# Appendix

# Artifact Description: Making Speculative Scheduling Robust to Incomplete Data

## A. Abstract

*This artifact contains the code for computing the sequence of request for an application given its past behavior (together with the corresponding cost as described in Section III), instructions on how to use the code and print the data to reproduce the results from Section V. Section VI uses neuroscience applications that are open source, however the execution logs as well as the input MRI data for each application is not available outside the Vanderbilt University.*

## B. Description

*1) Check-list (artifact meta information): Fill in whatever is applicable with some informal keywords and remove the rest*

- **Algorithm:**
- **Program: Python code**
- **Data set: Synthetic application runs created from the truncated normal and truncated exponential distributions**
- **Run-time environment: the code should run on any environment capable of running python, scipy and numpy**
- **Hardware: any configuration**
- **Execution: the compute_sequence_cost.py script can receive as input either a scipy distribution (between truncnorm and expon) or a file with a list of execution times (either one per line or separated by space)**
- **Output: the output of the script is a csv file containing the optimal cost as well as (Best Fit, Parameters for the best fit, Cost, EML) for the discrete and continuous fit of the input data (when using 10 to 500 in steps of 50 entries for training and all entries when computing the cost)**
- **Experiment workflow: download the code, run compute_sequence_cost.py script on different distributions; obtain output csv files, run the jupyter notebook to print the data, analyze the output figure**
- **Publicly available?: Yes**

*2) How software can be obtained (if available): The code used is available on github at https://github.com/anagainaru in the ReproducibilityInitiative repository (link in footnote)*[1] .

*3) Software dependencies: Python3 is required. The code is using scipy functions for fitting the data to a distribution. The pearson3 and rice have been removed from the list of distributions for which fitting is performed since often the CDF values given by scipy does not correspond with the PDF (Figure* **??***). Our results do not include these distributions. They can be added in the WorkloadFit.py file in the DistInterpolation class (in which case the outliers need to be removed from the results file).*

*The classes to compute the sequence of requests is part of the ScheduleFlow simulator (available for download at: https://github.com/anagainaru/ScheduleFlow).*

*4) Datasets: All datasets are generated during runtime by the script (to follow either truncated normal or truncated exponential distributions).*

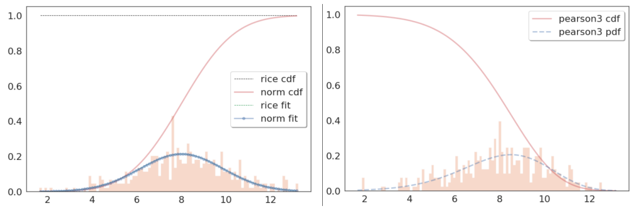

Fig. 1. Behavior of the CDF and PDF for the rice and pearson3 distributions, which best fit the datasets presented.

## C. Installation

```
 git clone \
git@github.com:anagainaru\
/ReproducibilityInitiative.git
```

## D. Experiment workflow

*Download or clone from git the source code.*

*To execute the code using a log file of past executions for an application:*

```
python compute_sequence_cost.py\
{folder/dataset_file}
```

*Dataset file needs to contain a list of execution times*

*To execute the code using synthetic data for the execution times:*

```
python compute_sequence_cost.py\
{distribution}
```

*Supported distributions: [truncnorm, expon]*

*The truncated normal distribution uses $\mu$=8, $\sigma$=2 with limits $[a, b]$=0,20. The truncated exponential distribution uses $\mu$=1, $\sigma$=1.5 with limits $[a, b]$=0,9 These values can be changed inside the compute_sequence_cost.py script.*

*After the successful execution of the script, a csv file will be created (wither dataset_file_cost.csv or distribution_cost.csv. This file holds the results of one experiments per line where each line contains the fields:*

```
Function, Fit, Parameters,\
Cost, Trainset, EML
```

*Function is either Optimal, Discrete or Continuous. The fit column contains values only for the Continuous function and can either be a distribution or a order for the polynomial. The parameters hold the parameters for the best interpolation function, wither the distribution parameters or the $k$-1 parameters for a $k$ degree polynomial function. The Cost represents the cost for the sequence obtain using the continuous or interpolation fits, EML is the relative cost to the cost of the optimal. The trainset defines the number of samples used for computing the sequence (from 10 to 500).*

## E. Evaluation and expected result

*The experiments ran 100 times each for a training set of 50 to 500 elements (in steps of 50).*

[1] https://github.com/anagainaru/ReproducibilityInitiative/tree/master/2019_scala

```
for i in range{1..100};\
python compute_sequence_cost.py truncnorm;\
done
```

*After running the experiments script, the results will be printed out in the corresponding csv file. This jupyter notebook print_sequence_cost.ipynb can be used to plot the figures from Section V.*

*We do not expect the results to be exactly the same as the ones presented in this paper but the observed trend should be similar.*

### *F. Experiment customization*

*There are four possible customizations:*

1) *The truncated normal and exponential parameters can be changed from the compute_sequence_cost.py script by changing the values from the ExponentialRun and TruncNormRun classes*
2) *The maximum degree of the polynomial interpolation can be changed in the compute_sequence_cost.py script when initializing*

   ```
   WorkloadFit.PolyInterpolation(\
   max_order=10)
   ```

3) *New distributions can be added in the compute_sequence_cost.py script by adding a class with the distribution parameter similar to ExponentialRun and TruncNormRun. The new class needs to inherit the DistributionRuns class.*
4) *The distributions used for the distribution interpolation can be changed from the DistInterpolation class in the WorkloadFit file.*