12-2023

Reliability Modeling and Improvement of Critical Infrastructures: Theory, Simulation, and Computational Methods

José Carlos Hernández Azucena
University of Arkansas-Fayetteville

Reliability Modeling and Improvement of Critical Infrastructures: Theory, Simulation, and Computational Methods

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering, with a concentration in Industrial Engineering

by

José Carlos Hernández Azucena
Escuela Superior de Economía y Negocios
Bachelor of Science in Business Engineering, 2014
University of Arkansas
Master of Science in Industrial Engineering, 2022

December 2023
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Haitao Liao, Ph.D.
Dissertation Director

Edward Pohl, Ph.D.
Committee Member

Om Prakash Yadav, Ph.D.
Committee Member

Shengfan Zhang, Ph.D.
Committee Member

Kelly Sullivan, Ph.D.
Committee Member

Abstract

This dissertation presents a framework for developing data-driven tools to model and improve the performance of Interconnected Critical Infrastructures (ICIs) in multiple contexts. The importance of ICIs for daily human activities and the large volumes of data in continuous generation in modern industries grant relevance to research efforts in this direction.

Chapter 2 focuses on the impact of disruptions in Multimodal Transportation Networks, which is explored from an application perspective. The outlined research directions propose exploring the combination of simulation for decision-making with data-driven optimization paradigms to create tools that may provide stakeholders with optimal policies for a wide array of scenarios and conditions. The flexibility of the developed simulation models, in combination with cutting-edge technologies, such as Deep Reinforcement Learning (DRL), sets the foundation for promising research efforts on the performance, analysis, and optimization of Inland Waterway Transportation Systems.

Chapter 3 explores data-driven models for condition monitoring and prognostics, with a focus on using Deep Learning (DL) to predict the Remaining Useful Life of turbofan engines based on sequential sensor measurements. A myriad of approaches exist for this type of problems, and the main contribution for future efforts might be centered around combining this type of data-driven methods with simulation tools and computational methods in the context of network resilience optimization.

Chapter 4 revolves around developing data-driven methods for estimating all-terminal reliability of networks with arbitrary structures and outlines research directions for data-driven surrogate models. Furthermore, the use of DRL for network design optimization and maximizing all-terminal network reliability is presented. This poses a promising research venue that has been extended to network reliability problems involving dynamic decision-making on allocating new resources, maintaining and/or improving the edges already in the network, or repairing failed edges due to aging.

The outlined research presents various data-driven tools developed to collaborate in the

context of modeling and improvement for Critical Infrastructures. Multiple research venues have been intertwined by combining various paradigms and methods to achieve this goal. The final product is a line of research focused on reliability estimation, design optimization, and prognostics and health management for ICIs, by combining computational methods and theory.

Acknowledgements

The author expresses their gratitude to the funding from the U.S. National Science Foundation (NSF) under the grants: OIA-2119691, OIA-1946391, and CMMI–1745353. The findings and opinions expressed in this dissertation are those of the author only and do not necessarily reflect the views of the sponsors.

Dedication

To my **Lia**, my Partner in Crime and wife.

Love of my life.

My best friend and all-mighty support.

To my brother, **Dani**; my mom, **Juanita**; and my dad, **Carlos**.

I would not be me without you by my side. Thanks for always being there for me.

To my *abuelita* and *abuelito*, **María Cruz** and **José Artemio**,

who planted the seeds of the desire for knowledge and education.

To the *tías*, *tíos*, *primos* y *primas* and all the *familia* always cheering for us.

Thanks to the most patient and supportive advisor, **Dr. Haitao Liao**.

Thanks to my role model, **Dr. César Ruiz**.

Thanks to my mentor, **Dr. Carlos Carcach**.

Thanks to all the friends, colleagues, support staff, and professors,

benefactors and blessings in human form,

who have been part of this adventure.

Contents

List of Figures

List of Tables

Published Works in this Dissertation

Chapter 2, Section 2.1: J. C. H. Azucena, B. Alkhaleel, H. T. Liao, and H. Nachtmann, "Hybrid simulation to support interdependence modeling of a multimodal transportation network," *Simulation Modelling Practice and Theory*, vol. 107, p. 102 237, 2021

Chapter 3, Section 3.1: J. C. H. Azucena and H. T. Liao, "Prognostic using dual-stage attention-based re-current neural networks," in *Proceedings of the 11th International Conference on Mathematical Methods in Reliability (MMR)*, Hong Kong, Jun. 2019

Chapter 4, Section 4.1: J. C. H. Azucena, H. Wells, H. T. Liao, K. Sullivan, and E. A. Pohl, "Applying deep reinforcement learning to improve the reliability of an infrastructure network," in *Proceedings of the 60th European Safety, Reliability & Data Association (ESReDA) Seminar*, ser. Advances in Modelling to Improve Network Resilience, Grenoble, France, May 2022, pp. 46–55

Chapter 4, Section 4.2: J. C. H. Azucena, F. Hashemian, H. T. Liao, and E. A. Pohl, "Applying machine learning to improve all-terminal network reliability," in *Proceedings of the 69th Annual Reliability and Mantainability Symposyum (RAMS)*, Orlando, FL, Jan. 2023

1. Introduction

1.1 Overview

Interconnected Critical Infrastructures (ICIs) are essential for daily human activities in modern societies worldwide. ICIs can take the form of communication, water distribution, power, and transportation networks. Maintaining functions as defined by service requirements is potentially crucial for various stakeholders, including multiple industries, governmental agencies, non-profit organizations, and the general public. The unavailability of these critical services and any loss of capacity in complying with the service requirements are failures. The network structures mentioned above are prone to failure due to natural degradation, disruptions due to extreme operating conditions, and adverse events. The wear and tear of normal operating conditions can eventually lead to failure. Adverse events and extreme operating conditions can also lead to failures. Therefore, quantifying the reliability of such infrastructure networks becomes essential to guarantee proper functioning. This is the first thread of research explored in the present work.

In parallel, expanding the thread of natural degradation into condition monitoring in a modern context, continuous monitoring of operational conditions and system status is possible using multiple types of sensors simultaneously. Said sensors may measure the intended quantities with varying noise levels, accuracy, and relevance for predictions. The continuous measurement can be framed as a multi-dimensional heterogeneous array. Therefore, the redundancy of sensors provides a multivariate time series that can be used to create prognostics and monitor the health of a system of interest. Creating estimation models for such situations becomes a crucial task in condition monitoring. This is the second thread of research explored in the present work.

Quantifying the resilience of ICIs becomes crucial in settings where decision-making is related to guaranteeing that the essential elements in the network remain communicated for determined time intervals. Using the probability that the elements in the network remain connected as a measure of resilience, grants relevance to methods estimating and maximizing the All-Terminal Reliability of the ICIs network. Data-driven methods for both estimating

and maximizing this relevant metric under varying and dynamic operational conditions and decisions become crucial in modern industrial processes in data-rich environments. This is the third thread of research explored in the present work.

From an application perspective, the impact of disruptions in Multimodal Transportation Networks is explored in Chapter 2. Section 1.2 contextualizes this problem and discusses the explored directions. For condition monitoring and prognostics, Chapter 3 explores a data-driven model using Deep Learning to generate predictions for Remaining Useful Life (RUL) for multiple sensors under various combinations of operating conditions. Section 1.3 discusses an introduction and future research directions. Methods for data-driven estimation of all-terminal network reliability for arbitrary graphs are explored in the work presented in Chapter 4. A brief introduction and discussion of the explored methods are presented in this Chapter under Section 1.4.

The research threads mentioned above connect as specific efforts to achieve a larger goal: create a data-driven framework for reliability estimation, design optimization, and prognostics and health management for ICIs. Multiple methods and paradigms will be combined to achieve this goal. The main contribution is making these various tools collaborate in a data-driven fashion, combining computational methods and theory.

1.2 Simulation of Inland Waterway Transportation Networks

The work presented in Chapter 2 corresponds to the journal article in [1], which is an extension of the conference article in [5]. The main goal of this work is to create a tool capable of simulating the flow of barges in the McClellan-Kerr Arkansas Navigation System (MKARNS) while emulating the seasonal and spatially correlated operation conditions of the Inland Waterway Transportation (IWT) network, such as the water levels along the river system. This will allow quantifying the impact of disruptions on this multimodal network. The primary motivation is how critical the U.S. transportation network is for national security and economic competitiveness. As part of this multimodal network, the

inland waterways mobilize a good proportion of farm exports and petroleum products. The IWT network is exposed to disruptions from natural events such as droughts and floods. A Spatio-Temporal Bayesian model is applied to capture the seasonality and correlation of the water-level measurements across the system.

As potential extensions, two main propositions are available at the moment. First, it becomes apparent that it is necessary to compare the effects of different rerouting policies for barges facing disruptions. The work in [1] did not count with rerouting policies. Developing and incorporating simple routes for rerouting was carried out for the simulation model from specific locks to designated rerouting points and changing modality to truck transportation. Additional development of the simulation model was explored to include logic related to assigning barges to different towing vessels in the traffic generation subroutines, assigning commodities to said barges, governing the distribution of commodities along the different routes with controllable configurations, and including lock queueing and random service times in the locks system logic. All of these developments were carried out while maintaining an emphasis on the efficiency of the use of computational resources to the point that, while having a richer and more complex logic than the model presented in [1], the current state of the development is several times faster. However, in terms of the current state's multimodal capabilities, including land transportation using railways is a necessary addition as it would be a more realistic alternative to truck transportation, as it is usually the most economical option due to economies of scale. Different alternatives for rerouting and the impacts of said policies could be explored to further extend this work.

The second extension is related to data-driven decision-making using Machine Learning models, such as Deep Reinforcement Learning (DRL) agents. Using the simulation environment to sample rewards for the agents under different action sequences and condition operation scenarios, a DRL model could be trained to optimize selected network metrics, such as maximizing satisfied demand, minimizing operation costs, or maximizing system availability. For example, the available actions could include the inspection, maintenance,

repair, or replacement of the system's different lock and dam elements to maximize the system availability while restraining the cost of these actions within a budget for every time step. As the selected sequence of actions also affects the flow of commodities through the system, a balance between scheduled condition monitoring and maintenance operations and regular operation of the system must be achieved. This was explored in the conference articles [6] and [7]. For the latter, the simulation environment has been updated to center the tool's focus on the flow of barges and use them as the smallest transportation unit. Locks' queues and lockage service dynamics are also included in the updated model. The scheduling and quantities of towboats traveling from each source port to selected destinations also change to become configurable by any user. The updated simulation and the DRL agents lead to data-driven decision-making for maintenance scheduling optimization. Further developments in this direction will be aimed at quantifying the effects of scheduling surplus working crews during seasonally affected high traffic or high disruption risk time intervals. As the simulation tool developed is flexible and can be parameterized to represent different scenarios, economic studies of the commodity flow in the region of interest are also potential extensions. The current development and immediate improvements to the codebase are expected to produce results in this direction to support decision-making for transportation operations.

1.3 Prognostics and Health Management

The work presented in Chapter 3 corresponds to the conference article in [2]. This research effort aims to develop a data-driven model to estimate the RUL of multiple turbine units while monitoring sensor information and interpreting the sensor measurements as a multivariate time series. Considering that a system might transition through different stages of degradation before failure, condition monitoring and observing the system behavior to gauge the likelihood of said transitions becomes relevant for applications where the system availability is crucial, such as in ICIs. Together with the ideas in Sections 1.4 and 1.2, a

data-driven condition monitoring framework could help guide a DRL agent for decisions related to maintenance, repair, and replacement of infrastructure components.

Data-driven models tend to become black-box models, and the interpretation of the reasoning behind the predictions is usually obscured. While investing efforts into improving performance and achieving robustness in the predictions is worthwhile, creating more straightforward and interpretable models might provide significant advantages on an alternative track. Work such as [8] challenges the notion of a trade-off between simplicity and explainability vs. predictive performance. Sparse and interpretable models like the one presented in [9] could lead to condition monitoring models with good predictive power and interpretable decision rules. Using a similar framework for multivariate time series could be a potential research direction.

1.4 All-Terminal Reliability

The work presented in Chapter 4 corresponds to the conference articles in [3] and [4]. Both pieces of research are focused on All-terminal Network Reliability. The former uses an automated procedure to generate reliability polynomials and evaluate multiple configurations of network topologies to train a DRL agent to maximize the Network Reliability subject to a budget constraint. The latter uses a Stochastic Variational Inference framework to train a Deep Neural Network as a surrogate model for estimating the all-terminal reliability of arbitrary graphs. Then, with this surrogate model, train a DRL agent to maximize the reliability measure. Both approaches rely on DRL agents to optimize the network design. Still, the methods to evaluate the all-terminal reliability differ, and the attempts to gain speed-ups in computation vary. As part of the explored extensions, there is a focus on extending the decision space by including maintenance, improvement, repair, and replacement activities into the agent's action space. Furthermore, there is an emphasis on aging components following individually parameterized distributions and evolving over specified mission cycles. This grants broader applicability for this problem in ICI contexts, such as the one mentioned in

Section 1.2. In this line of research, modifications to the current problem statement are also under exploration, such as the one explored in the work envisioned in [10]. Here, instead of maximizing the static all-terminal reliability of a given network configuration, the dynamic evolution of the network over multiple mission periods is considered. A heavier emphasis is placed on the efficiency of the surrogate models developed as the network configurations constantly fluctuate. Furthermore, the DRL agents need to consider the changes in the network states due to simulation dynamics and all the potential actions that could improve the network reliability. Alternatively, exploring novel data-driven optimization schemes beyond DRL could be an alternative research venue. The potential of using frameworks as novel and in the vanguard of research, such as Quantum Computing, has become a personal interest of this author in recent years. In particular, methodologies such as the work in [11] present ideas to combine Variational Inference with groundbreaking technologies such as Quantum Computation and simulation of quantum systems. Efforts in this direction could lead to fruition as these novel methods are being tested in problems closely related to network design contexts. Furthermore, as these technologies come to maturity, the design of algorithms to implement them to optimize all-terminal network reliability might become valuable. Preliminary explorations of these ideas are presented at the end of the chapter. The outlook for future developments in this direction is optimistic as there is a large variety of potential approaches both in the definition of the problems and the methods to combine and develop.

1.5 Conclusions and Future Research Directions

This author is optimistic in that following the presented research threads will lead to interwoven patterns that give rise to fertile areas and ideas to be explored: further developments in the field of systems reliability and simulation of complex systems, combining vanguard technological such as quantum computing with the pragmatically solid results of deep learning and machine learning. Finally, Chapter 5 remarks on the pursued development directions. The main contribution is making these various developed tools collaborate in a data-driven

fashion. Multiple methods and paradigms have been combined to achieve a unified goal: the coalescence of the different research venues to create a data-driven framework for reliability estimation, design optimization, and prognostics and health management for Interconnected Critical Infrastructures, combining computational methods and theory.

2. Simulation of Inland Waterway Transportation Networks

2.1 Hybrid simulation to support interdependence modeling of a multimodal transportation network

2.1.1 Introduction

2.1.1.1 Background and motivation

The physical distribution infrastructure is critical to national security, economic well-being, global competitiveness, and quality of life in the United States (U.S.) [12]. The distribution infrastructure, referred to as the transportation network, includes but is not limited to the interconnected network of ports, inland waterways, highways, and railroads. The U.S. transportation network comprises almost 4 million miles (6.43 million kilometers) of public roads and highways, more than 360,000 interstate trucking companies and 20 million trucks for business, and 1,900 seaports and 1,700 inland river terminals on 11,000 miles of inland waterways carrying grain, chemicals, petroleum products, and import and export goods [13]–[15].

Many industries rely on the U.S. transportation network; thus, the economic impacts of disruptions affecting the network are expected to be substantial. Such interruptions can cause a cascading effect that can become widespread due to the spatial and temporal distributions of commodity flows [16]. Even without large-scale disruptions, the Federal Highway Administration (FHWA) estimated the trucking industry losses to be around $8 billion a year due to highway congestion [16], [17]. Such losses are expected to increase in the future due to forecasted increases in the U.S. domestic freight tonnages by approximately 50% in the next fifteen years [15], [18], [19]. In addition to highway network impacts, railways are expected to experience more significant congestions and breakdowns due to increased demand for Class I railroads [20]. The U.S. Maritime Administration, an agency of the U.S. Department of Transportation, has called for investment in the domestic waterways for freight movement [21], recognizing the need to reduce road and rail congestion. The increased use of 25,000 miles of inland waterway freight transport could result in less congestion on

U.S. roads and a reduction in the risk of road and rail transport accidents and possibly even reduce emissions of air pollution [16]. Barge transport is frequently cheaper than rail and truck alternatives, and there are many products which are too large for other transport methods. In 2017, the U.S. inland waterways were used to transport approximately 20% of America's coal, 22% of U.S. petroleum products, and 60% of farm exports between 38 states summing up the annual weight transported to around 630 million tons [14], [15].

Although general freight movements via the inland waterways are expected to increase in the upcoming years due to economic and logistic drivers, research investigating the impacts of disruptions on waterway operations, multimodal commodity flow, and economic analysis are limited. Indeed, one reason for the limited number of studies may be the lack of tools that could facilitate research in this area by providing data-driven models. There is an urgent need to protect and coordinate U.S. multimodal transportation infrastructure to support strong economic growth and national security. Inland waterways and road and rail transport have a significant impact on various business operations in the U.S., especially in middle America along hundreds of miles of the Mississippi River. However, inland water transportation is significantly affected by weather, current and future waterway conditions, and operation strategies at different locks, dams, and ports [22]. For example, in the case of flooding or drought, inland water transport will be constrained by the water levels of dams and ports, and the effects will propagate downstream. In response to such emergencies, goods on cargo vessels need to be offloaded and re-routed through the available ground transportation system. Since these infrastructures are managed by different governing agencies [14], multiple stakeholders need to understand the characteristics of these Interdependent Critical Infrastructures (ICIs), such as ports, lock and dam systems, and ground transportation that cross administrative boundaries. Considering the large potential impact and lack of actual data availability, this research will generate simulated data to represent a multimodal transportation systems.

2.1.1.2 Related work

There are various simulation models discussed in the literature that focus on inland waterway operations with different problems to solve and goals to achieve; however, the literature that studies the simulation of traffic flow in inland waterways can be broadly divided into three categories: (1) literature that focuses primarily on lock operation simulation models to analyze lock delays and tow travel times and optimize waterway investment projects and other aspects of locks operations [23]–[25], (2) literature that discusses barge dispatching and vessel assignment scheduling problems in inland waterways [26]–[28], and (3) literature with a broader scope that considers ICI resilience, disruption management strategies and economic studies with a focus on inland waterways as the leading network of commodity flow [29]–[31].

There are multiple simulation models that were developed to analyze the different aspects of lock operations [23]; the earliest model can be found in 1969 [32] which was developed jointly by Resources for the Future Inc. and Pennsylvania State University. The model (referred to as RFF by [24]) was programmed to simulate the movement of shallow draft barge tows through a linear waterway having up to ten locks with one or two chambers, twenty ports, and ten delay points (channel restrictions). Model inputs include tow characteristics, tow itineraries, and attributes of the waterway system; model outputs include a variety of statistics including tows processed, transit and delay times, queue lengths, and tonnages [24], [32]. [24] developed an enhanced two-part model of the RFF; the first part processes information concerning commodity flows and waterway fleet characteristics to derive a list of tows that will move on the Illinois waterway and upper Mississippi River, where the second part of the model simulates the movement of these tows through the ports, locks, pools, and channel delay areas that comprise the waterway system. Moreover, [33] developed a waterway simulation model that estimates tow delays at a series of locks, tow travel time along waterways, and the means and variances of interarrival and interdeparture times at each lock; and was validated by comparing it to the well-established M/G/1 queue system.

Additionally, [34] applied the simultaneous perturbation stochastic approximation (SPSA) technique with simulation models to optimize the size and timing of investment projects in a waterway system with five locks. The discussed lock operation simulation studies rely on site-specific simulation models without network generality and comprehensive functionality, making it difficult to extend the developed simulation models to any other waterway networks [23]. To address the lack of generalized modeling, [23] developed a general waterway simulation model that is independent of network geometry to evaluate a waterway system over a multi-year planning horizon. Recently, [25] developed a robust Monte Carlo simulation-based method to assess port capacity and expansion plans. Their method helps to identify optimal resource configurations for expected throughputs.

A second category of waterway simulation studies uses simulation as an optimization tool to solve the barge dispatching and assignment problem, which is generally solved using classical optimization approaches [28]. [26] developed a Barge Operations Systems Simulator (BOSS) to assist in the task of fleet sizing when transporting refuse from New York City to Fresh Kills Landfill on Staten Island. Moreover, [27] developed a discrete event simulation model as a decision support tool for logistical management within a marine-based distribution system to determine fleet size and resource allocation to meet delivery requirements in a timely manner. [28] presented a simulation-based scheduling system designed to assist in barge dispatching and boat assignment problems for inland waterways.

Regarding the third category of literature, many studies have investigated the modeling and simulation of ICIs through empirical approaches, agent-based approaches, network-based approaches, and other approaches [31], [35]. However, only few articles addressed simulation of inland waterways transportation [15]. [36] developed an iterative technique between optimization and simulation models to check the feasibility of barge routings suggested by the optimization model based on a sampled dataset. Biles et al. [37] presented a simulation model of traffic flow in inland waterways with the incorporation of the Geographic Information System (GIS) to improve vessel scheduling. Recently, [29] used a Monte Carlo simulation

model to estimate the potential economic impacts of inland waterway disruptions. Moreover, several studies investigated the economic impact of disruptions on different transportation systems [38]–[40]. Furthermore, [30] created a simulation architecture of inland waterways based on Markov Decision Process (MDP) and climate projections under uncertainty. [15] developed an agent-based multimodal simulation tool, which is the initial study of the model presented in this article. With the exception of [15], all the studies in this third category do not consider predicting disruptions in advance based on statistical models, simulating multi-modal transportation, modeling the interdependency between waterway transportation and ground transportation, and allowing different scenario generation by controlling lock and dam systems.

2.1.1.3 Overview and research contributions

The ultimate goal of this work is to provide research methods and application opportunities from which the U.S. economic growth and homeland security can significantly benefit. A thorough understanding of multimodal freight movement processes that combine different data sources can provide open-sourced, multi-regional, multi-industry, data-driven statistical models, and simulation tools to benefit decision-makers, researchers, and other stakeholders. Thus, various data elements from historical events of natural inland waterway disruptions such as floods and droughts along the Mississippi River and the McClellan–Kerr Arkansas River Navigation System (MKARNS) were used to develop a spatio-temporal statistical model [41], [42]. This model predicts disruptions at different locations on both rivers, which guide the movement of multi-industry cargo vessels, operation of the lock-and-dam system in the area, and decisions regarding other modes of transportation for products shipped to and from inland waterway terminals.

The simulated data are derived from actual data on ICIs. The ICIs related data includes: 1) inland waterway and ground transportation networks (e.g., road type and capacity of road network) [43]; 2) locations of dams and locks [13], [44], [45]; 3) locations of major ports and

their top commodities [13], [14]; 4) historical hydrological observation data at ports and locks including water depth, changes in waterways, and the normal capacity of inland water transport [46]; 5) major types of cargo vessels and barges classified by their capacity and usual transport speed; and 6) weather data covering the studied regions [47]. Moreover, the Maritime Transportation Research and Education Center (MarTREC) at the University of Arkansas [48] provides the Transportation Resource Data Bank [49] that compiles rich information, such as freight commodity flow and ports. It is worth pointing out that, although the proposed simulation methods are centered on multimodal transportation networks, they can be used broadly in modeling other local, regional, and national infrastructures after proper modifications. Especially, the access to the most recent version of the open-sourced simulation tool addressed in this article is currently available for researchers, decision-makers, and other stakeholders to advance research on multimodal transportation systems [50].

The remainder of this article is organized as follows. Section 2.1.2 describes the development of the spatio-temporal statistical model used in this study and the basic features of the model. Section 2.1.3 introduces the simulation tool developed on an open-source platform. Section 2.1.4 presents a case study to illustrate the capabilities of the tool. Section 2.1.5 provides concluding remarks and future research directions.

2.1.2 Methodology

A hybrid methodology combining statistical analysis and simulation is applied. The statistical modeling is employed with two primary purposes: 1) to map the spatial fluctuations of gage height on a given river across sites, to interpolate spatially unobserved points on a river and 2) to forecast the gage height measurements on the sites of interest and anticipate possible interruptions in the flow of vessels. The simulation-based modeling is used to create scenarios for vessel and truck flow, utilizing the results from the statistical models. The dynamic interaction between different input parameters and simulation controls allows for the estimation of various metrics.

2.1.2.1 Geo-spatial model

Environmental variables are among the factors affecting the reliability of ICIs. To represent the waterway transportation network, modeling relevant variables of the corresponding water bodies, e.g., rivers, becomes central in understanding the processes that affect the availability of the infrastructures of interest. The selected statistical modeling approach must make accurate predictions and estimate the confidence intervals for relevant variables on the selected sites. To this end, developing a model capable of capturing the underlying relationship between the selected variables, the spatial correlation among the selected measuring sites and the associated variations in time is one of the key tasks in this stage. The chosen framework is *spTimer* [51], a spatio-temporal Bayesian modeling package using the R language for statistics. The main variable of interest is the Gage Height (GH), a measure of the water's depth filling the waterways on the measurement sites. The main purpose of this model is to generate data to estimate the GH on unobserved sites of interest. In this context, unobserved sites are selected locations with no available measurements, and it is necessary to infer the missing GH data from those from the observed sites. The model will learn a spatio-temporal mapping for the GH data from the observed sites and generate interpolations for new coordinates of interest along the same rivers.

The proposed model captures the seasonal variation for each site's time series of gage height measurements along with the spatial correlation in such measurements, driven by spatial location and their relation to each river. Although historical data over a long time interval is used to showcase the model's performance, the model is potentially useful for stakeholders to predict future conditions, even under changes in spatial or temporal structure. A potential approach, beneficial for planning and control, is to repeat the model fit as soon as more data is available and limit the prediction horizon to a short but useful time interval. For example, the model could be readjusted every week using a sliding window of two years of historical data with a one-week prediction horizon. This would guarantee that the model is up to date with the environmental conditions and that the predictions are current and

informative.

2.1.2.2 Data

The data used corresponds to GH's hourly measurements and lock availability data in eighteen different sites, equivalent to eighteen geo-related time series, with 17,542 observations each (more than 315 thousand in total). From these, 22,961 are missing measurements, representing 7.3% of the total observations. A statistical summary of the GH measurements is shown in Table 2.1. The time window begins on February 22, 2016, and finishes on February 21, 2018. The observed sites are shown in Figure 2.1. The sites are classified as connected to the MKARNS (red) or the Mississippi River (green). The selected unobserved locations of interest are marked with "X."

Min	**Q1**	**Median**	**Mean**	**Q3**	**Max**
0.00	7.45	11.83	14.04	19.43	44.63

Table 2.1: Summary of GH statistics

2.1.2.3 Theoretical background

To model the GH data, a hierarchical autoregressive model specifying distributions for data, process, and parameters in three stages is presented. The data is modeled by a Gaussian Process with spatio-temporal random effects. Model parameter estimation is conducted using Bayesian computation methods [51] implemented through Gibbs sampling with the *spTimer* package in R. A summary of the nomenclature is presented.

