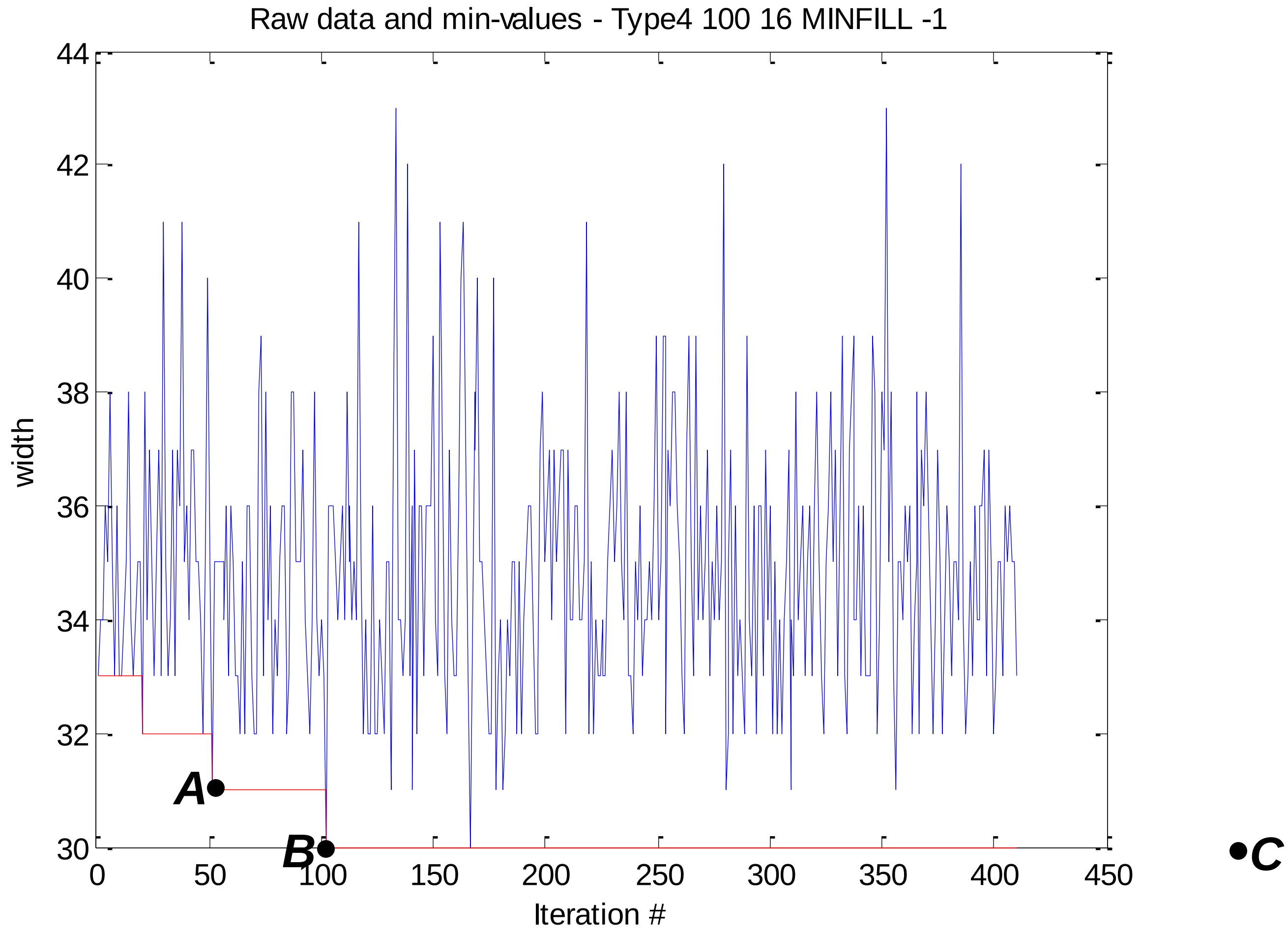
Motivation

- Many inference methods require a tree decomposition (TD)
  - Complexity is exponential in TD width
- Finding minimal width TD is NP-hard
- Leads to difficult dilemma:
  - How much time should I spend searching for a better TD?
- In practice, just run a heuristic for *k* iterations
- Desire a more principled approach

Illustration

- Should we stop at points A, B or some future time C?



