Learning the Empirical Hardness of Combinatorial Auctions

Kevin Leyton-Brown
Eugene Nudelman
Yoav Shoham

Computer Science
Stanford University

Thanks to:
Carla Gomes, Bart Selman, Rámon Béjar
Ioannis Vetsikas, and Henry Kautz

Introduction

- Recent trend: study of average/empirical hardness as opposed to the worst-case complexity ($\mathcal{NP}$-Hardness) [Cheeseman et al.; Selman et al.]
- Our proposal: predict the running time of an algorithm on a particular instance based on features of that instance
- Today:
  - a methodology for doing this
  - its application to the combinatorial auction winner determination problem (WDP)

Why?

- Predict running time
  - for its own sake
  - build algorithm portfolios
- Theoretical understanding of hardness
  - tune distributions for hardness
  - improve algorithms
- Problem specific
  - WDP: design bidding rules

Related Work

- Decision problems:
  - phase transitions in solvability, corresponding to hardness spike [Cheeseman et al.; Selman et al.]
  - solution invariants: e.g., backbone [Gomes et al.]
- Optimization problems:
  - experimental:
    - reduce to decision problem [Zhang et al.]
    - introduce backbone concepts [Walsh et al.]
  - theoretical:
    - polynomial/exponential transition in search algorithms [Zhang]
    - predict A* nodes expanded for problem distribution [Korf, Reid]
- Learning
  - dynamic restart policies [Kautz et al.]

Combinatorial Auctions

- Auctioneer sells a set of non-homogeneous items
- Bidders often have complex *valuations*
  - *complementarities*
    - *e.g.* $V(TV \,\&\, VCR) > V(TV) + V(VCR)$
  - *substitutabilities*
    - $V(TV_1 \,\&\, TV_2) < V(TV_1) + V(TV_2)$
- Solution: allow bids on *bundles* of goods
  - achieves a higher revenue and social welfare than separate auctions
- Two hard problems:
  - Expressing valuations
  - Determining optimal allocation

Winner Determination Problem

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

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

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

$$x_i \in \{0, 1\} \qquad \forall i$$

- Even constant factor approximation is $\mathcal{NP}$-Hard
- Square-root approximation known
- Polynomial in the number of bids

WDP Case Study

- Difficulty: highly parameterized, complex distributions
- Hard to analyze theoretically
  - large variation in edge costs and branching factors throughout the search tree [Korf, Reid, Zhang]
- Too many parameters to vary systematically [Walsh et al., Gomes et. al.]
- Parameters affect expected optimum: difficult to transform to decision problem [Zhang et al.]

Methodology

1. Select algorithm
2. Select set of input distributions
3. Factor out known sources of hardness
4. Choose features
5. Generate instances
6. Compute running time, features
7. Learn a model of running time

Methodology

1. **Select algorithm: ILOG's CPLEX 7.1**
2. Select set of input distributions
3. Factor out known sources of hardness
4. Choose features
5. Generate instances
6. Compute running time, features
7. Learn a model of running time

Methodology

1. Select algorithm
2. **Select set of input distributions**
3. Factor out known sources of hardness
4. Choose features
5. Generate instances
6. Compute running time, features
7. Learn a model of running time

WDP Distributions

- Legacy (7 distributions)
  - sample bid sizes/prices independently from simple statistical distributions
- Combinatorial Auctions Test Suite (CATS)
  - Attempted to model bidder valuations to provide more motivated CA distributions
    1. **regions**: real estate
    2. **arbitrary**: complementarity described by weighted graph
    3. **matching**: FAA take-off & landing auctions
    4. **scheduling**: single resource, multiple deadlines for each agent [Wellman]

