Machine learning-based surrogate models for network reliability analysis

R. E. Stern
University of Illinois at Urbana–Champaign, Urbana, IL, USA

J. Song
Seoul National University, Seoul, Korea

D. B. Work
University of Illinois at Urbana–Champaign, Urbana, Il, USA

This article introduces a framework to perform system reliability analysis in real–time. The framework replaces the computationally intensive step of determining network connectivity for realizations of an infrastructure network subject to component failure probabilities with a faster surrogate model. The surrogate model uses an AdaBoost classifier to predict network connectivity and to significantly reduce the computation time during Monte Carlo simulations. A guided random walk sampling algorithm is proposed to generate training data to improve the classifier performance, and a proof of concept numerical experiment is performed on the California gas pipeline network.

1 INTRODUCTION

In the aftermath of a natural disaster, prompt response is critical to minimize economic damage and the loss of life (Bruneau et al., 2003; Boin and McConnell, 2007). In order to improve post–disaster response times to hazards that impact lifeline networks such as gas, water, and electricity distribution systems, efficient methods are needed to quickly and accurately estimate the network failure probability under a given set of failure probabilities of individual components.

System reliability analysis (SRA) of infrastructure networks encompasses a number of methods to determine the probability that a network will be able to complete its designed function after a disaster event. The methods are used to analyze the resilience of the network with respect to failure under some disaster event (Song and Ok, 2009; Vugrin et al., 2010; Reed et al., 2009; Der Kiureghian and Song, 2008).

Traditional SRA techniques utilize *Monte Carlo simulations* (MCS) of different possible failure scenarios to determine the overall probability of network failure (Papadrakakis et al., 1996). To analyze infrastructure network reliability, current SRA techniques rely on Monte Carlo simulations combined with shortest path algorithms to simulate network damage and test for connectivity between points of interest. By simulating multiple different damaged versions of the same network caused by an event, it is possible to estimate the probability of network failure. Unfortunately, existing SRA techniques are computationally too slow to perform the analysis in real–time as a disaster is unfolding. The objective of this research is to enable real–time SRA through the use of fast, surrogate network connectivity models. The estimates from a surrogate model can be used as independent outcomes of an MCS to estimate the probability of network failure, and effectively decrease the real-time computation requirements while computing the network failure probability with sufficient accuracy.

The main contributions of this paper are as follows. First, a framework for using surrogate models within SRA is proposed. The framework is shown in Figure 1. Second, a specific surrogate model is developed using the *AdaBoost* classification algorithm. The AdaBoost algorithm predicts network connectivity given the state of the network components with low computational overhead. Finally, the framework is evaluated with respect to the computational time and prediction accuracy compared to traditional SRA techniques.

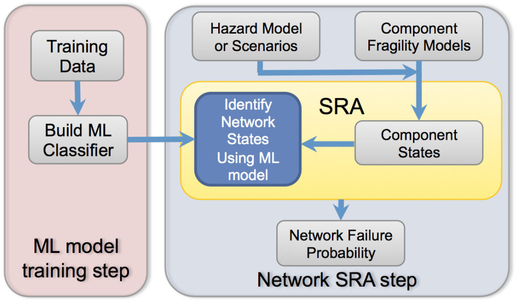
Training Data
Build ML Classifier
ML model training step
Hazard Model or Scenarios
Component Fragility Models
SRA
Identify Network States Using ML model
Component States
Network Failure Probability
Network SRA step

Figure 1: Proposed framework with machine learning-based SRA. A machine learning classifier is trained using training data and then used to classify network states generated by a hazard model. Multiple Monte Carlo simulations are used to estimate on the true network failure probability.

The remainder of the article is organized as follows. In Section 2, a surrogate modeling approaches are reviewed, and the AdaBoost algorithm is introduced. In Section 3, an efficient sampling method termed *guided random walk sampling* is introduced and used to generate training data for AdaBoost. Section 4 details a numerical experiment on the California gas distribution network to highlight the benefits and trade–offs of the proposed framework compared to traditional SRA approaches. Finally, future work and conclusions are presented in Section 5.

