Spring 2015

Modeling supply chain interdependent critical infrastructure systems

Varun Ramachandran

Department: Engineering Management and Systems Engineering

MODELING SUPPLY CHAIN INTERDEPENDENT CRITICAL INFRASTRUCTURE SYSTEMS

by

VARUN RAMACHANDRAN

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

ENGINEERING MANAGEMENT

2015

PUBLICATION DISSERTATION OPTION

This dissertation has been prepared in the form of three papers for publication. The first paper was submitted to *The Natural Hazards Review Journal* and starts on page 11 through to page 47. The second paper was submitted to *The International Journal of Geospatial and Environmental Research* and starts on page 48 through to page 78. The third paper was submitted to *The Computer Environment and Urban Systems Journal* and starts on page 79 through to page 108.

ABSTRACT

While strategies for emergency response to large-scale disasters have been extensively studied, little has been done to map medium- to long-term strategies capable of restoring supply chain infrastructure systems and reconnecting such systems from a local urban area to national supply chain systems. This is, in part, because no comprehensive, data-driven model of supply chain networks exists. Without such models communities cannot re-establish the level of connectivity required for timely restoration of goods and services. This dissertation builds a model of supply chain interdependent critical infrastructure (SCICI) as a complex adaptive systems problem. It defines model elements, data needs/element, the interdependency of critical infrastructures, and suggests metrics for evaluating success. Previous studies do not consider the problem from a systematic view and therefore their solutions are piecemeal, rather than integrated with respect to both the model elements and geospatial data components. This dissertation details a methodology to understand the complexities of SCICI within a real urban framework (St. Louis, MO). Interdependencies between the infrastructures are mapped to evaluate resiliency and a framework for quantifying interdependence is proposed. In addition, this work details the identification, extraction and integration of the data necessary to model infrastructure systems

ACKNOWLEDGMENTS

First and foremost, I want to thank my advisor, Dr. Suzanna Long for giving me the opportunity to work on this project and allowing me to pursue my dream. Her caring nature, guidance, patience and above all knowledge helped me conduct and complete this research. Next, I would like to thank Dr. Tom Shoberg for his continuous effort in making me excel. Dr. Shoberg also provided me with different ideas that moved the research forward. I would also like to thank him for making me a better technical writer. Finally, he created the datasets for this work and in doing so, allowed me to complete my research in a timely fashion. I would like to thank Dr. Steven Corns for teaching me the nuances of research and various modeling techniques that helped make this research possible. I grateful for my committee members Dr. Ruwen Qin, Dr. Brian Smith, and Dr. Curt Elmore for taking time to review the dissertation and giving me valuable inputs during the process of this research. I must express my gratitude to the United States Geological Survey (USGS) for funding this research.

I would like to thank both faculty and staff within Missouri S&T and Department of Engineering Management and Systems Engineering for their support. I am deeply indebted to Tisha for always being there for me and helping me through the tough times. I would like to thank my research colleagues Liz, Sean, and Cory. They were always there when I needed them. I would also like to thank my friends both here in Rolla and at home for supporting me throughout my studies.

Finally, I would like to thank my mother and father for all of their support and love. I could not have asked for anything more from them, and would like to dedicate this dissertation to them and hopefully I have made them proud.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTRODUCTION

1.1. BACKGROUND

In modern society, the basis of livelihood is determined by the availability and reliability of a nationally critical infrastructure. This infrastructure should include transportation networks, electrical networks, a water system, communication networks, banking and finance sectors, emergency services and so forth. (Rinaldi et al., 2001; Little, 2003). These infrastructures form an over-arching net that covers the normal, everyday activities of a society. Each individual infrastructure includes numerous interaction points and disturbances that can cascade very quickly bringing the entire system to a standstill. The multiple layers of an infrastructures system are interweaved, this restoration must incorporate the interdependent nature of infrastructures; it cannot be looked at as a stand-alone problem. Table 1 lists the types of disturbances that can affect an infrastructure.

