Comparing Planners
Beyond Coverage Tables

Caleb Hill

Department of Mathematics and Statistics

University of New Hampshire

Outline

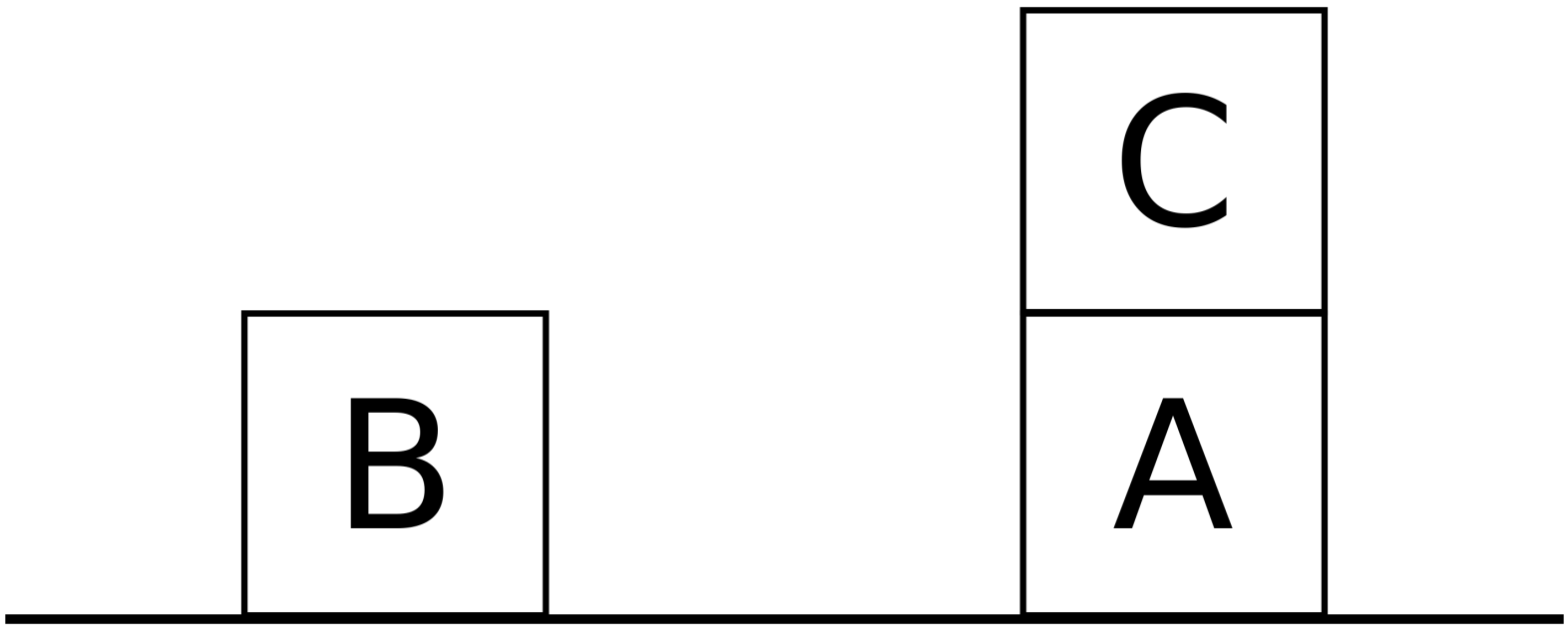
Acknowledgements

WIPC Collaborators:

- Prof Wheeler Ruml
  - Minor advisor
- Steve Wissow
  - WIPC 2024 talk

Advising/assisting:

- Prof Pei Geng
  - Committee member
- Prof Cain Edie-Michell
  - Committee member
- Prof Marek Petrik



Outline

Motivation

What is planning?

- Important subfield of AI: logistics, scheduling, agents
- A model for decision making problems
- Not known to be efficient

Algorithms map problem instances to runtimes or timeouts

This begs the question...

Problem

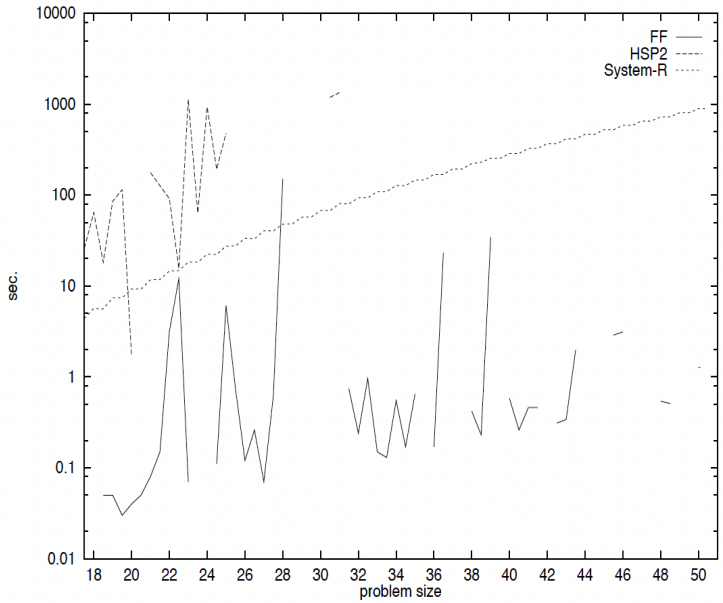
Given two planning algorithms, which one is better?