2 MACHINE LEARNING-BASED SURROGATE MODELS

2.1 *Surrogate modeling*

Surrogate models provide a similar output to a given model, but are computationally less intensive, and therefore can be used in place of the more accurate model for fast computations. This is typically done when the outcome of interest is computationally complex to obtain, and a simpler surrogate model suffices to generate the desired outcome (Qian et al., 2005). Surrogate models have been applied extensively in fields such as structural optimization, waste water modeling, and supply chain management (Jansson et al., 2003; Meirlaen et al., 2001; Wan et al., 2003).

In this work we apply surrogate modeling to SRA applications. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $n = |\mathcal{V}|$ is the cardinality of the vertex (node) set, and $E \subseteq \{(i,j) : i,j \in \mathcal{V}\}$ we define a network state as $x \in \{0,1\}^n$ where each element x_i represents the status of the corresponding component:

$$x_i = \begin{cases} 0 & \text{if node } i \text{ has failed} \\ 1 & \text{if node } i \text{ remains intact} \end{cases} \tag{1}$$

Similarly, the network status y_{st} with respect to source node s and terminal node t is defined as follows:

$$y_{st} = \begin{cases} 1 & \text{if } s \text{ and } t \text{ are connected} \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

where connectivity is defined as the existence of a path across only intact components. Common shortest path methods such as Dijkstra's algorithm (Dijkstra, 1956) can be used to relate the network x to the network status y_{st} as follows:

$$y_{st} = f(x, s, t) \tag{3}$$

In (3), the operator $f(\cdot,\cdot,\cdot)$ involves running a computationally slow but exact algorithm such as Dijkstra's algorithm to determine the connectivity of the source and terminal nodes under the network state x.

If the failure probability p_{f_i} of each node i is known, then the failure probability of the network with respect to the source and terminal node can also be computed. Typically this involves a Monte Carlo approach which requires running the operator f for each sample, which is the major computational bottleneck that prevents running the calculation in real–time.

To compute a faster estimate of the network failure probability, the exact model f can be replaced with a surrogate model $\tilde{f}$ as:

$$\tilde{y}_{st} = \tilde{f}(x, s, t) \tag{4}$$

where $\tilde{y}_{st}$ denotes the estimated network status predicted by the surrogate model.

For real–time applications, a good surrogate model should be significantly faster than the exact model f, while at the same time maintaining a high accuracy at predicting the network status across a wide range of network states x.

2.2 *AdaBoost as a surrogate network connectivity model*

To efficiently and accurately determine connectivity of s and t, we explore using a machine learning-based classifier known as AdaBoost (Freund and Schapire, 1995). AdaBoost is a classification algorithm that is able to learn complex structure through boosting (Freund and Schapire, 1999), which uses a weighted combination of multiple simple *weak* classifiers to form a more advanced classifier. For example, the AdaBoost classifier $H(\cdot)$ is given by:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{5}$$

which is composed of T weak classifiers h_t with weights α_t. The AdaBoost classifier $H(\cdot)$ is the surrogate model $\tilde{f}(\cdot, s, t)$, where the dependency on the source and terminal node are suppressed for notational simplicity. AdaBoost is a member of the broader family of boosting algorithms. AdaBoost is an appropriate choice for a surrogate model because it has demonstrated capabilities to learn complex structures with limited training data (Schapire et al., 1998).

In AdaBoost the weak classifiers h_t are one dimensional classification rules. AdaBoost sequentially computes each $t \in [0, \cdots, T]$ weak classifier by weighting the m training data points $(x^1, y^1), \cdots, (x^m, y^m)$ according to the weights $D_t \in \mathcal{R}^m$. For the first classifier, all training data receives the same weight (i.e., $D_1(i) = 1/m$). Subsequently, the t^{th} weak classifier is determined by minimizing the weighted misclassification error: $\epsilon_t = \Pr_{D_t}[h_t(x^i) \neq y^i]$.