Formulation

- Let C(w) be total cost of answering query

$$C(w) = C_{srch}(w) + C_{comp}(w)$$

- Example:
  - Let w_1 and w_2 be two induced widths $w_1 < w_2$
  - Assume $C_{srch}(w_1) > C_{srch}(w_2)$ & $C_{comp}(w_1) < C_{comp}(w_2)$
  - If have taken $C_{srch}(w_2)$ to find ordering, continue if

$$C_{srch}(w_1) + C_{comp}(w_1) < C_{srch}(w_2) + C_{comp}(w_2)$$

- Want to find ordering that minimizes C(w)

Formulation…

- Don’t know $C_{srch}(w)$ and $C_{comp}(w)$
- $C_{srch}(w)$ is a random quantity
  - Depends on stochastic heuristic (e.g. min-fill) and problem instance
- $C_{comp}(w)$ is function of problem instance and elimination order
  - Approximate by instance features and width

Estimating $C_{srch}()$

- Assume we run a heuristic repeatedly
- After each iteration we record:
  1. The induced width of the ordering found, X_i
  2. The time, T_i
- Model iterations as events in a Poisson Process with rate λ
- We classify each event as one of *m* widths
- If independent events, have *m* independent Poisson Processes with rates $\lambda_j=\lambda p_j$

Estimating $C_{srch}()$

- Estimate λ and p_j (j=1…m) given observations $X_1 \ldots X_N$ and $T_1 \ldots T_N$
- Posterior is

$$f(p_1,\ldots,p_m,\lambda \mid X^{(i)}) \propto (\lambda T_i)^{N_i} \exp\{-\lambda T_i\} \prod_{j=1}^{m} p_j^{n_j^i + a_j - 1}$$

- with posterior estimates

$$\hat{\lambda} = E\left[\lambda \mid X^{(i)}\right] = \frac{N^i}{T_i} \qquad \hat{p}_j = E\left[p_j \mid X^{(i)}\right] = \frac{n_j^i + a_j}{N^i + a_0}$$

- Let Z_j be arrival time of first type *j* event

$$E\left[Z_j \mid X^{(i)}\right] = \frac{1}{\hat{\lambda} \cdot \hat{p}_j}$$

Estimating $C_{srch}()$

- Don’t observe X_i but Y_i where $Y_i=\min(X_1 \ldots X_i)$
- Must aggregate when new Y_i is observed
- Changes posterior computation slightly

$$\hat{p}_j = E\left[p_j \mid X^{(i)}\right] = \frac{\tilde{n}_j^i + a_j}{N^i + a_0}$$