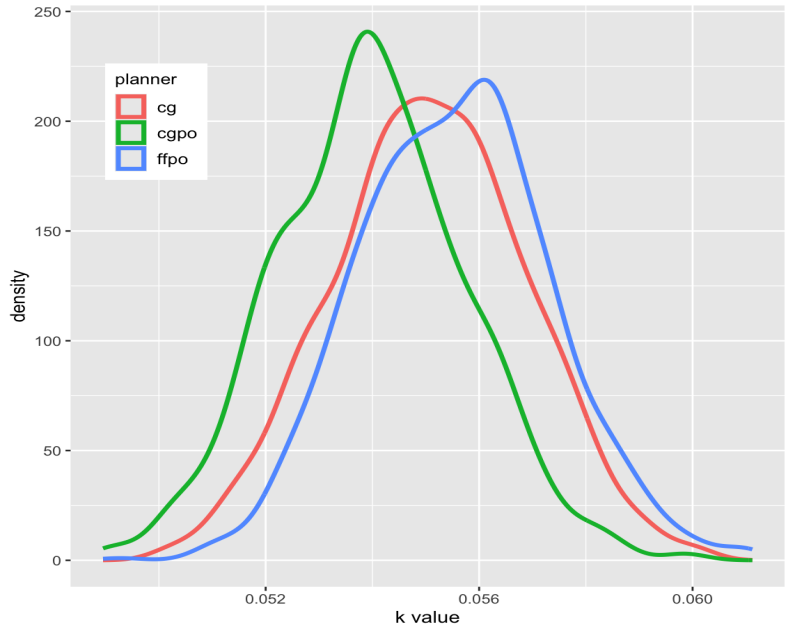
Contributions

In this project we have:

- Formalized the notion of problem **difficulty** and **size**
- Estimated and leveraged this new difficulty
- Clarified implicit assumptions behind common practice

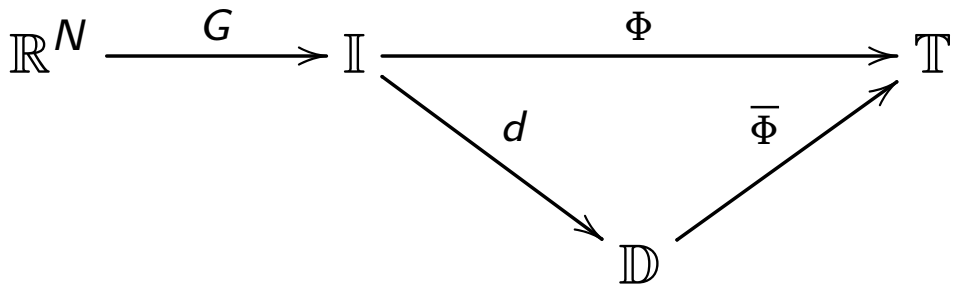
In this talk we will build a bridge between two types of charts:





Central Thesis

The bridge rests on an assumption which manifests as the diagram



and can be (vaguely) stated as

Hypothesis

There exists a function d such that for any planner Φ there is a $\overline{\Phi}$ that makes the above diagram approximately commute.

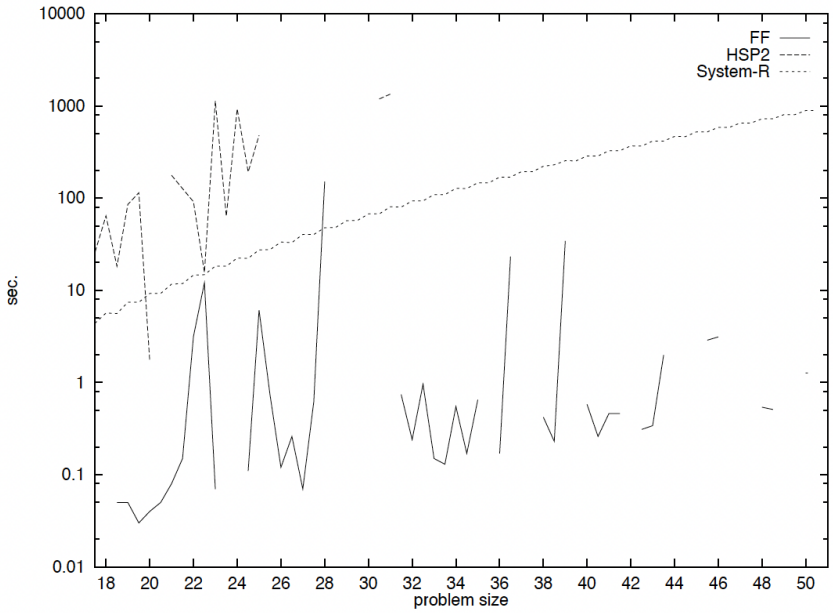
Tasks

1. *Find a concrete instance of a function d*
2. *Use d to compare different planners Φ_A and Φ_B via $\overline{\Phi}_A$ and $\overline{\Phi}_B$*