Methodology

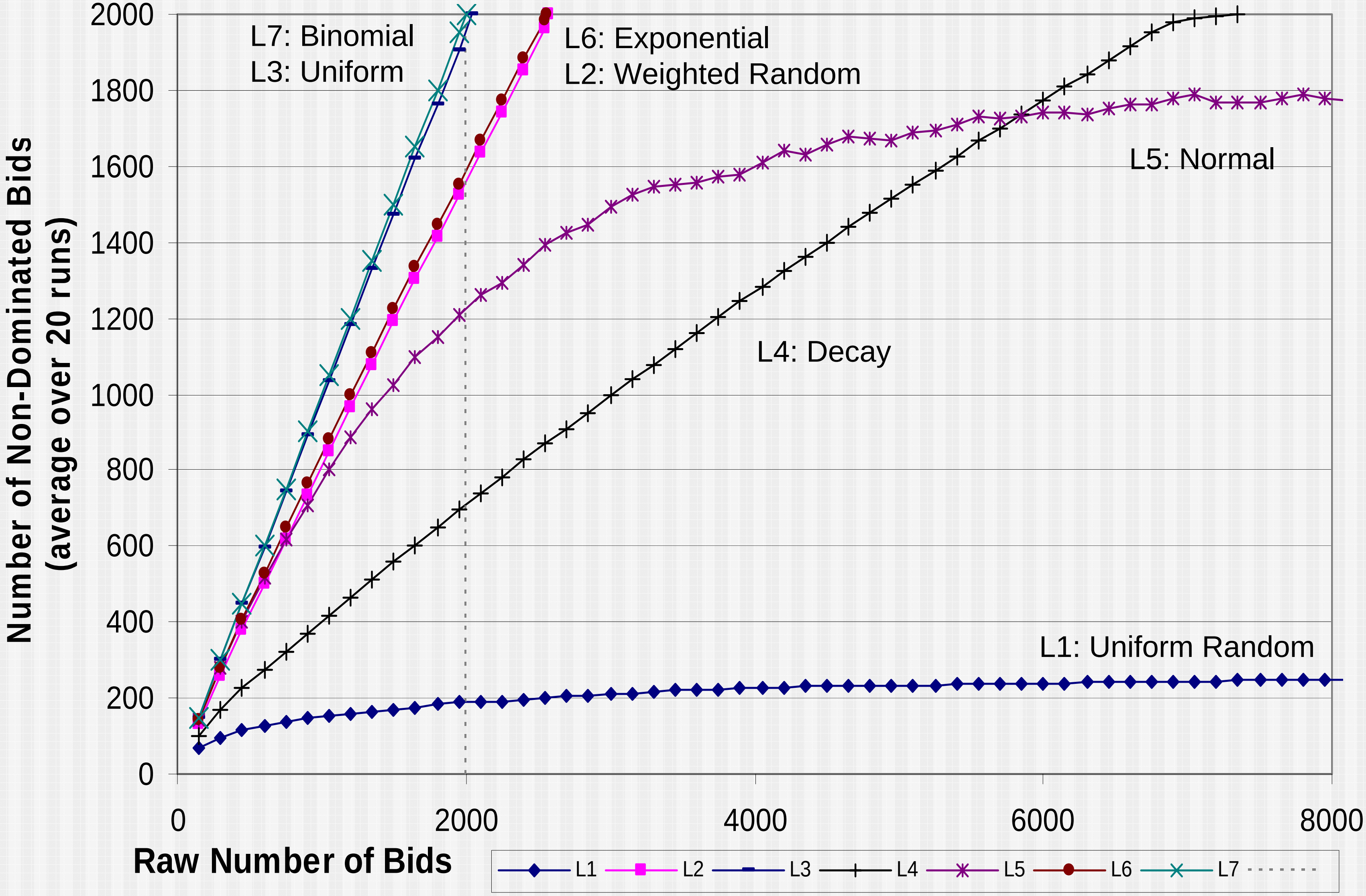
1. Select algorithm
2. Select set of input distributions
3. **Factor out known sources of hardness**
4. Choose features
5. Generate instances
6. Compute running time, features
7. Learn a model of running time

Problem Size

- Some sources of hardness well-understood
  - hold these constant to focus on unknown sources of hardness
- Common: input size
- Problem size is affected by preprocessing techniques! (e.g. arc-consistency)
- WDP: *dominated bids* can be removed
- *(raw #bids, #goods)* is a **very misleading** measure of size for legacy distributions
  - we fix size as (#non-dominated bids, #goods)

Raw vs. Non-Dominated Bids

(64 goods, target of 2000 non-dominated bids)



Methodology

1. Select algorithm
2. Select set of input distributions
3. Factor out known sources of hardness
4. **Choose features**
5. Generate instances
6. Compute running time, features
7. Learn a model of running time

Features

- No automatic way to construct features
  - must come from domain knowledge
- We require features to be:
  - polynomial-time computable
  - distribution-independent
- We identified 35 features
  - after using various statistical feature selection techniques, we were left with 25

Features

- Bid Good Graph (BGG)
  1. Bid node degree stats
  2. Good node degree stats
- Price-based features
  9. std. deviation
  10. stdev price/#goods
  11. stdev price/ √#goods
- Bid Graph (BG)
  3. node degree stats
  4. edge density
  5. clustering coef. and deviation
  6. avg. min. path. length
  7. ratio of 5 & 6
  8. node eccentricity stats
- LP Relaxation
  12. L_1, L_2, L_∞ norms of integer slack vector

