FINAL REPORT

Transportation Infrastructure Flooding: Sensing Water Levels and Clearing and Rerouting Traffic out of Danger

Date: October 10, 2017

Pamela Murray-Tuite, PhD, Associate Professor, Virginia Tech
Gaby Joe Hannoun, Graduate Research Assistant, Virginia Tech
Antonio Fuentes, Graduate Research Assistant, Virginia Tech
Kevin Heaslip, PhD, Associate Professor, Virginia Tech
Venkataramana Sridhar, PhD, Assistant Professor, Virginia Tech
Prasanth Valayamkunnath, Graduate Research Assistant, Virginia Tech
Jonathan Goodall, PhD, Associate Professor, University of Virginia
Jeffrey Sadler, Graduate Research Assistant, University of Virginia

Prepared by:
Department of Civil Engineering
Virginia Tech
7054 Haycock Rd
Falls Church, VA 22043

Prepared for:

Virginia Center for Transportation Innovation and Research
530 Edgemont Road
Charlottesville, VA 22903

1. Report No.	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle Transportation Infrastructure Flooding: Sensing Water Levels and Clearing and Rerouting Traffic out of Danger		**5. Report Date** October 10, 2017
		6. Performing Organization Code
7. Author(s) Pamela Murray-Tuite, Gaby Joe Hannoun, Antonio Fuentes, Kevin Heaslip, Venkataramana Sridhar, Prasanth Valayamkunnath, Jonathan Goodall, Jeffrey Sadler		**8. Performing Organization Report No.**
9. Performing Organization Name and Address Department of Civil Engineering Virginia Tech 7054 Haycock Rd Falls Church, VA 22043		**10. Work Unit No. (TRAIS**
		11. Contract or Grant No. DTRT13-G-UTC33
12. Sponsoring Agency Name and Address US Department of Transportation Office of the Secretary-Research UTC Program, RDT-30 1200 New Jersey Ave., SE Washington, DC 20590		**13. Type of Report and Period Covered** Final 5/10/16 – 8/30/17
		14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

Flooding in urban areas, driven by both precipitation and high tide events, can have a devastating effect on a region's transportation system and economic viability. In this multi-disciplinary project, the research team developed an approach to help increase the resilience of transportation operations during nuisance flooding and mitigate the danger to drivers and vehicle-related property damage by developing flood predictions, road closure messages, and re-routing.

Rainfall data were obtained and used in conjunction with weather forecasts in independent simulation using the Weather Research and Forecasting (WRF) Model to develop rainfall hyetograph forecasts. Overall, WRF simulated rainfall was within the range of uncertainty for a large scale model in capturing the rainfall events with data assimilation and the performance of WRF with data assimilation was similar to previous studies. The sources of bias in WRF rainfall estimates may be due to forecast error, error in the assimilated data, or bias in the initialization or boundary conditions.

To predict roadway flooding, the research team adopted the Random Forest data mining approach to identify relationships and patterns between tide levels and rainfall events. Water table level and wind conditions were also used in the model training. The model's false positive rate was 8.92%; however, most of the false positive predictions were less than 1.5 flood reports. The model had a much higher false negative rate, 31.50%. Importantly, the model did not have any false negative predictions when the true number of reported floods was high. The total daily rainfall value was by far the most important of the variables. The height of the water table did not add appreciable predictive power to the model. The model had some deficiencies, likely due to the limited amount of data and bias in the flood location reporting.

Given a flooding prediction and vehicles' origins, destinations, intended paths, and departure times, the research team developed a framework to provide routing assistance to affected vehicles and sending warnings to unaffected ones. If the vehicle is unaffected by the predicted flood, it can receive a warning outlining the areas to avoid in case of any deviation and it can resume its trip as intended. If affected, the vehicle either received a warning to stay at its origin or routing guidance in the form of a set of alternative paths. The framework was applied to two transportation networks modeled in VISSIM based on the City of Virginia Beach, VA. Computation time increased with the increase in network size and in the flood depth, but was still generally less than 1 second. With fewer weather and flood updates, more vehicles were rerouted, likely causing more congestion on alternate routes. Higher temporal resolution weather and flood prediction systems can help reduce this effect in the future.

17. Key Words Flooding, traffic routing, connected vehicles, weather, rain, nuisance flooding		**18. Distribution Statement** No restrictions. This document is available from the National Technical Information Service, Springfield, VA 22161	
19. Security Classif. (of this report) Unclassified	**20. Security Classif. (of this page)** Unclassified	**21. No. of Pages** 95	**22. Price**

Acknowledgments

The authors thank the employees of the City of Virginia Beach and Norfolk who provided the data needed for this study. In particular, Steve McLauglin, Mike Shahsiah, and George Hardin from the City of Virginia Beach assisted the team with acquiring traffic signal timings. The authors also appreciate the effort of Fatema Siddique who developed the smaller network in VISSIM.

Contents

List of Tables

List of Figures

1 Introduction

Many coastal areas (e.g., Virginia Beach) are prone to flooding due to inadequate stormwater management infrastructure, rising sea levels, tidal effects, and intense precipitation. Replacing large amounts of infrastructure simultaneously is cost prohibitive, indicating a need for alternative approaches to resilience enhancement. In this project, the research team developed an approach to help increase the resilience of transportation operations during flooding and mitigate the danger to drivers and vehicle-related property damage by developing flood predictions, road closure messages, and re-routing. Vehicles on affected roadway segments at the time of the flood are in danger as are those entering flooded roadways. Road closures currently require physical resources, but sensor and communication technology can reduce the need for these resources, allowing them to be deployed elsewhere. Driver delays can be reduced with knowledge of flooded areas and routing recommendations.

1.1 Problem

Flooding in urban areas, driven by both precipitation and high tide events, can have a devastating effect on a region's transportation system and economic viability. In the City of Virginia Beach, the problem is acute as nuisance flooding in heavily populated areas impacts both communities and transportation infrastructure. Nuisance flooding, by definition, is "flooding that leads to public inconveniences such as road closures" (US Department of Commerce 2014). Road closures due to flooding lead to the rerouting and/or the cancellation of affected trips (Pyatkova, Chen et al. 2015).

The critical needs to identify the magnitude of floods are to measure and model precipitation intensity with a short lead time and relate to high tide events to plan proper protective measures for and diversion from problem areas. This study adopted a multi-disciplinary approach (hydrology, regional climate and precipitation forecasting, and transportation engineering) to predict roadway flooding and mitigate travelers' danger from the flood and delays.

From the hydrology/precipitation perspective, the research addressed flooding due to a complex relationship between tide levels and rainfall events. Rainfall and tidal gauge data were obtained from the City of Virginia Beach and other organizations and then analyzed using standard data mining approaches to identify relationships and patterns. The research team adopted a data-driven approach whereby patterns of tidal levels and rainfall intensities and durations that cause flooding can be identified. The data were also used in conjunction with weather forecasts in independent simulation using the Weather Research and Forecasting (WRF) Model to develop rainfall hyetograph forecasts.

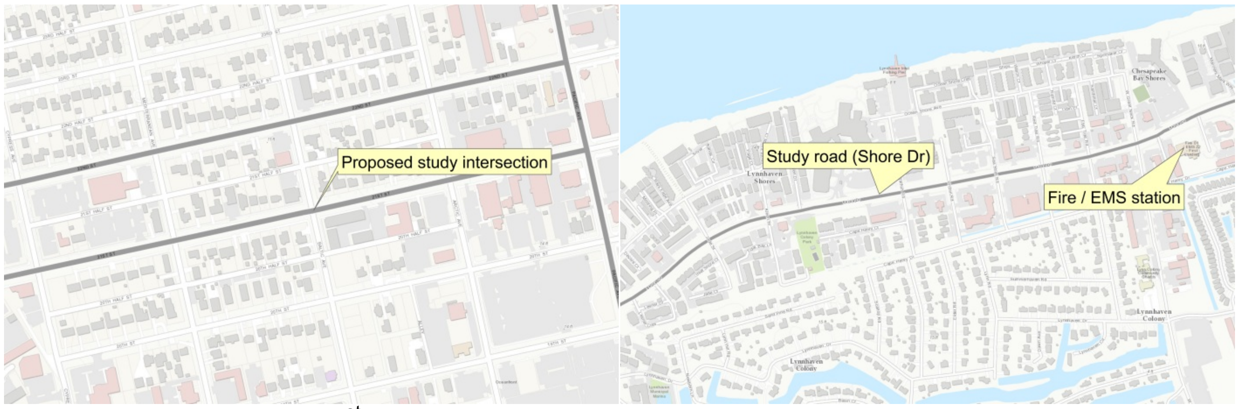
From the transportation perspective, the research team focused on two types of drivers: those who are on the road as the flood occurs and those who have not yet entered that particular road and must be re-routed. For the first group, warning and road closures must be provided in time to remove these drivers from the impact area. The amount of time required to clear the link depends on network traffic conditions and potentially other flooded areas. The second group must be re-routed so as not to enter the affected link(s) and place the drivers in danger from flooding. Both of these groups were affected by the flood. However, depending on their origins, and intended destinations and paths, some vehicles could proceed with their trip as planned. The affected vehicles had origins, destinations and/or intermediate node(s) on soon-to-be flooded or closed links. In these cases, in-vehicle route guidance systems can assist the users in leaving the danger zone and selecting the best alternative path. In addition, in-vehicle route guidance systems could communicate warning messages to unaffected vehicles alerting them about the areas to avoid.

The purposes of this project were to help assess the vulnerability of transportation infrastructure and increase the resilience of transportation operations during flooding and alleviate the adverse impacts to drivers and vehicle-related property damage. The focus of this research was on nuisance flooding, but the methods developed here can be extended to extreme weather events in the future. The overall goals of this project were to (1) build knowledge on the nature of precipitation events that cause flooding and (2) improve traveler safety for flooding events. To achieve these goals, this project's objectives were to:

- identify patterns in the rainfall and tidal gauge data where tidal water levels and precipitation lead to flooding of roadways,
- develop rainfall hyetograph forecasts,
- project whether roadways will flood, and
- provide routing guidance to en-route vehicles to avoid flooded links.

1.2 Study Area

This multi-disciplinary effort used the Tidewater region of Virginia as the study region. The regional climate and precipitation forecasting and transportation aspects were based in Virginia Beach with two locations illustrated in Figure 1. These locations were selected based on the preferences and recommendations of Greg Johnson (stormwater) and Steve McLaughlin (transportation), both with the City of Virginia Beach. According to these sources, the location shown in Figure 1a flooded four times in one summer. The flooding happened through a combination of both tidal conditions and rainfall meaning it can flood at low tide with enough rain or at a very high tide with just a little rain, which makes for a challenging hydrologic model that can account for both tidal and precipitation conditions/forecasts. The second location (Figure 1b) had the added challenge of a fire station that is occasionally out of commission due to localized flooding. However, due to data issues, the hydrology portion of the study could only be performed in Norfolk, VA.



(a) Baltic Ave. and 21st St. (b) Shore Drive

Figure 1 Case Study Locations in Virginia Beach

1.3 Organization of the Report

This report has three major chapters, each corresponding to one of the disciplines involved in this study. Chapter 2 describes the weather forecasting aspects of the study and was authored by Dr. Venkataramana Sridhar and Prasanth Valayamkunnath. Chapter 3 describes the hydrology portion of the study and was authored by Dr. Jonathan Goodall and Jeffrey Sadler. Chapter 4 discusses the transportation aspects of the study and was authored by Gaby Joe Hannoun, Antonio Fuentes, and Drs. Pamela Murray-Tuite and Kevin Heaslip. Each chapter contains its own methodology, findings, and conclusions. Finally, Chapter 5 provides some overall recommendations.

2 Weather

The purpose of the weather portion of this study was to derive precipitation intensities and magnitudes.

2.1 Data

2.1.1 Meteorological Observations

Gauge measured 15-minute precipitation data from 1 January 2016 to 31 December 2016 were accessed from the Hampton Roads Sanitation District (HRSD) for the City of Virginia Beach. This data was used in this study as the validation data for the Weather Research and Forecasting model simulated precipitation. The attributes of the rain gauge stations are provided in Table 1. Figure 2 shows the spatial distribution of the rain gauge stations.

Table 1 Details of HRSD Rain Gauge Stations in Virginia Beach

Station_ID	**Location**	**x**	**y**
MMPS-171	HRSD SP - Shipps Corner PRS	3717010	1053320
MMPS-185	HRSD SF - Lagomar IFM at Atlantic IFM	3725430	1051520
MMPS-163	HRSD SP - Providence PRS	3705870	1055870
MMPS-255	VBCH PS - Virginia Beach PS 606	3721690	1051630
MMPS-146	HRSD SP - Laskin Rd PRS	3722250	1060120
MMPS-004	HRSD SF - John B. Dey MLV-AT side	3717470	1065180
MMPS-256	VBCH PS - Virginia Beach PS 472	3705690	1051990
MMPS-140	HRSD SP - Independence PRS	3710690	1058910
MMPS-160	HRSD SP - Pine Tree PRS	3716430	1059490
MMPS-144	HRSD SP - Kempsville PRS	3710390	1054080
MMPS-036	HRSD SF - Northampton Blvd at Wesleyan Dr	3705530	1062830
MMPS-093-2	HRSD TP - Ches-Liz Main Flow (Influent)	3708140	1066330

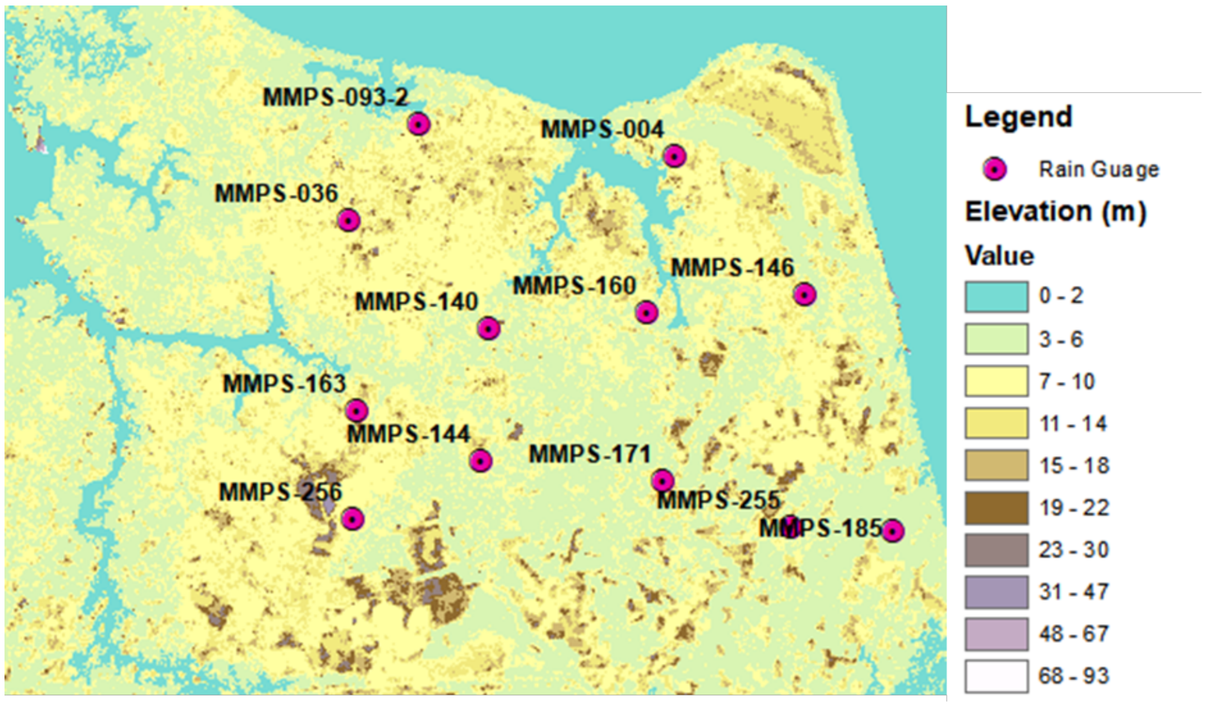


Figure 2 Spatial distribution of HRSD rain gauge stations shown on a base map of elevation for the study area.

2.1.2 NCAR Upper-Air Observational Weather Data

The US National Center for Atmospheric Research (NCAR) archives operational global meteorological observations which are used for data assimilation in real-time weather forecasting cases. The 6-hourly upper-air data assimilated in this study were accessed from the "ds351.0" dataset[1], which contains the upper-air measurements of pressure, geopotential height, air temperature, dew point temperature, wind direction and speed from 1000 millibars to 1 millibar. The data were then converted into LITTLE_R (WRF data assimilation) format before they were assimilated in the 4D-Var system.

2.1.3 Stage IV Precipitation Data (Radar +Gauge Observation)

There are many radar-based and gauge-based precipitation products available for the study region developed by different agencies and institutions. The Stage IV national mosaic is the best radar and gauge-derived precipitation data available for CONUS generated operationally. This data[2] is generated at 4km resolution using the regional hourly/6-hourly multi-sensor (radar + gauges) precipitation analyses produced by the 12 River Forecast Centers (RFCs). The data underwent manual quality control checking at the RFCs.

2.2 Weather Research and Forecasting (WRF) Model

The numerical experiments of precipitation data assimilation and high-resolution precipitation forecast were conducted using the Advanced Research WRF (ARW) model (version 3.7.1) (Skamarock and Klemp 2008). WRF is a state-of-the-art atmospheric modeling system designed for both mesoscale and microscale meteorological research and numerical weather prediction.

[1] Data Source: https://rda.ucar.edu/datasets/ds351.0/#!description
[2] Data Source: https://rda.ucar.edu/datasets/ds507.5/

WRF is a non-hydrostatic model with advanced dynamics, physics, and numerical schemes. The WRF model is fully compressible. Its vertical coordinate is a terrain-following hydrostatic pressure coordinate. Detailed descriptions of the model can be found in the model manual (Skamarock, Klemp et al. 2005) and also on the WRF user website[3].

In order to describe the vertical pressure levels, sigma coordinates were adopted in the model and a two-way nesting was used for the interaction between mother and child domains. Each domain is comprised of 40 vertical pressure levels with the top level set at 100 hPa. This study used a grid spacing of 4 km in the inner domain and 12 km for the outer domain. The nested domain centered at 36.746^0 N and 75.736^0 W. The parent domain contains 235 x 235 grid points with 12 km resolution and the nested domain contains 226 x 223 grid points with 1/16th of a degree (4 km) resolution (Figure 3). The Hampton Roads region was the center of the inner domain. The model used MODIS-derived 20 category land cover and soil texture at 30 arcs second resolution. The WRF model was implemented for the two-way nested domain with two domains to convert the coarse resolution of NARR (32 km) data to high-resolution WRF domain data (1/16th of a degree). The WRF model simulations were performed with the adaptive time step option to minimize processing time in the Advanced Research Computing (ARC) at Virginia Tech.

The WRF model comprises different physical parameterizations, including micro physics to cumulus, surface, planetary boundary layer and radiation physics. The model performance is highly dependent on the parameterization schemes. In this study, NCAR recommended parameterization schemes were used. The main physics schemes used in this study included the WRF Single-Moment 6-class (WSM6) microphysics scheme, the new Kain–Fritsch cumulus parameterization, the Yonsei University planetary boundary layer, the Dudhia shortwave radiation, and the Rapid Radiative Transfer Model (RRTM) longwave radiation schemes. Noah land surface model was used to represent the land surface physics and hydrology.

The North American Regional Reanalysis (NARR) data was used to define initial and boundary condition of the WRF model. NARR is an extension of the NCEP Global Reanalysis. It is a high resolution (~32 km) combined model and assimilated dataset, from 1979 to near present and is provided 8-times daily as 3-hour composite, daily and monthly on a Northern Hemisphere Lambert Conformal Conic grid. The data has 29 pressure levels from 1000 hPa to 100 hPa[4]. The WRF model was initialized on 1 October 2015 12:00 UTC and continued the simulation until 31 December 2016 23:00 UTC.

2.3 Precipitation and Upper Air Meteorology Data Assimilation

WRF Data Assimilation System (WRFDA) 4D-Var was used to assimilate NCEP Stage IV radar and gauge precipitation data along with upper air meteorology data. Both the data in the LITTLE_R format were assimilated in the 4D-Var system. NCEP Stage IV archived data at NCAR is in GRIB1 format. A tool "precip_converter" was used to reformat GRIB1 observations into the WRFDA-readable ASCII (LITTLE_R) format[5]. According to the WRFDA user manual, 6-hour precipitation accumulations from Stage IV are the ideal observations to be assimilated for simulating hourly precipitation. In this study, before the WRFDA exercise, the precipitation in

[3] http://www2.mmm.ucar.edu/wrf/users/
[4] Data source: https://rda.ucar.edu/datasets/ds608.0/
[5] source: http://www2.mmm.ucar.edu/wrf/users/wrfda/download/precip_converter.tar.gz

Stage IV 4 km cell corresponding to latitude-longitude locations of HRSD gauges was adjusted with gauge observations at 6-hour time scale. The data assimilation process was employed only on the outer domain (12km domain). The background error covariance (b_0) was obtained by computing the average difference between 12- and 24-h forecasts.

2.4 Results

2.4.1 Comparison of Rainfall Simulated by WRF and Gauge Observations

The WRF simulated precipitation was compared with 12 HRSD gauge observations at 15 minute, 1-hour and 24-hour time-steps. The statistics of comparison, percentage bias (PBIAS) and root mean square (RMSE) are given in Table 2. Percentage bias was significantly high with high temporal resolution (15-min) whereas the 24-hour time step analysis, PBIAS reduced significantly. RMSE was least with hourly rainfall data compared to 15-min and 24-hour analysis. The highest and lowest PBIAS was observed at location MMPS-144 and MMPS-185 respectively in all the three time-step analysis (Table 2). For all 12 locations, the WRF predicted rainfall was less by approximately 50% at the 15 –minute time step. However, the simulation results improved when the temporal aggregation was higher and the estimation was less by 21% at the hourly time-step.

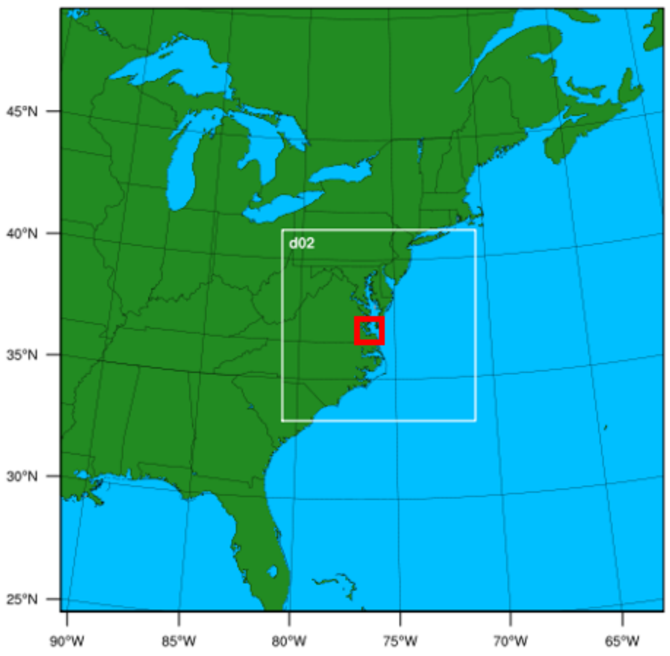


Figure 3 WRF model domain with study area highlighted in the inner red box (Figure 2).

Table 2 Validation Statistics for the WRF Simulated Rainfall with HRSD Observations

	15-min		1-hour		24-hour	
Station	PBIAS (%)	RMSE, mm	PBIAS (%)	RMSE, mm	PBIAS (%)	RMSE, mm
MMPS-004	-49.6	3.2	-15.7	1.3	-15.2	7.1
MMPS-0.36	-61.6	2.7	-34.7	1.4	-33.9	8
MMPS-140	-58.6	2.7	-31.4	1.5	-30.6	9.2
MMPS-144	-63.1	2.5	-38.7	1.4	-38.1	8.8
MMPS-146	-54.3	2.8	-24.8	1.2	-23.1	7.9
MMPS-160	-53.0	2.8	-22.4	1.0	-21.9	6.9
MMPS-163	-62.1	2.6	-36.1	1.2	-34.9	8.6
MMPS-171	-52.5	2.8	-21.4	1.7	-22.0	22.0
MMPS-185	-36.8	2.7	2.4	0.9	3.4	7.4
MMPS-255	-39.8	2.7	-1.2	0.9	0.3	6.7
MMPS-256	-51.0	2.5	-18.7	0.9	-17.3	4.5
MMPS-0932	-52.1	2.9	-18.8	1.4	-18.2	6.9

Since the proposed study area is Baltic Ave. and 21st Street and Shore Drive, the study selected three HRSD rainfall gauges close to this locations. The selected gauges were MMPS-004, MMPS-093-2, and MMPS-036. Figure 4 shows the 24-hour time series of WRF and HRSD rainfall (mm) for the selected three gauges. At MMPS-004 location, WRF-simulated rainfall was less compared to HRSD for most of the time. But for July and September events (see Table 3), WRF overestimated the rainfall by 11 to 43 mm. However for the October event, WRF underestimated rainfall by 90 mm. WRF simulated rainfall showed discrepancies of up to 75 mm depth (24-hour) for the rainfall event on August 18. Similar results were evident for all three locations. At MMPS-036 location, WRF simulated rainfall for two small events on August 8th and 18th showed some differences. Table 3 shows the event-wise comparison for MMPS-036. In general, WRF underestimated all three events and the highest underestimation was 108 mm for the October 8th event. A similar characteristic was observed at site MMPS-0932 where the highest underestimation was observed for July 31st event (80 mm) and lowest was for the September event (25 mm).

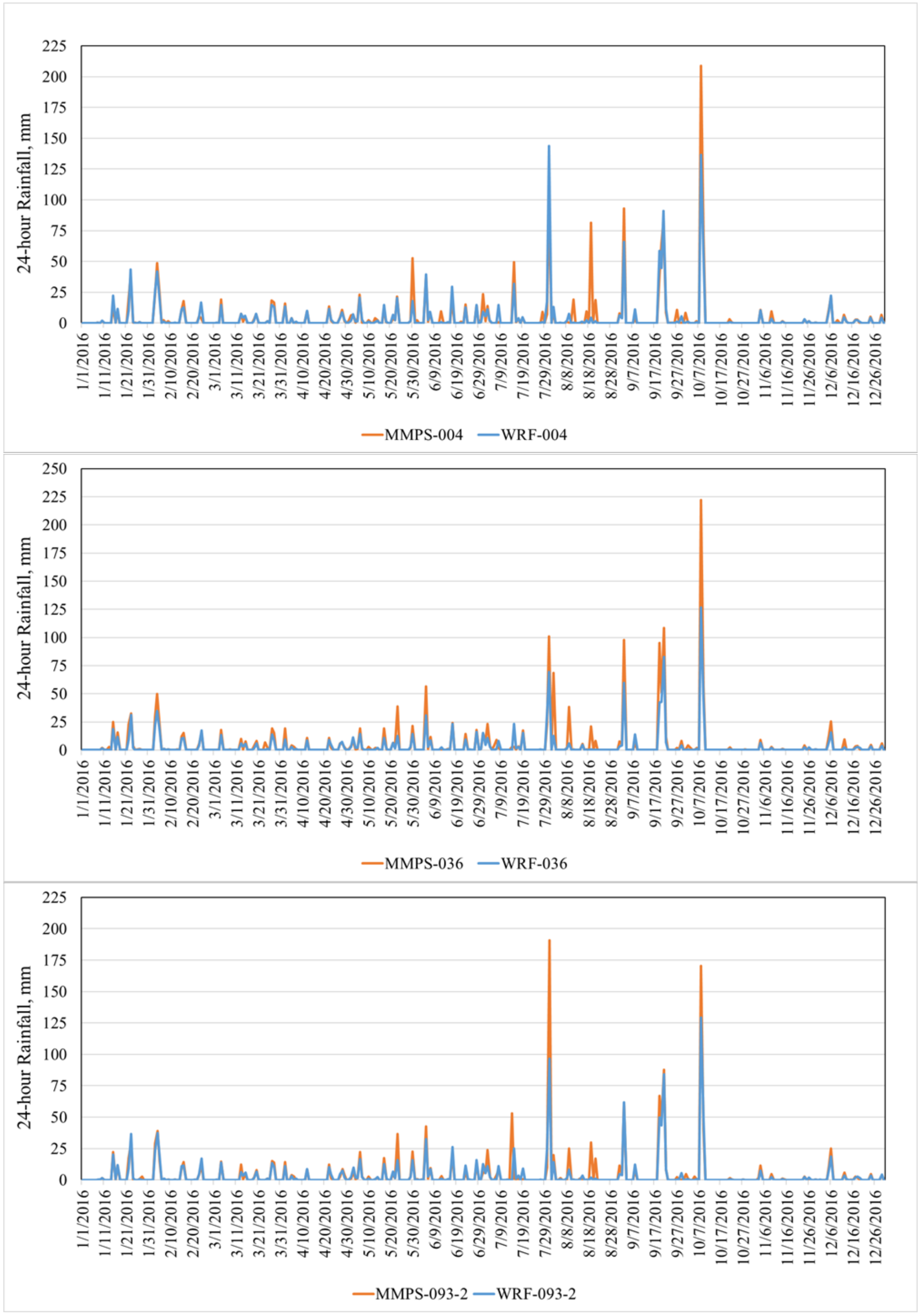


Figure 4 Comparison of 24-hour rainfall between WRF and HRSD observations for the three sites in the study area.

