Emprircal Performace Models

B.A. Rachunok

School of Industrial Engineering
Purdue University

September 10, 2016

History

- Early work by Cheeseman *et al.* (1991) $\rightarrow$Varied paramters and looked at results
- Fink (1998) $\rightarrow$Used regression to predict which of three algorithms would work best
- Leyton-Brown *et al.* (2003) $\rightarrow$Predicted runtimes for several solvers.
- I read the big papers by Leyton-Brown *et al.* and their work will be the focus of this presentation

History

- Early work by Cheeseman *et al.* (1991) $\rightarrow$Varied paramters and looked at results
- Fink (1998) $\rightarrow$Used regression to predict which of three algorithms would work best
- Leyton-Brown *et al.* (2003) $\rightarrow$Predicted runtimes for several solvers.
- I read the big papers by Leyton-Brown *et al.* and their work will be the focus of this presentation

History

- Early work by Cheeseman *et al.* (1991) $\rightarrow$Varied paramters and looked at results
- Fink (1998) $\rightarrow$Used regression to predict which of three algorithms would work best
- Leyton-Brown *et al.* (2003) $\rightarrow$Predicted runtimes for several solvers.
- I read the big papers by Leyton-Brown *et al.* and their work will be the focus of this presentation

History

- Early work by Cheeseman *et al.* (1991) $\rightarrow$Varied paramters and looked at results
- Fink (1998) $\rightarrow$Used regression to predict which of three algorithms would work best
- Leyton-Brown *et al.* (2003) $\rightarrow$Predicted runtimes for several solvers.
- I read the big papers by Leyton-Brown *et al.* and their work will be the focus of this presentation

EPH Motivation

- We already have complexity theory to analyze algorithm performance.
- Why this too?

Consider

- TSP is $O(n!)$ (if solved via brute force)
- However I can solve instances with 5000 points on my grad student laptop?
- Imagine an algorithm with $O(n^{\log(\log(n))})$ performance?
- ... still polynomial

EPH Motivation

- We already have complexity theory to analyze algorithm performance.
- Why this too?

Consider

- TSP is $\mathcal{O}(n!)$ (if solved via brute force)
- However I can solve instances with 5000 points on my gross old laptop?
- Imagine an algorithm with $\mathcal{O}(n^{4e10000})$ performance?
- ... still polynomial

EPH Motivation

- We already have complexity theory to analyze algorithm performance.
- Why this too?

Consider

- TSP is $\mathcal{O}(n!)$ (if solved via brute force)
- However I can solve instances with 5000 points on my gross old laptop?
- Imagine an algorithm with $\mathcal{O}(n^{4e10000})$ performance?
- . . . still polynomial

EPH Motivation

- We already have complexity theory to analyze algorithm performance.
- Why this too?

Consider

- TSP is $\mathcal{O}(n!)$ (if solved via brute force)
- However I can solve instances with 5000 points on my gross old laptop?
- Imagine an algorithm with $\mathcal{O}(n^{4e10000})$ performance?
- . . . still polynomial

EPH Motivation

- We already have complexity theory to analyze algorithm performance.
- Why this too?

Consider

- TSP is $\mathcal{O}(n!)$ (if solved via brute force)
- However I can solve instances with 5000 points on my gross old laptop?
- Imagine an algorithm with $\mathcal{O}(n^{4e10000})$ performance?
- . . . still polynomial

EPH Motivation

- We already have complexity theory to analyze algorithm performance.
- Why this too?

Consider

- TSP is $\mathcal{O}(n!)$ (if solved via brute force)
- However I can solve instances with 5000 points on my gross old laptop?
- Imagine an algorithm with $\mathcal{O}(n^{4e10000})$ performance?
- . . . still polynomial

EPH Motivation

- We already have complexity theory to analyze algorithm performance.
- Why this too?