Nomenclature

i	Index for sites
l	Index for longer time unit (e.g., months)
t	Index for shorter time unit (e.g., hours)

r	Total number of longer units
T_l	Total number of shorter units
n	Number of sites
N	Total number of observations
s_i	The i^{th} site
$Z_l(s_i, t)$	Observation at site i on time t
$O_l(s_i, t)$	True value of observation at site i and time t
$\epsilon_l(s_i, t)$	Error term at site i and time t
$\eta_l(s_i,\ t)$	Spatial random effect at site i and time t
$\mathbf{Z}_{lt}$	Vector of observations
$\mathbf{O}_{lt}$	Vector of true values
$\mathbf{X}_{lt}$	Matrix of covariates
$\boldsymbol{\epsilon}_{lt}$	Vector of error terms
$\boldsymbol{\eta}_{lt}$	Vector of spatial random effects
Σ_η	Covariance matrix of spatial random effects
S_η	Spatial correlation matrix
$\kappa\left(\mathbf{s_i}, \mathbf{s}_j; \phi, \nu\right)$	Correlation matrix entry for sites i and j
$\mathbf{z}$	Matrix of observations
$\mathbf{z}^*$	Matrix of missing observations
$\boldsymbol{\theta}$	Vector of parameters
ρ	Temporal correlation parameter
$\boldsymbol{\beta}$	Vector of covariate coefficients
σ^2_ϵ	Pure error variance
σ^2_η	Spatial random effects variance
ϕ	Rate of decay of the spatial correlation
ν	Smoothness of the correlation function
$\boldsymbol{\mu}_l$	Mean of the autoregressive component on the l^{th} time unit

σ_l^2	Variance of the autoregressive term on the l^{th} time unit

Let $Z_l(s_i,t)$ be the observed point-referenced data and $O_l(s_i,t)$ be the true value corresponding to $Z_l(s_i,t)$ at site $s_i,\ i=1,...,n$ at time denoted by the two indices l and t, where l and t represent two units of time, for which l denotes the longer unit (e.g., months), $l=1,...,r$, and t denotes the shorter unit (e.g., hours), $t=1,...,T_l$. Note that r and T_l are the total numbers of the two time units, respectively. Define two vectors $\mathbf{Z}_{lt} = (Z_l(s_1,\ t),\ \ldots,\ Z_l(s_n,\ t))^T$ and $\mathbf{O_{lt}} = (O_l(s_1,\ t),\ ...,\ O_l(s_n,\ t))^T$. Let $N = n\sum_{l=1}^{r} T_l$ be the total number of observations to be modeled. The observed data is represented by $\mathbf{z}$ and the missing data is denoted by $\mathbf{z}^*$. The hierarchical model used is expressed as follows with a description of variables and inputs, as presented in [51]:

$$\begin{aligned} \mathbf{Z}_{lt} &= \mathbf{O}_{lt} + \boldsymbol{\epsilon}_{lt} \\ \mathbf{O}_{lt} &= \rho\mathbf{O}_{lt-1} + \mathbf{X}_{lt}\boldsymbol{\beta} + \boldsymbol{\eta}_{lt} \end{aligned} \tag{2.1}$$

where $\boldsymbol{\epsilon}_{lt} = (\epsilon_l(s_1,t),...,\epsilon_l(s_n,t))^T$ denotes the nugget effect (i.e., the pure error term) and is assumed to follow $N(\mathbf{0},\sigma_\epsilon \boldsymbol{I}_n)$, ρ is the temporal correlation parameter, and $\boldsymbol{\beta}=(\beta_1,...,\beta_p)$ represents the regression coefficients of the $\mathbf{X}$ fixed effects or covariates. The spatio-temporal random effects are modeled by $\boldsymbol{\eta}_{lt} = (\eta_l(s_1,\ t),...,\eta_l(s_n,\ t))^T$, which is assumed to follow $N(\mathbf{0},\Sigma_\eta)$ and to be independent in time. Specially, $\Sigma_\eta = \sigma_\eta^2 S_\eta$, where σ_η^2 is the spatial variance assumed to be equal for all sites, and S_η is the spatial correlation matrix. In this article, S_η is obtained from the general Matérn correlation function [52], which is well suited to model a smooth process:

$$\kappa(\mathbf{s_i},\mathbf{s}_j;\phi,\nu) = \frac{1}{2^{\nu-1}\Gamma(\nu)}(2\sqrt{\nu}||s_i-s_j||\phi)^\nu K_\nu(2\sqrt{\nu}||s_i-s_j||\phi), \phi>0, \nu>0 \tag{2.2}$$

where $\Gamma(\nu)$ is the standard gamma function, K_ν is the modified Bessel function of second kind with order ν, $||s_i-s_j||$ is the distance between sites s_i and s_j, ϕ is the rate of decay of

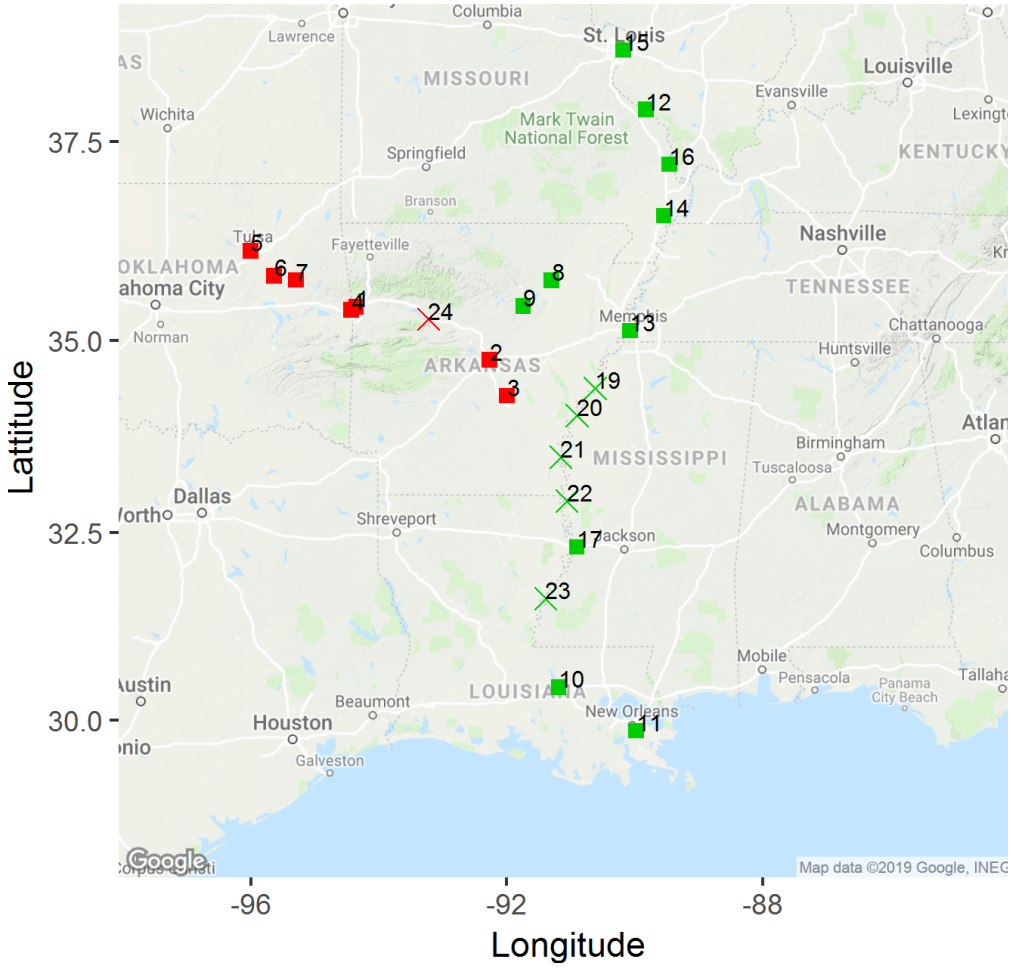


Figure 2.1: Measurement sites by rivers

the spatial correlation, and ν is the smoothness parameter. Note that for the autoregressive component, ρ, it requires the specification of $\mathbf{O}_{l0}$, the initial term, for each l. For this purpose, an additional mean parameter $\boldsymbol{\mu}_l$ and covariance matrix $\sigma_l^2 S_0$ must be estimated, with S_0 following the same structure as in Eq. 2.2.

Let $\boldsymbol{\theta} = (\beta, \rho, \sigma_\epsilon^2, \sigma_\eta^2, \phi, \nu, \boldsymbol{\mu}_l, \sigma_l^2)$ be the vector containing all the parameters of this model and $\pi(\boldsymbol{\theta})$ be the prior distribution of $\boldsymbol{\theta}$. The logarithm of the joint posterior distribution of

the parameters and the observed and missing data for this model is given by [51]:

$$\begin{aligned}\ln \pi\left(\theta, \mathbf{O}, \mathbf{z}^{*} | \mathbf{z}\right) \propto & -\frac{N}{2} \ln \sigma_{\epsilon}^{2}-\frac{1}{2 \sigma_{\epsilon}^{2}} \sum_{l=1}^{r} \sum_{t=1}^{T_{l}}\left(\mathbf{Z}_{l t}-\mathbf{O}_{l t}\right)^{T}\left(\mathbf{Z}_{l t}-\mathbf{O}_{l t}\right)-\frac{\sum_{l=1}^{r} T_{l}}{2} \ln \left|\sigma_{\eta}^{2} S_{\eta}\right| \\ & -\frac{1}{2 \sigma_{\eta}^{2}} \sum_{l=1}^{r} \sum_{t=1}^{T_{l}}\left(\mathbf{O}_{l t}-\rho \mathbf{O}_{l t-1}-\mathbf{X}_{l t} \boldsymbol{\beta}\right)^{T} S_{\eta}^{-1}\left(\mathbf{O}_{l t}-\rho \mathbf{O}_{l t-1}-\mathbf{X}_{l t} \boldsymbol{\beta}\right) \\ & -\frac{1}{2} \sum_{l=1}^{r} \ln \left|\sigma_{l}^{2} S_{0}\right|-\frac{1}{2} \sum_{l=1}^{r} \frac{1}{\sigma_{l}^{2}}\left(\mathbf{O}_{l 0}-\boldsymbol{\mu}_{l}\right)^{T} S_{0}^{-1}\left(\mathbf{O}_{l 0}-\boldsymbol{\mu}_{l}\right) \\ & +\ln (\pi(\theta))\end{aligned} \tag{2.3}$$

Using this posterior distribution and full conditionals as presented in [51], the estimation is carried out using Gibbs sampling. Then, the spatial interpolation or temporal extrapolation can be achieved using the predictive posterior for $Z_l(s_0, t')$ for any unobserved location s_0 and unobserved time point t':

$$\begin{aligned}\pi\left(Z_{l}(s_{0}, t') | \mathbf{z}\right) & =\int \pi\left(Z_{l}(s_{0}, t') | O_{l}(s_{0}, t'), \sigma_{\epsilon}^{2}\right) \times \pi\left(O_{l}(s_{0}, t') | \boldsymbol{\theta}, \mathbf{O}, \mathbf{z}^{*}\right) \\ & \times \pi\left(\boldsymbol{\theta}, \mathbf{O}, \mathbf{z}^{*} | \mathbf{z}\right) d O_{l}(s_{0}, t') d \mathbf{O} d \boldsymbol{\theta} d \mathbf{z}^{*}\end{aligned} \tag{2.4}$$

2.1.3 Hybrid simulation model

This simulation model is developed using NetLogo, an agent-based programming language and simulation platform offered as freeware [53]. NetLogo is also a cross-platform and integrated environment for modeling both simple and complex systems that evolve dynamically. In NetLogo, "Agents" (turtle, link, patch, and observer) are the integral part of the NetLogo world and can follow instructions given by the designers. Turtles move around in the two-dimensional world, whereas the world contains a grid of patches. Every patch represents a square piece of land. All these agents can operate simultaneously without interfering with one another. NetLogo permits users to run the simulation in a browser or desktop application, interact with the software, and analyze its behavior under various settings [54].

2.1.3.1 Overview of the simulation model

The developed model was built on four extensions of NetLogo: GIS (Geographic Information System), R (R Language for Statistics), NW (Networks), and CSV (Comma Separated Values). GIS extension provides the ability to load vector GIS data in the form of ESRI shapefiles. The GIS extension is used to import several maps in the simulation model. Initially, a map of the U.S. is loaded as the base of NetLogo environment. Then, maps of inland waterways and highways are imported and drawn on top of the base map. Here, our simulation focus is primarily on the MKARNS and lower Mississippi River, representing the case study in this article (see Section 2.1.4). Figure 2.2 shows NetLogo's user interface after opening and setting the basic environment of the model. The graphic window makes the two-dimensional "world" of the model visible. It is divided up into a grid of patches that have *pxcor* and *pycor* coordinates. The basic idea here is to create a NetLogo graph (nodes and links) by importing the GIS maps and creating vessel and truck "agents" that travel along with the links. The main components of the program are:

- A map of the United States, drawn on NetLogo in a simplified form. Each state border is drawn for reference. Figures 2.3–2.4, provide a zoomed version of the interface.

- Maps of navigable waterways and highways. Both maps are made of nodes, turtles with own properties, connected by links. While the waterways/highways are only figurative, the nodes play an active role in the simulation because they facilitate the simulation understanding of waterways/highways maps.

- Vessels. These are turtles with their own variables such as current location, destination, distance-traveled, speed, vessel category (1 for large-sized, 2 for medium-sized, and 3 for small-sized), product weight, product type, extreme events, total delay, and others related to the control of the travel logic.

- Trucks. These are also turtles with properties such as current location, destination,

distance-traveled, speed, product-weight, product-type, and others related to the control of the travel logic.

- Ports along the waterways. Eight ports are considered and modeled as a type of turtle. These are located in Tulsa, Fort Smith, Little Rock, Greenville (Mississippi), Baton Rouge, Helena, Memphis, and St. Louis along the MKARNS and Mississippi River. In Figure 2.3, the yellow nodes represent the ports.

- Fifteen locks along MKARNS. They are also made of nodes (a type of turtle) with properties such as ID and location.

- Twenty-four sites, modeled as turtles, along the MKARNS and Mississippi River. In each site, the gage height level is checked and a decision is made regarding whether the vessels will move forward or not. The red nodes in Figure 2.3 represent the sites.

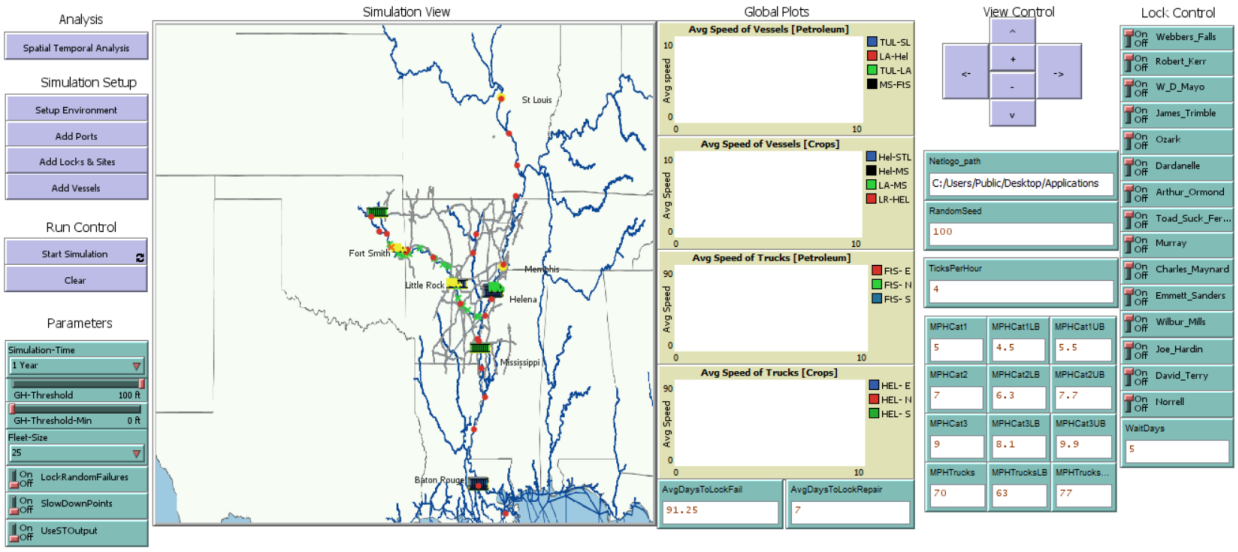


Figure 2.2: Simulation model interface on NetLogo

- An algorithm that makes the vessels and trucks move on the waterways and highways, respecting defined interaction rules of movement between source and destination, navigation time and speed, and other agents. For example, during the simulation, a vessel always takes the shortest path between its source and destination. The travel logic

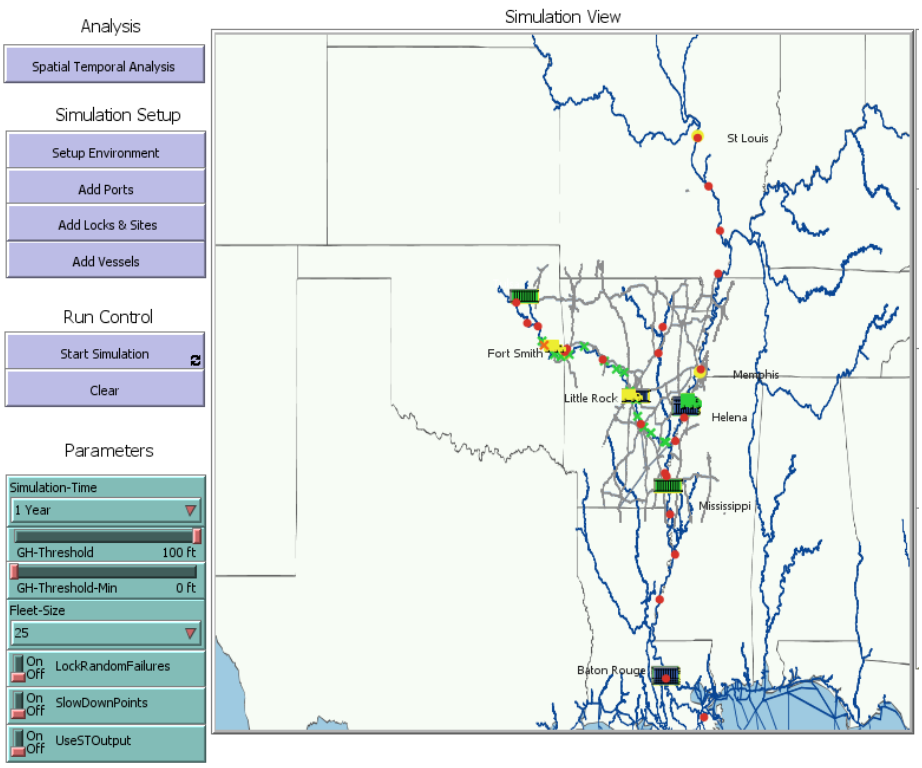


Figure 2.3: Simulation model interface setup controllers in NetLogo

controls that a defined speed is enforced for each vessel and truck. The speed is defined when the vessel or truck is created as a random variate following a truncated exponential distribution. The range and mean of said distribution are parameters that can be controlled by the user. During the simulation, each vessel and truck checks that the distance traveled along the next node in the path is consistent with its defined speed. If the distance is larger than what it should travel during a tick, it waits for another tick. If it is shorter than what it should travel, it progresses another step in the path and checks that the cumulative distance is consistent with the speed. If it encounters an obstacle in its path (e.g., a vessel facing an extreme event or disabled lock), it waits until the path is enabled again.

The main assumptions of our simulation model are as follows:

- Vessels are uniformly distributed based on the annual demand for commodities. The decisions for instantiating different vessels are encoded in the model following the times

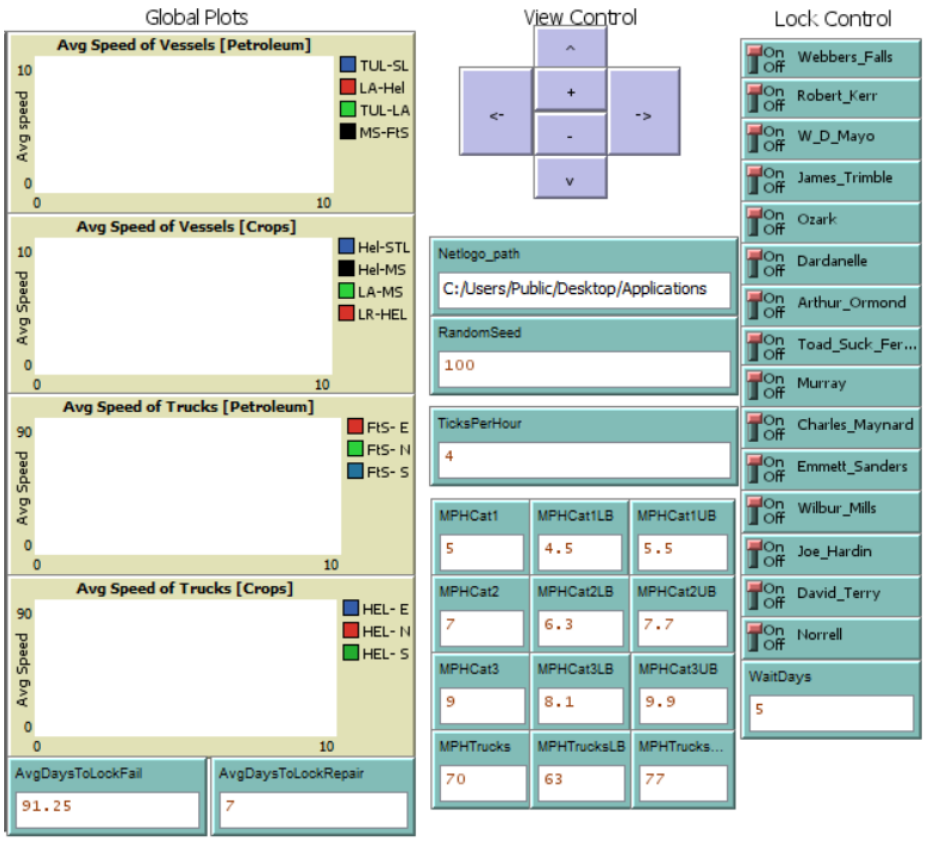


Figure 2.4: Simulation model interface view and lock controllers in NetLogo

between departures designed to have a uniform distribution throughout the year.

- The speed of vessels varies with its capacity and size [37], [55]. The smallest vessel is the fastest one with an average speed of nine mph. The medium-sized vessel moves at seven mph, where the largest one moves with five mph [37], [55]. A truncated exponential distribution is used to draw random values for the speed each time a vessel is created in the simulation.

- Each vessel and truck carries only one commodity type.

- All the vessels and trucks travel only once to their predefined destinations and do not return to their origin ports.

2.1.3.2 Interdependencies of critical infrastructures

The functional interdependencies among ICIs are modeled by simulating a certain number of traveling cargo vessels along the waterways and a number of available ports with various capacities and conditions. In the case of a natural disturbance (e.g., elevated water levels), a decision of offloading and re-routing based on the expected size and duration of the disturbance and the current and future conditions of the ground transportation network is made. Given the flexibility of the proposed simulation model, different scenarios can be tested to assess the decision making process.

Additionally, another form of functional interdependency is available through the simulation of the interconnected operations of dam, lock, port, and ground transportation in case of traffic congestion or disruption. Considering such interdependencies of critical infrastructures and the multimodal transportation components, different scenarios involving human interactions, such as flood discharge, dredging, and use and maintenance of locks, can be analyzed. Furthermore, cost analysis approaches can be implemented to estimate the economic impact of commodity flow decisions.

Potential response plans will be simulated to help researchers understand how enacted emergency plans impact the multimodal transportation system and the surrounding infrastructure. Moreover, the developed agent-based simulation tool is capable of simulating the evacuation and re-routing processes, for which the system performance can be presented for different points in time. Thus, the interdependency among multiple decision-makers at ports, ground transportation, and government in this simulation environment can be captured to generate various scenarios. Such scenarios can help coordinate the efforts to optimize the decision making process for all stakeholders involved.

2.1.4 Case study

2.1.4.1 Problem description

In the simulation, the system analyzes a pre-determined set of representative quantities based on the input parameters listed in Table 2.2. In addition, Figure 2.5 represents a sample run and Figure 2.6 shows a sample of plots that were generated during the simulation. We generate an output file at the end of the simulation which presents all the measurements

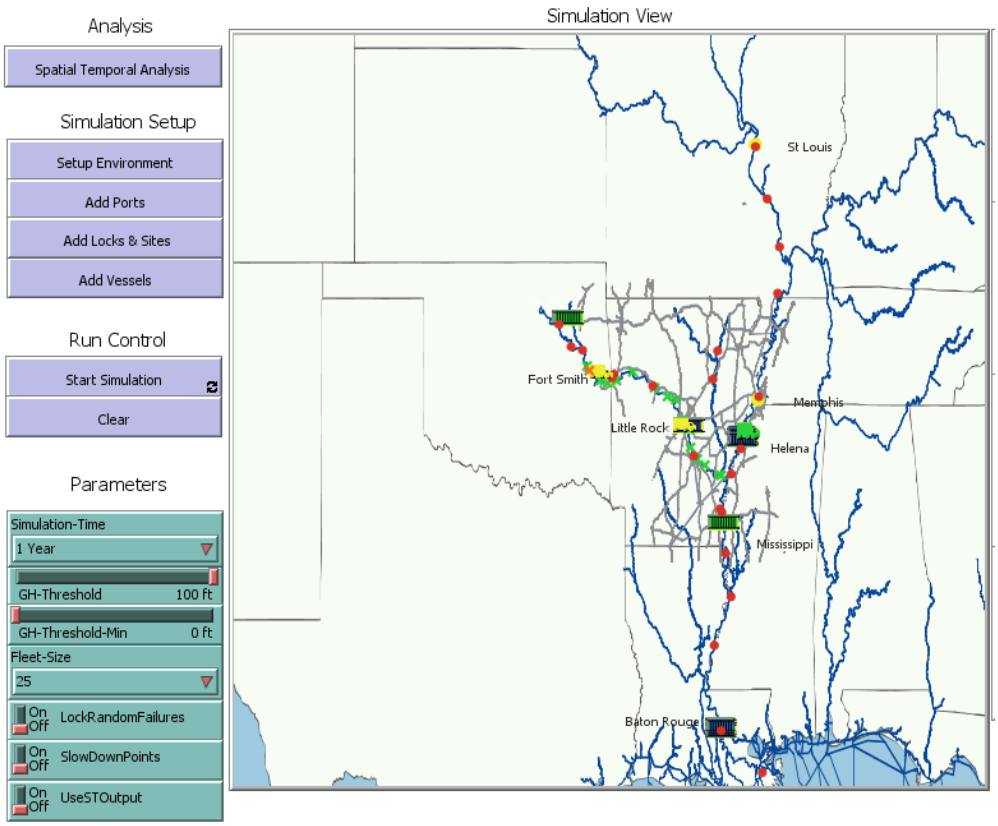


Figure 2.5: Simulation running on NetLogo

listed above. The simulation time is set for 12 months (1 year), the fleet size is set to 25 trucks, and GH global threshold is 100 feet with individual thresholds set to action stage level based on the National Weather Service (NWS) data [56]. Initially, vessels began from the ports of Tulsa, Baton Rouge, Little Rock, Greenville, and Helena, and the locks were all open. The vessels were moving towards their destination ports, and extreme events were checked by measuring gage height and lock availability at each site. For example, the vessels

passing the sites with GH greater than the allowable threshold between the LA-MS route along the Mississippi River could not move and had to wait until the GH level falls below the threshold. Whenever a vessel faces any extreme event and stops moving forward, its color turns red to represent a stoppage. The vessel gets back to its original color when movement resumes. If the disruption is prolonged, it may potentially inhibit the vessel from reaching its destination. This is moderated by the parameter "WaitDays" that controls the number of days a given barge waits before detouring into ground transportation as an alternative. To illustrate the capability of the model, three different simulation runs covering few different aspects of available inputs were generated. First, a base-case run is generated without including the developed spatio-temporal statistical model with level predictions or possibility of random lock failures. Second, a run that includes random lock failures was generated to show how the model handles new input data and how output results are affected. Third, the spatio-temporal statistical model is used to predict water levels; hence, it predicts extreme events resulting from elevated water levels. To validate the spatio-temporal statistical model predictions, we provide water level time series comparisons with our available true data for multiple sites. A summary of these conditions is presented in Table 2.3.

For the scope of the current model, random lock failures are assumed to follow exponential distributions, both for the time between failures and time between repairs. Locks are initiated in a working state, and the time until the next failure is drawn from the exponential distribution with the mean time of three months as the default value. The time to complete a repair is drawn from the exponential distribution with the mean time of one week as the default value. The alternation of states is continued until the simulation run ends. Note that the mean time to failure and the mean time to repair have been coded as two parameters that the user can modify as needed.

Moreover, when a vessel reaches its destination port, trucks are used to carry its products to the final destinations. Figure 2.6 (left) shows the number of vessels that were used to carry products (Crops) between two ports. At the end of the simulation, we generate an

Model Inputs	Model Outputs
• Gage height from a spatio-temporal model • Supply and demand between ports (movement of commodities) • Gage height threshold limit • Lock availability • Vessel distribution at each port • Fleet size • Number of trucks	• Average speed of each vessel category between every two ports (mph) • Number of delays between every two ports • Total time lost due to extreme events (hour) • Total number of vessels delayed and their tonnages • Overall average speeds for the three types of vessels (mph) • Number of extreme events and length (time) in MKARNS and lower Mississippi River • Number of vessels from each category traveled and arrived between every two ports • Average speed of trucks for each product type (mph)

Table 2.2: The main inputs and outputs of the developed simulation tool

output report summarizing multiple statistics of vessel (e.g., average speed and the number of extreme events) and truck behavior and summary plots (e.g., boxplots) that help the user understand varying aspects of the hybrid model.

2.1.4.2 Simulation results

This subsection reports the results obtained for a one-year base-case simulation run for the lower Mississippi River and MKARNS and compares selected results with the other two runs. These results are shown in Tables 2.4–2.7 and Figures 2.7–2.9. Table 2.4 shows summary statistics for vessels traveled between every two ports classified by the vessel category, where category 1 represents small-sized vessels carrying up to 6 ktons of cargo, category 2 represents medium-sized vessels carrying up to 12 ktons of cargo, and category 3 represent large-sized vessels carrying up to 18 ktons of cargo. Table 2.4 information includes the number of vessels that have traveled between every pair of ports, average speed of travel, product type carried (petroleum or crops in our case), records of vessel category, and the total weight

Scenario	Lock Failures	GH Measurements	Unobserved Sites
Base Case	Real observations	Real observations	Not included
Random Lock Failures	Randomly generated	Real observations	Not included
Spatio-temporal model	Real observations	Predictions from model	Interpolated from model

Table 2.3: Detail on simulation scenarios

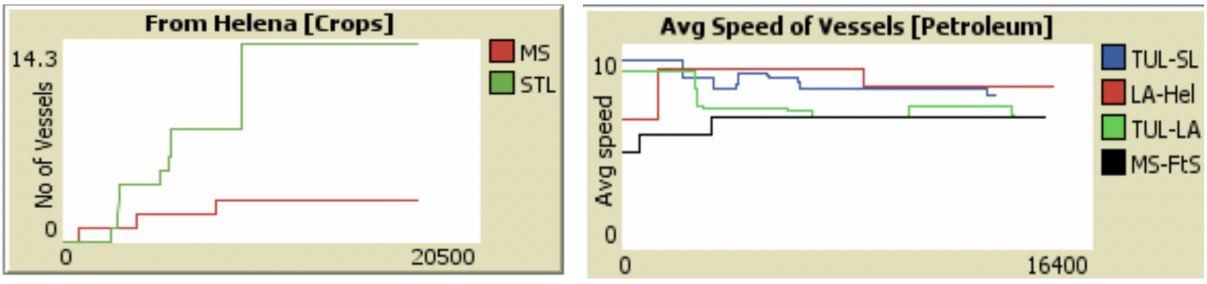


Figure 2.6: Instantaneous plotting information while simulation is running on NetLogo

of carried products (in kilo tons). The number of extreme events (disruptions) faced by the vessels is shown, where a disruption occurs in our setting whenever the water level exceeds a predefined threshold. The total delay (in hours) caused by such disruptions and the distance traveled by vessels are also recorded. Some of these statistical summaries are plotted such as the ones in Figures 2.7–2.9 showing a boxplot of the distributed time to destination, a bar chart of number of extreme events by vessel category, and a boxplot of average speed of each vessel category, respectively. The outputs of the simulation model also include detailed information about all vessels appeared in the model. Like the records mentioned before, the detailed information includes the product type carried, category of the vessel, total weight of the product, and a defined ID for each vessel (modeled as a turtle). Also registered, each vessel's start time, time of departure from its origin, and its end time (time of arrival at its destination). These records can help extend the simulation analysis and contribute to the model's debugging and validation. Table 2.7 shows summary statistics for trucks traveled from all ports to four defined exit points labeled as cardinal directions

(i.e., East, West, North, and South) located at the edges of the studied area. For example, we have 16 fleets of trucks that have traveled from the Fort Smith port to the eastern exit point with an average speed of 71.57 mph. Trucks that reach these boundary points are assumed to have left the area to other states to deliver goods. There are four random chosen boundary points located at the east, west, south and north of the studied area map. To model possible delays (disruptions) of trucking-delivery of goods, possible congestion on the highways is represented using "slow down points" which are sections where truck speed is reduced, generating a similar behavior of possible congestion on the road. This feature of the simulation model is useful when considering commodity flow planning with information about traffic data. These results are available with additional detailed information about the trucks, as shown in Table A.1 in Appendix.

