Algorithm Portfolios through Empirical Hardness Models

Case Studies on Combinatorial Auction Winner Determination and Satisfiability

Kevin Leyton-Brown

University of British Columbia

Eugene Nudelman

James McFadden

Galen Andrew

Yoav Shoham

Stanford University

We'd like to acknowledge assistance from Ryan Porter, Carla Gomes and Bart Selman, and support from the Cornell Intelligent Information Systems Institute, a Stanford Graduate Fellowship and DARPA (F30602-00-2-0598).

The Algorithm Selection Problem

- What is the best algorithm for a given problem?
  - worst-/average-case measure doesn't tell the whole story
  - ideally, select algorithm on a per-instance basis [Rice]

- Our approach:
  - Identify:
    - a target distribution of problem instances, D
    - a set of algorithms, where each algorithm has a significant probability of outperforming the others on instances drawn from D
    - polytime-computable features of problem instances
  - Learn per-algorithm empirical hardness models
  - Use the models to construct an algorithm portfolio by choosing the algorithm with the best predicted runtime

Combinatorial Auction Winner Determination

- Equivalent to weighted set packing
- Input: n goods, m bids $< S_i, p_i >$, $S_i \subseteq \{1, \ldots, n\}$
- Objective: find revenue-maximizing non-conflicting allocation

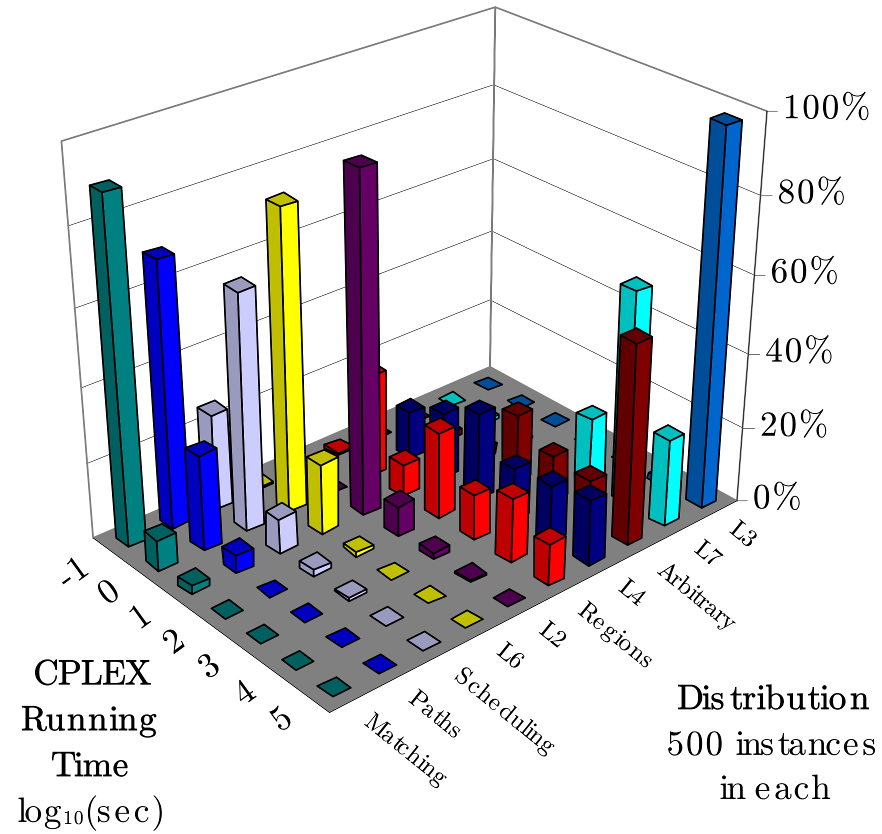
$$
\begin{array}{lll}
\text{maximize:} & \displaystyle\sum_{i=1}^{m} x_i p_i & \\
\text{subject to:} & \displaystyle\sum_{i \mid g \in S_i} x_i \leq 1 & \forall g \\
 & x_i \in \{0, 1\} & \forall i
\end{array}
$$

WDP: Runtime Variation

- Complete algorithms:
  - CPLEX [ILOG Inc.]
  - CASS [Leyton-Brown et.al],
  - GL [Gonen and Lehman]

