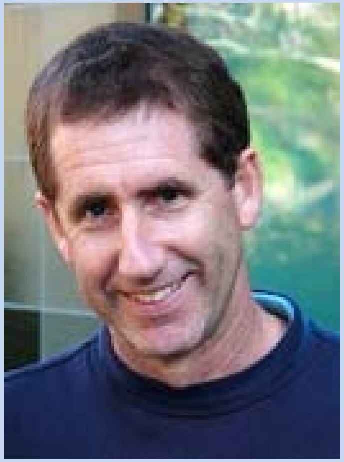
Empirical Hardness Models

A Statistical Approach to Describing Hardness in Practice

Kevin Leyton-Brown
Computer Science Department
University of British Columbia

This Talk Describes 10 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, 2002; 2009]
[Nudelman, L-B, Hoos, Devkar, Shoham, 2004]
[Xu, Hoos, L-B, 2007]
[Hutter, Xu, Hoos, L-B, ongoing work]

Motivating Question

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

The best available methods tend

- to offer **no interesting theoretical guarantees**
- work **astoundingly well** in practice
- often exhibit **exponentially varying performance** (e.g., milliseconds to days) even on fixed-size problems

Our Key Finding

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.

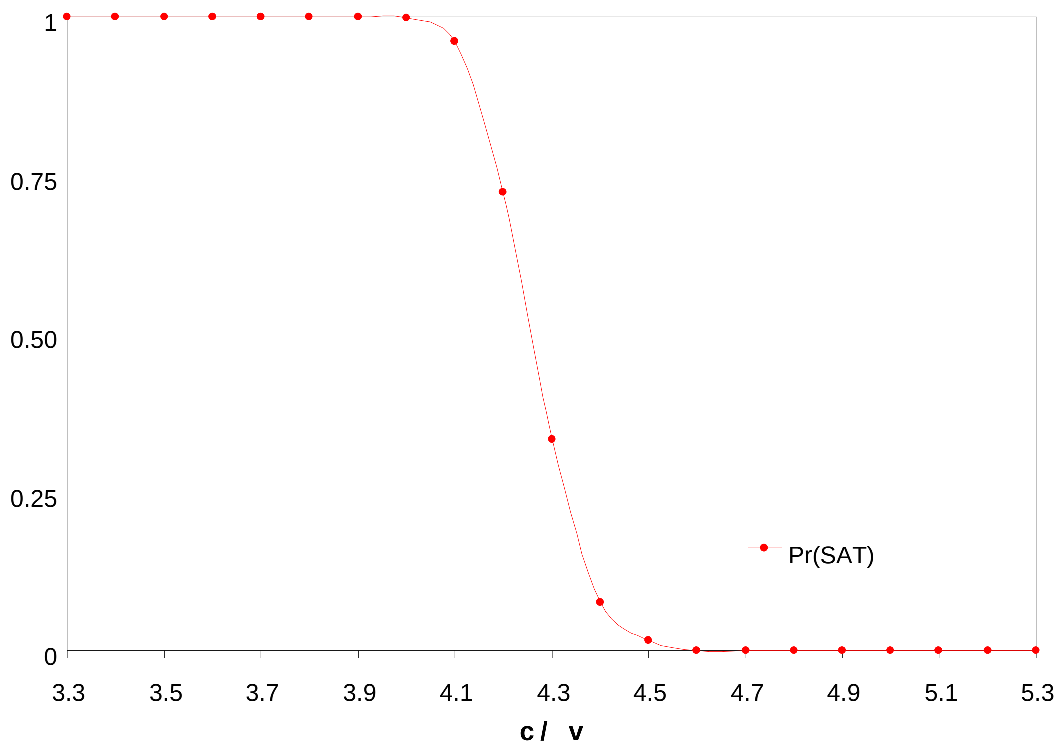
I suspect that many here prefer complexity-theoretic analysis to statistical methods that aim "only" to work in practice

Why I think you should still care:

- the success of statistical methods points to patterns in algorithm performance that aren't yet captured theoretically

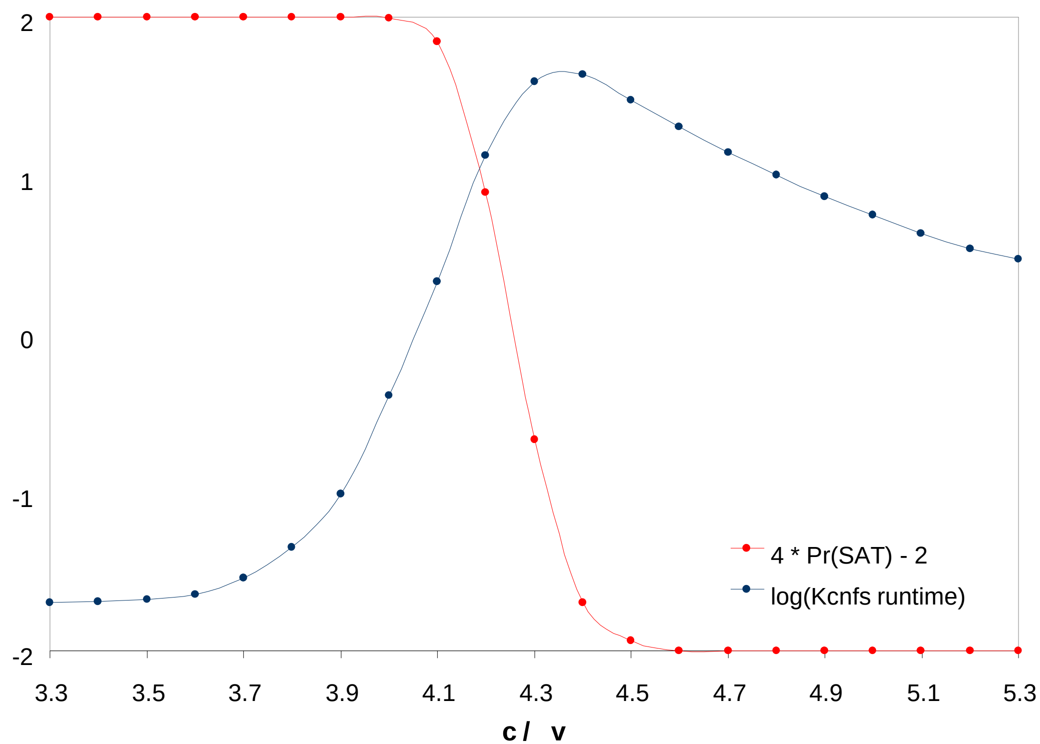
Phase Transitions for SAT

- **Uniform-random 3-SAT:** phase transition in probability of solvability at clauses / variables ≈ 4.26