The weak classifier is given the weight $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$, based on the misclassification error. After the t^{th} weak classifier and weight α_t are determined, the training data is re-weighted to place a larger emphasis on points that were misclassified. This is achieved as follows: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot \exp\left(-\alpha_t y^i h_t(x^i)\right)$, where Z_t is a normalization constant.

3 GENERATING TRAINING DATA FOR ADABOOST

In order to run AdaBoost, a dataset must be generated in order to train the classifier. Due to the massive state space generated by even modest infrastructure networks (i.e., 2^n states), only a small fraction of the state space can be used as training data. Moreover, due to the inherent robustness of infrastructure networks (e.g., through designed redundancies), it is common for many network state realizations to result in a *connected* network status. Consequently, selection of a balanced training dataset is critical for the classifier to perform well over a wide range of network states.

For example, it is important that the number of positive and negative examples in the training set are approximately equal to allow the classifier to be trained on sufficiently many examples of networks that are disconnected, and networks that remain connected after the disaster event (Japkowicz and Stephen, 2002).

Algorithm 1 Guided Random Walk Sampling

```
Pick initial value $n_f^1$
Set $n_f = n_f^1$
for $k = 2 : K$ do
    Calculate: $p_f^{n_f}$
    Draw $s \sim \mathcal{N}(0, \gamma)$ and set $n_f^t = \text{round}(n_f + s)$
    Calculate: $p_f^{n_f^t}$
    if $B_k^t < B_k$ then Accept: Set $n_f = n_f^t, n_f^k = n_f$
    else
        Calculate $\lambda = \min\left\{\frac{B_k}{B_k^t}, 1\right\}$
        Draw $u \sim \mathcal{U}[0, 1]$
        if $u \leq \lambda$ then Accept: Set $n_f = n_f^t$, $n_f^k = n_f$
        else Reject: Set $n_f^k = n_f$
        end if
    end if
    Generate and record network state: $x^k$ s.t. $\left(\sum_{i=1}^n \left(x_i^k == 0\right)\right) = n_f^k$
    Compute and record network status: $y_{st}^k = f(x^k, s, t)$
    Update $p_f^{n_f}$
end for
```

3.1 *Developing guided random walk sampling*

A random-walk sampling technique is proposed to generate training data points, which is based on the Metropolis-Hastings algorithm (Metropolis et al., 1953). We categorize the network states based on the the number of failed nodes, n_f, and track the proportion of network states with n_f failures which correspond to failed networks, $p_f^{n_f}$. Similarly we track the the proportion of failures at the trial point n_f^t given by $p_f^{n_f^t}$. The goal is to sample points such that the failed and intact network states are balanced. This is achieved by sampling points as to minimize $B_k = \left|p_f^{n_f} - 0.5\right|$ by preferentially selecting trial points with $B_k^t = \left|p_f^{n_f^t} - 0.5\right| < B_k$.

In the exploratory phase of the algorithm we randomly generate network states for each n_f, and use a shortest path algorithm (Dijkstra, 1956) to identify whether a path exists between the source and terminal node. Once the baseline estimate on network failure probability for each level of node failure (1 to n nodes failing) is obtained, we continue by randomly selecting a level of node failure and determine whether to include the point in the training data. A trial point n_f^t is generated from the previous point n_f using a Gaussian random step size with zero mean and variance: $\gamma = a \cdot \left(\frac{1}{n} \sum_{n_f=1}^{n} P_f^{n_f}\right)^b$, rounded to the nearest integer, where a and b are tuning parameters. The trial point is accepted or rejected to balance the proportion of failed network states. The algorithm is summarized in Algorithm 1.

3.2 *Node importance factors*

If the network state x is the only feature used for classification, the classifier is not given any explicit information about the network topology. To provide information on the network topology and the relative importance of each node to network survival, two node importance factors are considered. These node importance factors can be used as weights for each feature, to place particular emphasis on important nodes, while de-emphasizing less predictive nodes.

