Prioritizing Mutation Operators using Probabilistic Sampling
International Symposium on Software Reliability Engineering (ISSRE 2010)

Mohan Sridharan\textsuperscript{1}  Akbar Siami Namin\textsuperscript{2}

\textsuperscript{1}Stochastic Estimation and Autonomous Robotics Lab (SEARL)  
Department of Computer Science  
Texas Tech University, USA  
mohan.sridharan@ttu.edu

\textsuperscript{2}AdVanced Empirical Software Testing and Analysis (AVESTA) Research Group  
Department of Computer Science  
Texas Tech University, USA  
akbar.namin@ttu.edu
Outline

- **Introduction:**
  - Motivation.
  - Related Work.

- **Proposed Approach:**
  - Hypothesis.
  - Stochastic sampling.
  - Probabilistic formulation.

- **Experimental Results:**
  - Case studies.
  - Figures and tables.
  - Discussion.

- **Conclusions:**
  - Conclusions.
  - Further Reading.
Introduction

Motivation.

Related Work.

Proposed Approach:

Hypothesis.

Stochastic sampling.

Probabilistic formulation.

Experimental Results:

Case studies.

Figures and tables.

Discussion.

Conclusions:

Conclusions.

Further Reading.
Mutation Testing

- Fault-based testing technique for assessing the adequacy of test suites.
- Artificially inject syntactic faults in the program.
- **Mutation operators**: mathematical transformations to synthesize faults, i.e., mutants.
- E.g. Binary operator to binary operator (BoB):

  \[
  \ldots \ldots \ldots
  \]

  \[
  \text{if}(a > b) \text{ then } \rightarrow \text{if}(a < b) \text{ then}
  \]

  \[
  \ldots \ldots \ldots
  \]

- Augment test suites to detect unexposed faults.
Mutation Generation

- Each operator represents a specific type of fault.
- Several mutation operators have been created.
- Proteum:
  - A mutation tool for C with 108 operators.
- MuJava:
  - A mutation tool for Java.
  - 12 traditional mutation operators. 28 class mutation operators.
Feasibility of Mutation Testing

- Each mutant has to be compiled and executed for the entire pool of test cases.

- The number of mutants generated for the target program can be large.
  - For instance, Proteum (108 operators) generated 4935 mutants on a program with 137 lines of code.

- Examining all mutants can be intractable!
Mutation Score

Mutation score definition:

\[ MS = AM = \frac{\# \text{killed mutants}}{\# \text{Total non-equivalent mutants}} \]  

Two possible options:
- Mutants are exposed by the existing test suite.
- Mutants are hard to kill and the test suite needs to be augmented.

Drawback: The mutation score can grow without reflecting the quality of testing.
Reducing Cost of Mutation Testing

- Standard mutation testing [DeMillo et al, 1978; Hamlet 1977].
- Meta-mutants [Offutt and Untch, 2000]:
  - Creating and compiling one file to include all mutants.
- Selective mutation [Offutt et al., 1996]:
  - Considering only a subset of mutation operators.
- Sufficient mutation operators [Namin et al., 2008]:
  - Identifying a sufficient set of mutation operators using regression and other estimation methods.
## Outline

- **Introduction:**
  - Motivation.
  - Related Work.

- **Proposed Approach:**
  - Hypothesis.
  - Stochastic sampling.
  - Probabilistic formulation.

- **Experimental Results:**
  - Case studies.
  - Figures and tables.
  - Discussion.

- **Conclusions:**
  - Conclusions.
  - Further Reading.
Our Hypothesis:

A mutation operator, most of whose mutants have been caught by the existing test suite, is likely to produce mutants that can be exposed using the existing test suite.
Focus on operators that are *likely* to generate mutants that will need an augmentation of the test suite.

Definition of *important* operators:
- Operators that tend to generate mutants that need an augmentation of the test suite.

Prioritize mutation operators based on their importance.
- Consider the top few operators for detailed analysis.
Stochastic Sampling

- Used to track multiple hypotheses in practical domains.
- **Probabilistic representation** for each hypothesis.
- Iteratively focus on the more important hypotheses.

