Distribution Testing and Probabilistic Programming: A Match made in Heaven

Kuldeep S. Meel

School of Computing, National University of Singapore

Joint work with Sourav Chakraborty, Priyanka Golia, Yash Pote, and Mate Soos

Relevant publication: AAAI-19, NeurIPS-20, NeurIPS-21, FMCAD-21, CP-22
Probabilistic Programs and Distributions

program P
\[ X[1] = \text{Bernoulli}(p_1); \]
\[ X[2] = \text{Bernoulli}(p_2); \]
\[ \ldots \]
\[ X[n] = \text{Bernoulli}(p_n); \]
\[ \text{return } Y \]

- Semantics of discrete probabilistic programs: Distributions over \( \{0, 1\}^n \)
- Distance Measures
  - Closeness
    - \( d_\infty(P, Q) = \max_{x \in \{0, 1\}^n} |P(x) - Q(x)| \)
  - Farness
    - \( d_{TV}(P, Q) = \frac{1}{2} \sum_{x \in \{0, 1\}^n} |P(x) - Q(x)| \)
• Hardness of probabilistic program distance computation
• Grey-box Testing
• The Boon of Testers
Hardness of probabilistic program distance computation
Product Distributions

• Represented by list of probabilities: \( \{p_1, p_2, \ldots p_n\} \)

\[
P(x) = \prod_{x_i=1} p_i \prod_{x_i=0} (1 - p_i)
\]
Product Distributions

- Represented by list of probabilities: \{p_1, p_2, \ldots p_n\}
- \[ P(x) = \prod_{x_i=1} p_i \prod_{x_i=0} (1 - p_i) \]

**Question:** How hard it is to compute \(d_{TV}(P, Q)\) when \(P\) and \(Q\) are product distributions
Product Distributions

• Represented by list of probabilities: \( \{p_1, p_2, \ldots, p_n\} \)

• \( P(x) = \prod_{x_i=1} p_i \prod_{x_i=0} (1 - p_i) \)

**Question** How hard it is to compute \( d_{TV}(P, Q) \) when \( P \) and \( Q \) are product distributions

**program P**

\[
\begin{align*}
X[1] &= \text{Bernoulli}(p_1); \\
X[2] &= \text{Bernoulli}(p_2); \\
\vdots \\
X[n] &= \text{Bernoulli}(p_n); \\
\text{return } X;
\end{align*}
\]

**program Q**

\[
\begin{align*}
Y[1] &= \text{Bernoulli}(q_1); \\
Y[2] &= \text{Bernoulli}(q_2); \\
\vdots \\
Y[n] &= \text{Bernoulli}(q_n); \\
\text{return } Y
\end{align*}
\]
Theorem

*Given two product distributions $P$ and $Q$, computation of $d_{TV}(P, Q)$ is \#P-hard.*
Theorem

*Given two product distributions $P$ and $Q$, computation of $d_{TV}(P, Q)$ is $\#P$-hard.*

- We reduce from $\#\text{Knapsack}$, i.e., we construct $P$ and $Q$ such that $\#\{x : P(x) = Q(x)\}$ will solve $\#\text{Knapsack}$
Hardness

Theorem

Given two product distributions $P$ and $Q$, computation of $d_{TV}(P, Q)$ is $\#P$-hard.

- We reduce from $\#\text{Knapsack}$, i.e., we construct $P$ and $Q$ such that $\#\{x : P(x) = Q(x)\}$ will solve $\#\text{Knapsack}$
- Now given $P$ and $Q$, we construct $P'$ and $Q'$ where $p'_i = p_i$ and $q'_i = q_i$ for $i \leq n$ and $p'_{n+1} = \frac{1}{2} + \gamma$ and $q'_{n+1} = \frac{1}{2} - \gamma$ such that whenever $P(x) > Q(x)$ we have $P(x)(\frac{1}{2} - \gamma) > Q(x)(\frac{1}{2} + \gamma)$

\[
d_{TV}(P', Q') = \sum_{x \in \{0,1\}^n} \left( \left| P(x)(\frac{1}{2} + \gamma) - Q(x)(\frac{1}{2} - \gamma) \right| + \left| P(x)(\frac{1}{2} - \gamma) - Q(x)(\frac{1}{2} + \gamma) \right| \right) \\
= d_{TV}(P, Q) + 2\gamma \sum_{x: P(x) = Q(x)} P(x)
\]
Grey-box Testing
Basic Access

