BURST: Benchmarking Uniform Random Sampling Techniques
Mathieu Acher, Gilles Perrouin, Maxime Cordy

To cite this version:

HAL Id: hal-03897639
https://inria.hal.science/hal-03897639
Submitted on 14 Dec 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
BURST: Benchmarking Uniform Random Sampling Techniques

Mathieu Acher\textsuperscript{a}, Gilles Perrouin\textsuperscript{b}, Maxime Cordy\textsuperscript{c}

\textsuperscript{a}Univ Rennes, CNRS, Inria, IRISA, Institut Universitaire de France (IUF), France
\textsuperscript{b}PReCISE, NaDI, Faculty of Computer Science, University of Namur
\textsuperscript{c}SaT, University of Luxembourg

Abstract

BURST is a benchmarking platform for uniform random sampling (URS) techniques. Given: i) the description of a sampling space provided as a Boolean formula (DIMACS), and ii) a sampling budget (time and strength of uniformity), BURST evaluates ten samplers for scalability and uniformity. BURST measures scalability based on the time required to produce a sample, and uniformity based on the state-of-the-art and proven statistical test Barbarik. BURST is easily extendable to new samplers and offers: i) 128 feature models (for highly-configurable systems), ii) many other models mined from the artificial intelligence/satisfiability solving benchmarks. BURST envisions supporting URS assessment and design across multiple research communities.

Keywords: configurable systems, software product lines, variability model, sampling, SAT, benchmark
## Code metadata

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Code metadata description</th>
<th>Please fill in this column</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Current code version</td>
<td>1.1</td>
</tr>
<tr>
<td>C2</td>
<td>Permanent link to code/repository used for this code version</td>
<td><a href="https://archive.softwareheritage.org/swh:1:dir:c7e0d46e4cfa63c5365073422b5b973140a6d3ab;origin=https://github.com/diverse-project/BURST;visit=swh:1:snp:ab9b0c4b26bf096da2c3f0cf943bb3d3c28f4b9c6b;anchor=swh:1:rev:bceeb56500d25432f220eb4b86da95ff4956417eb/">https://archive.softwareheritage.org/swh:1:dir:c7e0d46e4cfa63c5365073422b5b973140a6d3ab;origin=https://github.com/diverse-project/BURST;visit=swh:1:snp:ab9b0c4b26bf096da2c3f0cf943bb3d3c28f4b9c6b;anchor=swh:1:rev:bceeb56500d25432f220eb4b86da95ff4956417eb/</a></td>
</tr>
<tr>
<td>C3</td>
<td>Permanent link to Reproducible Capsule</td>
<td>Github for command line interface (CLI) use: <a href="https://github.com/diverse-project/BURST">https://github.com/diverse-project/BURST</a> and via Docker at <a href="https://hub.docker.com/repository/docker/macher/usampling">https://hub.docker.com/repository/docker/macher/usampling</a></td>
</tr>
<tr>
<td>C4</td>
<td>Legal Code License</td>
<td>MIT</td>
</tr>
<tr>
<td>C5</td>
<td>Code versioning system used</td>
<td>Git and Github</td>
</tr>
<tr>
<td>C6</td>
<td>Software code languages, tools, and services used</td>
<td>Python, Jupyter notebooks, Bash, SAT samplers written in Python, C++, and .NET, and statistical testing procedure (Barbarik)</td>
</tr>
<tr>
<td>C7</td>
<td>Compilation requirements, operating environments &amp; dependencies</td>
<td>Python &gt; 3, Docker image</td>
</tr>
<tr>
<td>C8</td>
<td>If available Link to developer documentation/manual</td>
<td><a href="https://github.com/diverse-project/BURST/#README.md">https://github.com/diverse-project/BURST/#README.md</a></td>
</tr>
<tr>
<td>C9</td>
<td>Support email for questions</td>
<td><a href="mailto:mathieu.acher@irisa.fr">mathieu.acher@irisa.fr</a></td>
</tr>
</tbody>
</table>