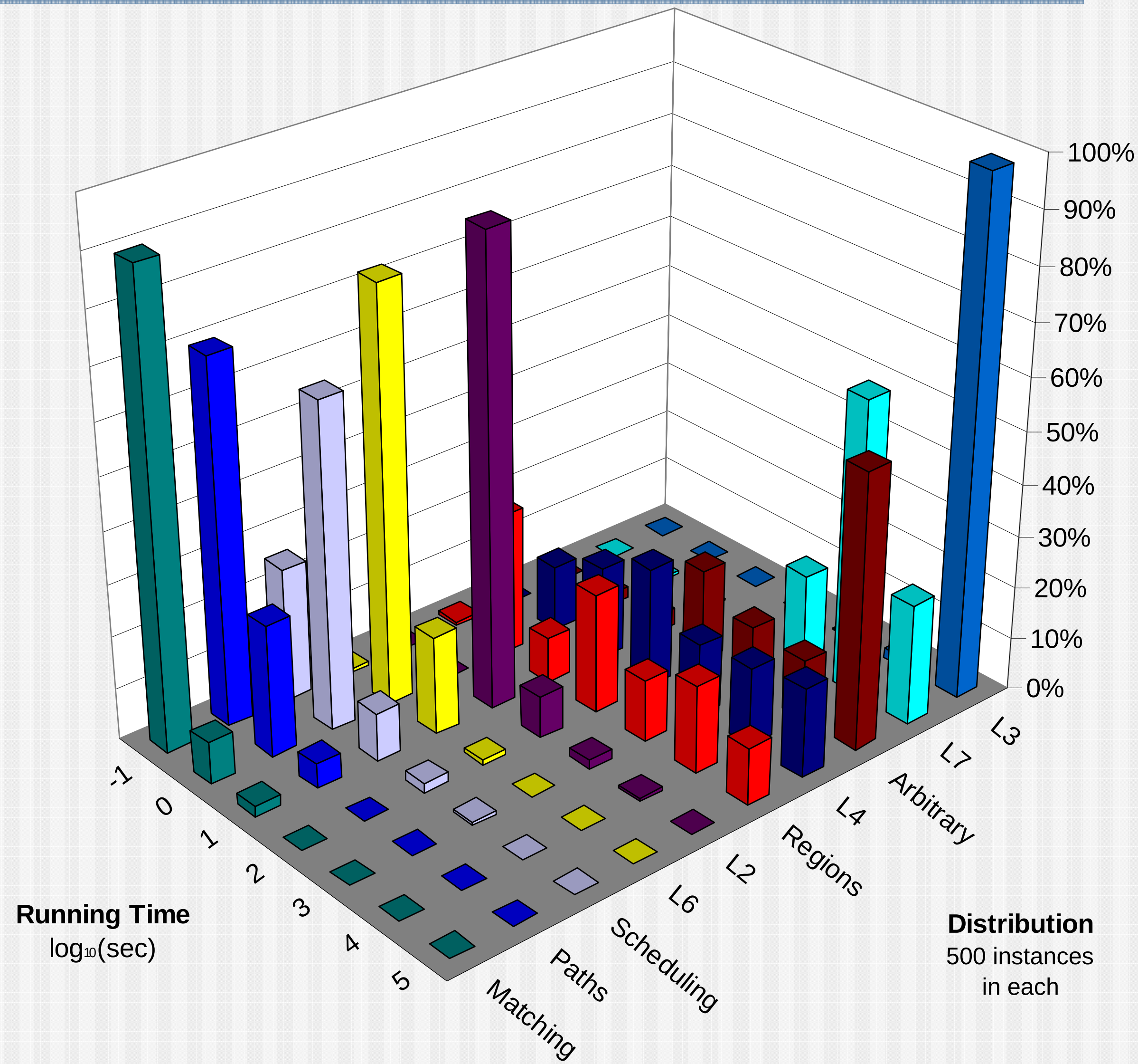
Methodology

1. Select algorithm
2. Select set of input distributions
3. Factor out known sources of hardness
4. Choose features
5. **Generate instances**
6. **Compute running time, features**
7. Learn a model of running time

Experimental Setup

- **Sample** parameters uniformly from range of acceptable values
- 3 separate datasets:
  - 256 goods, 1000 non-dominated bids
  - 144 goods, 1000 non-dominated bids
  - 64 goods, 2000 non-dominated bids
- 4500 instances/dataset, from 9 distributions
- Collecting data took approximately **3 years** of CPU time! (550 MHz Xeons, Linux 2.12)
- Running times varied from 0.01 sec to 22 hours (CPLEX capped at 130000 nodes)

Gross Hardness (256 goods, 1000 bids)



