SATzilla-07: The Design and Analysis of an Algorithm Portfolio for SAT

Lin Xu, Frank Hutter,
Holger H. Hoos and Kevin Leyton-Brown

University of British Columbia

{xulin730, hutter, hoos, kevinlb}@cs.ubc.ca

Outline

- Motivation
- History of SATzilla and related work
- SATzilla methodology
- Example Problem
- SATzilla for the SAT Competition
- Conclusions and ongoing research

Motivation

- Lots of high performance solvers, but ….
  - No single SAT solver dominates all others on all types of instances

- **Question**: How to select the best solver for a given SAT instance?

Algorithm Selection Problem [Rice, 1976]

- **Reference**:
  Select solvers based on previous experience or research papers
- **"Winner-Take-All":**
  Test solvers on some samples from target distribution; select the solver with best performance.
- **SATzilla:**
  Automatically based on instance characteristics'

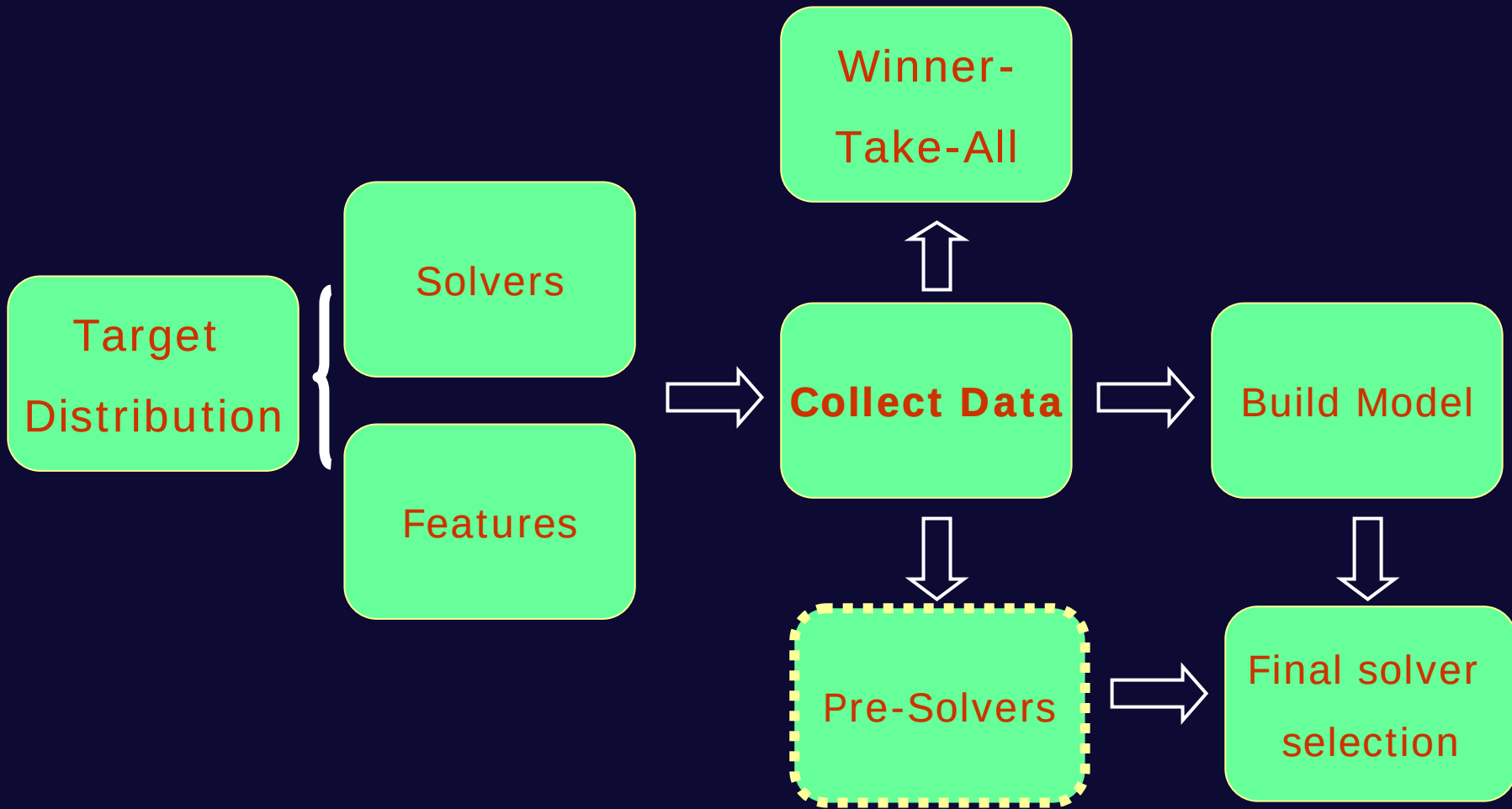
Related work:

- Portfolio of stochastic algorithm [Gomes & Selman,1997]
  - Running multiple algorithms at the same time
- Reinforcement learning [Lagoudakis & Littman, 2001]
  - Select branching rule at each decision point
- Branch & bound algorithm selection [Lobjois & LemaÎter, 1998]
  - Based on an estimation of search tree size

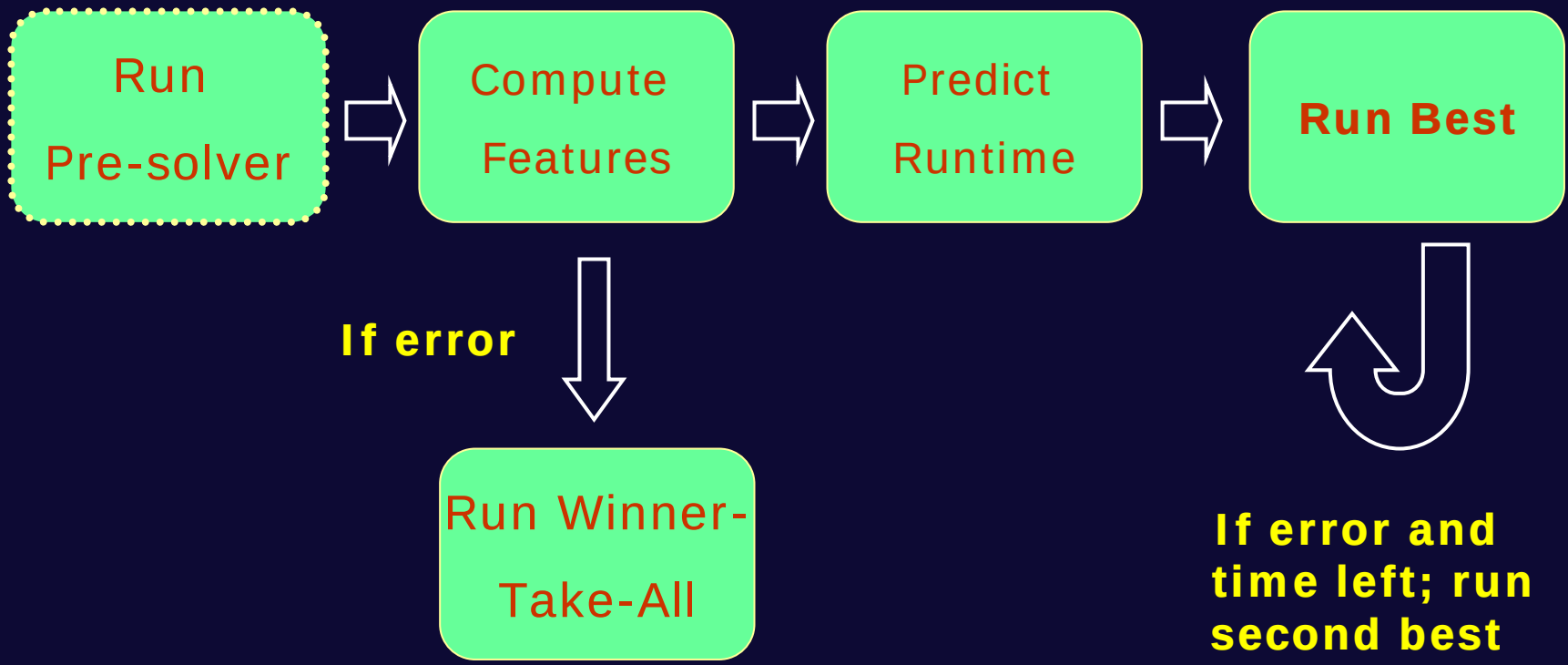
History of SATzilla

- Old SATzilla [Nudelman, Devkar, et. al, 2003]
  - **2nd Random**
  - **2nd Handmade (SAT)**
  - **3rd Handmade**
- SATzilla-07
  - **1st Handmade**
  - **1st Handmade (UNSAT)**
  - **1st Random**
  - **2nd Handmade (SAT)**
  - **3rd Random (UNSAT)**