Table 1 Disturbances within an infrastructure system

Disturbances	**Factor**
Natural Hazard	Earthquakes, floods, tornadoes, hurricanes, volcanic activities (McEntire, 2004)
Man-Made Hazard	Terrorism, war, mismanagement (Hollman et al., 2007)
Technical Failures	Design faults, maintenance issues, unskilled workers (Wen et al., 2009)
Weather Related	Extreme winds, snow, ice, sleet (Helbing et al. 2006)

The vulnerability and the importance of these infrastructures has long been recognized. In the Executive order 13010, July 15, 1996, [Clinton, 1996] stated:

> Certain national infrastructures are so vital that their incapacity or destruction would have a debilitating impact on the defense or economic security of the United States. These critical infrastructures include telecommunications, electrical power systems, gas and oil storage and transportation, banking and finance, transportation, water supply systems, emergency services (including medical, police, fire, and rescue), and continuity of government. These are the foundations of our prosperity, enablers of our defense, and the vanguard of our future. They empower every element of our society. There is not more urgent priority that assuring the security, continuity, and availability of our critical infrastructures...

To protect critical infrastructure against disruptions, a better understanding of the behavior of each component within a critical infrastructure is necessary. Once, the behavior of each component is understood the internal interaction mechanisms among the different components of critical infrastructure needs to be understood. Since, the related historic data is incomplete and not freely available, and real-world physical experiments are expensive there is a need to analyze the critical infrastructure and its interdependencies by using computer modeling and simulation.

Modeling restoration of a critical infrastructure presents several challenge. There is a need to incorporate ideas and tools from a wide spectrum of research areas, including simulation-based optimization, structural engineering, human behavior modeling, geographic information systems (GIS), and supply chain management. A number of studies have been conducted in the past on either disaster management or facility locations following natural or man-made disasters. Little has been done, however that considers medium or long-term restoration strategies that are capable of reconnecting urban areas with national supply chain infrastructure systems. Without a comprehensive, data-driven model of a strategic supply chain infrastructure, communities cannot re-establish the level of vibrant connectivity required for the timely restoration of goods and services.

A systematic approach to identifying the interdependency between a critical infrastructure and the restoration after a perturbation has not been undertaken (OHS, 2002). Fragility increases as a systems complexity increases. The system will also have more sub-systems associated with it. These sub-systems will most likely be inter-dependent on each other for their functioning and, thus, they will have an unpredictable behavior. In today's increasingly interconnected infrastructure networks, the probability that a perturbation in one network will affect the functioning of other systems is quite high (McEntire, 2004; Lecomte, 1998; Mills, 2005). The challenges related to identifying, understanding, and analyzing the interdependent nature of critical infrastructure is magnified by the wide breadth and complexity of the infrastructure's system and related factors. These factors include social, societal, political, technical, economic, legal, and security concerns.

This research proposes a framework for restoration of critical infrastructure in the aftermath of a perturbation. A methodology was developed for understanding the complexities of the system and was validated against a real-world scenario. Next, an evaluation of all the data the publically available was performed to understand and find out the data required for this research. Finally, with the use of the publically available data interdependencies between the different infrastructures was mapped to understand the complexity of the real-world system. The framework that will thus be developed will be scalable across regions and extreme event and will be a plug-and-play model.

1.2. LITERATURE REVIEW

The literature review reveals that the bulk of the research conducted in the field of disaster management or restoration after an extreme event deals with either models focused on facility location, inventory management, resource distribution strategies, or on estimation of short-term resource requirements after an extreme event. The literature that is available is infrastructure-specific and only looks at the problem of protecting a complex and interdependent infrastructure system from a single infrastructure stand-point. A number of organizations, institutions, and universities have focused their research on the critical infrastructure protection like the Department of Homeland Security [DHS], Department of Energy [DOE], and Sandia National Laboratories.

Many researchers have looked at a single infrastructure only and studied the reliability and vulnerability (Adachi, and Ellingwood, 2008; Liu et al., 2005; Davidson et al., 2003). Little sufficient data is available when examining critical infrastructures. A probabilistic method is used to estimate how the infrastructure will react (Lewis et al., 1979; Apostolakis, 2004; Pate-Cornell, 2001). Liu et al. (2005) used statistical regression models combined with historical data to analyze the damage of an earthquake on the power supply. The problem with this type of work is that it does not take into account the multiple network topology and the models thus created have a lot of assumptions associated with it. Koutsourelakis (2010), and Ellingwood and Kinali (2009) look at the vulnerability of built environment when subjected to an earthquake. Distributed

Engineering Workstation (DEW, 2006) is being used to analyze and identify interdependencies from an electrical power systems standpoint and uses graph theory to create the model.

