Implementing Distribution Testing Algorithms

Aoxi Li, Wanting Mao, Juyoung Park, John Wu

Advisor: Daniel Kane

UC San Diego Early Research Scholars Program (ERSP)

UC San Diego JACOBS SCHOOL OF ENGINEERING Computer Science and Engineering

MOTIVATION

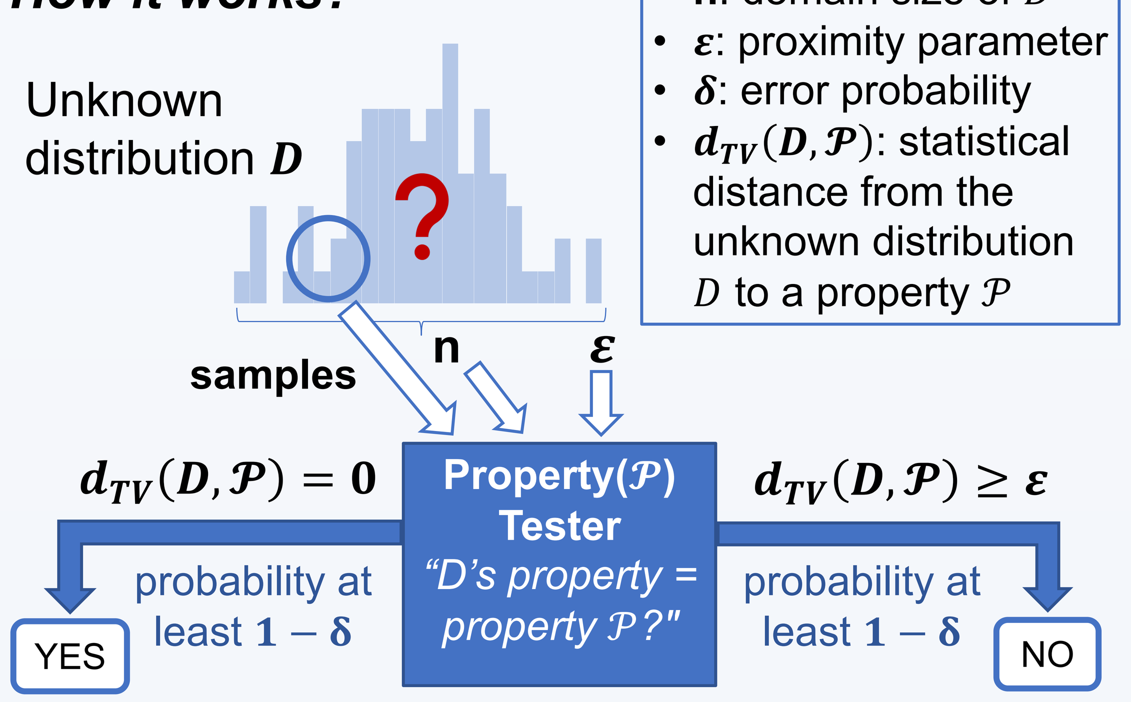
- Sample-efficient distribution testing algorithms have yet to be implemented and utilized
- Specific number of samples required for these algorithms remain undetermined
- Theoretical claims of papers need to be verified

BACKGROUND

Distribution Testing Test if an unknown distribution has some property

How it works?

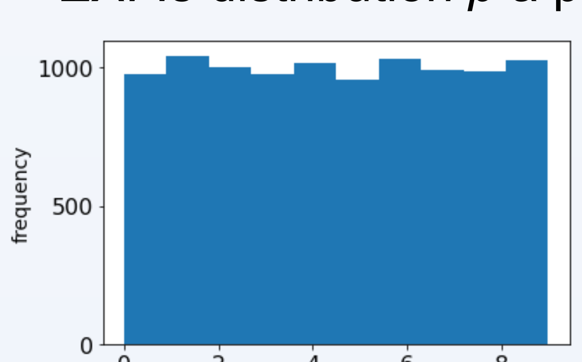
- **n**: domain size of D
- $\boldsymbol{\varepsilon}$: proximity parameter
- $\boldsymbol{\delta}$: error probability
- $\boldsymbol{d_{TV}(D, \mathcal{P})}$: statistical distance from the unknown distribution D to a property $\mathcal{P}$