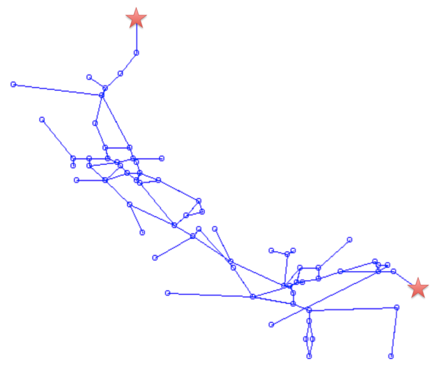

Figure 2: Graphical abstraction of California gas distribution network with source and sink identified with a $\star$.

A commonly used measure of node importance is the *degree centrality* (DC) of the node (Opsahl et al., 2010). Degree centrality measures the total degree (in degree and out degree) as a relative proportion of the highest degree node in the network.

In addition to degree centrality, we use *eigenvector centrality* (EC) to provide the classifier with more information about the network topology. Eigenvector centrality is a measure of the relative influence of a node in the network. The eigenvector centrality of a node is computed as the corresponding component of the eigenvector associated with the greatest eigenvalue of the network adjacency matrix (Chartrand, 1984).

4 NUMERICAL EXAMPLES

4.1 *California gas distribution network*

The California gas distribution network is used for the numerical examples in this study. The topology of the network obtained from Lim et al. (2013) is reproduced in Figure 2. This is an interesting network to study since it provides complexity while remaining sufficiently simple to visually inspect the connectivity. This network consists of 244 components (70 nodes and 87 bi-directional edges). The nodes represent substations and end consumers in the distribution network and the edges are pipelines between these substations. For simplicity of demonstration in this study, only nodes are allowed to fail, although the assumption can be easily relaxed by assigning a node and corresponding failure probability to each pipe segment. This example considers the same source and terminal node for all experiments, and they are identified in Figure 2.

4.2 *Explanation of experiments*

In order to simulate an uncorrelated random failure, we examine the case where all nodes in the network have the same uncorrelated probability of failure. In reality, components may have correlation due to geographic proximity, and the framework can be extended to accommodate this case. For this first simple experiment, we generate a vector of length n of uncorrelated uniform random variables, and compare them to the node failure probability to determine if each node has failed or is still intact for the simulation.

For each case tested, 500 network states are generated and tested for connectivity using the AdaBoost classifier and using Dijkstra's algorithm. The results are used to estimate the true probability of network failure under a surrogate and exact connectivity model. The *coefficient of variation* (COV) on the true probability of network failure $\delta_{\hat{p}_f^n}$ is computed as:

$$\delta_{\hat{p}_f^n} = \frac{\frac{1}{\sqrt{N}} s_q}{\hat{p}_f^n} \tag{6}$$

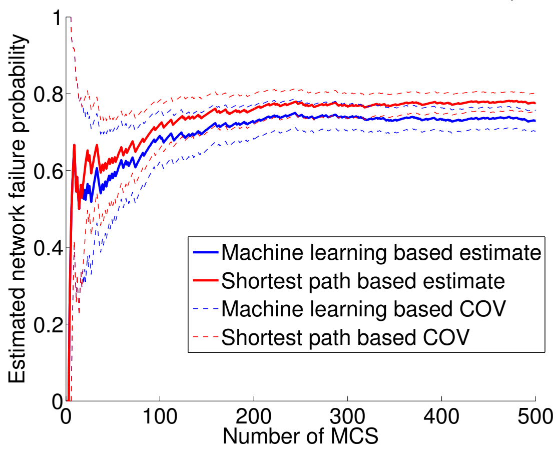

Figure 3: Convergence of MCS estimates on true network failure probability for $p_{f_i} = 0.1$ with 500 simulations.

where N is the total number of MCS, s_q is the sample standard deviation of the network failure probability estimates, and $\hat{p}_f^n$ is the estimated network failure probability. We are then able to compare the accuracy of the estimates, as well as the computation time for each method to evaluate the machine learning-based SRA framework.