- Gathered runtime data using various distributions
  - randomly sampled generator's parameters for each instance

- Even holding problem size constant, runtimes vary by many orders of magnitude across and within distributions



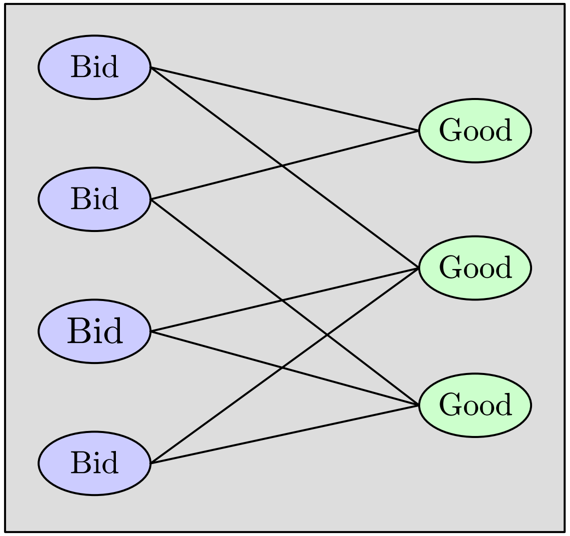
WDP: Features

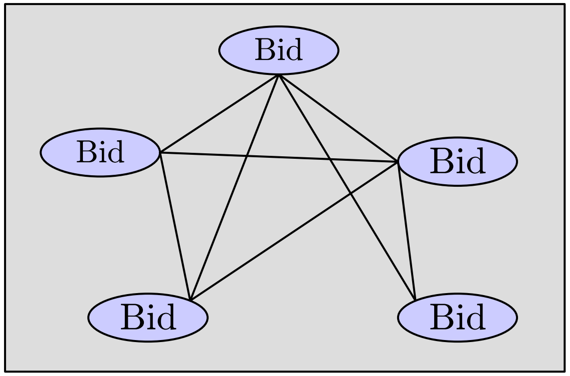
$$\text{maximize: } \sum_{i=1}^{m} x_i p_i$$

$$\text{subject to: } \sum_{i|g \in S_i} x_i \leq 1 \quad \forall g$$

$$0 \leq x_i \leq 1 \quad \forall i$$

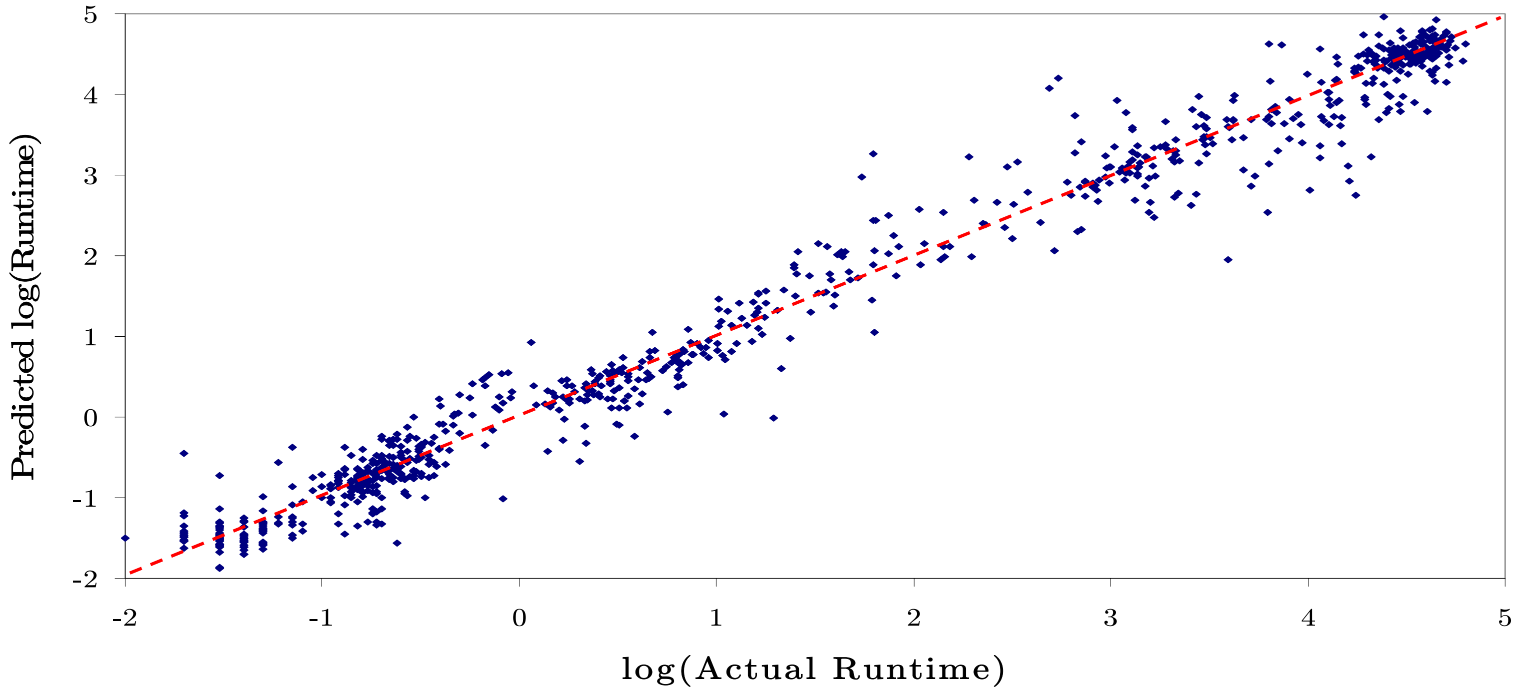
1. Linear Programming
   – L_1, L_2, L_∞ norms of integer slack vector
2. Price
   – stdev(prices)
   – stdev(avg price per good)
   – stdev(average price per sqrt(good))
3. Bid-Good graph
   – node degree stats (max, min, avg, stdev)
4. Bid graph
   – node degree stats
   – edge density
   – clustering coefficient (CC), stdev
   – avg min path length (AMPL)
   – ratio of CC to AMPL
   – eccentricity stats (max, min, avg, stdev)