Outline

Runtime Plots and Coverage Tables

- Runtime charts
- Coverage tables
- Virtually no statistical tests

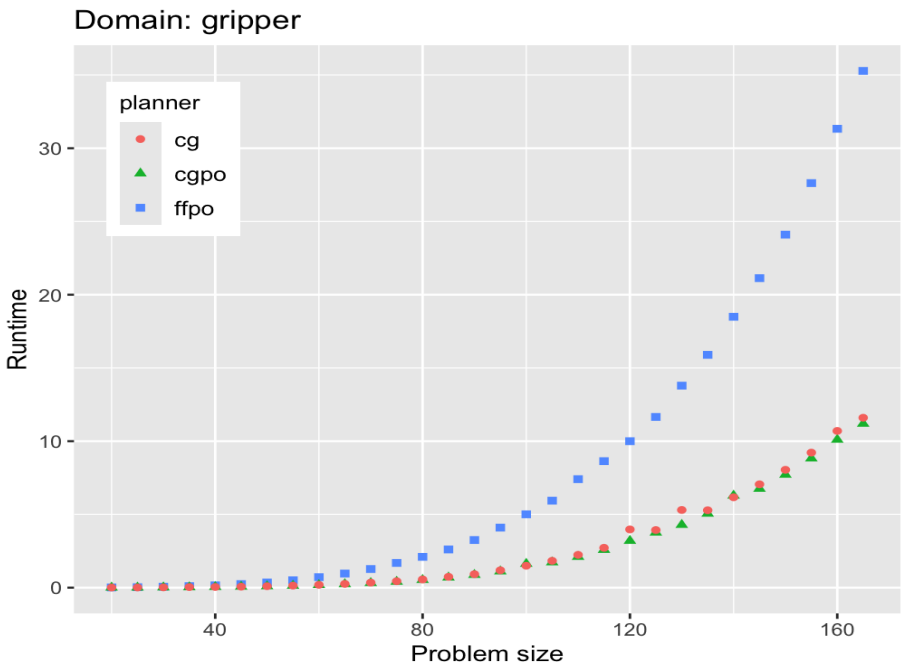


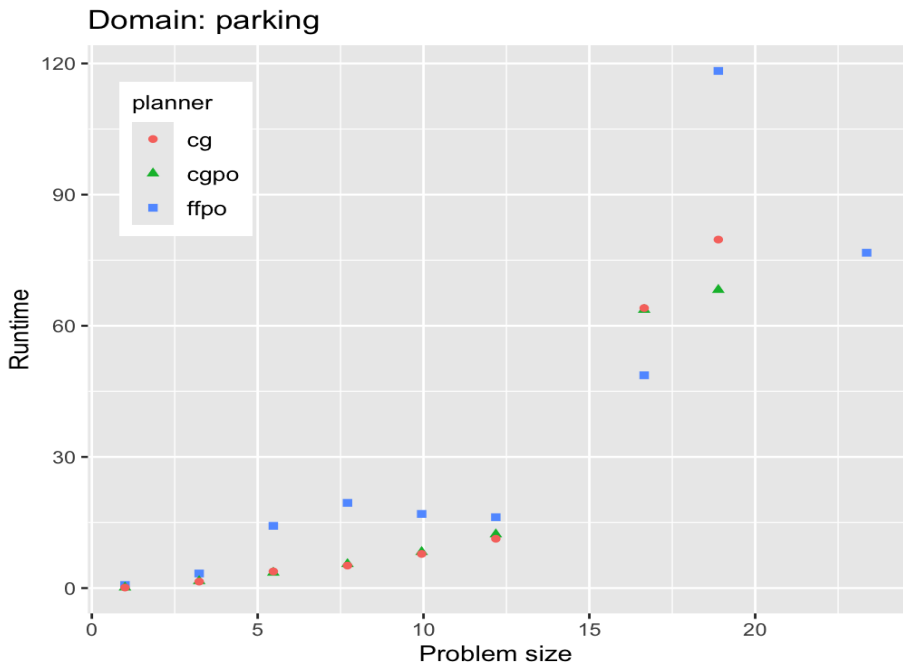
Coverage	WA*	EES	DPS	DXES	CDXES↻	RR-d	RR-DPS	RR-DXES
Airport (50)	31	30	**33**	29	32	**33**	32	**33**
Blocks (35)	33	31	34	28	31	34	32	**35**
DataNetwork (20)	14	14	14	14	14	**16**	13	**16**
Depot (22)	9	10	10	8	9	**11**	8	**11**
DriverLog (20)	**15**	**15**	**15**	14	**15**	**15**	**15**	**15**
Elevators (30)	23	21	23	21	22	**28**	23	**28**
Floortile (40)	22	17	**23**	12	16	21	21	22
Freecell (80)	15	16	15	15	17	17	15	**19**
GED (20)	15	15	**19**	15	15	15	**19**	15
Grid (5)	2	2	2	2	2	**3**	2	**3**
Gripper (20)	**20**	15	**20**	8	9	9	16	18
Hiking (20)	10	10	10	9	10	**12**	10	11
Logistics (63)	**48**	42	**48**	43	42	41	45	44
Mprime (35)	22	22	**23**	22	21	22	**23**	**23**
Mystery (19)	**17**	**17**	**17**	15	15	**17**	**17**	**17**
Nomystery (20)	**20**	**20**	**20**	14	17	**20**	**20**	**20**
Openstacks (80)	36	34	36	33	33	37	31	**38**
OrgSynth-split (20)	**16**	**16**	**16**	15	**16**	**16**	**16**	**16**
Parcprinter (30)	**30**	**30**	**30**	**30**	**30**	**30**	28	**30**
Parking (40)	18	18	20	7	13	22	17	**23**
Pathways (30)	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**
Pegsol (36)	35	33	35	33	34	**36**	35	**36**
Pipes-notank (50)	23	24	23	17	24	26	24	**27**
Pipes-tank (50)	13	13	13	14	14	17	13	**18**
PNetAlignment (20)	11	11	11	8	9	**13**	10	**13**
PSR (50)	49	49	**50**	49	**50**	**50**	**50**	**50**
Rovers (40)	14	12	13	11	15	14	14	**18**
Satellite (36)	**14**	13	**14**	11	13	13	**14**	12
Scanalyzer (28)	17	12	**20**	12	15	16	17	18
Snake (20)	7	7	7	6	**8**	7	7	7
Sokoban (30)	**29**	28	**29**	28	28	**29**	**29**	**29**
Spider (20)	**12**	11	**12**	11	11	**12**	11	**12**
Storage (30)	16	17	16	15	17	**18**	16	17
Termes (20)	7	**11**	9	6	6	10	7	10
Tetris (17)	7	7	**8**	6	6	**8**	7	**8**
Tidybot (30)	21	21	21	19	22	23	20	**26**
TPP (30)	7	8	8	8	**9**	**9**	7	8
Transport (59)	18	19	18	18	20	22	18	**23**
Trucks (30)	**21**	18	18	14	18	20	**21**	19
VisitAll (33)	21	20	21	18	21	**23**	21	**23**
Woodworking (30)	26	27	26	26	28	29	27	**30**
Zenotravel (20)	14	14	**15**	13	13	14	14	14
Others (274)	**191**	**191**	**191**	**191**	**191**	**191**	**191**	**191**
Sum (1652)	995	967	1012	894	957	1025	982	**1052**
Normalized (%)	58.7	57.0	60.0	51.5	55.6	60.7	57.9	**62.5**
Expansions	569	558	472	734	511	383	665	**371**
Search time (s)	0.65	0.91	**0.55**	1.09	0.83	0.65	1.05	0.65