Methodology

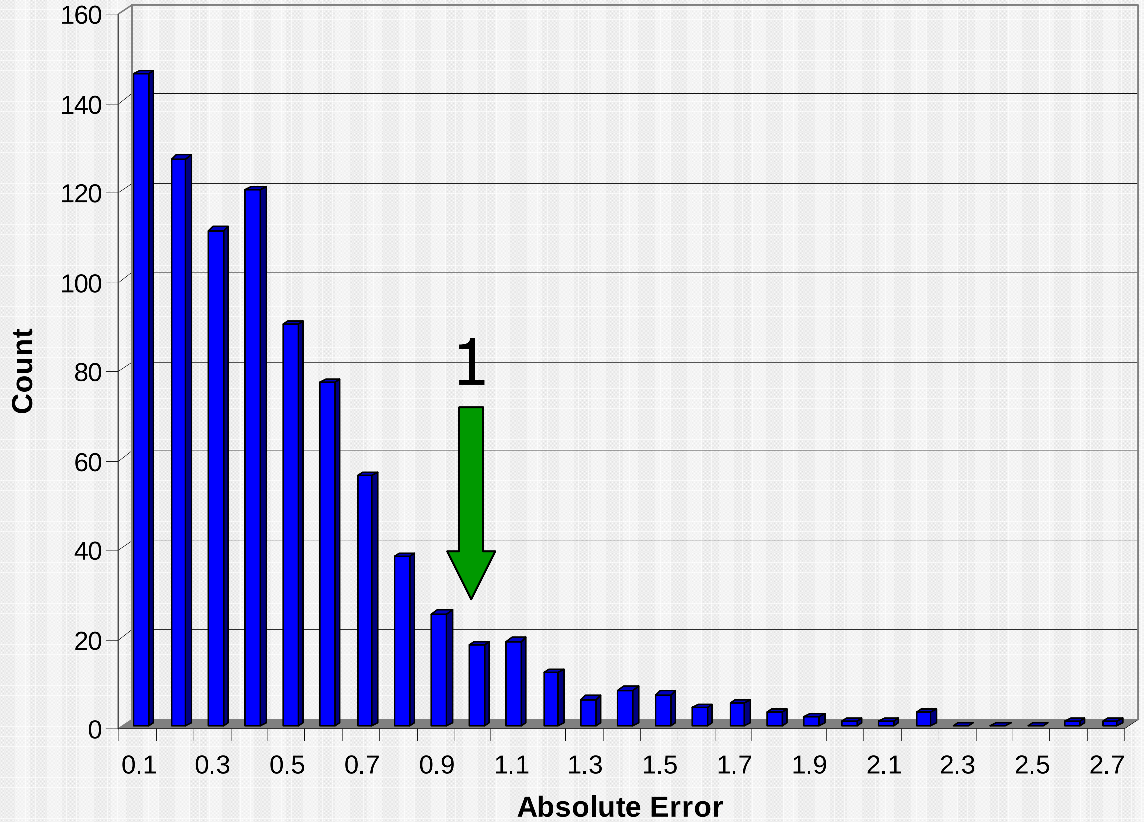
1. Select algorithm
2. Select set of input distributions
3. Factor out known sources of hardness
4. Choose features
5. Generate instances
6. Compute running time, features
7. **Learn a model of running time**

Learning

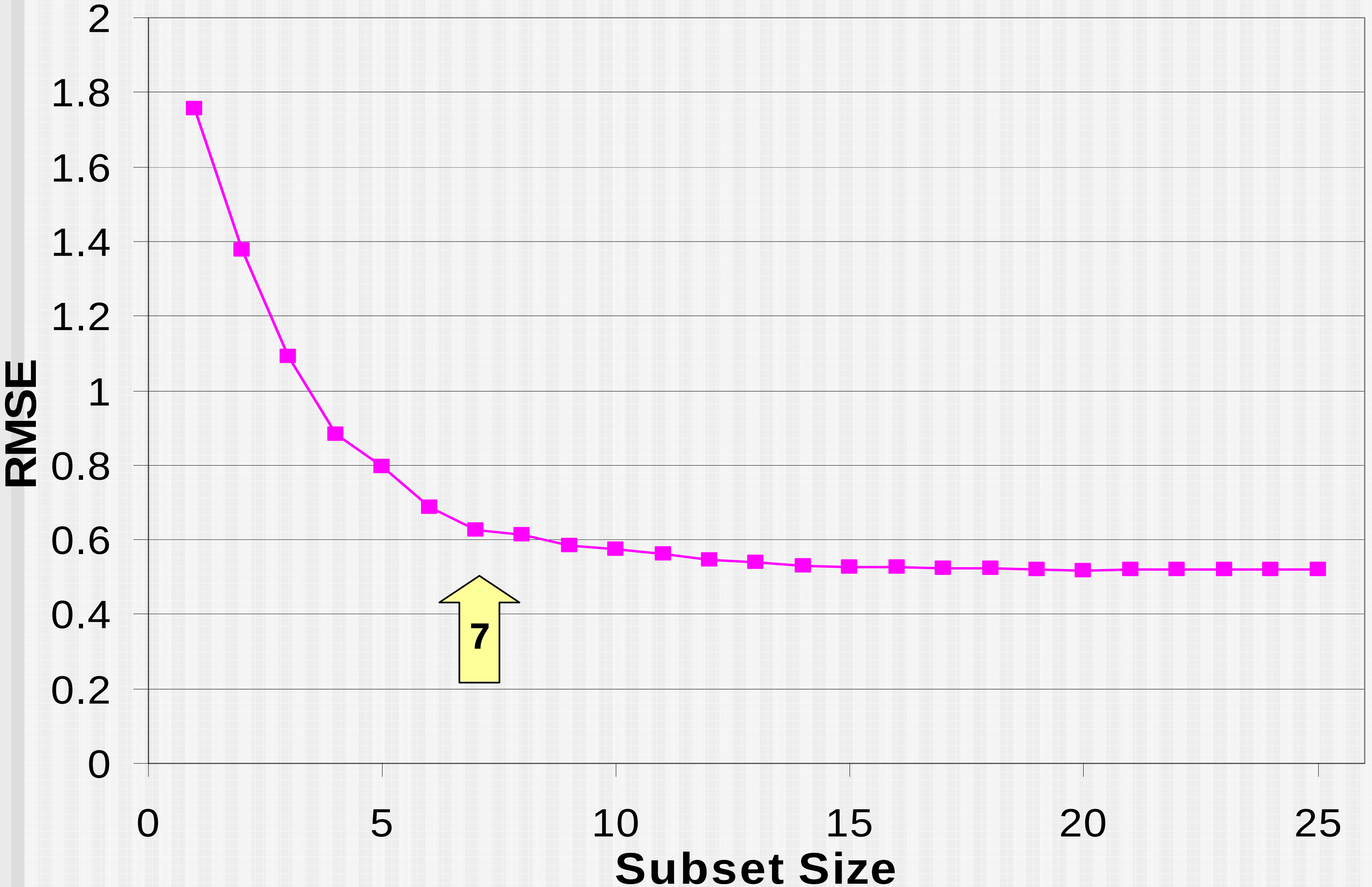
- Classification: misleading error measure
- Statistical regression: learn a continuous function of features that predicts **log of running time**
- Supervised learning: data broken into 80% training set, 20% test set
- Started with simplest technique: linear regression
  - find a hyperplane that minimizes root mean squared error (RMSE) on training data
- Linear regression is useful:
  - as a (surprisingly good) baseline
  - yields a very interpretable model with understandable variables