We consider individual component failure probabilities p_{f_i} of 0.01, 0.1, 0.2, and 0.5, in order to observe a range of network failure probabilities. Additionally, we use graph theoretic node importance factors that measure to relative centrality of each node to improve the network failure estimate for the case when $p_{f_i} = 0.1$. All experiments are performed using an AdaBoost based model with $T = 200$ weak classifiers.

We also consider a non-uniform component failure probability where component failure probability varies linearly with the distance from the epicenter. In the case considered, component failure probabilities vary from a maximum $p_f^{max} = 0.01$ at the epicenter to a minimum $p_f^{min} = 0.20$ at the furthest point from the epicenter.

4.3 *Results*

Based on the cases outlined above, it is found that if the component failure probability is very high or very low, AdaBoost is able to classify network states with almost zero error. In the case where $p_{f_i} = 0.1$, the machine learning classifier is able to correctly predict each tested network state, as seen in Table 1. The MCS estimate of the network failure probability generated with the proposed machine learning classifier converges with the MCS estimate using the shortest path method as seen in Figure 3, which also shows bounds for the estimate based on the coefficient of variation. Table 1 shows the results of all the MCS conducted. Here, p_{f_i} is the individual component failure probability, ML time and SP time are the time for the machine learning algorithm and shortest-path algorithm to complete all the MCS, respectively, and error is computed as the overall number of incorrect predictions. This table shows that the worst performance is achieved around $p_{f_i} = 0.1$. Furthermore, when a non-uniform component failure probability is considered, Table 1 shows that this has slightly worse overall performance then if each component is assumed to be equally likely to fail.

As illustrated in Table 1, there is a significant time savings when using the machine learning-based method (T_{ML}) for evaluating each Monte Carlo simulation as compared to traditional shortest path methods (T_{SP}) for evaluating the connectivity of two points of interest within the network.

In order to improve the performance of the machine learning-based classifier, the node importance factors introduced previously are used to give the classifier more information about the

p_{f_i}	T_{ML} (s)	T_{SP} (s)	Error (%)
0.01	1.6605	8.8806	0.4
0.1	1.6758	8.9012	4.6
0.2	1.6911	8.9399	3.6
0.5	1.7817	9.0739	0.0
0.01 - 0.2	1.6196	9.0994	8.6

Table 1: Run time and error for 500 MCS at selected component failure probabilities (p_f).

Method	T_{ML} (s)	T_{SP} (s)	Error (%)
Baseline	1.6758	8.9012	4.6
DC	4.5024	10.3058	2.0
EC	4.2358	9.7206	4.2

Table 2: Influence of node importance factors for 500 MCS with $p_f = 0.1$, which is the worst performing classifier (Table 1).

network topology. Based on the results highlighted in Table 2, supplying topological information about the graph to weight each feature has the effect of slightly decreasing the error when degree centrality is used, while causing the computation time to increase. When information about the connectedness of the graph is provided to the machine learning algorithm during training, there is a significant reduction in error as seen in Table 2. Providing topological information to the classifier allows it to classify more accurately.

5 CONCLUSION AND FUTURE WORK

A framework to perform system reliability analysis in real–time using machine-learning-based surrogate models was introduced. AdaBoost was proposed as a method to construct surrogate model for classification of network states. In order to obtain an adequate training sample to developed the classifier a guided random walk sampling algorithm was proposed. The framework was demonstrated on the California gas distribution network. All functions and scripts to reproduce these results can be found at (Stern, 2014).

The current classifier is trained of a single source-destination node pair, and thus works when the specific nodes of interest are known in advance. For real–time SRA applications, having a general model that can classify the connectivity of any two nodes within the network will broaden the applicability of the approach. Future work will focus on constructing a general-purpose classifier for all nodes and all pairs. Further research will be devoted to training the surrogate model for cases with spatially-correlated component failure probability. This advanced classifier will be tested on a variety of infrastructure networks for validation.