SATzilla-07 Methodology (offline)



SATzilla-07 Methodology (online)



Solvers Used

- Eureka [Nadel, Gordon, Palti & Hanna, 2006]
- Kcnfs2006 [Dubois & Dequen, 2006]
- March_dl2004 [Heule & Maaren, 2006]
- Minisat2.0 [Eén & Sörensson, 2006]
- OKsolver [Kullmann, 2002]
- Rsat1.04 [Pipatsrisawat & Darwiche, 2006]
- Vallst [Vallstrom, 2005]
- Zchaff_Rand [Mahajan, Fu & Malik, 2005]

SATzilla-07 Example

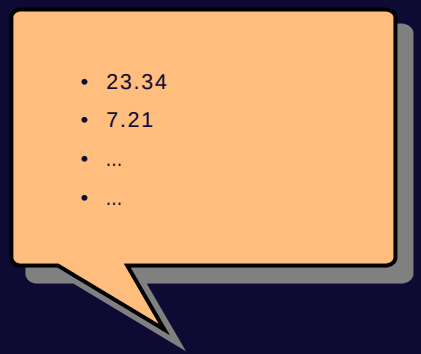
Using quasi-group completion Problems (QCP) to validate our general approach

SATzilla-07 Example Problem

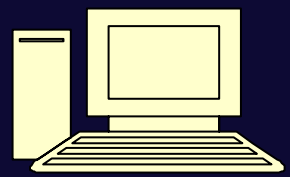
- Problem distribution
  - QCP problems generated near phase transition [Gomes & Selman, 1997]
- Solvers
  - Eureka, OKsolver, Zchaff_Rand
- Features
  - Same as in previous work [Nudelman, et al. 2004]
- Collect Data
  - Compute instances' features and determine solvers' runtime
- Pre-Solver & "Winner take all"
- **Build Models**
- Final solver selection

Empirical Hardness Model (EHM)

- The Core of SATzilla ---→ EHM
  - Accurately predict algorithm's runtime based on cheaply computable features
  - Linear basis function regression



$f_w(\Phi) = w^T \Phi$



Features (Φ)

Runtime (y)

Improve EHM (deal with censoring)

- Heavy-tailed behavior and censoring
- Three ways for censored data
  - Drop them
  - Keep them as if finished at cutoff
  - Censored sampling
- Schmee & Hahn ‘s approach [1979]