From	To	Product type	Vessel category	Vessel count	Weight (ktons)	Avg. Speed (mph)	Extr. events	Distance (miles)	Total delay (hours)
Baton Rouge	Helena	Petroleum	1	1	18	3.79	0	540.69	0
Baton Rouge	Helena	Petroleum	2	1	12	7.83	0	540.30	0
Baton Rouge	Helena	Petroleum	3	1	6	10.65	0	548.47	0
Baton Rouge	Mississippi	Crops	1	3	54	4.20	1	355.04	1
Baton Rouge	Mississippi	Crops	2	4	48	9.06	0	376.49	0
Baton Rouge	Mississippi	Crops	3	8	48	10.88	3	360.63	2.5
Baton Rouge	Mississippi	Petroleum	1	1	18	3.53	1	356.43	1
Baton Rouge	Mississippi	Petroleum	2	1	12	8.74	0	342.95	0

Baton Rouge	Mississippi	Petroleum	3	1	6	10.90	0	375.89	0
Helena	Baton Rouge	Petroleum	1	1	18	4.41	0	549.37	0
Helena	Baton Rouge	Petroleum	2	1	12	8.04	0	552.42	0
Helena	Baton Rouge	Petroleum	3	1	6	10.76	0	546.18	0
Helena	Mississippi	Crops	1	1	18	3.79	0	186.45	0
Helena	Mississippi	Crops	2	1	12	6.35	0	182.55	0
Helena	Mississippi	Crops	3	1	6	10.11	0	164.31	0
Helena	Mississippi	Petroleum	1	1	18	3.90	0	170.69	0
Helena	Mississippi	Petroleum	2	1	12	6.94	0	173.61	0
Helena	Mississippi	Petroleum	3	1	6	10.42	0	198.04	0
Helena	St. Louis	Crops	1	3	54	3.69	0	603.78	0
Helena	St. Louis	Crops	2	4	48	8.72	0	604.63	0
Helena	St. Louis	Crops	3	7	42	10.44	0	600.64	0
Little Rock	Helena	Crops	1	12	216	4.40	0	222.02	0
Little Rock	Helena	Crops	2	18	216	7.67	0	221.72	0
Little Rock	Helena	Crops	3	31	186	10.25	0	222.14	0
Mississippi	Baton Rouge	Crops	1	6	108	3.75	2	381.82	1.5
Mississippi	Baton Rouge	Crops	2	8	96	9.06	3	380.19	3
Mississippi	Baton Rouge	Crops	3	16	96	11.03	1	382.06	1
Mississippi	Baton Rouge	Petroleum	1	1	18	4.07	1	379.89	1
Mississippi	Baton Rouge	Petroleum	2	1	12	8.43	0	381.25	0

Mississippi	Baton Rouge	Petroleum	3	1	6	10.38	1	378.96	0.5
Mississippi	Fort Smith	Petroleum	1	1	18	4.33	0	445.88	0
Mississippi	Fort Smith	Petroleum	2	1	12	7.65	0	441.58	0
Mississippi	Fort Smith	Petroleum	3	1	6	11.42	0	445.21	0
Mississippi	St. Louis	Crops	1	10	180	4.01	0	766.42	0
Mississippi	St. Louis	Crops	2	15	180	8.12	0	766.67	0
Mississippi	St. Louis	Crops	3	29	174	10.29	0	765.51	0
Tulsa	Baton Rouge	Petroleum	1	5	90	4.64	4	992.74	4
Tulsa	Baton Rouge	Petroleum	2	7	84	8.13	2	989.29	2
Tulsa	Baton Rouge	Petroleum	3	13	78	10.87	6	994.07	6
Tulsa	Helena	Petroleum	1	1	18	4.26	0	621.25	0
Tulsa	Helena	Petroleum	2	1	12	8.01	0	612.57	0
Tulsa	Helena	Petroleum	3	2	12	10.26	0	620.40	0
Tulsa	Memphis	Petroleum	1	2	36	4.42	0	691.78	0
Tulsa	Memphis	Petroleum	2	3	36	8.02	0	693.74	0
Tulsa	Memphis	Petroleum	3	6	36	10.32	0	693.14	0
Tulsa	Mississippi	Petroleum	1	3	54	4.73	0	613.82	0
Tulsa	Mississippi	Petroleum	2	5	60	8.30	0	624.32	0
Tulsa	Mississippi	Petroleum	3	9	54	10.70	0	625.59	0
Tulsa	St. Louis	Petroleum	1	4	72	4.08	3	1,222.94	107.5
Tulsa	St. Louis	Petroleum	2	6	72	7.44	1	1,217.46	190.5
Tulsa	St. Louis	Petroleum	3	12	72	8.54	4	1,217.97	820.25

Table 2.4: Summary of vessels' statistics for the base-case simulation runs

From	To	Product type	Vessel category	Vessel count	Weight (ktons)	Avg. Speed (mph)	Extr. events	Distance (miles)	Total delay (hours)

Baton Rouge	Helena	Petroleum	1	1	18	3.79	0	536.62	0
Baton Rouge	Helena	Petroleum	2	1	12	7.79	0	541.13	0
Baton Rouge	Helena	Petroleum	3	1	6	10.66	0	559.80	0
Baton Rouge	Mississippi	Crops	1	3	54	4.16	2	361.48	2
Baton Rouge	Mississippi	Crops	2	4	48	8.79	3	364.60	2.75
Baton Rouge	Mississippi	Crops	3	8	48	10.85	2	354.17	2
Baton Rouge	Mississippi	Petroleum	1	1	18	3.49	1	372.51	1
Baton Rouge	Mississippi	Petroleum	2	1	12	8.79	0	362.73	0
Baton Rouge	Mississippi	Petroleum	3	1	6	10.86	0	361.19	0
Helena	Baton Rouge	Petroleum	1	1	18	4.41	0	537.93	0
Helena	Baton Rouge	Petroleum	2	1	12	8.00	0	539.78	0
Helena	Baton Rouge	Petroleum	3	1	6	10.75	0	548.47	0
Helena	Mississippi	Crops	1	1	18	3.74	0	187.75	0
Helena	Mississippi	Crops	2	1	12	6.20	0	175.08	0
Helena	Mississippi	Crops	3	1	6	10.34	0	191.32	0
Helena	Mississippi	Petroleum	1	1	18	3.90	0	164.87	0
Helena	Mississippi	Petroleum	2	1	12	7.07	0	178.47	0
Helena	Mississippi	Petroleum	3	1	6	10.34	0	180.94	0
Helena	St. Louis	Crops	1	3	54	3.68	0	606.06	0
Helena	St. Louis	Crops	2	4	48	8.73	0	601.40	0
Helena	St. Louis	Crops	3	7	42	10.45	0	601.31	0

Little Rock	Helena	Crops	1	12	216	4.38	0	224.75	0
Little Rock	Helena	Crops	2	18	216	7.63	0	226.20	0
Little Rock	Helena	Crops	3	31	186	10.33	0	224.12	0
Mississippi	Baton Rouge	Crops	1	6	108	4.11	3	382.48	3
Mississippi	Baton Rouge	Crops	2	8	96	8.49	3	381.50	3
Mississippi	Baton Rouge	Crops	3	16	96	10.92	5	382.18	4
Mississippi	Baton Rouge	Petroleum	1	1	18	4.12	0	381.25	0
Mississippi	Baton Rouge	Petroleum	2	1	12	8.43	0	379.39	0
Mississippi	Baton Rouge	Petroleum	3	1	6	10.48	0	379.89	0
Mississippi	Fort Smith	Petroleum	1	1	18	4.32	0	442.14	0
Mississippi	Fort Smith	Petroleum	2	1	12	7.64	0	445.21	0
Mississippi	Fort Smith	Petroleum	3	1	6	11.38	0	440.96	0
Mississippi	St. Louis	Crops	1	10	180	4.00	0	767.11	0
Mississippi	St. Louis	Crops	2	15	180	8.23	0	766.19	0
Mississippi	St. Louis	Crops	3	29	174	10.23	0	768.11	0
Tulsa	Baton Rouge	Petroleum	1	5	90	4.45	2	992.51	2
Tulsa	Baton Rouge	Petroleum	2	7	84	8.35	3	994.31	1.75
Tulsa	Baton Rouge	Petroleum	3	13	78	10.93	3	990.76	2.5
Tulsa	Helena	Petroleum	1	1	18	4.77	0	612.57	0
Tulsa	Helena	Petroleum	2	1	12	8.14	0	614.94	0
Tulsa	Helena	Petroleum	3	2	12	10.81	0	617.21	0

Tulsa	Memphis	Petroleum	1	2	36	4.64	0	692.79	0
Tulsa	Memphis	Petroleum	2	3	36	7.11	0	692.13	0
Tulsa	Memphis	Petroleum	3	6	36	10.45	1	692.36	1
Tulsa	Mississippi	Petroleum	1	3	54	4.71	0	631.48	0
Tulsa	Mississippi	Petroleum	2	5	60	7.87	1	631.82	1
Tulsa	Mississippi	Petroleum	3	9	54	11.13	0	634.82	0
Tulsa	St. Louis	Petroleum	1	4	72	4.02	0	1,217.46	0
Tulsa	St. Louis	Petroleum	2	6	72	7.71	3	1,222.25	231.75
Tulsa	St. Louis	Petroleum	3	12	72	7.54	5	1,221.08	1216.75

Table 2.5: Summary of vessels' statistics for the random lock failure case

From	To	Product type	Vessel category	Vessel count	Weight (ktons)	Avg. Speed (mph)	Extr. events	Distance (miles)	Total delay (hours)
Baton Rouge	Helena	Petroleum	1	1	18	3.79	0	540.69	0
Baton Rouge	Helena	Petroleum	2	1	12	7.81	0	541.12	0
Baton Rouge	Helena	Petroleum	3	1	6	10.66	0	541.13	0
Baton Rouge	Mississippi	Crops	1	3	54	4.19	0	372.45	0
Baton Rouge	Mississippi	Crops	2	4	48	9.05	0	368.37	0
Baton Rouge	Mississippi	Crops	3	8	48	10.94	0	359.85	0
Baton Rouge	Mississippi	Petroleum	1	1	18	3.53	0	382.63	0
Baton Rouge	Mississippi	Petroleum	2	1	12	8.73	0	340.31	0

Baton Rouge	Mississippi	Petroleum	3	1	6	10.89	0	362.05	0
Helena	Baton Rouge	Petroleum	1	1	18	4.41	0	549.37	0
Helena	Baton Rouge	Petroleum	2	1	12	8.02	0	555.68	0
Helena	Baton Rouge	Petroleum	3	1	6	10.67	0	557.39	0
Helena	Mississippi	Crops	1	1	18	3.79	0	186.45	0
Helena	Mississippi	Crops	2	1	12	6.32	0	172.26	0
Helena	Mississippi	Crops	3	1	6	10.11	0	161.73	0
Helena	Mississippi	Petroleum	1	1	18	3.93	0	167.99	0
Helena	Mississippi	Petroleum	2	1	12	6.73	0	159.89	0
Helena	Mississippi	Petroleum	3	1	6	10.43	0	185.07	0
Helena	St. Louis	Crops	1	3	54	3.68	0	605.75	0
Helena	St. Louis	Crops	2	4	48	8.72	0	608.37	0
Helena	St. Louis	Crops	3	7	42	10.45	0	606.94	0
Little Rock	Helena	Crops	1	12	216	4.40	0	224.23	0
Little Rock	Helena	Crops	2	18	216	7.67	0	221.62	0
Little Rock	Helena	Crops	3	31	186	10.26	0	221.21	0
Mississippi	Baton Rouge	Crops	1	6	108	3.76	0	382.54	0
Mississippi	Baton Rouge	Crops	2	8	96	9.12	0	384.15	0
Mississippi	Baton Rouge	Crops	3	16	96	11.05	0	382.47	0
Mississippi	Baton Rouge	Petroleum	1	1	18	4.12	0	390.17	0
Mississippi	Baton Rouge	Petroleum	2	1	12	8.40	0	377.97	0

Mississippi	Baton Rouge	Petroleum	3	1	6	10.40	0	390.17	0
Mississippi	Fort Smith	Petroleum	1	1	18	4.33	0	445.88	0
Mississippi	Fort Smith	Petroleum	2	1	12	7.65	0	441.58	0
Mississippi	Fort Smith	Petroleum	3	1	6	11.41	0	442.14	0
Mississippi	St. Louis	Crops	1	10	180	4.01	0	766.41	0
Mississippi	St. Louis	Crops	2	15	180	8.12	0	767.70	0
Mississippi	St. Louis	Crops	3	29	174	10.29	0	765.47	0
Tulsa	Baton Rouge	Petroleum	1	5	90	4.65	0	995.40	0
Tulsa	Baton Rouge	Petroleum	2	7	84	8.14	0	992.99	0
Tulsa	Baton Rouge	Petroleum	3	13	78	10.92	0	993.82	0
Tulsa	Helena	Petroleum	1	1	18	4.26	0	612.57	0
Tulsa	Helena	Petroleum	2	1	12	8.01	0	612.57	0
Tulsa	Helena	Petroleum	3	2	12	10.25	0	619.71	0
Tulsa	Memphis	Petroleum	1	2	36	4.43	0	692.60	0
Tulsa	Memphis	Petroleum	2	3	36	8.02	0	692.70	0
Tulsa	Memphis	Petroleum	3	6	36	10.31	0	690.98	0
Tulsa	Mississippi	Petroleum	1	3	54	4.71	0	619.98	0
Tulsa	Mississippi	Petroleum	2	5	60	8.31	0	634.12	0
Tulsa	Mississippi	Petroleum	3	9	54	10.70	0	630.51	0
Tulsa	St. Louis	Petroleum	1	4	72	4.09	1	1,222.59	106.5
Tulsa	St. Louis	Petroleum	2	6	72	7.44	1	1,219.24	191.5
Tulsa	St. Louis	Petroleum	3	12	72	8.54	4	1,221.53	824.25

Table 2.6: Summary of vessels' statistics for the spatio-temporal model

Comparing the results for the base-case scenario with the random lock failures (shown in Table 2.5), one can observe an overall increase in the total delay hours for random lock failures. In fact, the total delay in hours for the base-case scenario is about 1,141 hours compared to approximately 1,474 hours for the case of random lock failure. This shows

that the model responds as expected to changes across simulation runs. For the third case (i.e.,spatio-temporal predictions) shown in Table 2.6, one can see that there is no significant difference between the model's final output of approximately 1,122 hours of delay compared to the base-case one with 1,141 hours. This shows that the developed spatio-temporal statistical model is predicting close water levels compared to the actual data available used in the base-case run. In addition, to validate the spatio-temporal model predictions of water levels, Figure 2.10 compares the predicted values (with a 90% point-wise confidence band) to the real observed data in two selected sites. Two important elements can be seen in Figure 2.10: (1) the prediction captures the seasonality of the actual data and (2) the trend of the spatio-temporal statistical model behaves as the actual data. Specially, the developed statistical model captures both trend and seasonality with a low mean squared error (MSE) of 1.7 ft^2 for the fitted values vs. the GH observations.

From	Product Type	Destination	Count	Avg. Speed (mph)
Fort Smith	Petroleum	E	16	71.57
Fort Smith	Petroleum	N	2	68.99
Fort Smith	Petroleum	S	2	67.42
Helena	Crops	E	61	68.32
Helena	Crops	N	61	68.78
Helena	Crops	S	61	61.41
Helena	Petroleum	E	16	66.25
Helena	Petroleum	N	2	69.35
Helena	Petroleum	S	2	57.92

Table 2.7: Summary of trucks' statistics for the base-case simulation runs

2.1.5 Conclusions and future research

In this study, multiple contributions are made to the ICIs risk analysis and commodity flow literature. First, a spatio-temporal statistical model was developed to capture extreme natural events causing disruptions in inland waterways and predict them in the future to facilitate commodity flow planning and response actions. The developed statistical model can also handle missing data without a noticeable degradation in its overall performance. In

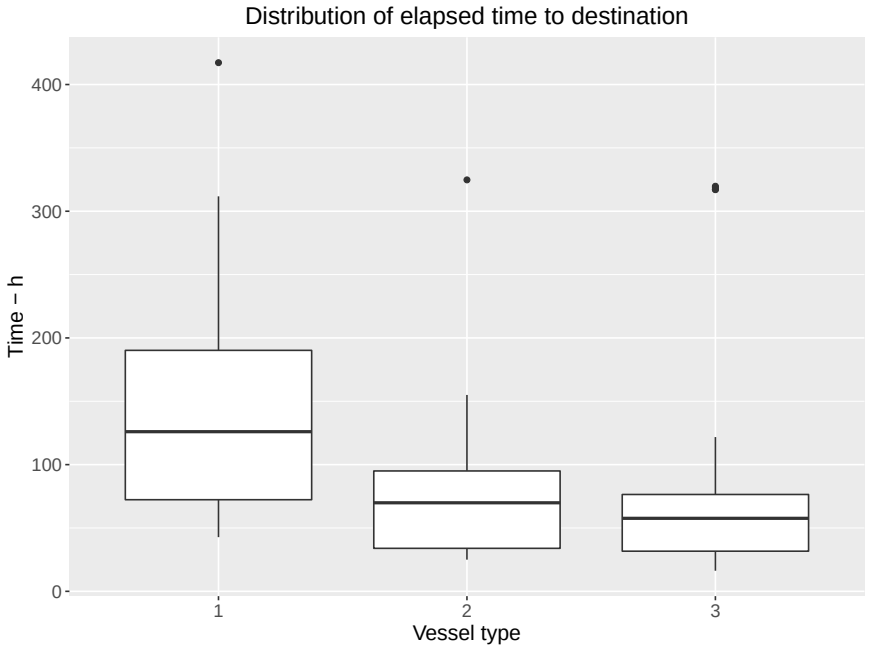


Figure 2.7: Simulation model output of time to reach destinations for vessels in different categories (in hours)

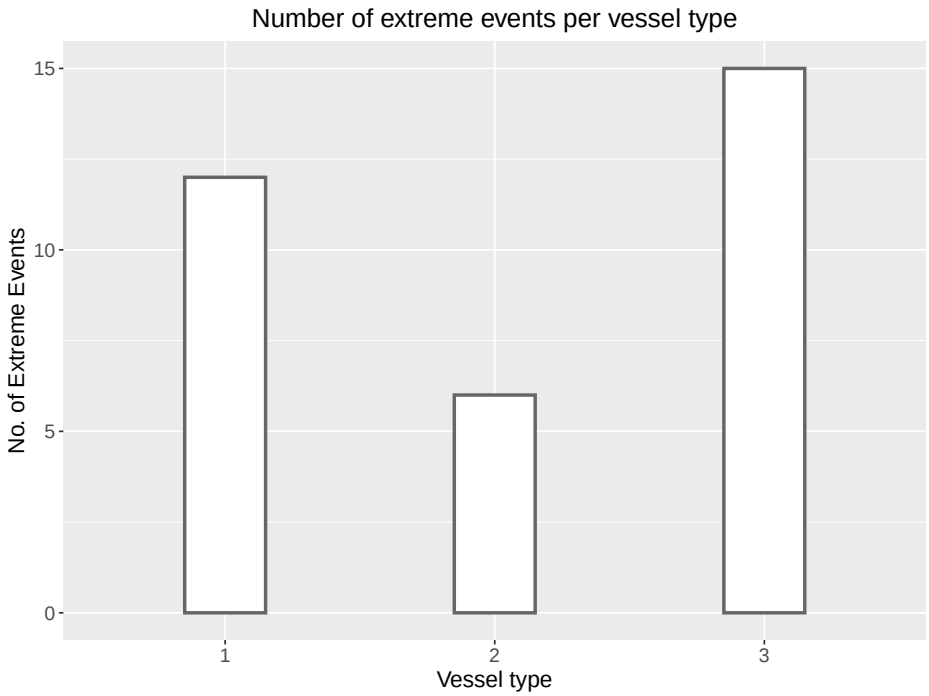


Figure 2.8: Number of extreme events for vessels in different categories

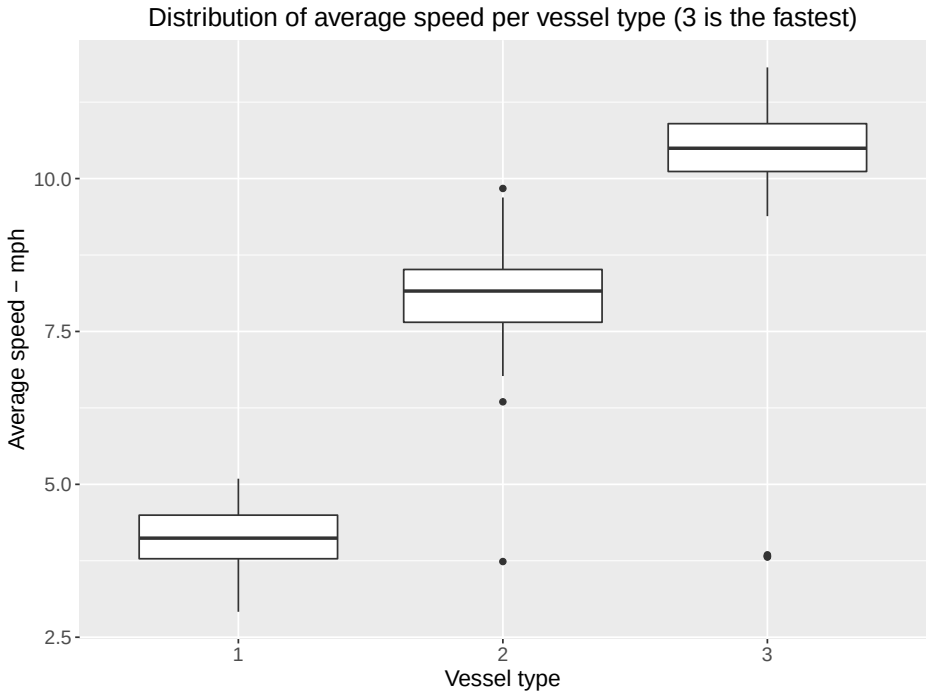


Figure 2.9: Average speeds of vessels in different categories (mph)

addition, the statistical model was developed and tested on the lower Mississippi River and the MKARNS. Second, a simulation tool is built to capture the effect of inland waterways disruptions on the commodity flow through other ICIs, which provides a broad understanding of the multimodal transportation system interdependencies in action. Third, access to the most recent version of the simulation model is currently available as an open-sourced tool for researchers, decision-makers, and other stakeholders to advance research in multimodal transportation [50].

The current version of the model has limitations that can be reduced by this research team or potential users of this open-sourced tool. Especially, distributing vessels uniformly over time might not be the best representation of real demand as it most likely is not stationary and presents some forms of seasonality. This is a promising research direction that can be explored in the future. Another limitation is the capability of the statistical model for emulating outliers in predicting GH measurements. The current model uses a Bayesian Gibbs sampling approach that relies on the mean of sampled predictions. This

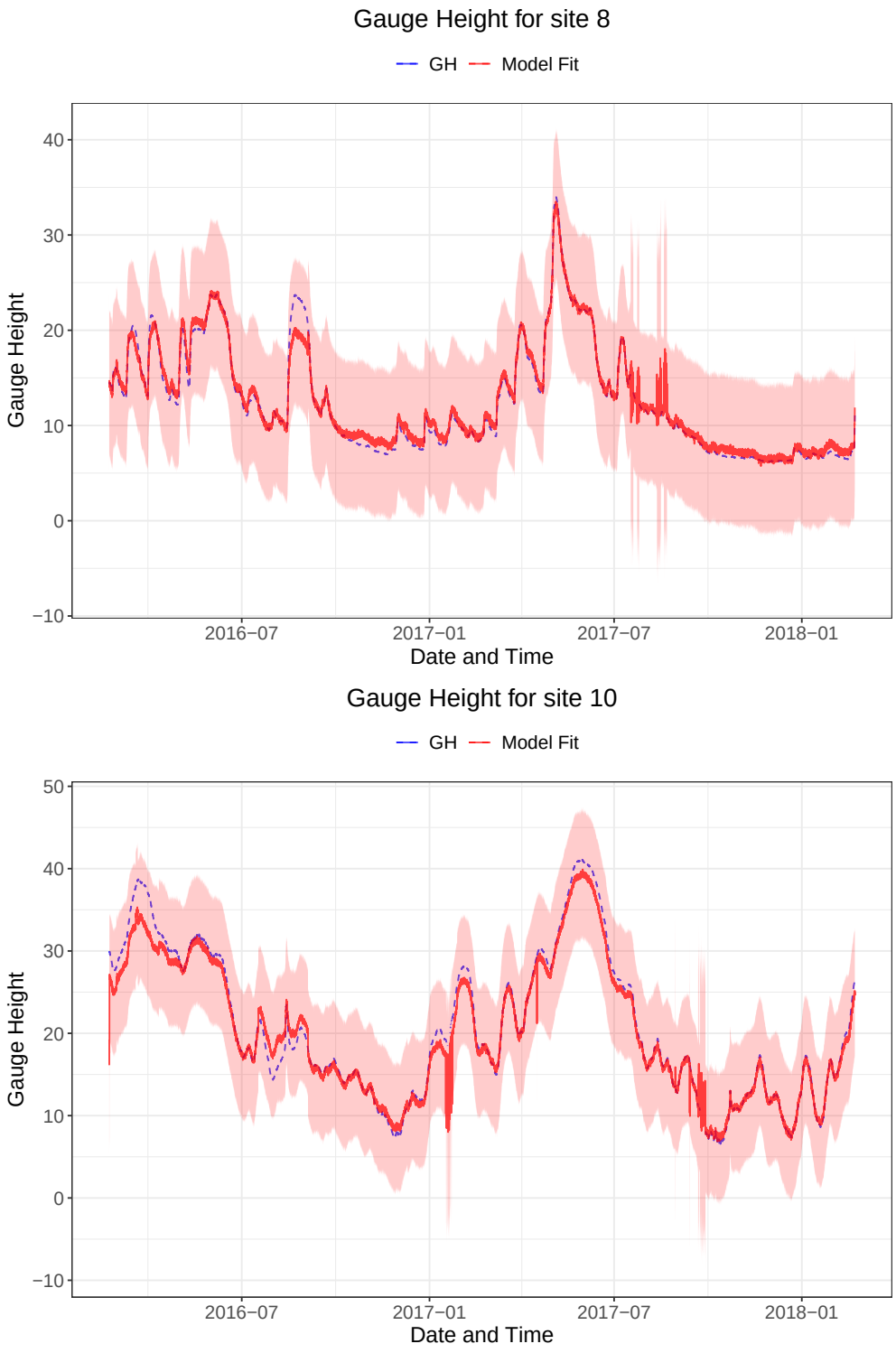


Figure 2.10: Water levels (in ft) predicted by the Spatio-temporal statistical model vs. actual data at some selected sites

may lead to conservative estimates. Clearly, emphasis on this limitation can be another valuable research direction.

This work could be extended to support emergency service response and detailed analysis of ports operations. In addition, national economic and transportation studies centered on inland waterways and their interdependency with ground transportation can be investigated by extending the developed simulation tool to include features such as private trucking companies routes, real-time traffic data, and railroads information. An economic study based on the developed simulation tool covering the current case study is among the near future research directions.

2.2 Extensions to the Inland Waterway Transportation Simulation Model

As part of the explored applications of simulation in ICIs contexts, developing the NetLogo simulation tool was a crucial step in creating a testbed to quantify the impact of disruptions due to failures in Locks along the MKARNS corridor. The additional developments of this tool are centered on adding flexibility in the simulation logic while garnering computational efficiency such that the simulation environment becomes useful for exploring data-driven solutions in decision-making related to inspection, maintenance, and repair operations related to the continuous control of the locks system in along the Arkansas River. With this goal in mind, the additional developments follow two main routes. The first one is defining a new paradigm for the simulation logic such that the codebase is organized around transportation entities, with capabilities for seamless integration with new transportation modes and elements in the transportation process, and unifying the control logic such that further improvements are feasible within reasonable development timelines. The second route is creating a DRL environment that connects this new simulation model with state-of-the-art continuous control algorithms using data-driven decision-making through Reinforcement Learning.