• We can run the program: sample!
What does Complexity Theory Tell Us

Figure: $\mathcal{U}$: Reference Distribution

Figure: $\mathcal{A}$: 1/2-far from uniform
What does Complexity Theory Tell Us

Figure: $\mathcal{U}$: Reference Distribution

Figure: $\mathcal{A}$: 1/2-far from uniform

- If $< \sqrt{S}/100$ samples are drawn then with high probability you see only distinct samples from either distribution.

Theorem (Batu-Fortnow-Rubinfeld-Smith-White (JACM 2013))

Testing whether a distribution is $\epsilon$-close to uniform has query complexity $\Theta(\sqrt{|S|}/\epsilon^2)$. [Paninski (Trans. Inf. Theory 2008)]

- If the output of a program is represented by 3 doubles, then $S > 2^{100}$
Definition (Conditional Sampling)

Given a distribution $\mathcal{A}$ on $S$ one can

- Specify a set $T \subseteq S$,
- Draw samples according to the distribution $\mathcal{A}|_T$, that is, $\mathcal{A}$ under the condition that the samples belong to $T$. 
Beyond Basic Access

Definition (Conditional Sampling)

Given a distribution $A$ on $S$ one can

- Specify a set $T \subseteq S$,
- Draw samples according to the distribution $A|_{T}$, that is, $A$ under the condition that the samples belong to $T$.

Conditional sampling is at least as powerful as drawing normal samples.

But how more powerful is it?
Testing Uniformity Using Conditional Sampling

An algorithm for testing uniformity using conditional sampling:

1. Draw two elements \( x \) and \( y \) uniformly at random from the domain.
2. Let \( T = \{x, y\} \).
3. In the case of the "far" distribution, with probability \( \frac{1}{2} \), one of the two elements will have probability 0, and the other probability non-zero.
4. Note \( p|_T = \sqrt{2} \) is a constant.
5. Now a constant number of conditional samples drawn from \( A|_T \) is enough to identify that it is not uniform.
An algorithm for testing uniformity using conditional sampling:

1. Draw two elements $x$ and $y$ uniformly at random from the domain. Let $T = \{x, y\}$.
2. In the case of the “far” distribution, with probability $\frac{1}{2}$, one of the two elements will have probability 0, and the other probability non-zero.
3. Note $\sqrt{|T|} = \sqrt{2}$ is a constant.
4. Now a constant number of conditional samples drawn from $A|_T$ is enough to identify that it is not uniform.
What about other distributions?

Previous algorithm fails in this case:

1. Draw two elements $\sigma_1$ and $\sigma_2$ uniformly at random from the domain. Let $T = \{\sigma_1, \sigma_2\}$.

2. In the case of the "far" distribution, with probability almost 1, both the two elements will have probability the same, namely $\epsilon$.

3. Probability that we will be able to distinguish the far distribution from the uniform distribution is very low.
What about other distributions?

Previous algorithm fails in this case:

1. Draw two elements $\sigma_1$ and $\sigma_2$ uniformly at random from the domain. Let $T = \{\sigma_1, \sigma_2\}$.

2. In the case of the “far” distribution, with probability almost 1, both the two elements will have probability same, namely $\epsilon$.

3. Probability that we will be able to distinguish the far distribution from the uniform distribution is very low.
Draw $\sigma_1$ uniformly at random from the domain and draw $\sigma_2$ according to the distribution $A$. Let $T = \{\sigma_1, \sigma_2\}$.

In the case of the "far" distribution, with constant probability, $\sigma_1$ will have "low" probability and $\sigma_2$ will have "high" probability.

We will be able to distinguish the far distribution from the uniform distribution using constant number of samples from $A|_T$.

The constant depend on the farness parameter.
Draw \( \sigma_1 \) uniformly at random from the domain and draw \( \sigma_2 \) according to the distribution \( \mathcal{A} \). Let \( T = \{\sigma_1, \sigma_2\} \).