What Do We Want in a Test?

Some key desiderata are:

- Quantify certainty of result





- Take runtimes as input
- Require no more than one run per problem size

Why Not Use Established Tests?

1. Sign test
   - H_0: median runtime difference is zero
   - Insensitive to magnitudes
2. Wilcoxon rank-sum test
   - Ranking differences are due to chance
   - Insensitive to magnitudes

Established Tests II

3. Paired t-test
   - H_0: mean runtime difference is zero
   - Parametric, probably not applicable

4. McNemar's test
   - Precedent in the literature [1]
   - Chances of not solving a problem are equal
   - Paired coverage only
   - Does not account for structure
   - Requires censored data

Example

Paired coverage:

	ffpo+	*ffpo-*
cg+	*29*	*10*
cg-	*7*	*74*

These tests fit the above desiderata, but **we want more!**

Coverage Tables and Beyond

More desiderata:

- Incorporate and articulate common assumptions
- Exploit *censored* runtimes, not coverage data
- Be sensitive to magnitudes of differences

At their best, the use of coverage tables implicitly takes into account the increasing difficulty of problems as **size** increases.

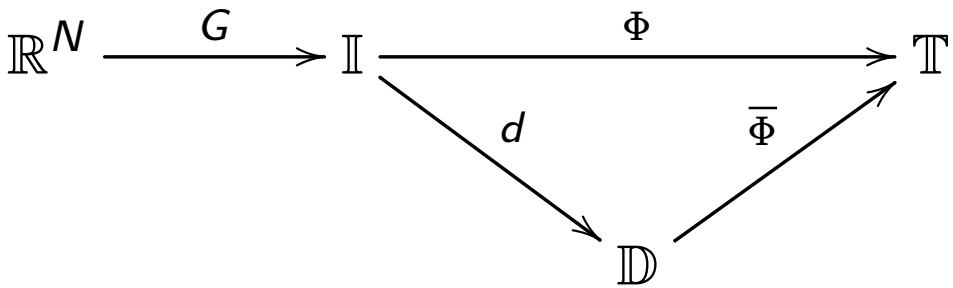
We would like to take that into account **explicitly**