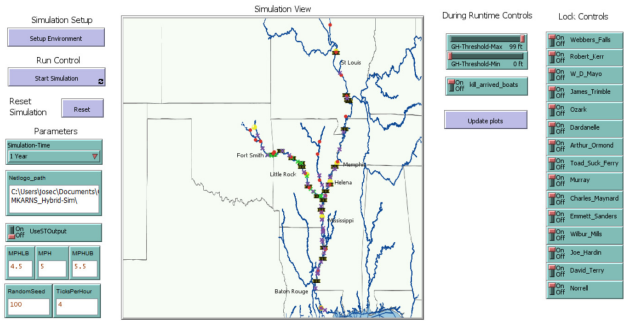


Figure 2.11: Updated Graphic Interface for the Simulation Model

2.2.1 New Paradigm: Barges as Minimal Transportation Unit

The new simulation emphasizes using barges as the minimal transportation unit. In practical terms, this means that most of the quantities that are tracked during runtime and after a simulation interval is finished are related to statistics about barges. From counting barges in traffic per route, barges that are in transit, barges affected by extreme events, and service and travel times related to barges. A barge becomes the smallest unit that can be transported, and one tow boat can move several barges simultaneously. There is now a correspondence between the types of commodities transported and the quantities of these commodities on each barge. This exists now in opposition to the previous paradigm where the smallest transportation unit was the towboat, and as a proxy for the number of barges that each towboat carried, three vessel categories were defined. These categories encapsulated the number of barges carried by each towboat, with the most significant category corresponding to having the most barges while not explicitly keeping track of the number or service times affecting them. Additionally, the graphic interface was also updated; the current version looks as in Figure 2.11.

The redefined paradigm focuses the event control logic on the flow of barges along the system. Specific agents and entities now control all events in the simulation, and a unified event flow logic is proposed. This is expected to grant future developmental advantages in the flexibility of the code base to new additions and readability and faster integration of new entities due to the standardized event control logic.

2.2.1.1 Setup Reorganization

The simulation setup has been overhauled to remove repetitive steps in the previous code base as efficiently as possible. While some of the fixed infrastructural elements were loaded only once at the beginning of the simulation, others were constantly called during each step. Some fixed elements were even called on each step by each active towboat entity, resulting in multiple unnecessary calls to repetitive steps. Reducing these inefficiencies has led to significant speed ups and now the simulation setup is timed on a regular PC in around 10 seconds, in opposition to previous results where each setup was closer to 5 minutes.

2.2.1.2 Modules Reorganization

The codebase is now separated into modules and submodules based on functionality. This allows for increased readability, friendlier and maintainable code, and easier identification of bugs and errors by grouping similar functions. The main modules are separated in the following structure:

- Entities: code related to agents and objects in the simulation
  - Turtles: general properties for entities
  - Routes: definition of source, destination, and paths to follow along waterways
  - Commodities: defining the classes and properties of the transported cargo
  - Barges: minimal transportation unit. Barges flow along the waterways to carry cargo
  - Boats: towboats carrying barges along the waterways
  - Nodes: river segments that define the minimal geographical unit along the waterways
  - Sites: locations with available information about water levels along the waterways

  - Ports: source and destination locations used to define the start and end points of routes along the waterways
  - Reroutings: particular geographical points controlling the logic of redirecting traffic under extreme events

- Initialization: code related to the simulation setup
  - Constants: centralizing all constant values in the simulation
  - Setup: creating the simulation environment
  - Display: creating the visual interface for the simulation

- Logic: code related to the flow of events
  - Simulate: code controlling the step logic
  - Move: code controlling the transportation logic
  - Go: code controlling the geographical distance logic
  - Random Variables: code related to the generation of random variates

- Results: code related to displaying the metrics tracked during runtime
  - Plots: visual representation of results
  - Outputs: reports and files generated after simulation

- Utilities: various auxiliary functions
  - File reading: reading external files
  - Utilities: miscellaneous functions

2.2.1.3 Logic Reorganization

In this new paradigm, the flow of events is controlled by the different entities. A global simulation step calls a general update to all entities with an internal simulation step. This

internal simulation step checks the start and end of individual events, which governs the event flow logic. Some entities do not have an internal step and thus are static during the whole simulation interval. Using this idea, a previously static entity can be granted properties that evolve dynamically over time by adding an internal simulation step and including its entity class in the call from the global update step. Additionally, some efficiency is gained through this as not all entities need an update step, and thus selecting which ones do, and even for those that do update, conditions, and checks can be added in the individual step to skip updates if they are not relevant for a given time interval, therefore saving some computational resources and gaining efficiency.

2.2.2 New Entities

The entities included in the simulation can be classified as transportation entities and infrastructure entities. Transportation entities move along the system and control the logic for this movement. This includes barges, towboats, commodities, and routes. For these entities, reporter functions are created to keep track of relevant statistics. Infrastructure entities do not flow along the system but control most of the logic related to spatial positions, and some events turn the flow on or off through the system. This includes ports, locks, river segments (nodes), sites (water level measurement locations), and rerouting points.

2.2.2.1 Transportation Entities

- Barges: smallest transportation unit. The number of barges transported by each towboat is generated randomly. The commodity type carried is selected randomly depending on factors such as the route to which this barge belongs. Statistics are kept for the number of barges generated, in transit, and successfully arrived at destination.

- Towboats: boats carry multiple barges along the waterways. The distribution of barges per boat is controllable depending on the selected route. The generation of boats is controlled with random numbers and is also parameterizable depending on each specific

route. Statistics are kept for the number of boats generated, in transit, and successfully arrived at destination.

- Commodities: each barge carries one commodity type. These commodities are defined in the configuration files. The distribution of the proportion of each type is defined in the configuration files and can be specified generally for all routes or individually for specific routes. Statistics are kept for the number of barges of each commodity type generated, in transit, and successfully arrived at the destination.

- Routes: these entities define the sources and destinations for towboats in transit. Routes also define the path that each towboat will follow along the waterways as a sequence of node entities that the boat will visit along the way, depending on road conditions such as water levels and inherent properties such as speed. Statistics are kept for the number of boats, barges, and commodities generated, in transit, and arrived on each route.

2.2.2.2 Infrastructure Entities

- Nodes: river segments along the waterways. The smallest geographical unit in the transportation simulation along the waterways. All other infrastructure elements are assigned to one of these elements to keep track of spatial locations.

- Ports: source and destination locations used to define the start and end points of routes along the waterways.

- Locks: infrastructure that assists the flow of towboats along the waterways by moving them from different elevation levels using gravity and the available water levels. For the purposes of this simulation, locks are treated as a multiple server single queue system, with a configurable number of servers and specifications on the service time distributions. The current logic makes the service dependent on the number of barges each

serviced towboat carries. Statistics are tracked for the number of serviced towboats and barges.

- Sites: points along the waterways with information about water level measurements. The available data in these points controls traversability logic along the waterways, dependent on the water levels within specified thresholds. If the observed water level is not in the allowed range, the given river segment is considered as not traversable, and the flow of towboats is stopped until the water level changes.

- Reroutings: points along the waterway that can be utilized to reroute vessels to different destinations, following pre-specified policies. Reroutings have been previously used to redirect traffic flow in case of disruptions due to extreme water level events or interrupted lock system service.

2.2.3 New Simulation Logic

The new event control logic is centered around the entities in the simulation. All entities that change over time have an internal simulation step that is called from a class simulation step, and all class simulation steps are called from a global simulation step. This allows for controlling events in non-static entities with a unified event control framework while allowing the integration of multiple events on each agent.

2.3 Decision-making using DRL

Further extending this work, optimal control of maintenance, repair, and inspection decisions using DRL is explored in [6], [7]. This author's contributions include developing the simulation model used in both pieces of work and the majority of the Python interface connecting the NetLogo simulation with the DRL environment, the DRL environment itself, the training scheme, the results display, and the processing of outputs. The main goal of these works is to develop a data-driven maintenance and repair operations control for the

infrastructure network of locks along the waterway corresponding to the MKARNS corridor.

2.3.0.1 Problem Description

The locks in the lock network are prone to failure and follow some time-to-failure distribution for a binary status and a single failure mode, with individual parameters that evolve independently for each lock. Consider a set of actions, call them action space, that can be performed on each lock to modify the parameters of these distributions or return the status of each lock from failure to operational. A DRL agent is developed as a control model that optimally allocates resources in a sequence of actions to maximize the traffic flow through the system by prioritizing the availability of relevant locks in the system, subject to feasibility and budget constraints.

For this problem, these actions include "do nothing", "maintenance," "inspection," and "repair." There is a limited number of resources, called the "repair crew", and to perform each action, it is necessary to take hold of one available repair crew and to temporarily set the lock as out of commission while the action is performed. A limited action budget constrains the number of actions that can be performed at a determined time interval, for example, a year of simulation time. Another constraint limits the number of maintenances by enforcing a minimum time interval between maintenances for each lock. Each specific action, at any time step, can only be carried out if feasible for a given lock: repairs are only enacted on failed locks, and maintenance and inspections only on operational locks. For the version presented in [6], repairs are compulsory and pre-specified by a policy prioritizing locks with higher traffic volume. Repairs are compulsory for [7], but the DRL agent developed has the freedom to specify which failed lock will be repaired first. Due to the ample action space and the combinatorial nature of the sequence of actions in the allocation of repairs, maintenance, and inspections, together with the stochasticity in the transportation network and locks statuses, approaching this problem using DRL is considered a reasonable course of action.

The results presented in [6], [7] will be further extended to incorporate more complex logic and eventually make the number of repair crews part of one of the actions that the agent can modify to service the lock network, effectively transforming the problem into a scheduling problem with seasonal use of resources.

2.3.1 Conclusions

The modifications to the NetLogo simulation model are crucial to using data-driven decision-making tools, such as DRL, as the gains in computational efficiency enable this interaction between the emulation of the waterway transportation system and the statistical learning tool while using reasonable amounts of computational resources. The new simulation logic represents an increase in the fidelity of the modeling of the transportation logic and enables additional customization and flexibility in simulating different scenarios and operation conditions along the waterway system. These contributions are expected to further enable research in the area of optimal maintenance control by extending the viability of incorporating DRL by creating a testbed environment where the policies can be learned and evaluated.

3. Prognostics and Health Management

3.1 Prognostic Using Dual-Stage Attention-Based Recurrent Neural Networks

3.1.1 Introduction

A system usually undergoes a transition through multiple degraded states before failure [57]. In practice, it is crucial to accurately assess the status of such a system and make the right decision on maintenance and spare parts inventory planning. Condition monitoring aims at observing the system's behavior to gauge the system's reliability and take appropriate actions. In particular, one of the most important tasks of condition monitoring is to estimate the Remaining Useful Life (RUL) of the system during operation.

This paper proposes the use of a Dual-Stage Attention-Based Recurrent Neural Network (DA-RNN) to estimate the RUL of turbine engine units [58] in a prognostics setting. These turbine engines belong to a non-linear complex system monitored by multiple sensors under different operational settings. A data-driven approach will be used in this paper to map multiple sensor readings to the RUL of an engine. The main focus will be on the performance and prediction capabilities of the model.

The variety of methods previously used to map the RUL as a function of selected sensor variables and the operational settings is remarkable [59]. However, even though the decisions that drove them were based on data and defined on clear rules, manual selection in some of the approaches would require additional inspection to migrate them to any other problem settings.

This paper is focused on the development of an adaptive algorithm handling multiple sensor measurements for RUL prediction. In particular, the goal is to apply a prognostics model capable of mapping the multidimensional sensor input variables to the RUL through an adaptive selection of driving features.

The main focus of this work will be on the desired capabilities of the model:

1. Identify automatically the relevant sensor inputs for prediction

2. Capture time dependencies between variables

3. Enable working with varying length time series for prediction

3.1.2 Problem Statement

The data, which is part of the C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) Turbofan datasets, represents a series of run-to-failure simulations of turbine engine units. The dataset was used as training and test sets and is referred to as "5T" in [59]. The authors in [59] offer a good clarification about the different C-MAPSS datasets and also present a framework to understand the known approaches and viable comparisons of the algorithms explored on them. The selected dataset has been widely used for benchmarking of prognostic methods: there are over 60 publications using it [58]. There are 6 datasets derived from C-MAPSS. Datasets 1 to 4 are referred to as "Turbofan data" and numbers "5T" and "5V" as the "PHM 2008 Data Challenge". The differences between each dataset lie in the number of failure modes simulated, the number of operation conditions, and the number of units in training and test segmentations [59].

To evaluate the performance of a prediction method, the following criterion is considered:

$$s = \begin{cases} \sum_{i=1}^{n} e^{\frac{-d_i}{a_1}} - 1 & d_i < 0 \\ \sum_{i=1}^{n} e^{\frac{d_i}{a_2}} - 1 & d_i \geq 0 \end{cases} \tag{3.1}$$

$$d_i = \widehat{RUL} - RUL \tag{3.2}$$

where a_1=13 and a_2=10. The evaluation metric shown in Eqn. (3.1) is defined as a loss function that penalizes deviations in the predicted RUL from the true value. In Eqn. (3.2), d_i represents the difference between the estimation and the true value of the RUL. The penalization is asymmetric: over estimating the RUL grants higher scores. Lower scores represent more accurate predictions.

3.1.3 Data

The data is comprised of 218 multivariate time series in the Training and Test data sets, corresponding to a total of 436 turbine engines. Each engine unit has 21 associated sensor measurements per run cycle and 3 additional variables that encode the 6 operational settings. These variables make the mutivariate time series where, for a given unit, each cycle is a point in time where the measurements are obtained. The time interval between cycles is not available in the data, and therefore will not be used in the prognostic calculations. The shortest series in the training and test samples are 128 and 15 cycles long; the longest series, 357 and 364.

The obtained score will be computed on the test data set using the NASA data repository website [58] for benchmarking on the held out RUL. All test evaluations are on dataset 5T-Test. This allows to have a comparable benchmarking to the methods mentioned in [59].

3.1.3.1 Previous Work

The authors in [60] use a Multi-layer Perceptron (MLP) and a Kalman Filter (KF) to create ensembles of the best predictions and capture the time related features of the same. [61] uses a Recurrent Neural Network (RNN) for RUL prediction. A Similarity Based Approach is presented in [62]. An analysis is carried out to group units on the training set that have a similar pattern in sensor trends and operational conditions. The relevant variables used for prediction were selected manually considering how seemingly smooth and continuous the plots of said variables look over cycle time. Regressions were created for each operational condition and then fused together using rules defined by the observed values of the RUL order statistics and expert criterion. The predicted values were manually capped on a threshold selected based on changes on the test score.

3.1.4 Proposed Method

3.1.4.1 Model

The considerations previously exposed led to the proposed model, the Dual-Stage Attention-Based Recurrent Neural Network (DA-RNN), which follows the architecture proposed by [63]. It was designed for multivariate time series data analysis to approach problems of the type known as Nonlinear Autoregressive Exogenous Models (NARX). This implies predicting a time series of interest using several exogenous driving time series as predictors. The main strengths of this model match with the capabilities desired: adaptively selecting the relevant time series for prediction and capturing long-term temporal dependencies. The model is composed of two Attention Mechanisms connected in series. The first one maps the multiple exogenous variables to the response series, learning the relevance of each of them in the feature space. This is the Input Attention Mechanism. The Temporal Attention Mechanism maps the time dependent components of the response variable at a given point in time to its future state.

The following equations define the components of the Input Attention Mechanism:

$$\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^n)^\intercal = (\mathbf{x}_T^1, \mathbf{x}_T^2, ..., \mathbf{x}_T^n)^\intercal \tag{3.3}$$

$$e_t^k = \mathbf{v}_e^\intercal \tanh\big(\mathbf{W}_e[\mathbf{h}_{t-1}; \mathbf{s}_{t-1}] + \mathbf{U}_e \mathbf{x}^k\big) \tag{3.4}$$

$$\alpha_t^k = \frac{\exp\big(e_t^k\big)}{\sum_{i=1}^{n} \exp(e_t^i)} \tag{3.5}$$

$$\tilde{\mathbf{x}}_\mathbf{t} = (\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, ..., \alpha_t^n x_t^n)^\intercal \tag{3.6}$$

where $[\cdot\,;\cdot]$ represents concatenation.

The encoder adaptively selects the relevant driving series among the multidimensional input (3.3). The encoder maps the relationship between the input and the hidden states. It is composed of three different layers in sequence: an MLP (3.4), a softmax (3.5) and a Long-Short Term Memory Network (LSTM) (3.7) to capture long-term dependencies. The

LSTM layers are composed of a memory cell with hidden states $\mathbf{s_t}$, and access to this cell is controlled by the forget (3.8), input (3.9) and output (3.10) gates. As a major adaptation from the original work, in the proposed model 3 LSTM layers are stacked. This makes the encoder a Deep Neural Network.

A single LSTM layer in the encoder is defined with the following equations:

$$\mathbf{h_t} = f_1(\mathbf{h_{t-1}}, \tilde{\mathbf{x}}_\mathbf{t}) \tag{3.7}$$

$$\mathbf{f_t} = \sigma(\mathbf{W_f}[\mathbf{h_{t-1}}; \mathbf{x_t}] + \mathbf{b_f}) \tag{3.8}$$

$$\mathbf{i_t} = \sigma(\mathbf{W_i}[\mathbf{h_{t-1}}; \mathbf{x_t}] + \mathbf{b_i}) \tag{3.9}$$

$$\mathbf{o_t} = \sigma(\mathbf{W_o}[\mathbf{h_{t-1}}; \mathbf{x_t}] + \mathbf{b_o}) \tag{3.10}$$

$$\mathbf{s_t} = \mathbf{f_t} \odot \mathbf{s_{t-1}} + \mathbf{i_t} \odot \tanh(\mathbf{W_s}[\mathbf{h_{t-1}}; \mathbf{x_t}] + \mathbf{b_s}) \tag{3.11}$$

$$\mathbf{h_t} = \mathbf{o_t} \odot \tanh(\mathbf{s_t}) \tag{3.12}$$

The decoder captures the relevant encoder hidden states acrosss all time steps [63]. For this, it uses another LSTM layer (3.17) and input attention mechanisms (3.13). The attention weights β_t^i represent the importance of the hidden states steps (3.14). The context vector is the weighted sum of said states (3.15). After this, adapting the original work to be feasible in the prognostics setting, the context vectors are combined with the number of elapsed cycles, $\dot{y}_{t-1}$, in Eqn. (3.16). In the original work, it was combined with the previous value of the series of interest. In this case, as the RUL will be unknown on the test units, we use instead the number of elapsed cycles. An estimation for the RUL value on the previous step is obtained and used to update the hidden states of the LSTM layer. Then, for the final prediction Eqn. (3.23) weighs the final hidden states and context vector using an MLP. As an additional step used to improve the predictions, a Kalman Filter was used to smooth the predicted RUL values.

The following equations define the components of the Temporal Attention Mechanism, including its LSTM layer:

$$l_t^i = \mathbf{v}_d^\intercal \tanh(\mathbf{W}_d[\mathbf{d}_{t-1}; \mathbf{s'}_{t-1}] + \mathbf{U}_d\mathbf{h}_i), 1 \leq i \leq T \tag{3.13}$$

$$\beta_t^i = \frac{\exp(l_t^i)}{\sum_{j=1}^{T} \exp\left(l_t^j\right)} \tag{3.14}$$

$$\mathbf{c_t} = \sum_{i=1}^{T} \beta_t^i \mathbf{h}_i \tag{3.15}$$

$$\tilde{y}_{t-1} = \tilde{\mathbf{W}}[\dot{y}_{t-1}; \mathbf{c}_{t-1}] + \tilde{b} \tag{3.16}$$

$$\mathbf{d_t} = f_2(\mathbf{d_{t-1}}, \tilde{y}_{t-1}) \tag{3.17}$$

$$\mathbf{f'_t} = \sigma(\mathbf{W'_f}[\mathbf{d_{t-1}}; \tilde{y}_{t-1}] + \mathbf{b'_f}) \tag{3.18}$$

$$\mathbf{i'_t} = \sigma(\mathbf{W'_i}[\mathbf{d_{t-1}}; \tilde{y}_{t-1}] + \mathbf{b'_i}) \tag{3.19}$$

$$\mathbf{o'_t} = \sigma(\mathbf{W'_o}[\mathbf{d_{t-1}}; \tilde{y}_{t-1}] + \mathbf{b'_o}) \tag{3.20}$$

$$\mathbf{s'_t} = \mathbf{f'_t} \odot \mathbf{s'_{t-1}} + \mathbf{i'_t} \odot \tanh(\mathbf{W'_s}[\mathbf{d_{t-1}}; \tilde{y}_{t-1}] + \mathbf{b'_s}) \tag{3.21}$$

$$\mathbf{d_t} = \mathbf{o'_t} \odot \tanh(\mathbf{s'_t}) \tag{3.22}$$

$$\hat{y}_T = \mathbf{v}_y^\intercal \tanh(\mathbf{W}_y[\mathbf{d}_T; \mathbf{c}_T] + \mathbf{b}_w) + b_v \tag{3.23}$$

3.1.4.2 Training

The algorithm is implemented in Python using the PyTorch deep learning framework [64]. Pytorch was chosen because the use of GPU for calculations seemed easy to implement and the use of tensor data types and the efficient memory management are possible. A working example of an initial implementation [65] was adjusted. Following the work in [63], minibatch stochastic gradient descent (SGD) was used to train the model through backpropagation. The optimization method used was QH-ADAM [66]. The size of the minibatch is 10 and the learning rate was set to a determined schedule: it starts from 0.005 and is reduced 10% after every 10K iterations. The size of the encoder and decoder hidden layers was set to 512. The objective function chosen was the mean squared error (MSE).

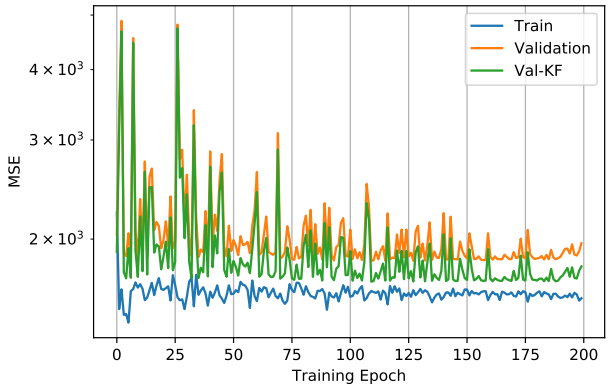


Figure 3.1: Evolution of MSE

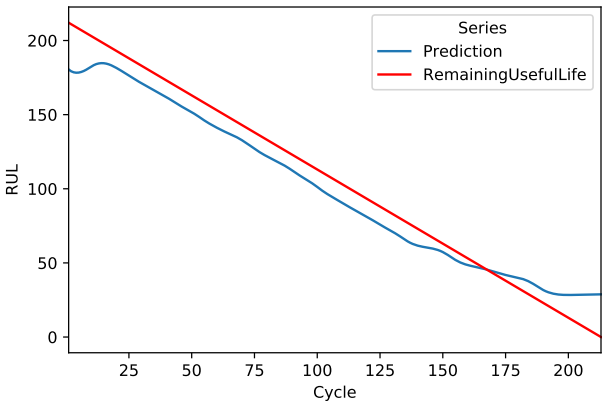


Figure 3.2: RUL estimation for Unit 100

A held out validation dataset was created as a subset of dataset 5T-train. 30% of the units in the sample were selected using a stratified sample, preserving the proportion of different series lengths on the training and validation sets.

All the models were trained on a laptop computer with 8 GB of RAM, an Intel Core i7-8750H processor and a NVIDIA GeForce GTX 1060 with Max-Q Design GPU was used for the backpropagation calculations. Using this setup, the time used for training 1 model during 200 epochs is around 22 hours, including validations on each epoch.

3.1.5 Results

The evolution of the training and validation MSE for the selected configuration is shown in Fig. 3.1. The MSE was computed at each epoch for the training and validation subsets. For the validation, the MSE previous to applying the Kalman Filter was also computed on each epoch. As expected, the error while training is lower than the validation error, and the KF decreases the error.

The prediction for unit 100 in the 5T-train dataset is shown in figure 3.2. This plot shows the RUL in the vertical axis and the number of elapsed cycles in the horizontal axis. For units in the train sample, this will always look like a diagonal line with negative slope, starting with RUL equal to the total number of cycles in the Run-to-Failure and ending in zero. It seems necessary to emphasize that even when this plot looks linear, the relationship

between the multiple sensor measurements and the RUL is not necessarily linear. This plot shows that the prediction is accurate on early to mid-life stages, but it tends to overestimate the RUL on cycles that are close to the failure event. The predictions for the units in the training sample follow a similar pattern. With these predictions, the evaluation metric is $s = 11,556$, which is good enough as the top 25 approach in the ranking presented in [59]. However, there is still room for improvement.

3.1.6 Conclusion

The DA-RNN model provides a novel and robust approach in prognostics. However, there are still some improvement opportunities to address so this model will be on the same level as the top approaches presented in [59]. Some of the model identified strengths are the good fit on RUL estimation for early to mid-life cycles. In addition, the model seems to capture well the time-dependent nature of the RUL values. Additional experiments will test variations in hyperparameter combinations, using a systematic search like Bayesian Optimization. Indeed, the potential of the proposed DA-RNN and the technical modifications that may arise as research progresses are expected to be of great benefit for the advancement of data-driven prognostics methods.

4. Study on All-Terminal Network Reliability

4.1 Deep Reinforcement Learning and All-Terminal Network Reliability

4.1.1 Introduction

Infrastructure networks, such as highways, communication networks, power networks, and water networks, play an essential role in our daily activities. Unfortunately, natural disasters and malicious attacks pose serious threats to these infrastructure networks. Historically, many failures in infrastructure networks occurred which have caused issues for many people. One well-known example is the 2003 Northeast blackout that affected fifty million people in the United States and Canada [67]. Another failure in infrastructure networks include the levee failure in Louisiana during Hurricane Katrina [68]. The levees in Louisiana were not adequately prepared to handle the water from Hurricane Katrina, thus, they breeched due to the pressure and caused much of New Orleans to flood. Clearly, these examples show how essential it is to ensure infrastructure networks are reliable.

To quantify the reliability of an infrastructure network, one essential task is to investigate the connectivity of components in the network. Mathematically, the problem can be formulated as an all-terminal network reliability problem. In practice, quite a few infrastructure networks can be modelled as an all-terminal network, such as highways, communication networks, power networks and water networks. To calculate all-terminal network reliability, numerous methods have been used. These methods provide either an exact value or an estimate of the reliability. Ball et al. [69] summarizes exact methods for calculating network reliability such as exponential time exact algorithms for general networks and polynomial time exact algorithms for restricted classes of networks, as well as other methods such as bounds on network reliability, and Monte Carlo simulation. Gaur et al. [70] also detailed many different network reliability methods including state enumeration, minimal cut, and neural networks, and they discussed the limitations of each method. Technically, cut enumeration entails enumerating the minimal subsets of links whose failure causes the network to fail. This method is an exact method and very useful for small networks, but it reaches

its computational limitations very quickly. Monte Carlo simulation (MCS) methods choose a random sample of states to explore and estimate the network reliability as the proportion of sampled states in which the network is functioning properly. Karger [71] found one of the flaws of the MCS approach is that it is very slow when the probability of failure is very low. Cardoso et al. [72] studied Monte Carlo simulation in conjunction with neural networks to investigate the structural reliability of different structures. MCS only allows one network structure to be calculated at a time, so it can be very time consuming to calculate the reliability. As a solution, they combined neural networks with MCS which allowed them to save computational time and obtain more precise reliability measurements.

Srivaree-ratana et al. [73] used an Artificial Neural Network (ANN) to estimate network reliability. In their study, they trained the ANN using a set of network topologies and link reliabilities. They then used the ANN to estimate the network reliability based on the link reliabilities and the topology in finding the optimal network topology by simulated annealing. They demonstrate that their approach performs well empirically through comparisons to an exact approach as well as to an upper bound derived from a polynomial time algorithm. However, the disadvantages of their method are that the training of ANN needs to be performed first for a topology of a fixed number of nodes and optimal network design can be carried out only for this topology. It would be more useful to develop a method that finds the optimal network via reliability evaluation and learning without such limitations.

In this paper, a new method based on Deep Reinforcement Learning (DRL) along with the use of a reliability polynomial is proposed for maximizing the all-terminal reliability of a network under the constraints on total budget and available types of edges for each step. To demonstrate the use of the proposed method, the initial structures of example networks are in the form of all nodes being connected in series. It is worth pointing out that although this paper focusses on maximizing the all-terminal reliability of a network by adding additional links, the proposed method can be extended to solve network design problems with the flexibility of adding additional nodes.

The remainder of this paper is organized as follows. Section 2 describes the reliability model for an infrastructure network and the method of calculating all-terminal reliability using a reliability polynomial. Section 3 introduces the proposed DRL method for network reliability improvement and elaborates on several important computational issues. Section 4 provides numerical examples to illustrate the use of the proposed method in improving infrastructure network reliability. Finally, we summarize our results and draw conclusions in Section 5.

4.1.2 Reliability model for an infrastructure network

An infrastructure network can be described by a network model, which in its simplest form is a collection of nodes connected by edges. Chartrand [74] formally defines a general network using the notation $N = (V, E, w)$, where V is the set of nodes (e.g., v_1, v_2, ..., v_n) and E is the set of edges (e.g., $e_{1,2}$, ..., $e_{i,j}$, ..., $e_{n-1,n}$) with the corresponding weights given in w. In this paper, the weights of the edges are the corresponding reliability values. Moreover, networks can either be directed or undirected. In this work, an infrastructure network is modelled as an undirected network, and reliability improvement decisions are made with respect to the network's all-terminal reliability.