Several notable studies have been conducted by Holguín-Veras et al. (2012), who examined the allocation of resources after hurricane Katrina. Akkihal (2006) developed an algorithm that was focused on the location of distribution centers for non-perishable supplies. Duran et al. (2011), Balcik and Beamon (2008), Ozbay and Ozguven (2007), and Jaller et al. (2007) created techniques that can be used to review levels of disaster relief supplies by considering either a stochastic approach (looking at resource requirement patterns), or used mathematical models to do the same. An operations research (OR) perspective is provided by a number of papers including Altay and Green (2006) where they considered disaster operations management, Rawls and Turnquist (2010) developed an OR tool for pre-positioning emergency supplies before an extreme event, Mete and Zabinsky (2010) created an optimization technique for medical supply location and distribution, a study by Simpson and Hancock (2009) focused on an OR based research that has been done in this field in the last fifty years. The disaster problem's attributes must be understood before theories and algorithms can be applied to disaster operations. The comprehensive model developed during this research can be used to examine preparedness, planning, response, and recovery activities.

Another problem while focusing on infrastructure rebuilding after an extreme event is related to the involvement of multiple organizations including both private and public entities working together. This leads to managerial confusions and ambiguity as to who is in-charge, and what are the responsibilities (Altay & Green, 2006). Public entities have problems within themselves such as ill-defined goals, being authoritative, and have strong political connections (Gass, 1994). Most of the literature that is available does not examine this problem; it ignores it. The handoffs between public and private entities needs to be dealt with seamlessly for optimizing the recovery process. In the literature that is available only a few papers come close to addressing these issues. Gass (1994) presents a decision making methodology advocating for the decision power to be vested in the hands of a select group of people based on an algorithm.

Several of the most advanced critical infrastructure modeling techniques use agent based modeling. Agent-based Infrastructure Modelling and Simulation (AIMS, 2007) was developed at the University of New Brunswick It is used to simulate and model the survivability of a critical infrastructure in Canada. This model did take interdependencies into consideration, little real-data, however was used in this research. The Critical Infrastructure Modelling System (CIMS, 2006) was developed by Idaho National Laboratories (INL), uses geospatial information and performs 'what-if' analysis. The primary problem with this model is that it does not consider the interdependency that exists between critical infrastructures.

1.3. RESEARCH OBJECTIVES

This work was focused on the medium- or long-term restoration strategies that are capable of reconnecting urban areas with national supply chain infrastructure systems. Without a comprehensive, data-driven model of supply chain networks, communities cannot explore strategies to re-establish the level of vibrant connectivity required for a timely restoration of goods and services. This project approached modeling Supply Chain Interdependent Critical Infrastructure (SCICI) in the wake of a large-scale disaster and defines model elements, data needs per element, metrics for success, system modeling, and interdependency mapping. By focusing on SCICI this research can directly map the impact of perturbations on the infrastructure identified as critical by the United States by the Department of Homeland Security (Department of Homeland Security, 2009) as well as related infrastructure elements required for socioeconomic growth and livable communities. The goal is to better acquire, understand, analyze, and simulate SCICI in the context various disasters and develop decision making tools, which can help policy makers and infrastructure service providers to get back to normalcy and to minimize the down time. Table 2 summarizes the objectives of this research and also lists the techniques involved in conducting this research.

Table 2 Objectives and Techniques used to conduct this Research

Objective	**Techniques/Requirements**
Use publically available data for creating a graph model of SCICI	Use graph theory to combine Geospatial data
Map Interdependency	Agent Based Modeling & Simulation, GIS analysis
Propose a methodology for understanding SCICI	Utilize acquired data to analyze and find data trends
Optimize restoration	Use ABMS optimization techniques

REFERENCES

AIMS: Ulieru, M. (2007). Design for resilience of networked critical infrastructures. In Digital EcoSystems and Technologies Conference, 2007. DEST'07. Inaugural IEEE-IES (pp. 540-545). IEEE.

Adachi, T., Ellingwood, B., (2008). Serviceability of earthquake-damaged systems: effects of electrical power availability and back-up systems on system vulnerability. Reliab Eng Syst Safety, 93(1):78–88.

Akkihal, A. R. (2006). Inventory pre-positioning for humanitarian operations, Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA

Altay, N., & Green III, W. G. (2006). OR/MS research in disaster operations management. European Journal of Operational Research, 175(1), 475-493.

Apostolakis, G.E., (2004). How useful is quantitative risk assessment? Risk Analysis 24:515–20.