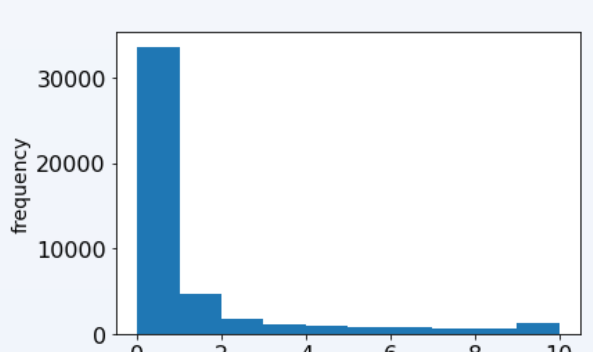
Uniformity Testing Test if all discrete bins have the same weight

Identity Testing Test whether an unknown distribution matches a known distribution

EX. Is distribution *p* a power law distribution?



Discrete Uniform Distribution



Discrete Power Law Distribution

Potential Applications of Distribution Testing

- Check if the distribution of NY State Lotto's winning numbers is uniform
- Determine if the dataset to regress is linear by checking if residual is normally distributed

PROBLEM STATEMENT

- To provide working implementations of theoretically-efficient distribution testing algorithms, with determined sample-complexity and relative run-times

METHODS

Implemented Algorithms

Uniformity tester	Identity tester
l_2 estimator	Chi-squared tester
Coincidence-based tester	instance-optimal tester
"New approach" uniformity	

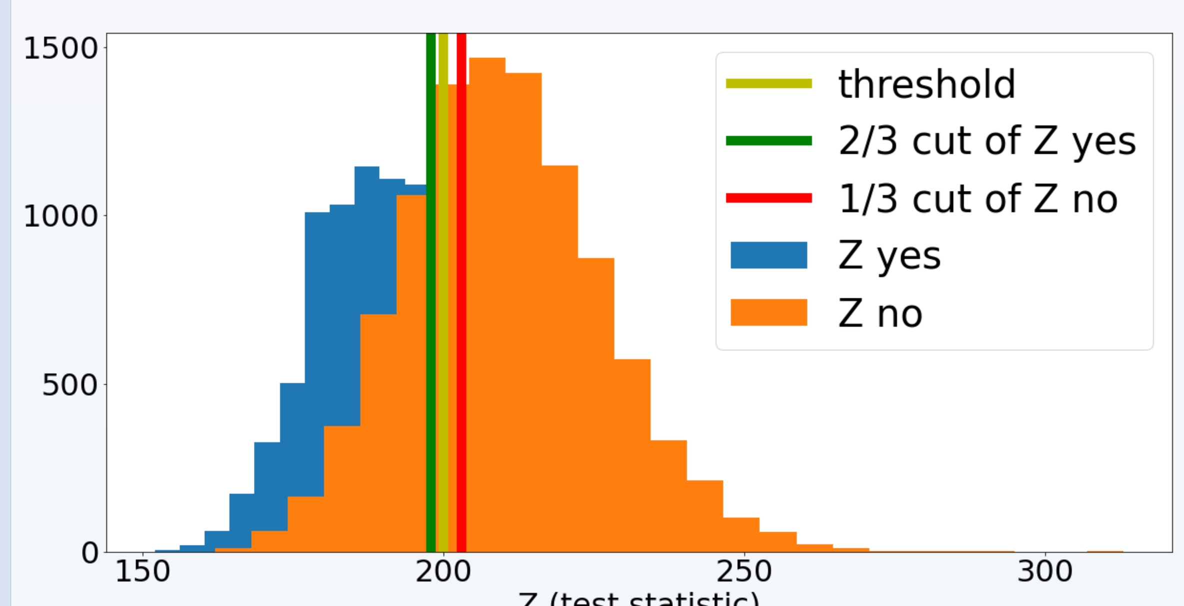
How to Pin Down Constants?