LR: Error

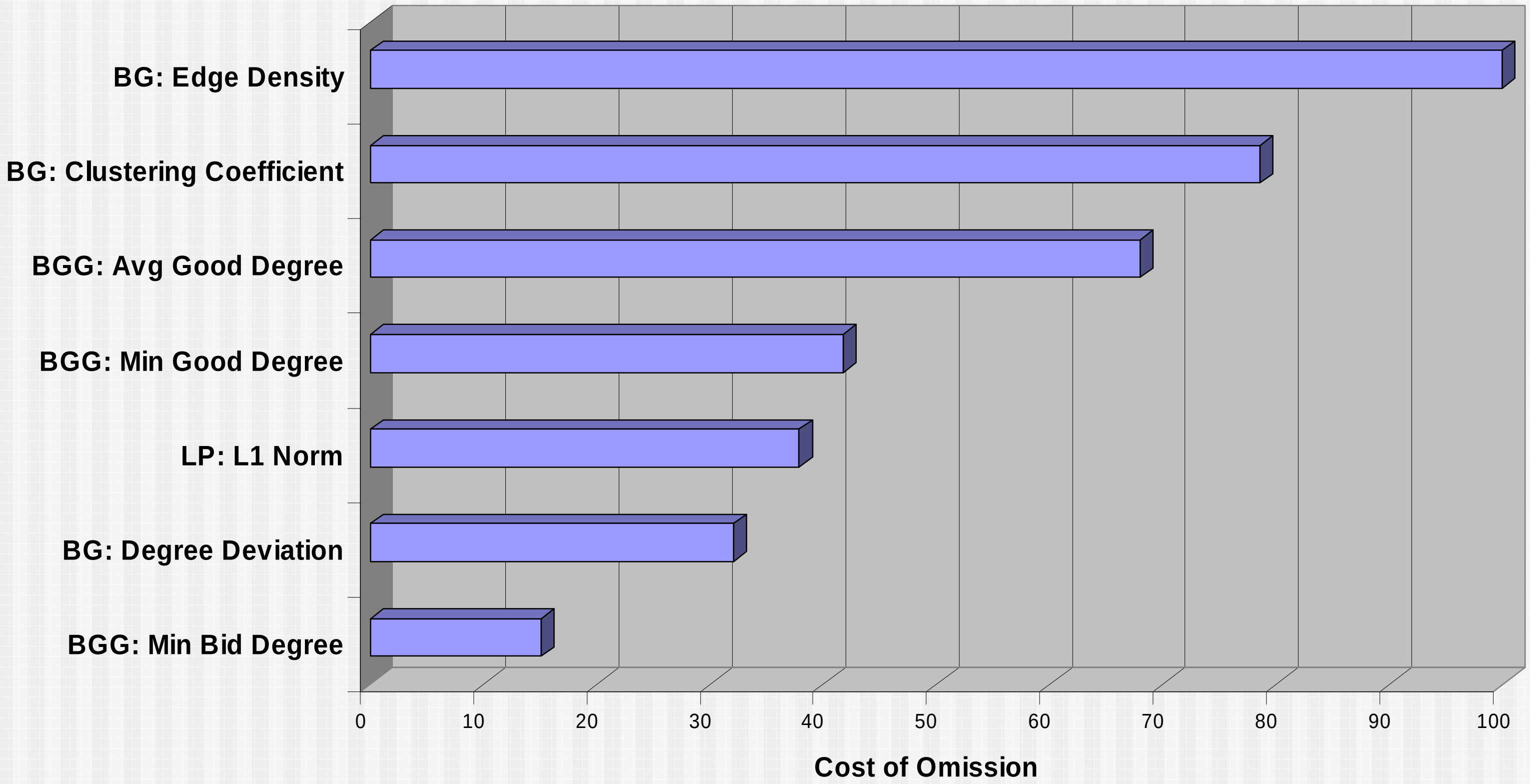
Dataset	RMSE	MAE
1000 Bids, 256 Goods	0.581	0.436