Balcik, B., Beamon, B. M. (2008). Facility location in humanitarian relief. International Journal of Logistics, 11(2), 101-121.

CIMS: Dudenhoeffer, D. D., Permann, M. R., & Manic, M. (2006). CIMS: A framework for infrastructure interdependency modeling and analysis. InProceedings of the 38th conference on winter simulation (pp. 478-485). Winter Simulation Conference.

Davidson, R. A., Liu, H., Sarpong, I., Sparks, P., and Rosowsky, D. V., (2003). Electric power distribution system performance in Carolina hurricanes. Nat. Hazards Rev. 4(1):36–45.

Duran, S., Gutierrez, M. A., and Keskinocak, P. (2011). Pre-positioning of emergency items worldwide for CARE International. Interfaces, 41(3), 223–237.

Ellingwood, B. R., & Kinali, K. (2009). Quantifying and communicating uncertainty in seismic risk assessment. Structural Safety, 31(2), 179-187.

Gass, S.I., (1994). Public sector analysis and operations research/ management science. In: Pollock, S.M., Rothkopf, M.H., Barnett, A. (Eds.), Handbooks in OR & MS: Operations Research and the Public Sector. Elsevier Science Publishers, Amsterdam, pp. 23–46.

Helbing, D., Ammoser, H., & Kühnert, C. (2006). Disasters as extreme events and the importance of network interactions for disaster response management. In Extreme events in nature and society (pp. 319-348). Springer Berlin Heidelberg.

Holguín-Veras, J., Jaller, M., Van Wassenhove, L. N., Pérez, N., & Wachtendorf, T. (2012). On the unique features of post-disaster humanitarian logistics. Journal of Operations Management, 30(7), 494-506.

Hollman, J. A., Marti, J. R., Jatskevich, J., & Srivastava, K. D. (2007). Dynamic islanding of critical infrastructures: a suitable strategy to survive and mitigate extreme events. International Journal of Emergency Management, 4(1), 45-58.

Jaller, M., Ukkusuri, S., and Holguín-Veras, J. (2007). A stochastic humanitarian inventory model for fixed lifetime goods for disaster planning. INFORMS Annual Meeting, Hanover, MD.

Koutsourelakis, P. S. (2010). Assessing structural vulnerability against earthquakes using multi-dimensional fragility surfaces: a Bayesian framework. Probabilistic Engineering Mechanics, 25(1), 49-60.

Lecomte, E. L., Pang, A. W., & Russell, J. W. (1998). Ice storm'98 (p. 99). Ottawa,, Canada: Institute for Catastrophic Loss Reduction.

Lewis H.W., Budnitz R.J., Rowe W.D., Kouts H.J.C., von Hippel F., Loewenstien W.B., (1979) Risk assessment review group report to the US Nuclear Regulatory Commission. IEEE Transactions Nuclear Science 1979;NS-26:4686–90.

Little, R. G. (2003). Toward more robust infrastructure: observations on improving the resilience and reliability of critical systems. In System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on (pp. 9-pp). IEEE.

Liu, H., Davidson, R.A., Rosowsky, D.V., Stedinger, J.R., (2005). Negative binomial regression of electric power outages in hurricanes. J Infrastructure Systems, 114: 258–267 DEW:

Michaud, D., Apostolakis, G. E. (2006). Methodology for ranking the elements of water-supply networks. Journal of infrastructure systems, 12(4), 230-242.

McEntire, D. A. (2004). The status of emergency management theory: Issues, barriers, and recommendations for improved scholarship. University of North Texas. Department of Public Administration. Emergency Administration and Planning.

Mete, H. O., & Zabinsky, Z. B. (2010). Stochastic optimization of medical supply location and distribution in disaster management. International Journal of Production Economics, 126(1), 76-84.

Mills, E. (2005). Insurance in a climate of change. Science, 309(5737), 1040-1044. OHS, 2002. Office of Homeland Security, National Strategy for Homeland Security, U.S.

Executive Order of the President, Office of Homeland Security, Washington, DC.

Ozbay, K., and Ozguven, E. (2007). Stochastic humanitarian inventory control model for disaster planning. Transportation Research Record No. 2022, Transportation Research Board, Washington, DC, 63–75.

Pate-Cornell, M.E., Dillon, R., (2001). Probabilistic risk analysis for the NASA space shuttle: a brief history and current work. Reliable Engineering System Safety 74: 345–52

Rawls, C. G., & Turnquist, M. A. (2010). Pre-positioning of emergency supplies for disaster response. Transportation research part B: Methodological, 44(4), 521-534.