Algorithms compute test statistic Z on tested distribution and use Z to indicate if the distribution has the desired property.

STEP 1 Run the algorithms on both yes and no distributions for 10^6 times, to get z_{yes} and z_{no}.

STEP 2 Adjust number of samples to make the z_{yes} and z_{no} distribution overlap at percentile of $100 * \delta$ (ex. $\delta = \frac{1}{3}$, intersect at 33rd percentile).

STEP 3 Adjust the constant of threshold, making the threshold match the "true" threshold under all circumstances.

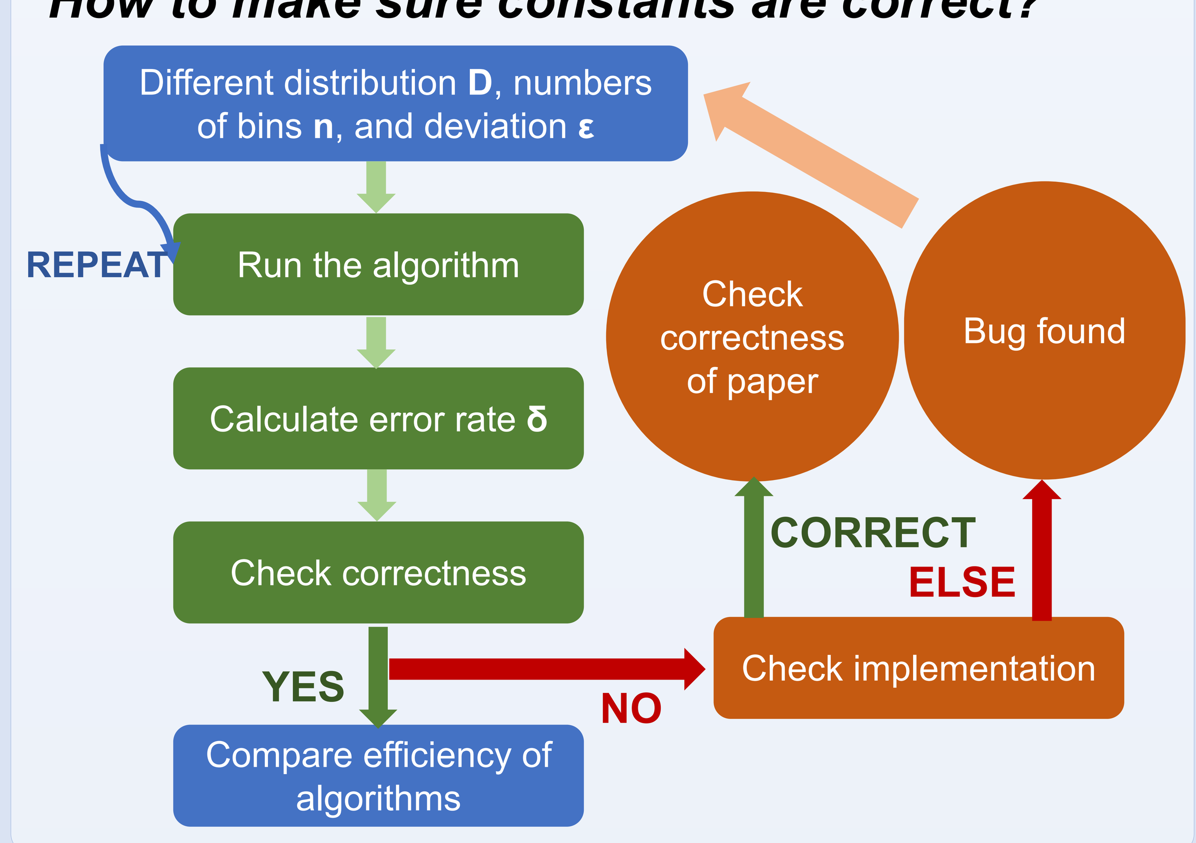


Distribution of Z_{yes} and Z_{no} returned by l_2-estimator tester

Testing Distributions

Uniformity Yes Distribution	Uniform D. over N
Uniformity No Distribution	$(1\pm\varepsilon)/n$ for all n bins
Identity Yes Distribution	$q_i = c_1(i+c_2)^{(-3/2)}$
Identity No Distribution	$p_i = q_i(1\pm\varepsilon)$

How to make sure constants are correct?