- **Sampling algorithms:** rejection sampling, importance sampling, Gibbs sampling, MCMC.
- **Several applications:**
  - Tracking multiple humans in image sequences.
  - Finding likely locations of celestial objects, i.e., in astronomy.
Stochastic Sampling

- Used to track multiple hypotheses in practical domains.
- Probabilistic representation for each hypothesis.
- Iteratively focus on the more important hypotheses.

**Sampling algorithms:** rejection sampling, importance sampling, Gibbs sampling, MCMC.

**Several applications:**
- Tracking multiple humans in image sequences.
- Finding likely locations of celestial objects, i.e., in astronomy.
Important Sampling Overview

- Assign probability to each hypothesis.
- Generate initial set of *samples* of each hypothesis based on the corresponding probabilities.

In each of a set of iterations:
- Adjust samples to account for *dynamic changes* in the system: *prediction* step.
- Use observations of the system to update probabilities of the samples: *correction* step.
- Resample, i.e., generate samples of each hypothesis proportional to the updated probabilities.
Importance Sampling Overview

- Assign probability to each hypothesis.
- Generate initial set of *samples* of each hypothesis based on the corresponding probabilities.

- In each of a set of iterations:
  - Adjust samples to account for *dynamic changes* in the system: *prediction* step.
  - Use observations of the system to *update* probabilities of the samples: *correction* step.
  - Resample, i.e., generate samples of each hypothesis proportional to the updated probabilities.
Importance Sampling Overview

- Assign probability to each hypothesis.
- Generate initial set of *samples* of each hypothesis based on the corresponding probabilities.

In each of a set of iterations:
- Adjust samples to account for *dynamic changes* in the system: *prediction* step.
- Use observations of the system to *update* probabilities of the samples: *correction* step.
- *Resample*, i.e., generate samples of each hypothesis proportional to the updated probabilities.
The typical importance sampling framework:

1. **Generate Initial Samples**
2. **Modify Samples**
3. **Re-sampling**
4. **Update Probabilities**
   (Prediction)
   (Correction)
Probabilistic update and re-sampling:
Importance Sampling for Mutation Testing

- Probability for each mutation operator: \( p_i \) for \( \mu_i \forall i \in [1, N] \).

- Select initial (small) set of mutant samples of each operator, choosing *uniformly* or *proportional* to operator probabilities:

\[
numMutantSamps_i^0 \sim \begin{cases} 
\frac{c}{\propto \frac{Nm_i}{NM}} & \text{uniform} \\
\propto \frac{Nm_i}{NM} & \text{proportional} \end{cases}
\]  (2)
Importance Sampling for Mutation Testing

- Probability for each mutation operator: $p_i$ for $\mu_i \forall i \in [1, N]$.

- Select initial (small) set of mutant samples of each operator, choosing *uniformly* or *proportional* to operator probabilities:

$$numMutantSamps_i^0 \sim \left\{ \begin{array}{ll} c & \text{uniform} \\ \propto \frac{Nm_i}{NM} & \text{proportional} \end{array} \right.$$

(2)
Sampling Iterations

- Examine the ability of existing test suites to expose selected mutants.
- Increase probabilities of operators whose mutants are unexposed.

\[ p_i^t = p_i^{t-1} + \frac{\delta p_i^t}{\text{totalMutantSamps}_t^t} \quad (3) \]

\[ \delta p_i^t = -1.0 + 2.0 \frac{\text{numAlive}_i^t}{\text{numMutantSamps}_i^t} : \in [-1.0, 1.0] \]

\[ \text{totalMutantSamps}^t = \sum_{i=0}^{N-1} \text{numMutantSamps}^t_i \]

- Generate samples of each operator proportional to probabilities.
Recap: Pictorial Representation

- Probabilistic update and re-sampling:
Innovations in Sampling

- Sample without replacement: *stationary* system.
- *Adapt* number of samples based on uncertainty:

\[ y^{t+1} = \frac{1}{2\epsilon} \chi_{q^t-1,1-\delta}^2 \]

\[ \simeq \frac{q^t - 1}{2\epsilon} \left\{ 1 - \frac{2}{9(q^t - 1)} + \sqrt{\frac{2}{9(q^t - 1)} z_{1-\delta}} \right\}^3 \]  