In the case of the “far” distribution, with constant probability, \( \sigma_1 \) will have “low” probability and \( \sigma_2 \) will have “high” probability.

We will be able to distinguish the far distribution from the uniform distribution using constant number of samples from \( \mathcal{A}|_T \).

The constant depend on the farness parameter.
Input: A Distribution under test $\mathcal{A}$, a reference uniform distribution $\mathcal{U}$, a tolerance parameter $\varepsilon > 0$, an intolerance parameter $\eta > \varepsilon$, a guarantee parameter $\delta$

Output: ACCEPT or REJECT with the following guarantees:

- if the generator $\mathcal{A}$ is $\varepsilon$-close (in $d_\infty$), i.e., $d_\infty(\mathcal{A}, \mathcal{U}) \leq \varepsilon$ then Barbarik ACCEPTS with probability at least $(1 - \delta)$.

- If the generator $\mathcal{A}$ is $\eta$-far in ($d_{TV}$), i.e., $d_{TV}(\mathcal{A}, \mathcal{U}) > \eta$, then Barbarik REJECTS with probability at least $1 - \delta$. 
Theorem

Given $\varepsilon$, $\eta$ and $\delta$, Barbarik need at most $K = \tilde{O}\left(\frac{1}{(\eta - \varepsilon)^4}\right)$ samples for any input formula $\varphi$, where the tilde hides a poly logarithmic factor of $1/\delta$ and $1/(\eta - \varepsilon)$.

- $\varepsilon = 0.6$, $\eta = 0.9$, $\delta = 0.1$
- Maximum number of required samples $K = 1.72 \times 10^6$
- Independent of the number of variables
- To Accept, we need $K$ samples but rejection can be achieved with lesser number of samples.
Extensions to general case

**Setting** Given $P$ and $Q$, test whether $P$ and $Q$ are $\varepsilon$-close (in $d_\infty$) or $\eta$-far (in $d_{TV}$)

- For all $\sigma_1, \sigma_2 \in 0, 1^n$, $\frac{P(\sigma_1)}{P(\sigma_2)}$ is known: A prior distribution conditioned by constraints: dependence on $\max_{\sigma_1, \sigma_2 \in 0, 1^n} \frac{P(\sigma_1)}{P(\sigma_2)}$ (MPC20; PM21)

- Arbitrary $P$ and $Q$ (DCM22 – under submission)
The Boon of Testers
Where are the Samplers?

• Implicit representation of a set $S$: Set of all solutions of $\varphi$.
• Given a CNF formula $\varphi$, a Sampler $A$, outputs a random solution of $\varphi$.

**Definition**

A CNF-Sampler, $A$, is a randomized algorithm that, given a $\varphi$, outputs a random element of the set $S$, such that, for any $\sigma \in S$

$$\Pr[A(\varphi) = \sigma] = \frac{1}{|S|},$$
Where are the Samplers?

- Implicit representation of a set $S$: Set of all solutions of $\varphi$.
- Given a CNF formula $\varphi$, a Sampler $A$, outputs a random solution of $\varphi$.

**Definition**

A CNF-Sampler, $A$, is a randomized algorithm that, given a $\varphi$, outputs a random element of the set $S$, such that, for any $\sigma \in S$

$$
\Pr[A(\varphi) = \sigma] = \frac{1}{|S|},
$$

- Uniform sampling has wide range of applications in automated bug discovery, pattern mining, and so on.
- Several samplers available off the shelf: tradeoff between guarantees and runtime.
Constrained Sampling

- Input formula: \( F \) over variables \( X \)
- **Challenge:** Conditional Sampling over \( T = \{\sigma_1, \sigma_2\} \).
- Construct \( G = F \land (X = \sigma_1 \lor X = \sigma_2) \)
Constrained Sampling

- **Input formula:** $F$ over variables $X$
- **Challenge:** Conditional Sampling over $T = \{\sigma_1, \sigma_2\}$.
- Construct $G = F \land (X = \sigma_1 \lor X = \sigma_2)$
- Most of the samplers enumerate all the points when the number of points in the Domain are small
- Need way to construct formulas whose solution space is large but every solution can be mapped to either $\sigma_1$ or $\sigma_2$. 
Input: A Boolean formula $\varphi$, two assignments $\sigma_1$ and $\sigma_2$, and desired number of solutions $\tau$
Output: Formula $\hat{\varphi}$