Table 3 Comparison of WRF Rainfall with HRSD Observations for Three Rainfall Events

Events	WRF-004	MMPS-004	MMPS - WRF	WRF-0.36	MMPS-0.36	MMPS-WRF	WRF-0932	MMPS-0932	MMPS-WRF
July-28-31	162	119	-43	101	124	23	121	201	80
Sep-18-21	206	195	-11	175	266	92	186	210	25
Oct-5-10	194	283	90	174	282	108	180	214	34

Figure 5 shows the cumulative rainfall distribution at the three sites considered in this study. The difference in cumulative rainfall between WRF and HRSD was different at all the three sites. The highest difference in cumulative rainfall (189 mm) was observed at MMPS-0932. Both the MMPS-004 and MMPS-036 sites showed similar differences (79 mm and 88 mm respectively) in cumulative rainfall. Even though, at all the three sites, WRF showed similar rainfall distribution compared to HRSD observations, WRF underestimated rainfall. This was true with all 12 validation sites over Virginia Beach.

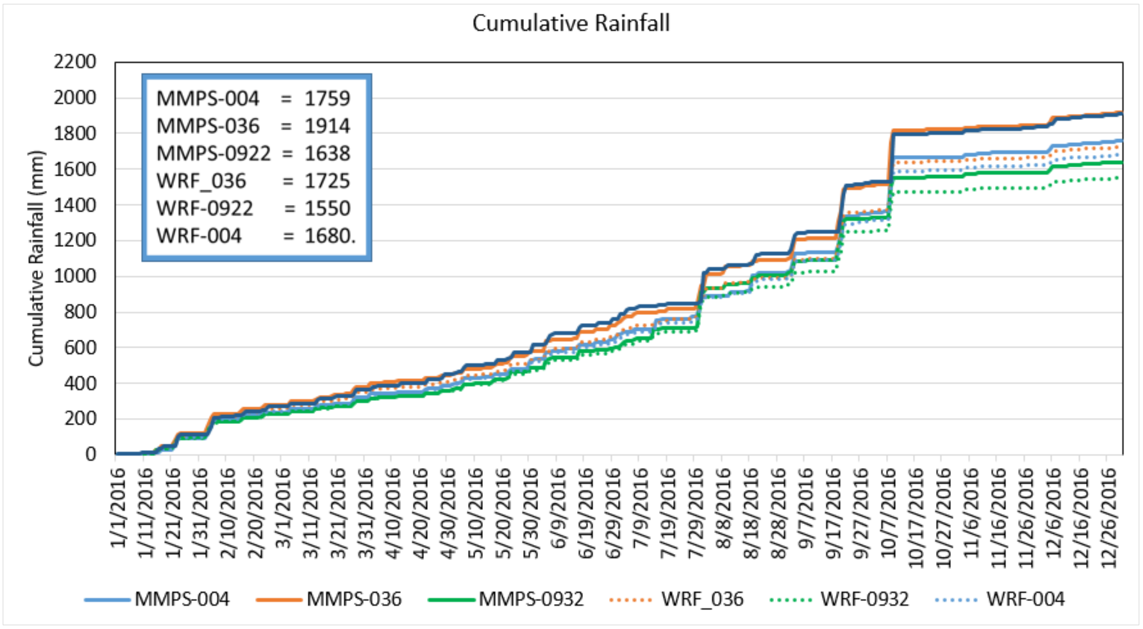


Figure 5 Comparison of cumulative rainfall between WRF and HRSD observations for selected three sites.

2.4.2 Comparison of Rainfall Simulated by WRF and Gauge Observations

Figure 6 compares the spatial distribution of rainfall simulated by WRF with STAGE IV observations. It is clear from Figure 6 that, for all the three selected high rainfall events, WRF was relatively dry compared to STAGE IV. Even though STAGE IV was assimilated to WRF through the WRFDA system, WRF was unable to produce the same intensity rainfall. A similar finding was reported by Lin, Ebtehaj et al. (2015). They used point –scale 4D-Var precipitation data assimilation to WRFDA to produce precipitation estimates. Their results showed significant underestimation of summer precipitation over the central US by WRF with a noticeable difference in spatial rainfall distribution. From Figure 6, while WRF showed a large difference (depth, mm) in the spatial distribution of high rainfall events (September and October), the model was effective in simulating similar spatial patterns to that of STAGE IV.

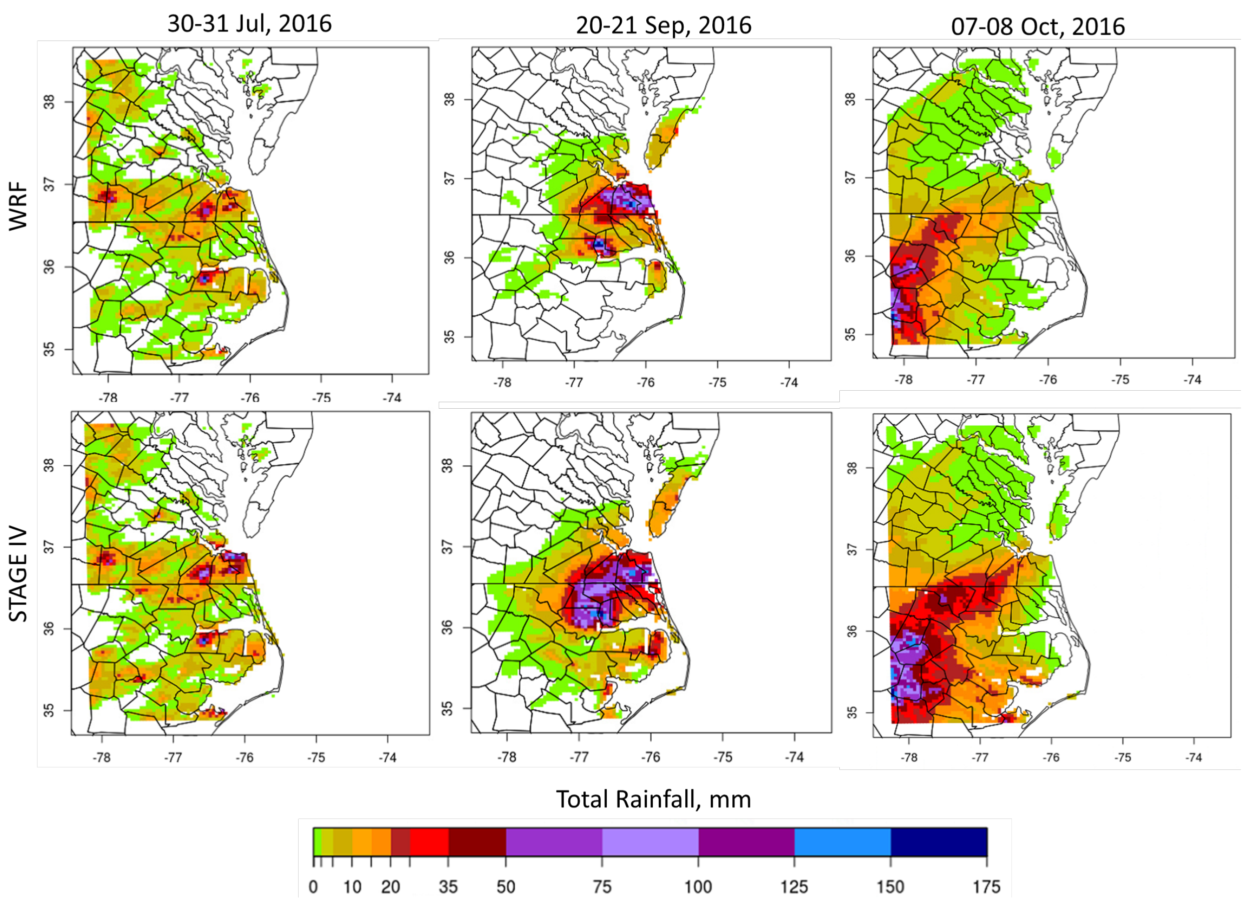


Figure 6 Comparison of event based cumulative rainfall between WRF and STAGE IV observations.

2.4.3 Surface Runoff from WRF with Gauge Observed Rainfall

As an extension to proposed research of producing high-resolution hourly rainfall maps, the study analyzed the surface runoff. WRF Hydro model was used to simulate the streamflow at 100 meters resolution. Since the WRF Hydro model needs to be reconfigured with additional data including, stream characteristics such as the depth and width, stream slope, groundwater table information, the model results are not included in this study. However, WRF-Noah LSM derived surface runoff was analyzed (at 4 km and 15-minute time step) for the study area. Figure 7 shows the comparison of observed rainfall with WRF-Noah LSM simulated surface runoff. The simulated runoff showed similar patterns of rainfall at all three sites considered. The highest runoff (2 mm at the 15 minute time-step) was simulated at the MMPS-004 site, which is close to Shore Drive.

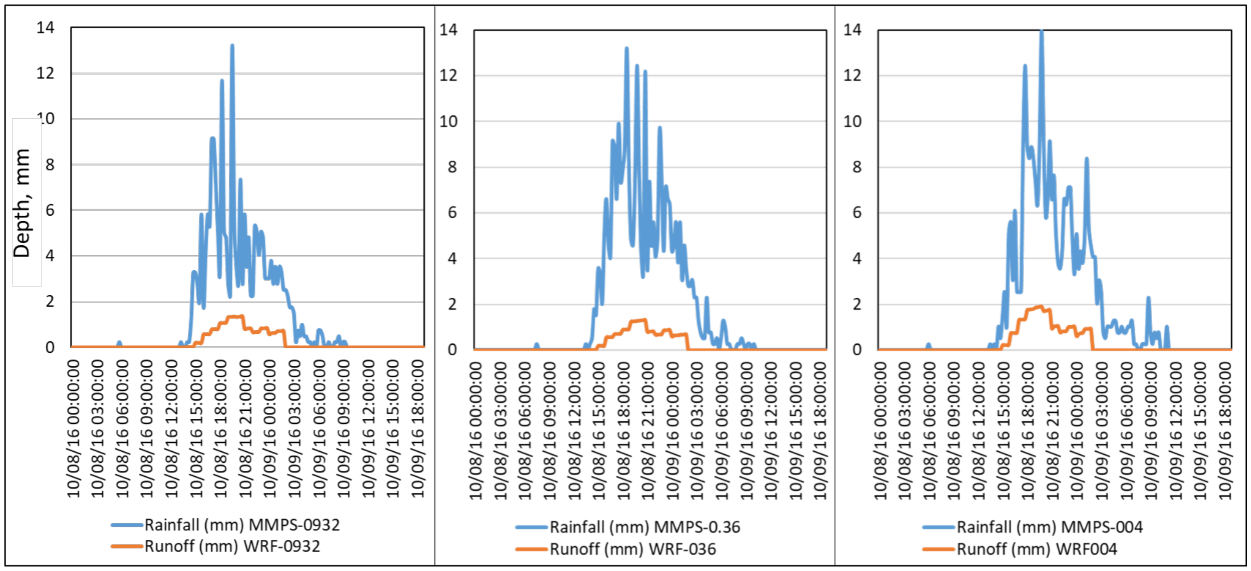


Figure 7 HRSD rainfall observations with WRF-Noah derived surface runoff for three sites and the highest rainfall event (October 8th)

2.5 Conclusions

Overall, WRF simulated rainfall was within the range of uncertainty for a large scale model in capturing the rainfall events with data assimilation. The performance of WRF with data assimilation was similar to previous studies. The sources of bias in WRF rainfall estimates may be due to forecast error, error in the assimilated data, and bias in the initialization or boundary conditions. While the products generated from this model are useful to link with hydrologic, hydraulic models and therefore assessing the transportation network under flooding conditions, for the improved dynamical downscaling of convective precipitation, future research is needed to improve the predictability of extreme rainfall events by assimilating accurate radar precipitation and soil moisture data.

3 Hydrology

Urban coastal flood events can be modeled using physically-based 1D (e.g. (Mark, Weesakul et al. 2004)) or 2D models (e.g. (Mignot, Paquier et al. 2006, Hunter, Bates et al. 2008)). However, the simplified representations of reality used in physically-based models can be a limitation given the combination of variables and their interactions and the complexity of the physical environment which can influence urban coastal flooding. Another approach in hydrology is data-driven modeling (Solomatine and Ostfeld 2008). These models do not only adjust certain parameters of a model, as is done in the calibration of a physically-based model, rather they map model inputs to outputs without attempting to model the governing physical processes to any degree. Thus, the relationship between the inputs and outputs is not assumed, as in physically-based models, but learned. No simplifying assumptions are made when using data-driven models.

The recent increase in availability of earth observation data, coupled with advances in machine learning algorithms, have expanded the possibilities and use of data-driven modeling in hydrology. Machine learning algorithms have been used extensively in hydrology for applications such as predicting reservoir operations (Yang, Gao et al. 2016), soil mineral weathering (Povak, Hessburg et al. 2014), streamflow (Solomatine and Xue 2004, Wang, Chau et al. 2009, Yang, Asanjan et al. 2017), groundwater potential (Naghibi, Moghaddam et al. 2017), and groundwater level (Sahoo, Russo et al. 2017). The use of data-driven and machine learning algorithms in flooding applications specifically include Tehrany, Pradhan et al. (2013), Wang, Lai et al. (2015), and Tien Bui, Pradhan et al. (2016) who predicted areas susceptible to flooding, Adamovic, Branger et al. (2016) who modeled flash flooding on a regional scale, and who predicted stream flow for flood forecasting.

Both physically-based models and data-driven models have strengths and weaknesses. With physically-based models, it is easy to determine if model results make physical sense. This is not often true of data-driven models. Because data-driven algorithms do not attempt to mimic physical relationships, they can produce results that do not make physical sense. Many of the models produced by data-driven techniques are considered black-box models meaning that there is no way to see how the algorithm arrives at the model or the results it produced. Related to this is that when black box models are used, custom guidelines or rules which make physical sense but are not evident in the model training data cannot be added to the model. This may be a drawback to decision makers who would like to incorporate custom rules into the model based on domain knowledge gained through experience.

Given the complexity of modeling urban coastal flooding, the objective of the hydrology portion of this project was to use data-driven models to predict flooding of roadways in Virginia Beach given rainfall predictions and tide levels. Water table level and wind conditions were also used in the model training. This study used Random Forest, a popular machine learning algorithm. Different approaches have been used to model flooding in urban coastal settings such as Virginia Beach. Tidal records have been statistically analyzed from tide gauges in the United States to estimate the amount of time that coastal cities have experienced flooding in the past several decades and project flooding in the coming decades (Ezer and Atkinson 2014, Sweet and Park 2014, Moftakhari, AghaKouchak et al. 2015, Ray and Foster 2016). Many physically-based models have been used to model urban coastal flooding (Bates, Dawson et al. 2005, Smith, Bates et al. 2011, Gallien, Sanders et al. 2014). This is some of the first research to demonstrate the use of machine learning algorithms to model urban coastal flood occurrences.

3.1 Background

3.1.1 Study Area and Street Flooding Record

Although the study area for this project is Virginia Beach, the street flooding model was developed using data only available from the adjacent city of Norfolk, Virginia. Often observational data from flooding events is a limiting factor in creating useful flood models (Smith, Bates et al. 2011). Because such data is often sparse, photographs of flooded locations and personal interviews have been used out of necessity in the calibration and verification of flood models (Smith, Bates et al. 2011). Even satellite imagery has been used to estimate flooding extents (Ireland, Volpi et al. 2015). Norfolk city workers have kept a record of individual flooded street locations starting with Hurricane Nicole on 30 September 2010 up until the time of writing this report. In the past six years, many of these locations, seen in Figure 8, have been reported as flooded only one or two times but others have been reported as flooded up to 19 times.

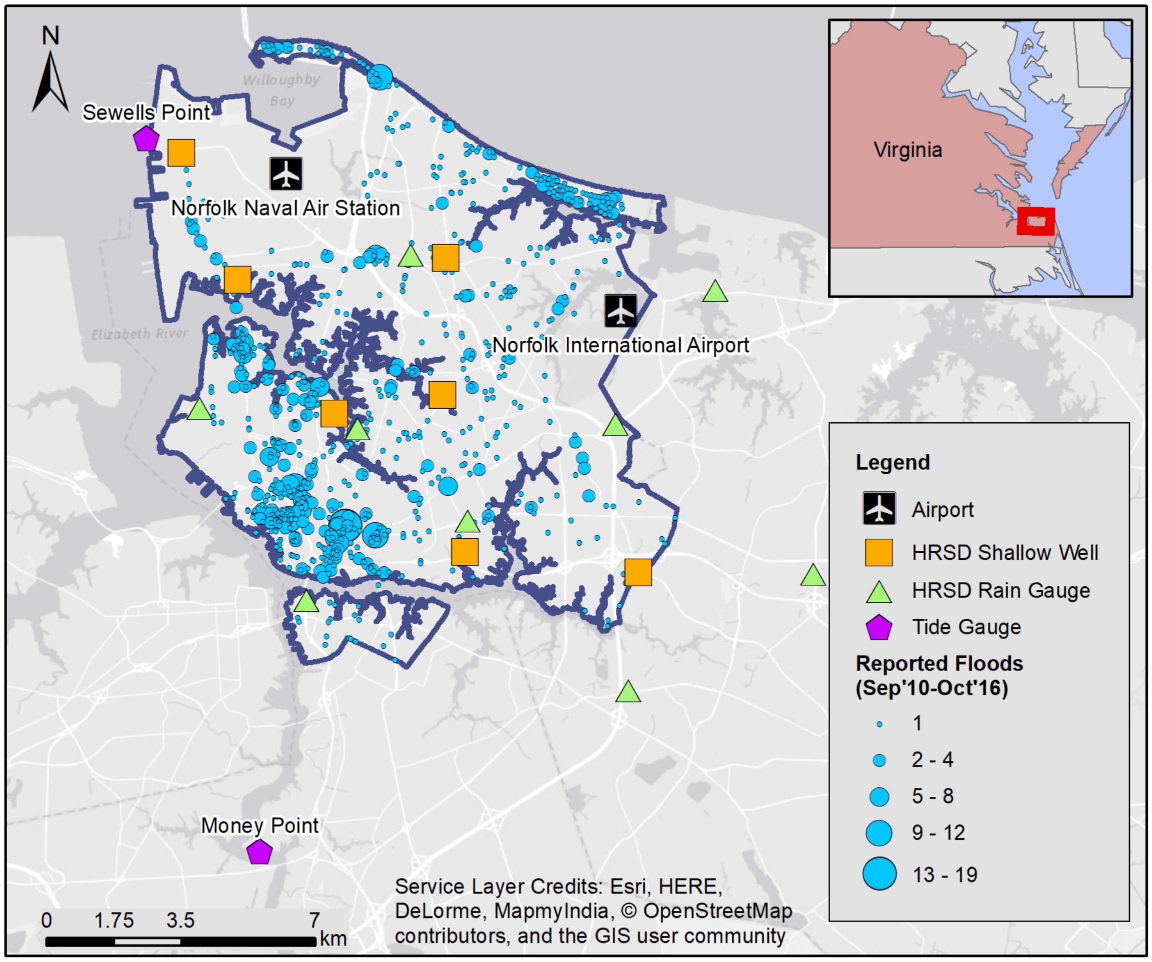

N
Sewells Point
Willoughby Bay
Norfolk Naval Air Station
Norfolk International Airport
Elizabeth River
Money Point
Virginia
Legend
Airport
HRSD Shallow Well
HRSD Rain Gauge
Tide Gauge
Reported Floods (Sep'10-Oct'16)
1
2 - 4
5 - 8
9 - 12
13 - 19
0 1.75 3.5 7 km
Service Layer Credits: Esri, HERE, DeLorme, MapmyIndia, © OpenStreetMap contributors, and the GIS user community

Figure 8 Street flooding, environmental sensing stations and airports in Norfolk, Virginia USA

3.1.2 Description of Model Input and Output Data

The objective of the hydrology portion of the modeling is to predict the number of flood reports resulting from a given storm event based on the environmental conditions of that event. The environmental condition input data for our model consisted of rainfall, water table level, wind, and tide level observations. These were obtained from the Hampton Roads Sanitation District (HRSD) and the US National Oceanic and Atmospheric Administration (NOAA). Rainfall, water table elevation, and wind direction and wind speed data were obtained from HRSD. The rainfall observations from HRSD were on a 15-minute time scale, and the water table elevations and wind data were on a 2-minute time scale. From NOAA we obtained 6-minute water elevations and daily high and low tides recorded at the Sewells Point (NOAA 2017) and Money Point (NOAA 2017) tide gauges. Wind speed, wind gust, and wind direction data recorded at the Money Point station at 6-minute time intervals were used as well. Daily rainfall and wind data collected at two airports in the study area, Norfolk International Airport (NOAA 2017) and Norfolk Naval Air Station (NOAA 2017), were also obtained from NOAA. The rain gauge, water table, wind, and tide gauge stations and airports are shown in Figure 8. All of the raw data together consisted of more than 15 million observations. To keep the time series data organized, a simplified version of the Consortium of Universities for the Advancement of Hydrologic Sciences Incorporated (CUAHSI) Observations Data Model (Horsburgh, Tarboton et al. 2009) was implemented in a sqlite database.

The output data used in model training and evaluation was the number of flooded locations per storm event as reported by City of Norfolk workers. A total of 45 storm events (listed in Table 4) were reported to have caused flooding in the period of record with the number of reported floods per event ranging from 1 to 159 (Figure 9). Eight of the events were hurricanes, the rest were unnamed or given generic names by city workers.

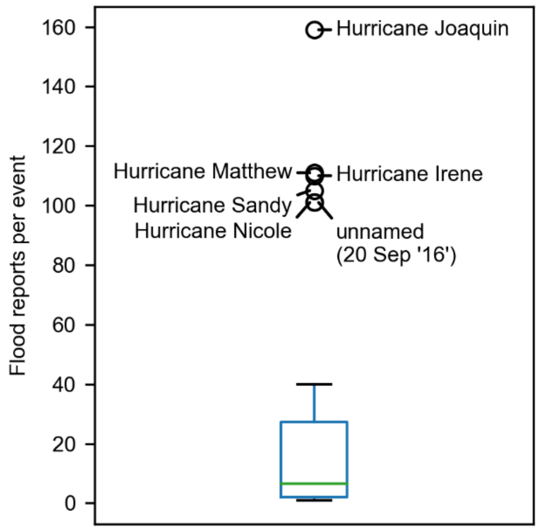


Figure 9 Summary plot of reported floods per event in Norfolk, VA Sep. 2010-Oct. 2016

Table 4. Events recorded to have caused flooding in Norfolk, VA Sep. 2010-Oct. 2016

Event Date	Event Name	Flood Reports
29 Sep 2015	Hurricane Joaquin	159
05 Oct 2016	Hurricane Matthew	111
27 Aug 2011	Hurricane Irene	110
28 Oct 2012	Hurricane Sandy	105
20 Sep 2016	unnamed	101
30 Sep 2010	Hurricane Nicole	101
02 Sep 2016	Hurricane Hermine	40
10 Jul 2014	Thunderstorms	39
09 Oct 2013	Heavy Rain	36
16 May 2014	Heavy Rain	35
08 Sep 2014	Rainy Monday	31
20 Jan 2016	January Winter Weather	26
24 Jul 2014	unnamed	18
24 Sep 2015	Noreaster	16
19 Sep 2016	Heavy Rain	11
02 Mar 2015	unnamed	10
11 Jul 2015	Thunderstorm	10
19 Jul 2016	Thunderstorm	9
25 Feb 2016	unnamed	8
03 Jul 2014	Hurricane Arthur	8
31 Jul 2016	Thunderstorm	8
02 Jul 2015	unnamed	7
15 Jan 2016	unnamed	6
03 Jun 2016	Severe Weather - 6/5	6
04 Sep 2014	Thunderstorm	5
19 Jun 2014	Thunderstorms	5
01 Feb 2016	unnamed	5
23 Nov 2014	unnamed	4
13 Sep 2014	Saturday Storm	3
30 Dec 2015	Heavy Rainfall	3
09 Jul 2014	Thunderstorms	2
25 Jul 2016	Bernie (Training)	2
10 Jun 2016	unnamed	2
29 Sep 2014	unnamed	2
16 Dec 2010	Snow	2
24 Feb 2016	February 24th Storm	1
17 Nov 2014	Storm	1
30 Oct 2015	unnamed	1
20 Jul 2016	unnamed	1
17 Sep 2015	unnamed	1
02 Sep 2015	unnamed	1

18 Aug 2014	unnamed	1
24 Sep 2014	Heavy Rain	1
09 Jun 2014	unnamed	1

3.1.2.1 Random Forest

Random forest, developed by (Breiman 2001), is an ensemble machine learning algorithm which aggregates a large number of classification or regression trees (CART) to make a prediction (Breiman, Friedman et al. 1984). In the training of a CART, rules based on the response variable are developed to divide observations until the resulting groups of predictions have an acceptably low amount of impurity. The CART's rules are a collection of linear divisions of the observation data which, together, create a non-linear decision surface which can be complex.

One of the main problems of CARTs is that they are prone to overfitting to the training data and thus perform poorly when given unseen data (Murphy 2012). Random Forest is an approach that addresses this weakness. When an individual CART is trained in the Random Forest algorithm, a portion of the input records and predictor variables are randomly selected as input used in the training. This process is repeated for the number of CARTs specified by the modeler creating a group of CARTs, each trained on a randomly selected subset of the records and input variables. This group makes up Random Forest. When a prediction is made, each CART makes a prediction based on the rules developed in the training and the aggregate of their individual predictions becomes the overall prediction. The random selection of input records and variables in the training of the individual CARTs creates variety in the weak learners thus avoiding overfitting of the model to the training data. This is one of the major strengths of the Random Forest algorithm.

Beyond the actual predictive capabilities of Random Forest, the algorithm can be used to understand feature importance. Because many CARTs are being produced with different sets of input variables, the Random Forest algorithm learns and records the relative importance of the input variables in predicting the output. This capability is especially attractive in the present case as one of the objectives of this study is to understand the relative importance of explanatory variables in predicting street flooding.

3.2 Methods

3.2.1 Input Data Pre-processing

For the modeling, all of the raw input environmental data were aggregated to a daily time scale to match the time scale of the flood reports. The resulting dataset consisted of 2,171 records of daily average environmental conditions from September 2010 through October 2016. The aggregated environmental input variables for the model are shown in Table 5. Different approaches of aggregation were taken for the different measurements. Four derivatives of the raw HRSD 15-minute rainfall data were included in the model as inputs: the daily cumulative rainfall, the maximum hourly rainfall, the maximum 15-minute rainfall, and the cumulative rainfall in the previous three days. The different derivatives of the 15-minute rainfall were included with the intent of accounting for different types of floods. For example, in a flash flood, a large amount of rain falls in a short amount of time. In such a case, the maximum 15-minute rainfall value, which would be high, would be a better predictor variable than the total daily rainfall, which could be low.

As with the 15-minute rainfall data, several tide-related variables were model inputs including daily high and low tides, daily average tide level, and tide levels at the time of maximum 15-minute and maximum hourly rainfall totals. In coastal cities, such as Norfolk, the timing of rainfall and the tide levels can have an effect on flooding. For example, if tide level is especially high when a large amount of rain falls, the stormwater infrastructure may be underwater and therefore not function properly causing more flooding than if the tide were low and the same amount of rain fell. To, in some measure, account for such interactions between tide and rainfall, the tide level at the time of the maximum 15-minute rainfall and the tide level at the time of the maximum hourly rainfall were included as model inputs.

Table 5 Predictor Variable Names and Descriptions

Feature Name	Units	Source Organization	Abbreviation
Daily cumulative rainfall	Inches	HRSD, Airports	RD
Maximum hourly rainfall	Inches	HRSD	RHRMX
Maximum 15-minute rainfall	Inches	HRSD	R15MX
Cumulative rainfall in previous three days	Inches	HRSD	R3D
Average water table elevation	Feet above NAVD88	HRSD	GW_AV
Average tide level	Feet above MSL	NOAA	TD_AV
Tide level at time of maximum 15-minute rainfall	Feet above MSL	NOAA	TD_R15
Tide level at time of maximum hourly rainfall	Feet above MSL	NOAA	TD_RHR
High tide	Feet above MSL	NOAA	HT
Higher high tide	Feet above MSL	NOAA	HHT
Low tide	Feet above MSL	NOAA	LT
Lower low tide	Feet above MSL	NOAA	LLT
Average daily wind speed	Miles per hour	Airports, NOAA, HRSD	AWND
Average daily wind direction	Degrees	Airports, NOAA, HRSD	AWDR
Average wind speed over 6-minutes	Miles per hour	Airports, NOAA, HRSD	WSF6
Average wind direction over 6-minutes	Degrees	Airports, NOAA, HRSD	WDF6
Average maximum 2-minute wind gust over 6-minutes	Miles per hour	Airports, NOAA, HRSD	WGF6
Average wind speed over 2-minutes	Miles per hour	Airports, NOAA, HRSD	WSF2
Average wind direction over 2-minutes	Degrees	Airports, NOAA, HRSD	WDF2

The values on the daily time scale were averaged across all the stations which recorded that variable. For example, the 'Daily cumulative rainfall' is the total daily rainfall averaged across all 11 the rain gauges. This spatial averaging was done for two reasons. First, this was a way of making the model applicable more generally. If the values were not averaged across the stations, when using the model, an input would be needed for every station. In this case, for example, to make a prediction a daily rainfall value would be needed for all 11 rainfall stations. This could be a problem if all 11 stations were not functioning. Second, by averaging the variables across stations, the importance of the environmental variables can be more clearly learned. For some of the stations there is a considerable amount of missing data over the six years of the study period. If the variables were not averaged across measuring stations, it would appear that the stations that had less missing data were more important which would make it more difficult to understand the importance of the actual environmental variables compared to the consistency of measurements at an individual station.