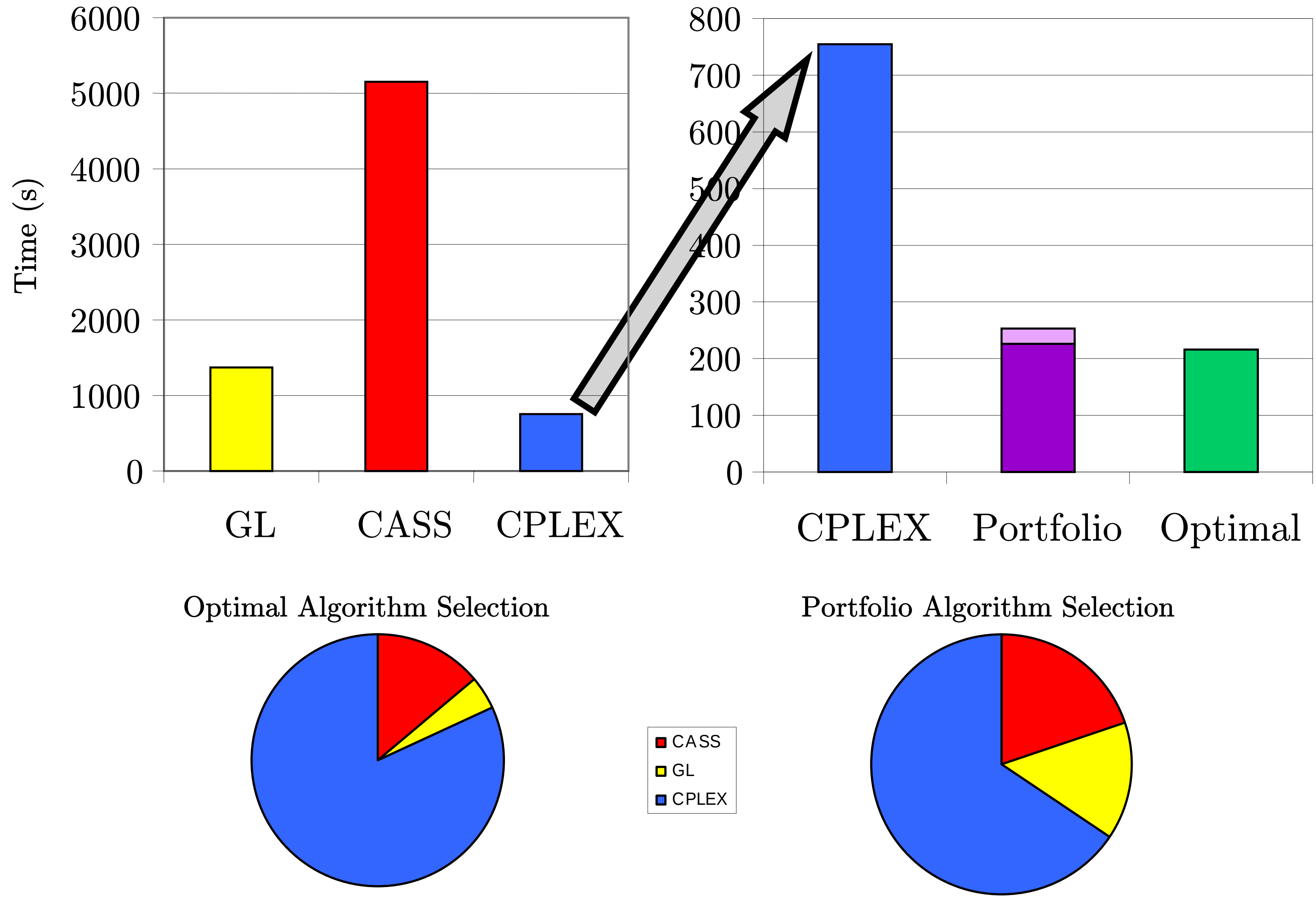
WDP: Empirical Hardness Models

- Quadratic regression can be used to learn very accurate models
  - predicting $\log_{10}$ of CPLEX runtime



  - Root mean squared error: 0.216 (test data)

WDP: From Models to a Portfolio



SATZilla: A Portfolio for SAT

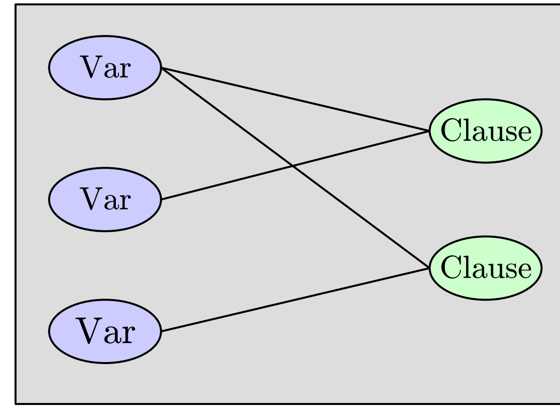
- Algorithms in the portfolio:
  - 2clseq [Bacchus]