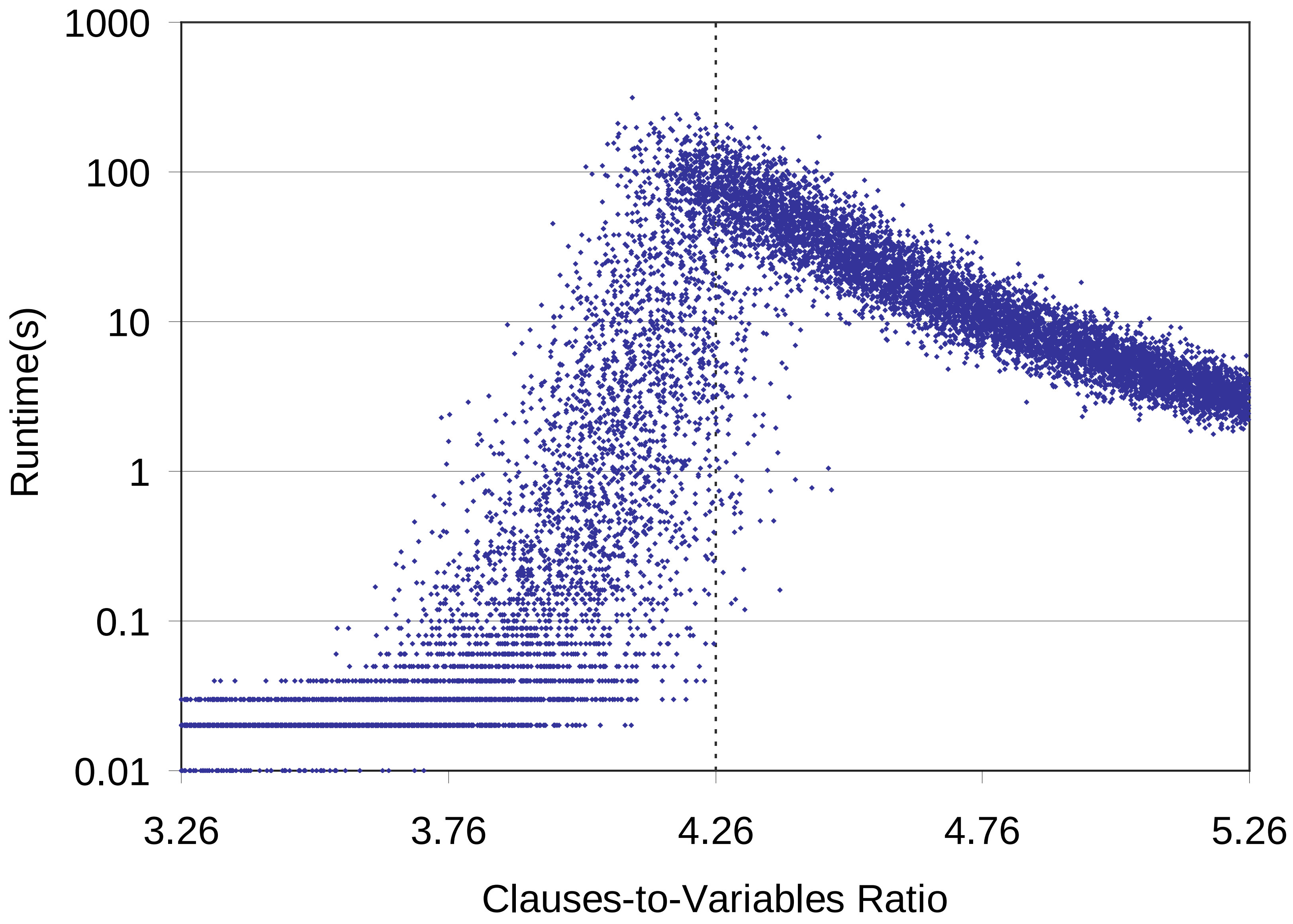


Phase Transitions for SAT

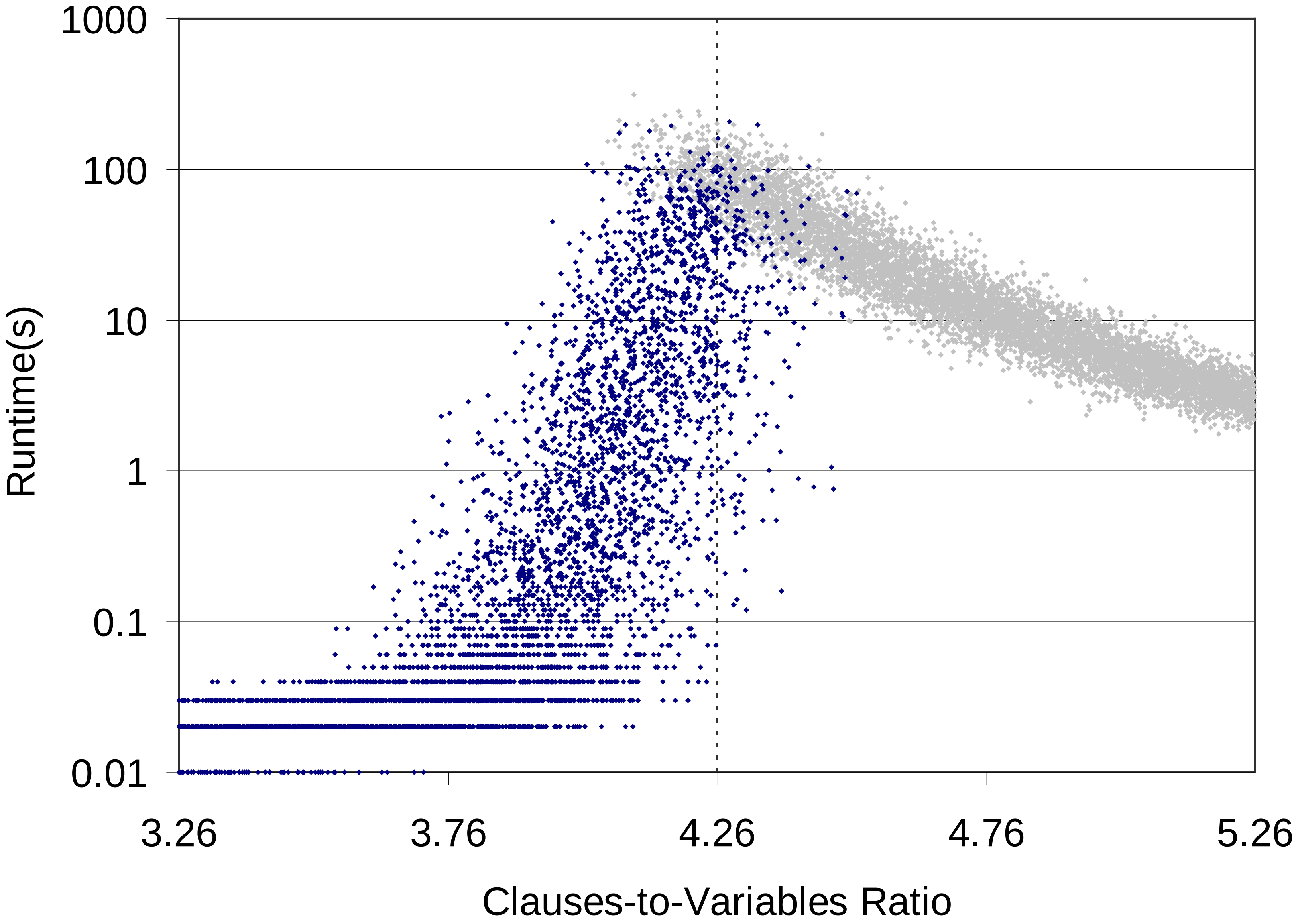
- **Uniform-random 3-SAT:** phase transition in probability of solvability at clauses / variables ≈ 4.26
- Corresponding **easy–hard–less hard** transitions discovered in the behavior of DPLL-type solvers [Cheeseman et al, 1991; Selman et al., 1996]
  - Spawned a new enthusiasm for using empirical methods to study algorithm performance



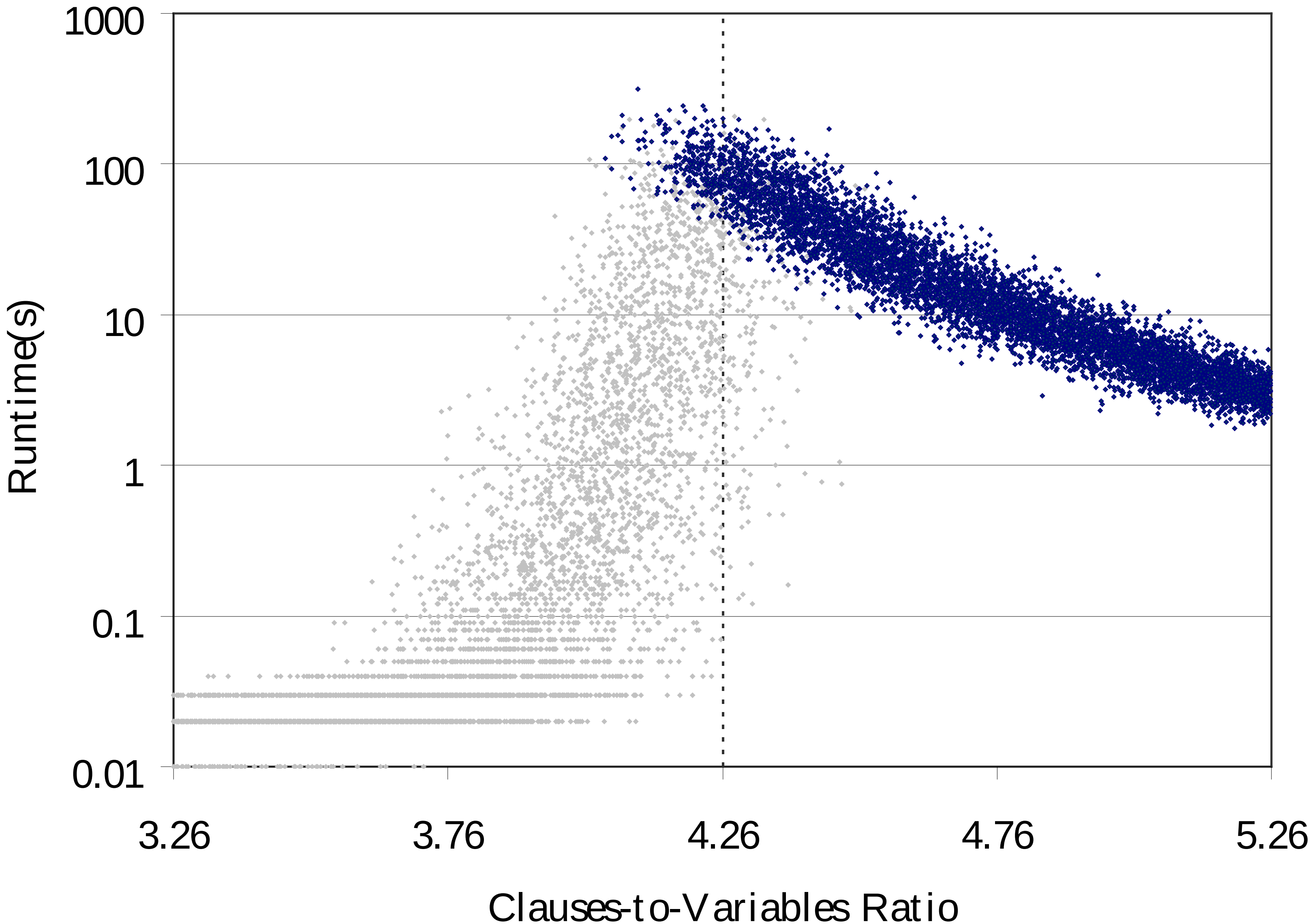
Kcnfs Data


1000
100
10
1
0.1
0.01
Runtime(s)
3.26
3.76
4.26
4.76
5.26
Clauses-to-Variables Ratio

Kcnfs Data


1000
100
10
1
0.1
0.01
Runtime(s)
3.26
3.76
4.26
4.76
5.26
Clauses-to-Variables Ratio

Kcnfs Data


1000
100
10
1
0.1
0.01
Runtime(s)
3.26
3.76
4.26
4.76
5.26
Clauses-to-Variables Ratio

Where We Stand

Probability of solvability **correlates strongly** with instance hardness in practice

- However, lots of residual variance
- There's much more going on here

Is it possible to make **more accurate predictions**?