LR: Subset Selection



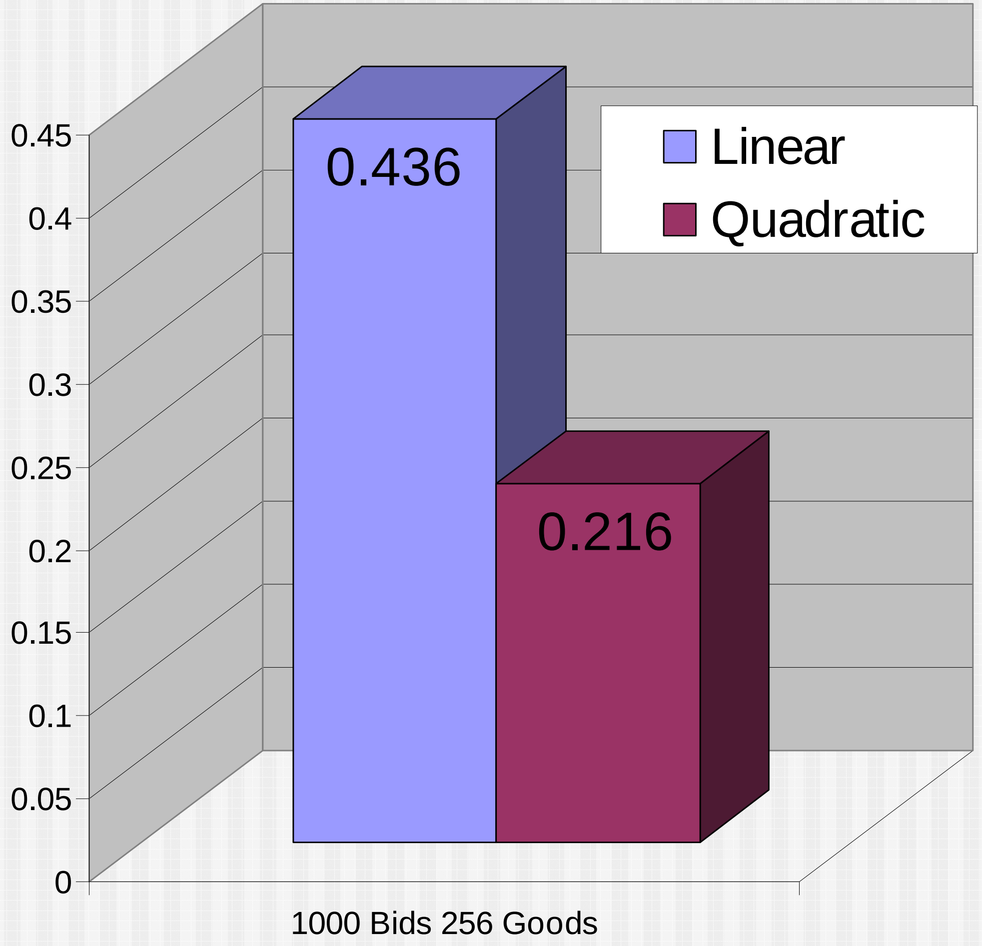
LR: Cost of Omission (subset size 7)



