyton-Brown, 2007]:
  1. Predict satisfiability status
  2. Use this prediction as a feature to combine the predictions of SAT-only and UNSAT-only models
- Not necessarily easy: SAT-only and UNSAT-only models can make **large errors when given wrong data**

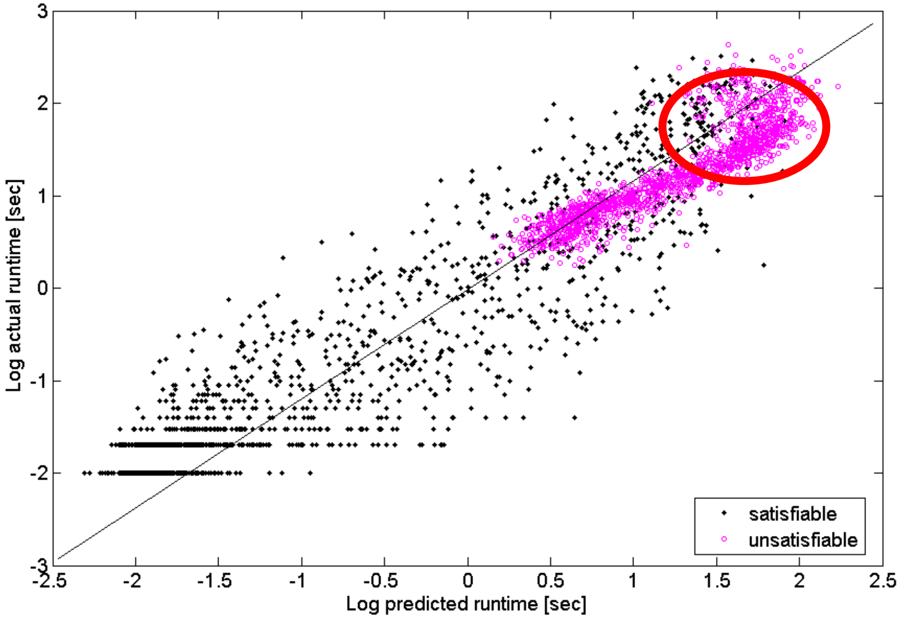


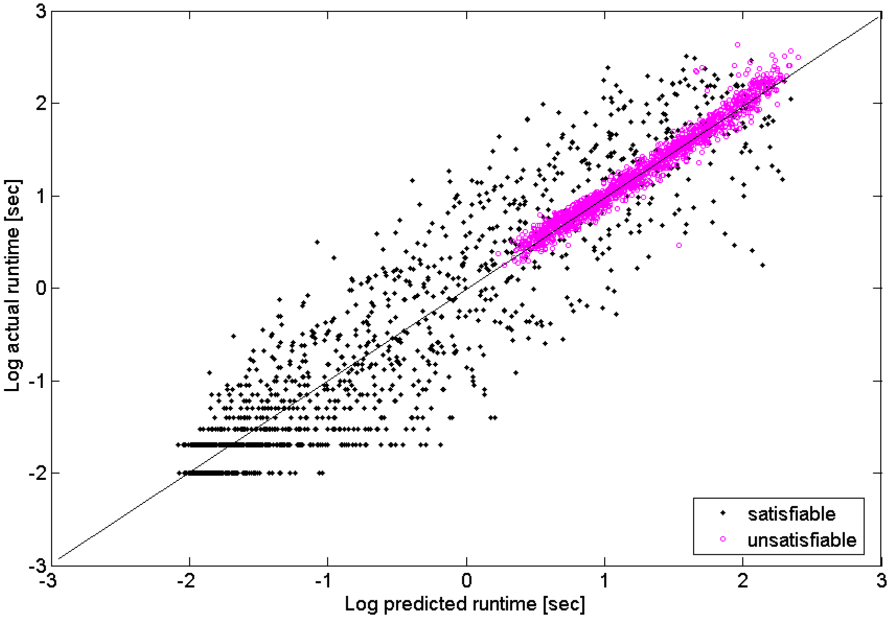
SAT-only



UNSAT-only

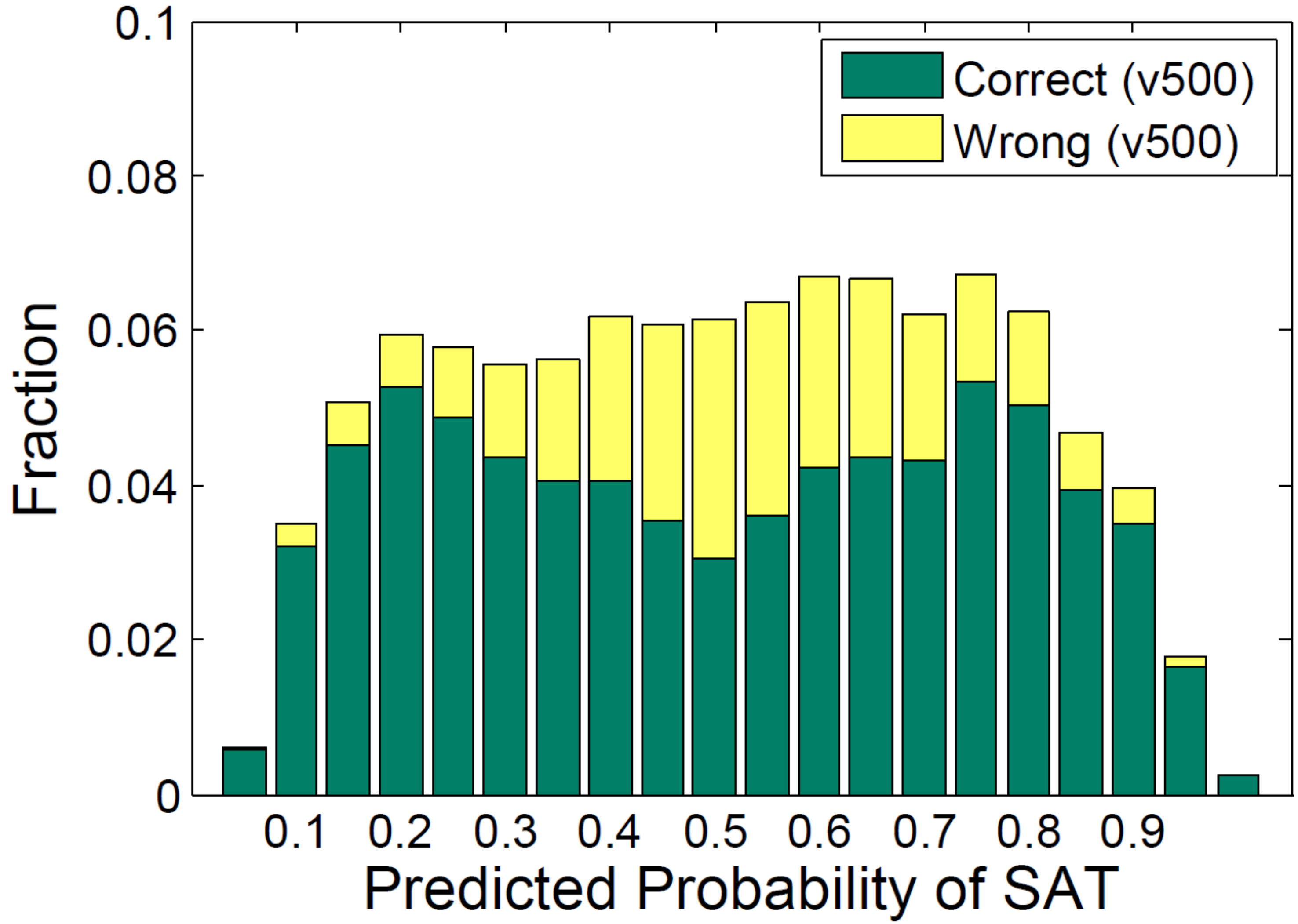
Empirical Performance of HHMs