4.1.2.1 All-terminal reliability of a network

The probability that a network is performing its intended function at a given point in time is known as its reliability. Specially, the two-terminal reliability of a network is the probability of having at least one operational path between the source and end nodes. Consider the simple undirected network shown in Figure 4.1. The network has four nodes and five links with corresponding reliability values. If node 1 and node 4 are the source and end nodes, respectively, and the nodes are perfectly reliable, the two-terminal reliability of the network can be calculated by considering three possible paths: for path 1-3-4, the reliability is $R_1 = 0.85(0.8) = 0.68$; for path 1-4, the reliability is $R_2 = 0.95$; for path 1-2-4, the reliability is

$R_3 = 0.9(0.75) = 0.675$. Since the three paths are in parallel, the two-terminal reliability of the network is simply $R = 1–(1 - R_1)(1 - R_2)(1 - R_3) = 0.9948$.

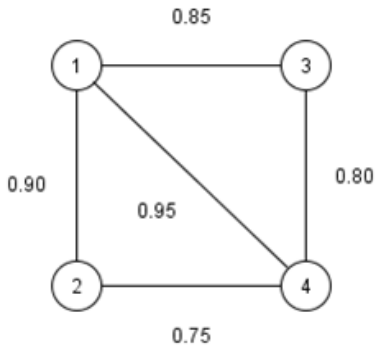


Figure 4.1: An example of simple series-parallel network.

Unlike two-terminal reliability problems, all-terminal reliability problems are interested in that every node in the network is connected to every other node, and the reliability is defined as the probability that the network is fully connected. Consider an n-node network (V, E, w) with edge topology $X = [x_{1,2}, \ldots, x_{i,j}, \ldots, x_{n-1,n}]$ with $x_{i,j}$= 1, if edge $e_{i,j}$ is present; 0, otherwise. Let $p(x_{i,j})$ be the reliability of edge $e_{i,j}$. Then, the all-terminal reliability of the network can be expressed as in Equation 4.1 [73].

$$R = \sum_{X' \in \Omega} \left[\prod_{(i,j) \in X'} p(x_{i,j}) \right] \left[\prod_{(k,l) \in (X \setminus X')} (1 - p(x_{k,l})) \right] \tag{4.1}$$

In Equation 4.1, Ω consists of all operational states (i.e., edge subsets $X' \subset E$ that connect all nodes in the network). For example, to calculate the all-terminal reliability of the network in Figure 4.1, we can simply calculate the probabilities of all network configurations where all nodes remain connected even if one or more edges fail. Then, after adding all the probabilities together, we obtain the all-terminal network reliability to be 0.9414. Clearly, it becomes more difficult to calculate all-terminal reliability for complex networks with more nodes and edges [75].

4.1.2.2 Reliability polynomial for all-terminal reliability evaluation

The all-terminal reliability of a network can be expressed as a function of the edge reliabilities. This expression is a property arising from the network topology, and it is often known as the reliability polynomial of the network. For a network N, when all edges have identical and constant reliability of r, the all-terminal reliability is equivalent to Equation 4.2 [76].

$$RP(r) = r^{n-c}(1-r)^{m-n+c}T(1,(1-r)^{-1}) \tag{4.2}$$

In Equation 4.2, n is the number of nodes, m is the number of edges, and c is the number of connected components. T is the Tutte Polynomial of the network, a property arising from the network topology, defined as in Equation 4.3 [77].

$$T(x,y) = \sum t_{i,j}x^i y^j \tag{4.3}$$

In Equation 4.3, $t_{i,j}$ represents the number of spanning trees of the network whose internal activity is i and external activity is j. The summation is over all the subgraphs in the network [77].

4.1.2.3 Basic method

While this polynomial can be computed using Equation (2) for the identical reliability case, our algorithmic procedure keeps track of the individual link reliabilities. The resulting expression of the all-terminal reliability is an equation that takes the link reliabilities as arguments. Using an algorithmic procedure to create a symbolic representation of this polynomial, we can automate the algebraic expression for any arbitrary network N. This allows for computing the polynomial once per every network configuration. It is enough for any specific edge reliability values to replace the appropriate variables in the reliability polynomial to calculate the all-terminal reliability. As computing time grows with the number of edges, in our experiments, we limit our networks to at most 10 nodes and 20 edges with no parallel edges

between any two nodes. For the network topology presented in Figure 4.1, the reliability polynomial that represents the all-terminal reliability if all the identical links are identical is as in Equation 4.4.

$$RP(r) = 4r^5 - 11r^4 + 8r^3 \tag{4.4}$$

For a more general case with nonidentical links, the reliability polynomial is as in Equation 4.1.2.3. By substituting the link reliability values as shown in Figure 4.1 into this equation, we arrive at the same network reliability value of 0.9414 as we obtained earlier.

$$\begin{aligned} RP(r_{12}, r_{13}, r_{14}, r_{24}, r_{34}) = {} & 4r_{12}r_{13}r_{14}r_{24}r_{34} - 2r_{12}r_{13}r_{14}r_{24} - 2r_{12}r_{13}r_{14}r_{34} \\ & + r_{12}r_{13}r_{14} - 3r_{12}r_{13}r_{24}r_{34} + r_{12}r_{13}r_{24} \\ & + r_{12}r_{13}r_{34} - 2r_{12}r_{14}r_{24}r_{34} + r_{12}r_{14}r_{34} \\ & + r_{12}r_{24}r_{34} - 2r_{13}r_{14}r_{24}r_{34} + r_{13}r_{14}r_{24} \\ & + r_{13}r_{24}r_{34} + r_{14}r_{24}r_{34} \end{aligned} \tag{4.5}$$

4.1.2.4 Computational Algorithm

We have tested computing the polynomial using recursive and enumerative methods. The recursive methods rely on finding the subgraphs by contracting or removing edges in the network and applying the same procedure to each substructure until reaching disconnected or fully connected states while keeping track of the symbolic multiplications. The enumerative methods list all the possible states on which the edges can be configured, remove the ones that result in a disconnected network, and apply the appropriate operations on the reliability variables to obtain the polynomial.

In Algorithm 2, the *PossibleStates* are composed of arrays of zeros and ones that denote if the corresponding edges present or not. Each of these arrays is considered a Combination

Algorithm 1 Recursion-based Reliability Polynomial

Input $\leftarrow N = \{V = \{1, 2, \ldots, n\}, E = \{e_{ij}\}, R = \{r_{ij} = p(x_{ij})\}\}$
function RecursiveReliabilityPolynomial(N)
    if N is not connected **then**
        Output $\leftarrow 0$
    else if $|V| > 0$ **then**
        $e_{kl} \leftarrow$ First element in E
        $N_{contracted} \leftarrow N$ with e_{kl} contracted
        $N_{deleted} \leftarrow N$ with e_{kl} removed
        $RP_{contracted} \leftarrow$ RecursiveReliabilityPolynomial($N_{contracted}$)
        $RP_{deleted} \leftarrow$ RecursiveReliabilityPolynomial($N_{deleted}$)
        $RP_N \leftarrow r_{kl} RP_{contracted} + (1 - r_{kl}) RP_{deleted}$
        Output $\leftarrow RP_N$
    else
        Output $\leftarrow 1$
    end if
return Output
end function

and each combination is composed of states s_ij that represent if the edge is included in the configuration or not. As a connected network needs at least $n_{nodes} - 1$ edges, we filter those combinations that are guaranteed to lead to disconnected configurations before evaluation. The recursive algorithm is based on a similar approach designed for the case with identical links [78]. We have modified this procedure to account for the individual edge reliability values. The final algorithm keeps track of the individual edges. We use the enumeration-based version to validate our results. To further exploit reusing these polynomials, we use a NoSQL database based on MongoDB [79] to store the precomputed representations. To account for the potentially large equations, we also use GridFS for a distributed storage of files [80].

4.1.3 Reliability improvement using deep reinforcement learning

ANNs are based on the biological neural networks within the human body. Just like the brain, the components of ANNs work together in parallel and series to learn based on experiences. This learning occurs using a training set which is a set of inputs with known, target outputs.

Algorithm 2 Enumeration-based Reliability Polynomial

Input $\leftarrow N = \{V = \{1, 2, \ldots, n\}, E = \{e_{ij}\}, R = \{r_{ij} = p(x_{ij})\}\}$
$n_{nodes} \leftarrow |V|$
$n_{edges} \leftarrow |E|$
$PossibleStates \leftarrow \prod_{i=1}^{n_{edges}} \{0,1\}_i = \{0,1\}_1 \times \{0,1\}_2 \times \cdots \times \{0,1\}_{n_{edges}}$
$FeasibleStates \leftarrow \{Combination = \{s_{ij}\} \in PossibleStates : \sum Combination \geq (n_{nodes} - 1)\}$
$Terms \leftarrow \emptyset$
for all $Combination \in FeasibleStates$ **do**
    $N_{temp} \leftarrow \{V = \{1, 2, \ldots, n\}, E = \{e_{ij} : s_{ij} = 1\}, R = \{r_{ij} = p(x_{ij})\}\}$
    if N_{temp} is connected **then**
        $Result \leftarrow 1$
        for all $s_{ij} \in Combination$ **do**
            if $s_{ij} = 1$ **then**
                $Result \leftarrow r_{ij} Result$
            else if $s_{ij} = 0$ **then**
                $Result \leftarrow (1 - r_{ij}) Result$
            end if
        end for
    else
        $Result \leftarrow 0$
    end if
    Append $Result$ to $Terms$
end for
Output $\leftarrow \sum Terms$

In sequential decision-making, ANN can be used to create functional maps from system states or observations to the best action among a finite set of possible actions. In general, when the decision system is trained in a loop that assigns rewards to any of the actions taken, and the system learns the mapping from actions and observations to rewards, this is known as Reinforcement Learning (RL). When the function mapping the relationship between actions, observations, and rewards is an ANN, it is known as Deep Reinforcement Learning (DRL) [81].

4.1.3.1 Problem Formulation

For reliability improvement, this takes the form of deciding the best next edge to add to an infrastructure network to maximize the all-terminal reliability. When it is also possible to choose the quality of the new edges, the decision space grows. By considering cost constraints on the decision problem, the edge quality affects the reliability value and the added cost of the decision. Then, a finite sequence of edge decisions that will maximize the all-terminal reliability exists. Mathematically, the problem can formulated as follows:

$$\max_{\boldsymbol{\mathcal{A}}_t|\boldsymbol{\mathcal{O}}_t} \boldsymbol{\mathcal{R}}_t = \ln R_{network,t} - \ln(1 - R_{network,t}) + \lambda \boldsymbol{\mathcal{R}}_{t-1} \tag{4.6}$$

$$\boldsymbol{\mathcal{A}}_t = [x_{ij}, q_{ij}] \tag{4.7}$$

$$\boldsymbol{\mathcal{O}}_t = [x_{ij}, c_{ij}, C_{t-1}] \tag{4.8}$$

$$r_{ij} = p(x_{ij}, q_{ij}) \tag{4.9}$$

$$s.t. \tag{4.10}$$

$$\sum_i \sum_j c_{ij} x_{ij} = C_t \leq B \tag{4.11}$$

On each decision step t, the agent decides which set of actions $\boldsymbol{\mathcal{A}}_t$ will maximize the reward $\boldsymbol{\mathcal{R}}_t$ given the observations from the environment $\boldsymbol{\mathcal{O}}_t$. The reward is a function of the current all-terminal reliability and the value on the previous time step, discounted by a

factor λ. The actions include the new edge to add, x_{ij}, and its quality level q_{ij}. Observations include the edges already in the network, the cost associated with each edge in the network, c_{ij}, and the total cost of the network at the previous time step C_{t-1}. The budget constraint keeps the current cost of the network C_t within the the budget, B.

The current implementation uses the log-odds of the system being connected for the reward function: a transformation of the all-terminal reliability. It is worth pointing out that our initial experiments used the all-terminal reliability. We found more consistent performance using the negative log of the unreliability, and after further experiments, this led to using the log-odds of the system being connected. For actions, the options are the links not yet in the network and the quality level, with discrete options defining the edge reliability value. For the observations, the states, we propose the network topology, the cost of each link in the network, and the total cost of the current configuration. A cost constraint defines the budget for the added links limiting the number and quality of the added edges.

4.1.3.2 Implementation Framework

For the implementation, we base our training environment on the OpenAI-Gym framework [82]. This provides the basic elements to train and test DRL models. As there is a common interface for the models to train on, this allows for quick prototyping and testing.

Stable Baselines [83] is a set of DRL models that can be tested using the OpenAI-Gym interface. This grants access to a collection of algorithms that can be explored using an appropriate training environment. Each model is a different agent that can learn from the tuples of observations, actions, and rewards: striving to maximize the defined rewards while adjusting to the conditions posed by the environment, such as conditions for stopping and feasible actions.

4.1.3.2.1 Training Environment An environment requires four basic elements: observations, rewards, actions, and a way to evolve. The current environment starts with a path network with n nodes, and the $n-1$ links all have a reliability value of r_0, this makes the

initial all-terminal reliability r_0^{n-1}. Then, the possible actions are $(n^2-n)/2-(n-1)$ link options to add, with $q_{ij} = 1, 2, \ldots, m$, m is the number of quality levels, with $r_{ij} = 1-(1-r_0)^{q_{ij}}$, which is equivalent to considering each quality level to having q_{ij} basic links in parallel. This translates into each link cost as $c_{ij} = q_{ij}$. On each decision step, a DRL agent observes the state of the network, the connected links, the cost of each link, and the total cost. Then, it can choose one of the links to add, and one of the quality levels, if it is within budget. After adding the link, the reliability is computed from the corresponding polynomial and the different edge probabilities and the agent receives the associated reward. If the budget has not been exhausted, and there are feasible edges that can be added, the next decision step proceeds; otherwise, the episode stops.

4.1.3.3 Selected model

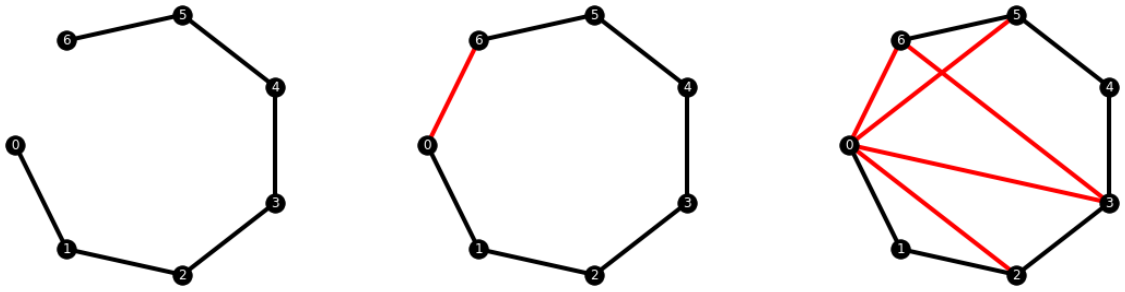
For our experiments, we work with a variant of Proximal Policy Optimization (PPO) [84]. PPO is a DRL model that explores decision policies in sets of actions that tries to balance the exploration of new decision policies with the optimization of a surrogate objective function. Specifically, it is a Policy Gradient method that limits itself to exploring points in a neighboring policy space by taking small incremental steps when the actions lead to an advantageous increase in rewards but is clipped, restricted to a neighboring range, when a disadvantageous direction is found [84]. This is designed to avoid stalling the decision in regions difficult to escape.

The variant used is a Maskable Proximal Policy Optimization (M-PPO) [85], an algorithm that considers the feasibility constraints posed by the training environment. For the formulated problem, this is equivalent to restricting the action space only to those links that are not yet in the network and are within budget. The M-PPO model uses a validity mask, a vector that keeps track of the valid actions, and operates it with the probability of taking a given action before updating the weights on each training step. This is useful to ensure the agent only learns to take feasible action and, for our problem of interest, guarantees that

the network reliability increases on every decision step.

4.1.4 Numerical examples

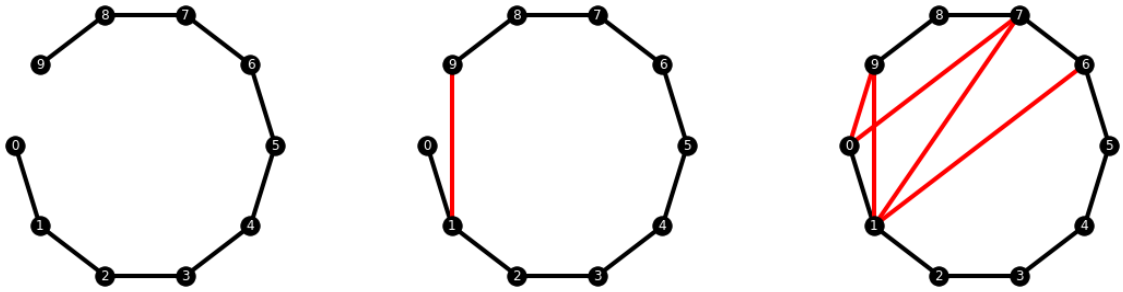
Experiments for different network configurations are conducted in this section. The results presented correspond to networks with $n = 7$ and $n = 10$ nodes. $r_0 = 0.8$, and there are $m = 3$ levels of edge reliability: 0.8, 0.96, and 0.992. The budget is set on $B = 5$, so at most five links can be added.



(a) (b) (c)

Figure 4.2: Results for the 7-node network with $B = 5$

Figure 4.2 shows the results for n=7. The first network (a) is the original configuration. The black edges represent the original $n - 1$ edges in the path network. The red edges represent those with $q_{ij} = 1$. For this case, the DRL agent only chose to add red links: it chose to maximize connectivity versus edge quality. The second network (b) is the configuration after one decision step and the third network (c) is the configuration at the final step. The all-terminal reliabilities are 0.26, 0.58, and 0.88 respectively. With the current approach, training the DRL agents while evaluating the reliabilities with no precomputed polynomials takes around 1.35 hours for 6144 training episodes of this experiment. This leads to an average of 0.8 seconds per training episode. The number of episodes was an arbitrary choice and further experiments are needed to decide the appropriate number of training steps, as well as to quantify the learning progress on the model. Further comparisons with baselines, such as total enumeration, are required to identify the optimality gap of the current approach.



(a) (b) (c)

Figure 4.3: Results for the 10-node network with $B = 5$

Figure 4.3 shows the results for n=10. The first network (a) is the original configuration. Again, for this case, the DRL agent only chose as many low-quality links as possible: maximizing connectivity. The second network (b) is the configuration after one decision step, and the third network (c) is the configuration at the final step, exhausting the budget. The all-terminal reliabilities are 0.1342, 0.3490, and 0.6722 respectively. With the current approach, training the DRL agents while evaluating the reliabilities with no precomputed polynomials takes around 4.36 hours for 6144 training episodes of this experiment. This leads to an average of 2.56 seconds per training episode. Similar to the previous experiment, more informed decisions about the number of episodes and procedures to performance are required.

4.1.5 Conclusions

The DRL method proposed in this paper enables reliability improvements of infrastructure networks. As a promising alternative to total enumeration and evolutionary optimization methods, the proposed method along with the use of reliability polynomial take advantage of machine learning capability in finding the best design of a general network with respect to all-terminal reliability. The polynomial computation is exact and challenging to scale, but as it only has to be computed once per network topology, it can be reused for different edge reliability values. This, combined with the permanent NoSQL database, allows for faster

training of the DRL agents. The M-PPO model for network reliability improvement is a data-driven approach that learns to solve the sequential decisions for the network topology while considering constraints on the feasible actions. For the experiments considered, it learns to optimize the choice of edges, maximizing connectivity, and quickly improving the all-terminal reliability of the networks of interest.

For future research directions, the computed polynomial can be used to generate datasets mapping network topologies and individual edge reliabilities to all-terminal network reliability. These datasets can then be used to train surrogate models capable of approximately estimating the network reliability. The DRL agents can also be used to sequentially improve the network reliability in scenarios where each link can degrade over time and eventually fail and become disconnected. The objective is now to maximize the network reliability while managing the new and degrading edges. For further complexity, we can include inspections, maintenance, and repairs of links among the set of actions. We are optimistic that DRL agents can handle this type of maintenance problem and be competitive in comparison with traditional process control methods.

4.2 Stochastic Variational Inference Neural Networks for All-Terminal Network Reliability

4.2.1 Introduction

Interconnected infrastructures provide critical services for the general population. Such critical infrastructures can take the form of power networks, communication networks, water networks, and transportation networks. Like other physical entities, these networks are prone to failure due to natural degradation and adverse events. An essential task in practice is to evaluate the reliability of an infrastructure network in terms of the connectivity of its components. Focusing on the probability that all components remain communicated with each other, the requirement can be conceptualized as the all-terminal reliability of the network [86].

Consider an n-node network (V, E) with edge topology $X = [x_{1,2}, \ldots, x_{i,j}, \ldots, x_{n-1,n}]$ with $x_{i,j} = 1$, if edge $e_{i,j}$ is present; 0, otherwise. Let $p(x_{i,j})$ be the reliability of edge (i, j), and Ω be all operational states with all nodes being connected. Then, the general equation for computing the all-terminal network reliability is [87]:

$$R = \sum_{X' \in \Omega} \left[\prod_{(i,j) \in X'} p(x_{i,j}) \right] \left[\prod_{(k,l) \in (X \setminus X')} (1 - p(x_{k,l})) \right] \tag{4.12}$$

The computational effort of exact algorithms for computing the all-terminal reliability of a network scales exponentially as the number of edges and nodes increases and eventually becomes prohibitive [88]. As a result, this computational step can become burdensome in applications that require repeated computations of all-terminal reliability for different network configurations. As an alternative to exact methods, approximate methods have been developed with varying degrees of precision. The gamut of such methods includes simulation-based methods such as Monte Carlo simulation [89], surrogate models like Artificial Neural Networks (ANNs) [87], and highly flexible models such as Deep Neural Networks (DNNs) supported by Graph Embeddings [87].

Beyond network reliability evaluation, improving the all-terminal reliability of a network is an essential task and can be implemented in different ways. These include, but are not limited to, Dynamic Programming and evolution-based methods [[90] such as Simulated Annealing, Ant Colony Optimization, and Artificial Bee Colony algorithm. By taking advantage of machine learning methods, it would be practically valuable and more efficient to tackle the network reliability improvement problem using machine learning.

In this work, our goal is to develop a machine learning-based framework for evaluating and improving all-terminal network reliability. In particular, the reliability improvement problem focuses on data-driven network design optimization, for which for an initial network structure, we determine the best sequence for adding links to maximize the metrics related to the all-terminal reliability over a finite time horizon.

It is worth pointing out that for our problem of interest, the complexity of the problem

is heightened by making the design space larger, as multiple link options of varying quality are considered. Additional practical constraints, such as budget constraints, pose tracking the feasibility of each action as an essential factor to consider in algorithm development. Given these, we consider the network design as a Reinforcement Learning (RL) problem. Furthermore, by using Deep Neural Networks to model network reliability, we frame the design problem as a Deep Reinforcement Learning (DRL) problem [91].

4.2.2 Related Work

With the extent of interconnected networks in everyday life, there is an essential need to understand network reliability and achieve better resilience in such critical infrastructures. Numerous studies in this area try to maximize all-terminal network reliability, and such problems are classified as NP-hard problems [88]. Indeed, in solving such problems, calculating all-terminal network reliability, and optimizing the network structure are of the most computational challenges.

Dengiz et al. [90] used a genetic algorithm to maximize a network's all-terminal reliability. Ramirez-Marquez and Rocco [89] suggested a novel algorithm based on hybrid optimization, which combines a probabilistic solution discovery with a Monte Carlo simulation to estimate the all-terminal reliability of the network. Recently, Goharshady and Mohammadi [92] suggested an innovative method for computing reliability for networks with small treewidth (as in subway networks), which can be scaled to networks with a higher count of vertexes.

Scholars have attempted to estimate network reliability by developing ANNs as surrogate models. Srivaree-Ratana and Smith [87] developed an ANN model that used link reliability and a reliability upper bound as inputs and computed the all-terminal reliability as its target value for a network with ten nodes. Davila-Frias et al. [93] proposed an approach to predict the network's reliability with varying graph sizes. They used different graph embedding methods to represent the network as the input of the DNN model. Recently, Davila-Frias and Yadav [94] addressed all-terminal reliability estimation using Convolutional

Neural Networks (CNNs). They defined a multi-dimensional vector representing the network adjacency matrix, link reliability, and topological attributes as the input of the CNN model. The output layer is a regression preceded by a sigmoid layer that predicts the network's reliability. This allows for exploiting the power of CNNs to identify correlations in the spatial patterns of the adjacency matrix that might be relevant for minimizing the loss function during training.

4.2.3 Methodology

DNN models are a powerful tool for creating a surrogate model for all-terminal reliability. We propose using a Bayesian variant of a DNN model to estimate the reliability of networks with vertex counts in a defined range. Exploiting the flexibility of this type of models to estimate all-terminal reliability will be balanced by the regularization capabilities of the Bayesian component in the inference procedures.

A sequence of subtasks will be tackled before obtaining a viable surrogate model. We first create a dataset comprised of several sample networks with a relatively wide range of all-terminal and individual edge reliability values. Then, the all-terminal reliability is calculated using an exact state enumeration algorithm. Next, we create a tensor representation of these networks based on adjacency matrices and spectral analysis. As these tensors arise from 2-dimensional arrays, we consider they encode relevant spatial information that can be extracted. To exploit this in a data-driven fashion, we train a baseline CNN model to create two desired outputs: an initial estimation of the all-terminal reliability and a data-driven embedding of the sample networks. We then train a DNN model in a Bayesian framework using Stochastic Variational Inference with the CNN embedding as input. This procedure enables inference on the all-terminal reliability values while quantifying the uncertainty of the estimates through Bayesian reasoning by sampling from the posterior distributions using DNNs as learnable transformations conditioned on the observed data. The complete modeling pipeline from tensor representation to SVI+DNN is now the surrogate model.

One of the most valuable use cases for this surrogate model is to speed up network design optimization tasks. Furthermore, as this approximation is not as computationally expensive as those exact methods, we can use iterative optimization methods and take sequential samples from the decision space. To take full advantage of this, we propose using a Deep Reinforcement Learning framework in a setup where it will actively learn from observed sequences of network designs while maximizing all-terminal network reliability.

4.2.3.1 Dataset Generation

Network graphs are generated with sizes of 8, 9, and 10 nodes with random numbers of undirected edges. We generate n graphs of each size for a total $N = 3n$ samples. We consider all nodes to be perfectly reliable, all edge failures are independent, and edge reliabilities can take varying values on pre-defined levels. Any given edge connecting two nodes is included with a random uniform chance defined by p_{add}. Link reliabilities are randomly assigned and uniformly chosen from a list of values r_{list}. For this dataset, all the graphs are connected: all nodes can communicate with each other, so their all-terminal reliability values are always non-zero.

The all-terminal reliability is first computed using an enumeration algorithm. This calculation is validated by another automated procedure developed to compute the reliability polynomial of an arbitrary graph [86]. Finally, the pairs of graphs and their exact all-terminal reliability values are stored for each sample. The generated sets of random graphs have varying numbers of edges and nodes, topology, and edge reliability configurations. This procedure was inspired by the work in [87] and [94]. However, more complexity was added by including multiple vertex counts in a single modeling pipeline and by making link reliability values vary both within a single graph and across graphs.

4.2.3.2 Tensor representation

For each of the generated graphs in the dataset, we created a tensor of size 10×10×3 comprised of three matrices of encoded information about the network structure. Each matrix is of size 10×10 as the largest vertex count in our samples is ten nodes. For graphs with 8 and 9 nodes, we added zeroes on the last 1 or 2 rows and columns correspondingly.

The adjacency matrix encoding the network edge and node structure is the first matrix. The second matrix is analog to the adjacency matrix but replaces the encoded values for each edge with their corresponding reliability values. We refer to this as the edge reliability matrix in this paper. The third matrix is the normalized Laplacian matrix. Given the adjacency matrix A and the node degrees in diagonal matrix form D, the Laplacian matrix L and normalized Laplacian matrix N are computed by:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{4.13}$$

$$\mathbf{N} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \tag{4.14}$$

We hypothesize that as this matrix encodes spectral information about the graph structure, it is a suitable input for a data-driven model with capabilities for exploiting spatial information. Therefore, the spatial information is encoded in the matrix arrays and will be used for the CNN embedding.

4.2.3.3 CNN embedding

Inspired by the work in [94], we use a CNN architecture for obtaining baseline estimation of all-terminal reliability. The following is a list of the sequential layers in this model:

- Convolution: 8 filters, 3×3×3
- Leaky ReLU

- Average Pool 2×2
- Convolution: 16 filters, 3×3×8
- Leaky ReLU
- Average Pool 2×2
- Convolution: 32 filters, 3×3×16
- Leaky ReLU
- Average Pool 2×2
- Convolution: 32 filters, 3×3×32
- Leaky ReLU
- Average Pool 2×2
- Fully Connected Layer (288,512)
- Leaky ReLU
- Fully Connected Layer (512,1024)
- Leaky ReLU
- Fully Connected Layer (1024,1)
- Sigmoid

We use a sigmoid function as the final activation to guarantee that the output is a real number between 0 and 1. We hypothesize that the three final fully connected layers work as a latent space embedding of the features extracted from the tensor representation.

The outputs of the second to last fully connected layer are used as a representation of the graph in the latent space. We refer to this as the data-driven CNN embedding in this

paper. This vector of size 1024, together with the baseline estimates, is used as the input for the SVI+DNN model.

4.2.3.4 SVI+DNN surrogate model

To create a surrogate model for calculating all-terminal reliability, we use a Bayesian framework to train the DNN model [95]. In addition, we selected the Pyro probabilistic programming framework [96] and used their proposed terminology to describe the procedures.

