SATzilla: Portfolio-based Algorithm Selection for SAT

Xu, Hutter, Hoos, Leyton-Brown

JAIR’08

Reading

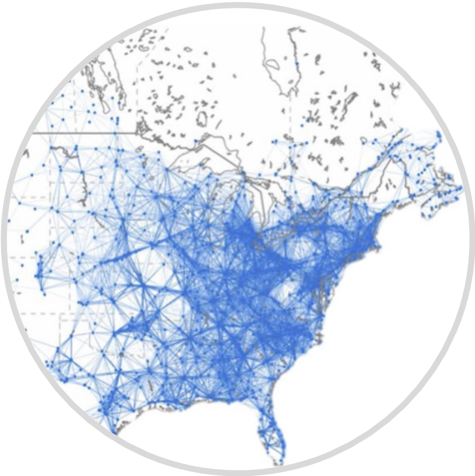
Another comprehensive journal paper on a seminal work

Check website for specific sections to read 👀

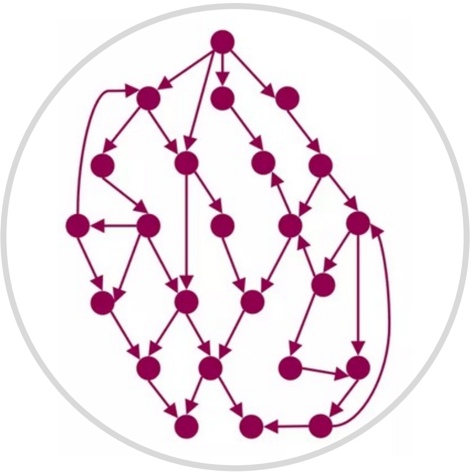
SAT

One of the most fundamental problems in computer science

Planning



Graph coloring



Formal verification

And many more applications!

SATzilla

Summary of approach:

1. Took **7 solvers** from the 2006 SAT competition

   AKA, a **portfolio** of solvers

2. **ML model** predicts which solver to use on any input

- Won several **gold medals** at the 2007 SAT competition
- Kept winning **gold medals** at future SAT competitions…
- In 2013, portfolio-based approaches were **disallowed**

Outline

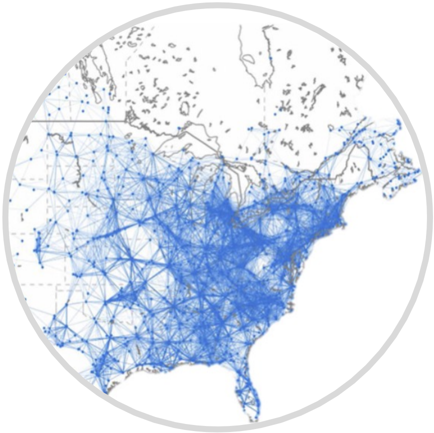
1. Introduction
2. **Overview of approach**
3. Censored data
4. Incorporating hardness predictions
5. Experiments
6. Follow-up research

1: Model the application domain

Recall:

Application-specific distribution $\mathcal{D}$ over SAT problems, e.g.:

- Distribution over radio spectrum **graph coloring** problems
- Uniform distribution over **benchmark** dataset



This paper:

Set of all instances from previous SAT competitions

2: Select candidate solvers

- Choose a set of candidate solvers
- Should have relatively **uncorrelated** runtimes on $\mathcal{D}$
- Each solver should perform well on **some** instances

This paper: 7 solvers entered in the 2006 SAT competition

3: Identify features

Identify features that characterize problem instances

This paper: 48 hand-designed features

Example: $(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_1 \vee x_4 \vee x_5) \wedge (x_2 \vee x_6)$

- Number of clauses: 4
- Number of variables: 6

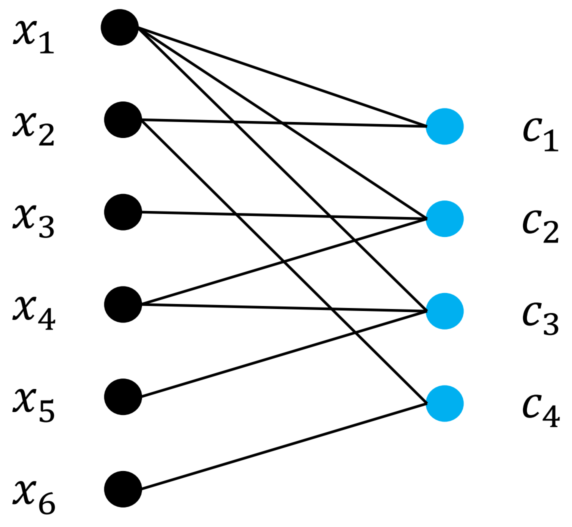
3: Identify features

Identify features that characterize problem instances

This paper: 48 hand-designed features

Example: $\overbrace{(x_1 \vee x_2)}^{c_1} \wedge \overbrace{(x_1 \vee \bar{x}_3 \vee \bar{x}_4)}^{c_2} \wedge \overbrace{(x_1 \vee x_4 \vee x_5)}^{c_3} \wedge \overbrace{(x_2 \vee x_6)}^{c_4}$