Predicted vs. Actual Log Runtime, SATZ on Uniform Random 3SAT, variable ratio

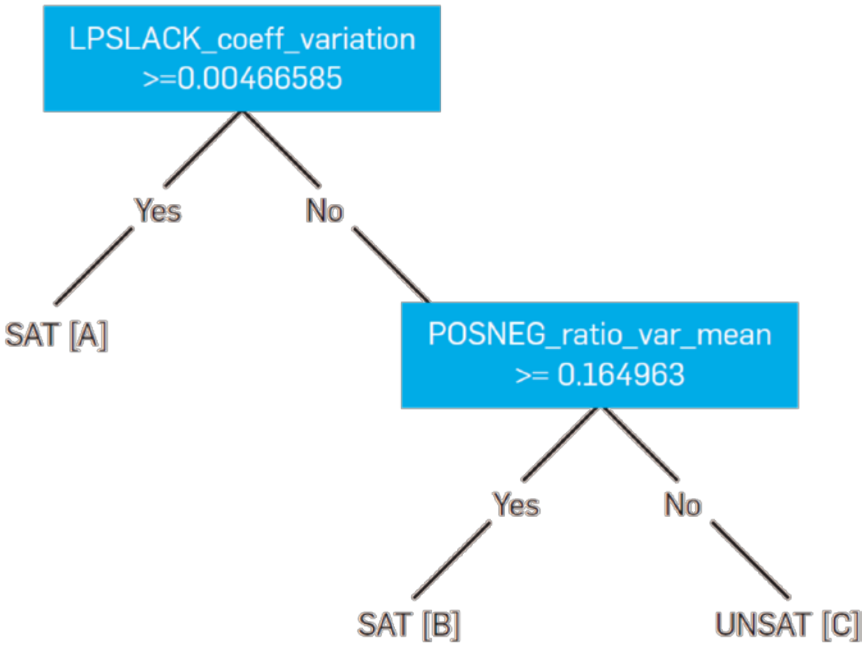
Predicting Satisfiability Status (fixed-ratio 3-SAT)

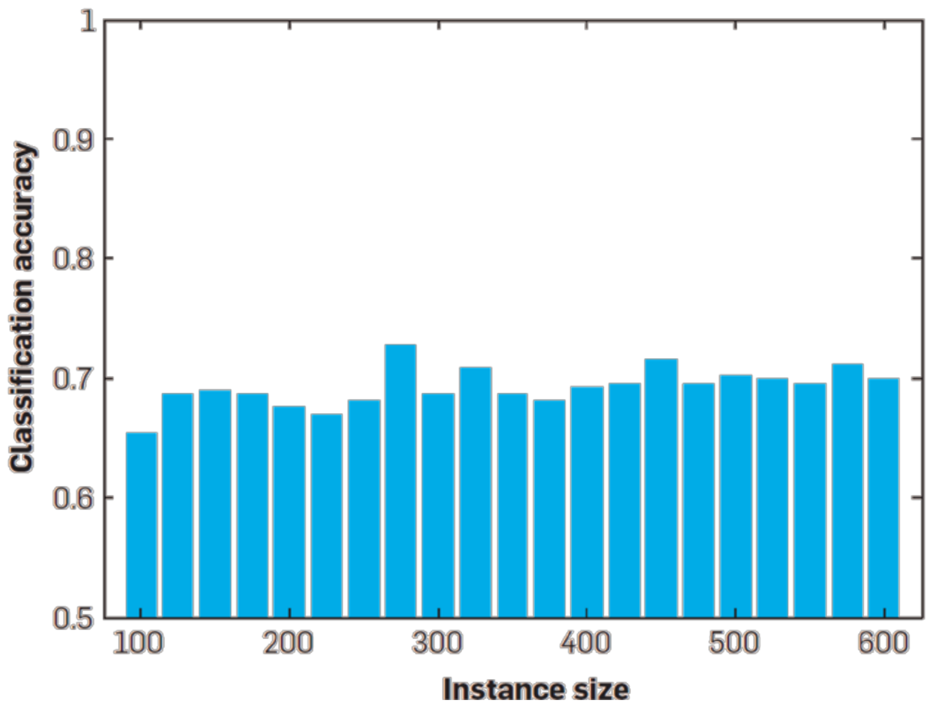


Can We Really Predict Satisfiability Status?