RESULTS & COMPARISONS

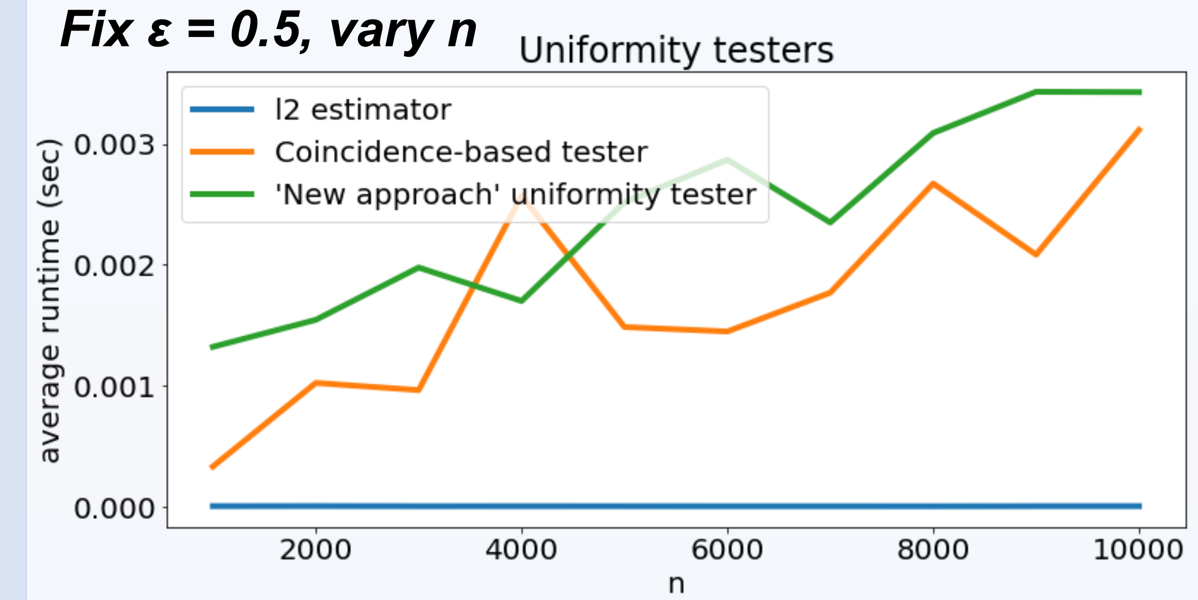
Worst-case Sample Size Pinned Down

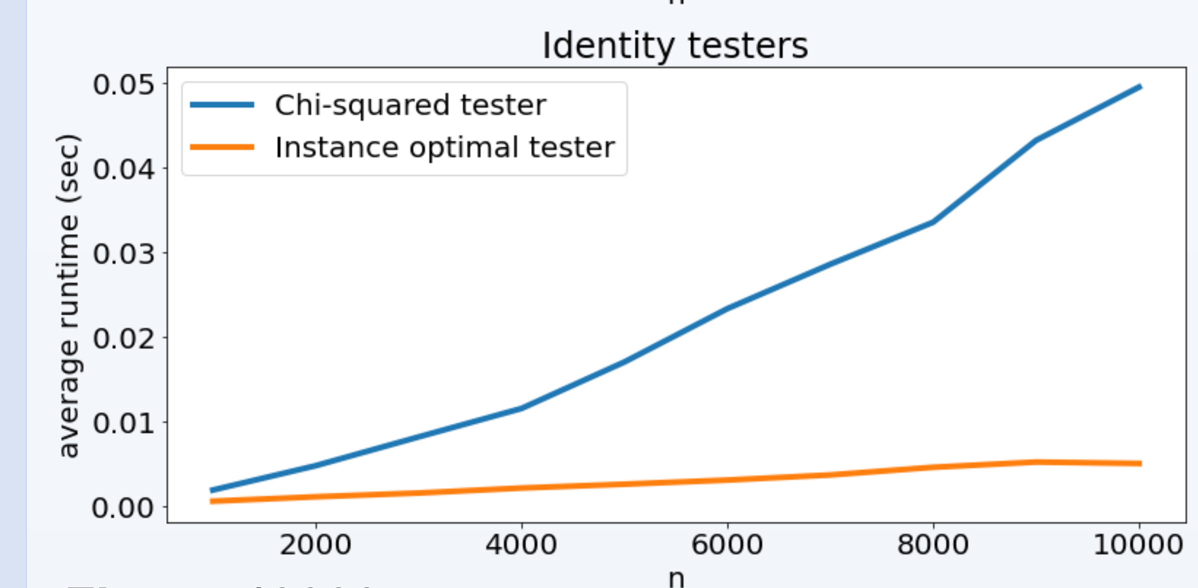
Uniformity tester	Sample size
l_2 estimator	$1.76 * \sqrt{n}/\varepsilon^2$
Coincidence-based tester	$2.1 * \sqrt{n}/\varepsilon^2$
"New approach" uniformity tester	$5.9 * \sqrt{n}/\varepsilon^2$

Identity tester	Sample size
Chi-squared tester	$1.45 * \sqrt{n}/\varepsilon^2$
Instance optimal tester	$1.8 * \max(\frac{1}{\varepsilon}, \|\|p\|\|_{\frac{2}{3}}/\varepsilon^2)$