Outline

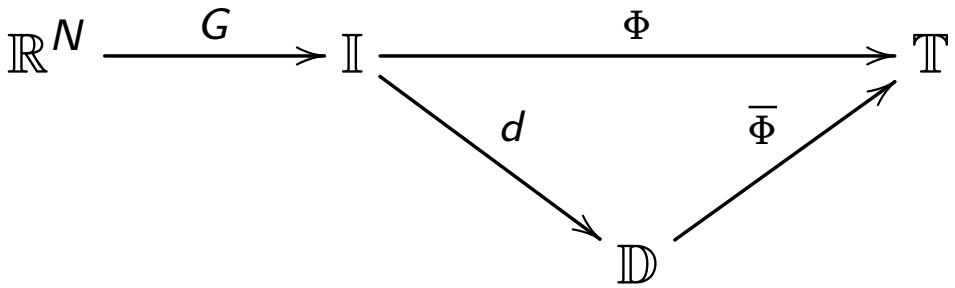
Modeling Planner Runtimes

Accounting for problem structure

- Problems have **size** and **difficulty** associated with them
  - One-dimensional parameter subspace
  - Runtime is exponential in certain parameters [2]
- Runtime is monotone in difficulty
- Difficulty is a good proxy for runtime



Modeling Planner Runtimes II



- $\mathbb{I}$: set of problem instances
- $\mathbb{T} = (0, \infty)$: set of runtimes
- $\mathbb{D} = (0, \infty)$: set of difficulties
- $d(x) \approx d(y) \implies \Phi(x) \approx \Phi(y)$
- $\overline{\Phi} : \mathbb{D} \to \mathbb{T}$ such that $\Phi(x) \approx \overline{\Phi}(d(x))$

If we knew Φ exactly, we would know everything. Instead, we approximate Φ by way of d and $\overline{\Phi}$.

Clarifying Assumptions

Assumption (Difficulty is exponential)

A difficulty function d exists, and is approximately exponential in the generator parameter:

$$d \circ G(n) \approx a2^{kn}$$

*where a and k (**scale parameter**) are positive.*

Usually the base is assumed to lie in $(1.5, 2)$ according to [2].

Assumption (Runtime is linear)

Pure runtime is approximately linear in problem difficulty:

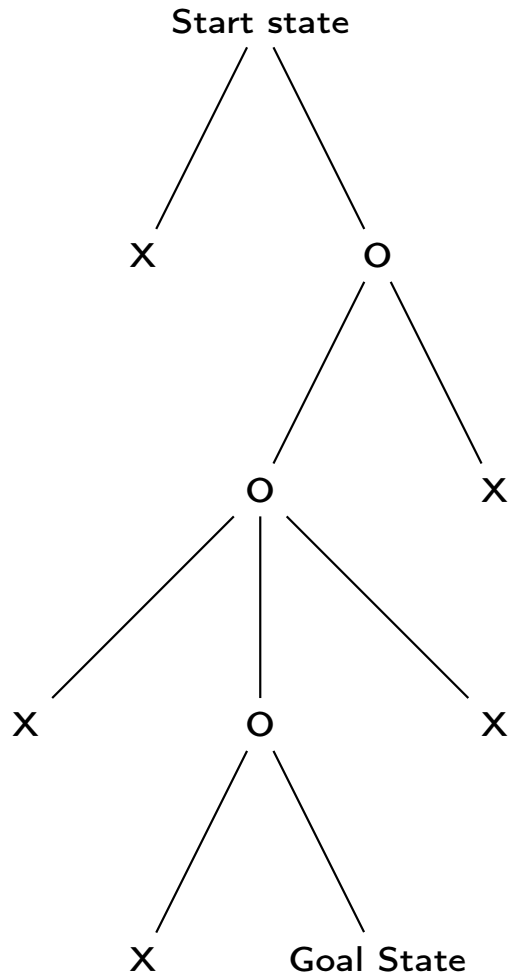
$$\overline{\Phi}(d) \approx md$$

*where m (**slope parameter**) is positive.*

Algorithmic Considerations

Exponential difficulty conforms to our knowledge of how state space (graph) search is performed

- Frontier expansion
- Effective branching number



Outline

Tool 1: Regression

- Assumptions 1 and 2 together say $\overline{\Phi} \circ d \circ G(n) \approx m \cdot 2^{kn}$
- Given planners' runtime data, we fit an exponential to each
- Baseline, **noiseless** runtime prediction

