Technical Report

System Analysis and Modeling Group

Hasso-Plattner Institute

University of Potsdam

January 30, 2018

Christian M. Adriano

Sona Ghahremani

Holger Giese

Contents

Introduction

This report details the technical steps taken to train machine learning models and compare decisions make with these models. The report is divided in three parts: training, validating, ranking decisions, computing reward. The steps we followed are depicted in Figure 1.

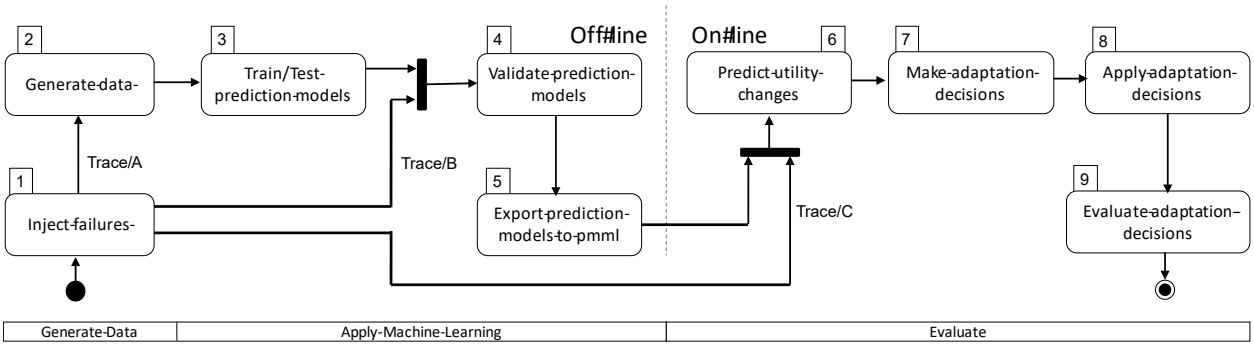

Figure 1 Methodology

The methodology that we followed divided between an off-line training process (performed in R) and an online execution of the trained models, which is performed in Java (utilizes an XML format exported by R). Both online and offline processes depend on injection of real failures. Below we explain each of these activities and provide data that was generated.

1. Inject Failures

We manually inject failures to the employed simulator of mRUBiS. The failures are based on the realistic failure profile models of real world computer systems differing in scale, volatility, and usage in computer systems provided [Gallet2010] to guarantee an unbiased and realistic input. Each failure profile model is described by a distribution for failure group sizes (i.e. number of failures at each cycle) and IAT (i.e. Inter Arrival Time between the failures). Table 1 describes the characteristics of the three failure profile models employed in this study as introduce in [Gallet2010].

Table 1- Characteristics of the Failure Profile Models

Failure Profile Model	Failure Group Size	IAT *(sec)*
Grid5000	*LOGN(1.88 , 1.25)*	*LOGN(-1.39,1.03)*
DEUG	*LOGN(2.15 , 0.70)*	*LOGN(-2.28,1.35)*
LRI	*LOGN(1.32 , 0.77)*	*LOGN(-1.46,1.28)*

We have two types of failure traces: sampled and full. In the sampled failure trace, we only sampled the cycles with 5, 25, and 50 failure group sizes that are randomly injected to components. The full trace is a trace with a certain number of cycles with different failure group sizes generated based on Table 1.

2. Generate Data

Data generation consisted in two steps. First, following the characteristics of the Grid5000 trace introduced in Table 1, we produced large data sets as the ground truth for different analytical utility functions (i.e. linear, saturating, discontinuous, and combined. The second step produced a predicted ranking (i.e. and ordered list of components to be repaired) to be compared with the optimal ranking. The second step utilized different sets of real failures traces (DEUG and LRI in Table 1).

3. Train and Test

The first step is to create Training comprises the machine learning models to predict utility functions We select the regression trees as training model two reasons: it automatically selects the features to be part of the model and it captured discontinuities and non-linearities. Among various options for decision trees we used xgBoostTree with 10-fold Cross Validation.