  - OKsolver [Kullmann]
  - Satz-Rand [Kautz, Li]
  - zChaff [Zhang]
  - Limmat [Biere]
  - relsat [Bayardo]
  - SATO [Zhang]
  - Jerusat [Nadel]

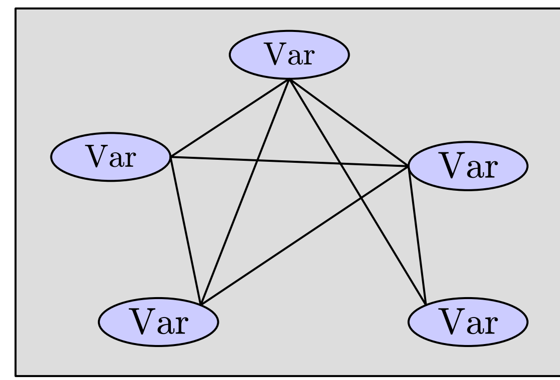
- Satzilla2 (Hors-Concours) added:
  - eqsatz [Li]
  - HeerHugo [Groote]
  - AutoWalkSat [Patterson, Kautz] (preprocessing)

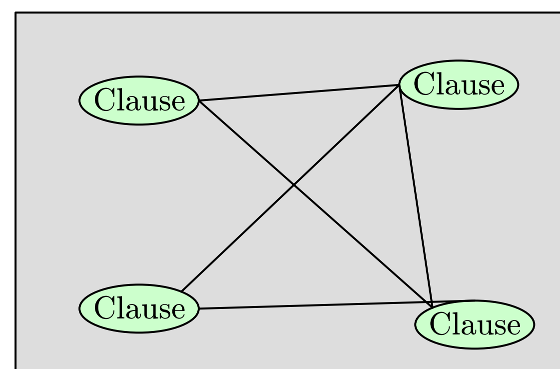
- Developed in just over two weeks!