REPEAT

1. Estimate runtime conditional on EHM and real runtime bigger than cutoff time
2. Build new EHM with estimated runtime

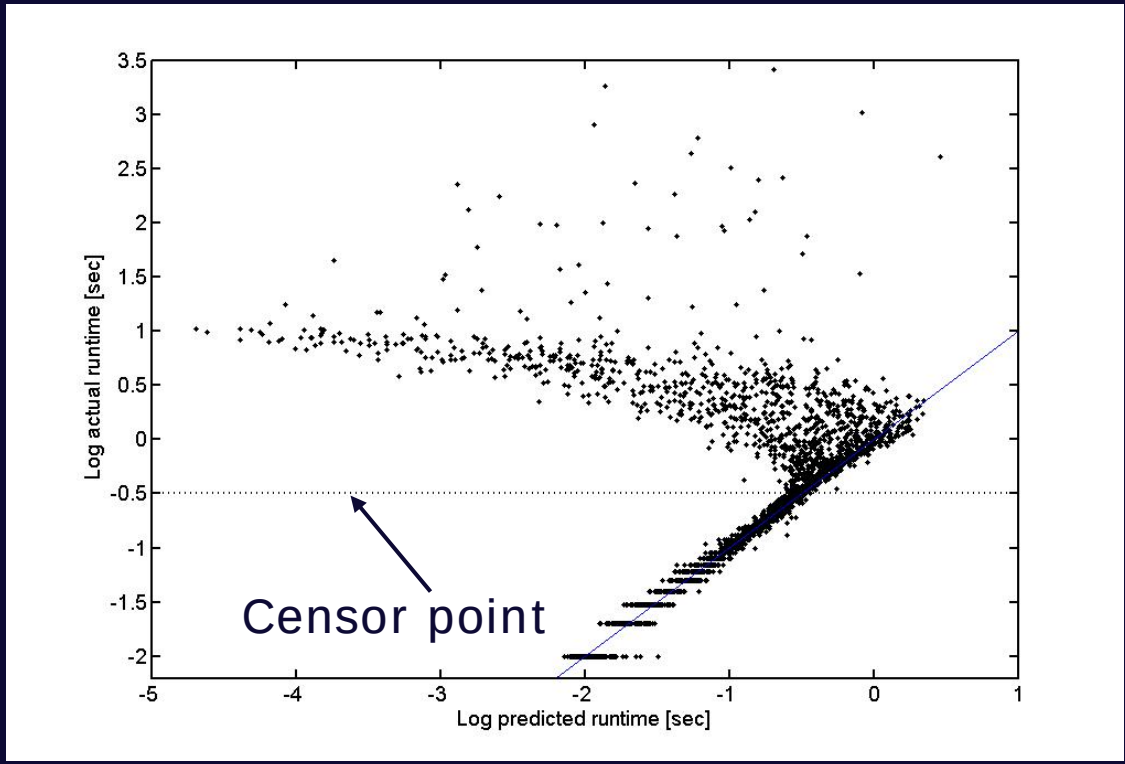
UNTIL no more changes in EHM

How to deal with censored data

A: Drop them

B: Finished at cutoff

C: Censored sampling

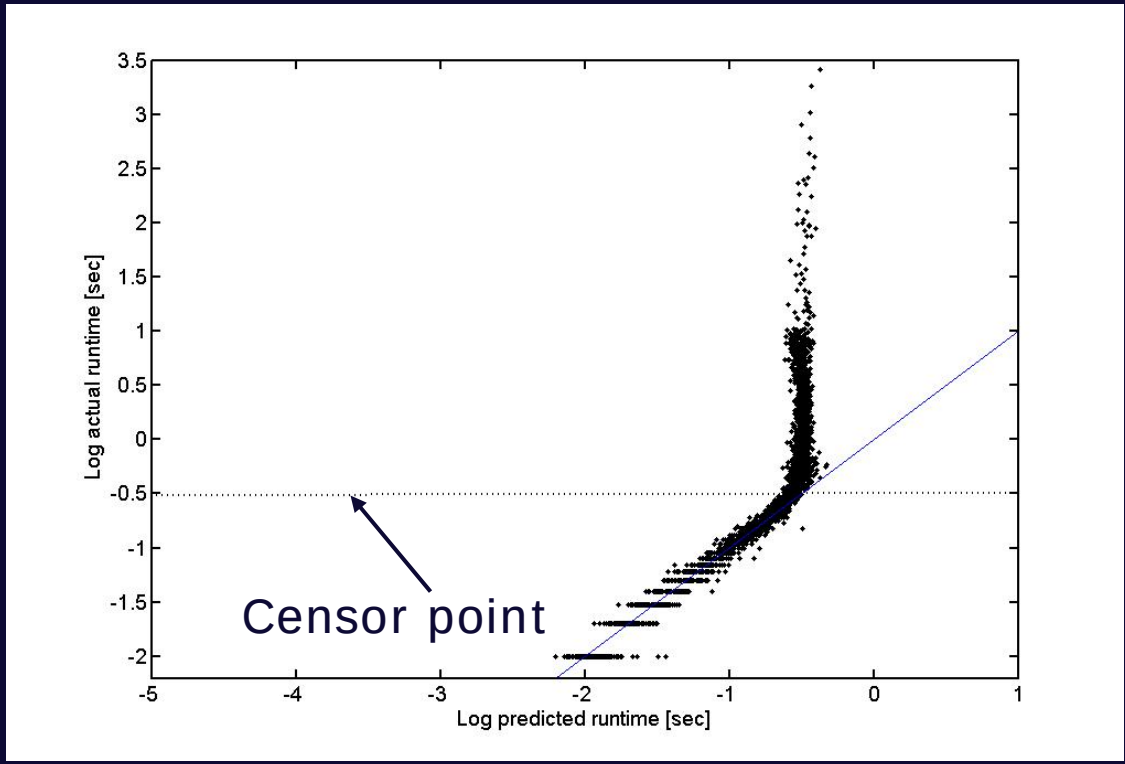


How to deal with censored data

A: Drop them

B: Finished at cutoff

C: Censored sampling

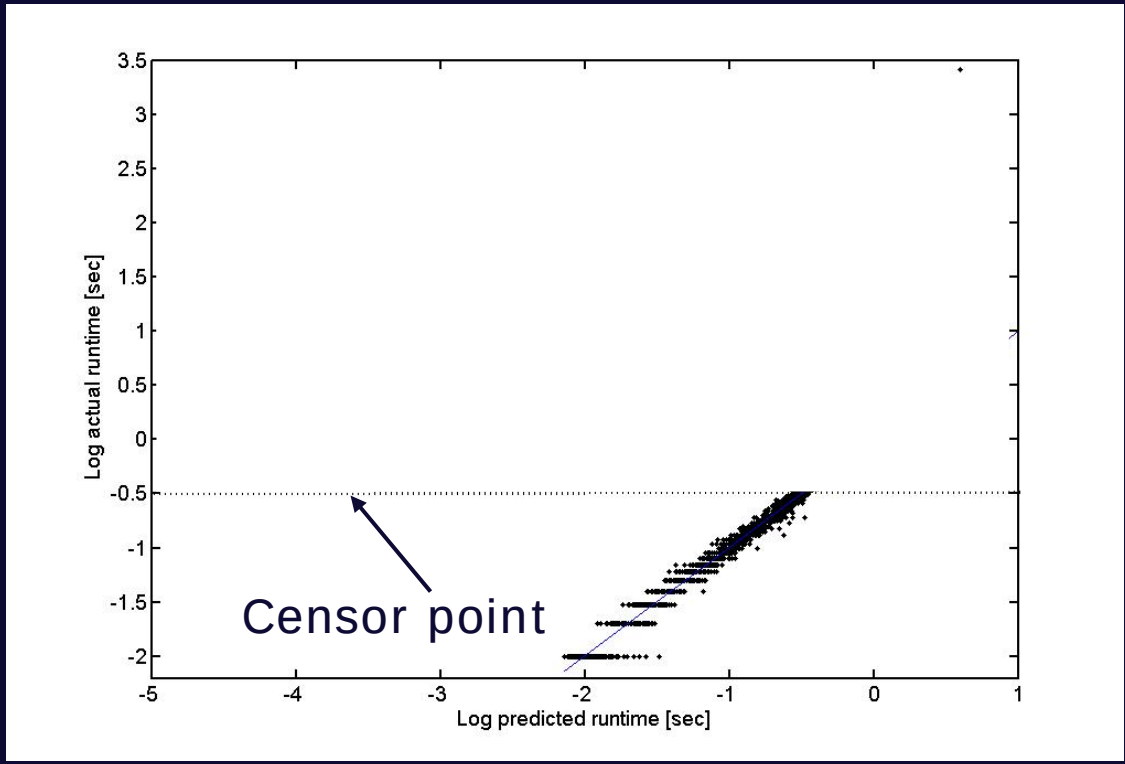


How to deal with censored data

A: Drop them

B: Finished at cutoff

C: Censored sampling



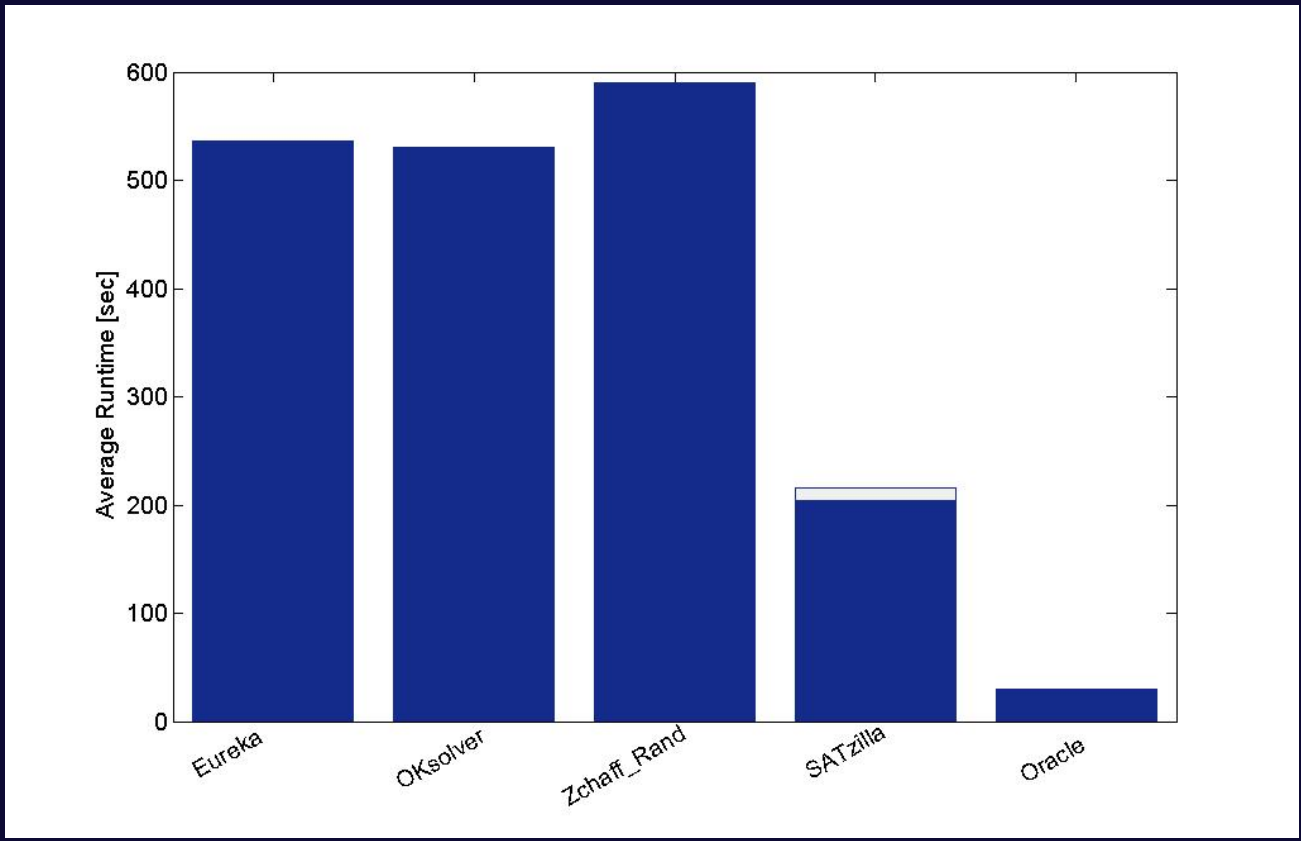
Improve EHM (using Hierarchal Hardness Models)