Table 1: Code metadata (mandatory)
Software metadata

<table>
<thead>
<tr>
<th>Nr.</th>
<th>(Executable) software metadata description</th>
<th>Please fill in this column</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Current software version</td>
<td>1.0</td>
</tr>
<tr>
<td>S2</td>
<td>Permanent link to executables of this version</td>
<td><a href="https://github.com/diverse-project/BURST/releases/latest">https://github.com/diverse-project/BURST/releases/latest</a></td>
</tr>
<tr>
<td>S3</td>
<td>Permanent link to Reproducible Capsule</td>
<td><a href="https://github.com/diverse-project/BURST/">https://github.com/diverse-project/BURST/</a></td>
</tr>
<tr>
<td>S4</td>
<td>Legal Software License</td>
<td>MIT</td>
</tr>
<tr>
<td>S5</td>
<td>Computing platforms/Operating Systems</td>
<td>Docker (solution tested on MacOS and Linux)</td>
</tr>
<tr>
<td>S6</td>
<td>Installation requirements &amp; dependencies</td>
<td>Docker (that includes necessary tools, including Python); samplers are embedded into the Docker</td>
</tr>
<tr>
<td>S7</td>
<td>If available, link to user manual - if formally published include a reference to the publication in the reference list</td>
<td><a href="https://github.com/diverse-project/BURST/#README.md">https://github.com/diverse-project/BURST/#README.md</a></td>
</tr>
<tr>
<td>S8</td>
<td>Support email for questions</td>
<td><a href="mailto:mathieu.acher@irisa.fr">mathieu.acher@irisa.fr</a></td>
</tr>
</tbody>
</table>

Table 2: Software metadata (optional)
1. Background & Motivation

This paper presents a platform to assess uniform random sampling (URS) algorithms for highly-configurable systems (HCS). HCS allow customisation of their behaviour though the activation of options or features. These features are organised within Feature Models (FMs) [1], amenable to formal analysis via a translation to propositional logic [2, 3]. HCS customisation comes at the price of the explosion of the number of variants one can derive from combinations of options, i.e., configurations. This explosion leads to the impossibility to assess all configurations. As a result, different sampling strategies for HCS have been devised [4, 5]. Uniform random sampling attributes the same selection probability to each configuration, and therefore makes no hypothesis on its characteristics, allowing an unbiased exploration of the configuration space. URS also supports verification [6] or search-based techniques [7]. To this end, several authors offered URS algorithms, either in the HCS community [8, 9] or in the constraint solving one [10, 11, 12]. Each algorithm exposes trade-offs in terms of scalability and statistical guarantees of uniformity.

Based on our previous experience [13], we designed BURST to explore the following research questions:

RQ1: How scalable are the different URS algorithms?

RQ2: How uniform are the different URS algorithms?

2. Implementation

BURST answers our research questions above by providing: i) a set of command line tools written in Python, ii) a set of 10 URS tools (see Section 5) to perform comparisons, and iii) a set of 128 feature models [14, 15, 16] capturing the configuration spaces of various HCS such Linux kernel versions, command line tools such as busybox, and a model of Jhipster [17] [5], a configurable WebApp generator. These models are provided as Conjunctive Normal Form (CNF) Boolean formulas in the DIMACS format. The whole BURST platform is integrated within a docker image to ease replicability.

The BURST platform is organised as follows:

- **Samplers.** All samplers are in samplers directory (and all utilities/dependencies are also in this folder).

- **Sampling experiments.** The usampling-experiments.py script pilots the scalability study of samplers over different models.
• **Uniformity experiments.** The file `barbarikloop.py` performs uniformity experiments and store results in a CSV file. It is based on the barbarik tool from Kuldeep Meel et al. [13]: https://github.com/meelgroup/barbarik. This version supports uniformity check for all the 10 solvers above and uses SPUR as a reference uniform sampler named SPUR [19] by default. The reference solver can be specified in BURST calls.

• **Models.** A set of various HCS and non-HCS models as introduced above.

### 3. Results

Early experiments with the platform tend to confirm our initial observations [13]: *it is generally difficult to marry scalability (i.e., producing samples fast on large formulas) with uniformity (i.e., ensuring that all configurations have equal selection probabilities).*

However, recent development of the CMS sampler, named CMSgen [20], seem to offer a very encouraging trade-off of producing samples fast with good uniformity properties.

### 4. Related Literature

BURST has been originally presented at the SPLC 2021 tool demonstrations track [21]. This original software publication adds more information about technical details of the platform (metadata) and the possibility to be updated as the platform evolves.

We are not aware other initiatives to benchmark uniform random sampling tools with the following characteristics: i) a diverse and extendable set of samplers and ii) a varied set of CNF formulas to choose from, including feature models of various size and models issued from the SAT/AI communities. However, we would like to point the work of Meel et al. who demonstrate, through the use of Barbarik tool BURST also relies on, that better evaluation of uniformity leads to better samplers, that in turn suggests improvement of benchmarking tools and evaluation metrics [20, 22]. We believe that BURST can contribute to this virtuous circle.

BURST also takes place in the wide initiative to evaluate sampling techniques for HCS [4, 5, 13, 23].