- Entropy in operator probability distribution:

\[ E^t = - \sum_{j=0}^{N-1} p_j^t \cdot ln(p_j^t) \]  

- *Termination condition:*  
  \[ E^t - E^{t-1} \leq \text{threshold} \]
Innovations in Sampling

- Sample without replacement: stationary system.
- Adapt number of samples based on uncertainty:

\[ y^{t+1} = \frac{1}{2\epsilon} \chi^2_{q^t - 1, 1 - \delta} \]

\[ \simeq \frac{q^t - 1}{2\epsilon} \left\{ 1 - \frac{2}{9(q^t - 1)} + \sqrt{\frac{2}{9(q^t - 1)}} z_{1-\delta} \right\}^3 \]

- Entropy in operator probability distribution:

\[ E^t = - \sum_{j=0}^{N-1} p_j^t \cdot \ln(p_j^t) \]  \hspace{1cm} (5)

- Termination condition: \( E^t - E^{t-1} \leq \text{threshold} \)
Innovations in Sampling

- Sample without replacement: *stationary* system.
- *Adapt* number of samples based on uncertainty:

\[
\gamma_{t+1} = \frac{1}{2\epsilon} \chi^{2}_q{t-1,1-\delta}
\]

\[
\approx \frac{q^t - 1}{2\epsilon} \left\{ 1 - \frac{2}{9(q^t - 1)} + \sqrt{\frac{2}{9(q^t - 1)}} z_{1-\delta} \right\}^3
\]

- **Entropy** in operator probability distribution:

\[
E^t = - \sum_{j=0}^{N-1} p_j^t \cdot \ln(p_j^t)
\]

- **Termination condition**: \( E^t - E^{t-1} \leq \text{threshold} \)

Mohan Sridharan, Akbar Namin

ISSRE-2010 Talk
Innovations in Sampling

- Sample without replacement: *stationary* system.
- *Adapt* number of samples based on uncertainty:

\[
Y^{t+1} = \frac{1}{2\epsilon} \chi^2_{q^t-1,1-\delta} \\
\approx \frac{q^t - 1}{2\epsilon} \left\{ 1 - \frac{2}{9(q^t - 1)} + \sqrt{\frac{2}{9(q^t - 1)} z^{1-\delta}} \right\}^3
\]

- **Entropy** in operator probability distribution:

\[
E^t = -\sum_{j=0}^{N-1} p_j^t \cdot \ln(p_j^t)
\]

- **Termination condition**: \(E^t - E^{t-1} \leq \text{threshold}\)
Siemens Dataset:

<table>
<thead>
<tr>
<th>Program</th>
<th>NLOC</th>
<th>NTC</th>
<th>NMG</th>
<th>NMS</th>
<th>NM</th>
</tr>
</thead>
<tbody>
<tr>
<td>printtokens</td>
<td>343</td>
<td>4130</td>
<td>11741</td>
<td>2000</td>
<td>1551</td>
</tr>
<tr>
<td>printtokens2</td>
<td>355</td>
<td>4115</td>
<td>10266</td>
<td>2000</td>
<td>1942</td>
</tr>
<tr>
<td>replace</td>
<td>513</td>
<td>5542</td>
<td>23847</td>
<td>2000</td>
<td>1969</td>
</tr>
<tr>
<td>schedule</td>
<td>296</td>
<td>2650</td>
<td>4130</td>
<td>2000</td>
<td>1760</td>
</tr>
<tr>
<td>schedule2</td>
<td>263</td>
<td>2710</td>
<td>6552</td>
<td>2000</td>
<td>1497</td>
</tr>
<tr>
<td>tcas</td>
<td>137</td>
<td>1608</td>
<td>4935</td>
<td>4935</td>
<td>4935</td>
</tr>
<tr>
<td>totinfo</td>
<td>281</td>
<td>1052</td>
<td>8767</td>
<td>2000</td>
<td>1740</td>
</tr>
</tbody>
</table>