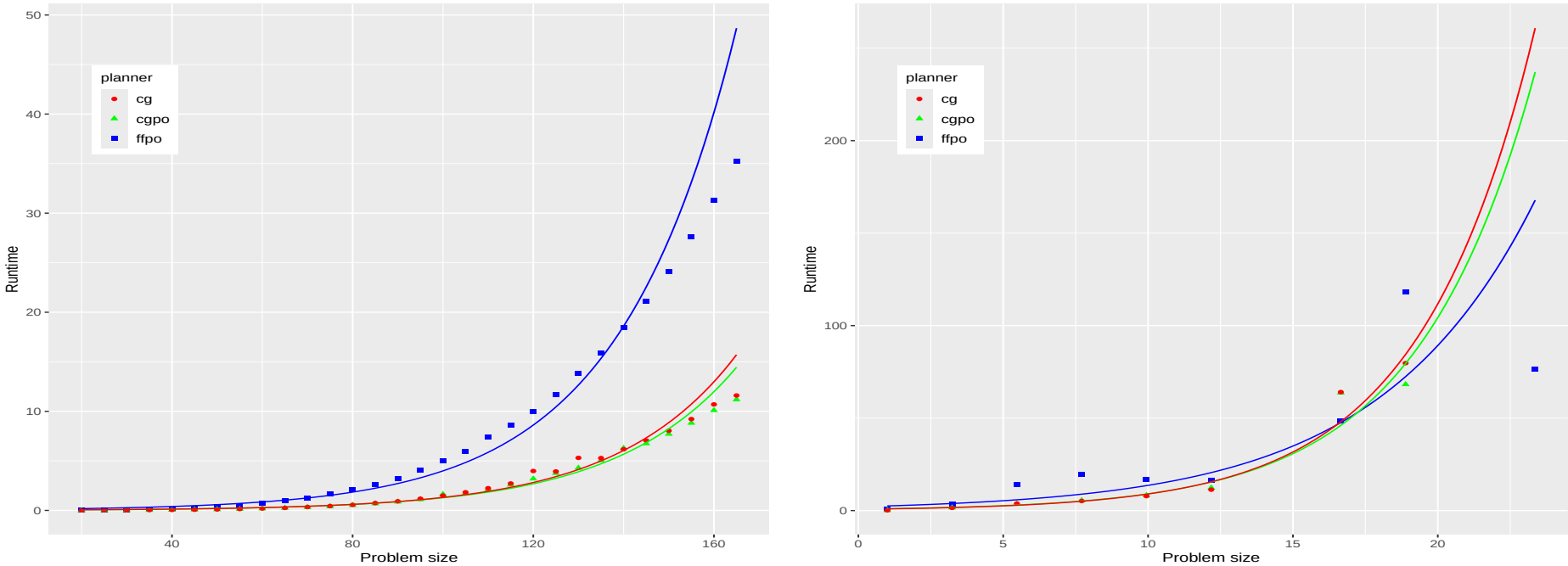


Figure: Runtimes for gripper (left) and parking (right) with exponential fits

Now we can talk about which is better, but **with no measure of certainty**

Tool 2: Bootstrap

- Hallucinate runtime data
- Fit an exponential to hallucination

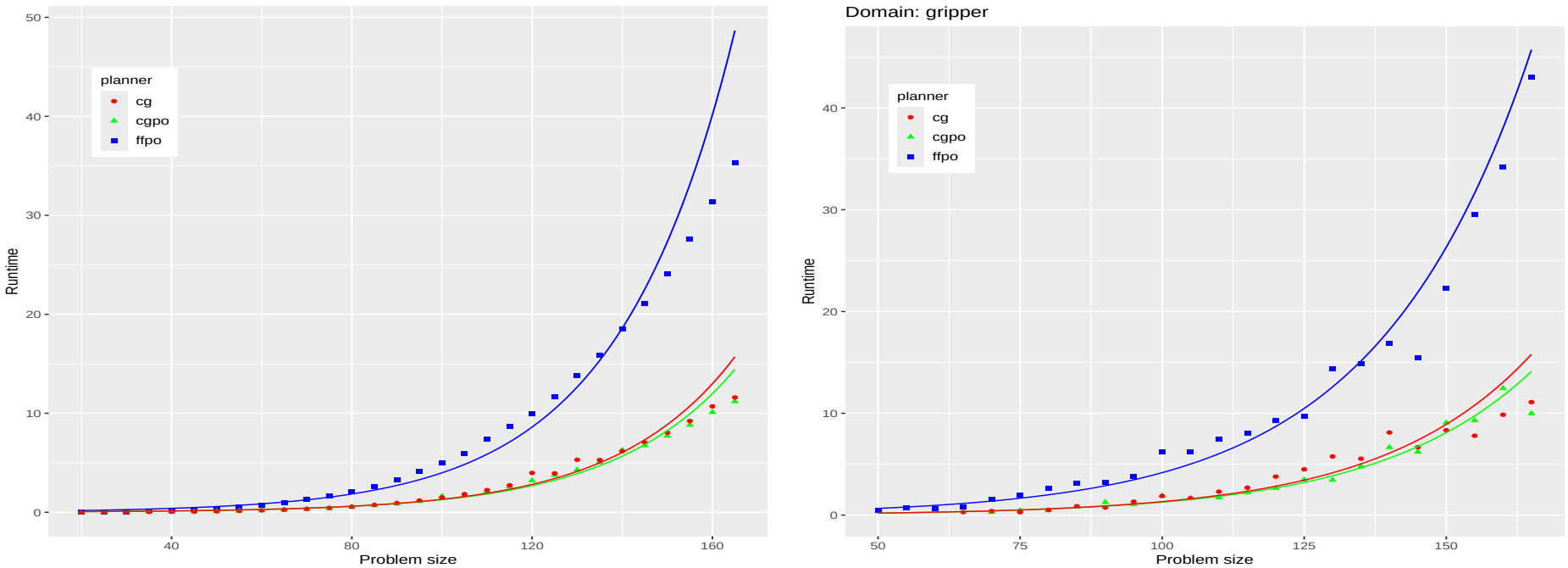
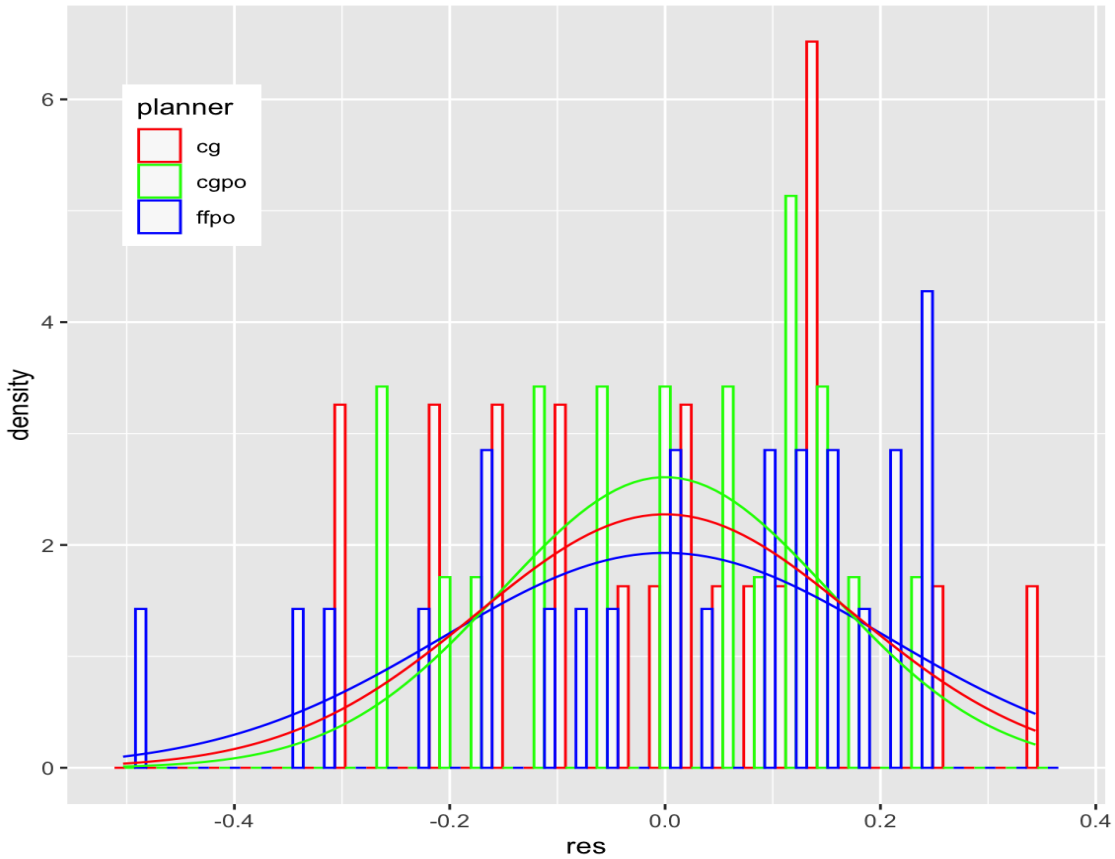