To reduce noise in the data, daily environmental conditions which, under reasonable assumptions, would not cause flooding were not used in the modeling procedure. The assumption on which this decision was made was based on total daily rainfall. Of the 45 events for which flooding was reported, 42 had an average daily rainfall total of 0.01 inches or greater. For two of the events with less than 0.01 inches of rainfall, only one flooded location was reported. Two flooded locations were reported for the third. Given that very minor flooding was reported for these three events, only daily records with an average of at least 0.01 inches of rainfall were considered in the model training and evaluation. This reduced the number of total daily records used to train and evaluate the model from 2171 to 814.

The dataset of daily environmental conditions for storm events from September 2010 to October 2016 and the number of reported flood locations for each event were randomly divided into a training set (70%) and an evaluation set (30%). This included the 42 storm events for which flooding was reported and the 772 events which for which no flooding was reported. The distribution of the number of reported floods (the output variable) in both the training and evaluation datasets was similar to the distribution of the number of reported floods of the entire data set.

3.2.2 Model Training and Evaluation

The R programming language (version 3.3.3) was used to train and evaluate the Random Forest model. The 'randomForest' package (version 4.6.12) was used for the Random Forest model. The model was trained using only the training data. For the Random Forest, training involved training 500 regression trees, each optimized for a randomly selected two-thirds of the training records and a maximum of six predictor variables.

Once trained, the evaluation dataset of input environmental variables which was kept back in the model training, was used as input for the model. Applying the model to the evaluation data produced a predicted number of reported floods for each of input values. The predicted numbers of reported floods were compared with the known number of reported floods. Root mean squared error (RMSE) and mean absolute error (MAE) were the two metrics used to evaluate the model. Given that our dataset is somewhat small, to account for potential bias in the division of the data into training and evaluation sets, the division was made, the model was run, and the results were recorded 100 times independently.

3.2.3 Applying model to Virginia Beach

Once the model was trained with the input and output data from Norfolk, it was applied to the Virginia Beach area and road network. Two major differences were made to the modeling procedure to apply the model to Virginia Beach. First the model was adapted so that the model output was in terms of the portion of roadways predicted to flood instead of the number of roadways predicted to flood, and second, input rainfall data was changed to be output from the

WRF model described above instead of the rainfall observations. To translate to a number of flood locations in Virginia Beach that would flood under a set of environmental input conditions, the number of total possible flood locations in both Norfolk and Virginia Beach was needed. Most of the flood reports in the record made in Norfolk were at roadway intersections, therefore, only intersections were considered as possible flood locations. Using the ArcGIS Intersect tool, the total number of roadway intersections was found in both cities. Many of the intersections output from the Intersect tool were located very close to each other, some at the exact same geographic location. Because of this, intersection near others were disregarded until every intersection was at least 100 feet from the next. Additionally, since workers could not report on floods inside military bases, the total number of intersections in Norfolk used to calculate the percentage of intersections flooded did not include intersections on the military bases.

To incorporate rainfall predictions, model output from WRF (described above) for the Virginia Beach was used as model input for the Random Forest model. Since average daily rainfall over the study area was one of the inputs to the Random Forest model, the predicted rainfall from WRF was spatially averaged over the study area. To have predictions on a 15-minute scale, a 24-hour rolling sum of the rainfall was performed. The maximum 15-minute and hourly rainfall values in a rolling 24-hour time period was also input for the Random Forest model. Tide values made up the other inputs for the Random Forest model. The tide values used were observed values since tidal prediction was not in the scope of this project.

The output from the adapted Random Forest model was a ratio of flooded intersections. Given the total number of intersections in Virginia Beach, the number of intersections predicted to flood was found. Predicting exactly which roadways will flood is difficult and since little data to explore this question was available, as a first approach to this question, the basic assumption was made that an intersection with a lower elevation will flood before an intersection with a higher elevation. Therefore, all of the intersections in Virginia Beach were ranked by elevation and the intersections predicted to flood, given 24-hour rainfall predictions and tide observations, were the number of lowest intersections predicted by the Random Forest model. Unfortunately, since no street flooding observations were available from Virginia Beach, it was not possible to verify the assumptions made or the application of the model to Virginia Beach.

3.3 Results and Discussion

3.3.1 Model Results

The RMSE, MAE, and standard deviations of training and evaluation predictions from the Random Forest model are reported in Table 6. The RMSE was significantly higher in the evaluation phase compared to the training phase both when considering all of the days and when considering only days where floods were recorded. In both cases, the evaluation RMSE was about two times the training RMSE, suggesting that the model was overfit to the training data. Figure 10 shows the predicted number of flood reports made by the Random Forest model in the training and evaluation phases.

Table 6 Summary of Model Training and Evaluation Results

	RMSE		MAE		SD	
	Training	Evaluation	Training	Evaluation	Training	Evaluation
All days	2.31	5.06	0.41	0.95	2.47	4.58
Non-zero flood days	9.79	21.41	5.38	12.29	2.55	4.7

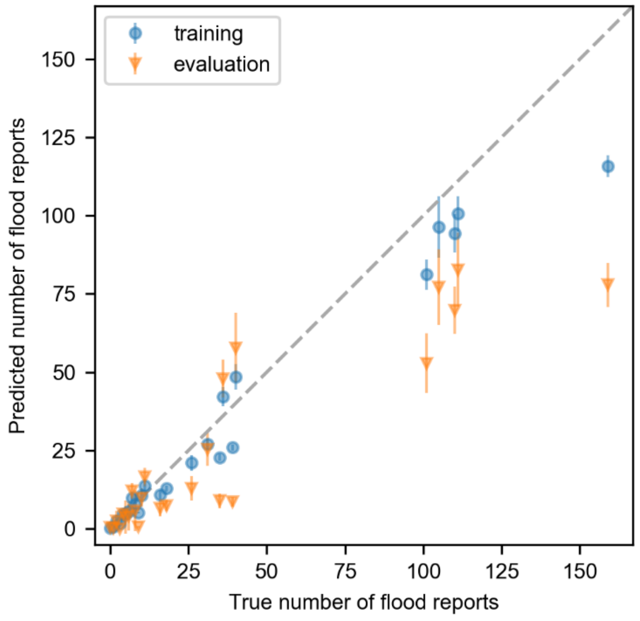


Figure 10 Random Forest model results. Bars represent +- one standard deviation.

3.3.2 False Negative and False Positive Predictions

Of particular interest to a decision maker when evaluating a model are false positive and false negative predictions. Statistics summarizing false negative and false positive predictions in the model evaluation phase are given in Table 7. Since predictions were on a continuous scale and true flood reports were integers, the predictions were rounded to the nearest integer when calculating the false negative and false positive rates. For example, a prediction was considered false positive when the number of flood predictions was at least 0.5 (which would round to 1) and the true number of flood reports was zero.

The false positive rate for the model was 8.92%, however, most of the false positive predictions were less than 1.5 flood reports, which would round down to one. The maximum false positive prediction was 36. The model had much higher false negative rate, 31.50%. Importantly, the model did not have any false negative predictions when the true number of reported floods was high. The maximum number of true flood reports when a false negative prediction was nine.

Table 7 False Positive and False Negative Statistics for Random Forest Model

	False Positives	False Negatives
rate (%)	8.92	31.5
count	2059	384
25%	0.75	1
50%	1.45	2
75%	3.85	5
max	36.04	9
mean	3.33	2.96
std	4.49	2.47

3.3.3 Variable Importance

Figure 11 shows the importance of each of the variables as calculated from model in terms of the percent increase in mean squared error (MSE) when each of the variables is permuted individually. The total daily rainfall value was by far the most important of the variables. This was much more important than any of the other variables derived from the raw rainfall data, including the maximum hourly and maximum 15-minute rainfall values, suggesting that, in this record, large rainfall volumes caused more flooding than high rainfall intensities. The next three variables in terms of prediction importance were related to tide: low tide, higher high tide, and lower low tide. The height of the water table did not add appreciable predictive power to the model. This suggests that the water table did not impact flooding in a significant way and that, for the floods reported in this record, groundwater emergence, which is expected to cause flooding in coastal environments in the future (Hoover, Odigie et al. 2016), was not a factor.

The variable importance results shown in Figure 11 are supported by the raw data shown in Figure 12. The number of flood reports has clear positive relationship with the daily rainfall. The same is true for low tide and higher high tide. The relationship is much less clear for the maximum hourly rainfall and the maximum 15-minute rainfall, but according to Figure 11, the Random Forest model was still able to glean some meaningful information from the data. Interestingly, the average tide, visually, has a much clearer relationship with the number of flood reports compared to the maximum hourly, and maximum 15-minute rainfall values but is considered less important by the Random Forest algorithm. This is likely because most of the information provided by the average tide level data is already given to the model, in a more useful form from the low tide, higher high tide, and lower low tide variables.

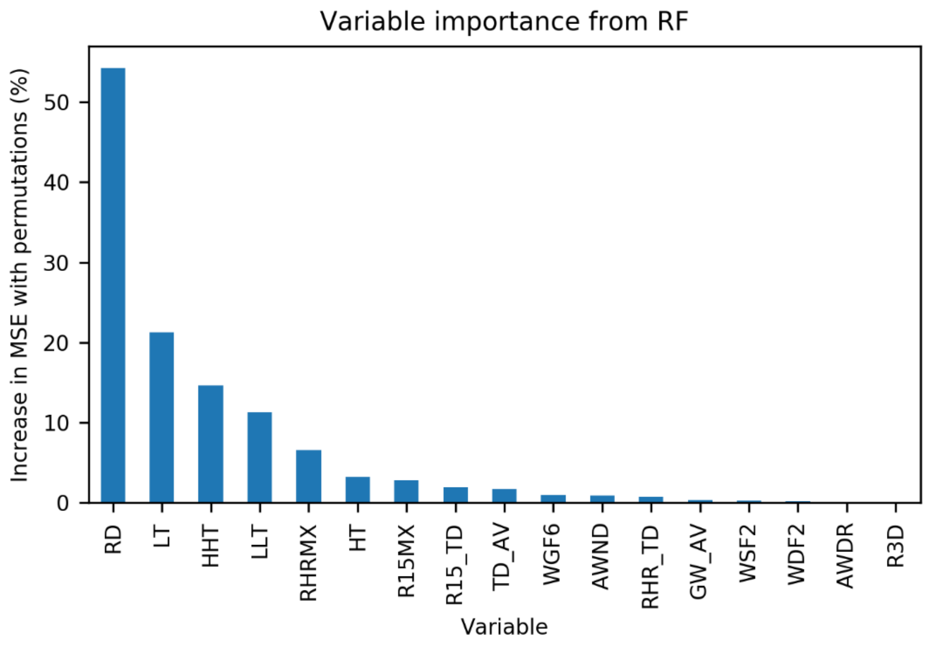


Figure 11 Importance of predictor variables

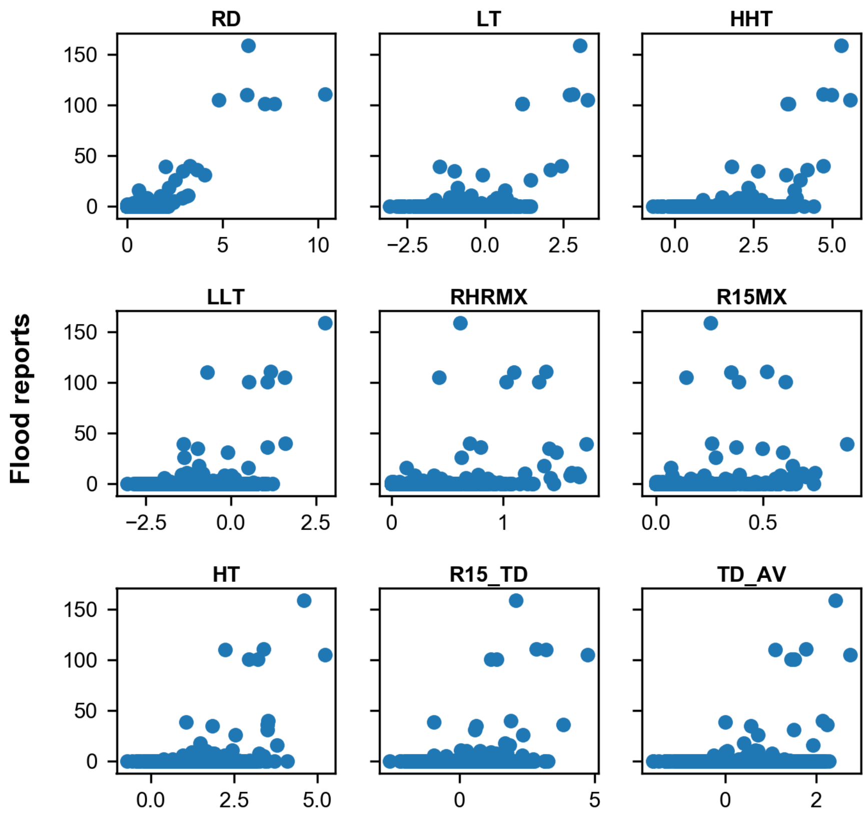


Figure 12 Flood reports plotted against top nine predictor variables

3.3.4 Potential Explanations for Model Shortcomings

A likely reason for the deficiency in the model is the limited amount of output data used to train the model. Although reported floods have increased over the past six years, as seen in Figure 9, they are still relatively rare events. Flood reports were made on only 42, or just over 5%, of the daily records used in model. In addition, the number of flooding reports were distributed very unequally among the 42 days on which flood reports were made. More than 65% of the total flood reports were recorded on just six days (0.6% of the total days modeled). The rarity of days with any flood reports, and especially a large number of flood reports, makes it difficult for the model training. Solomatine and Xue (2004) faced similar problems in training their machine learning model to accurately predict high peak flows which occurred rarely in their dataset.

The results also suggest that, compared to storm events with large volumes of rainfall which caused flooding, other types of storm events were not as well modeled by Random Forest. Figure 13 shows the percent error of the Random Forest evaluation predictions for the 11 events with the top 10 number of reported floods (two events had 101 flood reports: Hurricane Nicole and an unnamed event occurring on 20 September 2016). Two unnamed events, heavy rain which occurred on 16 May 2014 (35 reported flood locations) and thunderstorms which occurred on 10 July 2014 (35 reported flood locations) have average percent error magnitudes larger than the rest. Both of these events had much lower daily rainfall and tide levels, the most important variables of the model (see Figure 11), but higher maximum hourly rainfall and relatively high maximum 15-minute rainfall compared to the other high flooding events. Given this, it is likely that the events caused flash floods, which these results suggest was a type of flooding not well represented in the training dataset. It is expected that a machine learning model would better predict such flood events with a larger, more complete dataset, that contained more instances of such flooding events. Additionally, with more training data, the model could be trained on specific subsets of flood events to tailor the model to a flood event with specific characteristics (e.g. flash floods).

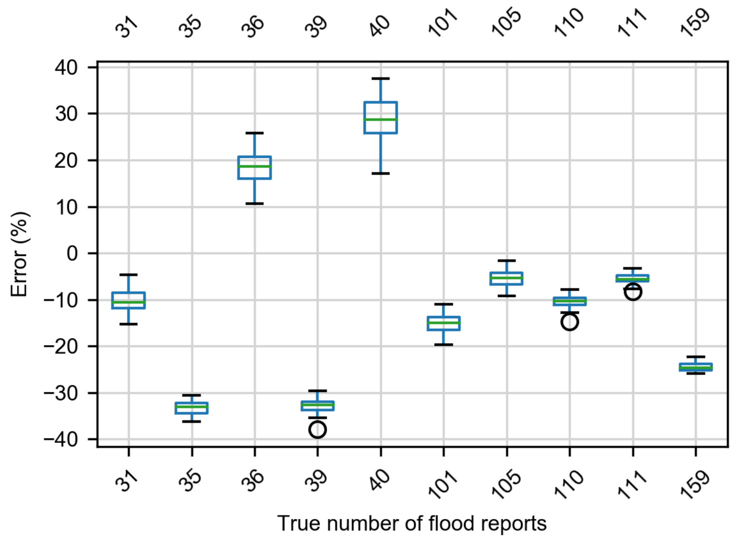


Figure 13 Top 10 flooding event percent error from model evaluation

Besides the limited number of flooding events with which to train the model, the bias present in the output data could have hampered model performance. There is an unknown amount of subjectivity and bias in the data because the flooded locations were reported by individuals. Since the model was trained on data reported by individuals, one individual may influence the trained model disproportionally. Figure 14 shows the total number of flood reports made by each individual reporter and the number of flood events for which each reporter recorded at least one flooded location during the period of record. The highest number of reports made by one reporter was 158, 14% of the total reports from all 71 reporters. Therefore, the model in its training, are significantly influenced by this one reporter and can inherit, to some extent, his/her biases.

One potential bias is in the under- or over-representation of different roadway types in the flooding record. Figure 15 shows the percentage of roadway length per VDOT roadway class in Norfolk and the percent of each roadway class at which flood reports were (Table 8 gives the descriptions for each of the classes). From the figure, it is seen that although public local streets (class 6) account for the majority of the roadway length of the city (close to 60%), only 40% of the flooded streets reported were public local streets. Conversely, principal arterials (class 3) accounted for nearly 30% of the flooded street reports even though the streets make up less than 10% of Norfolk's total roadway length. This suggests that a flooded street less traveled and, therefore, less important to the overall connectivity of the city's street network may have flooded but may not have been reported within the record with the same frequency as the more major roads.

Another example of bias may occur when unequal attention in reporting is given to certain geographic areas of the city or to certain storm events. One example of this bias is seen in the difference in reported floods between Hurricane Hermine and Hurricane Matthew which occurred only one month apart. For Hurricane Hermine, 25 flood reports, more than half of the total of 40 flood reports, came from one area in downtown Norfolk. In contrast, for Hurricane Matthew, which produced more than three times as much daily rainfall on average than Hurricane Hermine (10.4 inches compared to 3.3 inches) and was at least comparable in terms of tide, water table height, and wind conditions, only seven flood reports were made from the same area. It is unlikely that the actual flooding caused by Hurricane Matthew, a much larger storm, was in fact a quarter in severity, but more likely that there were significant differences in individual reporting between the two events.

Given the uncertainties involved in the original model, further introduced when applying it to the Virginia Beach area, and the disconnection of the output to physical characteristics (e.g., elevation) of the Virginia Beach road network, the research team used elevation based hypothetical road flooding scenarios in the transportation chapter. With more spatially and temporally detailed data collected regarding street flooding, it is anticipated that the model shown here could be used with the transportation models in the future.

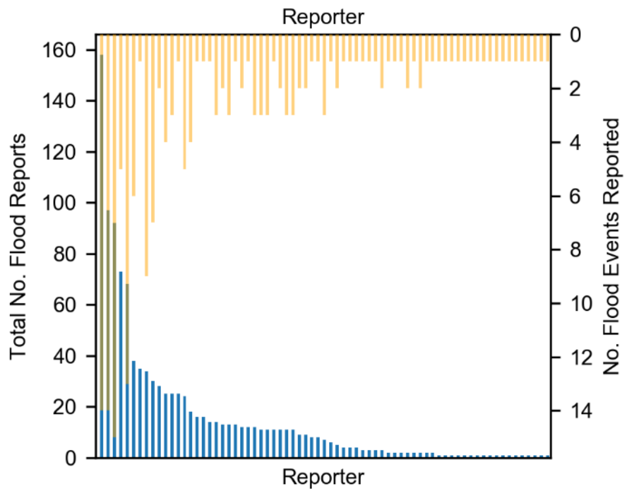


Figure 14 Number of total events made and events reported per reporter

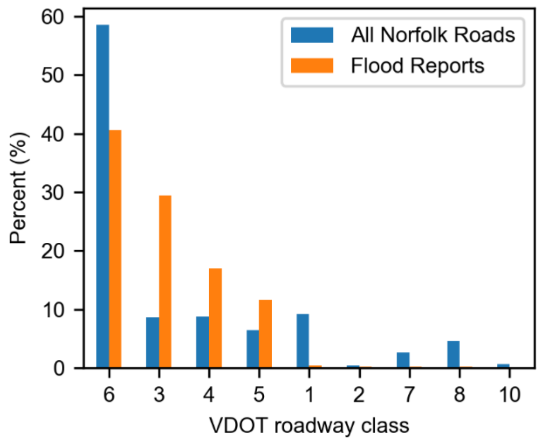


Figure 15 Percentage of total roadway length and percentage of reported floods per VDOT roadway class in Norfolk, VA

Table 8 VDOT Roadway Class Codes and Descriptions

VDOT Road Class Code	Description
1	Interstate
2	Tunnel Roads and other VDOT owned
3	Principal Arterials
4	Minor Arterials
5	Collectors
6	Local Streets- Public
7	Local Streets- Private
8	Miscellaneous
9	Base Roads
10	Public Alleys

4 Transportation

The transportation portion of this study focused on providing route assistance to the vehicles that are affected by nuisance flooding which requires road closures but not neighborhood evacuation. Depending on their origin, and intended destination and path, some vehicles are affected while others can proceed with their trip as planned. The affected vehicles have an origin, destination and/or intermediate node(s) on soon-to-be flooded or closed links. In these cases, in-vehicle route guidance systems can assist the users in leaving the danger zone and selecting the best alternative path. In addition, in-vehicle route guidance systems could communicate warning messages to unaffected vehicles alerting them about the areas to avoid.

The centralized dynamic route guidance framework relies on the connected vehicle environment and assumes that real-time information and accurate traffic measurements are available and can be used to determine the actual link travel times, flooding progress, and delays. These link performance measures are assumed to be broadcasted from the traffic management center to the vehicles in which in-vehicle guidance systems are integrated. To compute a set of optimal alternative paths for the vehicles that need to be rerouted, the time-dependent hyperstar routing algorithm (Bell, Trozzi et al. 2012) is adopted. The vehicles on soon-to-be flooded links are first directed out of the danger zone and then provided with a set of alternative paths to resume their trip to their original destination. The vehicles that were heading towards a destination on a soon-to-be flooded or closed link are assigned to a new safe destination. A new set of alternative optimal paths are generated for vehicles with one or more soon-to-be flooded and/or closed link(s) in the path. Since the hyperstar routing algorithm computes a set of optimal paths between an origin/destination pair, the selection of the actual path depends entirely on the user's preference. On the other hand, unaffected vehicles present in the study area, receive a warning that includes the locations of all the roads and intersections to be avoided. Due to the connected vehicles' capabilities, the traffic measurements are assumed to be continuously collected, allowing the regular update of link performance measures. Furthermore, communication with a traffic management center (TMC) is assumed to exist to close roads and prevent entry to the soon-to-be flooded links only for safety as needed.

4.1 Routing Algorithm Background

Dijkstra's (1959) algorithm computes a shortest path tree between an origin node and all destination nodes in a directed graph with non-negative weighted edges. A wide variety of techniques focusing on speeding-up Dijkstra's algorithm have emerged. They mainly consisted of preprocessing data in the graph and then using the information to reduce the computational time while outputting the shortest path tree. Based on their review, Wagner and Willhalm (2007) asserted that these speed-up techniques consisted of limiting the search space of the algorithm and they described two classical speed up techniques: the bidirectional search and the goal-directed search or Astar. The bidirectional search (Luby and Ragde 1989) was based on alternating between two unidirectional searches: one forward search from the origin node in the graph and one reverse search from the destination node in the reverse graph. When one node was transferred to the closed lists of both searches, the algorithm terminated and the shortest path between the origin and destination nodes was computed as the sum of the distance between the origin and the common node obtained from the normal search and the distance between the common node and the destination node obtained from the reverse search. On the other hand, the goal-directed search of Astar, which was derived from the heuristic approach developed by Hart et al. (1968), consisted of adjusting the way the nodes were labeled thus affecting the order in which the nodes in the open list were expanded. In the unmodified Dijkstra's algorithm, the node label reflected the distance from the origin to the node along the tree. However, the Astar speed-up technique added to each node a potential or heuristic that depended on the desired destination node. The adequacy and accuracy of node potentials efficiently guide the search towards the destination node. The unmodified Dijkstra's algorithm returns the shortest path from one node to all other nodes in a graph, but, it can also be terminated when the desired destination is reached. The bidirectional search and the goal-directed search techniques focused on speeding-up the search for the shortest path between an origin and one destination node by efficiently reducing the search space thus guiding the algorithm's search towards the destination node.

Bell (2009) developed a routing algorithm called Hyperstar that generated a reliable path set, rather than a single shortest path. It was a reinterpretation of the Spiess and Florian (1989) algorithm which was designed for transit assignment. The latter determines, for each link, a service frequency that was equal to the inverse of the waiting time on the link. Next, it finds a hyperpath between an origin and a destination (set of possible optimal paths) by minimizing the expected travel time to the destination. Bell adapted the Spiess and Florian algorithm to vehicle routing by considering the service frequency of a link equal to the inverse of the maximum delay that can be experienced on a link. For instance, a link with a high maximum delay has a low service frequency and is hence less reliable. Since the hyperstar algorithm was designed for vehicle routing systems, it incorporated the Astar algorithm to speed up the path generation. Ma et al. (2013) introduced the Dijkstra-Hyperstar algorithm which incorporated two techniques to speed-up the Hyperstar algorithm while maintaining the same generated hyperpath. It used Dijkstra's algorithm to determine the node potentials and adopted a node directed search to limit the number of links being evaluated.

Another method that considered the dynamic aspect of networks was introduced by Chabini and Lan (2002) who assigned a time-dependent cost to each link and computed a single path is computed between a pair of nodes. As an extension to the Chabini and Lan (2002) work, Bell et al. (2012) revisited the hyperstar algorithm discussed above and added a dynamic and more realistic aspect. An undelayed travel time along with a maximum delay were assigned to each link and were both dependent on the time of arrival at the link. The algorithm reversed the original hyperstar algorithm in which the travel time of a link depended on the time of departure from a link. The time-dependent hyperstar algorithm builds the hyperpath from the origin to the destination by minimizing the expected arrival at the destination and all intermediate nodes.

In this study, we implemented the time-dependent version of the hyperstar algorithm (Bell, Trozzi et al. 2012) to provide a set of alternative paths to the vehicles affected by the flooding; thus, enhancing their route selection while maintaining their preferences. During a flood event, the state of the system (i.e. road closures) as well as the link performance measures vary with time. For instance, a vehicle can travel a link that will become closed and unavailable for other vehicles in the future. So the time-dependent hyperstar algorithm was used because it can adapt to the dynamic aspects of the flood event and the transportation system itself by assigning each link a time-dependent travel time and maximum delay.

4.2 Framework

This framework relies on the connected vehicles environment in which communications among equipped vehicles and between equipped vehicles and infrastructure are enabled. Real-time traffic information from these vehicles are assumed to be collected and processed to provide reliable performance measures and for traffic optimization purposes.

This framework leverages the benefits of the connected vehicle environment by first collecting the information about the vehicles (origin, destination and path) detected in the system and about the links and intersections in the network (travel time, delay, open/closed status). Since a set of roads in the transportation network will be flooded and subsequently closed, vehicles which were located on one of these links and vehicles which intended to travel through one will require assistance in determining their new route to reach their destination safely. This study focuses on routing the affected vehicles just prior to and during a flood. The connected vehicles also allow the broadcast of warning messages to the unaffected vehicles.

The framework acts as a centralized dynamic route guidance system during flood emergency situations. A Traffic Management Center (TMC) is assumed to gather all traffic information from the connected vehicles. The TMC estimates and regularly updates the current link performance measures and the flood impacts at the street-level. In-vehicle systems generate the routing guidance after the receipt of all required information from the TMC.

The framework does not require the generation of optimal emergency evacuation strategies prior the emergency event. The objective here is to develop a framework that adapts to flood scenarios with different flood timelines and locations.

The vehicle routing algorithm used in this framework is the time-dependent hyperstar algorithm introduced by Bell et al. (2012), which computes a set of optimal paths, called a hyperpath, between an origin and a destination. For faster hyperpath computation, the algorithm requires the generation of node potentials to direct the search towards the destination. In this study, a node potential refers to the remaining undelayed travel time from the node to the destination and is calculated using the Partitioning shortest path algorithm (Glover, Klingman et al. 1985). The hyperstar algorithm is a theoretical routing algorithm that has been previously applied on a sample network for demonstration purposes (Bell, Trozzi et al. 2012). In this study, the hyperstar is tested on more realistic transportation networks that account for the wait time experienced at signalized intersections.