Consider

- TSP is $\mathcal{O}(n!)$ (if solved via brute force)
- However I can solve instances with 5000 points on my gross old laptop?
- Imagine an algorithm with $\mathcal{O}(n^{4e10000})$ performance?
- . . . still polynomial

Motivation Pt 2

- Knowing how a particular instance of a problem will work gives us some options
- Potentially select the "best" predicted solver for a given instance (more on this)
- Provide some insights into how solvers handle instances (*eg* 4.3 clauses-to-literal ratio)

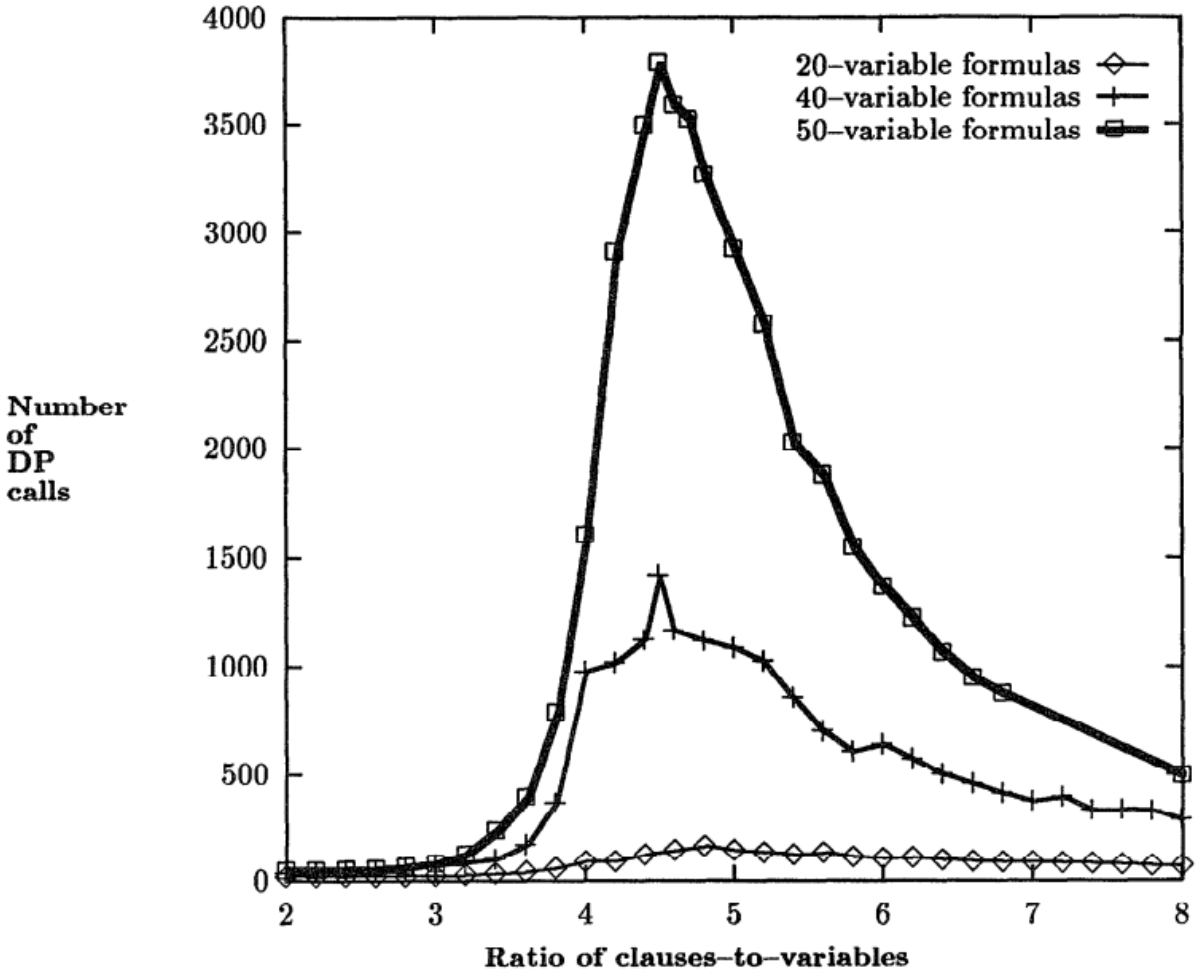
Motivation Pt 2

- Knowing how a particular instance of a problem will work gives us some options
- Potentially select the "best" predicted solver for a given instance (more on this)
- Provide some insights into how solvers handle instances (*eg* 4.3 clauses-to-literal ratio)