- EHM often more accurate, much simpler when trained with SAT/UNSAT samples only [Nudelman, et al. 2004]

- Building hierarchal hardness model by approximating a model selection oracle

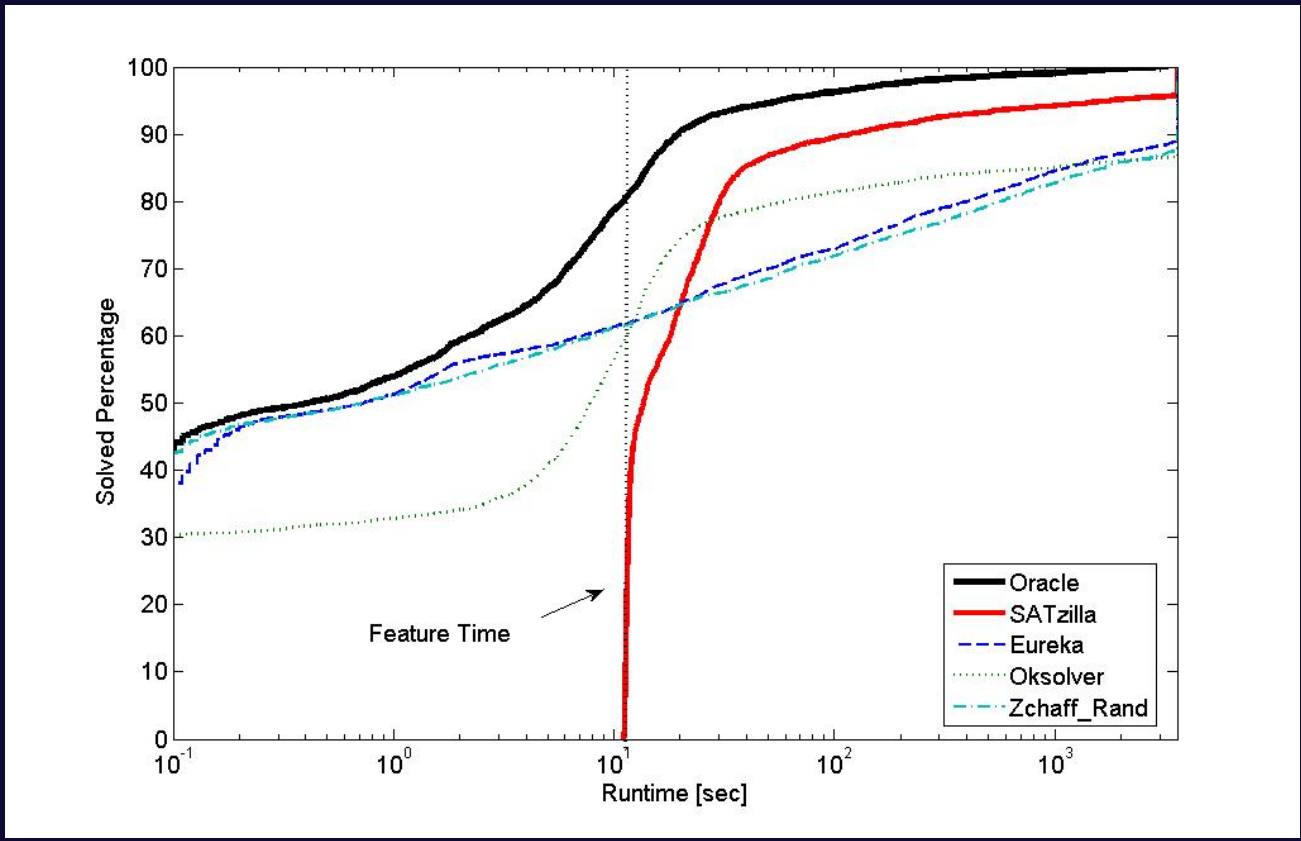
  → Mixture of experts problem with fixed experts

SATzilla-07 for QCP



Average Runtime

SATzilla-07 for QCP



Empirical CDF

2007 SAT Competition

Three submissions for
2007 SAT Competition
BIG-MIX for all three categories (demo)
RANDOM
HANDMADE

SATzilla-07 for SAT Competition

- Target Distribution
  - Previous SAT competition and SAT Race
- Solver (with/without preprocessing, Hyper)
  - Eureka, Kcnfs2006, March_dl2004, Minisat2.0 Vallsat, Rsat1.04, Zchaff_Rand
- Features
  - Reduce probing time to 1 second
  - Only cheap features, total about 3 seconds
- Pre-Solvers
  - March_dl 5 seconds, SAPS 2 seconds

SATzilla-07 for SAT Competition

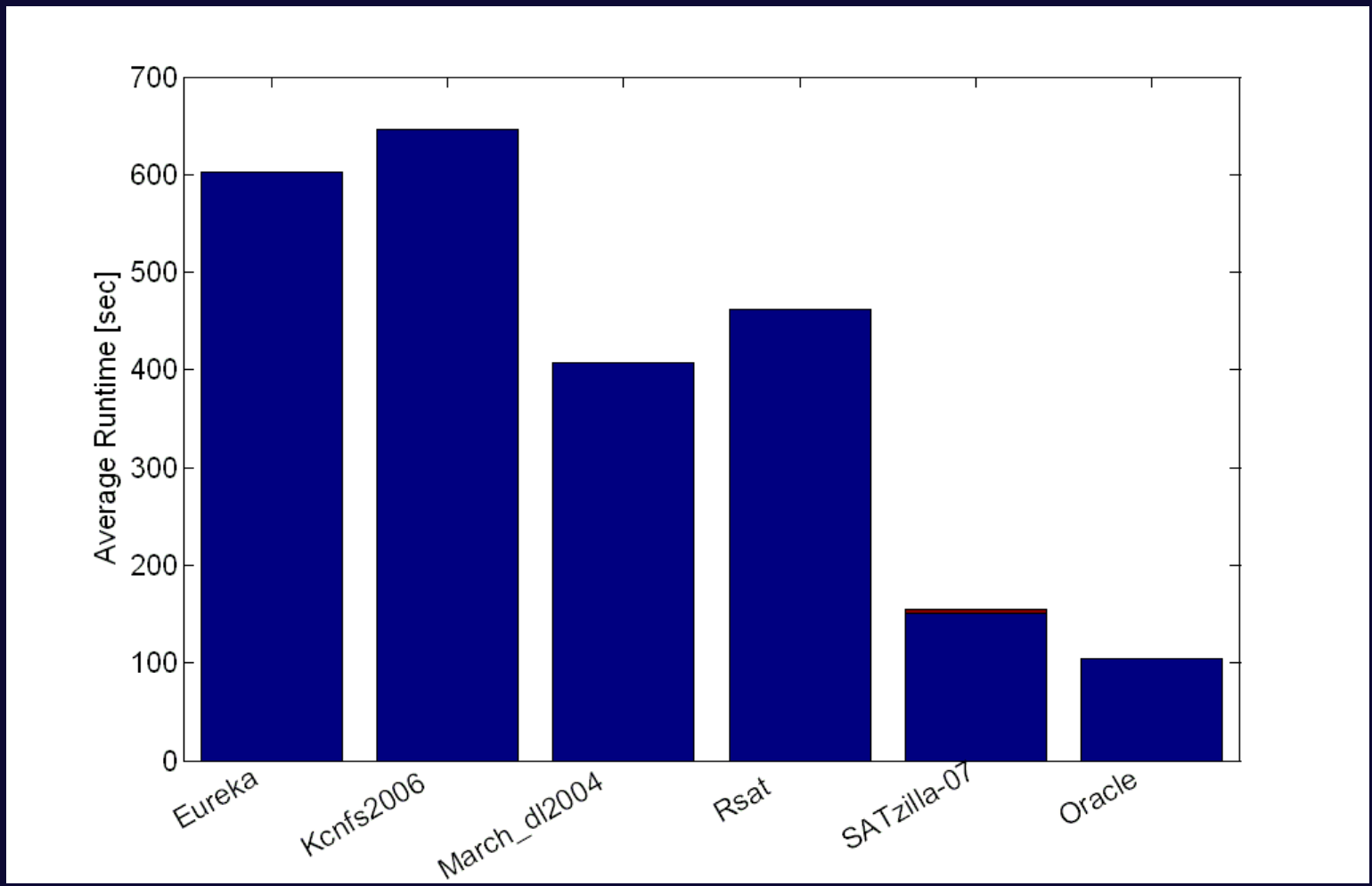
- "Winner-take-all" solver
  - March_dl2004
- Final candidates
  - BIG_MIX