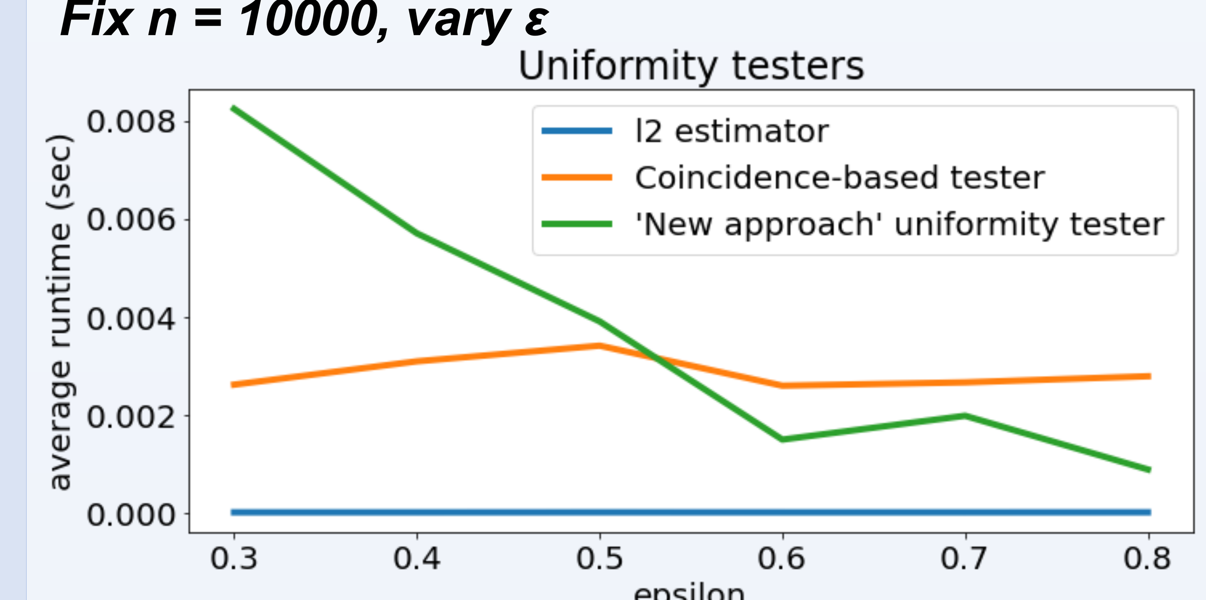
Graphs for Runtime

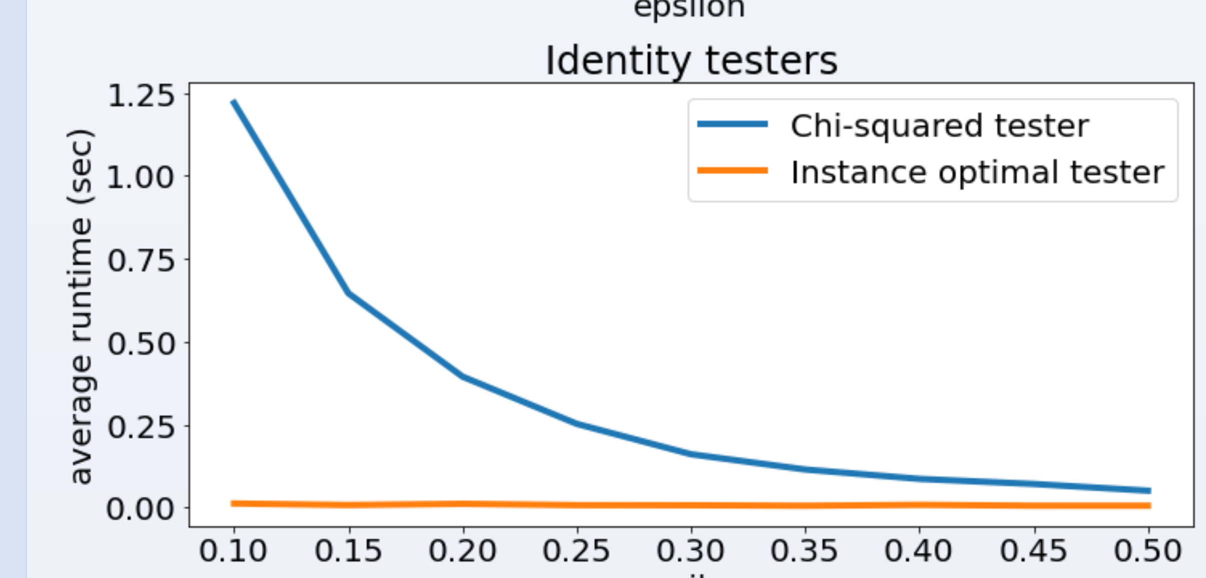
Fix ε = 0.5, vary n





Fix n = 10000, vary ε





Sample Complexity Analysis

- The worst-case sample size required for each tester is proportional to $\sqrt{n}/\varepsilon^2$ ($1.45 \leq C \leq 5.9$ for $C * \sqrt{n}/\varepsilon^2$)
- Our experimental results match the theoretical expectations.
- l_2 estimator is the most sample-efficient uniformity tester.
- Chi-squared tester is more sample-efficient when known distribution(q) is uniform; less efficient if q is a power law distribution.

Runtime Analysis

Ran on MacBook Air
Processor: 1.8 GHz Dual-core Intel Core i5
Memory: 8 GB 1600 MHz DDR3

- All algorithms are fast enough to run on even general-purpose computers (**~1** second).
- Outcome matches theoretical expectations.

Uniformity tester	
l_2 estimator	Fastest
Coincidence-based tester	Constant with all ε
"New approach" uniformity tester	Faster as ε increases

Identity tester	
Chi-squared tester	Fast with larger ε
Instance optimal tester	Faster

CONCLUSION

GitHub Repo for Implementations: SCAN ME

FUTURE WORK

- Implementation of closeness and independence testers
- Research paper to compile and share findings

ACKNOWLEDGEMENTS

We would like to thank Professor Kane and Sihan Liu for guiding our research, Professor Alvarado, Professor ElSherief, and Vaidehi Gupta for running the ERSP program and giving us this opportunity