Figure: Actual runtimes (left) and hallucinated runtimes (right)

Over a batch of many hallucinations, we can extract distributions of exponential parameters.

Tool 3: Residual Estimates

- Noise represents unmodelled phenomena
- Unmodelled but regular



Sampling this noise distribution is how we **induce hallucination**.

Bootstrapped Exponential Estimates

Our method works as follows:

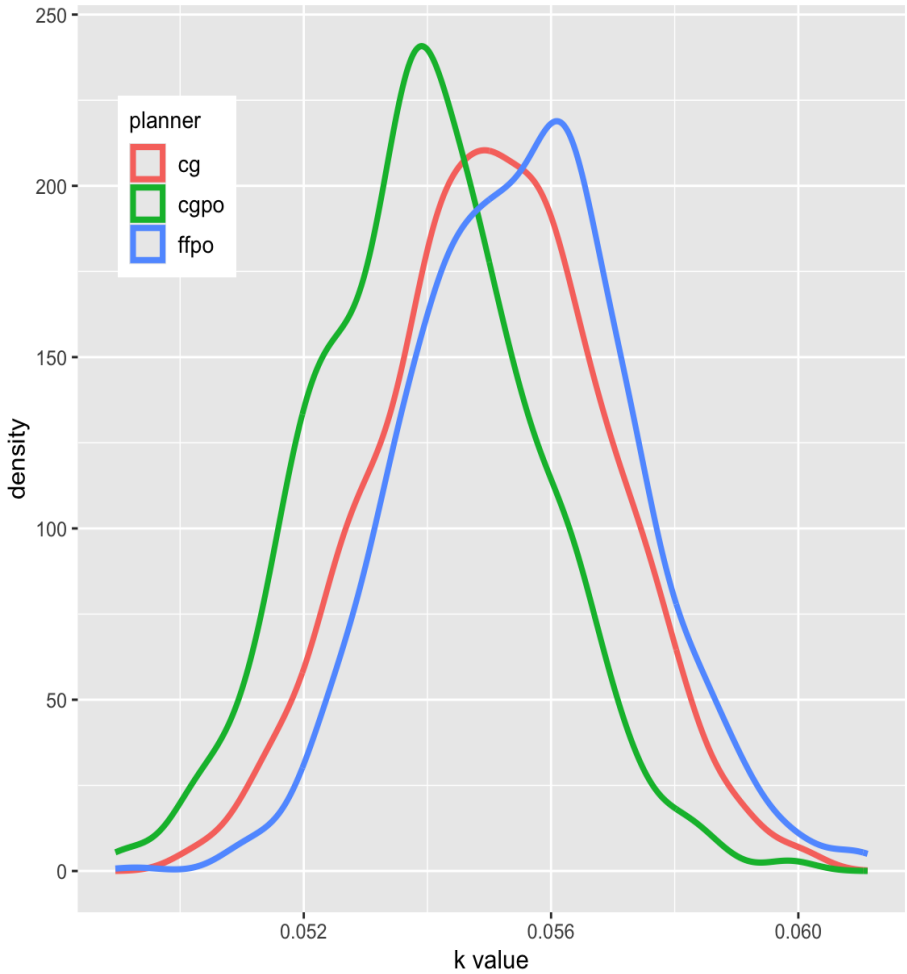
1. let there be data for planners 1,2
2. fit exponential to get $\hat{m}_i$ and $\hat{k}_i$
3. estimate the residuals of the exponential fits as $\varepsilon_i \sim N(0, \sigma_i^2)$
4. do r times:
5. simulate data for each planner using $\hat{m}_i$, $\hat{k}_i$, and ε_i
6. fit exponentials and extract estimates for m_i and k_i
7. form distributions for m_i and k_i
8. return $P(k_1 < k_2)$