Variable -clause graph



Variable node degree statistics:

- Mean: $\frac{1}{6}(3 + 2 + 1 + 2 + 1 + 1) = \frac{5}{3}$
- Min: 1
- Max: 3
- …

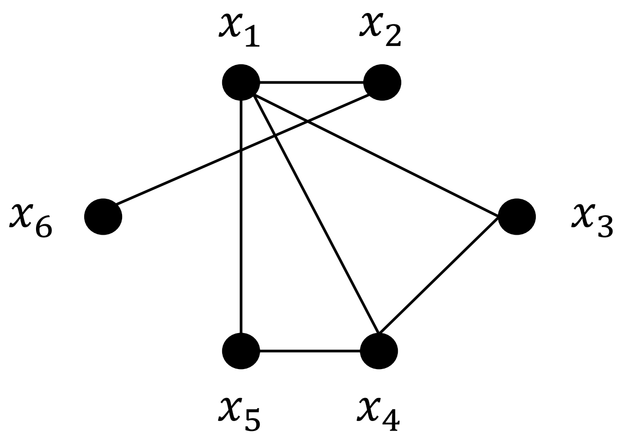
3: Identify features

Identify features that characterize problem instances

This paper: 48 hand-designed features

Example: $(x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_1 \vee x_4 \vee x_5) \wedge (x_2 \vee x_6)$

Variable graph



Node degree statistics:

- Mean: $\frac{1}{6}(5+2+3+3+2+1) = \frac{8}{3}$
- Min: 1
- Max: 5
- $\dots$

4: Obtain training data

- Sample SAT instances $\pi_1, \ldots, \pi_N \sim \mathcal{D}$
- For each π_i, compute features $\boldsymbol{x}(\pi_i) \in \mathbb{R}^d$

  In the paper, $d = 48$
- Run all solvers on all instances to determine runtimes

Training data:

	Instance π_1	**Instance** π_2		**Instance** π_N
Solver s_1	$(\boldsymbol{x}(\pi_1), \mathrm{runtime}(s_1, \pi_1))$	$(\boldsymbol{x}(\pi_2), \mathrm{runtime}(s_1, \pi_2))$		$(\boldsymbol{x}(\pi_N), \mathrm{runtime}(s_1, \pi_N))$
Solver s_2	$(\boldsymbol{x}(\pi_1), \mathrm{runtime}(s_2, \pi_1))$	$(\boldsymbol{x}(\pi_2), \mathrm{runtime}(s_2, \pi_2))$	$\ddots$	$(\boldsymbol{x}(\pi_N), \mathrm{runtime}(s_2, \pi_N))$
Solver s_3	$(\boldsymbol{x}(\pi_1), \mathrm{runtime}(s_3, \pi_1))$	$(\boldsymbol{x}(\pi_2), \mathrm{runtime}(s_3, \pi_3))$		$(\boldsymbol{x}(\pi_N), \mathrm{runtime}(s_3, \pi_N))$

5: Identify a presolver

Remove instances that solve quickly with presolver

Training effort will be focused on hard instances

Training data:

	Instance π_1	**Instance π_2**		**Instance π_N**
Solver s_1	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_1, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_1, \pi_2))$		$(\boldsymbol{x}(\pi_N), \text{runtime}(s_1, \pi_N))$
Solver s_2	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_2, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_2, \pi_2))$	$\ddots$	$(\boldsymbol{x}(\pi_N), \text{runtime}(s_2, \pi_N))$
Solver s_3	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_3, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_3, \pi_3))$		$(\boldsymbol{x}(\pi_N), \text{runtime}(s_3, \pi_N))$

6: Train ML model for each solver

- Train ML model $f_i : \mathbb{R}^d \to \mathbb{R}$ for each solver s_i
- Ideally, for $\pi \sim \mathcal{D}$, $\underbrace{f_i(\boldsymbol{x}(\pi))}_{\text{Predicted runtime}} \approx \underbrace{\text{runtime}(s_i, \pi)}_{\text{Actual runtime}}$

Training data:

	Instance π_1	Instance π_2		Instance π_N
Solver s_1	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_1, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_1, \pi_2))$		$(\boldsymbol{x}(\pi_N), \text{runtime}(s_1, \pi_N))$
Solver s_2	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_2, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_2, \pi_2))$	$\ddots$	$(\boldsymbol{x}(\pi_N), \text{runtime}(s_2, \pi_N))$
Solver s_3	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_3, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_3, \pi_3))$		$(\boldsymbol{x}(\pi_N), \text{runtime}(s_3, \pi_N))$

Runtime protocol

On a **new instance** π:

1. Run presolver until predetermined cutoff time reached
2. Compute $f_i(\boldsymbol{x}(\pi))$ for every solver s_i
3. Run solver with **smallest predicted runtime**

7: Optimizing the final portfolio

Since predictions $f_i(\boldsymbol{x}(\pi))$ aren't perfect,
It's possible to improve performance by **removing** a solver

Using validation set:
Find **subset** of solvers leading to best end-to-end runtime

Outline

1. Introduction
2. Overview of approach
3. **Censored data**
4. Incorporating hardness predictions
5. Experiments
6. Follow-up research

Accounting for censored data

Computing the runtime of any solver can take **weeks**!

Sometimes need to **terminate** before completing

Still want to **accurately predict** $\mathrm{runtime}(s_i, \pi)$

Accounting for censored data

1. Train each f_i treating **capped runtime** as true runtime
2. Repeat until convergence:
   i. Estimate runtime of capped runs using f_i

Training data:

	Instance π_1	**Instance π_2**		**Instance π_N**
Solver s_1	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_1, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_1, \pi_2))$		$(\boldsymbol{x}(\pi_N), \text{cap } \kappa)$
Solver s_2	$(\boldsymbol{x}(\pi_1), \text{cap } \kappa)$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_2, \pi_2))$	$\ddots$	$(\boldsymbol{x}(\pi_N), \text{runtime}(s_2, \pi_N))$
Solver s_3	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_3, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{cap } \kappa)$		$(\boldsymbol{x}(\pi_N), \text{runtime}(s_3, \pi_N))$

Accounting for censored data

1. Train each f_i treating **capped runtime** as true runtime
2. Repeat until convergence:
   i. Estimate runtime of capped runs using f_i
   ii. Retrain f_i treating **estimated runtimes** as true runtimes

Training data:

	Instance π_1	**Instance** π_2		**Instance** π_N
Solver s_1	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_1, \pi_1))$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_1, \pi_2))$	$\ddots$	$\left(\boldsymbol{x}(\pi_N), f_1\left(\boldsymbol{x}(\pi_N)\right)\right)$
Solver s_2	$\left(\boldsymbol{x}(\pi_1), f_2\left(\boldsymbol{x}(\pi_1)\right)\right)$	$(\boldsymbol{x}(\pi_2), \text{runtime}(s_2, \pi_2))$		$(\boldsymbol{x}(\pi_N), \text{runtime}(s_2, \pi_N))$
Solver s_3	$(\boldsymbol{x}(\pi_1), \text{runtime}(s_3, \pi_1))$	$\left(\boldsymbol{x}(\pi_2), f_3\left(\boldsymbol{x}(\pi_2)\right)\right)$		$(\boldsymbol{x}(\pi_N), \text{runtime}(s_3, \pi_N))$

Outline

1. Introduction
2. Overview of approach
3. Censored data
4. **Incorporating hardness predictions**
5. Experiments
6. Follow-up research

Hardness + runtime predictions

Can get better predictions if **only** train on **satisfiable** problems
…or **only** on **unsatisfiable** problems

Approach:

1. Train classifier $s(\boldsymbol{x}(\pi)) \in \{\mathrm{sat}, \mathrm{unsat}\}$ to predict if satisfiable
2. For each solver s_i, train model $f_{i,\mathrm{sat}}$ to predict runtime
   Train only on satisfiable instances
3. Also train $f_{i,\mathrm{unsat}}$ to predict runtime
   Train only on unsatisfiable instances

Hardness + runtime predictions

RV $z \in \{\mathrm{sat}, \mathrm{unsat}\}$ represents **belief** of whether $f_{i,\mathrm{sat}}$ or $f_{i,\mathrm{unsat}}$ is a **better prediction** of runtime

Predict runtime as

$$f_i(\boldsymbol{x}(\pi)) = \sum_{k \in \{\mathrm{sat},\mathrm{unsat}\}} f_{i,k}(\boldsymbol{x}(\pi)) \cdot \underbrace{\mathbb{P}[z = k \mid \boldsymbol{x}(\pi), s(\boldsymbol{x}(\pi))]}_{\textbf{Need to learn}}$$