Estimating $C_{comp}()$

- Need fine analysis of computation time
- Model time as

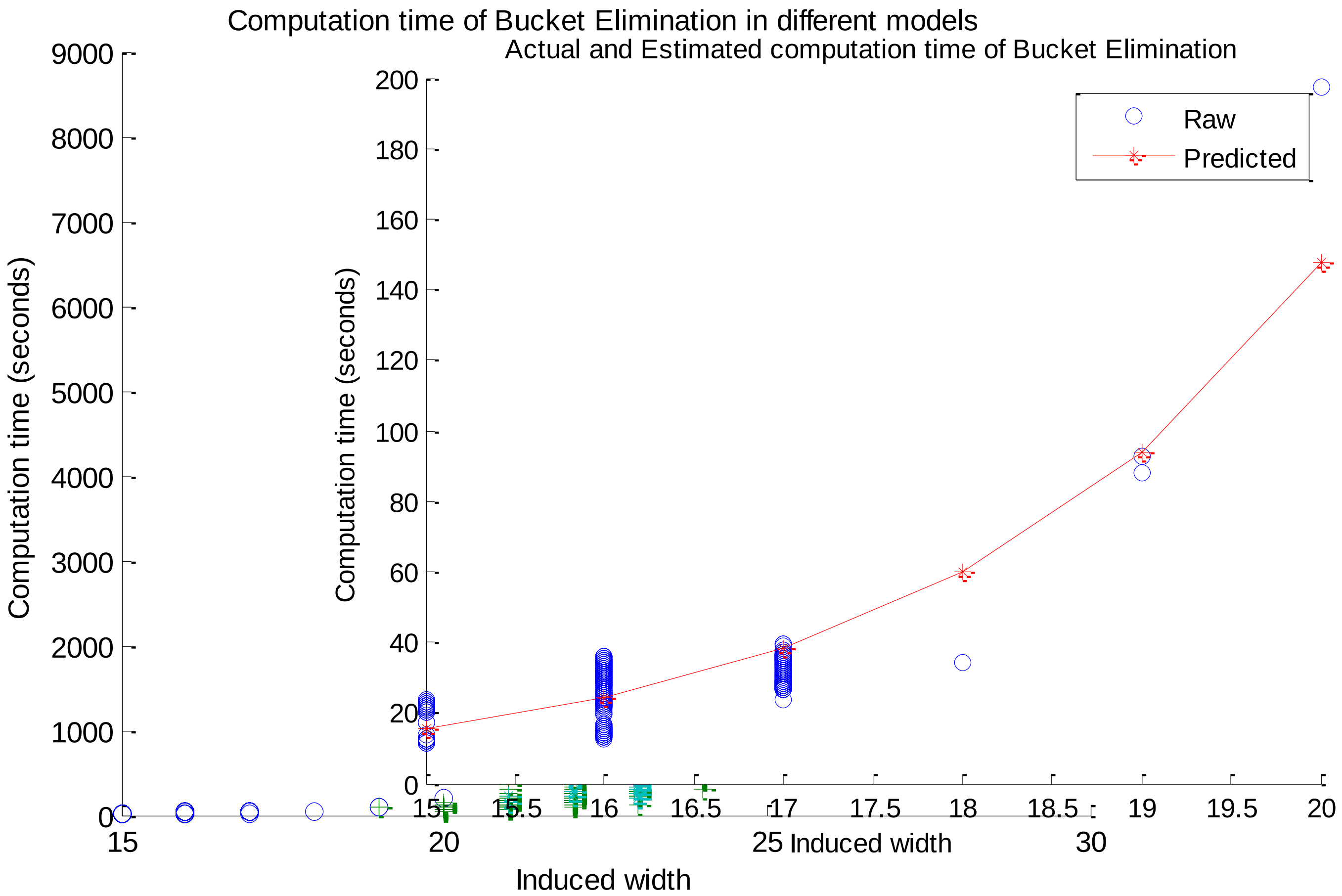
$$t_{comp}(w) = \exp\left\{\beta^T s\right\}$$

where **s** is a vector of features and β are parameters

- Fit using standard regression analysis
- Ex. Design Matrix

$$\begin{bmatrix} 1 & nv_1 & nf_1 & d_1 & s_1 & t_1 & w_1 \\ 1 & nv_2 & nf_2 & d_2 & s_2 & t_2 & w_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & nv_M & nf_M & d_M & s_M & t_M & w_M \end{bmatrix}$$

Estimating $C_{comp}()$



Stopping Criteria

- Current cost is: $C_i = T_i + t_{comp}(Y_i)$
- Predicted cost of width *j* (denoted W(j)) is:

$$E\left[C(W(j)) \mid Y^{(i)}\right] = E\left[Z_j \mid Y^{(i)}\right] + t_{comp}(W(j))$$