    Eureka, Kcnfs2006, March_dl2004, Rsat
  - RANDOM

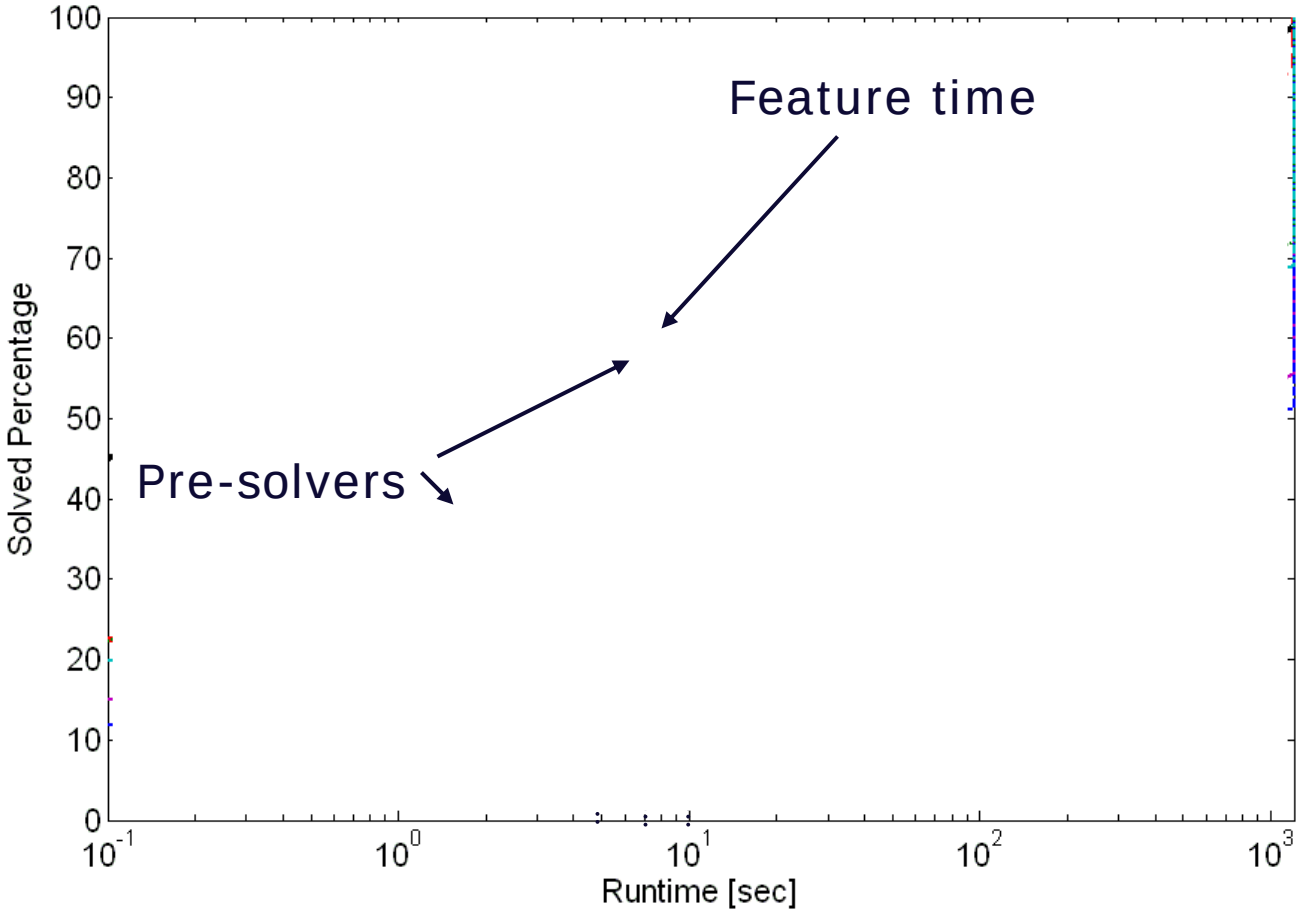
    March_dl2004, Kcnfs2006, Minisat2.0+
  - HANDMADE