The following assumptions were made when developing the framework:

- If a vehicle can travel, it will travel.
- If a link is flooded, its start and end nodes can still be travelled.
- A link is either open or closed (no partial closure).
- Only affected vehicles are rerouted.
- Unaffected vehicles receive a warning message that includes the areas to avoid.
- Vehicles with origins on closed links receive a warning message to wait at the origin.
- A traffic management center is available to process the real-time information obtained from the connected vehicles and to estimate the current link travel times and delays as well as road closures due to the flood.

- In-vehicle systems generate the route guidance (set of alternative paths) based on the link performance measures and flood information broadcasted from the traffic management center.
- Vehicles with destinations on soon-to-be flooded or closed links are assigned to new safe destinations where they can wait.
- Vehicles that cannot travel to any safe destination can wait at their origins.

4.2.1 Terminology

The first set of terms categorizes nodes near the flooding hazard and is illustrated in Figure *16*.

- The *flood boundary* includes the downstream nodes of the soon-to-be flooded links and closed links. These nodes can be connected to a safe node in Buffer 1 by a safe link.
- *Buffer 1* includes the safe (unflooded) nodes that are connected to one node in the flood boundary by a safe link.
- *Buffer 2* includes the safe (unflooded) nodes that are connected to one node in Buffer 1 by a safe link and are farther away from the hazard area than nodes in Buffer 1.

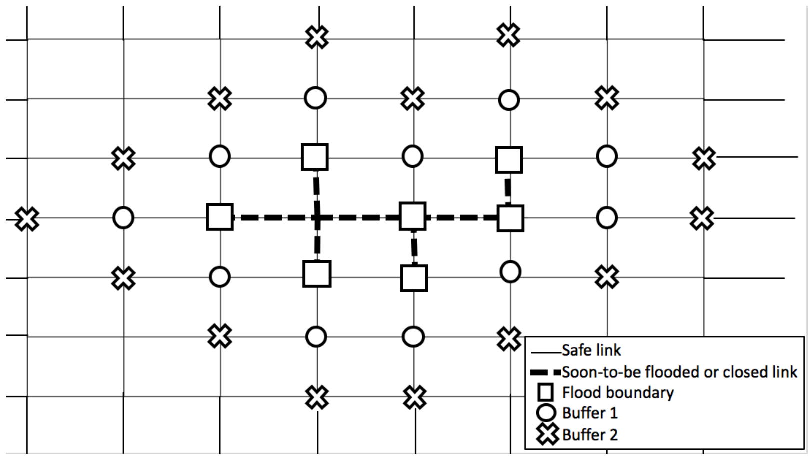


Figure 16 Flood boundary and buffers

The next set of terms pertains to the timeline of events for a link that experiences flooding. These are illustrated in Figure *17*. Not all links in the network experience flooding; some links are considered safe for the entire time. Only links that flood require the below designations.

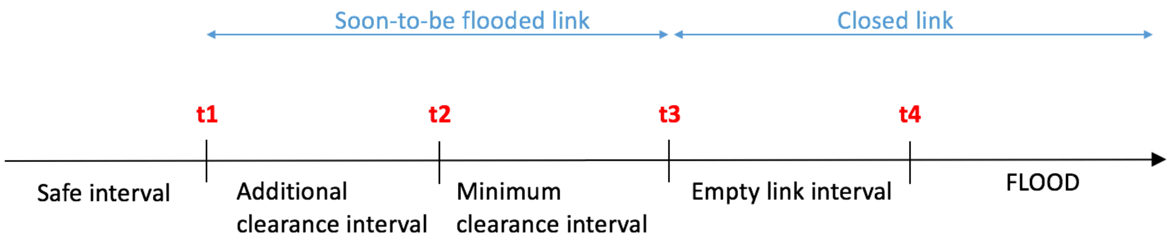


Figure 17 Link timeline

- Flood interval: This is the time interval during which the link cannot be travelled and is closed due to water on the road.
- Empty link interval: This interval acts as a safety margin during which the link is closed. No vehicles should be on the link during this time. This variable would be determined by policy makers or a transportation agency and can vary with the link characteristics. However, in this study, it is considered fixed for all links.
- Minimum clearance (and additional clearance) interval: This is the time interval during which the link is considered as soon-to-be flooded and only vehicles that are in the soon-to-be flooded zone can travel (enter and/or exit) these links. Once these vehicles reach a safe zone (i.e., exit the soon to be flooded zone), they are denied re-entry. The link minimum clearance interval (from time "t2" to time "t3") depends on the link characteristics, the traffic and weather conditions. The additional clearance interval (from time "t1" to time "t2") is an interval whose duration is determined by policy makers or a transportation agency based on how safe and sufficient the minimum clearance interval is.
- Safe interval: This is a time interval during which the link is not threatened by flood water, is considered safe, and can be travelled by any vehicle.
- Closed link: A link cannot be traversed when it is closed. A link that experiences flooding is closed at time t3 (Figure *17*) and thereafter.
- Soon-to-be flooded link: A link that will be flooded in the future is considered soon-to-be flooded before it actually floods. In Figure *17*, the soon-to-be-flooded designation is assigned to the relevant links between time t1 (>= t1) and t3 (<t3).
- Link time of closing: The time of closing is the time after which the entry to the link is prevented and it varies from an individual vehicle's perspective depending on the vehicle's initial position. For instance, vehicles with origins on soon-to-be flooded links are allowed to travel along soon-to-be (but not yet) flooded links to reach a safe stop, while vehicles originally on safe links are prohibited from entering soon-to-be flooded links. Thus, for vehicles outside the soon-to-be flooded and/or closed area, the time of closing of a soon-to-be flooded link is at time t1 in Figure *17*, while time of closing of the same link is at time t3 for vehicles that are located in the soon-to-be flooded area.

To sum up, at time t1, entry to the soon-to-be flooded zone is prohibited and only vehicles originally in this area are allowed to travel within it in order to exit and reach a safe stop. Thus, at time t1, barriers that block the entry points to the flooded zones should be placed. At time t3, no vehicle is allowed to travel on the link. Therefore, barriers that close the entry and exit points of all flooded links should be positioned.

Note that a link is considered soon-to-be flooded during the minimum clearance and additional clearance interval. Vehicles initially located outside the flooded zone are not allowed to travel along soon-to-be flooded and closed links, and the time of closing of these links is considered at time t1. Vehicles that are originally inside the flooded zone can travel along soon-to-be flooded links, and the links' time of closing, for these vehicles, is at time t3.

Also referring to Figure 17, an *affected vehicle* is a vehicle with an initial or regular path that is directly impacted by the flood or safety measures. The *affected vehicle* meets at least one of the following criteria:

- Has an **origin** on a soon-to-be flooded link (t1 <= departure time < t3) or on a closed link (departure time >= t3)
- Is heading to a **destination** on soon-to-be flooded or closed link (expected time at destination >= t1)
- Intends to travel (based on the initial or regular path) through one or more **intermediate** soon-to-be flooded or closed link (expected time of arrival at the link >= t1).

The hyperstar algorithm as originally developed, uses the terms *undelayed travel time* and *maximum delay (Bell 2009, Bell, Trozzi et al. 2012)*. In the context of this study, these terms are considered as follows:

- Undelayed travel time: This is the link travel time that varies with the regular time of traffic conditions. Peak travel times are higher than off-peak but still considered undelayed in the absence of adverse weather or incidents.
- Maximum delay: This is the maximum delay incurred on a link due to the flood and is set to a very large number (infinite) to represent the closed condition (i.e. it is infinite after the link time of closing).

4.2.2 Input data for the framework

The input data required for the framework includes:

- Undelayed link travel times: This information is required for all links in the network and is assumed to be available from a traffic management center either from current technology (e.g., sensors/detectors) or from connected vehicles.
- Maximum delay: This information is required for all links in the network. This link performance measure can be obtained from sensors, detectors or using the connected vehicles as long as the link can still be travelled. The maximum delay is infinite when entry to the link is prohibited.
- Flood input data: For research purposes, a list of links that will flood with their expected time of flooding is needed. In future deployment, similar information could be obtained from weather and flood forecast systems, as suggested in Sections 2 and 3.

In this research, outputs from VISSIM simulations along with generated flooding scenarios constitute the input data required for the numerical implementation of the framework.

4.2.2.1 Undelayed travel time

As mentioned above, the undelayed travel times could be collected from sensors. However, for the sample application in this study, these times were estimated based on simulations. Since the method used to generate the undelayed times were specific to this study but not the framework, it is discussed after the framework.

4.2.2.2 Maximum delay

For research purposes, the maximum delay experienced on a link while travel was still permitted was estimated based on information from the Federal Highway Administration. According to the Federal Highway Administration (2017), a heavy rain event leads to an estimated free-flow speed (FFS) reduction between 6% and 17%. This was reflected by an increase in the free-flow time (FFT) between 6.5% and 20.5% for the unflooded links in this study. Thus, if the vehicle was expected to enter the link before the link closing time, the maximum delay experienced on the link was a duration between 6.5% and 20.5% of the FFT. However, if the time of entry was after the link closing time, the maximum delay was infinite and as a result the entry to the link was indirectly prevented.

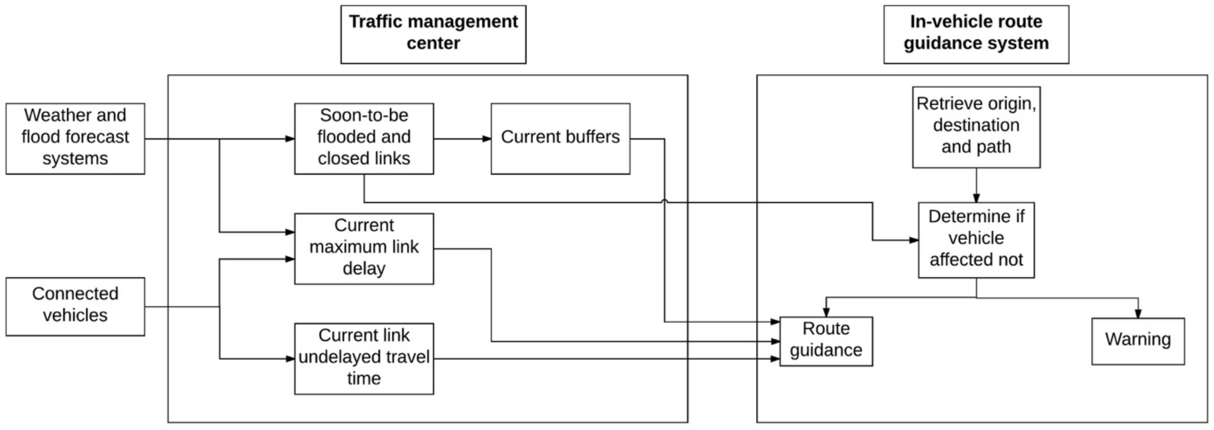
4.2.2.3 Flood data

Information about the flood in terms of location and timeline in the transportation network are crucial inputs to this framework. We assumed that these data could be obtained from future weather and flood forecast systems. For this study, hypothetical flood information was generated. The information included the links and intersections that flood along with the time of flooding.

4.2.3 Framework description

This framework was developed to be implemented in a connected vehicles environment. As shown in the flowchart in Figure 18, once a vehicle is detected in the transportation network, its trip is evaluated based on the network's state to determine whether the vehicle is affected by the flood or not. The remaining trip of an en-route vehicle is evaluated and its origin is considered its position at the time the framework is activated. However, the complete trips of new vehicles in the system are evaluated. If the vehicle is affected but is still able to leave its origin, it receives routing assistance to reach its final destination safely and as soon as possible. If the vehicle is affected but cannot reach a safe destination, it receives a warning stating that it should wait at the origin. If the vehicle is unaffected, a warning notifies the vehicle about the areas to avoid. The actual link data are assumed to be estimated, stored and regularly updated in the TMC's database through the broadcast of real-time information from the connected vehicles as well as updates from weather and hydrologic models.

Vehicles in the system are evaluated independently. To determine if the vehicle is affected or not, along with the corresponding route guidance or warning to be displayed, the in-vehicle route guidance system retrieves the current link data from the traffic management center.



Traffic management center
In-vehicle route guidance system
Weather and flood forecast systems
Soon-to-be flooded and closed links
Current buffers
Retrieve origin, destination and path
Current maximum link delay
Determine if vehicle affected not
Connected vehicles
Route guidance
Warning
Current link undelayed travel time

Figure 18 Framework flowchart (future connected vehicle environment)

In this research, and to demonstrate the implementation of this framework with simulated vehicle data, another flowchart has been adopted and is illustrated in Figure 19. Instead of assuming that the link data (i.e. travel time and information about flooding) are provided from the TMC, link performance measures were obtained from microscopic simulations of a transportation network modeled in VISSIM and information about the flood was generated for hypothetical scenarios. All vehicles recorded during a one-hour simulation in VISSIM are evaluated all at once based on their complete trips. Then, the vehicles are considered successively and depending whether they are affected or not, a vehicle either receives a warning or route guidance.

In the remainder of this chapter, the four major parts of the framework are discussed. First is the “buffer generation” part which explains how the safe node buffers around the flooded links are defined. These node buffers act as first safe stops or final destinations to some groups of vehicles. Second is the “vehicle analysis” in which the vehicles detected in the system are evaluated and grouped. The third part introduces the “routing” procedure which consists of executing the hyperstar routing algorithm and the partitioning shortest path algorithm whenever route guidance between two nodes in the network is needed. In the fourth part called “process”, the tasks specific to each group of vehicles are discussed.

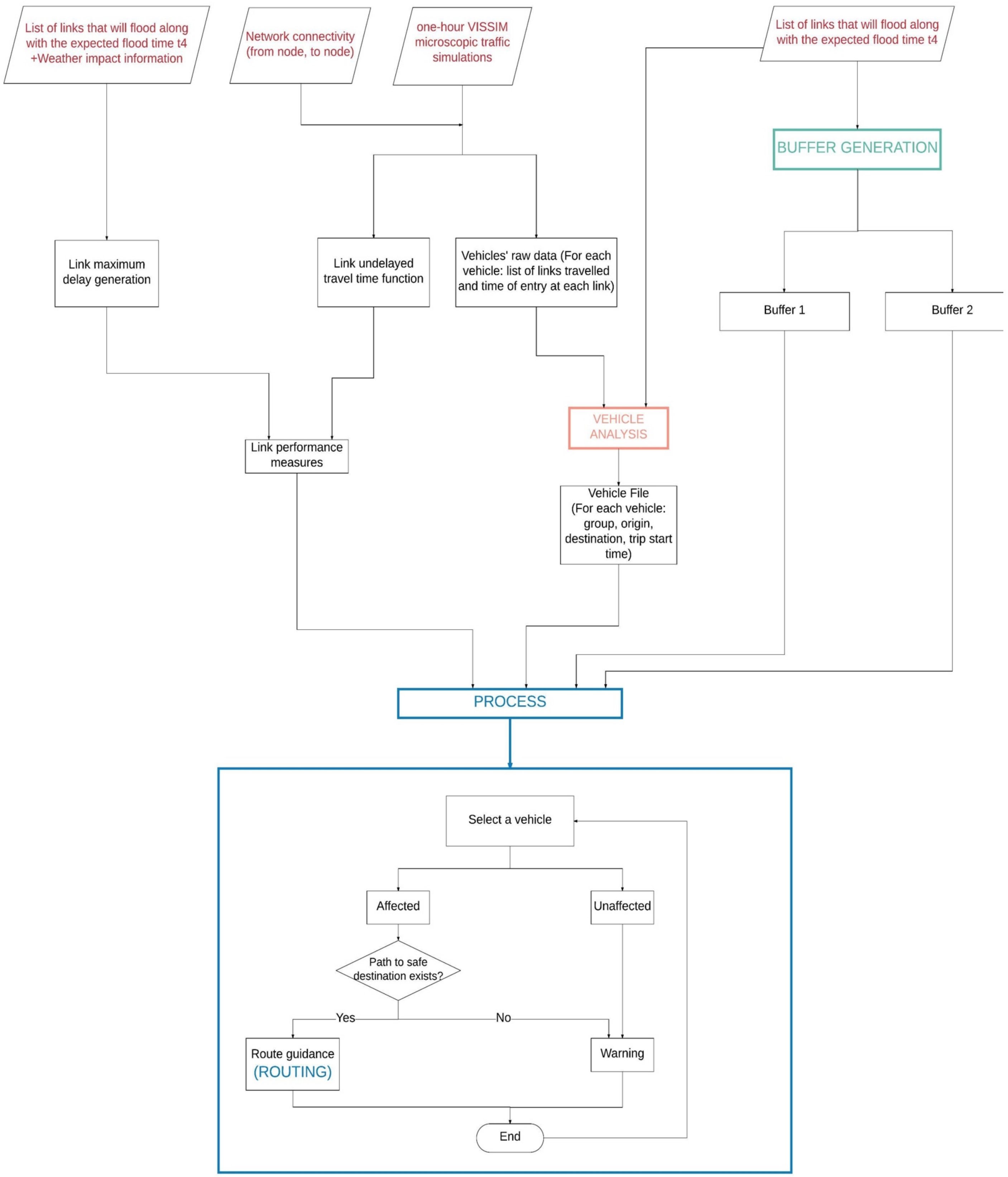

List of links that will flood along with the expected flood time t4 +Weather impact information
Network connectivity (from node, to node)
one-hour VISSIM microscopic traffic simulations
List of links that will flood along with the expected flood time t4
BUFFER GENERATION
Link maximum delay generation
Link undelayed travel time function
Vehicles' raw data (For each vehicle: list of links travelled and time of entry at each link)
Buffer 1
Buffer 2
VEHICLE ANALYSIS
Link performance measures
Vehicle File (For each vehicle: group, origin, destination, trip start time)
PROCESS
Select a vehicle
Affected
Unaffected
Path to safe destination exists?
Yes
No
Route guidance (ROUTING)
Warning
End

Figure 19 Framework flowchart (research)

4.2.3.1 Buffer generation
As shown in Figure 20, this step requires as inputs the network connectivity (link ID, upstream node, and downstream node) and the lists of links that are soon-to-be flooded or closed. In future connected vehicle environments, the node buffers would be updated every time the state of the system changes. In other words, the input list of the links is regularly updated so that the current node buffers are defined.

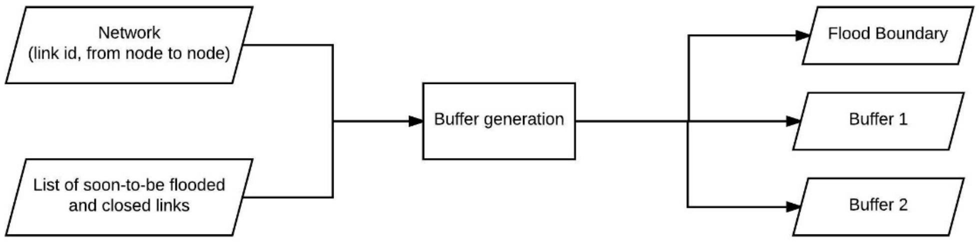

Network
(link id, from node to node)
List of soon-to-be flooded and closed links
Buffer generation
Flood Boundary
Buffer 1
Buffer 2

Figure 20 Input/ Output data of buffer generation step

This step was coded using the Java computer programming language and the corresponding pseudocode is shown in Appendix A. Note that in the future connected vehicle environment, the node buffers can be identified by computing the buffers at specific distances from the flooded locations.

4.2.3.2 Vehicle analysis
Once the links that are soon-to-be flooded or closed are defined, all vehicles in the area of interest are evaluated to identify the affected and unaffected vehicles. Note that the buffer generation and the vehicle analysis steps are independent and can be executed in parallel.

As defined above, an affected vehicle is a vehicle with an origin, destination or intermediate link that is soon-to-be flooded or closed within the duration of the trip. A vehicle that does not satisfy any of these conditions is not considered affected by the flood. In other words, its origin, destination and path are not expected to be soon-to-be flooded or closed during the trip.

Regarding the affected vehicles, if they can still leave their origins (origin not on a closed link at the departure time) and if at least one path can be found between their position and a safe destination, they receive route assistance that comprises a set of alternative paths from which the user selects his/her preferred one. On the other hand, affected vehicles that cannot travel only receive a warning instructing them to wait at the origin. All unaffected vehicles only receive a warning that discloses the areas to avoid in case of deviation from the planned path.

The vehicle data obtained from the one-hour microscopic traffic simulation with VISSIM, includes, for each vehicle, the list of links in the path and the time of entry to each link. In this step, we extract from the raw data obtained from the VISSIM simulation, the origin, destination and trip start time for each vehicle. In addition, we classify the vehicles into six groups based on the flooded links and corresponding closing time, as shown in Table 9.

Table 9 Vehicle Groups

Group	Priority	Description
1	2	Vehicle with an **origin** on a link that is soon-to-be flooded at departure time and a **destination** on a link that is soon-to-be flooded or closed at the expected arrival time
2	4	Vehicle with an **origin** on a link that is soon-to-be flooded at departure time
3	1	Vehicle with an **origin** on a link that is closed at departure time or vehicle that cannot reach any safe destination
4	3	Vehicle with a **destination** on a link that is soon-to-be flooded or closed at the expected arrival time
5	5	Vehicle with an intermediate link that is soon-to-be flooded or closed at the expected time of entry
6	6	Unaffected vehicles

In the future connected vehicle environment, all vehicles in the system at the time the framework is initiated, would be analyzed and grouped simultaneously. Then, once a new vehicle is detected in the area, it would be analyzed and grouped independently.

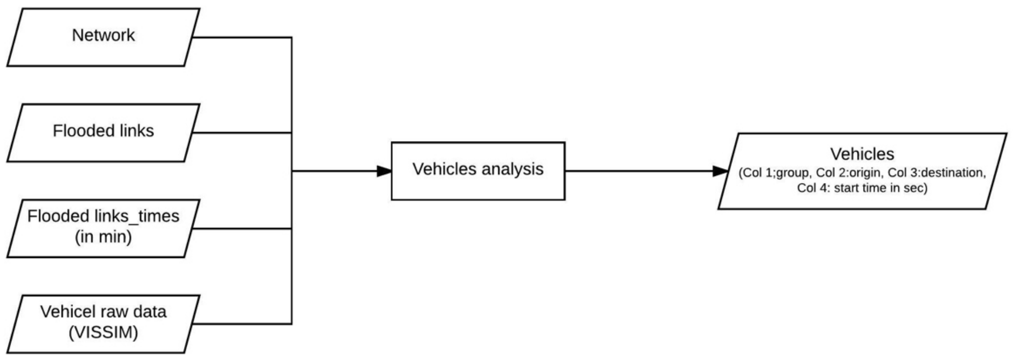

Network
Flooded links
Flooded links_times (in min)
Vehicel raw data (VISSIM)
Vehicles analysis
Vehicles
(Col 1;group, Col 2:origin, Col 3:destination, Col 4: start time in sec)

Figure 21 Input/ Output data of vehicles analysis step

Java code has been developed to complete this step and the pseudocode is shown in Appendix A. As shown in Table 9, each group is given a specific priority from 1 to 6, with 1 being the highest. A vehicle can belong to more than one group but the one with the highest priority dominates. For example, if a vehicle belongs to groups 5 and 3, the latter dominates because the vehicle cannot leave its origin and should only receive a warning to wait at the origin. Similarly, if a vehicle satisfies the conditions of groups 2 and 4, it is assigned to group 1.

This hierarchy is ensured through the order of the if-statements in the grouping method shown in the vehicle analysis pseudocode. Each if-statement evaluates the conditions of a given group. The order adopted goes from the lowest to the highest group priority. Hence, all vehicles are first assigned to group 6 which is the group with the lowest priority. Then, if the vehicle satisfies the conditions of the next if-statement's group (i.e. group with higher priority), the group will be over written to obtain at the end the final group of each vehicle.

4.2.3.3 Routing

In future connected vehicle environment, the link performance measures would be updated regularly through the connected vehicles and TMCs. Once the routing algorithm is triggered, the most recent link data (from TMCs) would be used in the hyperstar algorithm (Bell, Trozzi et al. 2012). In this research, we assigned, for each link, a time threshold after which the link is considered closed and the delay is infinite to reflect the flood.

Depending on the vehicle's group, the vehicle either receives a warning or route guidance. When the latter is needed, the hyperstar algorithm is initiated and requires, as input, a set of node potentials that is specific to the destination. In our study, a potential at a node refers to the remaining undelayed travel time from the node to the destination.

To obtain all node potentials to a specific node in the network, the partitioning shortest path algorithm (Glover, Klingman et al. 1985) is used. This algorithm computes the shortest path from one node to all other nodes in the network. Thus, computing all node potentials to a given destination consists of first, reversing the network (i.e., reversing the direction of each link while keeping the same link costs), and then implementing the algorithm from the destination to all nodes.

For translation to the field, we assume that the node potentials to each node in the network are already computed and stored in the traffic management center's database. As a result, whenever the hyperstar from an origin to a destination is to be implemented, the node potentials that correspond to the destination are retrieved and used to ensure faster computations of the hyperpath. In this research, once the hyperpath between an origin and a destination has to be identified, the partitioning algorithm is initiated and the node potentials to the given destinations are computed before calculating hyperpaths.

4.2.3.4 Process

In this final step, vehicles are evaluated successively. Once selected, and depending on the vehicle's group, the corresponding tasks are executed as follows:

- Group 1 (Figure 22): These vehicles have origins on links that are soon-to-be flooded at their departure times and destinations on links that are soon-to-be flooded or closed at the vehicles' expected arrival times. The objective is to assist them first in reaching safe stops as soon as possible, and then in resuming their trips toward final destinations. These vehicles cannot reach their original destinations because their expected times of arrival fall after time "t1" (refer to Figure 17) at which the link is soon-to-be flooded or closed. Thus, a new safe destination is proposed to the driver. It is a node in buffer 2 that is the closest to the original destination in terms of travel time. In this study, we assume that all users accept the proposed new destination in buffer 2. Once the new destination is defined, its closest node from buffer 1 is then identified and acts as the first safe stop. The vehicle is routed from its origin to the closest node in buffer 1 first, to ensure it exits the flooded zone as soon as possible while also minimizing its remaining distance to the final destination. During this part of the trip, the vehicle is allowed to travel on soon-to-be flooded links but entry to closed links is prohibited. Once the vehicle reaches the intermediate safe stop at buffer 1, the vehicle is provided with a new set of alternative paths to reach its final safe destination while preventing re-entry to the soon-to-be flooded and closed links.

When no hyperpath is found between the vehicle's position and the assigned first stop, the next closest node in buffer 1 is selected. If no path exists between the vehicle's origin and any node in buffer 1, the vehicle has to wait at the origin and is, hence, moved to group 3. When no hyperpath is found between the node in buffer 1 and the final destination, we assume that the user can wait at buffer 1.

- Group 2 (Figure 23): These vehicles have an origin on a link that is soon-to-be flooded at departure time. The goal is to assist them in reaching a safe stop first before resuming the trip toward the final destination. The closest node from buffer 1 to the destination is identified and acts as the first safe stop. A set of alternative paths is computed between the origin and the node in buffer 1 (this hyperpath may include soon-to-be flooded links but not closed links), then a new set of paths is identified from the node in buffer 1 to the final destination (this hyperpath cannot include soon-to-be flooded or closed links).

  When no hyperpath is found between the vehicle's initial position and the closest node in buffer 1, the next closest node to the destination is evaluated. If no hyperpath is found between the position and any node in buffer 1, the vehicle has to wait at the origin and is listed in group 3. When no hyperpath is found between the node in buffer 1 and the destination, the user waits at buffer 1.

- Group 3: These vehicles have an origin on a link that is closed at departure time. Thus, they receive a warning message instructing them to wait at the origin. This group includes as well the vehicles from groups 1, 2, 4 and 5 that cannot reach any safe destination.

- Group 4 (Figure 24): These vehicles have a destination on a link that is soon-to-be flooded or closed at the vehicle's expected time of arrival. A new destination at the closest node in buffer 2 to the original destination is assigned. The user subsequently receives a set of alternative paths between its origin and the new safe destination (this hyperpath cannot include soon-to-be flooded or closed links).

  When no hyperpath exists between the origin and the selected node in buffer 2, the next closest node to the destination is evaluated. If no hyperpath can be identified between the origin and any of the nodes in buffer 2, the vehicle has to wait at the origin and is listed in group 3.