To validate the results from training, we make different splits of the original dataset:70/30, 80/20, 90/10. Below we present the results of training with these splits. The validation results are presented in the next section.

Table 2- Training (RMSE) and Validation (MAPD) for different configurations

Complexity	Dataset_size	Split	Train_RMSE_MEAN	Train_RMSE_STD	Test_RMSE_MEAN	Test_RMSE_STD	RMSE	R_Squared	MAPD_%
Discontinuous	1k	90_10	0.003	0.000	26.774	16.521	3.350	1.000	0.643
Discontinuous	1k	80_20	0.002	0.000	26.879	11.300	22.850	0.990	1.968
Discontinuous	1k	70_30	0.002	0.000	26.091	14.796	32.464	0.980	2.657
Discontinuous	3k	90_10	0.003	0.000	18.549	6.138	6.823	0.999	0.197
Discontinuous	3k	80_20	0.003	0.000	18.275	5.494	11.043	0.998	0.479
Discontinuous	3k	70_30	0.018	0.003	16.528	7.459	13.210	0.997	1.256
Discontinuous	9k	90_10	0.008	0.001	21.963	9.060	4.923	1.000	0.127
Discontinuous	9k	80_20	0.005	0.000	21.739	11.884	5.393	1.000	0.185
Discontinuous	9k	70_30	0.069	0.007	20.633	9.277	21.539	0.994	0.912
Linear	1k	90_10	0.001	0.000	0.237	0.220	0.048	1.000	0.022
Linear	1k	80_20	0.001	0.000	0.294	0.294	0.106	1.000	0.048
Linear	1k	70_30	0.002	0.000	0.277	0.254	0.068	1.000	0.062
Linear	3k	90_10	0.002	0.000	0.584	0.556	0.066	1.000	0.013
Linear	3k	80_20	0.003	0.001	0.553	0.572	0.237	1.000	0.044
Linear	3k	70_30	0.002	0.000	0.726	0.606	0.342	1.000	0.066
Linear	9k	90_10	0.002	0.000	0.200	0.190	0.202	1.000	0.013
Linear	9k	80_20	0.002	0.000	0.137	0.099	0.300	1.000	0.022
Linear	9k	70_30	0.002	0.000	0.206	0.200	0.326	1.000	0.018
Saturating	1k	90_10	0.003	0.001	9.902	4.861	1.800	1.000	0.180
Saturating	1k	80_20	0.019	0.002	11.844	6.543	3.023	0.999	0.479
Saturating	1k	70_30	0.002	0.000	13.409	6.000	4.439	0.999	0.788
Saturating	3k	90_10	0.004	0.000	8.169	3.216	3.174	1.000	0.152
Saturating	3k	80_20	0.141	0.011	7.500	2.315	3.880	0.999	0.419
Saturating	3k	70_30	0.003	0.000	8.263	1.962	5.353	0.999	0.627
Saturating	9k	90_10	0.008	0.001	9.501	3.616	3.299	1.000	0.113
Saturating	9k	80_20	0.035	0.003	9.124	1.958	4.690	0.999	0.248
Saturating	9k	70_30	0.006	0.000	10.182	2.974	6.286	0.999	0.473

4. Validate Prediction Model

Validation consisted in evaluating the prediction error for the data that was not used during training. Column MAPD_% in Table 2 shows the results of validating the prediction models using 30%, 20%, and 10% of the original data. We can see that even though 90/10 split shows lower MAPD (mean absolute percentage deviation), the difference is very small (less than 1%). To be pessimistic, we decided to use the 70/30 split (larger prediction error).

5. Export Prediction Model to pmml (prediction modeling markup language)

The prediction model is developed in R and exported to an XML format that can later be instantiated and called from Java (see Utility Change Predictor in Figure 2).