1. $\tau = |\text{Sol}(\hat{\varphi})|$
2. $\text{Supp}(\varphi) \subseteq \text{Supp}(\hat{\varphi})$
3. $z \in \text{Sol}(\hat{\varphi}) \implies z_{\downarrow S} \in \{\sigma_1, \sigma_2\}$
4. $|\{z \in \text{Sol}(\hat{\varphi}) \mid z_{\downarrow \text{Supp}(\varphi)} = \sigma_1\}| = |\{z \in \text{Sol}(\hat{\varphi}) \mid z_{\downarrow \text{Supp}(\varphi) \cap \sigma_2}\}|$
5. $\varphi$ and $\hat{\varphi}$ has similar structure
Non-adversarial Sampler

Let \((\hat{\phi})\) obtained from \(kernel(\varphi, \sigma_1, \sigma_2, N)\) such that there are only two set of assignments to variables in \(\varphi\) that can be extended to a satisfying assignment for \(\hat{\varphi}\)

**Definition**

The **non-adversarial sampler assumption** states that the distribution of the projection of samples obtained from \(A(\hat{\varphi})\) to variables of \(\varphi\) is same as the conditional distribution of \(A(\varphi)\) restricted to either \(\sigma_1\) or \(\sigma_2\)

- If \(A\) is a uniform sampler for all the input formulas, it satisfies non-adversarial sampler assumption
- If \(A\) is not a uniform sampler for all the input formulas, it may not necessarily satisfy non-adversarial sampler assumption
Barbarik

Input: A sampler under test $\mathcal{A}$, a reference uniform sampler $\mathcal{U}$, a tolerance parameter $\varepsilon > 0$, an intolerance parameter $\eta > \varepsilon$, a guarantee parameter $\delta$ and a CNF formula $\varphi$

Output: ACCEPT or REJECT with the following guarantees:

- if the generator $\mathcal{A}$ is $\varepsilon$-close (in $d_\infty$), i.e., $d_\infty(\mathcal{A}, \mathcal{U}) \leq \varepsilon$ then Barbarik ACCEPTS with probability at least $(1 - \delta)$.
- If the generator $\mathcal{A}$ is $\eta$-far in $(d_{TV})$, i.e., $d_{TV}(\mathcal{A}, \mathcal{U}) > \eta$ and if non-adversarial sampler assumption holds then Barbarik REJECTS with probability at least $1 - \delta$. 
Experimental Setup

• Three state of the art (almost-)uniform samplers
  – UniGen2: Theoretical Guarantees of almost-uniformity
  – SearchTreeSampler: Very weak guarantees
  – QuickSampler: No Guarantees

• The study (in 2018) that proposed Quicksampler perform unsound statistical tests and claimed that all the three samplers are indistinguishable
## Results-I