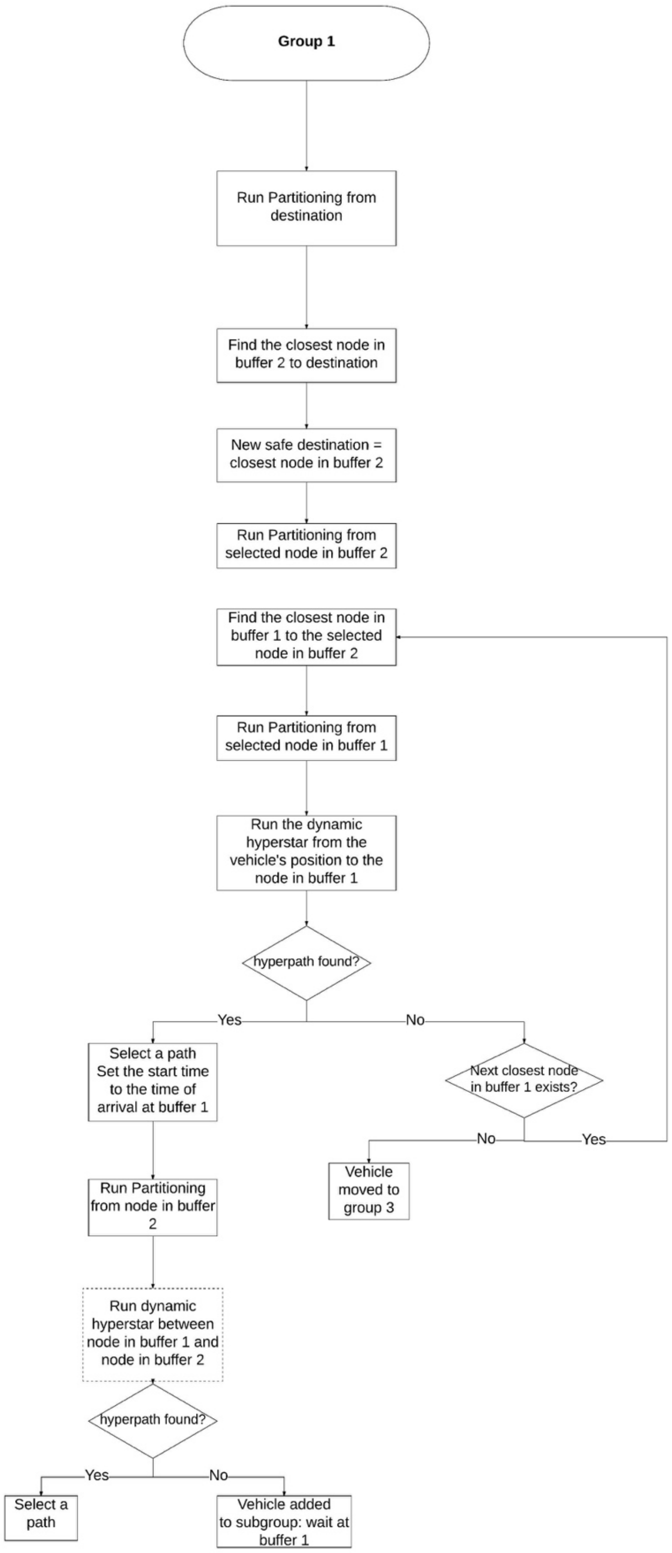

Group 1
Run Partitioning from destination
Find the closest node in buffer 2 to destination
New safe destination = closest node in buffer 2
Run Partitioning from selected node in buffer 2
Find the closest node in buffer 1 to the selected node in buffer 2
Run Partitioning from selected node in buffer 1
Run the dynamic hyperstar from the vehicle's position to the node in buffer 1
hyperpath found?
Yes
No
Select a path Set the start time to the time of arrival at buffer 1
Next closest node in buffer 1 exists?
No
Yes
Vehicle moved to group 3
Run Partitioning from node in buffer 2
Run dynamic hyperstar between node in buffer 1 and node in buffer 2
hyperpath found?
Yes
No
Select a path
Vehicle added to subgroup: wait at buffer 1

Figure 22 Group 1 Processing

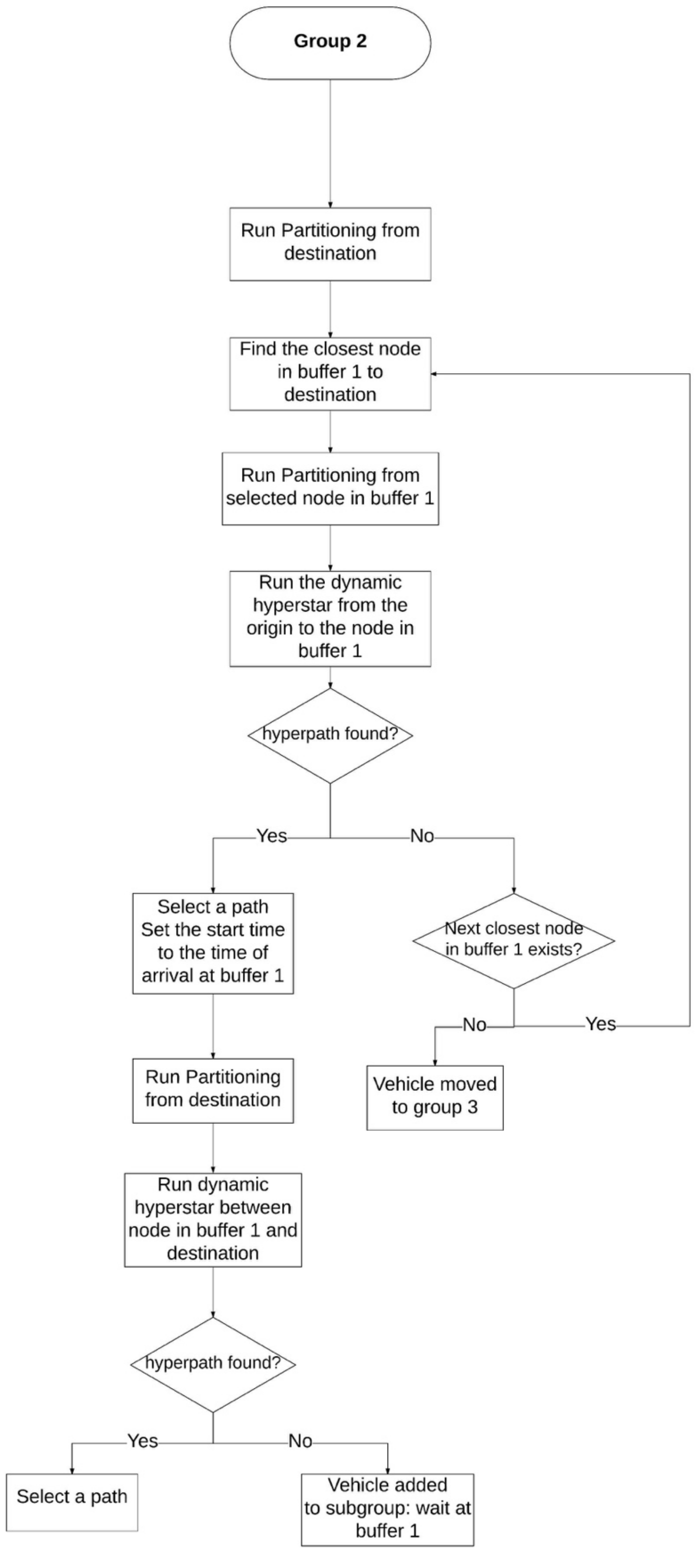

Group 2
Run Partitioning from destination
Find the closest node in buffer 1 to destination
Run Partitioning from selected node in buffer 1
Run the dynamic hyperstar from the origin to the node in buffer 1
hyperpath found?
Yes
No
Select a path Set the start time to the time of arrival at buffer 1
Next closest node in buffer 1 exists?
No
Yes
Vehicle moved to group 3
Run Partitioning from destination
Run dynamic hyperstar between node in buffer 1 and destination
hyperpath found?
Yes
No
Select a path
Vehicle added to subgroup: wait at buffer 1

Figure 23 Group 2 Processing

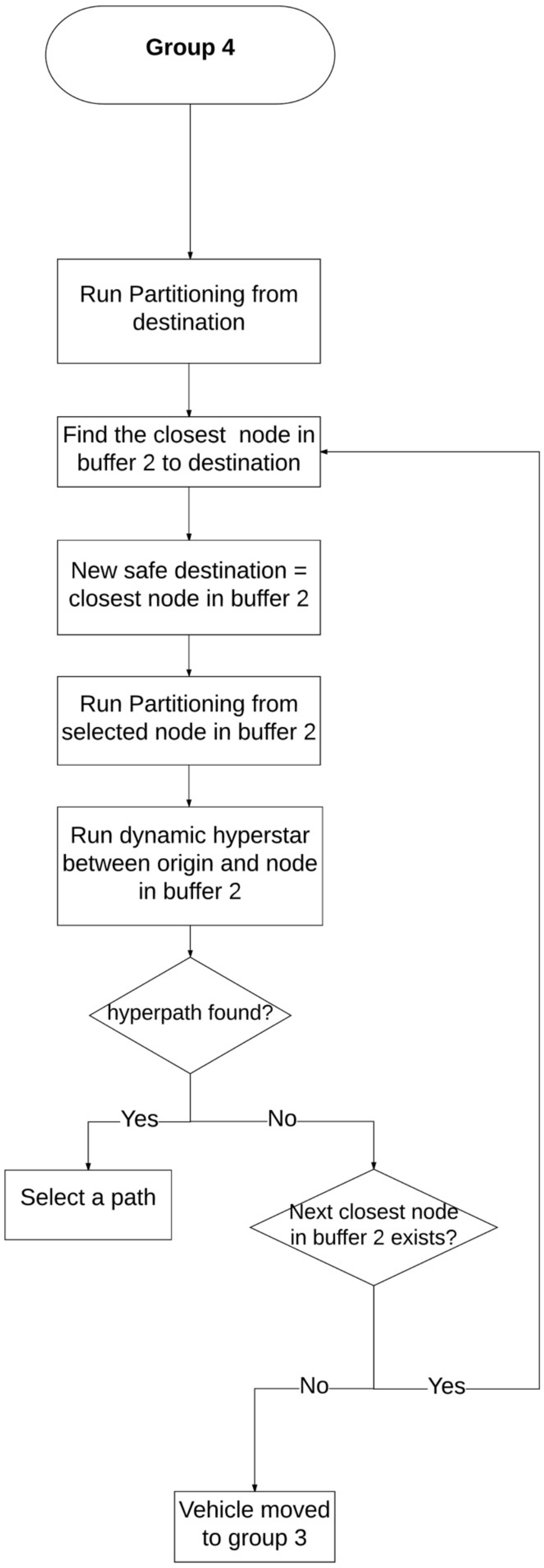

Group 4
Run Partitioning from destination
Find the closest node in buffer 2 to destination
New safe destination = closest node in buffer 2
Run Partitioning from selected node in buffer 2
Run dynamic hyperstar between origin and node in buffer 2
hyperpath found?
Yes
No
Select a path
Next closest node in buffer 2 exists?
No
Yes
Vehicle moved to group 3

Figure 24 Group 4 Processing

- Group 5 (Figure 25): These vehicles have one or more intermediate links that are soon-to-be flooded or closed at the expected time of entry. As a result, the driver has to modify his/her planned path. The driver is provided with a set of alternative paths between the original origin and destination (this hyperpath cannot include soon-to-be flooded or closed links). If no hyperpath exists between the vehicle's position and the destination, the vehicle waits at the origin and is listed in group 3.

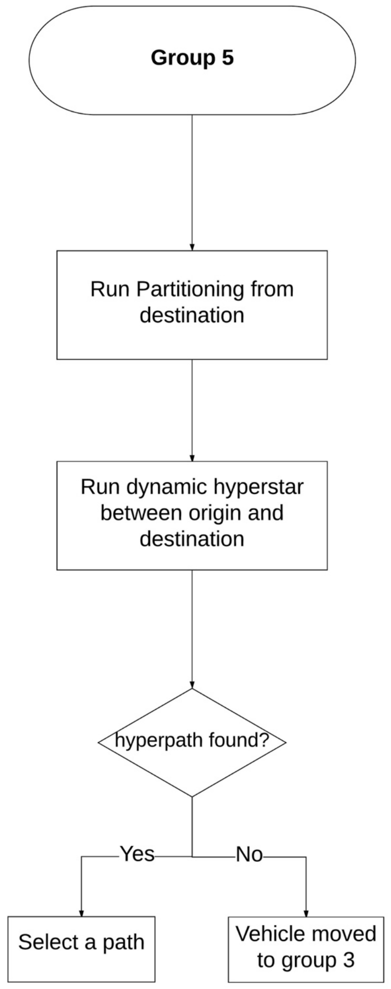

Group 5
Run Partitioning from destination
Run dynamic hyperstar between origin and destination
hyperpath found?
Yes
No
Select a path
Vehicle moved to group 3

Figure 25 Group 5 Processing

- Group 6: These vehicles are not affected by the flood. Their origin, destination, and path are not expected to flood or to be considered as soon-to-be flooded within the duration of the trip. As a result, they just receive a message that alerts them about the locations to be avoided in case of any deviation from the planned trip.

Table 10 summarizes the tasks that correspond to each group of vehicles. Affected vehicles that can still travel receive routing assistance (vehicles in groups 1, 2, 4 and 5), whereas affected vehicles that cannot reach any safe destination wait at the origin (group 3) and unaffected vehicles receive warning messages disclosing the areas to avoid (group 6).

Table 10 Vehicle Group Routing Tasks

Group	Description	Tasks
1	Vehicle with an **origin** on a link that is soon-to-be flooded at departure time and a **destination** on a link that is soon-to-be flooded or closed at the expected arrival time	a. Set of paths from origin to node in buffer 1 (*hyperpath may include soon-to-be flooded links but not closed links)* b. Set of paths from node in buffer 1 to node in buffer 2 *(hyperpath cannot include soon-to-be flooded or closed links)*or wait at buffer 1
2	Vehicle with an **origin** on a link that is soon-to-be flooded at departure time	a. Set of paths from origin to node in buffer 1 (*hyperpath may include soon-to-be flooded links but not closed links)* *b.* Set of paths from node in buffer 1 to destination *(hyperpath cannot include soon-to-be flooded or closed links)* or wait at buffer 1
3	Vehicle with an **origin** on a link that is closed at departure time; or vehicle that cannot reach any safe destination	Send warning: "WAIT at origin"
4	Vehicle with a **destination** on a link that is soon-to-be flooded or closed at the expected arrival time	Set of paths from origin to node in buffer 2 *(hyperpath cannot include soon-to-be flooded or closed links)*
5	Vehicle with an **intermediate** link that is soon-to-be flooded or closed at the expected time of entry	Set of paths from origin to destination *(hyperpath cannot include soon-to-be flooded or closed links)*
6	Unaffected vehicles	Send warning: areas to avoid

As shown in Figure 26, the output files of the buffer generation and vehicle analysis steps are the inputs of the process step. The output files of the process step are:

- A file that includes, for each vehicle that has been routed, the corresponding hyperpath along with the total travel time and the computation times in java, and
- A list of the vehicles in each group.

For translation to the field, since the route guidance system is assumed to be an in-vehicle system, the generation of the hyperpath for each vehicle is executed independently, after the retrieval of the corresponding current link data and potentials from the TMC's database. In this research, the vehicles in the system have been evaluated sequentially. Since the hyperstar generates multiple paths between an origin and a destination, we randomly select one path to simulate the human behavior. However, the path selection would be left to the user in future connected vehicle environment. The pseudocode of this step is shown in Appendix A.

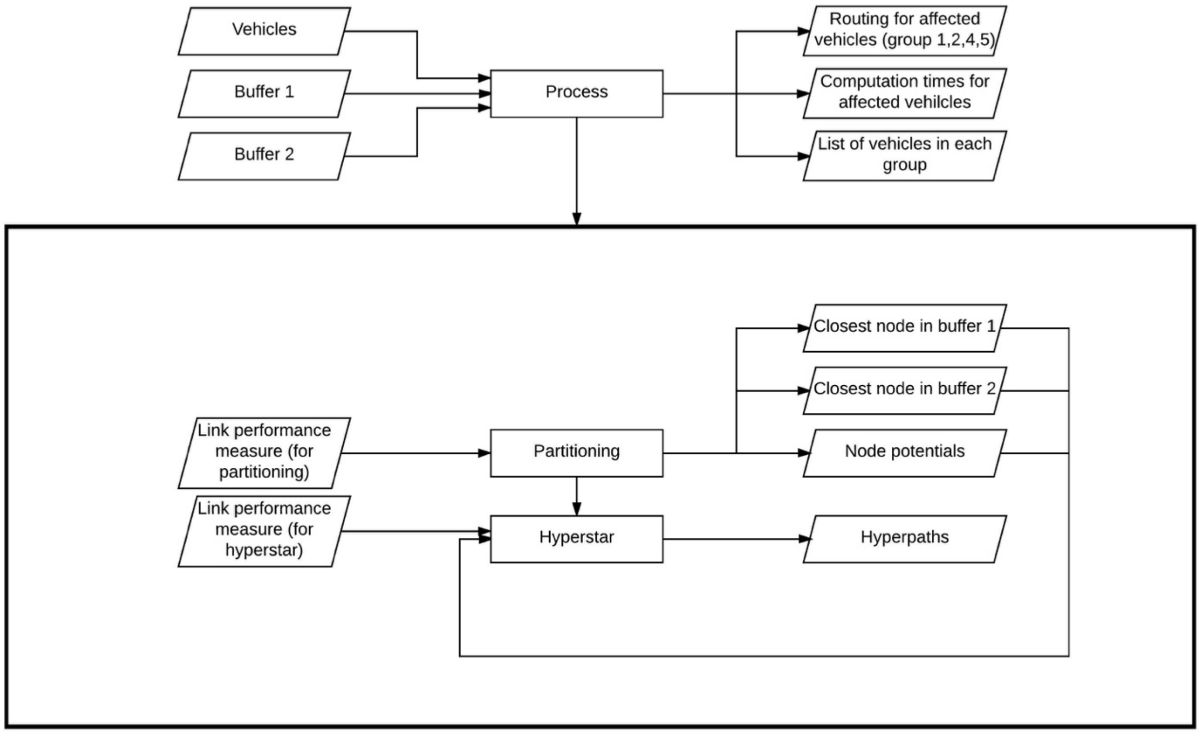

Vehicles
Buffer 1
Buffer 2
Process
Routing for affected vehicles (group 1,2,4,5)
Computation times for affected vehilcles
List of vehicles in each group
Link performance measure (for partitioning)
Link performance measure (for hyperstar)
Partitioning
Hyperstar
Closest node in buffer 1
Closest node in buffer 2
Node potentials
Hyperpaths

Figure 26 Input/ Output data of process step

In Figure 27, the four parts along with the corresponding inputs and outputs are illustrated.

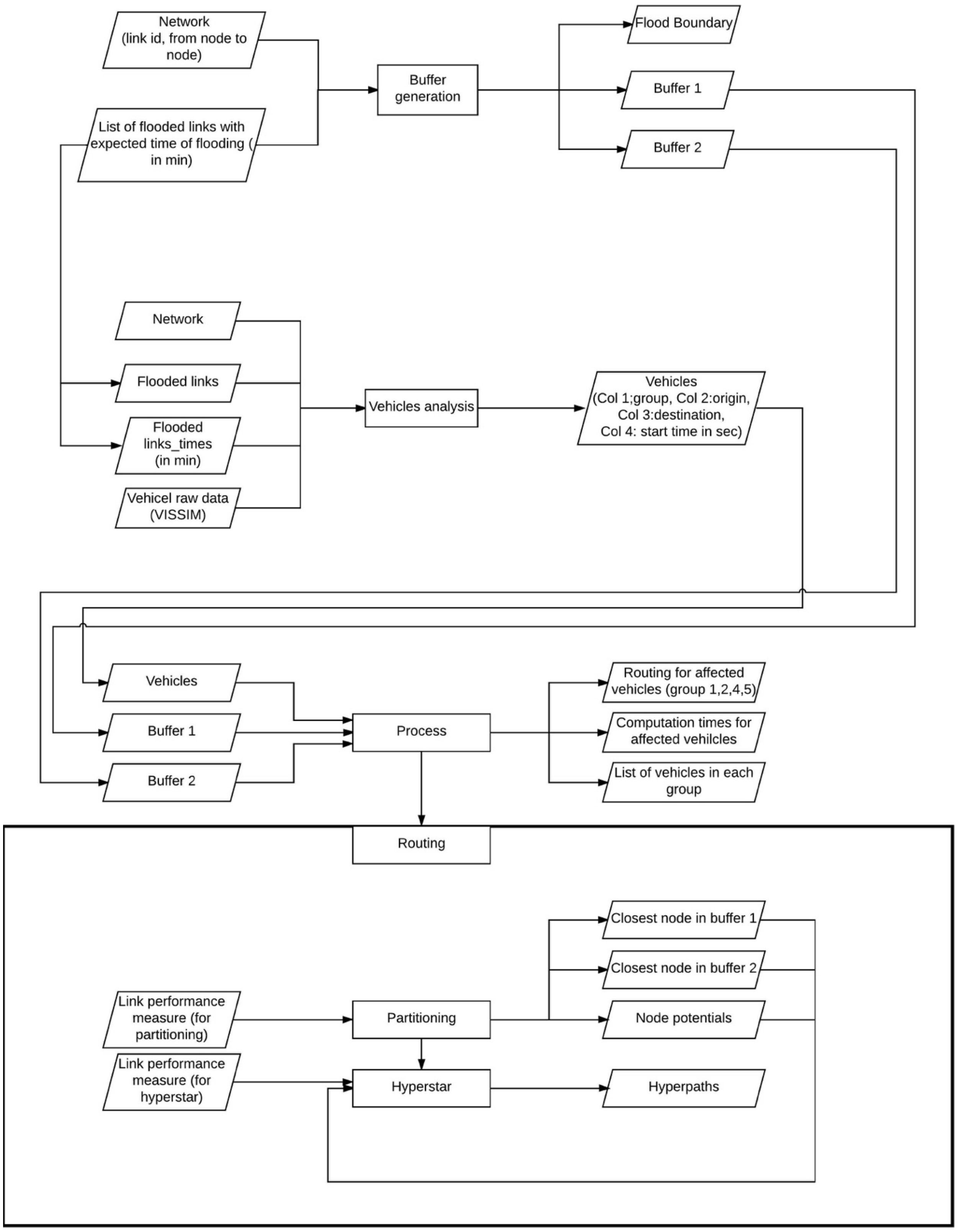

Network (link id, from node to node)
List of flooded links with expected time of flooding (in min)
Buffer generation
Flood Boundary
Buffer 1
Buffer 2
Network
Flooded links
Flooded links_times (in min)
Vehicel raw data (VISSIM)
Vehicles analysis
Vehicles (Col 1;group, Col 2:origin, Col 3:destination, Col 4: start time in sec)
Vehicles
Buffer 1
Buffer 2
Process
Routing for affected vehicles (group 1,2,4,5)
Computation times for affected vehilcles
List of vehicles in each group
Routing
Closest node in buffer 1
Closest node in buffer 2
Node potentials
Link performance measure (for partitioning)
Partitioning
Link performance measure (for hyperstar)
Hyperstar
Hyperpaths

Figure 27 Coding flowchart

4.3 Applications of the Framework

The two locations which were designated as the case study locations for this analysis consisted of the intersection of Baltic Ave. and 21st St. and the eastern section of the Shore Drive corridor in Virginia Beach, VA. These locations were recommended by personnel from the City of Virginia Beach. A network was then identified for each case study area through visual inspection using aerial photography and street maps. The network for each location ensured a large enough area and routing alternatives given a flooding scenario at or near the identified case study locations.

The first case study location network consists of the intersection of Baltic Ave. and 21st St., extends approximately 1 mile North, 0.35 miles East, 1.35 miles South and 2.5 miles West. Figure 28 illustrates the network outlined in red, where the case study location is denoted by a blue circle.

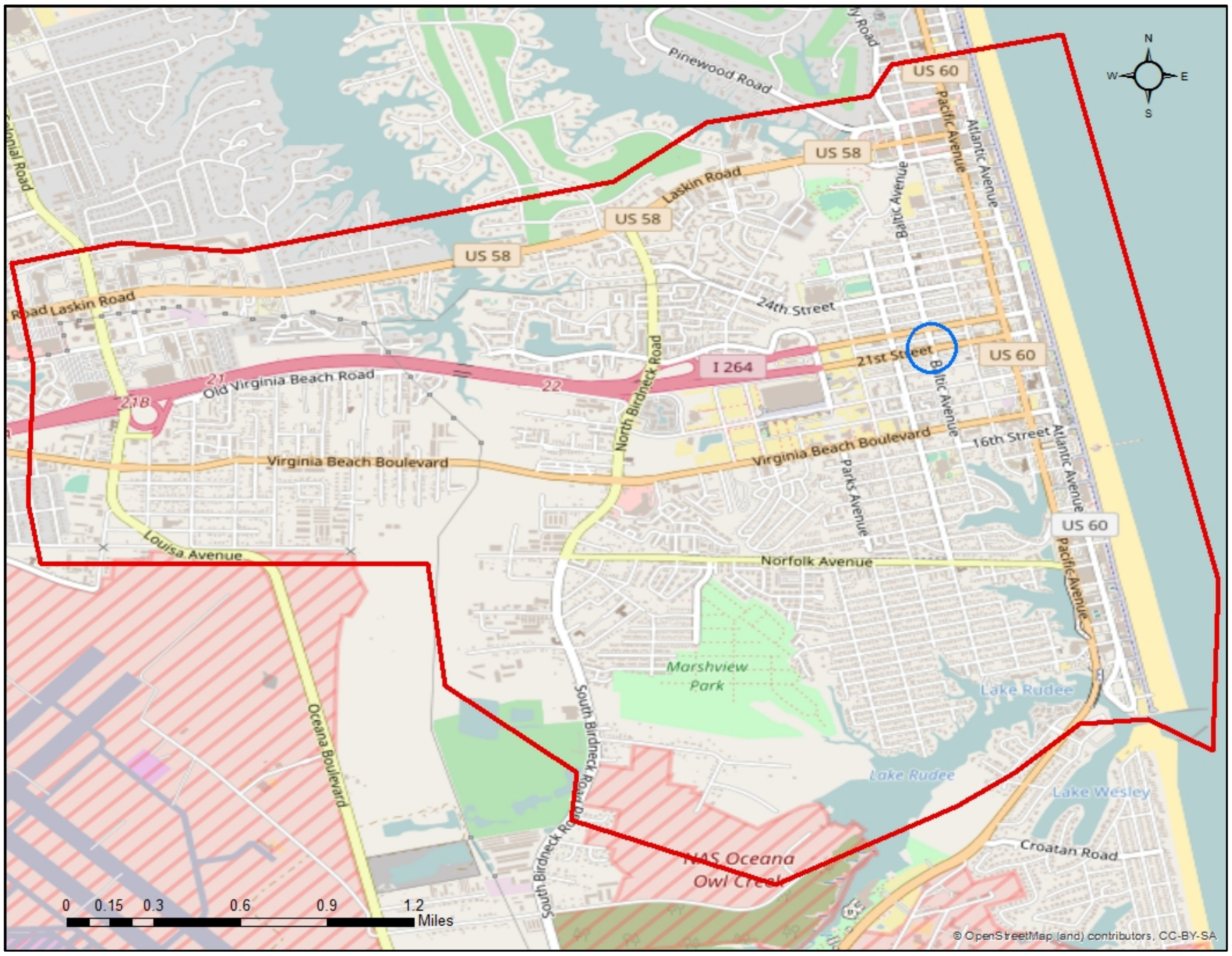


Figure 28 Baltic Ave. and 21st St Case Study Network

The second case study location network consists of the Shore Dr corridor, the selected boundaries extend approximately 7 miles from East to West, and ranges from a 1-2 mile North to South separation as illustrated in Figure 29. Only the eastern part of this network was used to test the framework.