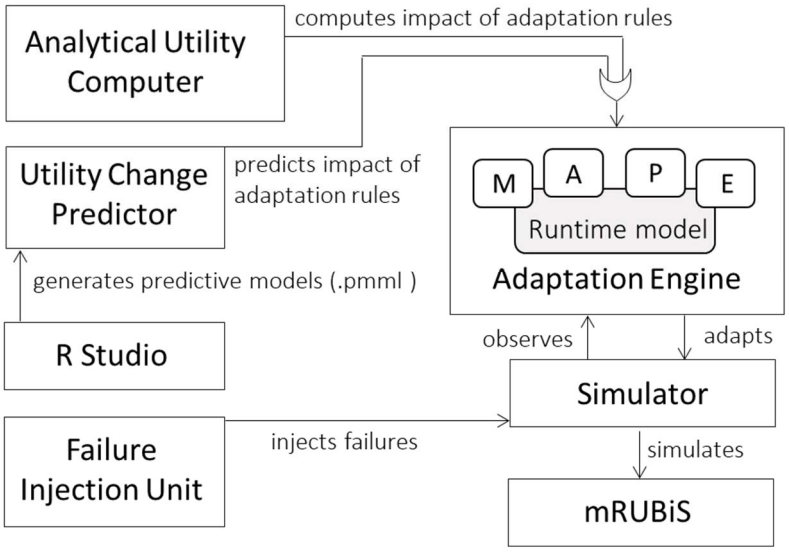

Figure 2 - On-line Prediction Architecture

6. Predict Utility Change

To guarantee a fair base to compare the predicted utility change and the optimal, we injected the same failure trace in two version of the system. One version had the Utility Change Predictor component, while the other had the Analytical Utility computer. Each component provided utility increase values for the pairs component-rule. Based on these values, the adaptation engine selected one rule for each failing component and produced a ranking (sorted lists) of rules to be applied.

7. Make Adaptation Decision

A decision consists of an order list of rules to be applied. The order of these rules in the list follow a decreasing value of utility change. Since distinct prediction models have different prediction errors when computing the value of utility change, we obtain lists with different order. To compare these lists we used similarity metrics, which we investigated their behavior in depth.

7.1. Uncertainty Analysis

7.1.1. Variance of metrics and variance of reward

Below are variance comparisons for models trained with all three dataset sizes (1k, 3k, and 9k), respectively in Table 3, Table 4, and Table 5.

These tables show that system reward does not vary as much the similarity values between predicted and optimal decisions (order lists of lists to be applied). This is positive results because it suggests that the propagation of prediction errors is followed by a reduction in error impact after each step.

Table 3 - Reward versus Similarity Metric Variance (1K dataset trained models)

Model	Reward	Jaccard	Kendall	DCG
Linear	0.0E+00	3.0E-02	1.5E-04	9.1E-03
Saturating	3.1E-05	3.6E-02	4.6E-05	1.7E-02
Discontinuous	3.0E-05	5.2E-02	1.4E-04	1.9E-02
Combined	4.0E-03	6.6E-02	9.1E-06	2.9E-02
Var(Reward) > Var(Metric)	-	**0**	**1**	**0**

Table 4 - Reward versus Similarity Metric Variance (3K dataset trained models)

Model	Reward	Jaccard	Kendall	DCG
Linear	0.0E+00	1.8E-02	1.8E-04	1.4E-02
Saturating	3.1E-05	2.1E-02	1.1E-04	6.7E-03
Discontinuous	3.0E-05	4.2E-02	2.7E-04	1.5E-02
Combined	4.0E-03	4.2E-02	4.1E-04	2.2E-02
Var(Reward) > Var(Metric)	-	**0**	**1**	**0**

Table 5 - Reward versus Similarity Metric Variance (9K dataset trained models)

Model	Reward	Jaccard	Kendall	DCG
Linear	0.0E+00	1.1E-02	5.9E-06	3.3E-03
Saturating	3.3E-11	2.0E-02	1.2E-04	7.0E-03
Discontinuous	2.9E-08	1.5E-02	1.5E-05	8.4E-03
Combined	4.1E-05	3.3E-02	2.3E-04	1.7E-02
Var(Reward) > Var(Metric)	-	**0**	**0**	**0**

7.2. Correlation between reward and similarity metrics

Although we noted that system reward does not vary more than the similarity metrics, variance comparison does not tell us about the direction and intensity of a potential co-variance. We studied that by analyzing the correlation between reward and similarity values.