- Idea: use machine learning methods to look for patterns

EMPIRICAL HARDNESS MODELS:

A Case Study on Characterizing Algorithm Performance Beyond the Clauses-to-Variables Ratio

[L-B, Nudelman, Shoham, 2002; 2009]

[Nudelman, L-B, Hoos, Devkar, Shoham, 2004]

[Xu, Hoos, L-B, 2007]

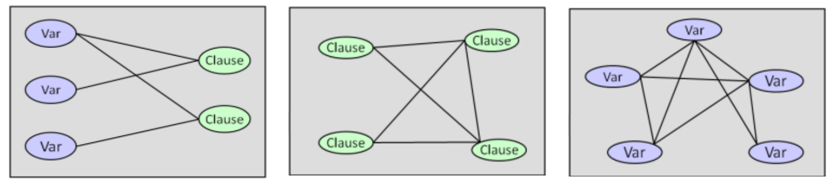
Empirical Hardness Models

- 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

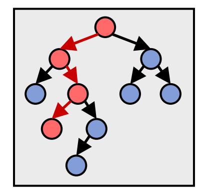
- We want a mapping $f(x) \mapsto y$ that **accurately predicts** y_i given $\boldsymbol{x}_i$
  - This is a **regression** problem
    - We've tried about a dozen different methods over the years
    - This choice (sometimes) matters, but features are more important
  - First, let's consider a straightforward, tractable, and often very effective approach: **basis function ridge regression**

SAT Instance Features

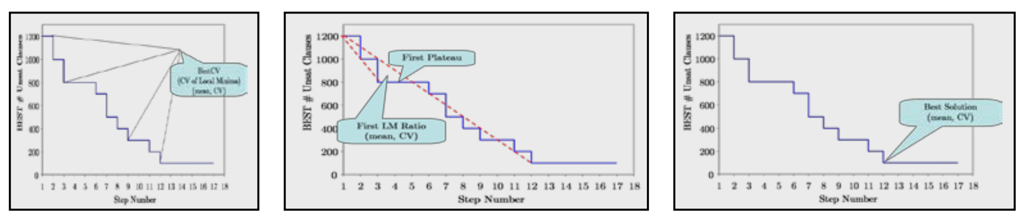
- Problem **Size** (clauses, variables, clauses/variables, …)
- **Syntactic** properties (e.g., positive/negative clause ratio)
- Statistics of various **constraint graphs**
  - factor graph
  - clause–clause graph
  - variable–variable graph



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



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

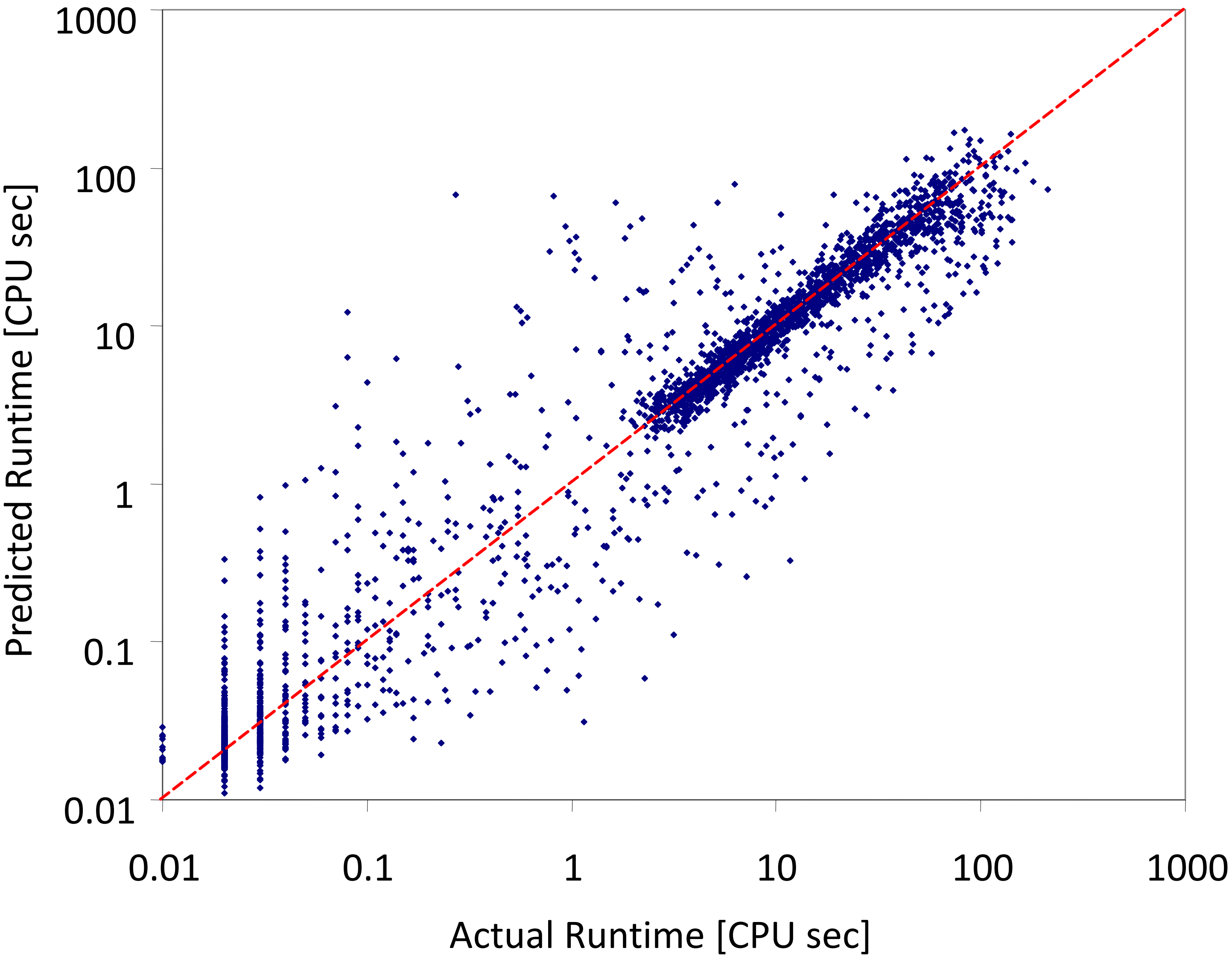


- **Linear programming** relaxation

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

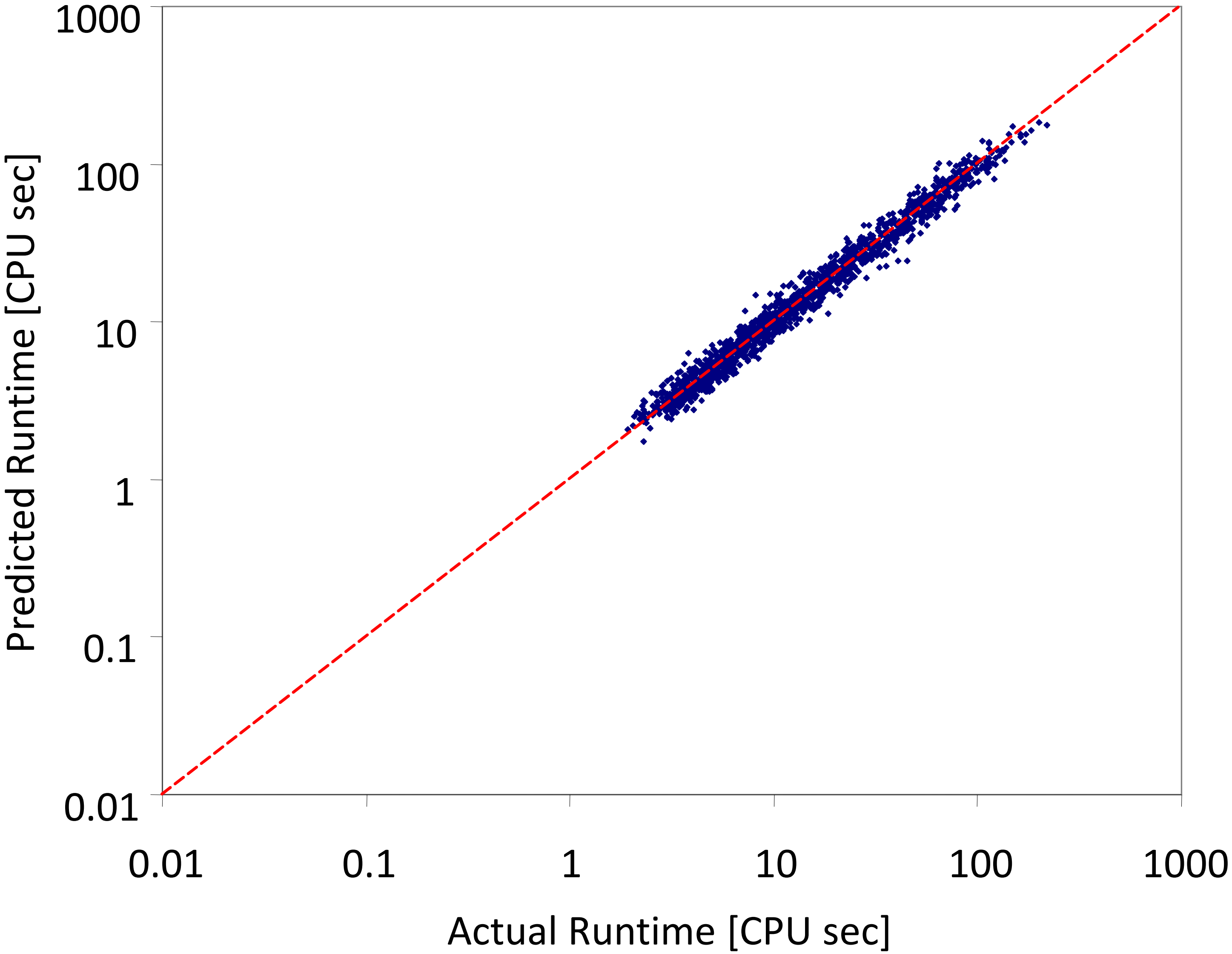
Variable Ratio Prediction (Kcnfs)

Note: each point corresponds to a "test" instance not used to train the model.