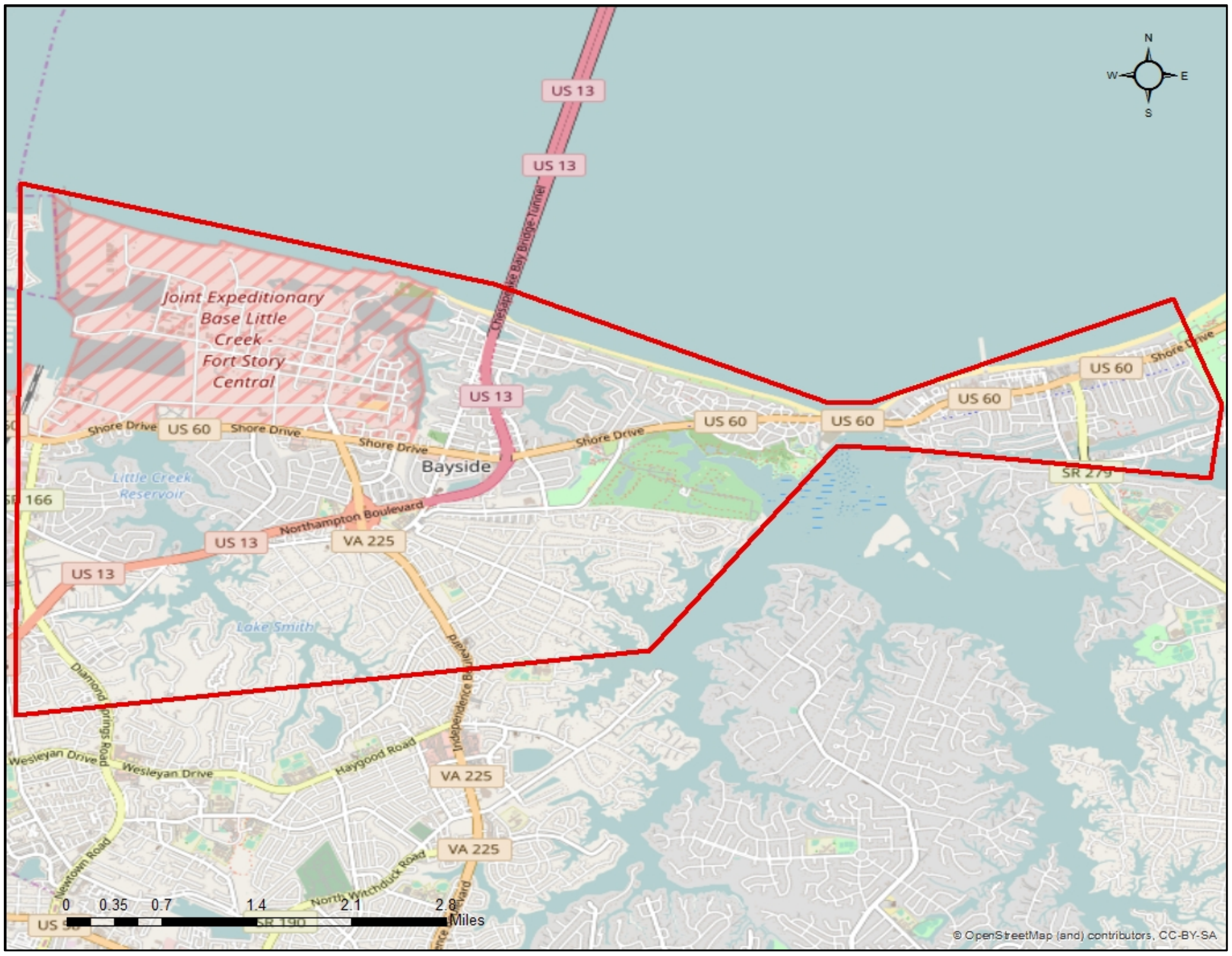


Figure 29 Shore Dr case study network

4.3.1 Generating Framework Input with VISSIM

The networks were constructed in the PTV VISSIM microscopic traffic simulation software for modelling purposes and the determination of travel time functions for use in the determination of undelayed travel times. Constructing the networks within the PTV VISSIM software required a significant amount of attention to detail as different attributes such as links, stop signs, signal timings, vehicle routes, speed decisions, conflict areas and specifying where to collect data throughout the networks had to be inserted individually.

A *link* in VISSIM can be described as the connection between two different street segments, the connection itself is accomplished using *connectors*. A link in VISSIM is only able to carry flow in one direction, making it a directed link. Connectors are commonly used at street intersections, when there are changes in the number of lanes in corridors, and merging locations in highway entrances and exits. Figure 30 illustrates the number of links and connectors in a common four-way intersection modeled in VISSIM; the links are denoted in blue, while the connectors are in purple. The construction of a four-way intersection model in VISSIM requires eight (8) links, and twelve (12) connectors.

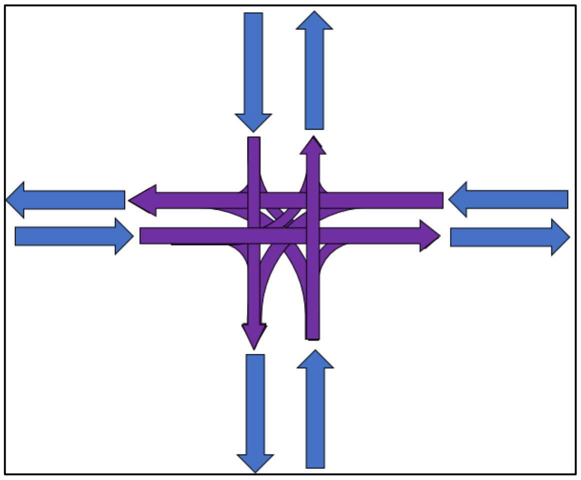


Figure 30 Links and connectors in a VISSIM model intersection

Stop sign locations were identified using Google Maps street view as well as Google Maps aerial images and VISSIM's built in BING maps aerial imagery. Stop signs were placed in VISSIM as a solid bar parallel to the link at their corresponding location on a per lane basis, thus if a two-lane segment is regulated by a stop sign, the stop bars were required on each lane. Traffic signal heads were similarly input into the VISSIM models and were also illustrated by bars. On most of the occasions, stop bars and traffic signal head bars were placed at the end/downstream section of the link. Traffic signal timings were provided by personnel from the City of Virginia Beach, and each signalized intersection required individual timings to be input into the VISSIM model.

Static vehicle "routes"[6] were utilized to specify the distribution of vehicle movements throughout the network and avoid possible gridlock situations. Static vehicle "routes" were placed in VISSIM as a set of two bars indicating a start and a finish, and were usually placed at the beginning/upstream section of the link were the "path" began and the beginning/ upstream section of the link of where the "path" ended. On most occasions, the "path" covered the distance of an entire link, connector and just the initial section of the next link in its route. These "routes" were placed on every link covering the entire network. Figure 31 illustrates an example of the static vehicle "routing." A fraction was required to be input to identify the percentage of vehicles that would make a left, through or right movement. Traffic count data made available through the City of Virginia Beach website as well as engineering judgement and simulation observations were used to determine the fraction values.

Speed decisions for a given road segment were also incorporated to have a more accurate model. Similar to stop signs and traffic signals, speed limits were identified utilizing Google Maps street view whenever possible and engineering judgment was used when the information was not available. On most occasions, speed decisions bars were commonly placed at the beginning /upstream section of the link. This varied when a speed limit would change within the link segment; then the speed decision bar was placed where appropriate.

[6] These routes, in VISSIM terminology, are microscopic considerations, separate from the macroscopic routes generated by the Hyperstar algorithm.

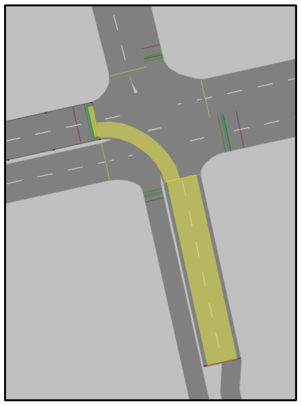


Figure 31 Vehicle "routing" for a left turn movement in VISSIM

Conflict areas were also addressed individually on all intersections. Conflict areas gave additional priority to specific vehicle movements and gave directions to other vehicles to yield until proper clearance was available. An example of a conflict area in a typical intersection had the left turning movement yielding to through traffic, or a right movement yielding to through traffic. Figure 32 shows an example of the conflict area described in a VISSIM model. The red denotes where vehicles yield while the green denotes right of way.

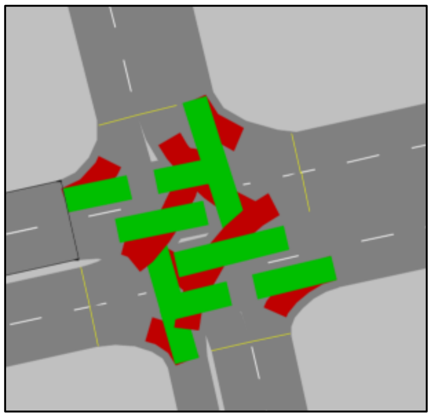


Figure 32 Conflict areas for a four-way signalized intersection in VISSIM

Next, vehicle input or volume/flow rate entering the networks needed to be added to VISSIM. Vehicle inputs were placed at the beginning/upstream section of the link. The volumes entered in the vehicle inputs were distributed in hourly rates, thus having units of flow in veh/hr. The vehicle volumes utilized were estimated through reference to traffic count data available through the City of Virginia Beach website. Vehicle inputs were placed at unique locations using engineering judgement to ensure that each link within a network could collect enough data for evaluation.

With the network and vehicle input locations established, vehicle travel times per link and per movement could be gathered. Vehicle travel time was collected by placing bars at both the beginning/upstream section of the link and indicating the distance to the end/downstream of the link and connectors. On most occasions, for a typical four-way intersection, a total of four travel time paths were required to be input. The first to collect data on the primary link, the second for the link and the left turn movement connector, the third for the link and the through movement connector, and the fourth for the link and the right turning movement, as illustrated in Figure 33.

It is important to note that all the bars begin at the same location, travel times were collected for vehicles traveling on the upstream link and through their respective movement. Differentiation between the link travel times and the link and connector travel times provided the amount of time a vehicle travels per movement.

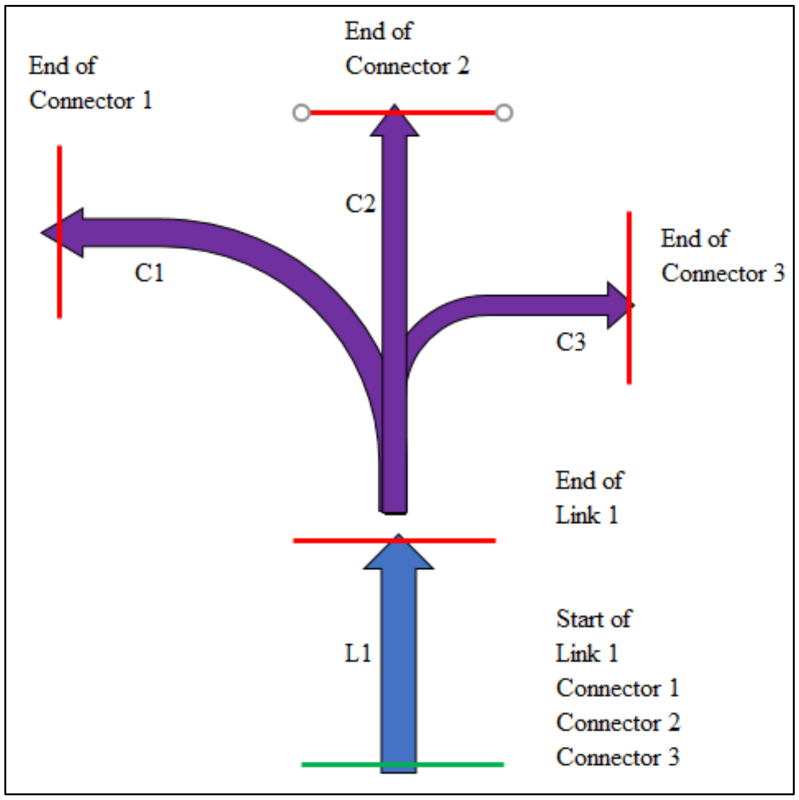


Figure 33 Placement of vehicle travel time bars in VISSIM

The methodology discussed above regarding travel time collection on links and collectors required an external tracking method to later identify which movement corresponded to each connector. Table 11 illustrates the tracking approach for the above illustration; input for all links and connectors were tracked on separate tables for determining travel time per movement.

Table 11 Link and Connector Tracking Approach for VISSIM Networks

Upstream Link	Connector		
	Left	Through	Right
1	1	2	3

Table *12* illustrates the summary values of the attributes for both the Baltic and 21st Network and the Shore Dr. Network.

Table 12 Summary of Attributes in VISSIM Networks

VISSIM Attributes	Network	
	Baltic & 21st Ave	Shore Dr - East
Links	2221	283
Connectors	4411	460
Total (Links + Connectors)	6632	743
Stop Signs	632	54
Signalized Intersections	62	4
Signal Heads	449	39
Vehicle Routes	1574	139
Speed Decisions	2199	67
Vehicle Inputs	311	45
Vehicle Travel Times	6632	743

4.3.1.1 Node Consideration

The consideration of nodes was required to successfully implement the hyperstar algorithm. The PTV VISSIM software however, does not consider nodes within its methodology as it works with *links* and *connectors*. Therefore, an external method of inserting and tracking nodes was developed. A node was required to be placed between every link and connector for both networks. Thus, information regarding the number of links and connectors was required. A useful summary of all the links and connectors was provided by VISSIM upon the completion of a network. Table 13 illustrates a reduced sample from the Baltic and 21st Network.

Table 13 Link and Connector Summary List from VISSIM

$LINK:NO	NAME	NUMLANES	LENGTH2D	ISCONN	FROMLINK	TOLINK
2373	Close Ave	1	175.172	0		
2374	Virginia Beach Blvd	2	255.74	0		
2375	Cardinal Rd	1	164.305	0		
10000	First Colonial Road	1	69.134	1	358	360
10001	First Colonial Road	2	30.886	1	358	361

Important items to identify from Table 13 were the unique link and connector numbers. The "ISCONN" column identifies whether the row was a link or a connector, by either a 0 or a 1 respectively. If it was a connector, information regarding the upstream link was in the "FROMLINK" column and the downstream link was in the "TOLINK" column. Therefore, assurance of a complete list of links and connectors was provided by VISSIM.

The incorporation of nodes was then accomplished by stating that only the links were to have a start node (S_n) and an end node (E_n) assigned, where n references the link. Nodes were added in increasing order for every link row. (For example, Link 1 was assigned $S_1 = 1$, $E_1 = 2$; Link 2 was assigned $S_2 = 3$, $E_2 = 4$, etc…) Thus, the start node of the connector was assigned as their respective upstream link's (FROMLINK) end node $(E_{n\text{-}UpstreamLink})$, and similarly their end node was their respective downstream link's (TOLINK) start node $(S_{n\text{-}DownstreamLink})$. Figure 34 illustrates the above explanation.

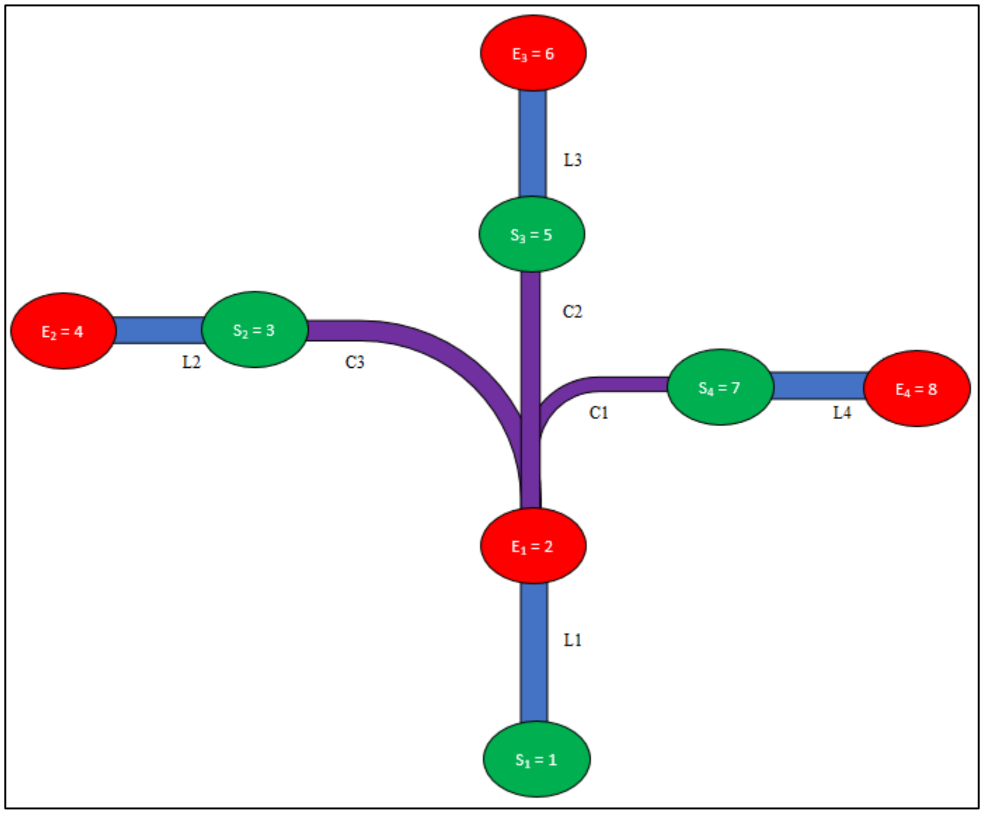


Figure 34 Nodes added to the VISSIM network

4.3.1.2 Data and Travel Time Function Procedure

4.3.1.2.1 VISSIM simulation organization

To obtain the required vehicle travel time data and develop travel time functions for individual links and connectors, VISSIM simulations were executed multiple times under various flow rates or volume vehicle inputs. As previously described, the vehicle inputs were determined through reference to traffic counts from the City of Virginia Beach website. The initial vehicle inputs were labeled as the base case, and a total of five simulations were executed under different seed values. Each simulation ran for a total of 3600 seconds equivalent to 1 hour. Similarly, the base case volumes were then increased by ten (10%) and re-executed five times until doubling the initial base case value (100%). This was conducted to track the varying travel time under different flow conditions. Table 14 illustrates a summary of the previous description.

The process was simplified by taking advantage of VISSIM's COM API. A VBA script (in Appendix B: VISSIM COM API: VBA Simulation Script with Varying Vehicle Inputs) was written that was capable of increasing the vehicle input volumes and running each simulation the specified number of times. Thus, the necessary data was accurately collected and human error was minimized.

Table 14 Simulations Conducted in VISSIM

Simulation Volume	Simulation
Base Case	1 – 5
+10%	5 – 10
+20%	10 – 15
+30%	15 – 20
+40%	20 – 25
+50%	25 – 30
+60%	30 – 35
+70%	35 – 40
+80%	40 – 45
+90%	45 – 50
+100%	50 - 55
Total	55

4.3.1.2.2 VISSIM data output

The data of interest obtained from the final VISSIM simulations consisted of records indicating when a vehicle had passed over the start of a vehicle travel time bar, and when it had passed the end of a vehicle travel time bar. As previously discussed, vehicle travel time bars were placed at the start and end of every link, and the start and end of every connector and their respective upstream link. Table 15 illustrates a condensed sample of the data provided by the VISSIM software.

Table 15 Raw Vehicle Travel Time Data from VISSIM

SimRunID	Time	No_	veh	VehType	Trav	Delay
1	2.55	893	3	100	2.19	0.89
1	2.68	2019	7	100	1.35	0
1	3.73	13995	7	100	2.4	0
1	4	117	1	100	3.82	0
1	4.71	726	4	100	4.19	1.8

In Table 15 the first column, “SimRunID” indicates the simulation run and to in vehicle volume input it was categorized. The “Time” column indicates the time in seconds of when a vehicle passed through the start of the vehicle travel time bar. The “No_” column, indicates the unique ID of the vehicle travel time start and finish. In this study, since the travel time was of interest on a per link and per connector basis, the “No_” was changed to match the link ID. In cases where there was a connector and an upstream link, the connector ID was input as the “No_”. The “veh” column identifies the unique vehicle throughout the simulation, thus being able to track the path as a vehicle navigates through the network. “VehType” indicates the vehicles type such as car, bus, etc. “Trav” indicates the vehicle travel time from the start to the end of the vehicle travel time bars, or in this situation, the travel time per link/upstream link and connector. Finally, “Delay” indicates the delay experienced by a vehicle when traveling from the start to the end of the vehicle travel time bars, or in this situation, the travel time per link/upstream link and connector.

4.3.1.2.3 Preparation of raw VISSIM data into the desired format

The raw data provided by VISSIM was substantial as every link recorded all vehicles traveling it. Additional measures were required to format the data into a configuration capable of having travel time functions extracted. The flow rate for each link/connector was obtained by counting the total number of vehicles that were recorded entering and exiting a segment. Since the simulation was executed for 3600 seconds, the addition of these vehicles resulted in a flow rate in units of veh/hr.

The re-formatting process was completed using the R-Studio software, but the steps were as follows.

1. Insert a "Helper" column. After importing the raw data into the R-Studio environment, an additional column titled "Helper" was inserted and composed of a string variable representing the simulation run and the link/connector ID, separated by an underscore. (Ex. If "SimRunID" = 1 and "No_" = 3, "Helper" = 1_3)
2. Create new data table with unique "Helper" values. The "Helper" column, later helped identify all unique values per link/connector. The total observations should equal the product of total number of links/connector in the network and the number of simulations (in this study, the simulations were 55)
3. Using a new data sheet, add a new column "Q" and count number of times "Helper" is observed in raw data. This resulted in a summary of observed flow rates (veh/hr) per link/connector per simulation
4. Create unique columns of Simulation and Link/Connector by splitting the "Helper" column.
5. Determine the average travel time per simulation per link/connector. This was accomplishing by averaging the travel time for all unique "Helper" values.
6. Determine and add the free flow travel time by subtracting the delay per link/connector from the observed travel time.

Table 16 illustrates a sample of what the final formatted table had in terms of data. Appendix C: R-Studio Script to Format Raw VISSIM Output contains the R-Studio script to complete this process from VISSIM's raw data.

Table 16 Formatted Data by R-Studio Script from VISSM's Raw Output

Helper	SimRun	Link	Q	Travel Time	Delay	Free Flow Travel Time
1_1	1	1	110	12.286	0.562	11.724
1_2	1	2	83	14.341	1.157	13.184
1_3	1	3	127	36.484	0.792	35.691
1_4	1	4	147	39.227	3.782	35.446
1_5	1	5	247	23.327	2.536	20.791

4.3.1.2.4 Travel time function procedure

Obtaining travel time functions for every link/connector in the constructed VISSIM networks consisted of utilizing the newly formatted data from the previous step. By sub-setting the entire data set by the "Link" column, the average travel time per link for each simulation under varying flow conditions was obtained. Therefore, a dataset with 55 observations was obtained for every link/connector constructed in the VISSIM network model. Figure 35 illustrates a plot from one link in one of the analyzed networks.

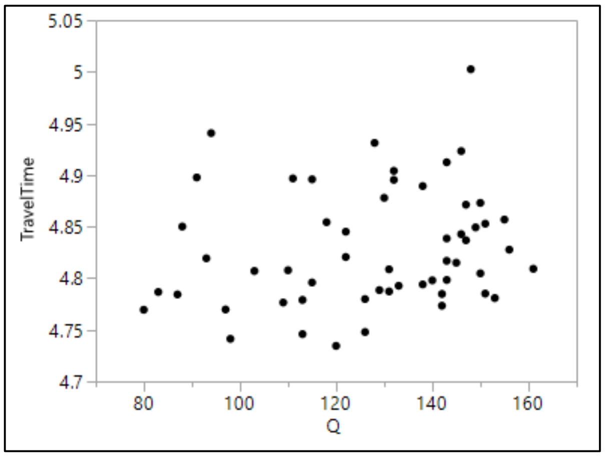


Figure 35 Example of Formatted Data Plotted with Observed Travel Time vs Observed Flow Rate (Q)

The travel time function for this study consisted of a special non-linear case with travel time as the dependent variable and flow as the independent variable. The travel time was considered in units of seconds and flow was considered in units of vehicles/hour. Additionally, the intercept was constrained to the free flow travel time and a square transformation was applied to flow. Equation (1) illustrates the travel time function form. Where *n,* represents the current link and *Q,* represents the flow rate in veh/hr.

$$Estimated\ Travel\ Time_n = Free\ Flow\ Travel\ Time_n + \beta_1(Q)^2 \qquad \text{Eq. 1}$$

Equation (1) was utilized to account for possible inconsistencies seen with a linear model. When considering the linear mode, there were cases where a negative intercept and/or a negative flow coefficient were encountered. Thus, by constraining the intercept to the free flow speed of a link with no additional vehicles on the segment; free flow travel time could be obtained for any given path throughout the network.

Figure 36 illustrates the differences between a linear model and the special form considered for this work. In Figure 36, the red line denotes the linear fit while the green line denotes the special from considered.

The above process was required to be completed for every individual link and their respective 55 data observations. To facilitate this process an R-Studio script was written to perform the non-linear regression procedure. Appendix D: R-Studio Script to Perform Non-Linear Regression and Obtain Travel Time Functions illustrates the R-Studio script for this process.

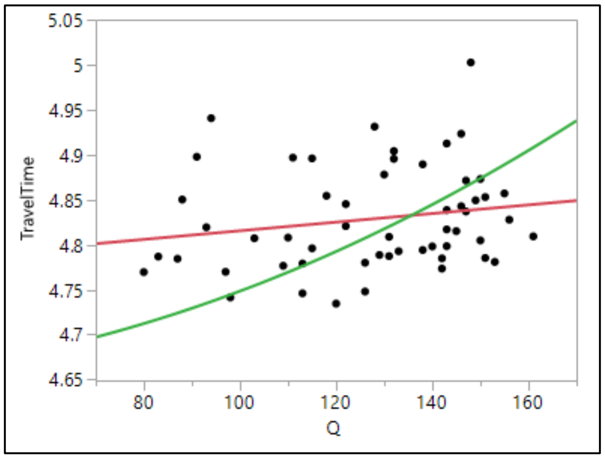


Figure 36 Example of linear and special form fit

4.3.2 Experiments

The two VISSIM coded transportation networks that were used in the experiments are shown in Figure 37 and Figure 38. The tests were executed on a MacBook Pro with a 2.7 GHz Intel Core i5 processor and a 16 GB 1867 MHz DDR3 memory.

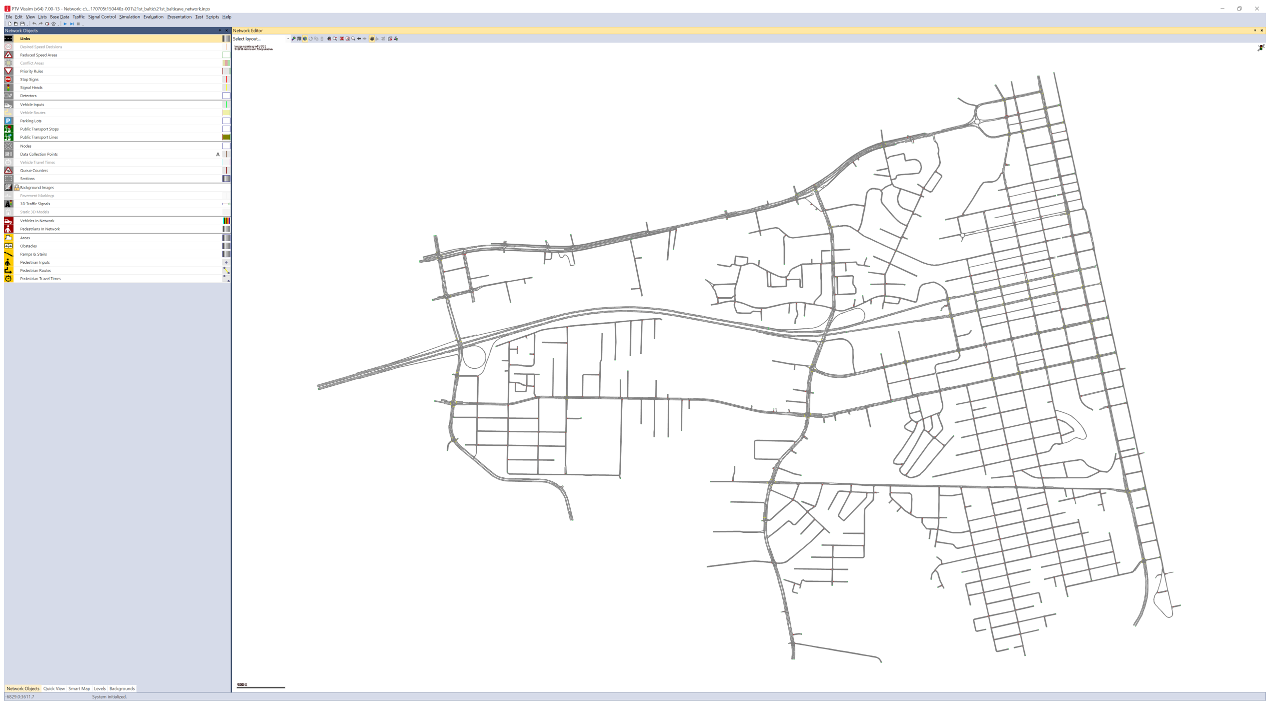


Figure 37 Network 1 in VISSIM (Baltic Ave. and 21st street)

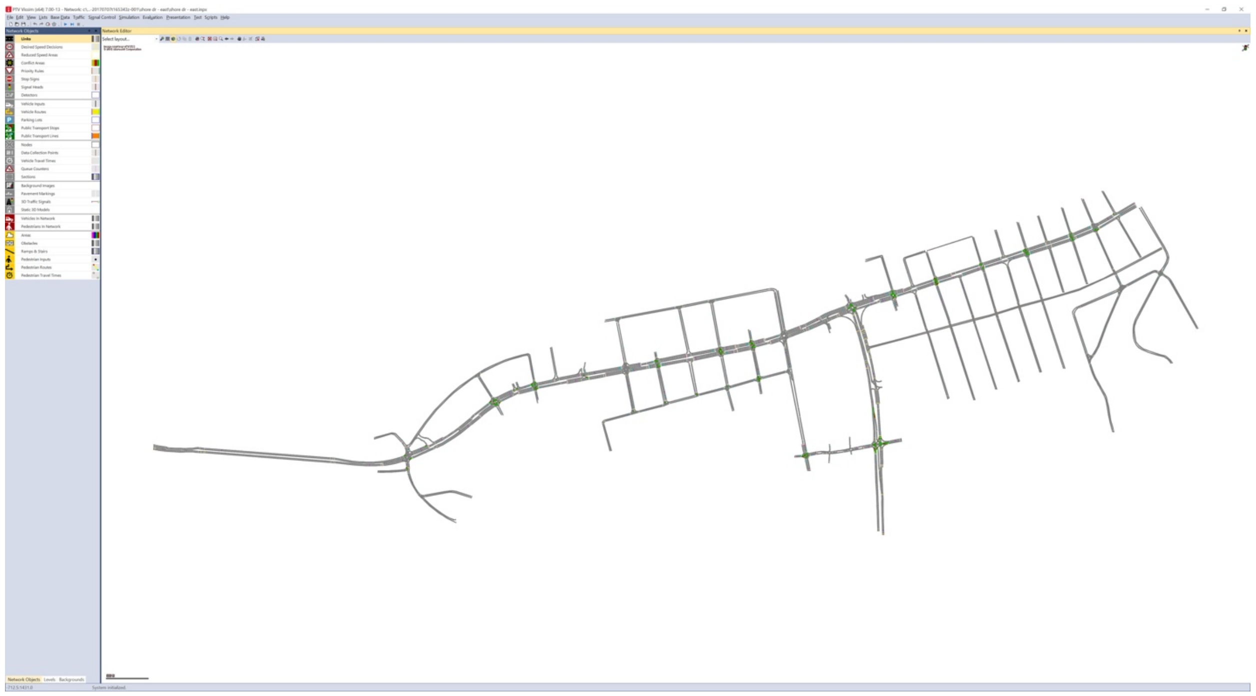


Figure 38 Network 2 in VISSIM (Shore Drive Ave.)