Motivation Pt 2

- Knowing how a particular instance of a problem will work gives us some options
- Potentially select the "best" predicted solver for a given instance (more on this)
- Provide some insights into how solvers handle instances (*eg* 4.3 clauses-to-literal ratio)

Clause-to-Literal Ratio



Graph taken from Mitchell, Selman, Levesque 1992

Marker time

To the board

Types of Models

- Ridge Regression
- Neural Networks
- Gaussian Process Regression
- Regression Trees
- INSERT WHATEVER I USED HERE

Types of Models

- Ridge Regression
- Neural Networks
- Gaussian Process Regression
- Regression Trees
- INSERT WHATEVER I USED HERE

Types of Models

- Ridge Regression
- Neural Networks
- Gaussian Process Regression
- Regression Trees
- INSERT WHATEVER I USED HERE

Types of Models

- Ridge Regression
- Neural Networks
- Gaussian Process Regression
- Regression Trees
- INSERT WHATEVER I USED HERE

Types of Models

- Ridge Regression
- Neural Networks
- Gaussian Process Regression
- Regression Trees
- INSERT WHATEVER I USED HERE

Types of Features for MIP

- Number of constraints and variables
- Variable types
- $\leq, \geq$, or $=$ for the RHS
- Mean of objective function coefficients

Types of Features for MIP

- Number of constraints and variables
- Variable types
- $\leq, \geq$, or $=$ for the RHS
- Mean of objective function coefficients

Types of Features for MIP

- Number of constraints and variables
- Variable types
- $\leq, \geq$, or $=$ for the RHS
- Mean of objective function coefficients

Types of Features for MIP

- Number of constraints and variables
- Variable types
- $\leq, \geq$, or $=$ for the RHS
- Mean of objective function coefficients

Types of Features for TSP

- Number of nodes
- Cluster distance
- Area spanned by nodes
- Centroid of points
- Nearest Neighbor path length

Types of Features for TSP

- Number of nodes
- Cluster distance
- Area spanned by nodes
- Centroid of points
- Nearest Neighbor path length

Types of Features for TSP

- Number of nodes
- Cluster distance
- Area spanned by nodes
- Centroid of points
- Nearest Neighbor path length

Types of Features for TSP

- Number of nodes
- Cluster distance
- Area spanned by nodes
- Centroid of points
- Nearest Neighbor path length

Types of Features for TSP

- Number of nodes
- Cluster distance
- Area spanned by nodes
- Centroid of points
- Nearest Neighbor path length

Example Time

- now to the example

Who cares?

- Why do we care about this at all?
- Consider *two* prediction models for *two* solvers
- Consider *three* prediction models for *three* solvers
- ...
- Consider *n* prediction models for *n* solvers

Who cares?

- Why do we care about this at all?
- Consider *two* prediction models for *two* solvers
- Consider *three* prediction models for *three* solvers
- ...
- Consider *n* prediction models for *n* solvers

Who cares?

- Why do we care about this at all?
- Consider *two* prediction models for *two* solvers
- Consider *three* prediction models for *three* solvers
- ...
- Consider n prediction models for n solvers

Who cares?

- Why do we care about this at all?
- Consider *two* prediction models for *two* solvers
- Consider *three* prediction models for *three* solvers
- ...
- Consider n prediction models for n solvers

Who cares?

- Why do we care about this at all?
- Consider *two* prediction models for *two* solvers
- Consider *three* prediction models for *three* solvers
- ...
- Consider n prediction models for n solvers

Who cares cont'd . . .

- If we can *cheaply* compute features and make predictions
- . . . then we can solve our instance with the best algorithm from a *portfolio*
- *eg* SATZILLA

Who cares cont'd . . .

- If we can *cheaply* compute features and make predictions
- . . . then we can solve our instance with the best algorithm from a *portfolio*
- *eg* SATZILLA

Who cares cont'd . . .

- If we can *cheaply* compute features and make predictions
- . . . then we can solve our instance with the best algorithm from a *portfolio*
- *eg* SATZILLA