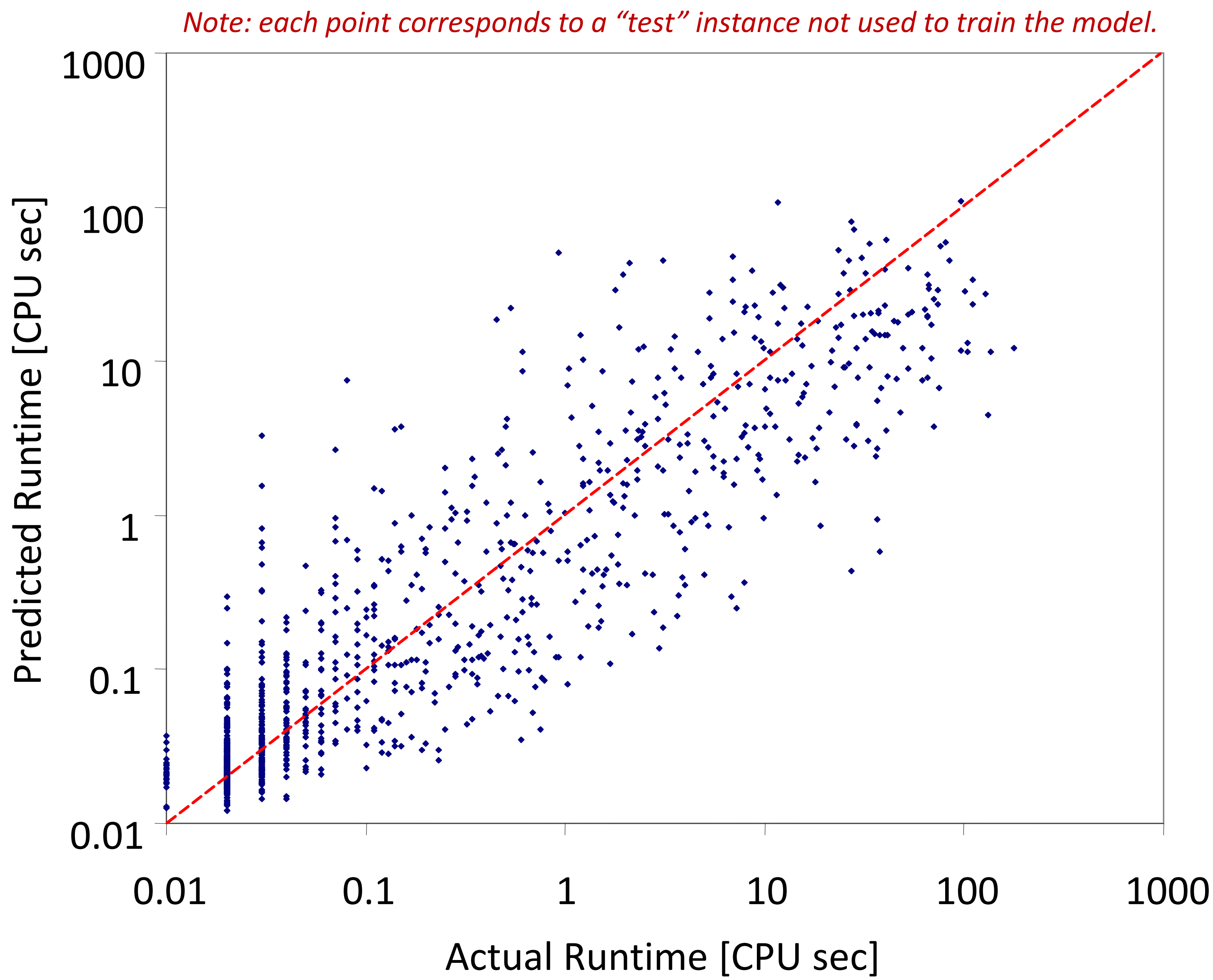


Variable Ratio - UNSAT

Note: each point corresponds to a "test" instance not used to train the model.



Variable Ratio - SAT



Feature Importance – Variable Ratio

- We can **analyze a model's features** to identify problem parameters that most affect empirical hardness
  - problem: very high-dimensional models
  - solution: subset selection
  - caveat: other subsets could potentially achieve similar performance

- Questions:
  - Do our models **discover the importance** of c/v?
  - If so, **in what form** do the models depend on this quantity?
  - What **other features** are important?

Feature Importance – Variable Ratio

- We can **analyze a model's features** to identify problem parameters that most affect empirical hardness
  - problem: very high-dimensional models
  - solution: subset selection
  - caveat: other subsets could potentially achieve similar performance

Variable	Cost of Omission
$\lvert c/v - 4.26\rvert$	100
$\lvert c/v - 4.26\rvert^2$	69
$(v/c)^2 \cdot$ SapsBestCVMean	53
$\lvert c/v - 4.26\rvert \cdot$ SapsBestCVMean	33

Feature Importance – Variable Ratio

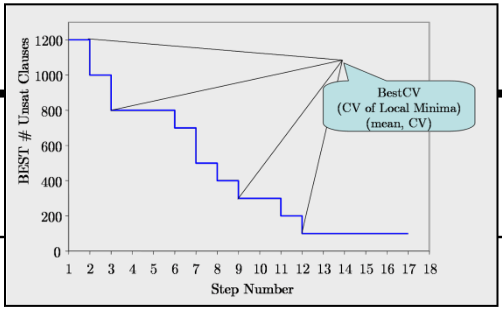
- We can **analyze a model's features** to identify problem parameters that most affect empirical hardness
  - problem: very high-dimensional models
  - solution: subset selection
  - caveat: other subsets could potentially achieve similar performance

Variable	Cost of Omission
$\lvert c/v - 4.26 \rvert$	100
$\lvert c/v - 4.26 \rvert^2$	69
$(v/c)^2$ · SapsBestCVMean	53
$\lvert c/v - 4.26 \rvert$ · SapsBestCVMean	33

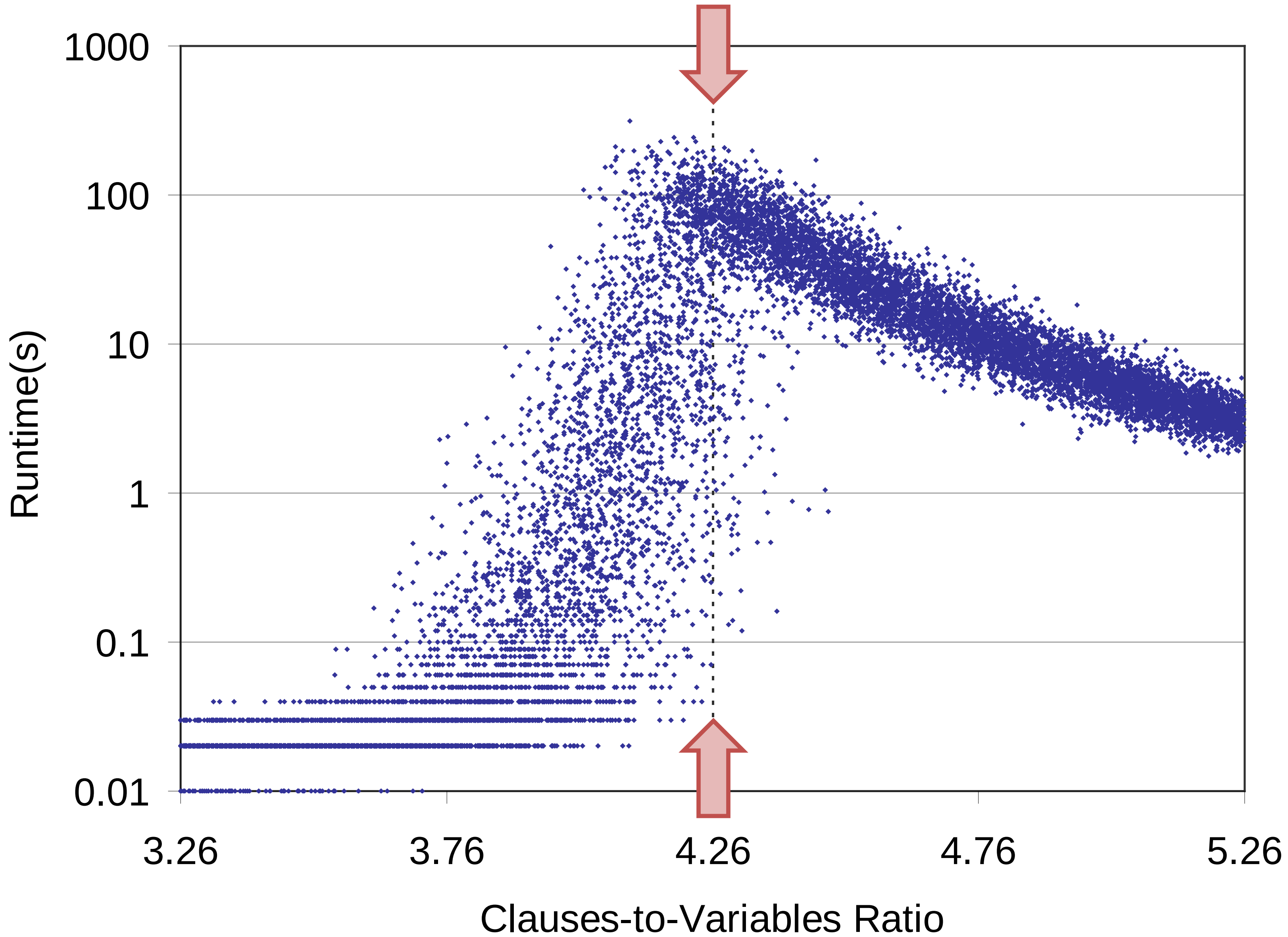
Feature Importance – Variable Ratio

- We can **analyze a model's features** to identify problem parameters that most affect empirical hardness
  - problem: very high-dimensional models
  - solution: subset selection
  - caveat: other subsets could potentially achieve similar performance