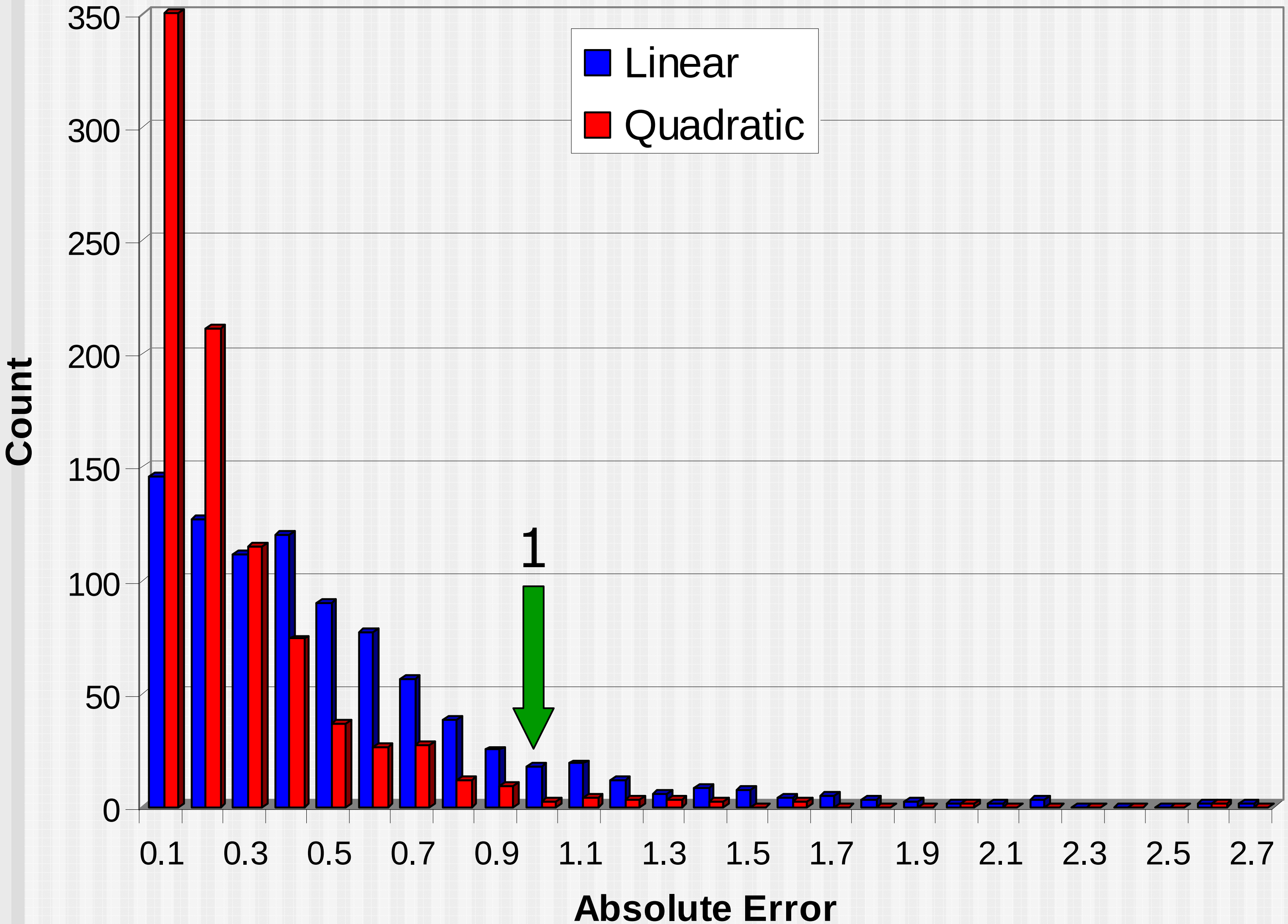
Non-Linear Approaches

- Linear regression doesn't consider interactions between variables; likely to underfit data
- Consider 2nd degree polynomials
- Variables = pairwise products of original features
  - total of 325 variables
  - (cf. clauses/variables)
- More predictability, less interpretability