- Consider phase-transition instances varying from **100 variables** (solvable in milliseconds) to **600 variables** (solvable in a day).
  - Does prediction accuracy fall to random guessing on larger problems?
  - If not, can we identify an easily comprehensible model that would offer theoretical insight?
- **Restrict models** in three ways:
  - train only on **100-variable** instances
  - consider only decision trees with at most **two decision nodes**
  - omit all **probing features**
    - disproportionately effective on small instances
    - based on complex, heuristic algorithms

A Simple Model Beats Random Guessing



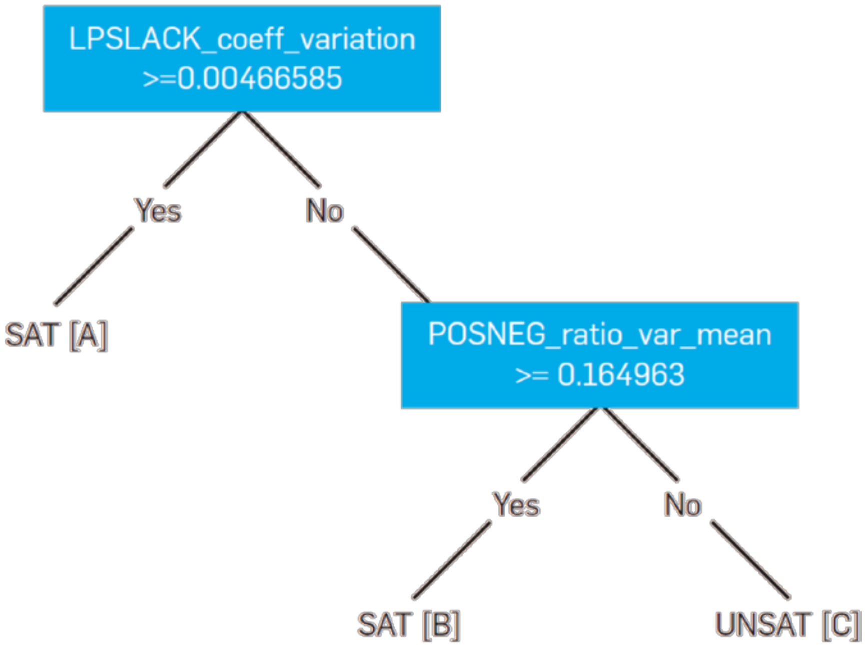


Predictive accuracies for instances falling into the three regions were between 60% and 70% [A]; a bit more than 50% [B]; and between 70% and 80% [C].

This model was trained **only on 100-variable problems**.

No evidence that **accuracy falls with size** (pairwise Mann-Whitney U tests)

A Simple Model Beats Random Guessing



LPSLACK_coeff_variation

- based on SAT's LP relaxation
- for each i with LP solution value $S_i \in [0,1]$, LPSLACK$_i$ is defined as $\min\{1 - S_i, S_i\}$
- LPSLACK_coeff_variation is the coefficient of variation (standard deviation divided by mean) of the vector LPSLACK

POSNEG_ratio_var_mean

- For each variable i with P_i positive occurrences and N_i negative occurrences, POSNEG _ ratio _ var$_i$ is $\left|0.5 - \frac{P_i}{P_i+N_i}\right|$.
- POSNEG_ratio_var_mean is then the average over elements of the vector

Both features **normalized to have mean 0, standard deviation 1** on the training set.

To evaluate on a test set instance of a new size:

- randomly sampled many instances of that size
- estimated new normalization factors
- used these factors to compute the features for the test instance

Conclusions

- **Empirical Hardness Models**
  - a statistically rigorous approach to characterizing the difficulty of solving a given family of problems using available methods
  - surprisingly **effective in practice**, across various domains
- EHMs are also useful for algorithm design
  - model-based **algorithm configuration**
  - automatic **design of algorithm portfolios**
- **Analysis of learned models** can open avenues for theoretical investigations beyond the worst case