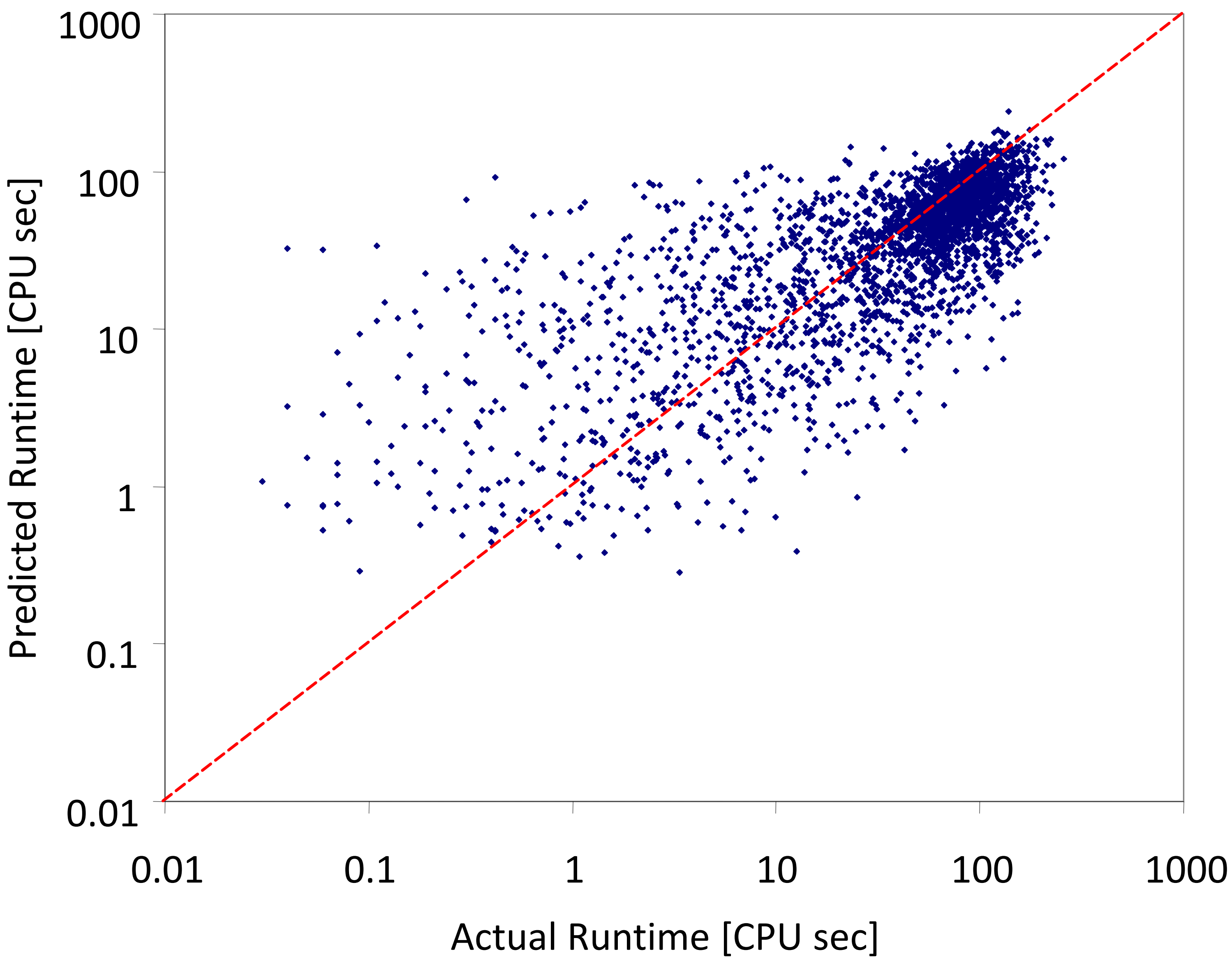
Variable	Cost of Omission
$\lvert c/v - 4.26\rvert$	100
$\lvert c/v - 4.26\rvert^2$	69
$(v/c)^2$ · **SapsBestCVMean**	53
$\lvert c/v - 4.26\rvert$ · **SapsBestCVMean**	33



Fixed Ratio Data



Fixed Ratio Prediction (Kcnfs)

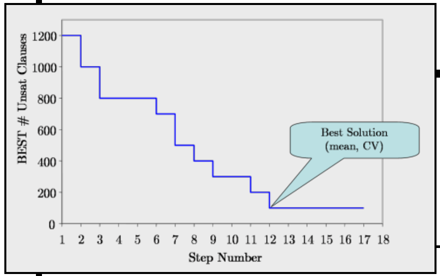


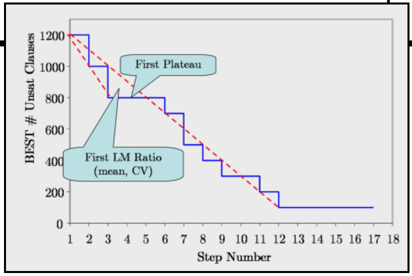
Feature Importance – Fixed Ratio

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

Feature Importance – Fixed Ratio

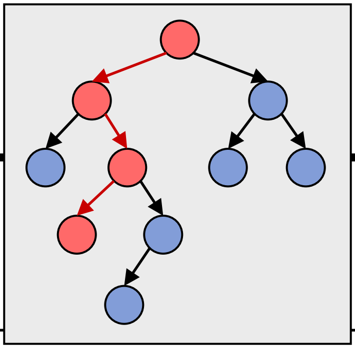
Variable	Cost of Omission
SapsBestSolMean^2	100
SapsBestSolMean · MeanDPLLDepth	74
GsatBestSolCV · MeanDPLLDepth	21
VCGClauseMean · GsatFirstLMRatioMean	9

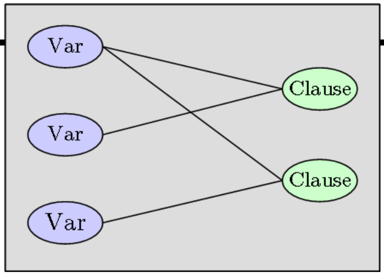




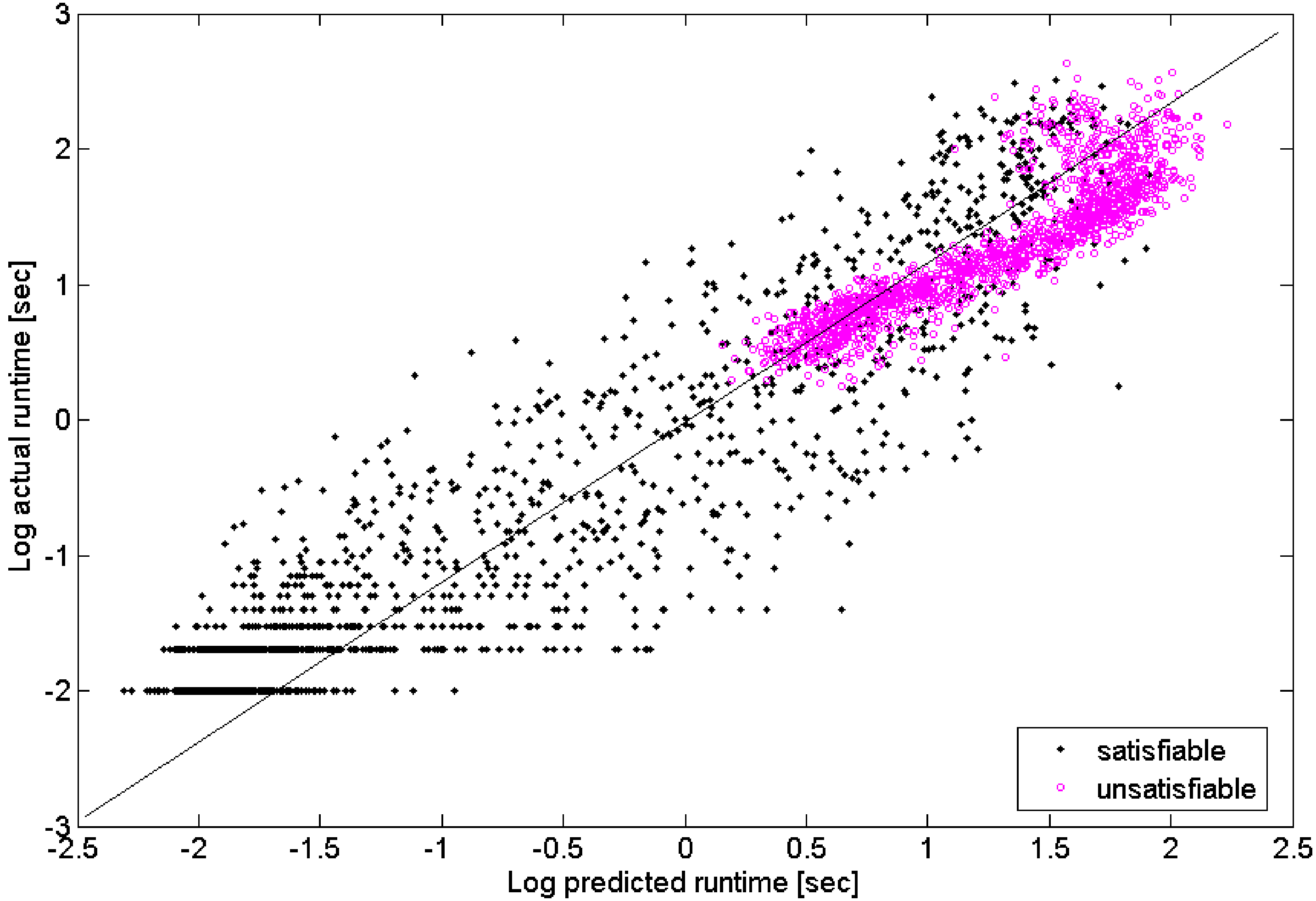
Feature Importance – Fixed Ratio

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





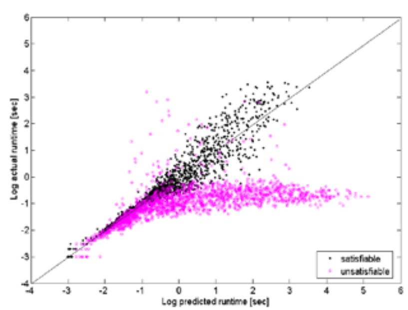
Empirical Performance of EHMs



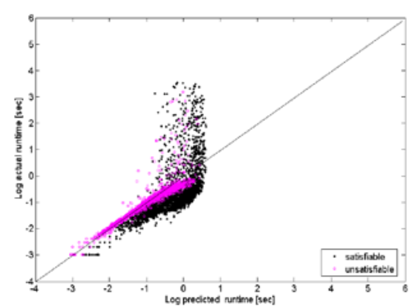
Predicted vs. Actual Log Runtime, SATZ on Uniform Random 3SAT, variable ratio