Hardness + runtime predictions

Predict runtime as

$$f_i\big(\boldsymbol{x}(\pi)\big) = \sum_{k\in\{\text{sat,unsat}\}} f_{i,k}\big(\boldsymbol{x}(\pi)\big) \cdot \underbrace{\mathbb{P}\big[\, z = k \mid \boldsymbol{x}(\pi), s\big(\boldsymbol{x}(\pi)\big) \,\big]}_{\textbf{Need to learn}}$$

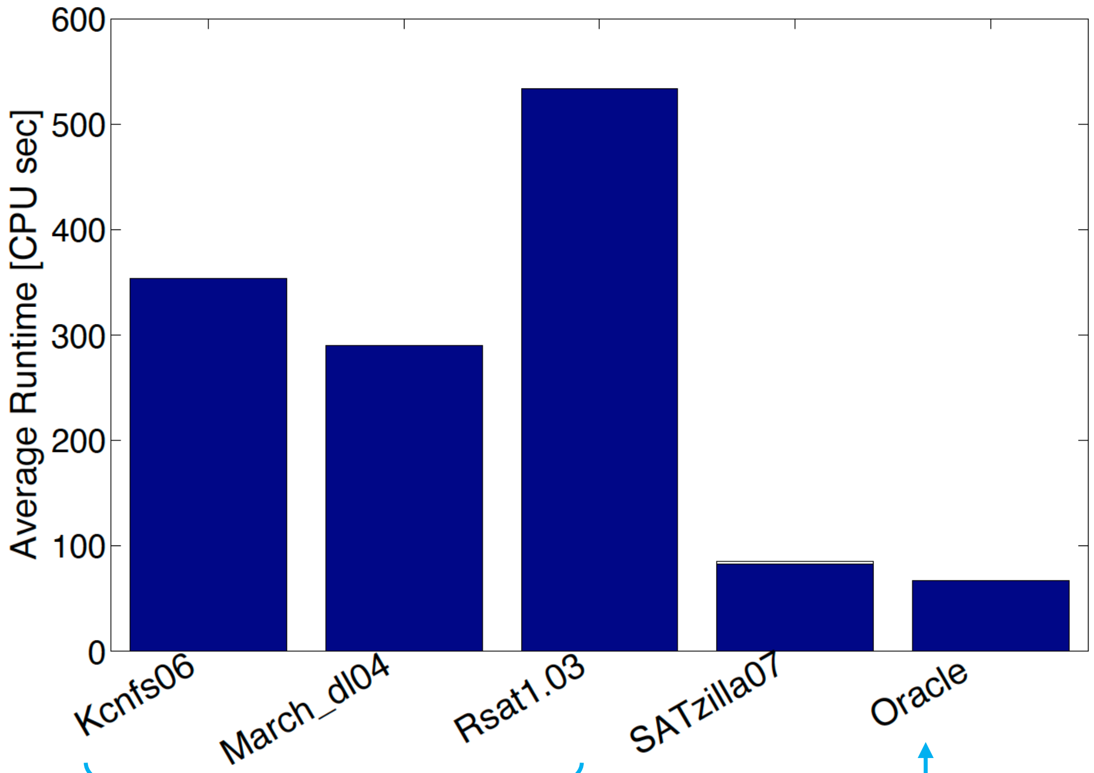
Choose $\mathbb{P}\big[\, z = k \mid \boldsymbol{x}(\pi), s\big(\boldsymbol{x}(\pi)\big) \,\big]$ to minimize

$$\underbrace{\sum_{j=1}^{N}}_{\text{Sum over the } N \text{ training instances}} \Big(\text{runtime}(\pi_j, s_i) - f_i\big(\boldsymbol{x}(\pi_j)\big)\Big)^2$$

Outline

1. Introduction
2. Overview of approach
3. Censored data
4. Incorporating hardness predictions
5. **Experiments**
6. Follow-up research

Experiments: example



Component solvers

Runs component solver with best runtime

Outline

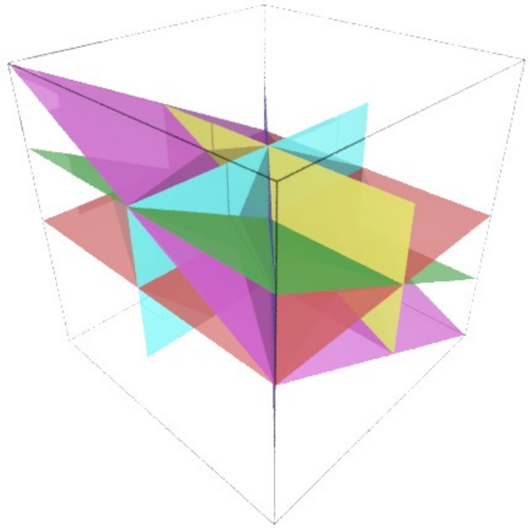
1. Introduction
2. Overview of approach
3. Censored data
4. Incorporating hardness predictions
5. Experiments
6. **Follow-up research**