Rinaldi, S. M., Peerenboom, J. P., & Kelly, T. K. (2001). Identifying, understanding, and analyzing critical infrastructure interdependencies. Control Systems, IEEE, 21(6), 11-25.

Simpson, N. C., & Hancock, P. G. (2009). Fifty years of operational research and emergency response. Journal of the Operational Research Society, S126-S139.

Wen, M., Yang, S., Kumar, A., & Zhang, P. (2009). An analysis of the large-scale climate anomalies associated with the snowstorms affecting China in January 2008. Monthly weather review, 137(3), 1111-1131.

William J. Clinton: "Executive Order 13010 - Critical Infrastructure Protection," July 15, 1996. Online by Gerhard Peters and John T. Woolley. The American Presidency Project. http://www.presidency.ucsb.edu/ws/?pid=53066.

PAPER

I. Framework for Modeling Urban Restoration Resilience Time in the Aftermath of an Extreme Event

V. Ramachandran[1], S.K. Long[2], T. Shoberg[3], S. Corns[4], H.J. Carlo[5]

1. Abstract

The impacts of extreme events continue long after the emergency response has terminated. Effective reconstruction of supply chain strategic infrastructure (SCSI) elements is essential for post-event recovery and the re-connectivity of a region with the outside. This study uses an interdisciplinary approach to develop a comprehensive framework to model resilience time. The framework is tested by comparing resilience time results for a simulated EF-5 tornado with ground truth data from the tornado that devastated Joplin, Missouri on May 22, 2011. Data for the simulated tornado was derived from *The National Map* of the U.S. Geological Survey for Overland Park, Johnson County, Kansas, in the Greater Kansas City area. Given the simulated tornado, a combinatorial graph considering the damages in terms of interconnectivity between different SCSI elements is derived. Reconstruction in the aftermath of the simulated tornado is optimized using the proposed framework to promote a rapid recovery of the SCSI. This research shows promising results when compared with the independent quantifiable data obtained from Joplin, returning a resilience time of 22 days compared with 25 days reported by city and state officials.

CE Database Subject Headings: Extreme Events, GIS, Infrastructure, Supply Chain, Resilience time, Damage, Graph Theory, Tornado

Author Keywords: Resilience time, Tornado, GIS, Supply Chain, Restoration

[1] PhD Candidate, Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, Rolla, Missouri, United States. E-mail: vrnq5@mail.mst.edu Tel # (281) 468-6769

[2] Associate Professor, Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, Rolla, Missouri, United States. E-mail: longsuz@mst.edu . Tel # (573) 341-7621

[3] CEGIS, U.S. Geological Survey, Rolla, Missouri, United States. E-mail: tshoberg@usgs.gov Tel # 572-308-3582

[4] Associate Professor, Department of Engineering Management and Systems Engineering, Missouri University of Science and Technology, Rolla, Missouri, United States. E-mail: cornss@mst.edu Tel # (573) 341-6367.

[5] Associate Professor, Department of Industrial Engineering, University of Puerto Rico at Mayagüez Call Box 9000 Mayagüez, PR 00681 Email: hector.carlo@upr.edu Tel # (787) 832-4040 x-3105

2. Introduction

Large-scale disasters impact a region, community, city, or a country in a myriad of ways. The aftermath invokes short-term emergency procedures (search, rescue, and recovery) followed by moderate- to long-term restoration efforts, the latter lasting from months to years. Existing decision–making methodologies of federal, state, and local government agencies focus primarily on emergency response functions (Veras and Jaller, 2011; Hale and Moberg, 2005; Horner and Widener, 2011). Longer-term problems associated with recovery are less well studied. To achieve substantive recovery, the restoration of an urban center’s infrastructure and that center’s reintegration into the national supply chain is necessary. The ability to restore operational performance and continuity between supply chain infrastructure elements damaged by a large-scale disaster needs to be studied in greater detail (Tamvakis and Xenidis, 2013). The complexity and interdependence of infrastructure subsystems coupled with chaotic damage from extreme events makes such restoration planning and preparation difficult.