- Let j^i be index of largest *j* such that $W(j) < Y_i$

  Stop when

$$E\left[C(W(j)) \mid Y^{(i)}\right] \geq C_i$$

  for all $j < j^i$
- Let i_{pred} denote the predicted termination index

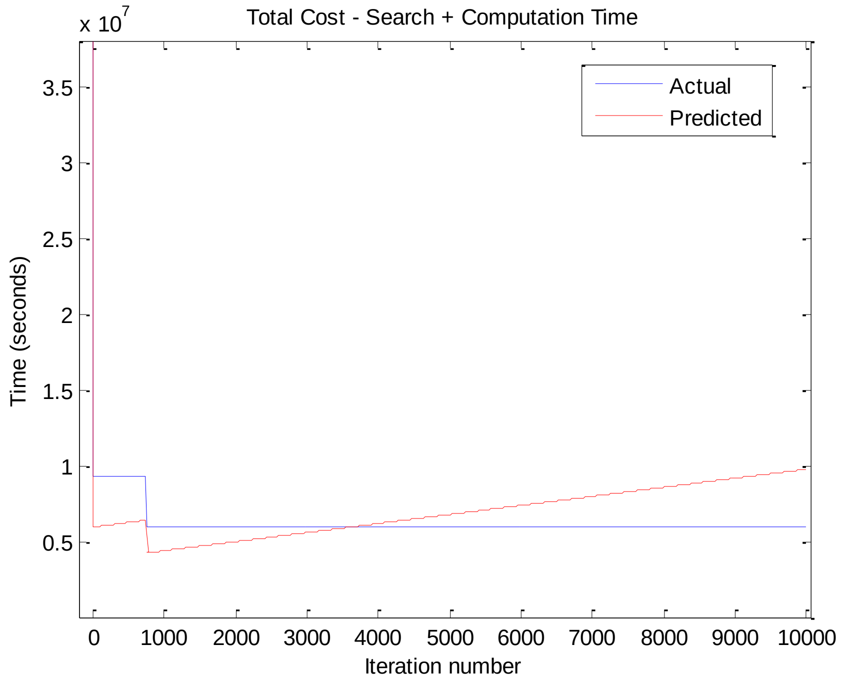
Experiments

- Evaluation metrics:

$$Ierr = \left| i_{pred} - i_{best} \right| \qquad Terr = \left| T_{i_{pred}} - T_{i_{best}} \right|$$

where i_{best} is the index where C_i is a minimum

- Ex:



Experiments

- Ierr:

	1e-5	1e-4	1e-3
pedigree	6458 (4262.8)	4792.9 (4548.2)	2508.5 (3806.9)
largeFam3	3443.5 (3570.2)	4883.8 (3493.3)	6355.4 (3374.8)
largeFam4	2991.5 (3125.2)	3016.1 (3145.7)	4036.8 (3377.7)
largeFam5	3770.8 (3677.2)	2648.6 (3111.5)	3234.6 (3350.3)

- Terr

	1e-5	1e-4	1e-3
pedigree	93.69 (74.22)	73.55 (77.23)	36.64 (52.12)
largeFam3	868.0 (1343.0)	1023.2 (1292.4)	1148.9 (1238.1)
largeFam4	572.45 (892.9)	668.29 (1009.9)	817.1 (1023.9)
largeFam5	624.0 (988.9)	586.5 (1029.9)	730.2 (1135.4)

Alternate estimator of t_{comp}

- Compute 'number of operations' per ordering
- For BE, eliminating a variable in bucket i involves $|S_i|$ x $|f_i|$ operations
  - where f_i is the set of functions in bucket i
    S_i is the the union of the scopes of f_i
- numOps is the sum across all buckets
- Can easily compute numOps given a heuristic
- Refer to this as t_{comp}(numOps)

BREN:ICS INFORMATION AND COMPUTER SCIENCES

UNIVERSITY of CALIFORNIA IRVINE

$t_{comp}(w)$ versus $t_{comp}(numOps)$