Comparing Planners: Beyond Coverage Tables

Caleb Hill[1], Stephen Wissow[2], Wheeler Ruml[2]

[1]Department of Mathematics and Statistics, [2]Department of Computer Science
University of New Hampshire, USA

Thanks to the NSF-BSF program for support via NSF grant 2008594.

Contributions

1. non-trivial problem that deserves attention—and standard tools don't apply
2. some early-stage ideas (collaborators welcome!)

The Problem

coverage table:

	planner A	planner B	planner C
tricky domain	**3**	**3**	**3**
normalized (%)	**100**	**100**	**100**

#1: requires failures

CPU times

planner A: 0.1, 0.2, 0.3

planner B: 1000, 2000, 3000

planner C: 0.01, 0.05, 0.3

B much slower:

#2: insensitive to magnitudes

C scales worst:

#3: insensitive to scaling

---

	planner A	planner B
tricky domain	1000	**1001**

reproducible with new instances?

#4: no measure of certainty

Sign Test (Arbuthnot 1710)

- non-parametric: pair of CPU times $\rightarrow$ coin flip
- measure bias of coin: which planner faster more often

CPU times

planner A: 1, 2, 3

planner B: 1.01, 2.01, 3.01

planner B′: 1000, 2000, 3000

A vs B	A vs. B′
$1 < 1.01$	$1 < 1000$
$2 < 2.01$	$2 < 2000$
$3 < 3.01$	$3 < 3000$
P(A<B) $= 1$	P(A<B′) $= 1$

Sign Test result

"A is better than B and B′"

#1: insensitive to magnitudes

---

planner A: 0.9, 1.9, 3000

planner B: 1.1, 2.1, 3.1

A vs B

$0.9 < 1.1$

$1.9 < 2.1$

$3000 > 3.1$

P(A<B) $= 2/3$

"A is better than B"

but B obviously scales better

#2: insensitive to scaling

Wilcoxon Rank-Sum (Deuchler 1914)

- non-parametric: rank CPU times, sum each algorithm's ranks
- #1: assumes CPU time distributions have same shape and spread

CPU times

planner A: 1, 2, 3

planner B: 1.1, 2.1, 3.1

planner B′: 1.1, 2.9, 3000

A vs. B	A vs. B'
1. 1	1. 1
2. 1.1	2. 1.1
3. 2	3. 2
4. 2.1	4. 2.9
5. 3	5. 3
6. 3.1	6. 3000
A = 9	A = 9
B = 12	B' = 12

Wilcoxon Rank-Sum result

"A is better than B and B′"

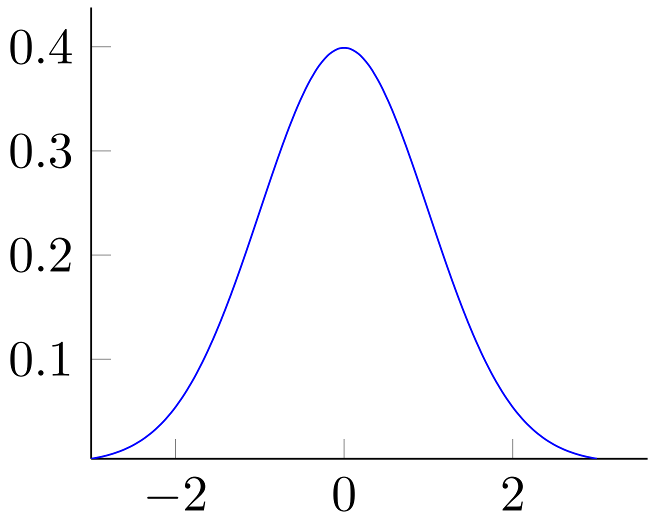
#2: insensitive to scaling

#3: insensitive to magnitudes

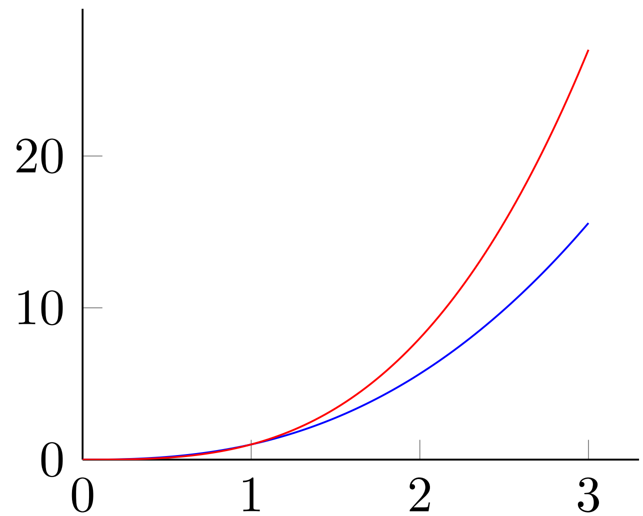
(#2 & #3 shared with sign test)

Paired t-test (Gosset 1908)

- probability that mean difference is not zero
- #1: assumes paired runtime differences normally distributed
- but we expect differences to continue to grow



$\mathcal{N}(0,1)$



two planners' running times

even with Gaussian noise we don't expect central tendency

McNemar's Test (McNemar 1947)