Empirical performance models

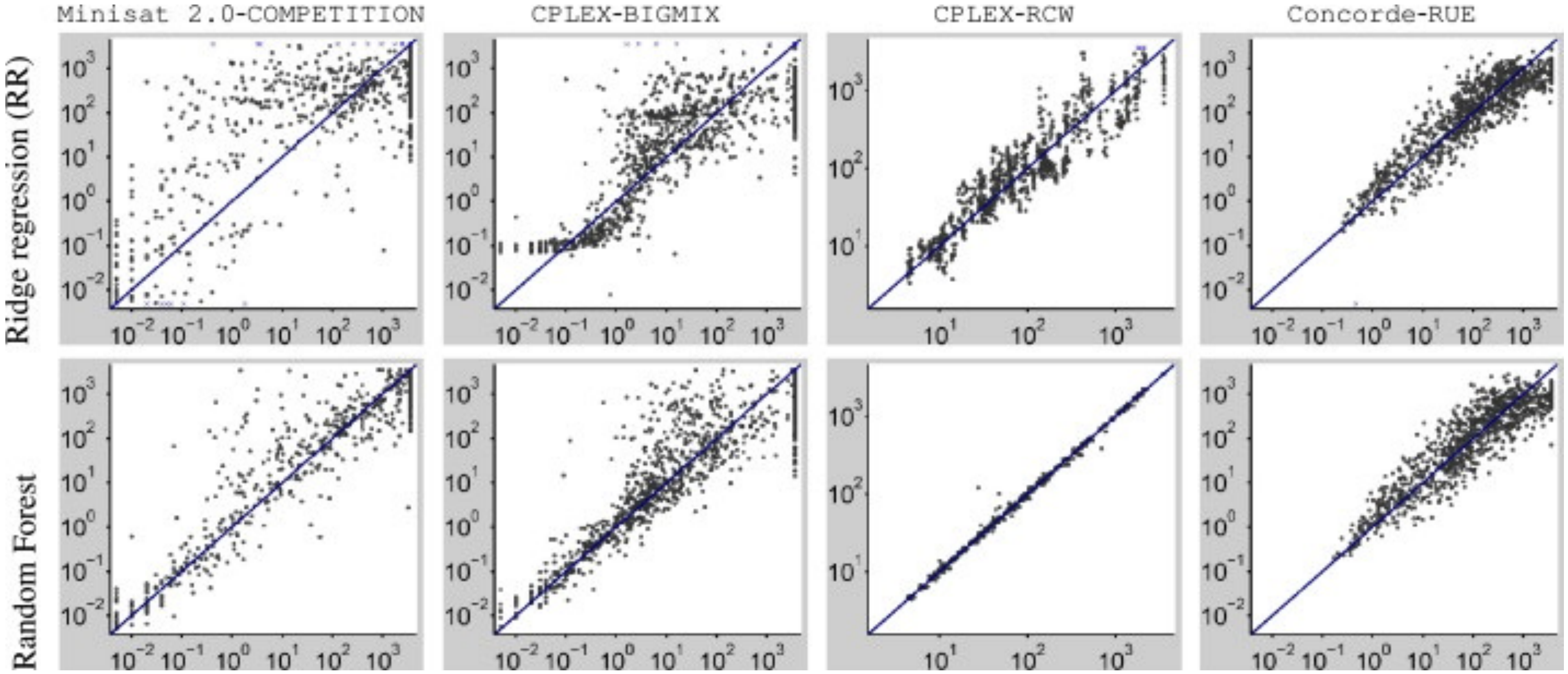
Follow-up work by the same group [Hutter et al. AIJ'14]:

- Expanded set of hand-designed features for SAT to 138
- **Random forests** are great for runtime prediction

  Suggests $\mathbb{R}^{138}$ *can be split into regions where runtime is ~constant*



Empirical performance models



x-axis: true runtime y-axis: predicted runtime

Overview

SATzilla: seminal paper on portfolio-based algorithm selection

Won several **gold medals** at the 2007 SAT competition

Uses **ML** to decide which solver from the '06 competition to use