Hierarchical Hardness Models

- Conditioning on satisfiability of the instance, **single-feature models become sufficient**, clauses/variables unimportant
  - 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**

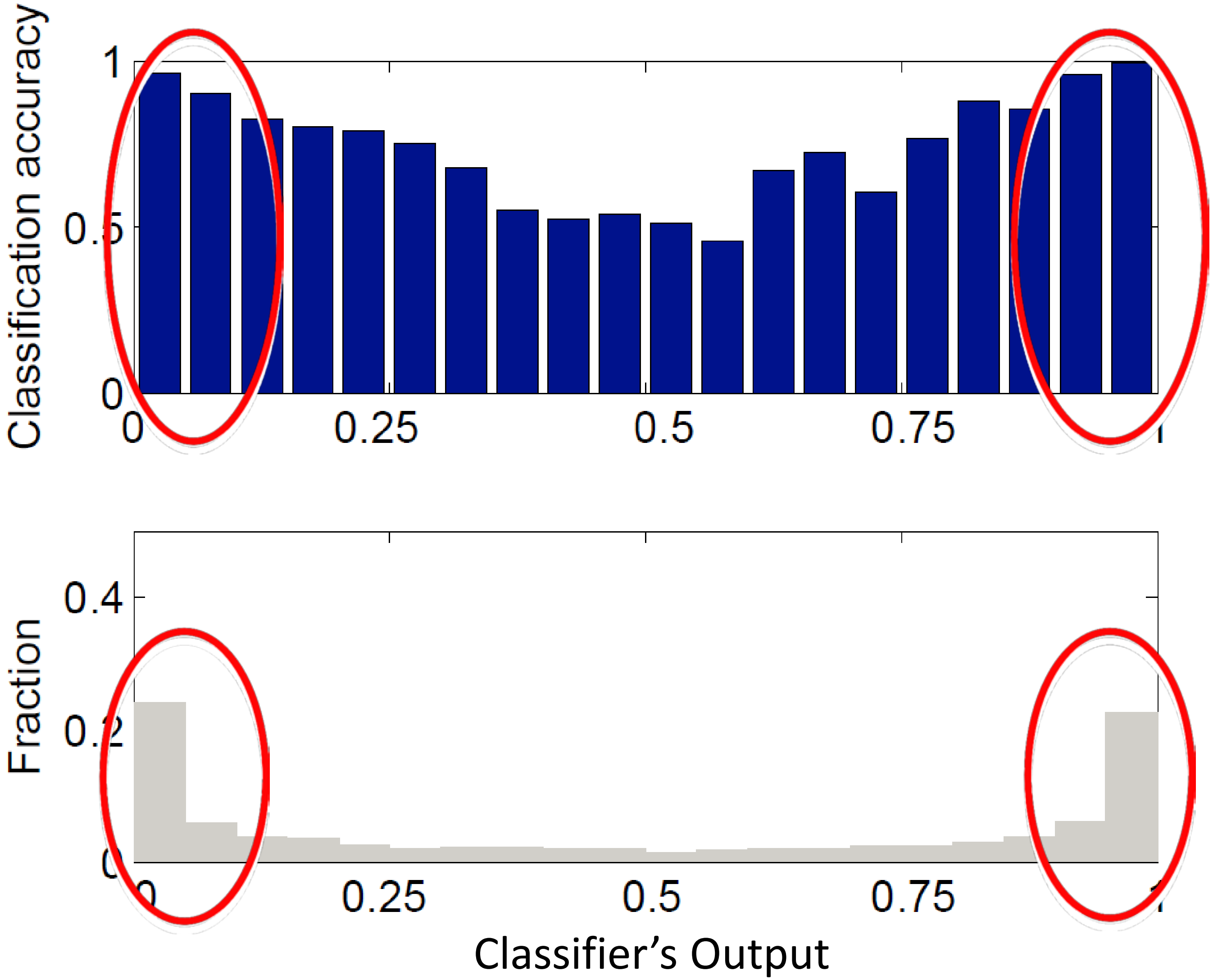


SAT-only

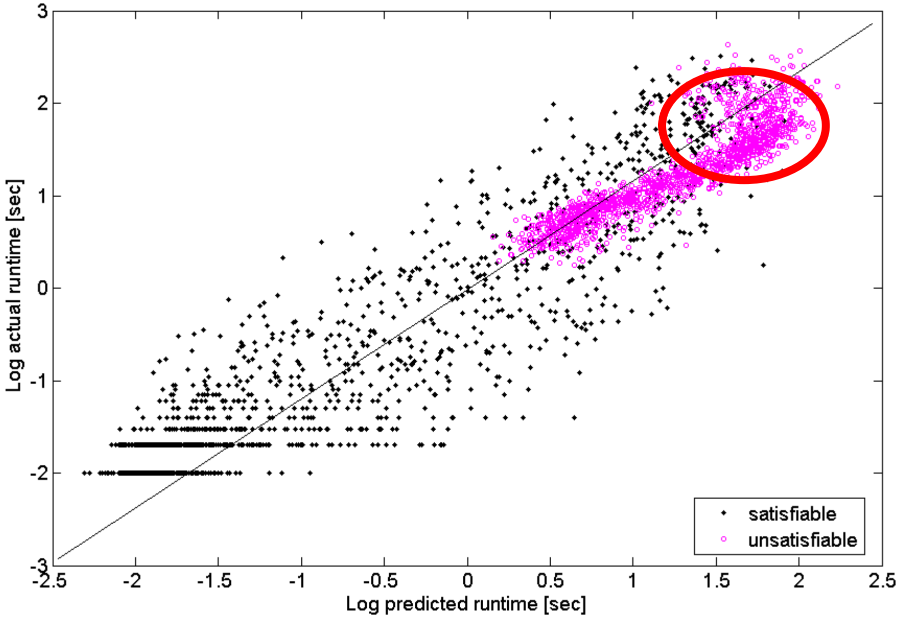


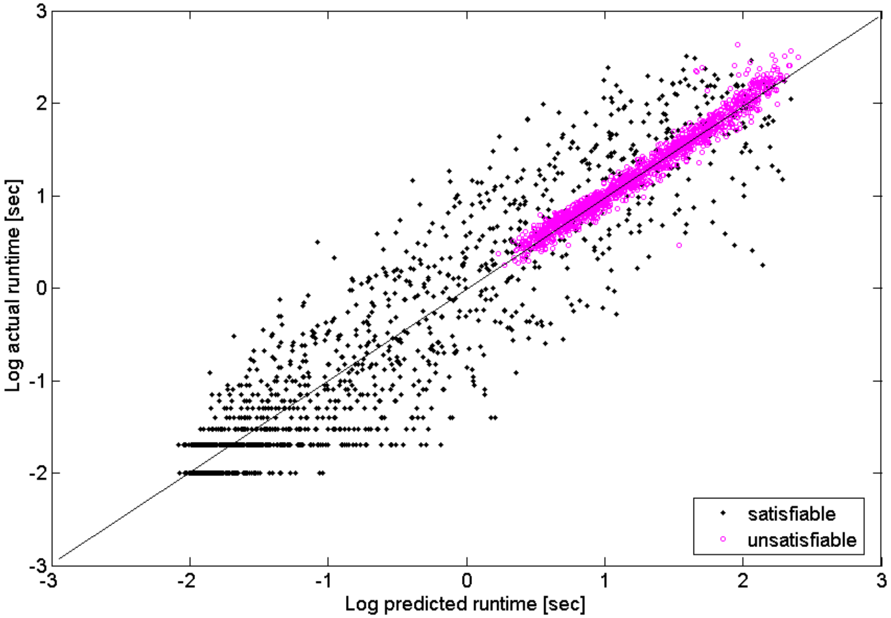
UNSAT-only

Predicting Satisfiability Status (fixed-ratio 3-SAT)



Empirical Performance of HHMs





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

BEYOND UNIFORM-RANDOM 3-SAT

[L-B, Nudelman, Shoham, 2002; 2009]
[Hutter, Xu, Hoos, L-B, 2006–ongoing]

Beyond Uniform-Random 3-SAT

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

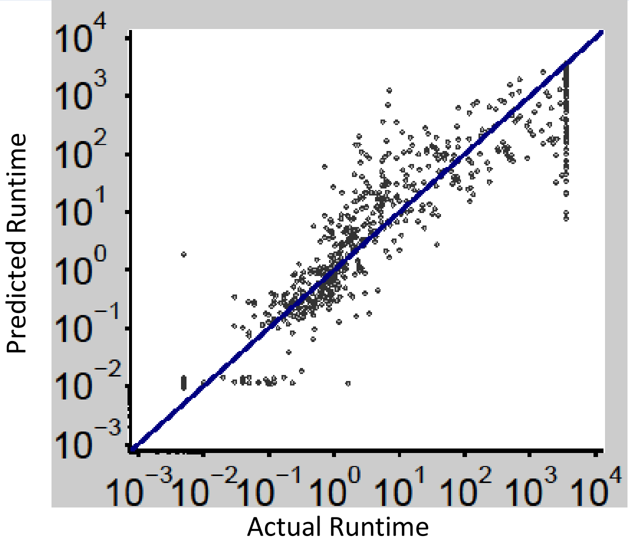
- 4 **problem domains** (with new features in each domain)
  - Combinatorial Auctions
  - Satisfiability (SAT)
  - Mixed Integer Programming (MIP)
  - Travelling Salesman Problem (TSP)
- dozens of **solvers**, including:
  - state of the art solvers in each domain
  - black-box, commercial solvers
-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

We've also investigated **different machine learning techniques**.
Overall, we recommend **random forests of regression trees**.