    March_dl2004, Vallst, March_dl2004+, Minisat2.0+, Zchaff_Random+

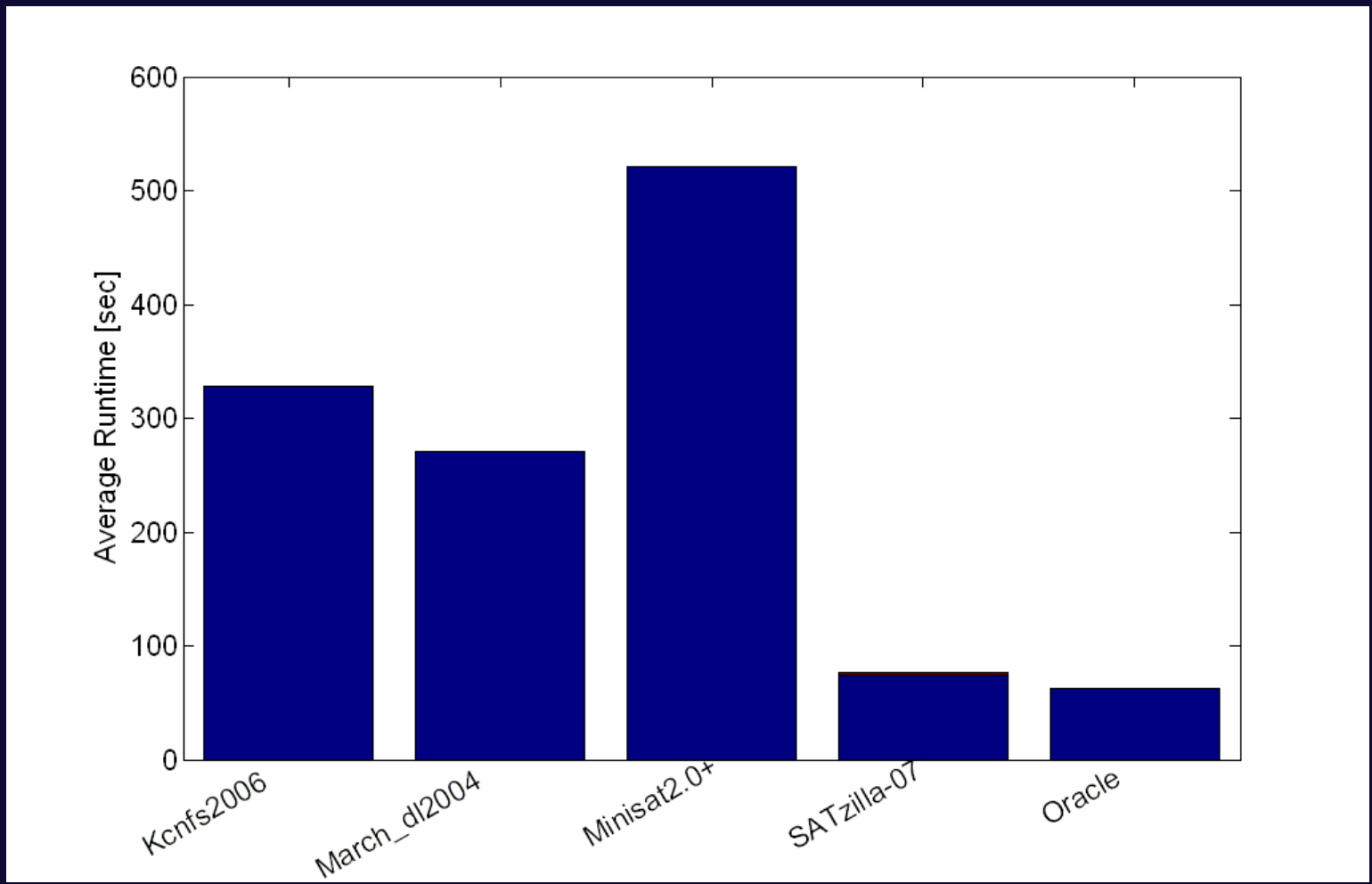
SATzilla-07 for BIG-MIX



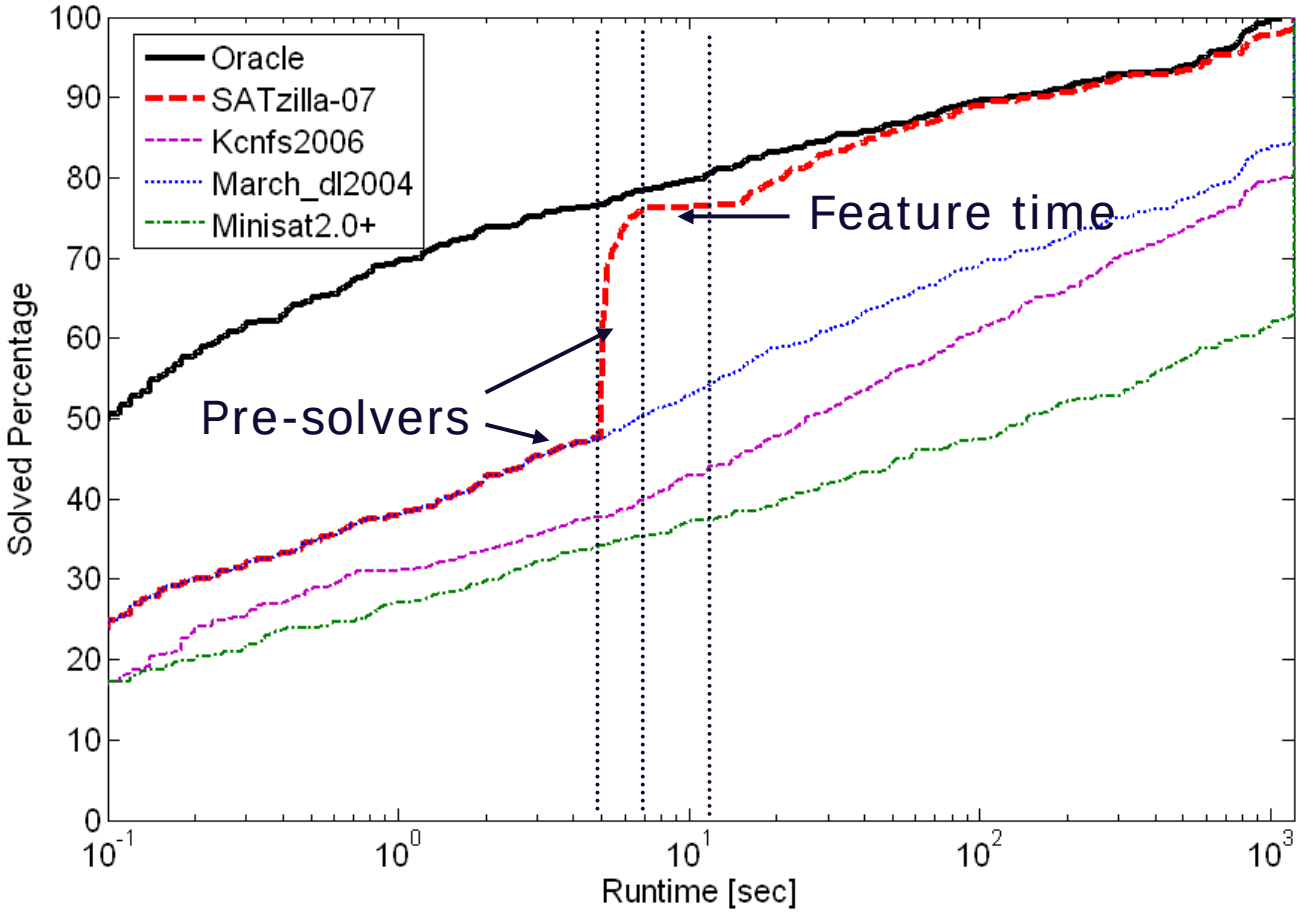
SATzilla-07 for BIG-MIX



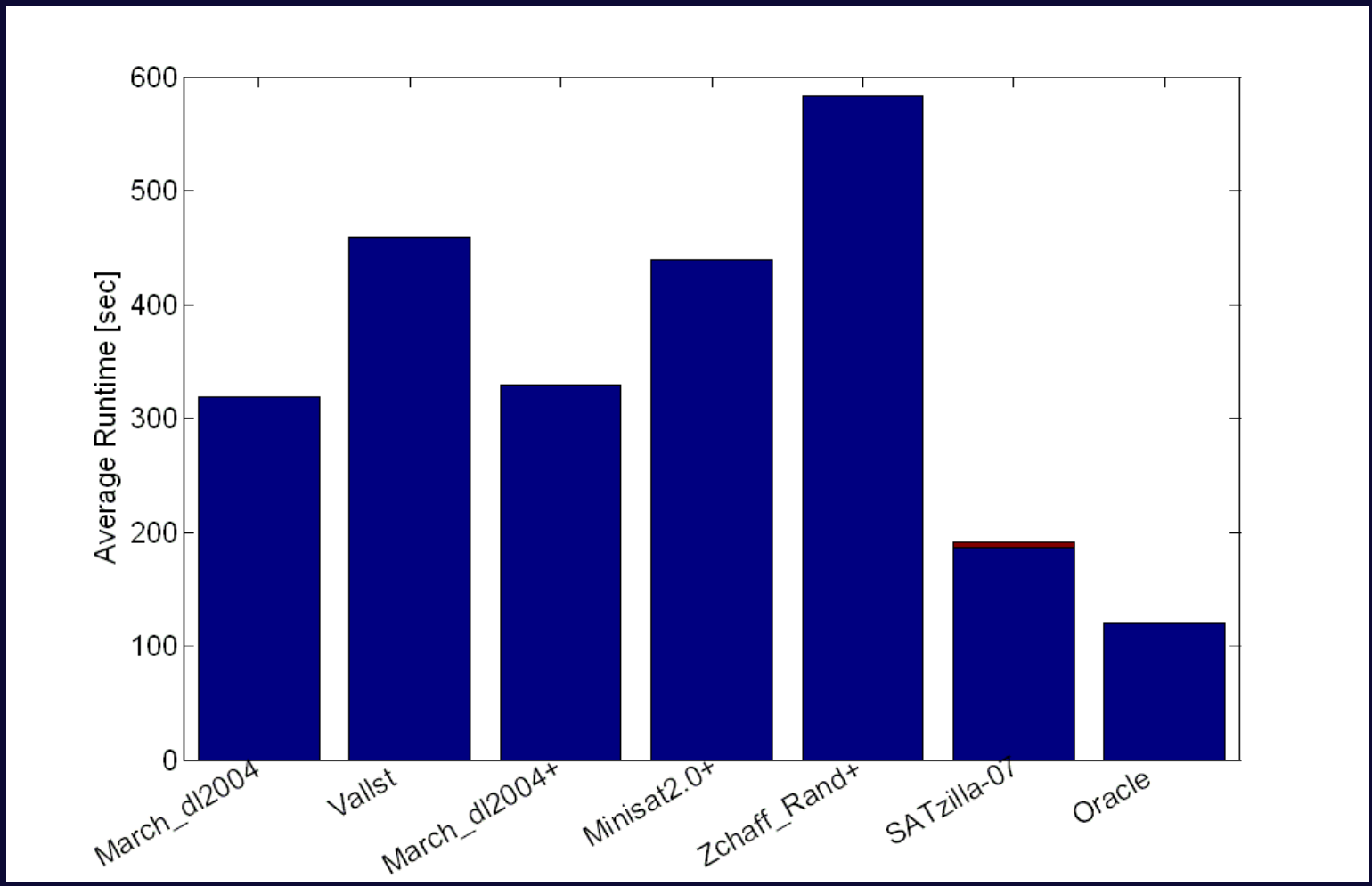
SATzilla-07 for RANDOM



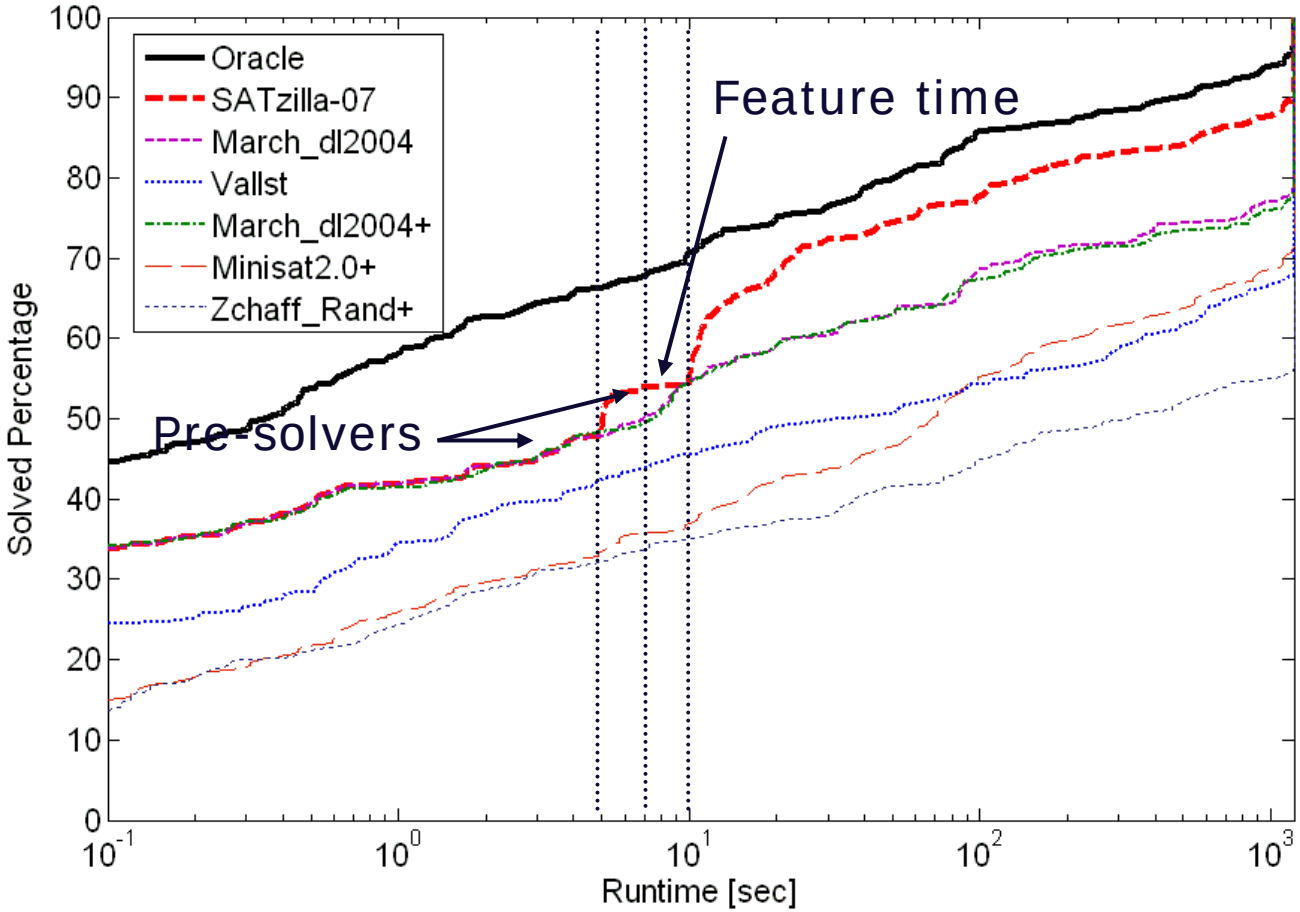
SATzilla-07 for RANDOM



SATzilla-07 for HANDMADE



SATzilla-07 for HANDMADE



Conclusions

- Can combine algorithms into portfolios, improving performance and robustness
- SATzilla approach has been proven to be successful in real world competition
- With more training data and more solvers, SATzilla can be even better