On the weights and biases of the layers of the DNN, we place probabilistic priors. Conditioning this on the observed data will lead to an intractable posterior formulation. To resolve the challenge, sampling methods such as Hamiltonian Monte Carlo (HMC) can be applied. However, such an approach using Stochastic Variational Inference (SVI) [97] This involves optimizing the Evidence Lower Bound (ELBO) using stochastic gradient steps, where the said lower bound is related to the Kulback-Leibler (KL) divergence of a proposed surrogate distribution function, called guide, and the true posterior. This proposed guide is based on the DNN model and a selected family of probability distributions. As the distribution family, we selected the Delta distribution, leading to Maximum a Posteriori estimates of the posterior distribution parameters.

The following equation shows the ELBO as an expectation with regards to the guide distribution:

$$ELBO = E_{q_\phi(z)} \left[\log p_\theta(x, z) - \log q_\phi(x, z) \right] \tag{4.15}$$

In Equation 4.2.3.4, observations are represented by x and latent random variables z. and $p_\theta(x, z)$ is their joint probability distribution with parameters θ. $q_\phi(x, z)$ is the guide distribution with parameters ϕ. Stochastic Gradient Descent is performed in the parameter space for ϕ to find a guide to decrease the divergence between the guide and posterior.

As this approximation would scale inefficiently for fully parameterized guides, we ex-

ploited the idea of amortization, which involves using a DNN to map the inputs to the required parameters in the guide model to reduce the number of trainable parameters. It is an exchange of parameters for a functional mapping.

This framework allows for estimating predictive distributions for all-terminal network reliability using the network CNN embedding and the baseline estimates as the inputs. The architecture of the model network is as follows:

- Bayesian Linear Layer (1025, 128)
- Leaky ReLU
- Bayesian Linear Layer (128, 128)
- Leaky ReLU
- Bayesian Linear Layer (128, 2)
- Sigmoid $\rightarrow$ (Output [1], Output [2])
- Normal Distribution
  - Mean: initial estimation + Output [1]
  - Variance: Output [2]

4.2.3.5 DRL for network design optimization

We use DRL to solve the sequential network design problem. The choice for DRL stems from the sequential nature of the design problem. Technically, feasibility constraints are factored-in by using Maskable Proximal Policy Optimization (M-PPO) as our optimization algorithm [98], which is a variant of Proximal Policy Optimization (PPO) that can take into account rule-based action feasibility.

We formulate the design problem with a reliability improvement objective. The DRL agent will decide the best next edge to be included in the graph for a given initial network to

maximize the all-terminal reliability. We enable discrete levels of edge quality with varying levels of cost and reliability. Moreover, we also consider a cost constraint such that a budget limits how many edges can be added. The agent must find a finite sequence of edge decisions that maximize its reward. Mathematically, the problem is formulated as:

$$\max_{\boldsymbol{\mathcal{A}}_t|\boldsymbol{\mathcal{O}}_t} \boldsymbol{\mathcal{R}}_t = \ln R_t - \ln(1 - R_t) + \lambda \boldsymbol{\mathcal{R}}_{t-1} \tag{4.16}$$

$$\boldsymbol{\mathcal{A}}_t = [x_{ij}, q_{ij}] \tag{4.17}$$

$$\boldsymbol{\mathcal{O}}_t = [x_{ij}, c_{ij}, C_{t-1}] \tag{4.18}$$

$$r_{ij} = p(x_{ij}, q_{ij}) \tag{4.19}$$

$$s.t. \tag{4.20}$$

$$\sum_i \sum_j c_{ij} x_{ij} = C_t \leq B \tag{4.21}$$

For all decision steps t. the DRL decides the actions $\boldsymbol{\mathcal{A}}_t$ to take given the observations $\boldsymbol{\mathcal{O}}_t$ to maximize the reward $\boldsymbol{\mathcal{R}}_t$ which depends on the all-terminal reliability R_t and a discount factor λ that balances the cumulative rewards. The available actions include adding a new edge x_{ij} with a chosen quality level q_{ij}. The quality level affects the cost of adding said link, c_{ij}, and the link reliability value. r_{ij}. The observations include the graph topology represented by the links in the network, the cost of each of the present links, and the total cost of the network up to the previous decision step C_{t-1}. The last equation shows the budget constraint to keep the total cost below B.

For implementation, we created an environment using the OpenAI-Gym [99] framework and trained the agents based on the Stable Baselines specified models [100]. The SVI+DNN surrogate model is used to approximate the all-terminal reliability to speed up the required computations.

4.2.4 Numerical Experiment

We conducted a numerical experiment by creating a dataset with $N = 6000$ randomly generated networks with vertex counts of 8, 9, and 10 nodes (i.e., $n = 2000$). Edge creation probability is $p_{add} = 0.3$. We considered five options for the levels of edge reliability: $r_{list} = [0.8, 0.85, 0.9, 0.95, 0.99]$. For model validation purposes, we separate these samples into "train," "validation," and "test" subsets, with proportions of 75%, 15%, and 10%, respectively.

We run our experiment on a Google Colaboratory Virtual Machine with 8 Intel Xeon processors with 4-Cores@2.2Ghz and 51GB of RAM. First, all the graphs in the sample were transformed to their respective tensor representation. This step took between 4ms and 6ms per graph. The initial CNN model was adjusted on the training set using an Adam optimizer to minimize an MSE loss metric. The learning rate was set on 0.002 following a cosine annealing schedule until 0.001. We use mini-batch updates of 128 samples during 1000 epochs. This procedure took 13m and 46s (i.e., 0.8s per epoch). This step output the initial estimation and the CNN embedding. For a new tensor representation, obtaining an initial estimation and embedding took between 4 and 8ms.

We trained the SVI+DNN model during 20 optimization steps, using mini-batch updates of 32 graphs and a ClippedAdam optimizer. The learning rate was set to 0.001. The training took 14m and 51s with around 40s per step for training and 5s per step to evaluate performance on the validation set. We sampled 100 times from the predictive posterior distribution for estimations and use the mean as the final prediction. Given a graph, executing the complete estimation from tensor representation to the surrogate model took around 200ms. For graphs of similar size, our implementation of the enumeration algorithm took 30s for an exact calculation of the corresponding all-terminal reliability.

Figure 4.4 shows the predictions on the training set (blue line) compared to the sorted actual values (black line). The shaded area around these lines represents the 90% credible interval for the posterior estimates. One can see that the predictions are generally close to

the true values. For example, the RMSE values of the predictions on the train, validation, and test sets are 0.003, 0.08, and 0.08, respectively. Compared to the results presented in the related studies [87], our proposed methodology achieves adequate precision in tackling a more difficult task of estimating all-terminal network reliability for sets of graphs with varying node counts and edge reliability values on a single modeling pipeline.

Figure 4.4: Prediction comparison

After completing the surrogate model, the model was used to estimate the reliability of sequential graphs during the training of our M-PPO agents. We defined a DRL environment with an initial path network with eight nodes. For the initial edges, the reliability was set to be 0.8. For any given decision step, the agent can add edges with one of three quality levels: 0.9, 0.95, and 0.99, and these edges have costs of 1, 2, and 3 units. Moreover, there is a maximum budget of 10 units of cost, and the agent stops adding edges when the budget is exceeded.

We trained the M-PPO agent during 5000 episodes, and it was completed in 24m and 42s. For a similar setup using a Reliability Polynomial for the all-terminal exact calculations, the observed training time was close to 16 hours.

The designed final network prioritizes connectivity over individual edge quality. This is observed from the exclusive use of links with the lowest quality and cost. The reliability estimated by the surrogate model is 0.9452, and the exact calculation is 0.9927, within the

Figure 4.5: Initial (left) and final (right) graphs

bounds of our validation error. Similar designs maximizing connectivity are also preferred by the M-PPO agent when the training is done using the exact computations in similar problem setups. Figure 4.5 shows the initial and final graphs for the best solution found after training, where the black links represent the existing edges initial network.

4.2.5 Case Study

To further illustrate the proposed methodology, we conduct a case study involving budget constraints. We focus our attention on the backbone computer network of the Gazi University in Ankara, Turkey [101]. The computer network with 11 nodes is shown in Figure 4.6. Three types of links can be chosen with individual edge reliability values of $[0.99, 0.995, 0.999]$. The quality level of each edge has an associated cost per meter of distance covered: [\$12, \$17, \$28].

Figure 4.6: Initial (left) and final (right) graphs for the Case Study

Before applying our method, some adjustments to the original formulation are made. First, to apply the current surrogate model, we merge one of the nodes in the network to

its neighbor for simplicity. Second, we assume that all the links in the original network are at the same quality level (set it at the lowest value). We use a budget value of $218,635, as in the solution with the lowest cost in the original formulation. Third, while the lengths of existing edges were provided in [101], to compute the distances not considered in the original formulation, we located the university buildings using Google Earth and measured the related distances so that the associated costs for adding new edges can be calculated.

We created a DRL agent to maximize the all-terminal reliability of the Gazi Network by adding new edges between nodes. The best solution in the original formulation [101] had an all-terminal reliability of 0.9945 and a cost of $322,865. This solution did not modify the network topology. Our best solution found has a higher reliability of 0.9998, and it is shown in Figure 4.6 as the final (right) network. The total cost is $199,140. It is worth pointing out that our solution added edges and modified the network topology. Moreover, our solution maximizes the number of edges by choosing the lowest quality level and adding edges with the lowest distances between pairs of nodes not yet in the network.

4.2.6 Conclusions

In this paper, we proposed a surrogate model to efficiently evaluate all-terminal network reliability. With the goal of improving the all-terminal reliability of a network, a DRL environment defining the action space, the per-period rewards, feasibility constraints, system evolution conditions, and state transition dynamics was developed for solving the network design problem. The practical value of this work is twofold. First, the Bayesian data-driven model is flexible enough to approximate the all-terminal network reliability of arbitrary graphs of varying sizes. It can work with varying edge reliability values and does not need additional computations as input, such as reliability upper bounds. The SVI framework allows for quantifying estimation uncertainty and incentivizing regularization to avoid overfitting. Second, enhanced and supported by the SVI+DNN approximation method, exploring the use of DRL for network reliability improvement provides a useful data-driven algorithm

capable of tackling complex design problems in ample design spaces.

Our future work will be focused on validating the network design solutions by diagnosing convergence and identifying suitable conditions. Indeed, validating such a data-driven model is challenging, but quantifying the optimality gap using total enumeration for small-scale problems is still possible. Furthermore, we have identified research opportunities for leveraging the estimation uncertainty naturally arising from the Bayesian model. We will exploit this by using other optimization methods such as Bayesian Optimization with Gaussian Process surrogate models or Thompson Sampling. In another direction, we will explore possibilities of balancing maximizing reliability with other objectives such as the equity across a network in humanitarian applications.

4.3 Dynamic Control using Deep Reinforcement Learning

4.3.1 Optimal Control for Dynamically Evolving Networks

Similar to the work presented in [4], where a surrogate model is developed and used to estimate the all-terminal reliability of an arbitrary structured graph and then applied to an optimization problem of network design to maximize the reliability metric, a similar research thread is developed in [10]. In this work, the first author developed a surrogate model based on Graph Neural Networks (GNNs) to estimate an arbitrary network's reliability with great precision. This surrogate model is so powerful and versatile that it shows accurate capabilities of interpolation and extrapolation in the context of incomplete data.

Furthermore, an optimal control problem is established for dynamically evolving networks, and the ability of the surrogate GNN model to efficiently generate approximations for the all-terminal network reliability is combined with a Maskable Proximal Policy Optimization (M-PPO) DRL agent that controls a specified set of actions to maximize the all-terminal reliability. The online monitoring and control of actions make the action space large and combinatorial in nature, making DRL a reasonable approach. This, combined

with the dynamically evolving network due to the edges aging and failing as time progresses, makes the DRL decision-making system relevant as it can react to changes in the network structure by learning from previous observations of the system evolution and reactions to control actions. This author's contribution to the work in [10] entails the simulation of dynamically changing edges in the network; the DRL environment to generate observations and rewards as outputs of actions that the agent takes; and the training and monitoring procedures to create and evaluate the M-PPO models. The problem and some numerical results are shown in the following sections.

4.3.2 Problem Description

We define four main elements of the control problem in dynamically evolving networks and formulate them in the context of DRL: evolution dynamics, action space, observation space, rewards, and termination condition. The network evolves on discrete simulation steps, and the agent can take actions on regular cycles after a pre-specified number of simulation steps. Call this mission cycle. During these mission cycles, the system evolves freely until the agent is allowed to take another action. For now, these cycles are pre-specified and fixed; it is a condition given in the problem. For future work, controlling the timing of these mission cycles could be included as part of the action space. A budget constraint on the number of actions taken is considered, and different costs for different actions can be included. Also, if these actions depend on a weighting variable for each edge (e.g., distance), this can be included in the current version. An initial budget and a replenishment of this budget during each mission cycle are included in the current formulation.

The system dynamics represent how the elements in the network change over time. For this work, the edges in the network age on each simulation step. The edges are considered as a binary state single failure mode element, and the time-to-failure is modeled using a Weibull distribution with individual and independent parameters for each edge. Using a three-parameter Weibull distribution with scale η, shape β, and location γ. These parame-

ters are all initialized with the same values for all edges, with $\gamma = 0$ in all examples, but can be changed by specific actions depending on the allocation of sequences that the DRL agent performs. During each time step, the edge status is sampled using the conditional probability of failure for an additional time step. Feasibility of actions is tracked using edge status, available edges to include, and budget constraints. For future work, considering interdependent parameterization between edges could be a valuable research thread to explore.

The action space represents the set of control activities the agent can undertake on a given action step, and for now, it is limited to one action per mission cycle. These actions include doing nothing, adding, maintaining, improving, and repairing edges. Doing nothing is just a placeholder for not taking any other actions. Adding edges implies changing the network structure by including a connection between two nodes that was not previously present. This connection is initialized at age zero, with default parameterization for the Weibull distribution. Edges included in the network can now be affected by the appropriate actions. Maintaining edges applies to edges already in the network and in operational status. Maintenance resets the selected edge's age and multiplies the β parameter by a factor of $\delta_m > 1$, and now the component is more prone to failure due to an increase in the failure rate. Improving an edge is also an action that is available for edges included in the network and with operational status. This action effectively increases the Mean Time to Failure (MTTF) of the selected age by modifying the η parameter by a multiplier $\delta_i > 1$. Repairing an edge is only available for edges already included in the network but with failed status. This action restores the edge to operational condition with age zero and returns it to the initial parameters. If the edge has been improved before, it keeps the improved η parameter.

The observation space represents all the elements in the system status that are available for the M-PPO agent to use as inputs to a learned policy or to a surrogate value function. The current version includes the following observations for each simulation step: network structure, the age of the edges, the reliability of each edge, the estimation of all-terminal reliability, the current budget, the current accumulated cost, the cost of each action per

weight, the weight of each edge, and the current reward. These observations are the inputs to the deep learning networks in the agents' training scheme.

The reward function is a transformation of the all-terminal reliability into a mapping that sets all values between -1 and 1. For reliability values below 0.5, the reward is negative, and it gets closer to 1 as the reliability gets closer to one. The current functional form chosen has the advantage of larger derivative values as the reliability value is closer to 0.5, quickly incentivizing network reliabilities with larger values than 0.5 in the deep learning training scheme. Empirical advantages of using this mapping versus the network reliability value have been observed, although further numerical comparisons and mathematical justifications are still in development. The surrogate GNN model is used on all action steps to approximate the all-terminal reliability value for the computation of the reward function, similar to the work presented in [4]. Therefore, there are as many calls to the surrogate model as action steps. The current termination condition is just completing a pre-specified number of mission cycles.

4.3.3 Problem Formulation

The following is a mathematical formulation of the Optimal Control for Dynamically Evolving Networks problem. It includes the action sequences $\boldsymbol{\mathcal{A}}_{1:t}$, the observation sequences $\boldsymbol{\mathcal{O}}_{1:t}$, and the cumulative rewards $\boldsymbol{\mathcal{R}}_t$.

$$\max_{\boldsymbol{\mathcal{A}}_{1:t}|\boldsymbol{\mathcal{O}}_{1:t}} \boldsymbol{\mathcal{R}}_t = \tanh\left[\ln R_t - \ln(1-R_t)\right] + \lambda \boldsymbol{\mathcal{R}}_{t-1} \tag{4.22}$$

$$= \frac{1 - \left(\frac{1-R_t}{R_t}\right)^2}{1 + \left(\frac{1-R_t}{R_t}\right)^2} + \lambda \boldsymbol{\mathcal{R}}_{t-1} \tag{4.23}$$

$$\boldsymbol{\mathcal{A}}_{1:t} = \{\mathcal{A}_k\}_{k=1}^{t} \tag{4.24}$$

$$\mathcal{A}_k = \left[A_{m,ij}\right]_k, m \in \text{Actions} \tag{4.25}$$

$$\text{Actions} = \{\text{nothing}, \text{add}, \text{maintain}, \text{improve}, \text{repair}\} \tag{4.26}$$

$$A_{m,ij} \quad \in \{0,1\} \tag{4.27}$$

$$\sum_{ij} \sum_{m \in Actions} [A_{m,ij}]_k \quad =1, \forall 1 \leq k \leq t \tag{4.28}$$

$$\boldsymbol{\mathcal{O}}_{1:t} \quad = \{\mathcal{O}_k\}_{k=1}^{t} \tag{4.29}$$

$$\mathcal{O}_k \quad = [O_l]_{l=0}^{p} \tag{4.30}$$

$$r_{ij} \quad =R(\tau_{ij}|\eta_{ij}, \beta_{ij}) \tag{4.31}$$

$$s.t.$$

$$C_k \quad =\sum_{ij} \sum_{m \in \text{Actions}} c_m w_{ij} [A_{m,ij}]_k, \forall 1 \leq k \leq t \tag{4.32}$$

$$B_t \quad =B_0 + tb - \sum_{k=1}^{t} C_k \tag{4.33}$$

$$C_k \leq B_k, \forall 1 \leq k \leq t \tag{4.34}$$

In this formulation, actions are represented as binary variables representing the action type selected and the edge affected. This is captured in the $A_{m,ij}$ variables that reflect the action type, m, and the edge affected ij. The observations are represented by a collection of O_l variables in p total observations. In this formulation, the reliability of each edge r_{ij} is a Weibull function of the age of the edge τ_{ij} and the individual parameters η_{ij} and $beta_{ij}$. In the cost evaluation, c_m represents the unweighted cost of each action, w_{ij} is the weight of each edge and C_k represents the total cost on each action step. B_0 is the initial budget and b is the replenishment value that is added to the budget on each action step.

4.3.4 Numerical Results

Simulated results are presented for a sample network with 12 nodes and a case study using the structure of an actual communication network with 11 nodes, including spatial locations for the nodes and, in consequence, distance weights for the possible edges. The 12 node network is initialized as a path network with 11 edges. The case study network is initialized to the configuration encountered in the original source material [10]. For both examples,

the dynamics for the Weibull distributions are set using $\beta_{ij} = 2$, and $\eta_{ij} = 400$, and mission cycles of 10 time steps. For the 12 node network, this makes it so that the all-terminal reliability after the first mission cycle is close to 0.5.

For the 12 node network, the action costs are 2 units for maintenance, 5 units for improvements, 10 units for adding an edge, and 7 units for repairs. The initial budget is set to 50 units, enough for adding 5 additional edges. The budget replenishment per action step is 5 units, enough for one improvement per each action step. The current mission cycle is of 10 time steps and the termination condition is 10 mission cycles.

Figure 4.7: Example 12 node network after 500 (left) and 999 (right) simulation steps

Figure 4.7 shows a comparison of a sample replication after 500 and 999 simulation steps. The results of training an M-PPO agent over 10 replications and testing it over 100 replications are shown in Figure 4.8. On each plot in this figure, the solid lines represent the mean value across replications, the dotted lines represent the median across replications, the lighter shaded area represents a 95% variation interval across replication, and the darker shaded areas represent a one standard deviation around the mean variation. The top left plot represents the evolution of the reward values, and the bottom left represents the evolution of the estimated all-terminal reliability. The top right plot represents the evolution of the available budget. The bottom right figure shows the cumulative number of actions taken on each action step, each color representing a different action. The current results show a consistent increase in all-terminal reliability with a median closer to 1 as the time steps progress.The budget is not completely exhausted in most cases, but it stabilizes at around 10 units. The actions are consistently composed of additions in the early action steps, with

a constantly growing number of maintenance and improvements, and a later increase in the use of repairs. This would make intuitive sense as the repairs become necessary as edges age out.

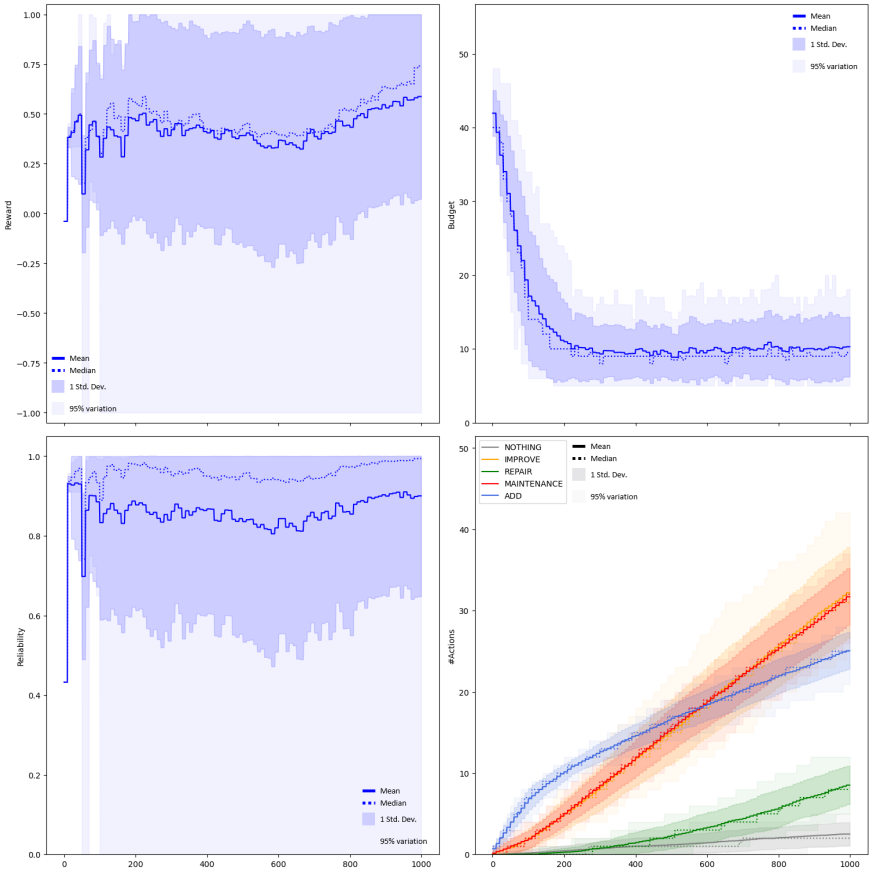


Figure 4.8: Example 12 node network: results for 100 replications

For the case study node network, the action costs are the same as in the 12 node network, but now costs are affected by the weight of each edge and this is proportional to the distances between nodes. The initial budget is set to 50 units times the mean distance value between all nodes, this is enough for adding 5 additional average size edges. The budget replenishment per action step is 5 units weighted by the mean distance between nodes, enough for one average sized improvement per each action step. The current mission cycle is of 10 time steps and the termination condition is 10 mission cycles.

Figure 4.8 shows a comparison of a sample replication after 500 and 999 simulation steps. The results of training an M-PPO agent over 10 replications and testing it over 100

Figure 4.9: Example case study network after 500 (left) and 999 (right) simulation steps

replications are shown in Figure 4.10. On each plot in this figure, the solid lines represent the mean value across replications, the dotted lines represent the median across replications, the lighter shaded area represents a 95% variation interval across replication, and the darker shaded areas represent a one standard deviation around the mean variation. The top left plot represents the evolution of the reward values, and the bottom left represents the evolution of the estimated all-terminal reliability. The top right plot represents the evolution of the available budget. The bottom right figure shows the cumulative number of actions taken on each action step, each color representing a different action. The current results show a consistent increase in all-terminal reliability with a median closer to 1 as the time steps progress.The budget is not completely exhausted in most cases, but it seems to remain above 2000 units in most cases. The actions are consistently composed of additions in the early action steps, with a constantly growing number of maintenance and improvements, and a later increase in the use of repairs. This would make intuitive sense as the repairs become necessary as edges age out.

4.3.5 Conclusions

The current results show a promising outlook for using DRL models in the context of optimal control of dynamically evolving networks. Assisted by the developed surrogate model for the estimation of all-terminal reliability, the current model manages to keep the reliability of the network in high values through the use of timely maintenance, repairs, edge additions, and improvements. This extension of the network design problem, instead of maximizing

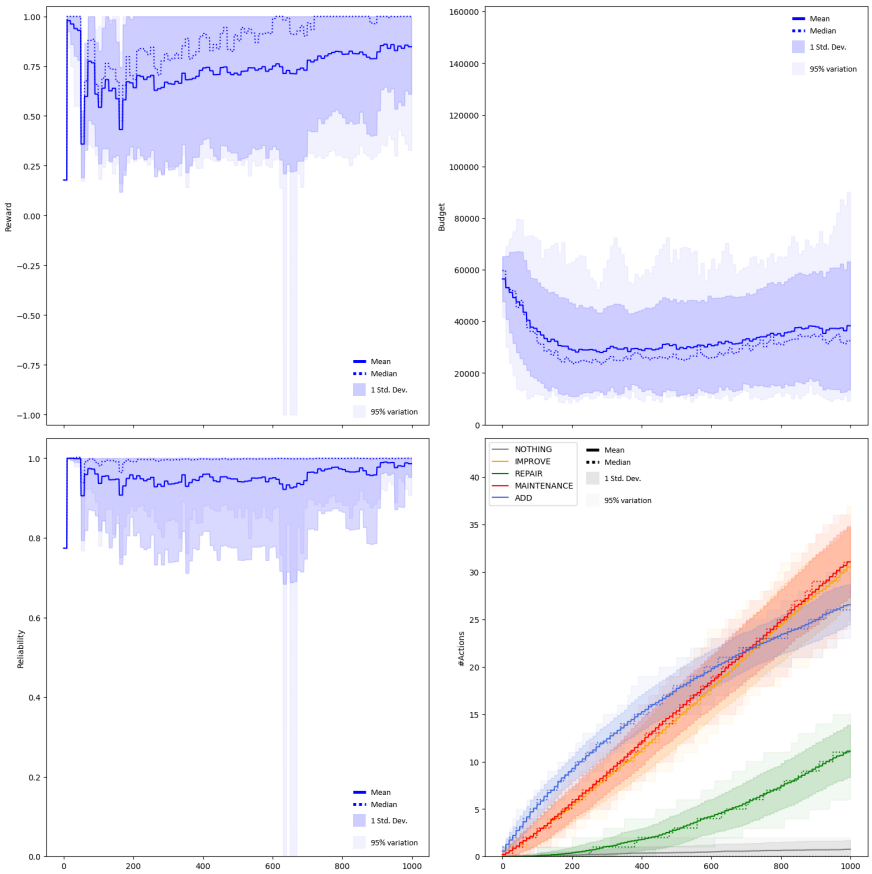


Figure 4.10: Example case study network: results for 100 replications

the static all-terminal reliability of a given network configuration, to control the dynamic evolution of the network over multiple mission periods poses as a valuable application that can be extended to different domains. These developments explore the promising outlook of data-driven decision making tools for responsive maintenance and control in the context of interconnected infrastructures.

4.4 Need-Based Sampling for All-Terminal Reliability Models

4.4.1 Motivation

Following a similar setup to [4], consider the case for interconnected infrastructures that provide critical services for the general population. However, like other physical entities, these networks are prone to failure due to natural degradation and adverse events. Therefore,

finding the reliability of a network is a crucial task necessary to understand the network's resiliency. One of the well-known measurements in this area is all-terminal reliability.

All-terminal reliability is the probability that all the nodes remain interconnected in a network. However, calculating all-terminal reliability is a computationally intensive problem; we used a data-driven surrogate model to approximate the all-terminal reliability computation. However, the performance of such data-driven models is highly dependent on the observed data and network complexity. Therefore, it becomes necessary to devise a framework capable of adaptively collecting samples to improve the accuracy of the estimations. Exploiting the estimation of uncertainty associated with the current state of the Bayesian Neural Network might be a key element to achieving this. Since a data-driven model depends on the selected training samples, when a surplus of data points is available, sieving appropriate training samples influences the model's estimation performance. In this work, we consider using additional samples to augment the observations used to fit a preliminary model and develop a method for effective improvement through active learning under a budget constraint. This approach is referred to as "need-based sampling." One of the main questions of this thread of research is, given a finite budget to collect additional samples, how can we ensure that the extra samples grant the surrogate model an improved estimation of the reliability function? An initial outline is provided in the following sections.

4.4.2 Conceptualization

Consider sequentially sampling from a generator of random connected graphs with a specific number of nodes, a range of edge densities, and associated edge reliability values for each edge. We will use these samples to augment the observations used to fit a data-driven model and thus develop a method for effective improvement through additional data points. We define a finite budget for these extra samples and quantify the usefulness for improving our estimation of the reliability function. Said usefulness metric will be defined in section 4.4.4.3. Moreover, we initially define a data-driven acquisition function based on similarity criteria

to decide if a given candidate sample will be included in the augmented data set.

We collect an initial sample of random graphs to initialize our active-learning procedure and generate additional samples without evaluating their all-terminal reliability unless the sample is deemed necessary for training. If a new sample is selected, it is added to the training set, and the model is adjusted until the budget for additional samples is exhausted. This results in expanding our observation space to include characteristics not present in the initial sample and improving the performance of our surrogate model.