<table>
<thead>
<tr>
<th>Instances</th>
<th>#Solutions</th>
<th>UniGen2</th>
<th>SearchTreeSampler</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Output</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#Samples</td>
<td>#Samples</td>
</tr>
<tr>
<td>71</td>
<td>$1.14 \times 2^{59}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case49</td>
<td>$1.00 \times 2^{61}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case50</td>
<td>$1.00 \times 2^{62}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>scenarios_aig_insertion1</td>
<td>$1.06 \times 2^{65}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>scenarios_aig_insertion2</td>
<td>$1.06 \times 2^{65}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>36</td>
<td>$1.00 \times 2^{72}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>30</td>
<td>$1.73 \times 2^{72}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>110</td>
<td>$1.09 \times 2^{76}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>scenarios_tree_insert_insert</td>
<td>$1.32 \times 2^{76}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>107</td>
<td>$1.52 \times 2^{76}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case211</td>
<td>$1.00 \times 2^{80}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case210</td>
<td>$1.00 \times 2^{80}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case212</td>
<td>$1.00 \times 2^{88}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case209</td>
<td>$1.00 \times 2^{88}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>54</td>
<td>$1.15 \times 2^{90}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>Instances</td>
<td>#Solutions</td>
<td>UniGen2</td>
<td>QuickSampler</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------</td>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Output</td>
<td>#Samples</td>
</tr>
<tr>
<td>71</td>
<td>$1.14 \times 2^{59}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case49</td>
<td>$1.00 \times 2^{61}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case50</td>
<td>$1.00 \times 2^{62}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>scenarios_aig_insertion1</td>
<td>$1.06 \times 2^{65}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>scenarios_aig_insertion2</td>
<td>$1.06 \times 2^{65}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>36</td>
<td>$1.00 \times 2^{72}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>30</td>
<td>$1.73 \times 2^{72}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>110</td>
<td>$1.09 \times 2^{76}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>scenarios_tree_insert_insert</td>
<td>$1.32 \times 2^{76}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>107</td>
<td>$1.52 \times 2^{76}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case211</td>
<td>$1.00 \times 2^{80}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case210</td>
<td>$1.00 \times 2^{80}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case212</td>
<td>$1.00 \times 2^{88}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>blasted_case209</td>
<td>$1.00 \times 2^{88}$</td>
<td>A</td>
<td>1729750</td>
</tr>
<tr>
<td>54</td>
<td>$1.15 \times 2^{90}$</td>
<td>A</td>
<td>1729750</td>
</tr>
</tbody>
</table>
How can we use the availability of Barbarik to design a good sampler?

- Is it even possible?
How can we use the availability of Barbarik to design a good sampler?

• Is it even possible?

Wishlist

• Sampler should pass the Barbarik test.

• Sampler should be at least as fast as STS and QuickSampler.

• Sampler should perform good on real world applications.
CMSGen

- Exploits the flexibility CryptoMiniSat.
CMSGen

- Exploits the flexibility CryptoMiniSat.

- Pick polarities and branch on variables at random.
  - To explore the search space as evenly as possible.
  - To have samples over all the solution space.
CMSGen

• Exploits the flexibility CryptoMiniSat.

• Pick polarities and branch on variables at random.
  – To explore the search space as evenly as possible.
  – To have samples over all the solution space.

• Turn off all pre and inprocessing.
  – Processing techniques: bounded variable elimination, local search, or symmetry breaking, and many more.
  – Can change solution space of instances.
CMSGen

• Exploits the flexibility CryptoMiniSat.

• Pick polarities and branch on variables at random.
  – To explore the search space as evenly as possible.
  – To have samples over all the solution space.

• Turn off all pre and inprocessing.
  – Processing techniques: bounded variable elimination, local search, or symmetry breaking, and many more.
  – Can change solution space of instances.

• Restart at static intervals.
  – Helps to generate samples which are very hard to find.
• Test-Driven Development of CMSGen.

• Parameters of CMSGen are decided with the help of Barbarik
  – Iterative process.
  – Based on feedback from Barbarik, change the parameters.

• Uniform-like-sampler.

• Lack of theoretical analysis.
Testing of Samplers

- Samplers without guarantees (Uniform-like Samplers):
  - STS (Ermon, Gomes, Sabharwal, Selman, 2012)
  - QuickSampler (Dutra, Laeufer, Bachrach, Sen, 2018)

- Sampler with guarantees:

<table>
<thead>
<tr>
<th></th>
<th>QuickSampler</th>
<th>STS</th>
<th>UniGen3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPTs</td>
<td>0</td>
<td>14</td>
<td>50</td>
</tr>
<tr>
<td>REJECTs</td>
<td>50</td>
<td>36</td>
<td>0</td>
</tr>
</tbody>
</table>
Testing of Samplers

- Samplers without guarantees (Uniform-like Samplers):
  - STS (Ermon, Gomes, Sabharwal, Selman, 2012)
  - QuickSampler (Dutra, Laeufer, Bachrach, Sen, 2018)
  - CMSGen

- Sampler with guarantees:
  - UniGen3 (Chakraborty, Meel, and Vardi 2013, 2014, 2015)

<table>
<thead>
<tr>
<th></th>
<th>QuickSampler</th>
<th>STS</th>
<th>UniGen3</th>
<th>CMSGen</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCEPTs</td>
<td>0</td>
<td>14</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>REJECTs</td>
<td>50</td>
<td>36</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
• Sampler should pass the Barbarik test. ✓

• Sampler should be at least as fast as STS and QuickSampler.