SATzilla: Features

1. Problem Size: #vars, #clauses, #vars/#clauses
   - rest of features are normalized by these
2. Graphs:
   - Variable-Clause (*VCG*, bipartite)
   - Variable (*VG*, edge whenever two variables occur in the same clause)
   - Clause (*CG*, edge whenever two clauses share a variable with opposite sign)
     - compute stats=(max, min, stdev, mean, entropy) over node degrees
       - for VCG, both for vars and clauses
     - # of unary, binary, ternary clauses
     - stats of the CG clustering coefficients







SATzilla: Features

k1

3. Stats of #positive/#negative literals in each clause
4. Stats of #positive/#negative occurrences for each var
5. Horn clauses
   - total #horn clauses
   - stats of #horn occurrences for each var
6. LP relaxation features
   - objective value
   - stats of integer slacks
   - #vars set to an integer
7. Probing features
   - DPLL probing features (to depth 256)
     - #unit props after reaching depths 1, 4, 16, 64, 256
   - Local search probing (100 probes, each probe runs to plateau/max)
     - stats of climb height (in #clauses)
     - stats of fraction of satisfied clauses
     - stats of #steps taken
     - stats of break counts/#vars
   - Search space size probing (5000 random search paths with unit-prop)
     - average depth to contradiction, estimate log-num-nodes in search tree

k1 # pos/# neg: should be abs(0.5 - #pos / (#pos + #neg)) so that flipping all pos and neg doesn't change the stat
kevinlb, 1/1/2004

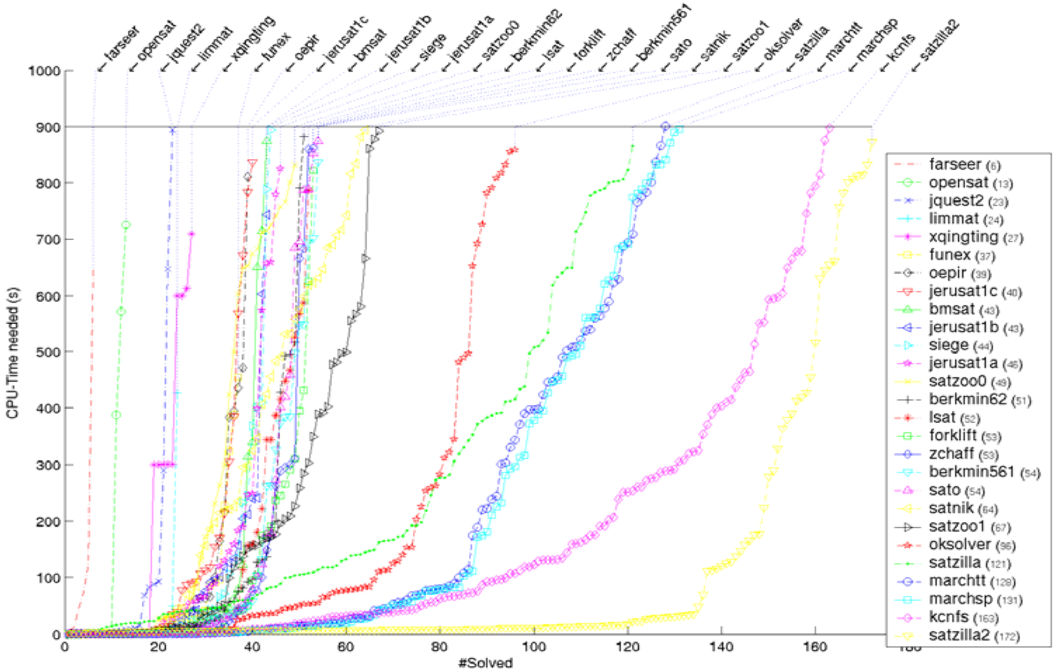
SATzilla: Models and Portfolio

- Learned linear regression models for each algorithm
  - trained on more than 20000 instances
    - included 2002 competition instances
    - highly skewed towards random instances
  - training set preprocessed to exclude instances that were solved by all solvers, or by none of them
  - terrible RMSE on test set
  - enough predictive power to discriminate well