Two experiments were conducted. The first examined the scalability of the framework based on the two networks. The second examined the sensitivity of the results to the frequency of weather and flood forecast updates. This latter test was only conducted on the larger network. The base-case scenario parameters are as shown in Table 17.

Table 17 Base-Case Scenario Parameters

Parameters	**Value**
Network	Network 1 (Baltic Ave. and 21st street)
Flood location and depth	Links at 2 m elevation and below
Frequency of weather and flood forecast updates	Every 10 minutes
Timing of flood	Elevation <= 0.5 m: t4=20 min 0.5 m< Elevation <=1 m: t4=30 min 1 m< Elevation <=1.5 m: t4=40 min 1.5 m< Elevation <=2 m: t4=50 min
Empty link interval	5 min
Minimum clearance interval	5 min *(In Baltic network: max link travel time = 2.22 min)*
Additional clearance interval	5 min

4.3.2.1 Scalability of the framework for different network sizes and structures (Test 1)

In this test, the framework was applied to Network 1 and Network 2 to evaluate the efficiency of the framework in terms of computational times. This comparison indicated how sensitive the computation times were to the network size and structure.

As shown in Figure 37 and Figure 38, the two networks have different sizes and structures. Network 1 is larger than Network 2 with approximately 9 times more links and 8 times more nodes. In addition, Network 2 has a more linear shape while Network 1 is more round. Network 1 has a higher percentage of local streets when compared to Network 2. The latter has a major arterial lying along the entire network from east to west. However, comparing the two networks visually is not sufficient. Table 18 presents graph theoretic indices that allow comparison of the two networks. First, the cyclomatic number is a measure that indicates the number of independent cycles that exist in a network. It is computed using equation (2). As shown in Table 19, Network 1 has a higher cyclomatic number than Network 2, yet this is mainly due to the fact that Network 1 is significantly larger than Network 2. The beta index is computed using equation (3) by dividing the total number of links by the total number of nodes. A tree network which is a connected network with no cycles has a beta index less than 1 while a network with only one cycle has a beta index equal to 1. Network 1 and network 2 both have a beta index higher than 1 and this indicates that the networks are complex networks and hence representing real-life systems. Regarding the connectivity of the networks, the alpha index is a measure that reflects the number of cycles with respect to the maximum possible number of cycles in the network. The higher the alpha index, the more connected the network is. Similarly, the gamma index is a descriptor that evaluates the total number of links with respect to the total number of possible links. A higher gamma index implies a more complete network. As shown in Table 19, Network 2 has a higher gamma index as well as a higher alpha index, thus, Network 2 is more connected and more complete than Network 1.

Table 18 Network Descriptors (Notation and Equation)

Descriptor	Notation	Equation	
Total number of links	A		
Total number of nodes	N		
Number of isolated subgraphs	p		
Cyclomatic number	μ	$\mu = A - N + p$	(1)
Beta index	β	$\beta = A/N$	(2)
Alpha index	α	$\alpha = 2\mu/((N-1)(N-2))$	(3)
Gamma index	γ	$\gamma = 2A/(N(N-1))$	(4)

In both networks, the same percentage of flooded links with respect to the total number of links is tested. Since the base-case scenario implies that 2.7 % (=178/6632) of the links in Network 1 are flooded, the 20 (=(743*178)/6632) lowest-lying links are considered flooded in Network 2.

Table 19 Networks Characteristics

Network characteristics	Network 1	Network 2
Number of links	6632	743
Number of nodes	4442	572
Number of flooded links	178	20
Cyclomatic number	2191	172
Beta index	1.49	1.3
Alpha index	0.00022	0.00106
Gamma index	0.00067	0.0045
Number of vehicles	5002	1984

As shown in Table 20, the number of affected vehicles in Network 2 is slightly higher than the one in Network 1. Yet, the percentage of affected vehicles in Network 2 is 6.8% (=135/1984) and is significantly higher than the 2.2% (=111/5002) for Network 1. The same percentage of flooded links was tested; however, the impact of the flood was more extensive in Network 2. The location of the flooded links relative to the origins and destinations of the vehicles affects the number of affected vehicles and is the main reason behind the difference in the numbers of vehicles per category (shown in Table *21*) even when the same percentage of links is flooded. In addition, the functional classification of the flooded links plays a major role in defining the impact of the flood on the vehicles. For instance, a flood on an arterial or a highway affects more vehicles than a flood on a residential and local street.

Table 20 Number of Vehicles per Category (Test 1)

	Network	
Category	1	2
Number of unaffected vehicles	4891	1849
Number of affected vehicles	111	135
Number of routed vehicles	100	130

Table 21 Number of vehicles per group (Test 1)

	Network			
	1		2	
Group	Initial grouping	Final grouping	Initial grouping	Final grouping
Group1	0	0	4	4
Group2	10	10	103	103
Group3	11	11	0	5
Group4	63	63	15	13
Group5	27	27	13	10

Table *21* shows the initial and final number of vehicles per group for each network. The initial grouping was the grouping that resulted from the “vehicle analysis” step in the framework while the final grouping was the one obtained after executing the “process” step. Different initial and final grouping occurred when vehicles that had to be routed (vehicles in group 1, 2, 4 or 5) were moved to group 3 after the "process" step. These vehicles that were moved to group 3 were not located on a closed link at departure time but could not reach a safe destination because of downstream floods which prevented the generation of any possible hyperpath between their origins and destinations. Regarding Network 1, the initial and final groupings were the same. However, in Network 2, two vehicles and three vehicles were moved from group 4 and group 5, respectively, to group 3 while vehicles in group 1 and group 2 all received routing assistance.

On the other hand, as shown in Table *21*, the affected vehicles were distributed differently across the groups in the two networks. For instance, no vehicle belonged to group 1 in Network 1 while in Network 2, 3% (= 4/135) of the affected vehicles were added to group 1 which represented the vehicles with a soon-to-be flooded origin and a soon-to-be or closed destination. Furthermore, 9% (=10/111) of the affected vehicles belonged to group 2 (i.e., vehicles with origins on soon-to-be flooded links) in Network 1 while 76% (=103/135) of the affected vehicles belonged to group 2 in Network 2. This significant difference was due to the fact that in Network 2, the flood occurred on links on which vehicles were generated. In addition, 10% (=11/111) of the affected vehicles belonged to group 3 and had to wait at the origin in Network 1; however, in Network 2, 0% of the affected vehicles had origins on closed links (initial grouping) but 3.7% (=5/135) were moved to group 3 because no hyperpath was found. In Network 2, no vehicle belonged to group 3 in the initial grouping because no vehicle was generated from the links with vehicle inputs after their corresponding time t3, while 103 vehicles (number of vehicles in group 2 in initial grouping) were generated between times t1 and t3 for these links that were considered soon-to-be flooded at the vehicles' departure time. Regarding group 4 (i.e., vehicles with soon-to-be flooded destinations) and group 5 (i.e., vehicles with soon-to-be flooded or closed intermediate link(s)), higher percentages per group were recorded in Network 1 compared to Network 2. The grouping of vehicles highly depended on the position of the flooded links relative to the position of the generation points in the modeled network (in VISSIM). In real life, a flood that leads to the closure of major production and attraction links as well as a flood on higher capacity links is expected to affect the system more than floods on links with low productions and attractions and floods on local roads.

Table 22 shows the average, minimum, and maximum computation times recorded for Network 2 were lower than those in Network 1. Similarly, all average computation times per group were smaller for Network 2. These results were expected because Network 2 was smaller than Network 1 and the hyperpath computations took less time because the hyperpaths in Network 2 were, by default, shorter than the ones in Network 1. The group with the highest average computation time was group 4 in Network 1. This group included the vehicles with soon-to-be flooded or closed destinations at their expected arrival time. This group recorded the highest average computation time due to the time-consuming search for a new destination. For instance, the maximum computation time of 49.04 seconds belonged to a vehicle (ID: 4394) in group 4 and the search for a new destination required 48.95 seconds. To minimize the significant computation times required for the search for a new destination for vehicles in groups 1 group 4 (i.e., vehicles with soon-to-be flooded or closed destinations), the search for a new destination can be stopped after evaluating a specific number of tentative destinations. In future implementation, the framework can also be extended to allow the users in groups 1 and 4 to enter new destinations of their choice; thus, enhancing the user's preferences and the computational efficiency by eliminating the search for a new destination. In Network 2, the group with the highest average computation time was group 2. The computation times for this group were the sum of the computation times needed to generate a hyperpath from the origin to a node in buffer 1 and the computation times needed to generate a hyperpath from buffer 1 to the destination. In Network 2, the average computation time was small which meant the search for new safe destination was less time-consuming in smaller networks.

Table 22 Computation Times in Test 1

	Network	
Computation times (seconds)	1	2
Average	1.47	0.06
Min	0.39	0.01
Max	49.04	0.27

Table 23 Average Computation Times per Group for Test 1

Average computation times (seconds)	Network	
	1	2
Group 1	NA	0.06
Group 2	1.20	0.07
Group 4	1.88	0.05
Group 5	0.61	0.02

As shown in Figure 39, in Network 1, the computation time interval that included the highest percentage of routed vehicles was the interval between 0.5 and 1 second. Three vehicles in group 4 recorded a computation time greater than 2 seconds; these vehicles were assigned to a new destination and the search for a new accessible destination in buffer 2 was time-consuming. In Network 2, the computation times for all routed vehicles fell in the interval between 0 and 0.5 second. The reason why the computation times in Network 2 were significantly smaller than in Network 1 was the network size. The time needed for the search for a new destination for vehicles in group 4 was the most sensitive to the size of the network. This test indicated that the increase of the network size resulted in an increase of the required computation times recorded by routed vehicles. Yet, the computation times recorded in Network 1, except for the vehicles that recorded a computation time greater than 2 seconds, were still reasonable and can be minimized by stopping the search after scanning a specific number of tentative destinations. Further increases in the network size can be tested in the future.

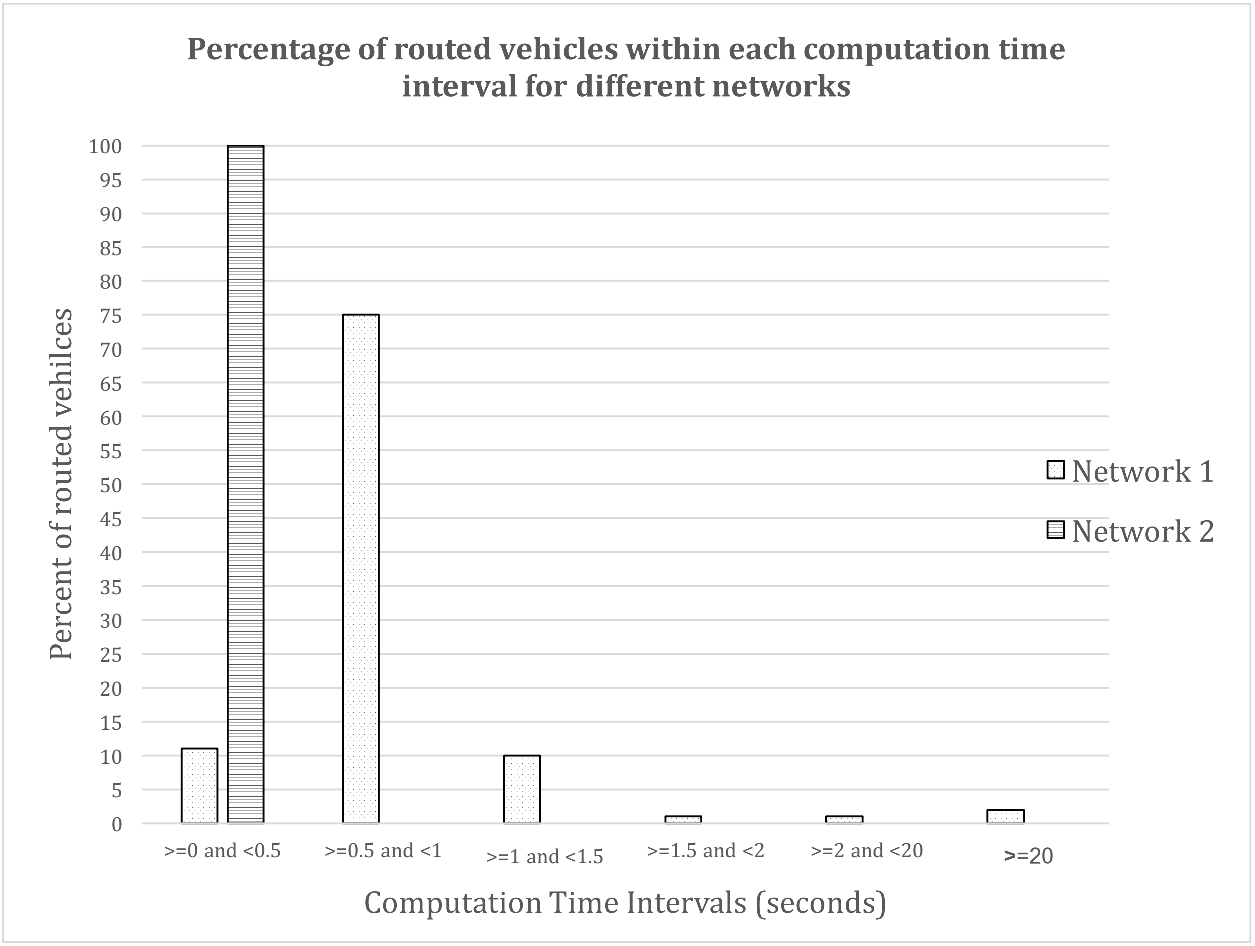


Figure 39 Percentage of routed vehicles within each computation time interval for different networks for Test 1

4.3.2.2   Sensitivity to Weather and Flood Update Frequency (Test 2)

In this study, we assumed that the TMC updated and received the flood input data from weather and flood forecasts systems at a specific frequency. This test evaluated the results' sensitivity to variation in the weather and flood forecasts' update frequency. The following scenarios were tested and represented in Table 24:

- Scenario 1: 1-minute update frequency (or near-continuous update)
- Scenario 2: 5-minutes update frequency
- Scenario 3: 10-minutes update frequency (base scenario)

Table 24 Time of Flood (t4) Corresponding to Each Elevation Interval in Each Scenario

Elevation interval	**Real time of flood**	**Scenarios**		
		1	2	3
Elevation <= 0.5 m	28	28	25	20
0.5 m< Elevation <=1 m	33	33	30	30
1 m< Elevation <=1.5 m	49	49	45	40
1.5 m< Elevation <=2 m	59	59	55	50

As shown in Table 24, as the weather and flood forecasts were updated less frequently, early closures resulted. For instance, if near-continuous updates (i.e., every 1-minute) indicated that a given link flooded at time t = 28 min, the time of flood (t4) of the same link was considered at time t= 25 min and at time t=20 min with an update frequency of 5 minutes and 10 minutes, respectively. In the case of the 10-minute updates, the links are considered closed and soon-to-be flooded 8 minutes earlier than the actual flood time.

Table 25 indicates that with less frequent weather and flood forecast updates, the numbers of affected vehicles and routed vehicles (i.e. vehicles in groups 1, 2, 4 and 5) increased. Links were closed and considered as soon-to-be flooded earlier in time. Hence, fewer weather and flood forecast updates resulted in a lengthier impact on the road users. In scenario 1, the first set of links considered as soon-to-be flooded were at time t= 13 min (=28 – Empty link interval of 5 min – Minimum clearance and additional clearance interval of 10 min) while the first set of links considered as soon-to-be flooded in scenario 2 was at time t=10 minutes and at time t=5 minutes in scenario 3.

Table 25 Number of Vehicles per Category for Different Update Frequencies

	Update Frequency		
Category	**1 min**	**5 min**	**10 min**
Number of unaffected vehicles	4939	4916	4891
Number of affected vehicles	63	86	111
Number of routed vehicles	56	79	100

As shown in Table 26, fewer updates led to the increase of the number of vehicles in each group for all cases except the shift from a 1-minute frequency to a 5-minute frequency in which the number of vehicles in group 3 did not change. For instance, fewer updates resulted in earlier “t3” for all links. Let durations X_i be the difference between the time t3 of a link in a 1-minute frequency update and the time t3 the link in a 5-minute frequency updates, “i" being the link’s elevation interval. Based on the simulated vehicle data obtained from VISSIM, no additional vehicle had a departure on one of the flooded links during this duration X_i; thus, no additional vehicle was added to group 3 when shifting from scenario 1 to scenario 2. However, when shifting from a 5-minute frequency to a 10-minute frequency, four additional vehicles were added to group 3 and were subsequently prohibited from leaving their origins because no hyperpaths to their final destinations were found. These vehicles belonged to group 2 in scenarios 1 and 2 during which they were able to leave their soon-to-be flooded origins to reach their corresponding destinations.

Table 26 Number of Vehicles per Group for Different Update Frequencies

	Update Frequency		
Group	**1 min**	**5 min**	**10 min**
Group1	0	0	0
Group2	5	9	10
Group3	7	7	11
Group4	37	51	63
Group5	14	19	27

In Table 27, the basic statistical measures of the computation times of the routed vehicles for each scenario are shown. The average computation time decreased with fewer updates. With more updates, the vehicles with longer trips remained affected and these vehicles required more computation time to generate their corresponding paths than the ones with short trips. The maximum recorded computation time in all the scenarios belonged to the same vehicle (ID: 4394) that was discussed earlier.

Table 27 Computation Times for Different Update Frequencies

	Update Frequency		
Computation times (sec)	**1 min**	**5 min**	**10 min**
Average	2.09	1.72	1.47
Min	0.36	0.35	0.39
Max	49.03	49.88	49.04

Figure 40 shows the percentages of routed vehicles that fell within each computation time interval for different update frequencies. The percentage of routed vehicles that had a computation time greater than or equal to 0.5 second and less than 1 second had the highest percentage at all update frequencies. Nearly identical percentages were reported for each computation time interval which indicated that the variation in update frequency did not imply impact on the computation time but it affected the impact on the system in terms of the number of affected vehicles. Note that all the vehicles that had a computation time greater than or equal to 2 seconds belonged to group 4. Since these vehicles could not reach their original destinations, new destinations that were reachable nodes in buffer 2 and that were the closest to the original destination were assigned for each vehicle. The search for a new destination was time-consuming but can be limited in future works.

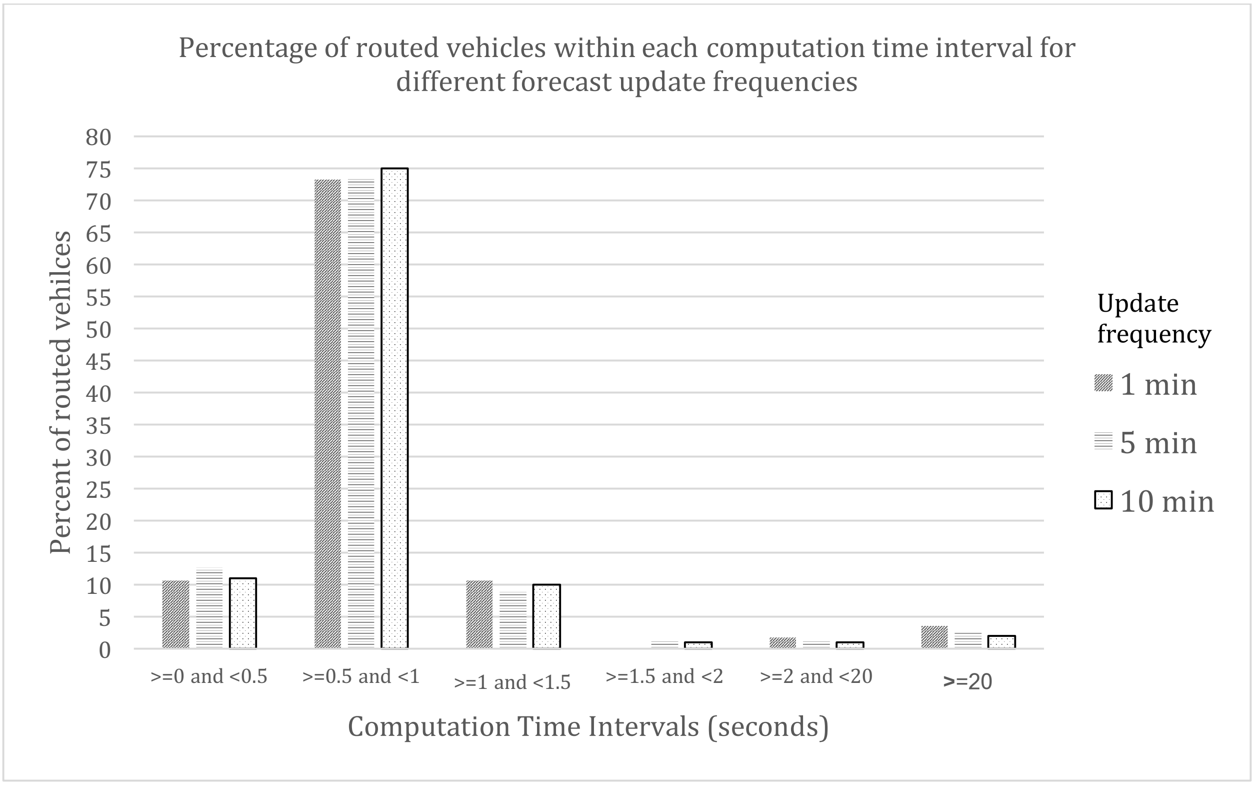


Figure 40 Percentage of routed vehicles within each computation time interval for different forecast update frequencies

4.4 Conclusion

This study proposed a traveler assistance framework for use in a connected vehicle environment and in conjunction with high resolution weather and road flooding prediction systems. The intended use of the framework was for nuisance flooding where neighborhood evacuation was unnecessary. Inputs to the framework included the transportation network with undelayed travel time and maximum delay for each link; predictions of road flooding, including location and timing; and vehicle origins, destinations, intended paths, and departure times. The framework identified vehicles affected by the flood to whom guidance was provided. Based on the affected vehicle's path and departure time and the location and time of the flood, the vehicle was either requested to wait at the origin or received routing assistance in the form of a hyperpath. The hyperpath, computed using the Hyperstar algorithm developed by Bell, Trozzi et al. (2012), was a set of alternative paths that connected the vehicle's origin to its destination while preventing entry to links that are soon-to-be flooded or closed. If the destination of the vehicle was positioned on a link that could not be entered due to the floods, the vehicle was assigned to a new safe destination. If the vehicle could not reach any safe destination, it was instructed to wait at the origin along with the vehicles that were originally positioned on a closed link. Vehicles that departed from a soon-to-be flooded link received a hyperpath to a safe node first to ensure that it exited the flooded link or area as quickly as possible. Then, they received a second hyperpath to resume the trip to their final destination.

The proposed framework was tested with simulated vehicle data on two transportation networks modeled in VISSIM based on the City of Virginia Beach. The tests included evaluating the scalability of the framework to different transportation network sizes and the sensitivity of the results to weather and flood forecasts' update frequency. With fewer updates, more vehicles were rerouted, likely causing more congestion on alternate routes. Higher temporal resolution weather and flood prediction systems can help reduce this effect in the future. The computation time of is influenced by the size of the network; however, the computation time for the larger network and was still reasonable. Thus, further increase in the network size can be tested in the future.

In the experiments, vehicles with soon-to-be flooded or closed destinations had the highest computation times due to the time-consuming search for new safe destinations. To enhance the computation times of these vehicles (and vehicles in group 1 as well), the search for a new destination can be limited and stopped after the evaluation of a maximum number of tentative destinations. Similarly, and to minimize the computation time of vehicles in groups 1 and 2 (vehicles on soon-to-be flooded origins directed to a safe stop first then to the final destination), the user can receive the first hyperpath and start travelling while the second hyperpath is being computed.

The computation times in this study were fairly small (generally less than 1 second, except when needing to search for a new destination). With improvements in the way identifying or searching for a new destination is performed, the overall performance could be improved. Based on the findings thus far, the framework has strong potential for use in the connected vehicle environment.

5 Recommendations

Overall, the research findings were promising and suggested that future implementation of a system like the one investigated here could be successful in a connected vehicle environment. However, a few steps are needed before it would be fully ready for implementation.

As demonstrated in the test of the influence of the weather and flood prediction update frequency, with less frequent updates, more vehicles are affected, leading to some unnecessary re-routing and potential congestion. Greater frequency and spatial localization of the rainfall predictions require future research for extreme rainfall events that assimilates accurate radar precipitation and soil moisture data.

To improve the road flood prediction, greater amounts of data need to be collected. Ideally, the data would be collected on all roads and the flooding observations would be automated, helping to remove some of the bias introduced into the data used for this study. Additionally, with more training data, the model could be trained on specific subsets of flood events to tailor the model to a flood event with specific characteristics (e.g. flash floods).

The routing framework could also be further improved. First, a faster approach is need to select a new safe destination for vehicles with an unreachable destination. An interface may ask a future user for a new destination and/or present the user with options for rejection/selection. Next, the framework could be further evaluated by combining it with microscopic traffic simulation tools to capture the flow variations due to the assignment of vehicles to the new routes, allowing for the determination of benefits in terms of travel time, fuel consumption, emissions, and other performance measures. Future work could also use link specific values of the empty and minimum clearance intervals. Finally, the hyperstar routing algorithm could triggered even after the trip initiation, at intermediate nodes. Since unexpected events are likely to turn an optimal route into a suboptimal one due to incurred delays, generating an updated hyperpath from downstream intermediate nodes to the destination is a technique that continuously searches for better routes. If significant time savings could be acquired, the new hyperpath from the intermediate node to the destination could be presented to the user.

References

Adamovic, M., F. Branger, I. Braud and S. Kralisch (2016). "Development of a data-driven semi-distributed hydrological model for regional scale catchments prone to Mediterranean flash floods." Journal of Hydrology **541**: 173–189.

Bates, P. D., R. J. Dawson, J. W. Hall, M. S. Horritt, R. J. Nicholls, J. Wicks and M. A. A. M. Hassan (2005). "Simplified two-dimensional numerical modelling of coastal flooding and example applications." Coastal Engineering **52**(9): 793-810.

Bell, M. G. (2009). "Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation." Transportation Research Part B: Methodological **43**(1): 97-107.

Bell, M. G., V. Trozzi, S. H. Hosseinloo, G. Gentile and A. Fonzone (2012). "Time-dependent Hyperstar algorithm for robust vehicle navigation." Transportation Research Part A: Policy and Practice **46**(5): 790-800.

Breiman, L. (2001). "Random Forests." Machine Learning **45**(1): 5-32.

Breiman, L., J. H. Friedman, R. A. Olshen and C. J. Stone (1984). Classification and Regression Trees. Belmont, CA, Wadsworth.

Chabini, I. and S. Lan (2002). "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks." IEEE Transactions on intelligent transportation systems **3**(1): 60-74.

Dijkstra, E. W. (1959). "A note on two problems in connexion with graphs." Numerische mathematik **1**(1): 269-271.

Ezer, T. and L. P. Atkinson (2014). "Accelerated flooding along the U.S. East Coast: On the impact of sea-level rise, tides, storms, the Gulf Stream, and the North Atlantic Oscillations." Earth's Future **2**(8): 362–382.

Federal Highway Administration (2017). "How Do Weather Events Impact Roads? ."

Gallien, T. W., B. F. Sanders and R. E. Flick (2014). "Urban coastal flood prediction: Integrating wave overtopping, flood defenses and drainage." Coastal Engineering **91**: 18-28.