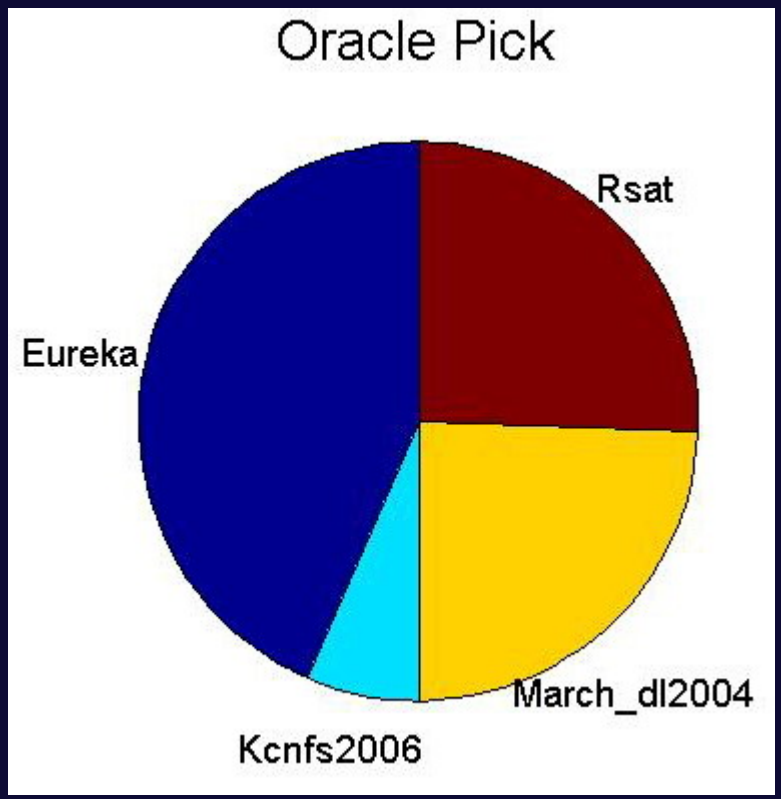
Ongoing research

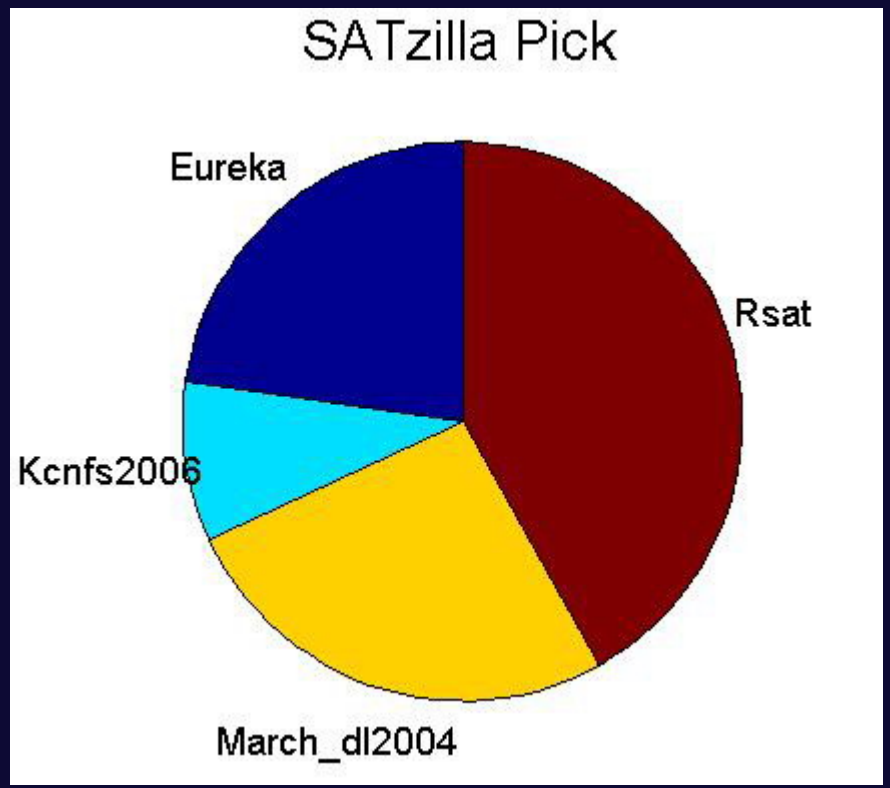
- SATzilla for industrial category
  - Use the same approach, SATzilla is 25% faster and solves 5% more instances
- Score function
  - Optimize objective function other than runtime
- Local search
  - Improve SATzilla performance by using local search solvers as component

Special Thanks

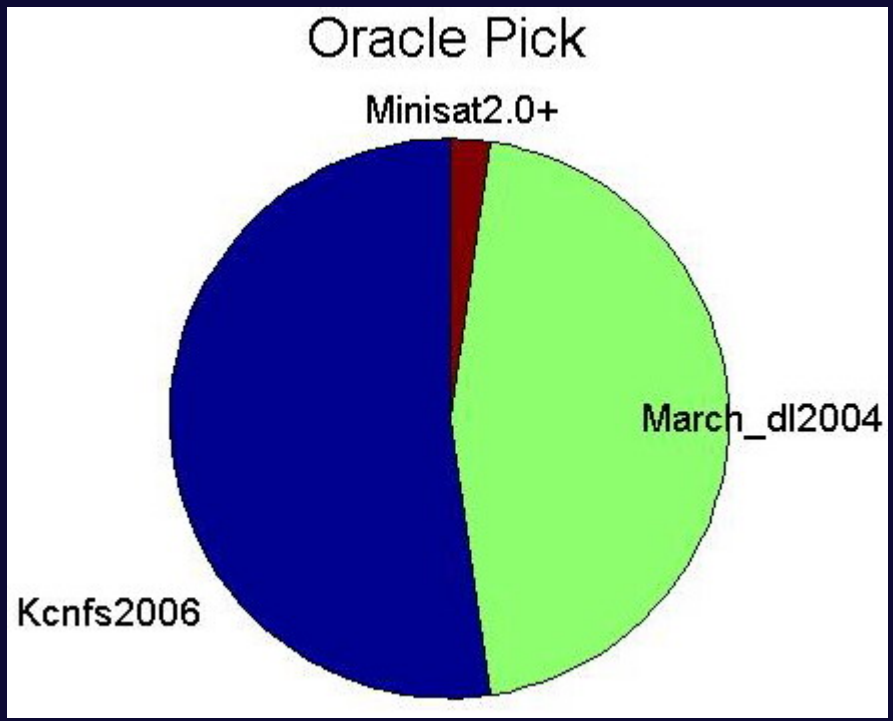
- Creators of solvers
  - Alexander Nadel, Moran Gordon, Amit Palti and Ziyad Hanna (Eureka)
  - Marijn Heule, Hans van Maaren (March_dl2004)
  - Niklas Eén, Niklas Sörensson (Minisat2.0)
  - Oliver Kullmann (OKsolver)
  - Knot Pipatsrisawat and Adnan Darwiche (Rsat 1.04)
  - Daniel Vallstrom (Vallst)
  - Yogesh S. Mahajan, Zhaohui Fu and Sharad Malik (Zchaff_Rand)

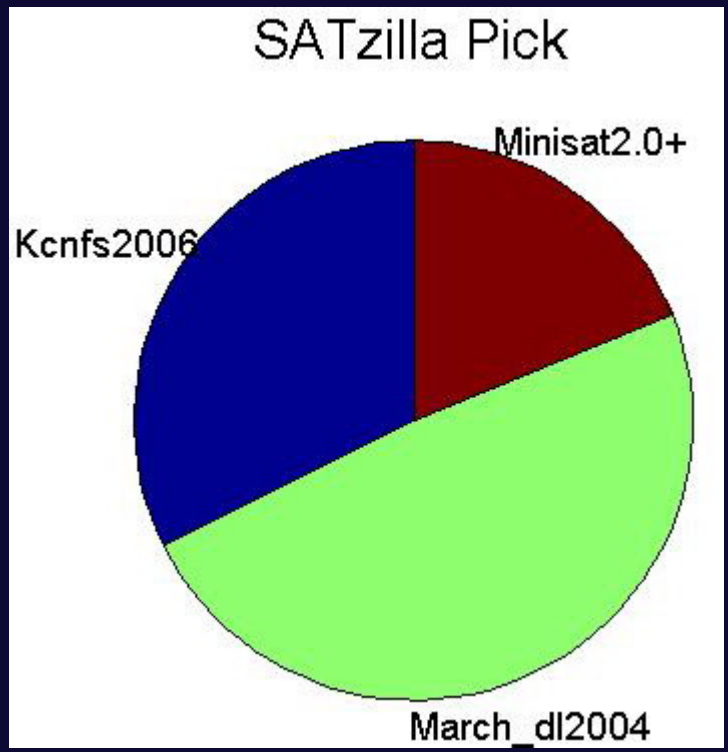
SATzilla Pick for BIG_MIX





SATzilla Pick for RANDOM





SATzilla Pick for HANDMADE