- On the training set, SATzilla's choice takes on average 92 seconds longer to run than the optimal choice
  - gives SATzilla an edge over its subsolvers, especially on harder instances

SATzilla: SAT-2003 Competition

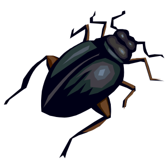
- 2nd in Random instances track
- 3rd in Handmade track; 2nd in Handmade track, SAT only



- Only solver with good performance in more than one track
- Success measured in #series solved, not #benchmarks solved
  - Satzilla 2 solved more random instances than kcnfs

SATzilla: Areas for Improvement

- Add new algorithms to the portfolio
  - SATzilla outperformed all its constituent algorithms
- Construct better models
  - as we continue to study and analyze SAT data, our model accuracy is increasing
- Spend more development time to eliminate bugs
  - LP features timed out on many industrial benchmarks



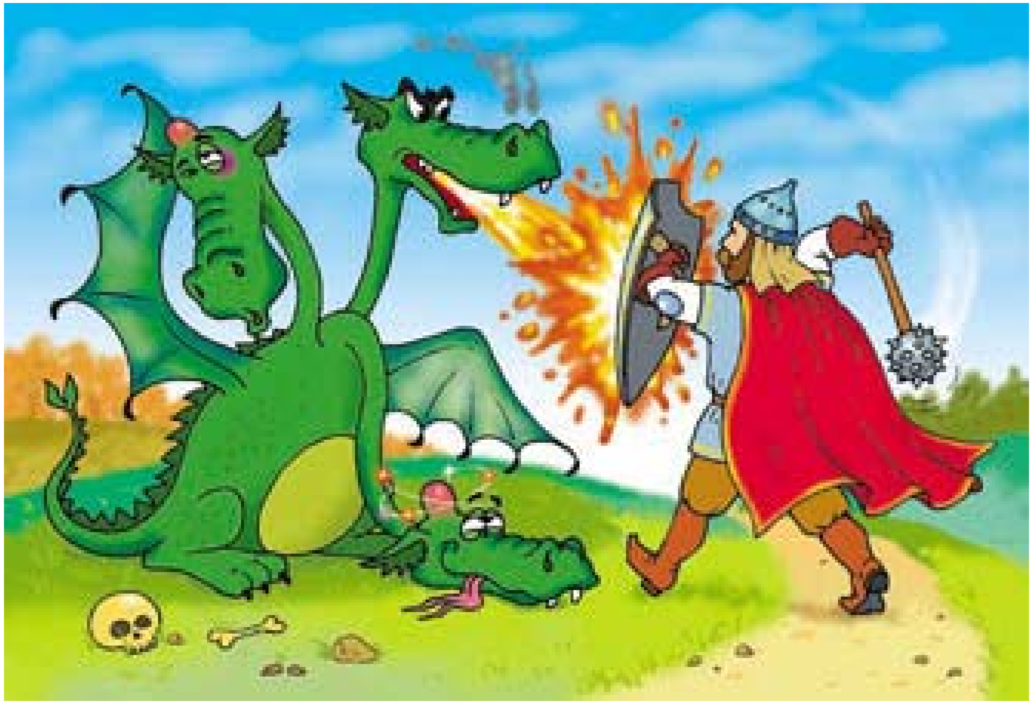
    - instead of using a fallback solver (zChaff), SATzilla picked one essentially at random, but most don't do well on industrial
  - some "random" instances were solved but didn't count!
    - Relsat was chosen, and actually solved them, but it had an output bug ☹

Conclusions

- WDP
  - models: very mature, high accuracy
  - algorithms: one is dominant, limiting the size of possible gains from a portfolio approach

- SAT
  - models: more of a proof of concept, much room for improvement. However, discrimination accuracy is much better than prediction accuracy.
  - algorithms: many are strong and correlation is fairly low, making this an excellent domain for future study

Conclusions



Overall, our techniques provide a quick and relatively automatic blueprint for building algorithm portfolios, suitable when there are:

- two or more algorithms with relatively uncorrelated runtimes
- a set of good features
- lots of data