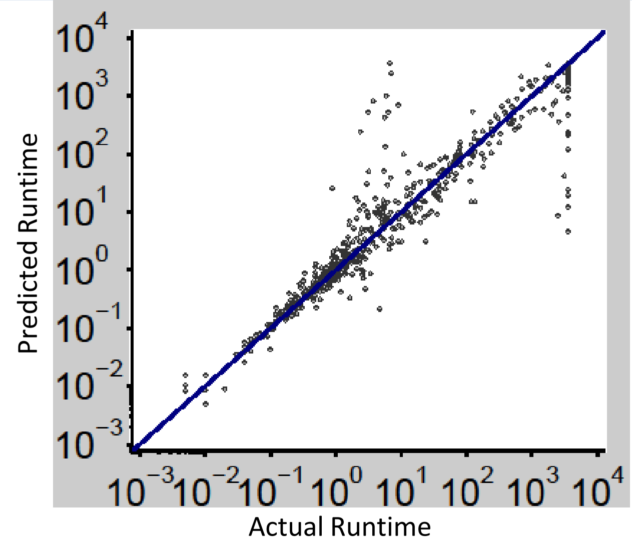
Examples: EHMs for SAT

IBM hardware verification data, SPEAR solver

Linear Regression (RMSE=0.60)

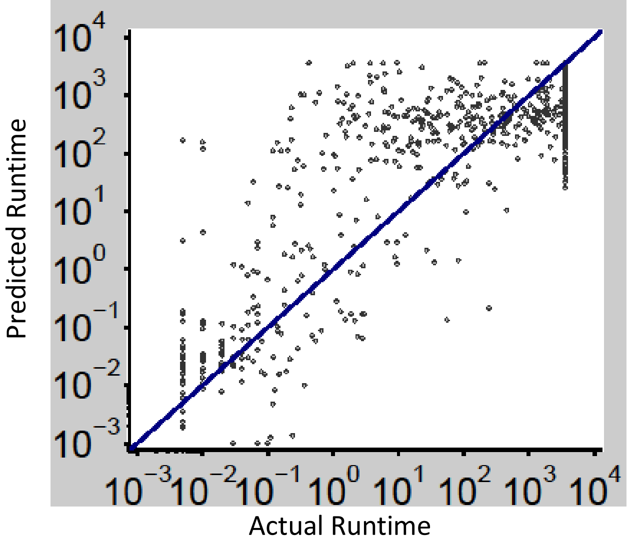


Random Forest (RMSE=0.38)

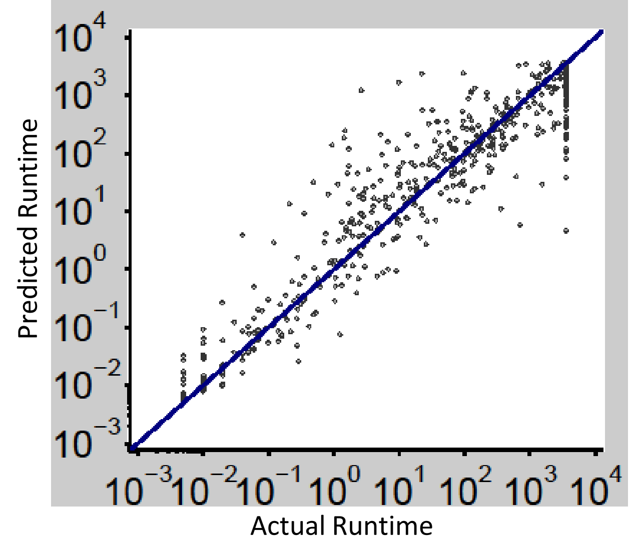


SAT Competition (Random + Handmade + Industrial) data, MINISAT solver

Linear Regression (RMSE=1.01)



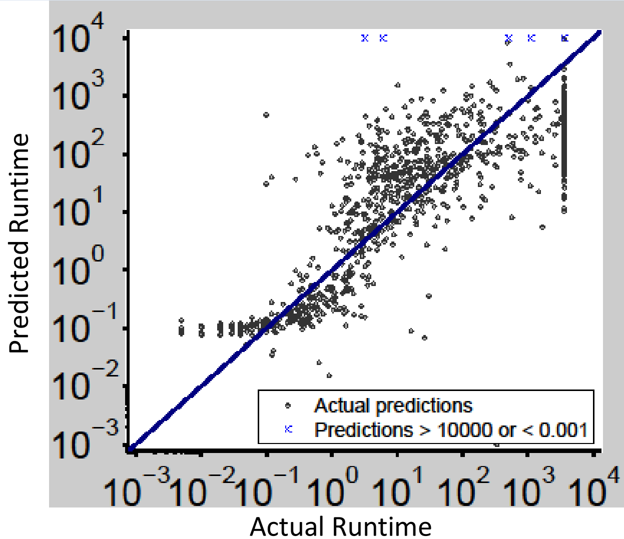
Random Forest (RMSE=0.47)



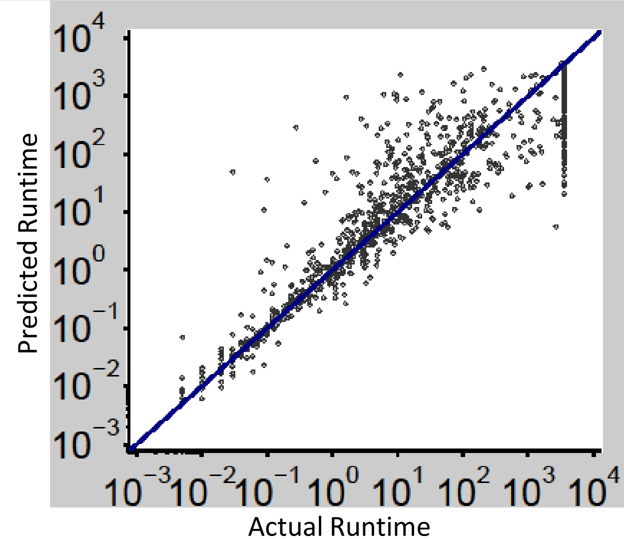
Examples: EHMs for MIP

MIPLIB data, CPLEX 12.1 solver

Linear Regression (RMSE=2.68×10^8)

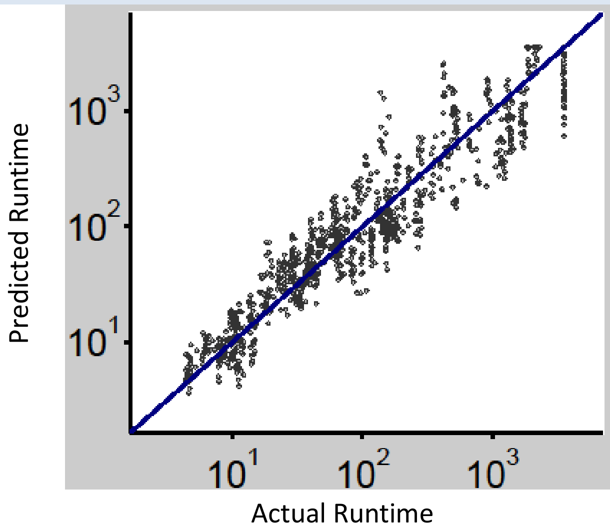


Random Forest (RMSE=0.63)

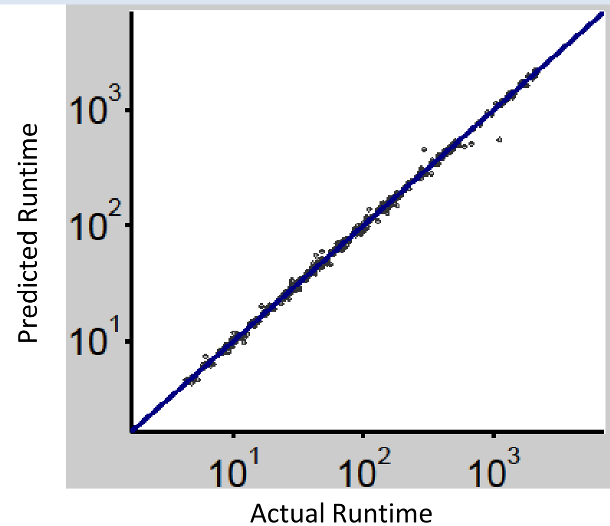


Red Crested Woodpecker habitat data, CPLEX 12.1 solver

Linear Regression (RMSE=0.25)



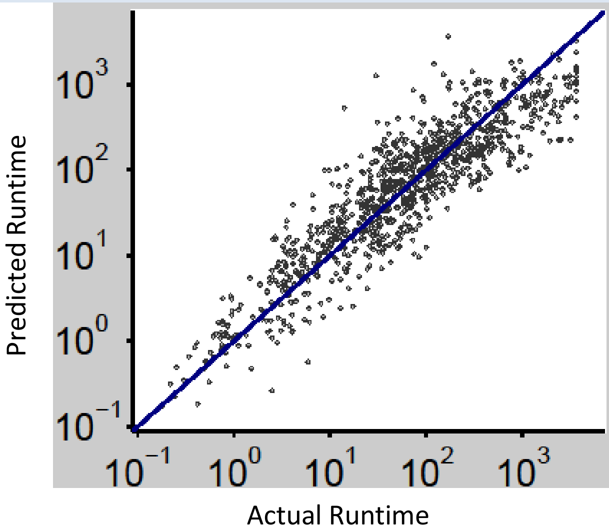
Random Forest (RMSE=0.02)