The two main elements of this research are the supply chain strategic infrastructure and an extreme event that damaged that infrastructure. Supply chain strategic infrastructure (SCSI) in this research is defined as water, wastewater, power, transportation, and communication systems required for the normal functioning of an urban environment. The infrastructure required for proper functioning of the supply chain provides key linkages of a community to regional and national supply chain networks (Chopra and Meindl, 2007). An extreme event is defined as a celestial, geologic, meteorologic, hydrologic, or anthropogenic phenomenon that exceeds the ability of a community to cope with that event and has both short and long-term implications (Lindell and Prater, 2003). The destruction caused by such events leads to the need for immediate decisions based on overwhelming, yet inadequate, data (Rosenthal et al., 1989). The restoration of SCSI in the aftermath of an extreme event can be modeled as part of disaster-planning scenarios to guide this decision-making process.

Modeling is the initial step in planning the restoration of the SCSI, and would be done based on publically available data. To adequately assess the models to determine the efficacy of the processes it is also necessary to quantify restoration using a well-defined metric. The metric considered in this paper is *resilience time*. Resiliency is, admittedly, a loaded term and has numerous definitions within different fields of research (Horne and Orr, 1998; Comfort et al, 2001; Hollnagel et al., 2006; Cook and Nemeth, 2006; Dekker et al., 2008; Reed et al., 2009; Boin et al., 2010). In this study, we follow Reed et al. (2009) and define resilience time as the time required to restore the movement of goods and services throughout the SCSI to a particular level. Performances of SCSI elements change drastically in the wake of a large-scale disaster. The resulting damage requires that limited resources must be allocated efficiently to a complex, interdependent system to restore normal operations. Ouyang et al. (2012) proposed a three-stage resilience framework that quantifies a system's resiliency and other characteristics for electrical systems, but this tends to be infrastructure or network specific. Pant et al. (2013) and Ulieru (2007) look at specific infrastructure systems. Although these studies consider system preparedness, capacity, and recovery from hazards, the main drawback is that they do not consider resiliency across multiple interdependent systems. Other papers focus on either a particular phase in restoration or on the type of disaster that had occurred or look at particular problems like facility location or emergency management. For example, Altay and Green (2006) consider the problem from an operations research perspective, Feng and Weng (2005) and Kondaveti and Ganz (2009) consider post-disaster management, but only address the management strategies immediately after an extreme event and do not consider at the middle-to-long-term strategies of restoration. The problem of establishing medium and long-term re-connectivity of a city after a man-made or natural disaster has not been extensively investigated. The metrics introduced in these works are insufficient because the problem of restoration of SCSI elements has not been studied by using a unified approach. The framework for restoration should be able to scale across different extreme events and different regions with minimal modifications. This paper looks at the type of data required for making such framework, and how to integrate the complex data so that the resiliency time is reduced and also there is a set decision making plan.

Some previous research efforts have viewed supply chain strategic infrastructures as lifeline systems (O'Rourke, 2007; Hernandez-Fajardo and Dueñas-Osorio, 2011) that physically tie together metropolitan areas, communities, and neighborhoods to facilitate growth of local, regional, and national economies. Cagnan and Davidson (2003) described three approaches for lifeline system restoration. The first method uses restoration curves developed from a statistical analysis of historical data to directly estimate the restoration time. The second method (Ballantyne et al., 1990) uses a resource constraint approach where restoration time is a function of available resources with respect to level of damage. The third method, called evolutionary restoration (Zhang, 1992), involves modeling SCSI elements as a Markov chain in which future states of restoration depend on the current state and a set of known or assumed probabilities. While this lifeline view is an accurate representation to show the system level function of the SCSI, none of these methods individually provides a robust method that can be used to create a framework for planning restoration efforts. However, elements of each of these three methods combined provide us with a starting point for a larger holistic framework.

The goal of this research is to propose a framework that facilitates re-connectivity of a city after a disaster by: (1) understanding the type of data needed to model SCSI restoration, (2) showing the feasibility of restoration models, and (3) evaluating the restoration of SCSI elements as parameterized by the resilience time metric in the aftermath of a simulated extreme event. The research first looks at different available data related to SCSI to make a comprehensive SCSI model, and then calculates resilience time by taking into consideration restoration time and priorities for reconstruction of all the different elements.