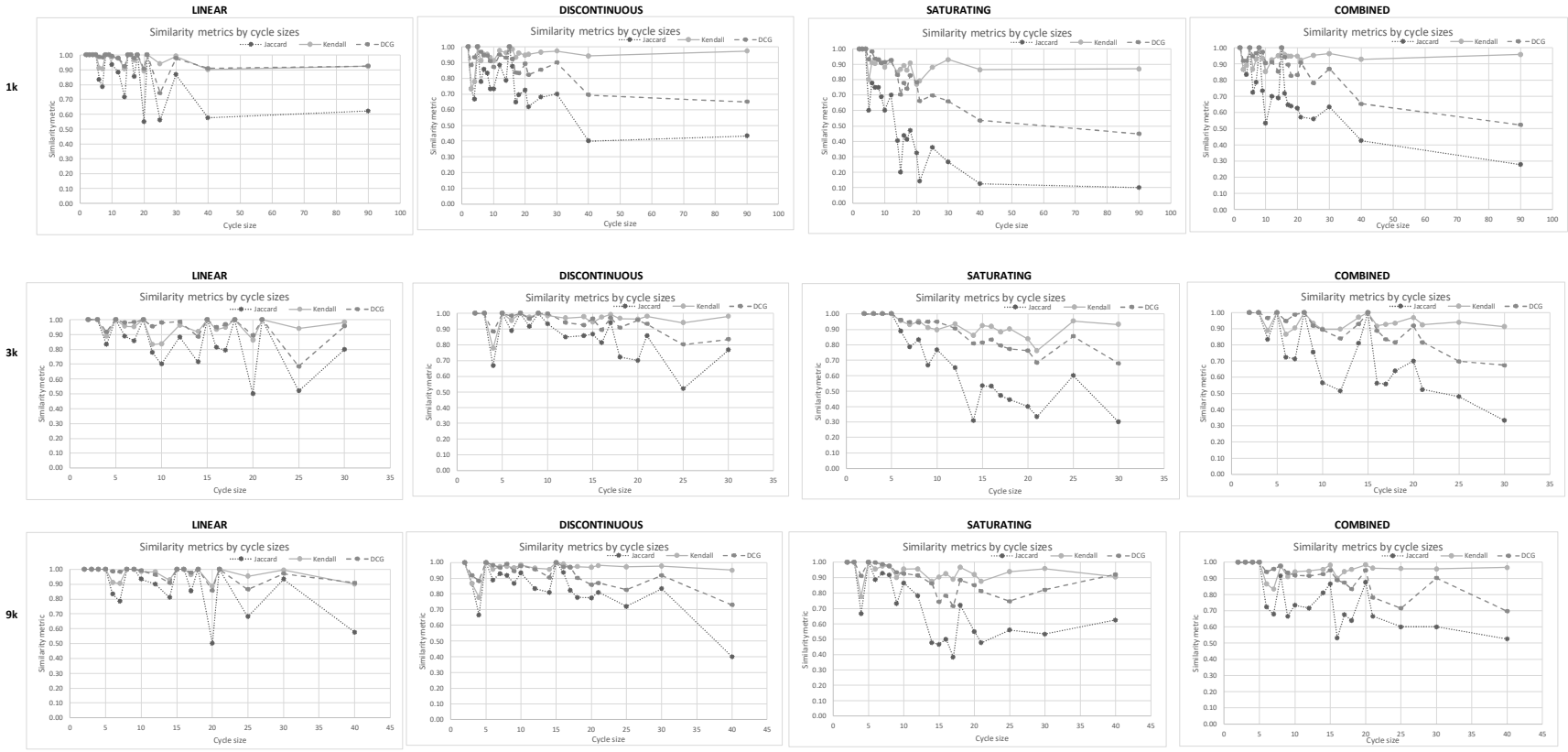

Figure 3 Similarity metrics by Failure Trace Cycles Sizes (up to 90 failures)