- contingency table = paired coverage table
- has been used in the planning literature

CPU times

planner A: 1, 2, 3

planner B: 1000, 2000, 3000

	B+	B-
A+	3	**0**
A-	**0**	0

McNemar's Test result

"A = B"

- #1: insensitive to scaling
- #2: requires failures
- just codifies reasoning behind coverage table

Desiderata

the community needs a metric that

- does not require failures
- is sensitive to scaling/magnitudes
- provides a measure of its certainty
- has assumptions that align with planning
- (see paper for more)

New Approaches

these tests fail:

- Coverage Tables
- Sign Test
- Wilcoxon Rank-Sum Test
- Paired t-test
- McNemar's Test

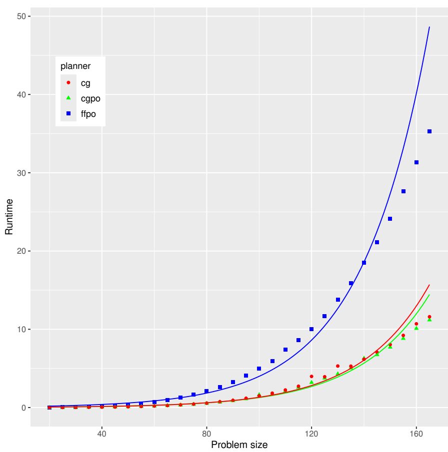
some works in progress:

- Bootstrapped Exponential Estimates (BEE)
- a Bayesian approach

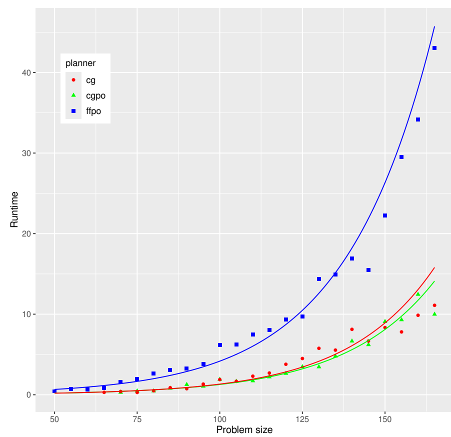
Bootstrapped Exponential Estimates (BEE)

fit $m2^{kn}$ to planner runtime data, use residuals for bootstrapping

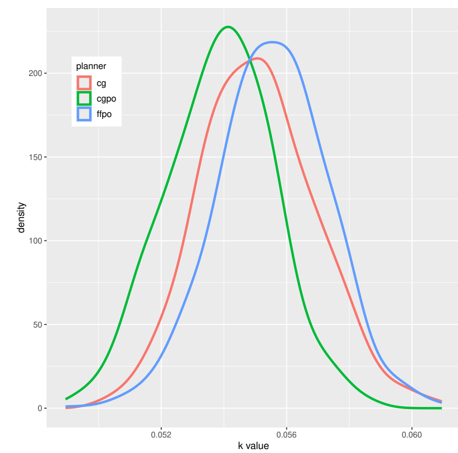
- sensitive to scaling/magnitudes
- provides a measure of its certainty
- assumptions align with planning
- (more pros in paper)



fit original data



hallucinate bootstrapping data



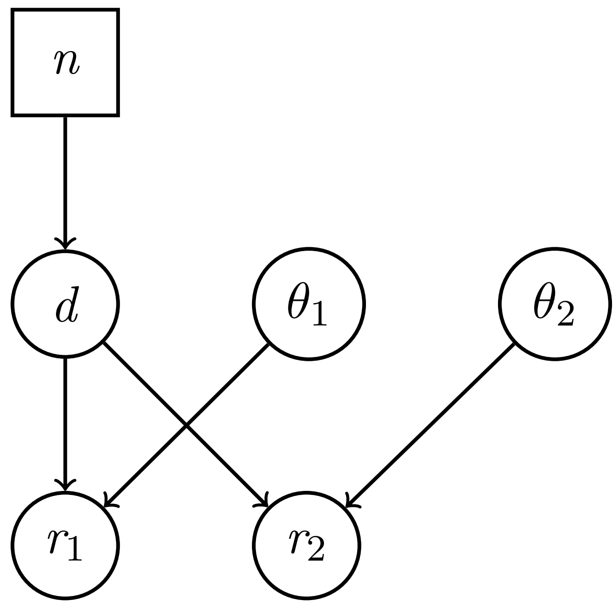
compare distributions of k

BEE Algorithm Sketch

1. fit $m2^{kn}$ to one planner's running time data
2. estimate μ, σ^2 of residuals in log-running time space
3. do r times:
4. hallucinate new residuals from μ, σ^2
5. add to original fit to hallucinate new data
6. fit hallucinated data, record m, k
7. compare planners' histograms of $k \rightarrow P(k_i < k_j)$

Another Approach: a Bayesian Model

- n: problem size (generator parameter)
- d: problem difficulty
- $\theta = \begin{bmatrix} m \\ k \end{bmatrix}$: planner parameters
- r: one CPU running time measurement



Conclusion

- planner comparison is central to our field
- running time data are rich
- coverage tables are not expressive enough

let's develop appropriate measures

Future Work

"which algorithm is better" $\rightarrow$ "which algorithm is better where/when/why"