ACKNOWLEDGMENTS

The authors would like to thank the US National Science Foundation for funding under grant number CMMI 1031318. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. The second author, Junho Song would like to acknowledge support by a grant entitled Development of cutting edge technologies for the multi-faceted representation of design earthquake ground motions based on analyses of acceleration records [NEMA-NH-2013-71] provided by the Natural Hazard Mitigation Research Group, National Emergency Management Agency of Korea.

REFERENCES

Boin, A. and A. McConnell (2007). Preparing for critical infrastructure breakdowns: The limits of crisis management and the need for resilience. *Journal of Contingencies and Crisis Management 15*, 50–59.

Bruneau, M., S. E. Chang, R. T. Eguchi, G. C. Lee, T. D. O'Rourke, A. Reinhorn, M. Shinozuka, K. Tierney, W. A. Wallace, and D. von Winterfeldt (2003). A framework to quantitatively assess and enhance the seismic resilience of communities. *Earthquake Spectra 19*, 733–752.

Chartrand, G. (1984). *Introductory Graph Theory*. Dover Publications.

Der Kiureghian, A. and J. Song (2008). Multi-scale reliability analysis and updating of complex systems by use of linear programming. *Reliability Engineering & System Safety 93*, 288–297.

Dijkstra, E. (1956). A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 269–271.

Freund, Y. and R. E. Schapire (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Science 55*, 119–139.

Freund, Y. and R. E. Schapire (1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence 14(5)*, 771–780.

Jansson, T., L. Nilsson, and M. Redhe (2003). Using surrogate models and response surfaces in structural optimization with application to crashworthiness design and sheet metal forming. *Structural and Multidisciplinary Optimization 25*, 129–140.

Japkowicz, N. and S. Stephen (2002). The class imbalance problem: a systematic study. *Intelligent Data Analysis 6*, 429–450.

Lim, H., J. Song, and N. Kurtz (2013). Seismic reliability assessment of lifeline networks using clustering-based multi-scale approach. *Earthquake Engineering and Structural Dynamics*.

Meirlaen, J., B. Huyghebaert, F. Sforzi, L. Benedetti, and P. Vanrolleghem (2001). Fast, simultaneous simulation of the integrated urban wastewater system using mechanistic surrogate models. *Water Science and Technology 43*, 301–307.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics 21*, 1087–1092.

Opsahl, T., F. Agneessens, and J. Skvoretz (2010). Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks 32*, 245–251.

Papadrakakis, M., V. Papadopoulos, and N. D. Lagaros (1996). Structural reliability analysis of elastic-plastic structures using neural networks and monte carlo simulations. *Computer methods in applied mechanics and engineering 136*, 145–163.

Qian, Z., C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. F. J. Wu (2005). Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design 128*, 668–677.

Reed, D., K. Kapur, and R. Christie (2009). Methodology for assessing the resilience of networked infrastructure. *IEEE Systems Journal 3*, 174–180.

Schapire, R. E., Y. Freund, P. Bartlett, and W. S. Lee (1998). *The Annals of Statistics 26*, 1651–1686.

Song, J. and S.-Y. Ok (2009). Multi-scale system reliability analysis of lifeline networks under earthquake hazards. *Earthquake Engineering and Structural Dynamics 39*, 259–279.

Stern, R. (2014). https://github.com/raphaelestern/systemreliability.

Vugrin, E. D., D. E. Warren, M. A. Ehlen, and R. C. Camphouse (2010). A framework for assessing the resilience of infrastructure and economic systems. *Sustainable and Resilient Critical Infrastructure Systems*, 77–116.

Wan, X., J. F. Pekny, and G. V. Reklaitis (2003). Simulation-based optimization with surrogate modelsapplication to supply chain management. *Computers & Chemical Engineering*.