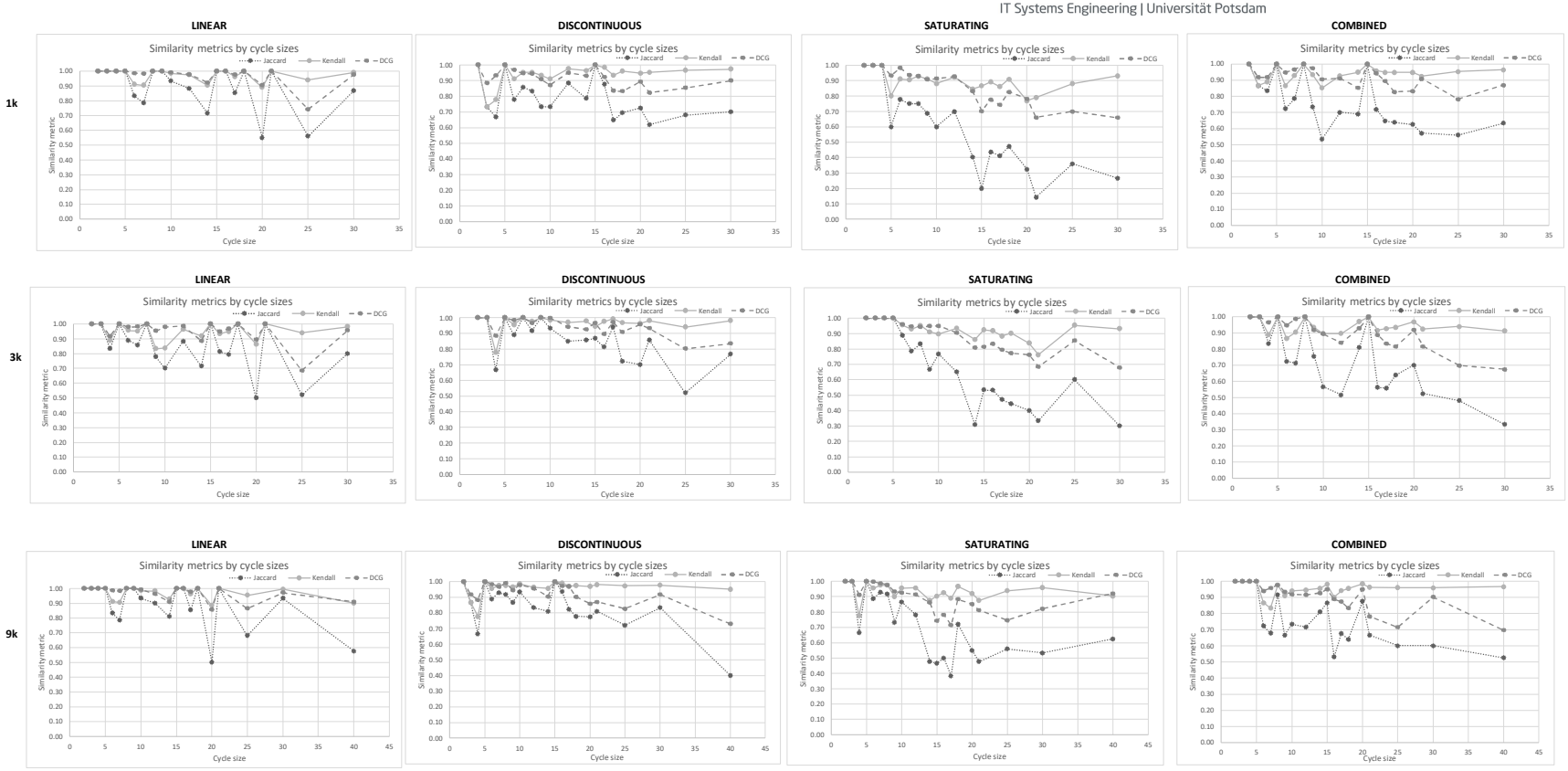

Figure 4 Similarity metrics by Failure Cycle Sizes (showing up to cycles with 40 failures)

Table 6 - Correlations between reward and similarity metric values

Cycle range	Linear	Discontinuous	Saturating	Combined	Metric	Trainind dataset size
All	-0.39	-0.86	-0.82	-0.90	DCG	1
All	-0.36	-0.28	0.25	0.15	Kendall	1
All	-0.57	-0.69	-0.68	-0.76	Jaccard	1
All	-0.30	-0.91	-0.91	-0.88	DCG	3
All	-0.53	-0.79	-0.75	-0.71	Jaccard	3
All	-0.19	-0.02	-0.58	-0.03	Kendall	3
All	-0.46	-0.88	-0.80	-0.86	DCG	9
All	-0.57	-0.80	-0.71	-0.74	Jaccard	9
All	-0.37	0.16	-0.20	0.08	Kendall	9
Up to 30	-0.14	-0.46	0.40	0.21	Kendall	1
Up to 30	-0.45	-0.86	-0.45	-0.65	Jaccard	1
Up to 30	-0.49	-0.93	-0.56	-0.68	DCG	1
Up to 30	-0.44	-0.72	-0.91	-0.86	DCG	3
Up to 30	-0.45	-0.62	-0.87	-0.75	Jaccard	3
Up to 30	0.01	0.07	-0.58	-0.17	Kendall	3
Up to 30	-0.50	-0.54	-0.78	-0.75	DCG	9
Up to 30	-0.36	-0.40	-0.76	-0.67	Jaccard	9
Up to 30	-0.10	0.34	-0.17	-0.01	Kendall	9

- We can look from Figure 3 and Figure 4 that larger cycle sizes imply lower Jaccard an DCG similarity values. This is confirmed by negative correlations between cycle size and the Jaccard and DCG similarity metrics (Table 6). This pattern is present in the results from prediction models trained by all three dataset sizes (1k, 3K, 9K).
- Two variables showed small or no correlation (Table 6) with cycle sizes: the Kendall-tau similarity metric and the Linear model.
- Kendall-tau similarity showed either small correlations or non-significant ones. Looking at the chart, the Kendall-tau correlation remains mostly constant across cycle sizes. This is confirmed by either non-significant correlation values (p-value>0.05) or low correlation values.
- Looking at the correlations for different complexities , the Linear function showed consistently lower correlations than the other functions. The reason is the low prediction error of the linear model, which causes very few ranking mismatches, even across different cycle sizes.
- Even when controlling for higher complexity models (DCG and Saturating) and varying dataset sizes (1k, 3k, 9k), we could not detect any pattern in the correlation between cycle size and the similarity metrics. i.e., increasing dataset sizes not necessarily showed an increase in the

correlation between similarity metric and cycle size. This means that, having models with larger prediction error (i.e., trained by smaller datasets) does not necessarily imply in a proportionate larger number of mismatches.

- One possible explanation is that the prediction error of the cheapest model (trained with 1k dataset) is already too small to cause major variations in the ranking that would reflect in major mismatches for larger cycles. i.e., smaller prediction error (produced by training with the 3k and 9K datasets) does not imply in fewer proportionate mismatches for larger cycles. Although this is a benefit in terms of cost by training with smaller datasets, we do not know if we could reduce the current level of mismatches by reducing the prediction error.