Table: Description of subject programs. NLOC: Net lines of code. NTC: Number of test cases. NMG: Number of mutants generated by all mutation operators. NMS: Number of randomly selected mutants. NM: Number of selected mutants that were non-equivalent.
Larger Programs

- Evaluated performance on gzip and space.
- Program gzip:
  - NLOC: 5680, 212 test cases.
  - 43 mutation operators, 28 based on sufficient set.
  - 317 non-equivalent mutants used.
  - Study effect of sufficient set of mutation operators.

- Program space:
  - NLOC: 5905, 13585 test cases.
  - 108 mutation operators.
  - 23708 non-equivalent mutants used.
  - Study effect of larger set of test cases.
Data Collection

- Proteum generates mutants with 108 mutant operators.
- Test suites of size 1 – 50 by choosing randomly from the available test cases.
- Use test suites on the mutants to record mutants killed and left alive.
- *This provides the ground truth.*

- Compare sampling-based approach against ground truth.
- Compare two schemes for initial operator probabilities: *uniform* and *proportional*. 
Measures Used

- **Mutation Importance Measure:** \( IM = 1 - AM \)

\[
IM = \frac{\# \text{alive mutants}}{\# \text{Total non-equivalent mutants}} \tag{6}
\]

Measures fraction of examined mutants left alive.

- **Operator Overlap Measure:**

\[
OpOverlap(P, S, T) = Overlap(Gt(P, S), Obs(P, S)) \tag{7}
\]

Measures fraction of top mutants in ground truth list \((Gt)\) that exist in the observed list \((Obs)\).
Probability Updates

Sampling converges on important mutation operators.
Entropy and adaptive sampling enable convergence while examining a small set of samples.
## Siemens: Uniform Initial Probability

<table>
<thead>
<tr>
<th>Programs</th>
<th>Operator Overlap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
</tr>
<tr>
<td>printtokens</td>
<td>0.93</td>
</tr>
<tr>
<td>printtokens2</td>
<td>0.92</td>
</tr>
<tr>
<td>replace</td>
<td>1.0</td>
</tr>
<tr>
<td>schedule</td>
<td>0.90</td>
</tr>
<tr>
<td>schedule2</td>
<td>0.91</td>
</tr>
<tr>
<td>tcas</td>
<td>1.0</td>
</tr>
<tr>
<td>totinfo</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**Table:** Operator overlap with uniform initial operator probability assignment. Dynamic selection of “T” provides better results.
## Siemens: Proportional Initial Probability

<table>
<thead>
<tr>
<th>Programs</th>
<th>Max</th>
<th>Min</th>
<th>Average</th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>printtokens</td>
<td>0.94</td>
<td>0.60</td>
<td>0.77 (\pm) 0.09</td>
<td>0.92</td>
</tr>
<tr>
<td>printtokens2</td>
<td>0.97</td>
<td>0.47</td>
<td>0.66 (\pm) 0.13</td>
<td>0.9</td>
</tr>
<tr>
<td>replace</td>
<td>1.0</td>
<td>0.65</td>
<td>0.86 (\pm) 0.07</td>
<td>0.94</td>
</tr>
<tr>
<td>schedule</td>
<td>0.90</td>
<td>0.48</td>
<td>0.67 (\pm) 0.11</td>
<td>0.98</td>
</tr>
<tr>
<td>schedule2</td>
<td>0.92</td>
<td>0.66</td>
<td>0.78 (\pm) 0.05</td>
<td>0.91</td>
</tr>
<tr>
<td>tcas</td>
<td>0.98</td>
<td>0.85</td>
<td>0.91 (\pm) 0.04</td>
<td>0.92</td>
</tr>
<tr>
<td>totinfo</td>
<td>0.96</td>
<td>0.51</td>
<td>0.74 (\pm) 0.09</td>
<td>0.96</td>
</tr>
</tbody>
</table>

**Table:** Operator overlap with proportional initial operator probability distribution. Dynamic selection of “T” provides better results.
OpOverlap for *gzip* and *space*:

<table>
<thead>
<tr>
<th>Programs</th>
<th>Operator Overlap</th>
<th></th>
<th></th>
<th>Dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max</td>
<td>Min</td>
<td>Average</td>
<td></td>
</tr>
<tr>
<td><strong>Uniform</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzip</td>
<td>0.98</td>
<td>0.95</td>
<td>0.96 ± 0.01</td>
<td>0.97</td>
</tr>
<tr>
<td>space</td>
<td>0.91</td>
<td>0.35</td>
<td>0.64 ± 0.17</td>
<td>0.81</td>
</tr>
<tr>
<td><strong>Proportional</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gzip</td>
<td>0.98</td>
<td>0.95</td>
<td>0.96 ± 0.01</td>
<td>0.97</td>
</tr>
<tr>
<td>space</td>
<td>0.96</td>
<td>0.34</td>
<td>0.67 ± 0.19</td>
<td>0.87</td>
</tr>
</tbody>
</table>

**Table:** Operator overlap for *gzip* and *space* with uniform and proportional initial operator probability distributions.
The proposed approach is able to identify 90% of the important operators by examining $\leq 20\%$ of the mutants over 3 – 4 sampling iterations.

There is a 6% increase in IM, i.e., larger proportion of examined mutants remain unexposed.

Automatic tuning of parameters to match the specific program and test suite.

Performance generalizes across different programs and test suite sizes.
Performance generalizes to program with sufficient set of mutants (gzip) and large number of test cases (space).

Similar performance with different schemes of initial probability assignment.

The operators identified as being important make logical sense—see paper for detailed results.

Initial hypothesis justified!
Threats to Validity

- Small to medium-sized C programs.
- Do not investigate object-oriented programs.
- Random selection of mutants for each program.
- Correctness of the data collection process.
Outline

- **Introduction:**
  - Motivation.
  - Related Work.

- **Proposed Approach:**
  - Hypothesis.
  - Stochastic sampling.
  - Probabilistic formulation.

- **Experimental Results:**
  - Case studies.
  - Figures and tables.
  - Discussion.

- **Conclusions:**
  - Conclusions.
  - Further Reading.
Conclusions

- Mutation testing formulated as a importance sampling (i.e., probabilistic) problem.

- Adaptive sampling and information theoretic measures enable reliable and efficient mutation testing.

- Detects $\geq 90\%$ of the truly important mutation operators by examining $\leq 20\%$ of the available mutants.

- Generalizes to medium-sized programs.

- Sampling is well-suited for many other software testing applications!
Mutation testing formulated as a importance sampling (i.e., probabilistic) problem.

Adaptive sampling and information theoretic measures enable reliable and efficient mutation testing.

- Detects $\geq 90\%$ of the truly important mutation operators by examining $\leq 20\%$ of the available mutants.
- Generalizes to medium-sized programs.

Sampling is well-suited for many other software testing applications!
Mutation testing formulated as a importance sampling (i.e., probabilistic) problem.

Adaptive sampling and information theoretic measures enable reliable and efficient mutation testing.

Detects $\geq 90\%$ of the truly important mutation operators by examining $\leq 20\%$ of the available mutants.

Generalizes to medium-sized programs.

Sampling is well-suited for many other software testing applications!
Conclusions

- Mutation testing formulated as an importance sampling (i.e., probabilistic) problem.

- Adaptive sampling and information theoretic measures enable reliable and efficient mutation testing.

- Detects $\geq 90\%$ of the truly important mutation operators by examining $\leq 20\%$ of the available mutants.

- Generalizes to medium-sized programs.

- Sampling is well-suited for many other software testing applications!
Conclusions

- Mutation testing formulated as a importance sampling (i.e., probabilistic) problem.

- Adaptive sampling and information theoretic measures enable reliable and efficient mutation testing.

- Detects ≥ 90% of the truly important mutation operators by examining ≤ 20% of the available mutants.

- Generalizes to medium-sized programs.

- **Sampling is well-suited for many other software testing applications!**
Further Reading