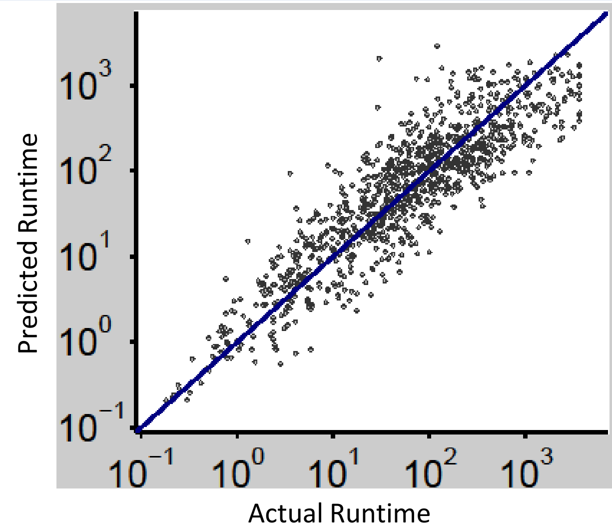
Examples: EHMs for TSP

PORTGEN uniform-random data, Concorde solver

Linear Regression (RMSE=0.41)

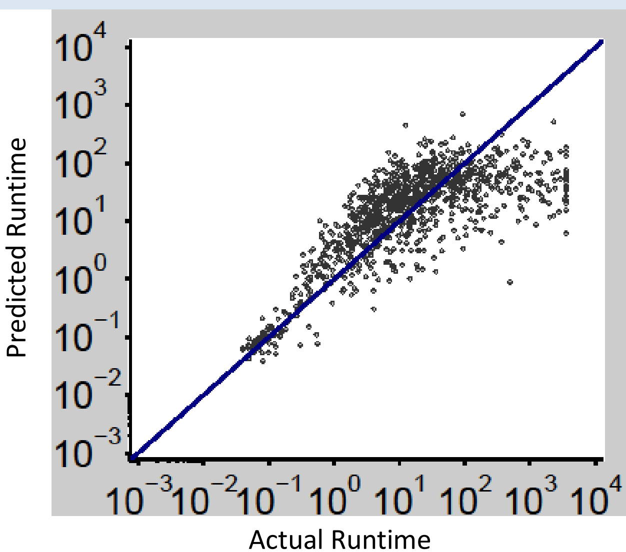


Random Forest (RMSE=0.44)

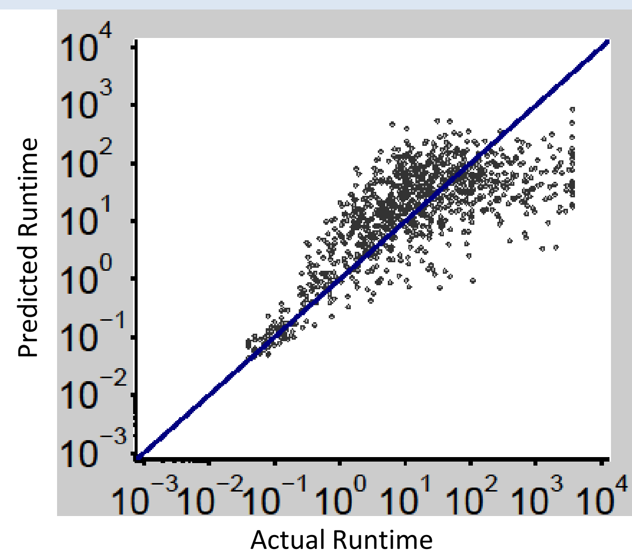


PORTCGEN random clustered data, LK-H solver

Linear Regression (RMSE=0.78)



Random Forest (RMSE=0.74)



Modeling Algorithm Design Spaces

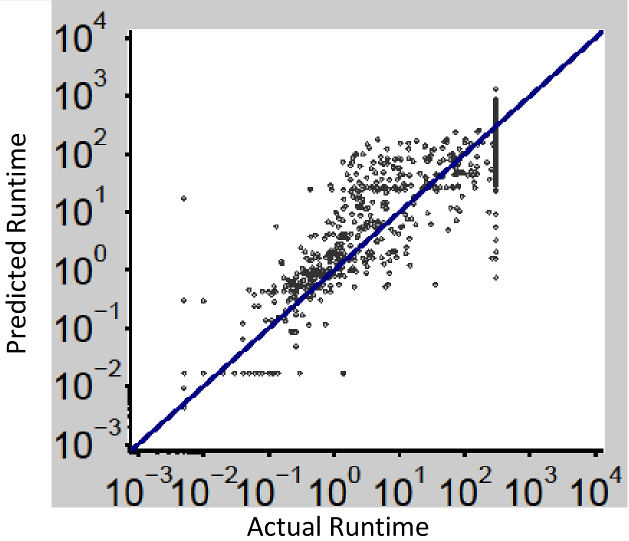
- **Models can be extended to the *sets* of algorithms** described by solvers with parameters that are:
  - continuous or discrete
  - ordinal or categorical
  - potentially conditional on the values of other parameters

- These models are useful for:
  - **understanding hardness** of an instance distribution across a (potentially infinite) family of algorithms
  - **choosing a solver design** to use in practice
    - we can iterate between identifying a design with good predicted performance, and gathering data about this new design
    - "sequential model-based optimization" paradigm in Bayesian statistics

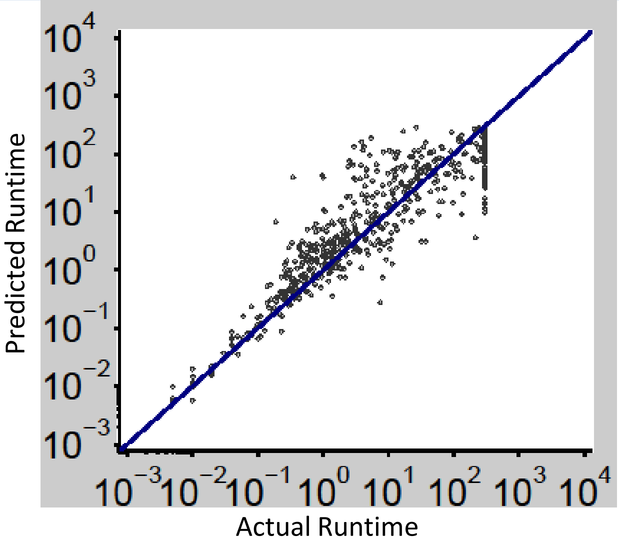
Previously Unseen Instances *and* Configurations

SAT: IBM hardware verification data, SPEAR solver

Linear Regression (RMSE=0.58)

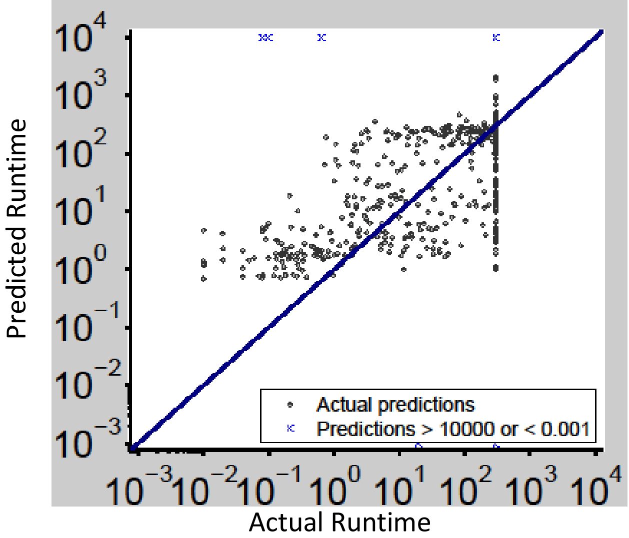


Random Forest (RMSE=0.43)

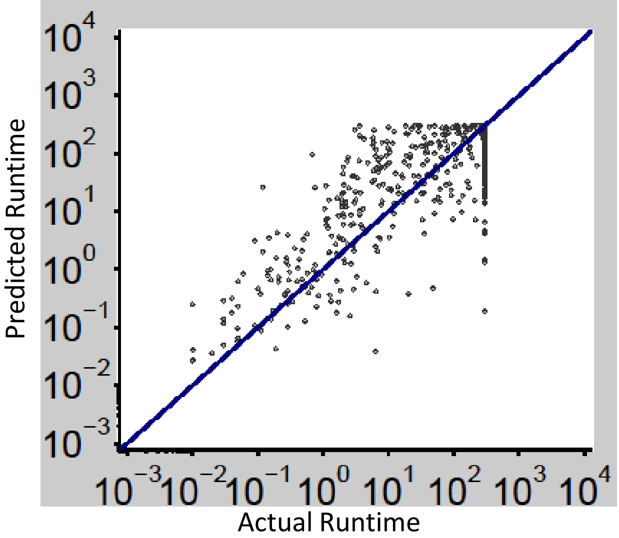


MIP: MIPLIB data, CPLEX 12.1 solver

Linear Regression (RMSE > 1×10^{99})



Random Forest (RMSE=0.55)



Summary and Applications of EHMs

- **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
  - **analysis of these models** can open avenues for theoretical investigations beyond the worst case

- EHMs are also useful for practical applications:
  - job **scheduling** (e.g., to minimize makespan)
  - automatic **design of algorithm portfolios**
  - automatic synthesis of **hard benchmark distributions**
  - model-based **solver tuning/design**