Outline

Example Results

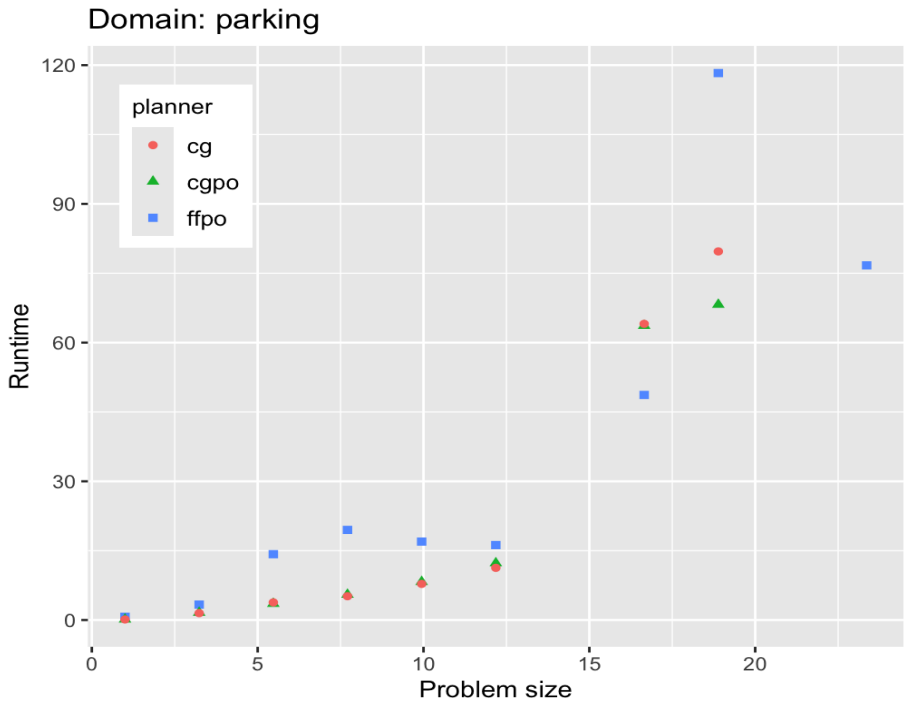
Given parameter distributions, compute $P(k_{row} < k_{col})$

	ffpo	cgpo	cg
ffpo	0.499	0.107	0.121
cgpo	0.892	0.4995	0.495
cg	0.878	0.504	0.499



Sensitivity

- How few data points do different tests need to detect a difference in planner?
- Run BEE on longer and longer prefixes of runtime data



Outline

Possible Extensions

- Richer model
  - Fit base of exponential (effective branching)
  - Simultaneous fits
  - Error: function of size?
  - More expressive model: exponential by polynomial?
- More dimensions
  - Multiple *linear parameters*
- Bayesian approach
  - Greater flexibility
  - Incorporate runtime data from all planners

Conclusion

In this project, we have contributed by:

- Formalizing problem **difficulty** and **size**
- Clarifying common **implicit assumptions**
- **Hypothesizing a model** of planner performance
- Exploring **two instantiations** of the model
- Using these instantiations to propose a **comparison method**

References

[1] Jendrik Seipp et al. "Learning Portfolios of Automatically Tuned Planners". In: *Proceedings of the International Conference on Automated Planning and Scheduling* 22.1 (2012), pp. 368–372. doi: 10.1609/icaps.v22i1.13538. url: https://ojs.aaai.org/index.php/ICAPS/article/view/13538.

[2] Álvaro Torralba, Jendrik Seipp, and Silvan Sievers. "Automatic Instance Generation for Classical Planning". In: 31.1 (2021), pp. 376–384.