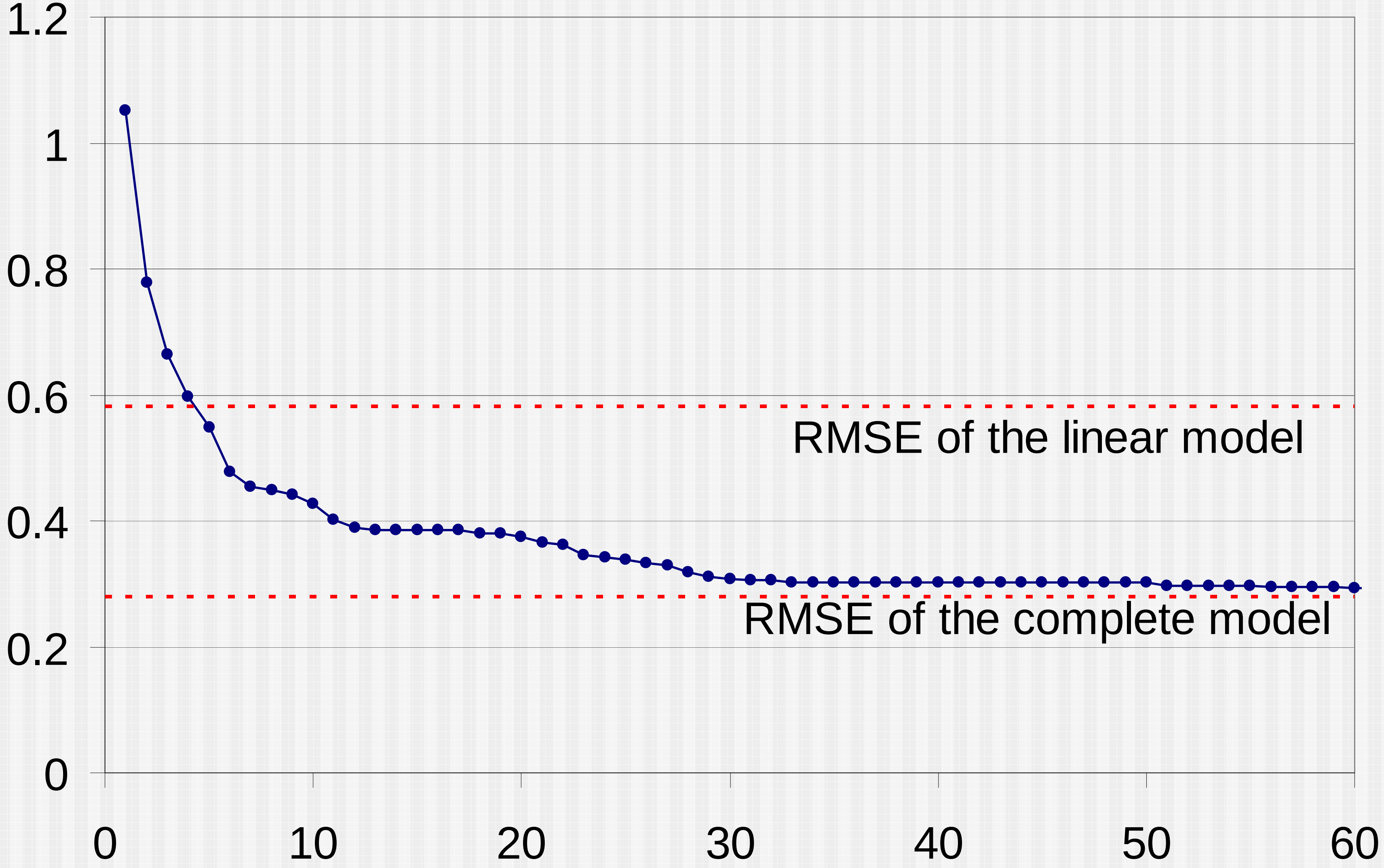
Quadratic vs Linear Regression



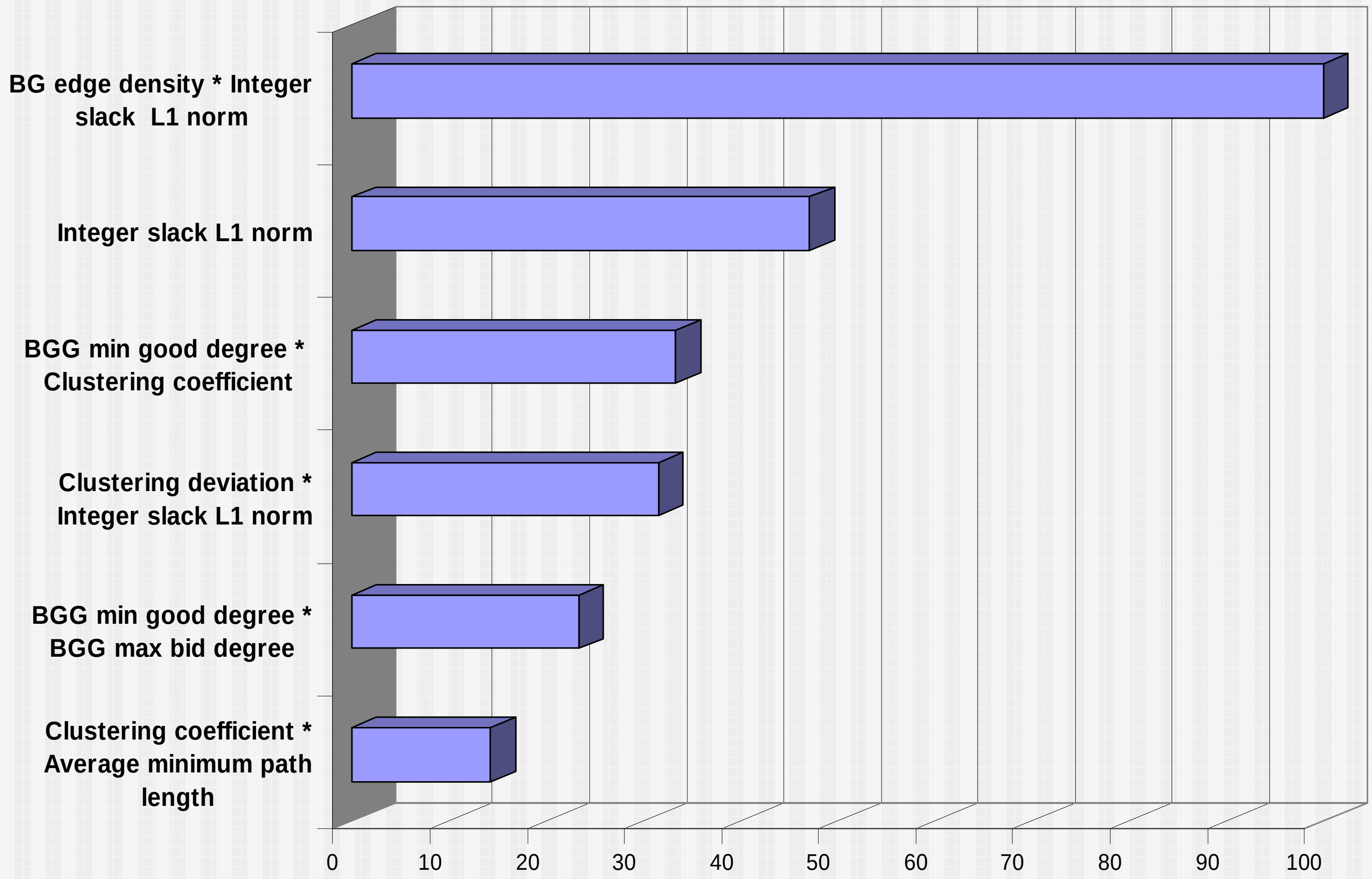
Quadratic vs Linear Regression



QR: RMSE vs. Subset Size



Cost of Omission (subset size 6)



What's Next?

- Constructing algorithm portfolios
  - combine several uncorrelated algorithms
  - good initial results for WDP
- Tuning distributions for hardness
  - releasing new version of CATS

Summary

- Algorithms are predictable
  - Empirical hardness can be studied in a disciplined way
- Once again: Structure matters!
  - Uniform distributions aren’t the best testbeds
  - Constraint graphs are very useful
  - Hypothesis: good heuristics make good features (e.g. LP)
- Our methodology is general and can be applied to other problems!