Algorithm 3 Need-based Sampling for Surrogate Reliability Models

Precondition: $\mathcal{D}_0 = \{(G_i, R_i)\}_{i=1}^{n_0}$ ▷ Initial sample
Precondition: $\tau = \left[\{r_k\}_{k=k}^{K}, \{n_k\}_{k=k}^{K}, p\right]$ ▷ Generator Parameters
Precondition: $\mathcal{G}(\tau)$ ▷ Graph generator function
Precondition: $f^{\mathcal{R}}(\cdot)$ ▷ Exact Reliability Evaluation function
Precondition: $\hat{f}^{\mathcal{R}}(\cdot)$ ▷ Surrogate Model
Precondition: $\mathcal{E}(\cdot)$ ▷ Model Performance Evaluation function
Precondition: $\mathcal{A}(\cdot)$ ▷ Acquisition function
Precondition: $n_0 > 0$ ▷ Initial sample size
Precondition: $n_b > 0$ ▷ Budget for additional samples

$j \leftarrow 0$. Initialize sample indexes
$\hat{f}_j^{\mathcal{R}}(\cdot) \leftarrow \hat{f}^{\mathcal{R}}(\mathcal{D}_0)$. Initial model fit
$\mathcal{A}_j(\cdot) \leftarrow \mathcal{A}(\mathcal{E}(\cdot), \hat{f}_j^{\mathcal{R}}(\cdot), \mathcal{D}_j, n_b)$. Define initial acquisition function
while $j \leq n_b$ **do**
    $G_j \leftarrow \mathcal{G}(\tau)$. Get one sample graph
    $a_j \leftarrow \mathcal{A}_j(G_j)$. Acquisition decision
    if $a_j = 1$ **then**
        $R_j = f^{\mathcal{R}}(G_j)$. Evaluate reliability
        $\mathcal{D}_j \leftarrow \{\mathcal{D}_j, (G_j, R_j))\}$. Update data
        $\hat{f}_j^{\mathcal{R}}(\cdot) \leftarrow \hat{f}_j^{\mathcal{R}}(\mathcal{D}_j)$. Update model
        $E_j \leftarrow \mathcal{E}(\hat{f}_j^{\mathcal{R}}(\cdot), \mathcal{D}_j)$. Evaluate model
        $j \leftarrow j + 1$
    end if
end while

In Algorithm 4.4.2, $D_0 = \{(G_i, R_i)\}_{i=1}^{n_0}$ is the initial sample from the available data, with n_0 as the total number of observations in this sample, composed of graphs G_i with associated all-terminal reliability values R_i. τ represents the set of parameters in the graph generator; this defines the population of graphs from which we can sample.$f^{\mathcal{R}}(\cdot)$ is the function to

evaluate all-terminal reliability, usually computationally costly. $\hat{f}^{\mathcal{R}}(\cdot)$ is the surrogate model to be trained. $\mathcal{E}(\cdot)$ is a function to evaluate model performance, such as the mean squared error or any comparison of the reliability values predicted by the surrogate model and the true values. $\mathcal{A}_j(\cdot)$ is the function that will decide if any new sample should be acquired into our pool of available samples based on the value that this potential sample represents for improving the model performance.

4.4.3 Literature Review

Current search terms and threads have not returned any other results with similar goals to the current endeavor of sequential need-based sampling. However, terms like "adaptive augmentation", "category discovery," "representation learning," and "deep clustering" have returned various interesting works that do not share the same objectives but could be implemented as building blocks of a sequential need-based sampling scheme. The following paragraphs explore some of these ideas and outline how these could be used in a need-based sampling framework.

4.4.3.1 Adaptive Augmentation

The work presented in [102] aims to overcome data deficiencies and make robust deep-learning models while facing noisy data. They propose an adaptive augmentation of observations to improve selected performance metrics. The focus is on protecting the model performance from adversarial attacks using noise in image classification tasks. This is very close to the current concept of need-based sampling because they are trying to improve the models by augmenting the observation space. However, it also diverges from our current objective as it is not focused on gaining new observations on a constrained budget but rather on exploiting variations in the current data to gain robustness in unexpected variations on unobserved data and even malicious adversarial examples. The authors use an iterative pipeline of classification and re-classification with a measure of "Peak Signal to Noise Ratio" to quantify

the similarity of images and then decide if they should be included in the training data or the augmented set. Using signal-to-noise metrics to quantify similarity could be exploited in our need-based sampling framework.

4.4.3.2 Representation Learning and Clustering

Using unsupervised learning to identify categories in the data while evaluating representation learning loss constitutes a joint approach to multiple tasks using the same data. The work in [103] presents a model following this idea. The authors introduce a framework that learns to cluster in the forward pass and does representation learning in the backward pass. Then, they use the learned clusters and representations to improve performance on supervised tasks. This idea could be applied to the need-based sampling work by doing clustering and representation learning simultaneously, then using these classes in the supervised task of all-terminal reliability estimation. After this, whenever there is a sample of a new network structure, classify it before evaluating reliability and use it for learning if it is not part of any of the predominant clusters or if the performance in the corresponding cluster is poor.

4.4.3.3 Discovering Categories

Ideas from category discovery could be used similarly to the previous subsection. By discovering categories from the data, we can choose to oversample the categories that are not predominant in the original training data. The work of [104] discovers categories from the data using self-supervised learning and adaptive prototypes. They use a feature extractor to create the prototypes and then use self-supervised learning to refine the categories. This idea could be applied to need-based sampling to discover classes from the data using the prototypes and then oversample the less predominant classes.

The work in [105] uses a similar idea by discovering categories using deep embeddings for clustering. Learning both representations and clusters in a single model. They use KL divergences for the clustering objectives. Again, this could be used to discover categories

from the data and then oversample the categories that are either not as predominant or that, within that category, have poorer performance metrics.

4.4.3.4 Deep Clustering

On a similar approach, creating clusters using deep learning is possible, such as in works like [106], where the focus is on doing feature learning and clustering assignments simultaneously. They use random transformations in the data to grant robustness to the features and clusters learned. By relating the unsupervised learning of clusters to supervised learning, they naturally incorporate data augmentation. They use an estimation of the clustering confidence to decide if further refinements in the classes are needed by gradually adding "easy" to classify examples. They name this approach "adaptive self-paced learning". Similarly to previous references, this could be used to create data-driven clusters and oversample the ones not predominant in the training set or those with lower performance metrics within a given cluster.

The work in [107] does representation learning and clustering simultaneously while preserving geometric features of the original data. Their method can be applied to generalized data and is not constrained to 2D images. This could be useful as geometric features are potentially relevant not only for clustering but could also be used in the all-terminal reliability estimation model as additional inputs. For example, these refined geometric features could replace the latent representation vector currently extracted from the CNN model.

4.4.4 Proposed Research Direction

4.4.4.1 Cluster discovery

Based on the literature, using a cluster discovery approach can lead to advantages. This would allow creating data-driven classes and categorizing each of the existing new samples into this unsupervised labeling scheme. Then, evaluating performance within and across clusters is a possibility. Also, by clustering using features from the network structure, it is

possible to apply the sequential sampling idea before evaluating the all-terminal reliability of the chosen samples.

4.4.4.2 Similarity Criteria

Based on the literature related to deep clustering, there seems to be a variety of effective approaches to measuring similarity. A wide array of choices exist, from KL divergences to simple Euclidean distances in latent spaces. A simple metric with a known range, like cosine distance for latent vector representations, can be explored for the current work.

4.4.4.3 Usefulness Metric

For initial experiments, we can define the usefulness metric of a given new sample considering a balance between exploration, exploitation, and innovation. Exploration would refer to acquiring samples from clusters with few observations in the training data. This is equivalent to preferring samples that belong to minority clusters. Exploitation would refer to selecting samples corresponding to clusters that have, as a group, poor performance. For example, using this criterion, sampling clusters with higher MSE in their observations would be preferred as it can potentially benefit the overall learning of the model. Finally, the innovation metric would refer to preferring new observations that, while belonging to already predominant clusters, are significantly different from observations already in the sample. This would target the within clusters variation and would be computed by using the similarity metric defined before. This would allow for learning to estimate the reliability function of network structures not observed, even if they are deemed to belong to clusters already explored.

4.4.4.4 Potential Advantages and Issues

For advantages, this author considers that attacking multiple tasks such as learning representations, discovering clusters, and fitting the estimations from the same data can have synergistic effects. Some of the representations can benefit the estimation of all-terminal

reliability, the discovered clusters can help refine the different estimations, and the learned estimation can further help to refine the clustering. However, as a data-driven approach, the risk of overfitting the training data is always present. Applying multiple tasks to the same data can lead to attempting to estimate a larger set of parameters from fewer observations. Balancing the number of additional weights and biases with the obtained data becomes relevant. Methods for regularization and estimation of uncertainties might be needed.

4.5 All-Terminal Reliability using Quantum Computing

4.5.1 Existing Algorithm

4.5.1.1 Motivation

The potential of using frameworks as novel and in the vanguard of research, such as Quantum Computing, has become a personal interest of this author in recent years. In particular, methodologies such as the work in [11] present ideas to combine Variational Inference with groundbreaking technologies such as Quantum Computing and simulation of quantum systems. While the complete understanding of the inner workings of these methodologies is beyond this author's current formation, efforts in this direction could lead to fruition as these novel methods are being tested in problems closely related to network design contexts. Specifically, it seems that problems related to large combinatorial spaces are good candidates to be tackled using quantum computing ideas. Furthermore, as these technologies come to maturity, the design of algorithms to implement them to optimize all-terminal network reliability might become valuable. Research efforts exploring this venue of ideas are presented in the following section.

4.5.1.2 Paper Review

The authors in [108] develop an algorithm to estimate network reliability using quantum computation. Their method can be generalized to k-terminal network reliability, although

the fully developed concept is for all-terminal network reliability. In this article, the authors start by defining all-terminal network reliability and classifying the computation of this metric as a #P-complete problem.

For the goals of this article, their target is developing an algorithm exploiting the properties of quantum bits, *qubits* [109], in a computational circuit to perform network reliability calculations with advantages over algorithms using classical computation. The main advantage of using qubits is that they are, in essence, analog units of computation that could be considered, as per my current understanding, as physical manifestations of probability distributions over defined states. In this sense, a qubit might encode multiple states simultaneously in what is known as *superposition*. Superposition is a qubit with some defined probability in various states simultaneously. Applying operations to multiple qubits in a quantum circuit, as they might exist in superposition states, the operations are applied to all combinations of states simultaneously. Measurement of qubits is the step in the circuits that reveals which states are available, and it is an intrinsically probabilistic result: the result of a measurement depends on the probabilities (in the quantum context, probability *amplitudes*) associated with each state and it will reflect these probabilities over repeated measurements.

The authors outline their algorithm as a quantum circuit composed of *gate* operations. In the physical implementation of quantum circuits, these gate operations represent interacting with the qubits, usually in the form of rotations along specified axes. By expressing their algorithm as a set of gate operations, they effectively describe a quantum circuit that can be implemented in modern quantum computation libraries such as [110].

In this author's opinion, the most clever part of the defined circuit is the definition of a network *reachability* operator implemented as a set of quantum gates. It allows for translating the logic of a connected network using qubits. After expressing this for all connections in a given graph, the authors propose using an *oracle* that verifies if the network is connected by checking that all nodes are reachable from an arbitrarily chosen starting node.

The authors define another clever element in their algorithm as the next step. They connect a *Grover Operator*, a quantum circuit that amplifies the probabilities of an oracle state to be observed. Using this operator, they search among all possible combinations of states of the defined qubits for those representing a connected network. After this, they proposed applying repeated measures on a "label" qubit, a qubit configured to have a value of 1 if the network is connected after applying the Grover search algorithm. This allows for estimating the all-terminal network reliability by finding the probabilities associated with this label qubit being measured with a state of 1.

By using repeated measures of an intrinsically stochastic set of states, this algorithm is similar to a Monte Carlo estimation of all-terminal network reliability with the added advantage that the Grover operator combined with the defined oracle allows for a guided search of the associated probability of the network being connected.

One of the most constraining limitations of the current state of quantum computers is the number of qubits available for computation and the noisy nature of the gate operations. For example, the current algorithm needs $|E|+|V|+2$ qubits. For IBM cloud quantum computers with a maximum of 127 qubits, this implies that, at most, a 15-node fully connected network can be evaluated using a physical machine.

The authors in [108] do not provide experimental results of their method. However, this author developed their implementation of a modified version of this circuit using Qiskit [110] in Python. Through preliminary experiments, the results have verified that the algorithm produces good approximations in reasonable times, at least using simulators of quantum computers provided by IBM. Section 4.5.1.4 presents a brief discussion of the results.

4.5.1.3 Literature Review

The work in [108] presents a use case of quantum computing applied directly to network reliability estimation. At the moment of developing this review, using keywords and threads of connected references, I have not found another article discussing a method of the same

nature. However, algorithms inspired by quantum computing and using Monte Carlo simulation are abundant. Additionally, methods for redefining all-terminal network reliability into a quantum measure of network reliability are also available.

Following a similar research venue, some work applied to network reliability estimation or network design optimization has taken some ideas from quantum computing to improve existing methods. For example, quantum-inspired ideas to complement established modeling paradigms have found their niche.

The work presented in [111] is an overview of quantum-inspired ideas for Machine Learning applied to communication networks. This includes optimization, network design, and statistical analysis of multiple states using quantum computing. In addition, the authors state that promising research directions include hybrid algorithms combining classical and quantum-based algorithms.

Consider [112], where efforts are directed to assess the reliability of power systems. Using Monte Carlo Simulation, the authors developed a quantum circuit for estimating reliability indices in distribution systems.

The authors in [113] present a method to create bounds for network reliability assessment using quantum computing. Their method relies on using quantum computing principles to count network configurations accurately. They state that with the current state of the development of physical quantum computers, validating their methods is possible by using systems with tens of qubits.

The authors in [114] present the Internet of Things Quantum Computing Inspired Optimization (IoT-QCiO) concept. The proposition entails using many sensors and optimizing the data accuracy of their collected data by considering characteristics such as vicinity and spacing. They present a case study where they monitor traffic using sensors and inform decisions in a vehicle routing problem. The quantum computations are done using simulators.

The authors in [115] present ideas on reliability analysis using quantum dynamics, where they "explored the concept of the reliability theory in the quantum domain and formalized

the concept of quantum reliability (as opposed to classical reliability)". In their definition, quantum reliability is the evaluation of the quantum superposition of states. After they extended these ideas to repairable systems in [116] where they incorporate the ideas of time-dependent probability amplitudes that can be affected by repair and maintenance.

While all of these efforts provide some new ground to advance the use of quantum computing ideas in network reliability analysis, none of them is a direct application of a quantum computing circuit to estimate all-terminal network reliability. In this sense, [108] remains a unique algorithm among the identified relevant work.

4.5.1.4 Implementation and Numerical Examples

Using Qiskit [110], this author implemented an interpretation of the quantum circuit. A sample schematic for a small network can be observed in Fig. 4.11. However, it can only be categorized as an "interpretation" as some details of sub-components in the circuit are not fully expressed in the diagrams and equations and had to be inferred. IBM local and cloud simulators were used to compute all-terminal network reliability values for test graphs.

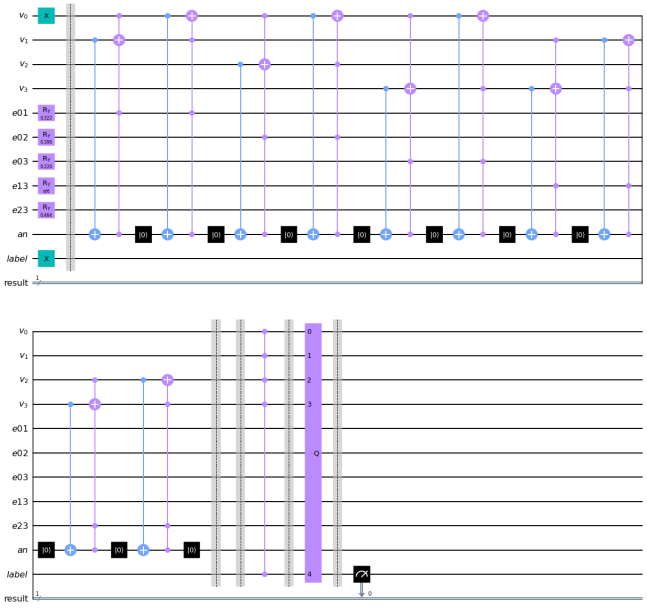


Figure 4.11: Sample circuit schematic in Qiskit

For example, consider a fully connected 7-node graph, as in Fig. 4.12, with edge reliability

values selected randomly from a predefined list. This leads to a total of $|E| = \frac{n(n-1)}{2} = 21$ edges with reliability values $r_{ij} \in [0.5, 0.75, 0.85]$. Implementing a total enumeration algorithm yields an exact calculation of 0.9987 in 14 minutes using Python on a Google Colab machine. Using one of the IBM cloud quantum computer simulators with 32 qubits, this reliability value is estimated at 0.9925 after taking 10,000 samples. The time for this computation is around 27 minutes.

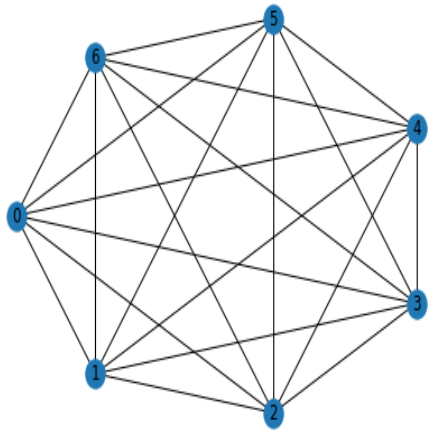


Figure 4.12: Fully Connected 7-node graph

In another example, consider a fully connected 12-node graph, as in Fig. 4.13 with edge reliability values selected randomly from a predefined list. This leads to a total of $|E| = \frac{n(n-1)}{2} = 66$ edges with reliability values $r_{ij} \in [0.5, 0.75, 0.85]$. The same total enumeration algorithm can not compute the value after 20 hours of computation. The Qiskit implementation returns a reliability value of 0.998 after taking 1000 measurements of the circuit. The time to complete this calculation is around 48 minutes using one of IBM cloud quantum computer simulators with 100 qubits.

4.5.1.5 Strengths and Weaknesses

Among the identified strengths of this method, I consider that the speed of the approximation is an advantage. It is surprising how accurate the estimates are in the simulators. However,

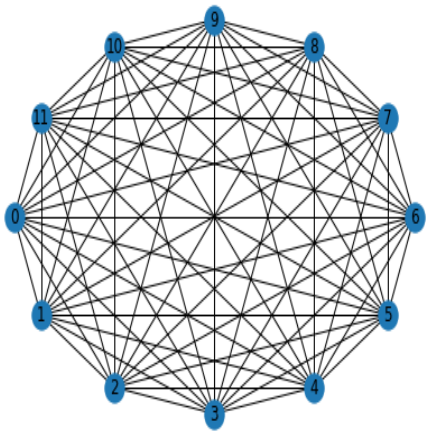


Figure 4.13: Fully Connected 12-node graph

the precision in a real quantum machine should be lower due to the noise inherent in the gate operations. I also consider that the fast progress in these technologies to achieve machines with more qubits and more reliable gate operations will make this approach viable as an estimation alternative in the near future.

Among the identified weaknesses of this method, this author considers that the number of qubits required is the most relevant restriction for applying it to large networks with real applications. At least when considering the use of simulators for calculations. However, in theory, the most significant speed-up would come from using real quantum systems. In that context, not only the number of available qubits is a constraint, but also the noise and cumulative errors in the gate operations. In real quantum machines, additional safeguards to guarantee the accuracy of the calculations are necessary. This could make this method's viability questionable if the objective is to differentiate it from other simulation-based methods.

One of the practical results from the observed experiments using Qiskit is that, although both nodes and edges require one qubit each to be represented, the size of the circuit grows faster with the number of nodes as the Grover Operator and Oracle are dependent on the number of nodes. This results in a larger number of gates being added to the circuit for each additional node, which can quickly become a restriction on resources while using simulators.

Likely, this will also become a bottleneck when using real quantum machines.

4.5.1.6 Research Directions

This work can be directly applicable to current research efforts in network reliability. Firstly, as an alternative estimation of all-terminal network reliability for moderate-size graphs using the Qiskit implementation. Second, The result from the quantum approximation can be used as an additional input for surrogate models, similar to using upper or lower bounds for the all-terminal reliability as part of the inputs; experiments on this research venue are currently under preparation. Third, the quantum circuit can be used as an oracle for Quantum Approximate Optimization Algorithms (QAOA). Adding another Grover operator to search for the best configuration of edges to maximize network reliability would also be a possibility, albeit the feasibility of this idea will be quickly constrained by the number of qubits required to express the different alternatives for adding arcs into a given network configuration. Even if the current technology is still in the very early stages, quantum computing applied to all-terminal network reliability looks like a promising research thread to be combined with the currently developed efforts.

5. Conclusions

This dissertation presented a framework for data-driven tools aiming to model and improve Interconnected Critical Infrastructures in multiple contexts. The importance of ICIs for daily human activities, and the large volumes of data being generated in the modern industries grants relevance to research efforts in this direction.

In Chapter 2, the impact of disruptions in Multimodal Transportation Networks is explored from an application perspective. The explored research threads combine simulation tools for decision-making with data-driven optimization paradigms to create tools that might provide stakeholders with optimal policies based on a wide array of scenarios and conditions. The flexibility of the developed simulation models together with cutting-edge technologies, such as DRL, sets the foundation for further promising research efforts on Inland Waterway Transportation Systems.

The exploration of data-driven models for condition monitoring and prognostics, in Chapter 3 focused on using Deep Learning for predicting the Remaining Useful Life of Turbofan engines using multiple sensors in sequential measurements as input. A myriad of approaches exists for this type of problem and the main contribution for future efforts might be centered around combining this type of data-driven methods with simulation tools and computational methods in contexts of network resilience optimization.

Chapter 4 presented methods for data-driven estimation of all-terminal network reliability for arbitrary graphs and outlined research directions for data-driven surrogate models. Furthermore, the use of DRL for network design optimization maximizing all-terminal network reliability was presented. This is a promising research venue that has been extended to network reliability problems that involve dynamic systems. Additional developments in this line of work are an active research interest of this author.

The outlined research presents various data-driven tools developed to collaborate in a data-driven fashion in the context of modeling and improvement for Critical Infrastructures. To achieve this goal, multiple research venues have been intertwined through the combination of a variety of paradigms and methods. The final product is a line of research focused

on reliability estimation, design optimization, and prognostics and health management for Interconnected Critical Infrastructures.

This author is optimistic in that expanding on the presented research threads will lead to unified efforts giving rise to further developments in the field of systems reliability and simulation of complex systems. The main contribution is making these various developed tools collaborate in a data-driven fashion. Multiple methods and paradigms have been combined to achieve a unified goal: the coalescence of the different research venues to create a data-driven framework for reliability estimation, design optimization, and prognostics and health management for Interconnected Critical Infrastructures, combining computational methods and theory.

Bibliography

[1] J. C. H. Azucena, B. Alkhaleel, H. T. Liao, and H. Nachtmann, "Hybrid simulation to support interdependence modeling of a multimodal transportation network," *Simulation Modelling Practice and Theory*, vol. 107, p. 102 237, 2021.

[2] J. C. H. Azucena and H. T. Liao, "Prognostic using dual-stage attention-based recurrent neural networks," in *Proceedings of the 11th International Conference on Mathematical Methods in Reliability (MMR)*, Hong Kong, Jun. 2019.

[3] J. C. H. Azucena, H. Wells, H. T. Liao, K. Sullivan, and E. A. Pohl, "Applying deep reinforcement learning to improve the reliability of an infrastructure network," in *Proceedings of the 60th European Safety, Reliability & Data Association (ESReDA) Seminar*, ser. Advances in Modelling to Improve Network Resilience, Grenoble, France, May 2022, pp. 46–55.

[4] J. C. H. Azucena, F. Hashemian, H. T. Liao, and E. A. Pohl, "Applying machine learning to improve all-terminal network reliability," in *Proceedings of the 69th Annual Reliability and Mantainability Symposyum (RAMS)*, Orlando, FL, Jan. 2023.

[5] T. Bipasha, J. C. H. Azucena, B. Alkhaleel, H. T. Liao, and H. Nachtmann, "Hybrid simulation to support interdependence modeling of a multimodal transportation network," in *Proceedings of the Winter Simulation Conference (WSC)*, National Harbor, MD, Dec. 2019, pp. 1390–1401.

[6] M. Aghamohammadghasem, J. C. H. Azucena, H. T. Liao, S. Zhang, and H. Nachtmann, "Preventive maintenance planning for an inland waterway transportation system using deep reinforcement learning," in *Proceedings of the IISE Annual Conference & Expo*, Accepted. Awaiting for the conference, New Orleans, LA, May 2023.

[7] M. Aghamohammadghasem, J. C. H. Azucena, F. Hashemian, H. T. Liao, S. Zhang, and H. Nachtmann, "System simulation and machine learning-based maintenance optimization for an inland waterway transportation system," in *Proceedings of the Winter Simulation Conference (WSC)*, Submitted. Awaiting for review, San Antonio, TX, Dec. 2023.

[8] R. Xin, C. Zhong, Z. Chen, T. Takagi, M. Seltzer, and C. Rudin, "Exploring the whole rashomon set of sparse decision trees," in *Neural Information Processing Systems (NeurIPS)*, 2022.

[9] J. Liu, C. Zhong, M. Seltzer, and C. Rudin, "Fast sparse classification for generalized linear and additive models," in *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, 2022.

[10] F. Hashemian, J. C. H. Azucena, H. T. Liao, and E. A. Pohl, "Machine learning-based reliability improvement of all-terminal networks," Forthcoming., 2024.

[11] D. Amaro, C. Modica, M. Rosenkranz, M. Fiorentini, M. Benedetti, and M. Lubasch, "Filtering variational quantum algorithms for combinatorial optimization," *Quantum Science and Technology*, vol. 7, no. 1, p. 015 021, Jan. 2022. DOI: 10.1088/2058-9565/ac3e54. [Online]. Available: https://doi.org/10.1088%2F2058-9565%2Fac3e54.

[12] J. Ellis, D. Fisher, T. Longstaff, L. Pesante, and R. Pethia, "Report to the president's commission on critical infrastructure protection.," Tech. Rep., Jan. 1997. DOI: 10.21236/ada324232.

[13] US Department of Transportation, *Geospatial at the bureau of transportation statistics*, available via https://maps.bts.dot.gov/arcgis/home/index.html, Washington, DC., 2017.

[14] US Army Corps of Engineers, *US Army Corps of Engineers: Navigation Data Center*, available via http://www.navigationdatacenter.us/, 2020. (visited on 03/03/2020).

[15] T. Bipasha, J. Azucena, B. Alkhaleel, H. Liao, and H. Nachtmann, "Hybrid simulation to support interdependence modeling of a multimodal transportation network," in *2019 Winter Simulation Conference (WSC)*, National Harbor, MD: IEEE, Dec. 2019, pp. 1390–1401. DOI: 10.1109/WSC40007.2019.9004813.

[16] R. Pant, K. Barker, and T. L. Landers, "Dynamic impacts of commodity flow disruptions in inland waterway networks," *Computers and Industrial Engineering*, vol. 89, pp. 137–149, Nov. 2015, ISSN: 03608352. DOI: 10.1016/j.cie.2014.11.016.

[17] P. R. Herr, "Approaches to mitigate freight congestion," Washington, DC, Tech. Rep., 2008, p. 12.

[18] Freight Research, "National cooperative freight research program: Current and completed projects," *Transportation Research Board*, 2010.

[19] US Department of Transportation, "Freight facts and figures 2009," Federal Highway Administration, Office of Freight Management and Operations, Washington, DC., Tech. Rep., 2009, p. 274.

[20] Cambridge Systematics Inc., "National rail freight infrastructure capacity and investment study," Cambridge, MA, Tech. Rep., Sep. 2007.

[21] US Department of Transportation. Maritime Administration, "America's marine highway report to congress," Washington, DC, Tech. Rep. April, 2011.

[22] J. Schweighofer, "The impact of extreme weather and climate change on inland waterway transport," *Natural Hazards*, vol. 72, no. 1, pp. 23–40, May 2014, ISSN: 0921030X. DOI: 10.1007/s11069-012-0541-6.

[23] S.-L. Wang and P. Schonfeld, "Scheduling interdependent waterway projects through simulation and genetic optimization," *Journal of waterway, port, coastal, and ocean engineering*, vol. 131, no. 3, pp. 89–97, May 2005, ISSN: 0733-950X. DOI: 10.1061/(asce)0733-950x(2005)131:3(89).

[24] J. L. Carroll and M. S. Bronzini, "Waterway transportation simulation models: Development and application," *Water Resources Research*, vol. 9, no. 1, pp. 51–63, Feb. 1973. DOI: 10.1029/wr009i001p00051.

[25] Y. Triska, E. M. Frazzon, and V. M. D. Silva, "Proposition of a simulation-based method for port capacity assessment and expansion planning," *Simulation Modelling Practice and Theory*, vol. 103, p. 102 098, 2020. DOI: 10.1016/j.simpat.2020.102098.

[26] R. Larson, A. Minkofft, and P. Gregory, *A computer simulation model for fleet sizing for the marine division of the New York City department of sanitation*. Elsevier, Aug. 1991, vol. 9, pp. 267–276. DOI: 10.1016/0734-242x(91)90017-2.

[27] J. Swedish, "Simulation of an inland waterway barge fleet distribution network," in *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, vol. 2, Washington, DC: IEEE, Dec. 1998, pp. 1219–1221. DOI: 10.1109/wsc.1998.745982.