Glover, F., D. Klingman and N. Phillips (1985). "A new polynomially bounded shortest path algorithm." Operations Research **33**(1): 65-73.

Hart, P. E., N. J. Nilsson and B. Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths." IEEE transactions on Systems Science and Cybernetics **4**(2): 100-107.

Hoover, D. J., K. O. Odigie, P. W. Swarzenski and P. Barnard (2016). "Sea-level rise and coastal groundwater inundation and shoaling at select sites in California, USA." Journal of Hydrology: Regional Studies.

Horsburgh, J. S., D. G. Tarboton, M. Piasecki, D. R. Maidment, I. Zaslavsky, D. Valentine and T. Whitenack (2009). "An integrated system for publishing environmental observations data." Environmental Modelling & Software **24**(8): 879–888.

Hunter, N. M., P. D. Bates, S. Neelz, G. Pender, I. Villanueva, N. G. Wright, D. Liang, R. A. Falconer, B. Lin, S. Waller, A. J. Crossley and D. Mason (2008). "Benchmarking 2D hydraulic models for urban flood simulations." Proceedings of the Institution of Civil Engineers: Water Management **161**(1): 13-30.

Ireland, G., M. Volpi and G. Petropoulos (2015). "Examining the Capability of Supervised Machine Learning Classifiers in Extracting Flooded Areas from Landsat TM Imagery: A Case Study from a Mediterranean Flood." Remote Sensing **7**(3): 3372–3399.

Lin, L. F., A. M. Ebtehaj, R. L. Bras, A. N. Flores and J. Wang (2015). "Dynamical precipitation downscaling for hydrologic applications using WRF 4D-var data assimilation: Implications for GPM era." Journal of Hydrometeorology **16**(2): 811-829.

Luby, M. and P. Ragde (1989). "A bidirectional shortest-path algorithm with good average-case behavior." Algorithmica **4**(1): 551-567.

Ma, J., D. Fukuda and J.-D. Schmöcker (2013). "A fast node-directed multipath algorithm: Dijkstra-Hyperstar."

Mark, O., S. Weesakul, C. Apirumanekul, S. B. Aroonnet and S. Djordjevic (2004). "Potential and limitations of 1D modelling of urban flooding." Journal of Hydrology **299**(3-4): 284–299.

Mignot, E., A. Paquier and S. Haider (2006). "Modeling floods in a dense urban area using 2D shallow water equations." Journal of Hydrology **327**(1-2): 186–199.

Moftakhari, H. R., A. AghaKouchak, B. F. Sanders, D. L. Feldman, W. Sweet, R. A. Matthew and A. Luke (2015). "Increased nuisance flooding along the coasts of the United States due to sea level rise: Past and future." Geophysical Research Letters **42**(22): 9846–9852.

Murphy, K. P. (2012). Machine learning: a probabilistic perspective, MIT press.

Naghibi, S. A., D. D. Moghaddam, B. Kalantar, B. Pradhan and O. Kisi (2017). "A comparative assessment of GIS-based data mining models and a novel ensemble model in groundwater well potential mapping." Journal of Hydrology **548**: 471–483.

NOAA. (2017). "Daily Summaries Station Details: NORFOLK INTERNATIONAL AIRPORT, VA US, GHCND:USW00013737 | Climate Data Online (CDO) | National Climatic Data Center (NCDC)." Retrieved June 1, 2017, from https://www.ncdc.noaa.gov/cdo-web/datasets/GHCND/stations/GHCND:USW00013737/detail.

NOAA. (2017). "Daily Summaries Station Details: NORFOLK NAS, VA US, GHCND:USW00013750 | Climate Data Online (CDO) | National Climatic Data Center (NCDC)." Retrieved June 1, 2017, from https://www.ncdc.noaa.gov/cdo-web/datasets/GHCND/stations/GHCND:USW00013750/detail.

NOAA. (2017). "Money Point - Station Home Page - NOAA Tides & Currents." Retrieved June 1, 2017, from http://tidesandcurrents.noaa.gov/stationhome.html?id=8518750#info.

NOAA. (2017). "Sewells Point - Station Home Page - NOAA Tides & Currents." Retrieved June 1, 2017, from https://tidesandcurrents.noaa.gov/stationhome.html?id=8638610.

Povak, N. A., P. F. Hessburg, T. C. McDonnell, K. M. Reynolds, T. J. Sullivan, R. B. Salter and B. J. Cosby (2014). "Machine learning and linear regression models to predict catchment-level base cation weathering rates across the southern Appalachian Mountain region, USA." Water Resources Research **50**(4): 2798–2814.

Pyatkova, K., A. S. Chen, S. Djordjevic, D. Butler, Z. Vojinović, Y. A. Abebe and M. Hammond (2015). "Flood impacts on road transportation using microscopic traffic modelling technique."

Ray, R. D. and G. Foster (2016) "Future nuisance flooding at Boston caused by astronomical tides alone." Earth's Future.

Sahoo, S., T. A. Russo, J. Elliott and I. Foster (2017). "Machine learning algorithms for modeling groundwater level changes in agricultural regions of the U.S." Water Resources Research **53**(5): 3878–3895.

Skamarock, W. C. and J. B. Klemp (2008). "A time-split nonhydrostatic atmospheric model for weather research and forecasting applications." Journal of Computational Physics **227**(7): 3465-3485.

Skamarock, W. C., J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, W. Wang and J. G. Powers (2005). A description of the advanced research WRF version 2 National Center For Atmospheric Research Boulder Co Mesoscale and Microscale Meteorology Div.

Smith, R. A. E., P. D. Bates and C. Hayes (2011). "Evaluation of a coastal flood inundation model using hard and soft data." Environmental Modelling & Software **30**: 35–46.

Solomatine, D. P. and A. Ostfeld (2008). "Data-driven modelling: some past experiences and new approaches." Journal of Hydroinformatics **10**(1): 3.

Solomatine, D. P. and Y. Xue (2004). "M5 Model Trees and Neural Networks: Application to Flood Forecasting in the Upper Reach of the Huai River in China." Journal of Hydrologic Engineering **9**(6): 491–501.

Spiess, H. and M. Florian (1989). "Optimal strategies: a new assignment model for transit networks." Transportation Research Part B: Methodological **23**(2): 83-102.

Sweet, W. V. and J. Park (2014). "From the extreme to the mean: Acceleration and tipping points of coastal inundation from sea level rise." Earth's Future **2**(12): 579–600.

Tehrany, M. S., B. Pradhan and M. N. Jebur (2013). "Spatial prediction of flood susceptible areas using rule based decision tree (DT) and a novel ensemble bivariate and multivariate statistical models in GIS." Journal of Hydrology **504**: 69–79.

Tien Bui, D., B. Pradhan, H. Nampak, Q.-T. Bui, Q.-A. Tran and Q.-P. Nguyen (2016). "Hybrid artificial intelligence approach based on neural fuzzy inference model and metaheuristic optimization for flood susceptibilitgy modeling in a high-frequency tropical cyclone area using GIS." Journal of Hydrology **540**: 317–330.

US Department of Commerce, N. O. a. A. A. (2014). "What is nuisance flooding?", from https://oceanservice.noaa.gov/facts/nuisance-flooding.html.

Wagner, D. and T. Willhalm (2007). Speed-up techniques for shortest-path computations. Annual Symposium on Theoretical Aspects of Computer Science, Springer.

Wang, W.-C., K.-W. Chau, C.-T. Cheng and L. Qiu (2009). "A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series." Journal of Hydrology **374**(3): 294–306.

Wang, Z., C. Lai, X. Chen, B. Yang, S. Zhao and X. Bai (2015). "Flood hazard risk assessment model based on random forest." Journal of Hydrology **527**: 1130–1141.

Yang, T., A. A. Asanjan, E. Welles, X. Gao, S. Sorooshian and X. Liu (2017). "Developing reservoir monthly inflow forecasts using artificial intelligence and climate phenomenon information." Water Resources Research **53**(4): 2786–2812.

Yang, T., X. Gao, S. Sorooshian and X. Li (2016). "Simulating California reservoir operation using the classification and regression-tree algorithm combined with a shuffled cross-validation scheme." Water Resources Research **52**(3): 1626–1651.

Appendix A: Framework Pseudocode

```
Buffer generation main class {
    Read the Network input file (link id, from node, to node)
    Read the input_FloodedLinks file (list of soon-to be flooded and closed links)

    Find the nodes in Flood Boundary(FB) {
        For (all links in input_FloodedLinks) {
            Add the origin and destination to the FB array;
        }
        For (all nodes in FB array) {
            If (node is not an origin or destination to a safe link)
                Remove node from FB array;
        }
        Remove duplicates in FB array;
        Sort FB array in ascending order;
    }
    Output: list of nodes in FB array

    Find the nodes in Buffer 1{
        For (all safe links) {
            If (origin is in FB && destination not in FB)
                Add destination in buffer 1 array;
            If (destination is in FB && origin not in FB)
                Add origin in buffer 1 array;
            If (origin is in FB && destination not in FB) or (destination is in FB && origin not
            in FB)
                Add link to Link_buffer array;
        }
        Remove node in FB from buffer 1 array;
        Remove duplicates in buffer 1 array;
        Remove duplicates in Link_buffer array;
        Sort nodes in buffer 1 array in ascending order of ID;
    }
    Output: list of nodes in Buffer 1;

    Find the nodes in Buffer 2 {
        For (all safe link && links not in Link_buffer array) {
            If (origin is in buffer 1 && destination not in buffer 1)
                Add destination in buffer 2 array;
            If (destination is in buffer 1 && origin not in buffer 1)
                Add origin in buffer 2 array;
        }
        Remove nodes in FB and in buffer 1 from the buffer 2 array;
        Remove duplicates in buffer 2 array;
        Sort buffer 2 array in ascending order;
    }
    Output: list of nodes in Buffer 2;

}
```

```
Vehicle analysis main class {

        Input: Raw vehicle file (number of links in the path for each vehicle; Vehicle raw data from vissim)
        Input: FloodedLinks file
        Input: FloodedNodesLinks_time (list of time at which each flooded link is closed or “t3”)
        Input: Network connectivity (link id, from node, to node)
        Input: CL (duration of the minimum clearance and additional clearance interval in min)

        For each vehicle v {
                v.group(input_FloodedLinks, FloodedNodesLinks_time, CL);
                v.origin();
                v.destination();
                v.start_time();
        }
}

method group (input: FloodedLinks, FloodedLinks_time, CL) {

        group=6;                                //Vehicle unaffected

        if (intermediate link in the path is soon-to-be flooded or closed && expected arrival at this link >= t3 – CL)
                group=5;                        //Vehicle with intermediate link soon-to-be flooded or closed

        if (first link is soon-to-be flooded && (t3-CL <= trip start time < t3))
                group=2;                        //Vehicle with origin on soon-to-be flooded origin

        if (last link is soon-to-be flooded or closed && expected time of arrival at destination >= t3- CL)
                group=4;                        //Vehicle with destination on soon-to-be or closed link

        if (group 2 && group 4)
                group=1;                        //Vehicle with origin on soon-to-be flooded link and
                                                destination on soon-to-be or closed link

        if (first link in path is closed && start_time >= t3)
                group=3;                        //Vehicle with origin on closed link


        return group;
}

method origin() {
        returns start node of first link in path;
}

method destination() {
        returns end node of last link in path;
}

method start_time() {
        returns time of entry at first link in path;
}
```

```
Process main class {

Read the input_vehicle file;

For each vehicle v (group, origin, destination, start_time): {
max_computation_time=0;

        If (group =1) {
                time1(1) = start_time;
                Find the closest node in buffer 2 to destination;
                New destination = closest node in buffer 2;
                Find closest node in buffer 1 to new destination;
                Run partitioning algorithm to the selected node in buffer 1;
                Run hyperstar between origin and node in buffer 1;

                while (hyperpath is empty && next closest node in buffer 1 exists) {
                        Find the next closest node in buffer 1 to the destination;
                        Run the partitioning algorithm to the selected node in buffer 1;
                        Run hyperstar between the origin and the selected node in buffer 1;
                }

                if (hyperpath is empty) {
                        add the vehicle to group 3;
                }

                else {
                        add the vehicle to group 1;

                        time1(1)=end;
                        if (time1(1)> max_computation_time)
                                max_computation_time=time1(1);

                        output = Select_path method (v);
                        add output to ROUTING file;
                        Get actual time of arrival at buffer 1;

                        If (node in buffer 1 == node in buffer 2)
                                Vehicle has reached destination → stop;

                        else {
                                time1(2) = start;
                                set start_time = actual time of arrival at buffer 1;
                                Run the partitioning algorithm to destination at buffer 2;
                                Run hyperstar between node in buffer 1 and node in buffer 2

                                if (hyperpath empty)
                                        Vehicle is blocked and has to wait at buffer 1; add to subgroup 1b;

                                else {
                                        time1(2)=end;
                                        if (time1(2)> max_computation_time)
                                                max_computation_time=time1(2);

                                        output= Select_path method (v);
                                        add output to ROUTING file;
                                }
                        }
                }
        }

        If (group =2 ) {
                time2(1) =start_time;
                Find the closest node in buffer 1 to destination;
                Run the partitioning algorithm to the selected node in buffer 1;
                Run hyperstar between the origin and the node in buffer 1;

                while (hyperpath is empty && next closest node in buffer 1 exists) {
                        Find the next closest node in buffer 1 to the destination;
                        Run the partitioning algorithm to the selected node in buffer 1;
                Run hyperstar between the origin and the selected node in buffer 1;
            }

            if (hyperpath is empty) {
                add the vehicle to group 3;
            }

            else {
                add the vehicle to group 2;

                time2(1)=end
                if (time2(1)> max_computation_time)
                    max_computation_time=time2;

                output = Select_path method (v)
                add output to ROUTING file
                Get actual time of arrival at buffer 1;

                If (node in buffer 1 == destination)
                    Vehicle has reached destination → stop;

                else {
                    time2(2)=start;
                    set start_time = actual time of arrival at buffer 1;
                    Run the partitioning algorithm to the destination;
                    Run hyperstar between node in buffer 1 and destination;

                    if (hyperpath empty)
                        Vehicle is blocked and has to wait at buffer 1; add to subgroup 2b;

                    else {
                        time2(2)=end;
                        if (time2(2)> max_computation_time)
                            max_computation_time=time(2);

                        output= Select_path method (v);
                        add output to ROUTING file;
                    }
                }
            }
        }

        If (group =3) {
            Add the vehicle to group3;
        }

        If (group 4){
            time4=start_time;
            Find closest node in buffer 2 to destination;
            New destination = closest node in buffer 2;
            Run the partitioning algorithm to the selected node in buffer 2;
            Run hyperstar between origin and the node in buffer 2;

            while (hyperpath empty && next closest node in buffer 2 exists)
                Find the next closest node in buffer to the destination;
                Run the partitioning algorithm to the selected node in buffer 2;
                Run hyperstar between the origin and the selected node in buffer 2;
            }

            if (hyperpath is empty) {
                add the vehicle to group 3;
            }

            else {
                add the vehicle to group 4;
                time4=end;
                if (time4> max_computation_time)
                    max_computation_time=time1;
                output = Select_path method (v);
                add output to ROUTING file;
            }
    }

    If (group 5) {
            time5=start_time;
            Run partitioning algorithm to destination;
            Run hyperstar between origin and destination;

            if (hyperpath empty)
                add the vehicle to group 3;
            }

            else
                add the vehicle to group 5;
                time5=end;
                if (time5> max_computation_time)
                    max_computation_time=time1;
                output = Select_path method (v);
                add output to ROUTING file;
            }
    }

    If (group =6) {
            Add the vehicle to Outputfile_Group6;
    }

Get max computation time
}

select_path method (vehicle) {
    Scan the hyperpath of vehicle v;
    Scan the actual network (with actual data);

    end= origin of v;
    time to destination = start time of v;
    path = empty;
    while(end is not destination){
            find the links in hyperpath exiting node end;
            select a link (generate a random number) and add it to the path;
            time to destination = time to destination + travel time of selected link;
            end= end node of the selected link;
    }

    return array of String (index 0: vehicle id, index1: time of arrival at
            destination and index 2: path to destination);
}
```

Appendix B: VISSIM COM API: VBA Simulation Script with Varying Vehicle Inputs

```
' SIMULATION 1
'=====================================================================
'global variables
'=====================================================================
Dim Vis, Sim, vnet
Dim VI_number
Dim new_volume
'=====================================================================
'main program
'=====================================================================
Set Vis = CreateObject("VISSIM.Vissim")
Set Sim = Vis.Simulation
Set vnet = Vis.Net
Vis.LoadNet("C:\...(Path to .net file) “) 'Load Net
Vis.LoadLayout("C:…(Path to .layx file) ") ' Load layout

'=====================================================================
'SIMULATION 1_Base
'=====================================================================
'Configure the Simulation
Vis.Simulation.AttValue("RandSeed") = 21
Vis.Simulation.AttValue("RandSeedIncr") = 3
Vis.Simulation.AttValue("NumRuns") = 5

' VI = Vehicle Input
VI_number1 = 1
VI_number2 = 2
VI_number3 = 3
VI_number4 = 4
VI_number5 = 5
(.... Continued until reaching number if Vehicle input bars in VISSIM model)

'Define Volume as variable

' vehicles per hour
new_volume1 = 100
new_volume2 = 75
new_volume3 = 75
new_volume4 = 75
new_volume5 = 75
(.... Continued until reaching number if Vehicle input bars in VISSIM model)
```

```
‘ Vehicle input final volume
new_volume11 = new_volume1 * 1
new_volume21 = new_volume2 * 1
new_volume31 = new_volume3 * 1
new_volume41 = new_volume4 * 1
new_volume51 = new_volume5 * 1
(.... Continued until reaching number if Vehicle input bars in VISSIM model)

‘ Allocating volume from previous step to Vehicle input in VISSIM
Vis.Net.VehicleInputs.ItemByKey(VI_number1).AttValue("Volume(1)") = new_volume11
Vis.Net.VehicleInputs.ItemByKey(VI_number2).AttValue("Volume(1)") = new_volume21
Vis.Net.VehicleInputs.ItemByKey(VI_number3).AttValue("Volume(1)") = new_volume31
Vis.Net.VehicleInputs.ItemByKey(VI_number4).AttValue("Volume(1)") = new_volume41
Vis.Net.VehicleInputs.ItemByKey(VI_number5).AttValue("Volume(1)") = new_volume51
(.... Continued until reaching number if Vehicle input bars in VISSIM model)

' Set maximum speed:
Vis.Simulation.AttValue("UseMaxSimSpeed") = True
'run the simulation
Vis.Simulation.RunContinuous

'==========================================================================
'SIMULATION 2_+10%
'==========================================================================
'Configure the Simulation
Vis.Simulation.AttValue("RandSeed") = 22
Vis.Simulation.AttValue("RandSeedIncr") = 3
Vis.Simulation.AttValue("NumRuns") = 5

' VI = Vehicle Input
VI_number1 = 1
VI_number2 = 2
VI_number3 = 3
VI_number4 = 4
VI_number5 = 5
(.... Continued until reaching number if Vehicle input bars in VISSIM model)

'Define Volume as variable

' vehicles per hour
new_volume1 = 100
new_volume2 = 75
new_volume3 = 75
new_volume4 = 75
```

```
new_volume5 = 75
(.... Continued until reaching number if Vehicle input bars in VISSIM model)
‘ Vehicle input final volume
new_volume12 = new_volume1 * 1.1
new_volume22 = new_volume2 * 1.1
new_volume32 = new_volume3 * 1.1
new_volume42 = new_volume4 * 1.1
new_volume52 = new_volume5 * 1.1

‘ Allocating volume from previous step to Vehicle input in VISSIM
Vis.Net.VehicleInputs.ItemByKey(VI_number1).AttValue("Volume(1)") = new_volume12
Vis.Net.VehicleInputs.ItemByKey(VI_number2).AttValue("Volume(1)") = new_volume22
Vis.Net.VehicleInputs.ItemByKey(VI_number3).AttValue("Volume(1)") = new_volume32
Vis.Net.VehicleInputs.ItemByKey(VI_number4).AttValue("Volume(1)") = new_volume42
Vis.Net.VehicleInputs.ItemByKey(VI_number5).AttValue("Volume(1)") = new_volume52
(.... Continued until reaching number if Vehicle input bars in VISSIM model)

' Set maximum speed:
Vis.Simulation.AttValue("UseMaxSimSpeed") = True
'run the simulation
Vis.Simulation.RunContinuous
```

(Continued until SIMULATION 11_ +100%)

Appendix C: R-Studio Script to Format Raw VISSIM Output

```
setwd("C:/… Location of data in computer, and were all output files will be placed ")

library(plyr)

library(stringr)

# Read full data

FullnetworkRaw <- read.csv("FullNetwork_Simulation.csv",header = FALSE)

# Add column headers

colnames(FullnetworkRaw) <-
c("SimRunID","Time","No_","veh","VehType","Trav","Delay")

# Add Helper Column

FullnetworkRaw$Helper <- paste(FullnetworkRaw$SimRunID,FullnetworkRaw$No_,sep =
"_")

##############################################################
################

# Count if Help = Helper

a <- as.data.frame(table(FullnetworkRaw$Helper))

colnames(a) <- c("Helper","Q")

# Re-split Helper Column to add Simulation & Link variables

a2 <- as.data.frame(str_split_fixed(a$Helper, "_", 2))

a$SimRun <- a2$V1

a$Link <- a2$V2

# Re-arrange columns

a <- a[c("SimRun", "Link", "Helper", "Q")]

#----------------------------------------------------------------------------------------------------------
#

# Average if, for all unique values of Help/Helper Travel Time

b <- as.data.frame(sapply(split(FullnetworkRaw$Trav,FullnetworkRaw$Helper), mean))

colnames(b) <- c("TravelTime")

b$Helper <- a$Helper
```

```
#---------------------------------------------------------------------------------------------------------------
#
# Average if, for all unique values of Help/Helper Travel Time
c <- as.data.frame(sapply(split(FullnetworkRaw$Delay,FullnetworkRaw$Helper), mean))
colnames(c) <- c("Delay")
c$Helper <- a$Helper
# Merge Final Data
d <- merge(a,b,by=c("Helper"))
Final <- merge(d,c,by=c("Helper"))
# Include FreeFlowTravelTime Variable Free Flow Travel Time -> FFTT
Final$FreeFlowTravelTime <- (Final$TravelTime - Final$Delay)
# Write to CSV
write.csv(Final,"FinalData.csv")
```

Appendix D: R-Studio Script to Perform Non-Linear Regression and Obtain Travel Time Functions

```
setwd("C:/... Location of data in computer, and were all output files will be placed ")
library(latticeExtra)
library(plyr)

FullData <- read.table("FinalData.txt", header = TRUE)

# Subset Data by Link
Link1 <- FullData[ which(FullData$Link=='1'), ]
Link2 <- FullData[ which(FullData$Link=='2'), ]
Link3 <- FullData[ which(FullData$Link=='3'), ]
Link4 <- FullData[ which(FullData$Link=='4'), ]
(.... Continued for all data links)

# Identify mean free flow travel time variable FFT
FFT_1<- mean(Link1$FreeFlowTravelTime)
FFT_2<- mean(Link2$FreeFlowTravelTime)
FFT_3<- mean(Link3$FreeFlowTravelTime)
FFT_4<- mean(Link4$FreeFlowTravelTime)

# Bind Free Flow Data
FreeFlowAverages <- rbind(
  FFT_1,
  FFT_2,
  FFT_3,
  FFT_4,
(.... Continued for all data links)
...)

# Export to csv
write.csv(FreeFlowAverages, "FreeFlowAverages.csv")

# Identify Travel Time Variable
TT_1 <- Link1$TravelTime
TT_2 <- Link2$TravelTime
TT_3 <- Link3$TravelTime
TT_4 <- Link4$TravelTime
(.... Continued for all data links)

# Identify Flow Variable
Q_1 <-Link1$Q
Q_2 <-Link2$Q
Q_3 <-Link3$Q
```

```
Q_4 <-Link4$Q
(.... Continued for all data links)

# Identify Delay Variable
D_1 <-Link1$Delay
D_2 <-Link2$Delay
D_3 <-Link3$Delay
D_4 <-Link4$Delay
(.... Continued for all data links)


# Fit nonlinear model [t = FFT + X*(Q)^2]
Est_TT_1 <-  lm(TT_1~-1+I(Q_1^2), offset=rep(FFT_1, length(Q_1)))
Est_TT_2 <-  lm(TT_2~-1+I(Q_2^2), offset=rep(FFT_2, length(Q_2)))
Est_TT_3 <-  lm(TT_3~-1+I(Q_3^2), offset=rep(FFT_3, length(Q_3)))
Est_TT_4 <-  lm(TT_4~-1+I(Q_4^2), offset=rep(FFT_4, length(Q_4)))
(.... Continued for all data links)

# Fit nonlinear model [Delay = Zero Delay + X*(Q)^2]
Est_D_1 <-  lm(D_1~-1+I(Q_1^2))
Est_D_2 <-  lm(D_2~-1+I(Q_2^2))
Est_D_3 <-  lm(D_3~-1+I(Q_3^2))
Est_D_4 <-  lm(D_4~-1+I(Q_4^2))
(.... Continued for all data links)

# Extract Model Estimates
Link1Coef <- Est_TT_1$coefficients
Link2Coef <- Est_TT_2$coefficients
Link3Coef <- Est_TT_3$coefficients
Link4Coef <- Est_TT_4$coefficients
(.... Continued for all data links)

# Bind Coefficient Data
NetworkTTCoefficients <- rbind(
  Link1Coef,
  Link2Coef,
  Link3Coef,
  Link4Coef,
(.... Continued for all data links)
...)

# Export to csv
write.csv(NetworkTTCoefficients, " NetworkTTCoefficients.csv")

# R-Square Values
Link1Rsquare <- summary(Est_TT_1)$r.squared
```

```
Link2Rsquare <- summary(Est_TT_2)$r.squared
Link3Rsquare <- summary(Est_TT_3)$r.squared
Link4Rsquare <- summary(Est_TT_4)$r.squared
(.... Continued for all data links)

# Bind Rquare Data
NetworkRSquare <- rbind(
  Link1Rsquare,
  Link2Rsquare,
  Link3Rsquare,
  Link4Rsquare,
(.... Continued for all data links)
…)

# Export to csv
write.csv(NetworkRSquare, " NetworkRSquare.csv")

# Adjusted R-Square Values
Link1AdjRsquare <- summary(Est_TT_1)$adj.r.squared
Link2AdjRsquare <- summary(Est_TT_2)$adj.r.squared
Link3AdjRsquare <- summary(Est_TT_3)$adj.r.squared
Link4AdjRsquare <- summary(Est_TT_4)$adj.r.squared
(.... Continued for all data links)

# Bind Adjusted Rquare Data
NetworkAdjRSquare <- rbind(
  Link1AdjRsquare,
  Link2AdjRsquare,
  Link3AdjRsquare,
  Link4AdjRsquare,
(.... Continued for all data links)
…)

# Export to csv
write.csv(NetworkAdjRSquare, " NetworkAdjRSquare.csv")

# Delay Coefficients
Link1DelayCoef <- Est_D_1$coefficients
Link2DelayCoef <- Est_D_2$coefficients
Link3DelayCoef <- Est_D_3$coefficients
Link4DelayCoef <- Est_D_4$coefficients
(.... Continued for all data links)

# Bind Delay Coefficients
NetworkDelayCoefficients <- rbind(
  Link1DelayCoef,
  Link2DelayCoef,
  Link3DelayCoef,
  Link4DelayCoef,
(.... Continued for all data links)
…)

# Export to csv
write.csv(NetworkDelayCoefficients, " NetworkDelayCoefficients.csv")

# Delay R-Square
Link1DelayRsquare <- summary(Est_D_1)$r.squared
Link2DelayRsquare <- summary(Est_D_2)$r.squared
Link3DelayRsquare <- summary(Est_D_3)$r.squared
Link4DelayRsquare <- summary(Est_D_4)$r.squared
(.... Continued for all data links)

# Delay R Squared Bind
NetworkDelayRSquare <- rbind(
  Link1DelayRsquare,
  Link2DelayRsquare,
  Link3DelayRsquare,
  Link4DelayRsquare,
(.... Continued for all data links)
…)

# Export to csv
write.csv(NetworkDelayRSquare, " NetworkDelayRSquare.csv")

# Delay Adj R-Square
Link1DelayAdjRsquare <- summary(Est_D_1)$adj.r.squared
Link2DelayAdjRsquare <- summary(Est_D_2)$adj.r.squared
Link3DelayAdjRsquare <- summary(Est_D_3)$adj.r.squared
Link4DelayAdjRsquare <- summary(Est_D_4)$adj.r.squared

# Bind Delay Adjustred-Square
NetworkDelayAdjRSquare <- rbind(
  Link1DelayAdjRsquare,
  Link2DelayAdjRsquare,
  Link3DelayAdjRsquare,
  Link4DelayAdjRsquare,
(.... Continued for all data links)
…)

# Export to csv
write.csv(NetworkDelayAdjRSquare, " NetworkDelayAdjRSquare.csv")
```