• Sampler should perform good on real world applications.
CMSGen vs. Other State-of-the-Art Samplers

- 70 Benchmarks arising from:
  - probabilistic reasoning, (Chakraborty, Fremont, Meel et al., 2015)
  - bounded model checking. (Clarke, Biere, Raimi, Zhu, 2001)
  - bug synthesis. (Roy, Pandey, Dolan-Gavitt, Hu, 2018)

- Runtime evaluation to generate 1000 samples.

- Timeout: 7200 seconds.
CMSGen vs. Other State-of-the-Art Samplers (II)

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>CMSGen</th>
<th>STS</th>
<th>QuickSampler</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>37</td>
<td>52</td>
<td></td>
</tr>
</tbody>
</table>
Wishlist

• Sampler should pass the Barbarik test. ✓

• Sampler should be at least as fast as STS and QuickSampler. ✓

• Sampler should perform good on real world applications.
Combinatorial Testing

• A powerful paradigm for testing configurable system.

• Challenge: To generate test suites that maximizes \( t \)-wise coverage.

\[
\text{t-wise coverage: } = \frac{\# \text{ of } t\text{-sized combinations in test suite}}{\text{all possible valid } t\text{-sized combinations}}
\]

• To generate the test suites use constraint samplers.

• Uniform sampling to have high \( t \)-wise coverage (Plazar, Acher, Perrouin et al., 2019).
Combinatorial Testing

• A powerful paradigm for testing configurable system.

• Challenge: To generate test suites that maximizes $t$-wise coverage.

\[
\text{t-wise coverage: } = \frac{\text{\# of t-sized combinations in test suite}}{\text{all possible valid t-sized combinations}}
\]

• To generate the test suites use constraint samplers.

• Uniform sampling to have high $t$-wise coverage (Plazar, Acher, Perrouin et al., 2019).

• Experimental Evaluations:
  - Generate 1000 samples (test cases).
  - 110 Benchmarks, Timeout: 3600 seconds
  - 2-wise coverage $t = 2$. 
Combinatorial Testing: The Power of CMSGen

Higher is better

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>CMSGen</th>
<th>STS</th>
<th>QuickSampler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage %</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100%</td>
<td>90%</td>
<td>80%</td>
</tr>
<tr>
<td>20</td>
<td>100%</td>
<td>90%</td>
<td>80%</td>
</tr>
<tr>
<td>40</td>
<td>100%</td>
<td>90%</td>
<td>80%</td>
</tr>
<tr>
<td>60</td>
<td>100%</td>
<td>90%</td>
<td>80%</td>
</tr>
<tr>
<td>80</td>
<td>100%</td>
<td>90%</td>
<td>80%</td>
</tr>
<tr>
<td>100</td>
<td>100%</td>
<td>90%</td>
<td>80%</td>
</tr>
<tr>
<td>120</td>
<td>100%</td>
<td>90%</td>
<td>80%</td>
</tr>
</tbody>
</table>

Avg. Coverage

<table>
<thead>
<tr>
<th></th>
<th>QuickSampler</th>
<th>STS</th>
<th>CMSGen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>51.5%</td>
<td>80.15%</td>
<td>~ 100%</td>
</tr>
</tbody>
</table>
A Long and Winding Road

- Hardness of probabilistic program distance computation
- Distribution Testing
- The Boon of Testers

Open Source Tools

- Barbarik  https://github.com/meelgroup/barbarik
- CMSGen  https://github.com/meelgroup/cmsgen
A Long and Winding Road

- Hardness of probabilistic program distance computation
- Distribution Testing
- The Boon of Testers

Open Source Tools

- Barbarik  https://github.com/meelgroup/barbarik
- CMSGen  https://github.com/meelgroup/cmsgen

Where do we go from here?

- Exploring the tradeoffs: the cost of conditioning and query complexity
- Modular proofs
- Benchmark suite generation

These slides are available at https://tinyurl.com/meel-talk