### 5. Illustrative Example

We illustrate the usage of the platform using docker for scalability and uniformity.
5.1. Sampling Performance

The following command performs scalability analysis on the JHipster feature model using KUS [24] as the target sampler:

```bash
docker run macher/usampling:squashed /bin/bash -c 'cd /home/usampling-exp/; echo STARTING; python3 usampling-experiments.py -flas /home/samplingfm/Benchmarks/FeatureModels/FM-3.6.1-refined.cnf --kus; echo END'
```

The current list of supported samplers is as follows.

<table>
<thead>
<tr>
<th>Sampler Name</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIGEN</td>
<td>1</td>
</tr>
<tr>
<td>QUICKSAMPLER</td>
<td>2</td>
</tr>
<tr>
<td>STS</td>
<td>3</td>
</tr>
<tr>
<td>CMS</td>
<td>4</td>
</tr>
<tr>
<td>UNIGEN3</td>
<td>5</td>
</tr>
<tr>
<td>SPUR</td>
<td>6</td>
</tr>
<tr>
<td>SMARCH</td>
<td>7</td>
</tr>
<tr>
<td>UNIGEN2</td>
<td>8</td>
</tr>
<tr>
<td>KUS</td>
<td>9</td>
</tr>
<tr>
<td>DISTAWARE</td>
<td>10</td>
</tr>
</tbody>
</table>

Typical outcomes are:

```
cat usampling-data/experiments-KUS.csv
formula_file,timeout,execution_time_in,dnnf_time,sampling_time,
\rightarrow model_count,counting_time,dnnfparsing_time
/home/samplingfm/Benchmarks/FeatureModels/FM-3.6.1-refined.cnf,
\rightarrow False, 0.1399824619293213, 0.011404275894165039,
\rightarrow 0.0007951259613037109, 26256, 0.0008006095886230469,
\rightarrow 0.0012776851654052734
```

The meaning of columns of the CSV file is as follows: ‘formula_file’: the name of the processed model; ‘timeout’: whether a timeout has been reached or not; ‘execution_time_in’ overall execution time; ‘dnnf_time’: the time required by KUS to compile the corresponding DNNF formula; ‘sampling_time’: time taken to produce the samples; ‘model_count’: the number of solutions in the model; ‘counting_time’: time taken to count solutions; ‘dnnfparsing_time’: time taken to parse the compiled DNNF formula. All times are reported in seconds.
5.2. Uniformity Analysis

Within the Docker image, the following command performs uniformity analysis on the JHipster FM using CMS as the target sampler and SPUR as reference for a sampling budget of 5000 samples:

```
python3 barbarikloop.py --maxSamples 50000 --minSamples 0 --ref
  -sampler 6 --sampler 4 --seed 1 --delta 0.05 --epsilon
  0.3 --eta 0.9 -flas /home/samplingfm/Benchmarks/
  FeatureModels/FM-3.6.1-refined.cnf
```

The content of the generated CSV file should look something like this:

```
cat output/c1f1b9a13035439383912ef57a98535d/Uniform-
  CustomSampler.csv
/home/gilles/FeatureModels/FM-3.6.1-refined.cnf,1.782,...,b'',
  True, FALSE
```

The meaning of columns of the CSV file is as follows: ‘file’: the name of the processed formula; ‘cmd_output’: the output of the command (debugging purposes); ‘err_output’: captures of possible errors (debugging purposes); ‘Uniform’: whether the model is uniform; ‘Timeout’: whether the timeout (in seconds) has been reached.

6. Conclusion

In this paper, we presented BURST, a generic Uniform Random Sampling evaluation platform for a variety of domains. Provided with a SAT formula describing the configuration space and a testing budget (time and strength of uniformity desired), BURST evaluates a variety of uniform random samplers with respect to scalability and uniformity. BURST records results in CSV file to ease further analyses. BURST is conveniently provided in Docker container, integrates 10 random uniform samplers, provides a large choice (128) of feature models and is easily to extensible to new samplers. We hope to make BURST a tool of choice to support the design of new uniform random sampling techniques.

Acknowledgements

The authors would particularly like to thank Kuldeep S. Meel from National University of Singapore, Mate Soos from Zalando Germany and their colleagues for their help setting up and fixing Barbarik as well as the CMS samplers. This research was partly funded by the ANR-17-CE25-0010-01
VaryVary project. Gilles Perrouin is a Research Associate at the FNRS. Maxime Cordy was supported by FNR Luxembourg (grant C19/IS/13566661-BEEHIVE/Cordy).

References