[28] G. D. Taylor, T. C. Whyte, G. W. DePuy, and D. Drosos, "A simulation-based software system for barge dispatching and boat assignment in inland waterways," *Simulation Modelling Practice and Theory*, vol. 13, no. 7, pp. 550–565, Oct. 2005, ISSN: 1569-190X. DOI: 10.1016/j.simpat.2005.02.005.

[29] F. Oztanriseven and H. Nachtmann, "Economic impact analysis of inland waterway disruption response," *The Engineering Economist*, vol. 62, no. 1, pp. 73–89, Apr. 2017, ISSN: 0013-791X. DOI: 10.1080/0013791x.2016.1163627.

[30] G. Desquesnes, D. Alves, E. Duviella, G. Lozenguez, and A. Doniec, "Simulation architecture based on distributive MDP for inland waterway management," in *HIC 2018. 13th International Conference on Hydroinformatics*, G. L. Loggia, G. Freni, V. Puleo, and M. D. Marchis, Eds., ser. EPiC Series in Engineering, vol. 3, EasyChair, 2018, pp. 555–563. DOI: 10.29007/fbmt.

[31] M. Ouyang, "Review on modeling and simulation of interdependent critical infrastructure systems," *Reliability Engineering & System Safety*, vol. 121, pp. 43–60, Jan. 2014, ISSN: 09518320. DOI: 10.1016/j.ress.2013.06.040.

[32] C. W. Howe, J. L. Carroll, A. P. Hurter Jr, *et al.*, *Inland waterway transportation: studies in public and private management and investment decisions*. Baltimore: Johns Hopkins, 1969.

[33] M. D. Dai, "Delay estimation on congested waterways," English, Ph.D. dissertation, 1991, p. 146.

[34] C.-J. Ting and P. Schonfeld, "Optimization through simulation of waterway transportation investments," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1620, no. 1, pp. 11–16, Jan. 1998. DOI: 10.3141/1620-03.

[35] M. Ouyang, L. Hong, Z.-J. Mao, M.-H. Yu, and F. Qi, "A methodological approach to analyze vulnerability of interdependent infrastructures," *Simulation Modelling Practice and Theory*, vol. 17, no. 5, pp. 817–828, 2009. DOI: 10.1016/j.simpat.2009.02.001.

[36] A. Bush, W. Biles, and G. DePuy, "Iterative optimization and simulation of barge traffic on an inland waterway," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.03EX693)*, Winter Simulation Conference, Piscataway, New Jersey: IEEE, 2004, pp. 1751–1756. DOI: 10.1109/wsc.2003.1261629.

[37] W. Biles, D. Sasso, and J. Bilbrey, "Integration of simulation and geographic information systems: Modeling traffic flow on inland waterways," in *Proceedings of the 2004 Winter Simulation Conference, 2004.*, IEEE, 2005, pp. 331–337. DOI: 10.1109/wsc.2004.1371477.

[38] C. Colon, S. Hallegatte, and J. Rozenberg, *Transportation and Supply Chain Resilience in the United Republic of Tanzania, Assessing the Supply-Chain Impacts of Disaster-Induced Transportation Disruptions*. World Bank, Jun. 2019. DOI: 10.1596/31909.

[39] J. Verschuur, E. Koks, and J. Hall, "Port disruptions due to natural disasters: Insights into port and logistics resilience," *Transportation Research Part D: Transport and Environment*, vol. 85, p. 102 393, 2020. DOI: 10.1016/j.trd.2020.102393.

[40] W. Zhu, K. Liu, M. Wang, and E. E. Koks, "Seismic risk assessment of the railway network of china's mainland," *International Journal of Disaster Risk Science*, vol. 11, no. 4, pp. 452–465, 2020. DOI: 10.1007/s13753-020-00292-9.

[41] N. A. C. Cressie and C. K. Wikle, *Statistics for spatio-temporal data*. Hoboken, N.J: Wiley, 2011, p. 624, ISBN: 9780471692744.

[42] P. C. Kyriakidis and A. G. Journel, "Geostatistical space-time models: A review," *Mathematical Geology*, vol. 31, no. 6, pp. 651–684, 1999, ISSN: 08828121. DOI: 10.1023/a:1007528426688.

[43] HSIP, *Homeland Security Infrastructure Program (HSIP) data*, available via https://hifld-dhs-gii.opendata.arcgis.com/, 2019. (visited on 03/20/2019).

[44] HIFLD, *Homeland Infrastructure Foundation-Level Data (HIFLD)*, available via https://hifld-dhs-gii.opendata.arcgis.com/datasets?q=*dam&sort_by=relevance, 2020. (visited on 03/03/2020).

[45] HIFLD, *Homeland Infrastructure Foundation-Level Data (HIFLD)*, available via https://data.navigationdatacenter.us/Locks/Public-Lock-Unavailability-Detailed-Report/p3mn-gzqj, 2020. (visited on 03/03/2020).

[46] HIFLD, *Homeland Infrastructure Foundation-Level Data (HIFLD)*, available via https://hifld-dhs-gii.opendata.arcgis.com/datasets?q=flood, 2020. (visited on 04/03/2020).

[47] NOAA, *National Oceanic and Atmospheric Administration (NOAA Climate)*, available via https://www.noaa.gov/climate, 2020. (visited on 01/20/2020).

[48] MarTREC, *Maritime Transportation Research and Transportation Center*, available via https://martrec.uark.edu/, 2020. (visited on 03/27/2020).

[49] MarTREC, *Maritime Transportation Resource Data Bank — MarTREC — University of Arkansas*, available via https://martrec.uark.edu/research/resource-data-bank.php, 2020. (visited on 03/27/2020).

[50] MarTREC Simulation Tool, *Data simulation to support interdependence modeling of a multimodal transportation network*, available via https://martrec.uark.edu/data/index.php, 2020.

[51] K. S. Bakar, S. K. Sahu, *et al.*, "spTimer: Spatio-temporal bayesian modelling using R," *Journal of Statistical Software*, vol. 63, no. 15, pp. 1–32, 2015.

[52] B. Matérn, *Spatial variation*. New York: Springer-Verlag, 1986, p. 153, ISBN: 9781461578925.

[53] U. Wilensky, *NetLogo*, available via https://ccl.northwestern.edu/netlogo/, Evanston, IL, 1999. (visited on 12/14/2019).

[54] S. Tisue and U. Wilensky, "Netlogo: Design and implementation of a multi-agent modeling environment," in *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, Chicago, IL, Oct. 2004, pp. 7–9.

[55] H. Nachtmann, K. N. Mitchell, C. E. Rainwater, R. Gedik, and E. A. Pohl, "Optimal dredge fleet scheduling within environmental work windows," *Transportation Research Record*, vol. 2426, no. -1, pp. 11–19, 2014, ISSN: 03611981. DOI: 10.3141/2426-02.

[56] National Weather Service, *Advanced Hydrologic Prediction Service*, available via https://water.weather.gov/ahps2/hydrograph.php?wfo=fsd&gage=lnni4, 2020.

[57] Z. Tian and M. J. Zuo, "Health condition prediction of gears using a recurrent neural network approach," *IEEE transactions on reliability*, vol. 59, no. 4, pp. 700–705, 2010.

[58] A. Saxena and K. Goebel. "Phm08 challenge data set." (2008), [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/%5C#phm08%5C_challenge.

[59] E. Ramasso and A. Saxena, "Performance benchmarking and analysis of prognostic methods for cmapss datasets.," *International Journal of Prognostics and Health Management*, vol. 5, no. 2, pp. 1–15, 2014.

[60] L. Peel, "Data driven prognostics using a kalman filter ensemble of neural network models," in *2008 International Conference on Prognostics and Health Management*, IEEE, 2008, pp. 1–6.

[61] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *2008 international conference on prognostics and health management*, IEEE, 2008, pp. 1–6.

[62] T. Wang, J. Yu, D. Siegel, and J. Lee, "A similarity-based prognostics approach for remaining useful life estimation of engineered systems," in *2008 International Conference on Prognostics and Health Management*, IEEE, 2008, pp. 1–6.

[63] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," *arXiv preprint arXiv:1704.02971*, 2017.

[64] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in pytorch," 2017.

[65] C. Zuo. "A pytorch example to use rnn for financial prediction." (2017), [Online]. Available: `https://github.com/chandlerzuo/chandlerzuo.github.io/tree/master/codes/da%5C_rnn`.

[66] J. Ma and D. Yarats, "Quasi-hyperbolic momentum and adam for deep learning," *arXiv preprint arXiv:1810.06801*, 2018.

[67] H. Editors, *Blackout hits northeast united states*, `https://www.history.com/this-day-in-history/blackout-hits-northeast-united-states`, Accessed: 2022-05-05, 2009.

[68] S. Pruitt, *How levee failures made hurricane katrina a bigger disaster*, `https://www.history.com/news/hurricane-katrina-levee-failures`, Accessed: 2022-05-05, 2020.

[69] M. O. Ball, C. J. Colbourn, and J. S. Provan, "Network reliability," *Handbooks in operations research and management science*, vol. 7, pp. 673–762, 1995.

[70] V. Gaur, O. P. Yadav, G. Soni, and A. P. S. Rathore, "A literature review on network reliability analysis and its engineering applications," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 235, no. 2, pp. 167–181, 2021.

[71] D. R. Karger, "A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem," *SIAM Journal on Computing*, vol. 29, no. 2, pp. 492–514, 1999.

[72] J. B. Cardoso, J. R. de Almeida, J. M. Dias, and P. G. Coelho, "Structural reliability analysis using monte carlo simulation and neural networks," *Advances in Engineering Software*, vol. 39, no. 6, pp. 505–513, 2008.

[73] C. Srivaree-Ratana, A. Konak, and A. E. Smith, "Estimation of all-terminal network reliability using an artificial neural network," *Computers & Operations Research*, vol. 29, no. 7, pp. 849–868, 2002.

[74] G. Chartrand, *Introductory graph theory*. Courier Corporation, 1977.

[75] J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 777–788, 1983.

[76] C. Godsil and G. F. Royle, *Algebraic graph theory*. New York: Springer Verlag, 2001, vol. 207, pp. 354–358.

[77] N. L. Biggs, *Algebraic graph theory*, 2nd ed. England: Cambridge university press, 1993, ch. 13, pp. 97–105.

[78] R. Dougherty, *Reliability polynomial calculation*, `https://codereview.stackexchange.com/questions/131709/reliability-polynomial-calculation`, Accessed: 2022-05-05, 2016.

[79] I. MongoDB, *How to use mongodb in python*, `https://www.mongodb.com/languages/python`, Accessed: 2022-05-05, 2022.

[80] I. MongoDB, *Mongodb manual: Gridfs*, `https://www.mongodb.com/docs/manual/core/gridfs`, Accessed: 2022-05-05, 2022.

[81] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[82] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[83] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, *Stable baselines 3*, `https://github.com/Stable-Baselines-Team/stable-baselines3-contrib`, Accessed: 2022-05-05, 2019.

[84] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[85] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.

[86] J. C. H. Azucena, H. Wells, H. Liao, K. Sullivan, and E. A. Pohl, "Applying deep reinforcement learning to improve the reliability of an infrastructure network," *Advances in Modelling to Improve Network Resilience*, p. 46, 2022.

[87] C. Srivaree-ratana and A. E. Smith, "Estimating all-terminal network reliability using a neural network," in *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)*, IEEE, vol. 5, 1998, pp. 4734–4739.

[88] L. G. Valiant, "The complexity of enumeration and reliability problems," *siam Journal on Computing*, vol. 8, no. 3, pp. 410–421, 1979.

[89] J. E. Ramirez-Marquez and C. M. Rocco, "All-terminal network reliability optimization via probabilistic solution discovery," *Reliability Engineering & System Safety*, vol. 93, no. 11, pp. 1689–1697, 2008.

[90] B. Dengiz, F. Altiparmak, and A. E. Smith, "Efficient optimization of all-terminal reliable networks, using an evolutionary approach," *IEEE transactions on Reliability*, vol. 46, no. 1, pp. 18–26, 1997.

[91] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[92] A. K. Goharshady and F. Mohammadi, "An efficient algorithm for computing network reliability in small treewidth," *Reliability Engineering & System Safety*, vol. 193, p. 106 665, 2020.

[93] A. Davila-Frias, S. Salem, and O. P. Yadav, "Deep neural networks for all-terminal network reliability estimation," in *2021 Annual Reliability and Maintainability Symposium (RAMS)*, IEEE, 2021, pp. 1–7.

[94] A. Davila-Frias and O. P. Yadav, "All-terminal network reliability estimation using convolutional neural networks," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 236, no. 4, pp. 584–597, 2022.

[95] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," *Advances in neural information processing systems*, vol. 28, 2015.

[96] E. Bingham, J. P. Chen, M. Jankowiak, *et al.*, "Pyro: Deep universal probabilistic programming," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 973–978, 2019.

[97] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic variational inference," *Journal of Machine Learning Research*, 2013.

[98] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," *arXiv preprint arXiv:2006.14171*, 2020.

[99] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[100] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, *Stable baselines3*, 2019.

[101] F. Altiparmak, B. Dengiz, and A. E. Smith, "Optimal design of reliable computer networks: A comparison of metaheuristics," *Journal of heuristics*, vol. 9, pp. 471–487, 2003.

[102] E. Kim, J. Kim, H. Lee, and S. Kim, "Adaptive data augmentation to achieve noise robustness and overcome data deficiency for deep learning," *Applied Sciences*, 2021. DOI: 10.3390/APP11125586.

[103] J. Yang, D. Parikh, and D. Batra, "Joint unsupervised learning of deep representations and image clusters," *arXiv: Computer Vision and Pattern Recognition*, 2016.

[104] L. Zhang, L. Qi, X. Yang, H. Qiao, M. Yang, and Z. Liu, "Automatically discovering novel visual categories with self-supervised prototype learning," *ArXiv*, 2022. DOI: 10.48550/ARXIV.2208.00979.

[105] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," 2016.

[106] X. Guo, X. Liu, E. Zhu, *et al.*, "Adaptive self-paced deep clustering with data augmentation," *IEEE Transactions on Knowledge and Data Engineering*, 2020. DOI: 10.1109/TKDE.2019.2911833.

[107] L. Wu, Z. Liu, Z. Zang, J. Xia, S. Li, and S. Z. Li, "Deep clustering and representation learning that preserves geometric structures," *arXiv: Learning*, 2021.

[108] S. Pabst and Y. Nam, *A quantum algorithm for network reliability*, 2022. DOI: 10.48550/ARXIV.2203.10201. [Online]. Available: https://arxiv.org/abs/2203.10201.

[109] IBM, *The atoms of computation*. [Online]. Available: https://learn.qiskit.org/course/introduction/the-atoms-of-computation.

[110] A-tA-v, M. S. ANIS, Abby-Mitchell, *et al.*, *Qiskit: An open-source framework for quantum computing*, 2021. DOI: 10.5281/zenodo.2573505.

[111] T. Duong, J. A. Ansere, B. Narottama, V. Sharma, O. Dobre, and H. Shin, "Quantum-inspired machine learning for 6g: Fundamentals, security, resource allocations, challenges, and future research directions," *IEEE Open Journal of Vehicular Technology*, 2022. DOI: 10.1109/OJVT.2022.3202876.

[112] N. Nikmehr and P. Zhang, "Quantum-inspired power system reliability assessment," *IEEE Transactions on Power Systems*, pp. 1–14, 2022. DOI: 10.1109/TPWRS.2022.3204393.

[113] L. Dueñas-Osorio, M. Y. Vardi, and J. Rojo, "Quantum-inspired boolean states for bounding engineering network reliability assessment," *Structural Safety*, 2018. DOI: 10.1016/J.STRUSAFE.2018.05.004.

[114] M. Bhatia and S. K. Sood, "Quantum computing-inspired network optimization for iot applications," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5590–5598, 2020. DOI: 10.1109/JIOT.2020.2979887.

[115] K. Lin, J. Zhu, and Y. Chen, "A non-hermitian quantum approach to reliability of a two-state system," *Physics Letters A*, 2020. DOI: 10.1016/J.PHYSLETA.2019.126207.

[116] Y. Cui, K. Lin, Y. Chen, and J. Zhu, "A quantum-inspired model for statistical analysis of repairable systems," *Computers & Industrial Engineering*, 2021. DOI: 10.1016/J.CIE.2021.107613.

[117] United States Geological Survey, *USGS Surface-Water Data for the Nation*, available via https://waterdata.usgs.gov/nwis/sw, 2020.

A. Appendix

A.1 Chapter 2, Section 2.1

A.1.1 Introduction to the developed simulation tool

A.1.1.1 Simulation setup

A user can use model controls (Figures 2.3–2.4) to quickly adjust the settings of the initial environment (see buttons, sliders, and other controls). To initiate a simulation run after setting up the environment, the user can input various attributes through sliders, choosers, and switches. The steps are as follows:

Step 1: First, click the "Spatial Temporal Analysis" button to run the spatio-temporal model to generate predicted GH and lock availability data.

Step 2: Click the "Setup Environment" button. This button is used to initialize the model, and it is a NetLogo "once-button" that runs its code once. After this step, all maps will be drawn on the user interface.

Step 3: Click the "Add Ports" button to draw eight circles on the waterways that depict the ports of interest in our model.

Step 4: Click the "Add Locks & Sites" button to initialize and draw all fifteen locks and twenty-four sites with their variables. Then, click "Add Vessels" to draw vessels on the ports based on the data. The vessels are categorized according to their size and speed into three groups, and they carry two types of products in the current setting: petroleum and crops.

After completing the above steps, a value for "Simulation-Time" needs to be set to indicate the number of months the simulation will run. The available options are 3, 6, 9, and 12 months. "Fleet-Size" provides the the number of trucks required to carry products from one vessel through the highways when vessels reach to their destinations, and when they are unable to move for a certain amount of time due to extreme events. The global

slider “GH-Threshold” may need to be adjusted to a reasonable value (higher than the individual sites' GHs which also can be adjusted as shown in Figure A.1). This value acts as the threshold value of GH, which is being compared with the hourly value of GH at each site. Such information about GH thresholds can be adopted from sources such as the United States Geological Survey (USGS) [117]. We can change the value at runtime. There are fifteen switches, each of which acts as a controller to turn on/off the corresponding lock. The selection can also be changed during the runtime. When one lock is closed, the vessels that are supposed to pass through the lock will wait in the previous node in the path and will not move forward until the lock is reopened. If the "LockRandomFailures" button is set to "On", locks' failure and repair events will be generated using random variates. The mean time to these events is controlled by the parameters "AvgTimeToLockFail" and "AvgTimeToLockRepair". After all the settings are completed, click the “Start Simulation” button and the vessels at each port start moving towards their predefined destinations. Each vessel checks for any unsafe circumstances at the locks and sites along its route and makes decisions accordingly.

A.1.1.2 Model outputs

After running the simulation tool, multiple numerical outputs are generated for the user. The outputs can be classified into summaries and detailed results. The summaries are shown in Tables 2.4–2.7 and Figures 2.7–2.9. As mentioned in Section 2.1.4.2, Tables 2.4 and 2.7 show summary statistics for the vessels traveled between every two ports classified by the vessel category. In addition, summarized vessel and truck tables such as Table 2.4 show the number of vessels (trucks) that have traveled from ports, average speed of travel, product type carried and other statistics that help the user understand the commodity flow changes under each scenario. Each summarized table output is exported as a “CSV” file format that can be used as an input file to other statistical software or programming languages for further analysis. Furthermore, some of these statistical summaries are plotted using the NetLogo

R extension and exported as publication-quality "PDF" files such as the ones shown in Figures 2.7–2.9. The outputs of the simulation tool also include detailed information about all vessels appeared in the model (see Table **??**). Like the records mentioned before, the detailed information includes the product type carried, category of the vessel, total weight of the product, and a defined ID for each vessel (modeled as a turtle). Also registered, each vessel's start time, time of departure from its origin, and its end time (time of arrival at its destination). These records can help extend the simulation analysis and contribute to the model's debugging and validation. These results are tabulated and exported as a "CSV" file with additional detailed information about the trucks, as shown in Table A.1. The detailed information in Table A.1 is useful to identify individual trucks by their ID, destination, departure (start) and arrival (end) times in the simulation along with distance traveled in miles and average speed in mph for each truck. Such information becomes useful in analyzing a given product's current logistic plan and finding possible ways to alter and improve the current one.

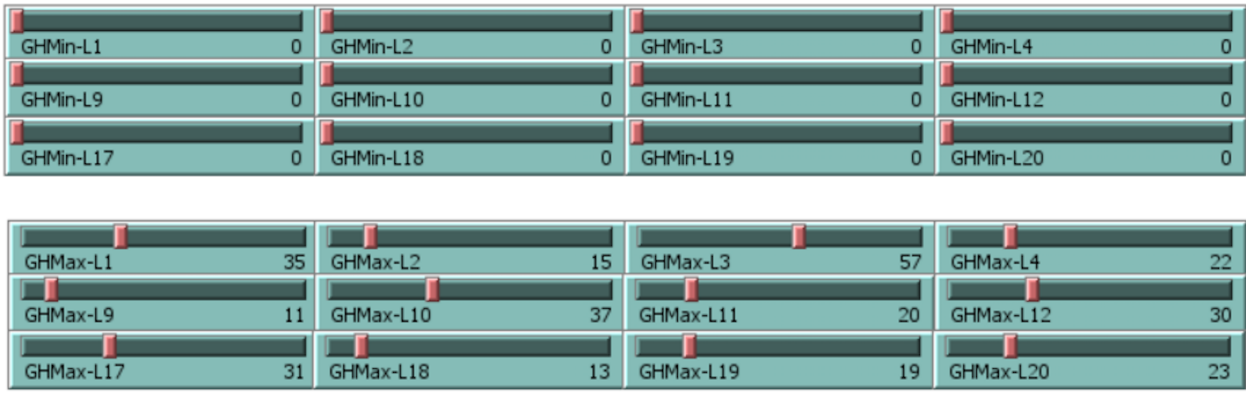


Figure A.1: Water level thresholds set by user for all sites to control extreme events criteria

ID	Product type	Distance traveled (miles)	Start time (ticks)	End time (ticks)	From	To	Time elapsed (ticks)	Avg. speed (mph)
5000	Petroleum	524.7424833	103	110.75	Fort Smith	E	7.75	67.71
5001	Petroleum	524.5594318	103	111.25	Fort Smith	E	8.25	63.58
5002	Petroleum	524.5952797	103	110.25	Fort Smith	E	7.25	72.36
5003	Petroleum	524.3400126	103	110.25	Fort Smith	E	7.25	72.32
5004	Petroleum	524.3189491	103	110.5	Fort Smith	E	7.5	69.91
5005	Petroleum	525.4226671	103	110.5	Fort Smith	E	7.5	70.06
5006	Petroleum	524.7314365	103	110	Fort Smith	E	7	74.96
5007	Petroleum	524.2494159	103	110.25	Fort Smith	E	7.25	72.31
5008	Petroleum	524.4084918	103	110	Fort Smith	E	7	74.92
5009	Petroleum	524.5227015	103	110.25	Fort Smith	E	7.25	72.35
5010	Petroleum	524.5717312	103	110.5	Fort Smith	E	7.5	69.94
5011	Petroleum	524.3143702	103	109.75	Fort Smith	E	6.75	77.68
5012	Petroleum	524.2374395	103	110.5	Fort Smith	E	7.5	69.90
5013	Petroleum	524.1374195	103	110.25	Fort Smith	E	7.25	72.29

5014	Petroleum	524.6757786	103	110.25	Fort Smith	E	7.25	72.37
5015	Petroleum	524.2429046	103	110	Fort Smith	E	7	74.89
5016	Petroleum	167.7360173	103	105.5	Fort Smith	N	2.5	67.09
5017	Petroleum	177.1946809	103	105.5	Fort Smith	N	2.5	70.88
5018	Petroleum	443.1499304	103	109.75	Fort Smith	S	6.75	65.65
5019	Petroleum	450.1915388	103	109.5	Fort Smith	S	6.5	69.26
7000	Petroleum	242.3998382	142.5	146.5	Helena	E	4	60.60
7001	Petroleum	239.7554172	142.5	146	Helena	E	3.5	68.50
7002	Petroleum	243.9796093	142.5	146.5	Helena	E	4	60.99
7003	Petroleum	240.2386683	142.5	145.75	Helena	E	3.25	73.92
7004	Petroleum	241.5537637	142.5	146	Helena	E	3.5	69.02
7005	Petroleum	239.2439371	142.5	146.25	Helena	E	3.75	63.80
7006	Petroleum	240.4777264	142.5	145.75	Helena	E	3.25	73.99
7007	Petroleum	241.2591156	142.5	146.25	Helena	E	3.75	64.34
7008	Petroleum	241.0668822	142.5	146.25	Helena	E	3.75	64.28
7009	Petroleum	242.7985141	142.5	146.25	Helena	E	3.75	64.75
7010	Petroleum	241.4463379	142.5	146.25	Helena	E	3.75	64.39
7011	Petroleum	239.9652814	142.5	146	Helena	E	3.5	68.56
7012	Petroleum	243.0676827	142.5	146.25	Helena	E	3.75	64.82
7013	Petroleum	243.2426233	142.5	146.5	Helena	E	4	60.81
7014	Petroleum	238.8765655	142.5	145.75	Helena	E	3.25	73.50
7015	Petroleum	239.9229721	142.5	146	Helena	E	3.5	68.55
7016	Petroleum	394.0866412	142.5	148.5	Helena	N	6	65.68
7017	Petroleum	386.0744743	142.5	147.75	Helena	N	5.25	73.54
7018	Petroleum	262.8503073	142.5	147	Helena	S	4.5	58.41
7019	Petroleum	258.4577559	142.5	147	Helena	S	4.5	57.44
8000	Crops	240.4638423	47	50.5	Helena	E	3.5	68.70

8001	Crops	385.549631	47	52.75	Helena	N	5.75	67.05
8002	Crops	258.635328	47	50.5	Helena	S	3.5	73.90
8003	Crops	241.2157411	52.75	56.25	Helena	E	3.5	68.92
8004	Crops	385.151909	52.75	58	Helena	N	5.25	73.36
8005	Crops	258.5213772	52.75	57.25	Helena	S	4.5	57.45
8006	Crops	239.7437967	1021.25	1025	Helena	E	3.75	63.93
8007	Crops	384.7920906	1021.25	1026.75	Helena	N	5.5	69.96
8008	Crops	263.6299675	1021.25	1025	Helena	S	3.75	70.30
8009	Crops	240.3038144	1026.75	1030.5	Helena	E	3.75	64.08
8010	Crops	391.4770111	1026.75	1032.75	Helena	N	6	65.25
8011	Crops	255.6389391	1026.75	1031.25	Helena	S	4.5	56.81
8012	Crops	239.9652814	1032.75	1036.5	Helena	E	3.75	63.99
8013	Crops	388.9339639	1032.75	1038.75	Helena	N	6	64.82
8014	Crops	259.789386	1032.75	1036.25	Helena	S	3.5	74.23
8015	Crops	242.9573568	1038.75	1042.25	Helena	E	3.5	69.42
8016	Crops	383.1177057	1038.75	1044.25	Helena	N	5.5	69.66
8017	Crops	255.8213064	1038.75	1043.5	Helena	S	4.75	53.86
8018	Crops	242.8259393	1044.25	1047.5	Helena	E	3.25	74.72
8019	Crops	390.222719	1044.25	1049.5	Helena	N	5.25	74.33
8020	Crops	262.4966935	1044.25	1049	Helena	S	4.75	55.26
8021	Crops	242.2872164	1049.5	1053	Helena	E	3.5	69.22
8022	Crops	389.1117699	1049.5	1055.75	Helena	N	6.25	62.26
8023	Crops	253.3064752	1049.5	1053.5	Helena	S	4	63.33
8024	Crops	242.9258206	1056	1059.5	Helena	E	3.5	69.41
8025	Crops	392.8540749	1056	1061.5	Helena	N	5.5	71.43
8026	Crops	259.0699308	1056	1060.5	Helena	S	4.5	57.57
8027	Crops	242.0606581	1061.5	1064.75	Helena	E	3.25	74.48
8028	Crops	380.523792	1061.5	1067	Helena	N	5.5	69.19
8029	Crops	259.2975286	1061.5	1065.25	Helena	S	3.75	69.15
8030	Crops	242.3815963	2020.75	2024.25	Helena	E	3.5	69.25
8031	Crops	386.7121652	2020.75	2025.75	Helena	N	5	77.34
8032	Crops	257.3860023	2020.75	2025.25	Helena	S	4.5	57.20
8033	Crops	243.0613488	2025.75	2029.25	Helena	E	3.5	69.45

8034	Crops	384.1817111	2025.75	2031	Helena	N	5.25	73.18
8035	Crops	258.8727207	2025.75	2030.25	Helena	S	4.5	57.53
8036	Crops	242.2048536	2031	2034.75	Helena	E	3.75	64.59
8037	Crops	394.453156	2031	2037.25	Helena	N	6.25	63.11
8038	Crops	253.9643062	2031	2035.5	Helena	S	4.5	56.44
8039	Crops	243.3186303	2037.25	2040.5	Helena	E	3.25	74.87
8040	Crops	388.3791161	2037.25	2042.5	Helena	N	5.25	73.98
8041	Crops	258.4626828	2037.25	2041	Helena	S	3.75	68.92
8042	Crops	241.4879116	2042.5	2046.25	Helena	E	3.75	64.40
8043	Crops	385.1115147	2042.5	2047.75	Helena	N	5.25	73.35
8044	Crops	264.8154224	2042.5	2047	Helena	S	4.5	58.85
8045	Crops	241.4091439	2047.75	2051.25	Helena	E	3.5	68.97
8046	Crops	390.574694	2047.75	2053	Helena	N	5.25	74.40
8047	Crops	254.1111329	2047.75	2052.25	Helena	S	4.5	56.47

Table A.1: Detailed outputs for trucks