3. Data Description and Assumptions

The proposed framework starts by acquiring geospatially located SCSI data with supply chain network parameters, restoration resource data, and hazard damage data, to construct a model of an urban center. Figure 1a shows a flow chart of the proposed framework. The urban center infrastructure is modelled as a graph by integrating the SCSI geospatial data and the SCSI element data. The destruction of the urban center is based on the Hazard damage data and is reflected in the created graph. A priority matrix is created which is used to determine the order of restoration after the damage to the different elements of the SCSI. A nearest-neighbor heuristic is used to prioritize the order of restoration of the urban infrastructure. Critical path method (CPM) and a Program evaluation and review technique (PERT) analysis utilize this heuristic along with the priority metric to calculate resilience time. A workflow depicting the same is shown in Figure 1b. The rest of the paper is organized in the same way. This section gives details about the different types of data required, the methodology section explains data integration and model building strategies and the final section describes the resilience time calculation techniques.

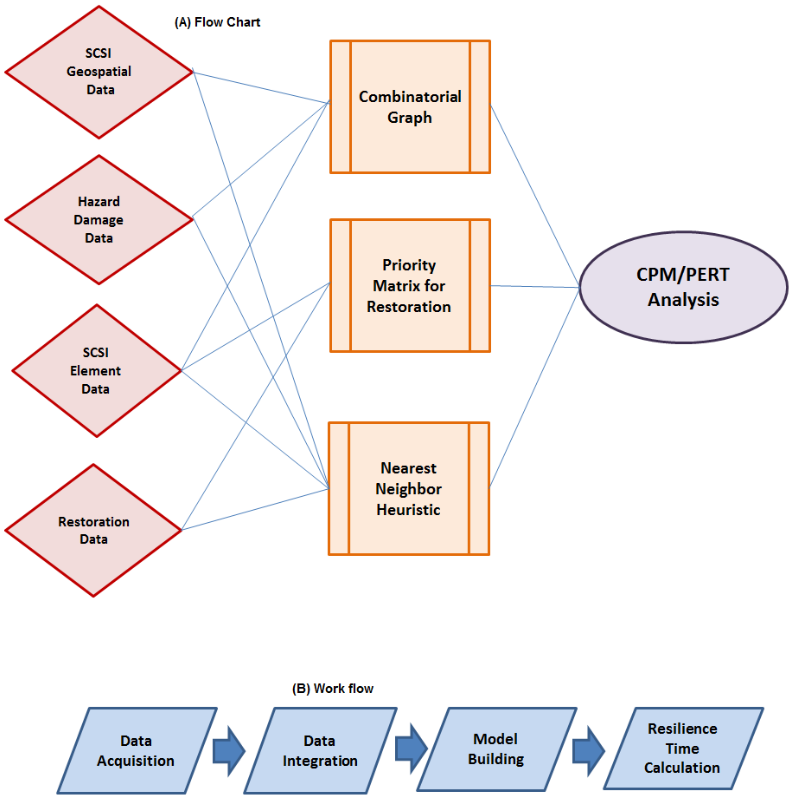


Figure 1. Outline of the Restoration Framework

A case study will be used throughout the paper in order to help describe and validate the proposed framework. The area chosen for this case study is within the city of Overland Park, located in Johnson County in northeastern Kansas. Johnson County, with a population of 542,737 (CENSUS, 2010), is adjacent to Kansas City, Missouri and is the most populous county in Kansas. This region was selected because it is an industrial and transportation-logistics hub with national connectivity, it has a considerable amount of publically accessible SCSI data, and it experiences tornadic activity at a rate that is 4.1 times the national average (Johnson County Government, 2010). Finally, a case study is presented in order to help validate the proposed framework

Populating the model space on which simulations are to be executed is essentially a problem of data integration. For this study there are four general categories of data that need to be integrated: geospatially-located SCSI element data that includes location of infrastructure elements basic geospatial data upon which the SCSI data will be located, hazard damage data, and restoration data. These categories prove necessary for creating a resilience time framework.

The SCSI elements, sometimes called lifeline system elements, chosen are shown in Figure 2. Lifeline system elements are interdependent due to the operational interaction between all the elements. These interdependencies are such that damage to one infrastructural component can rapidly cascade into damage to surrounding components, with system-wide consequences. Figure 2 shows the interdependent nature of a subset of SCSI. Circles represent basic utilities required by the supply chain. Squares represent different modes of transportation used by the supply chain. Color-coded lines connect each utility usage to the transportation mode that requires it. In this case, the light blue lines connects these water utility water and air transportation, green lines connect the power utility to the road, rail and air transportation system, likewise, black lines for fuel, and purple for communications. The red line with the double arrows shows inter-modal transportation capability, and the triangle represents holding facilities such as ports, docks, warehouses, etc.

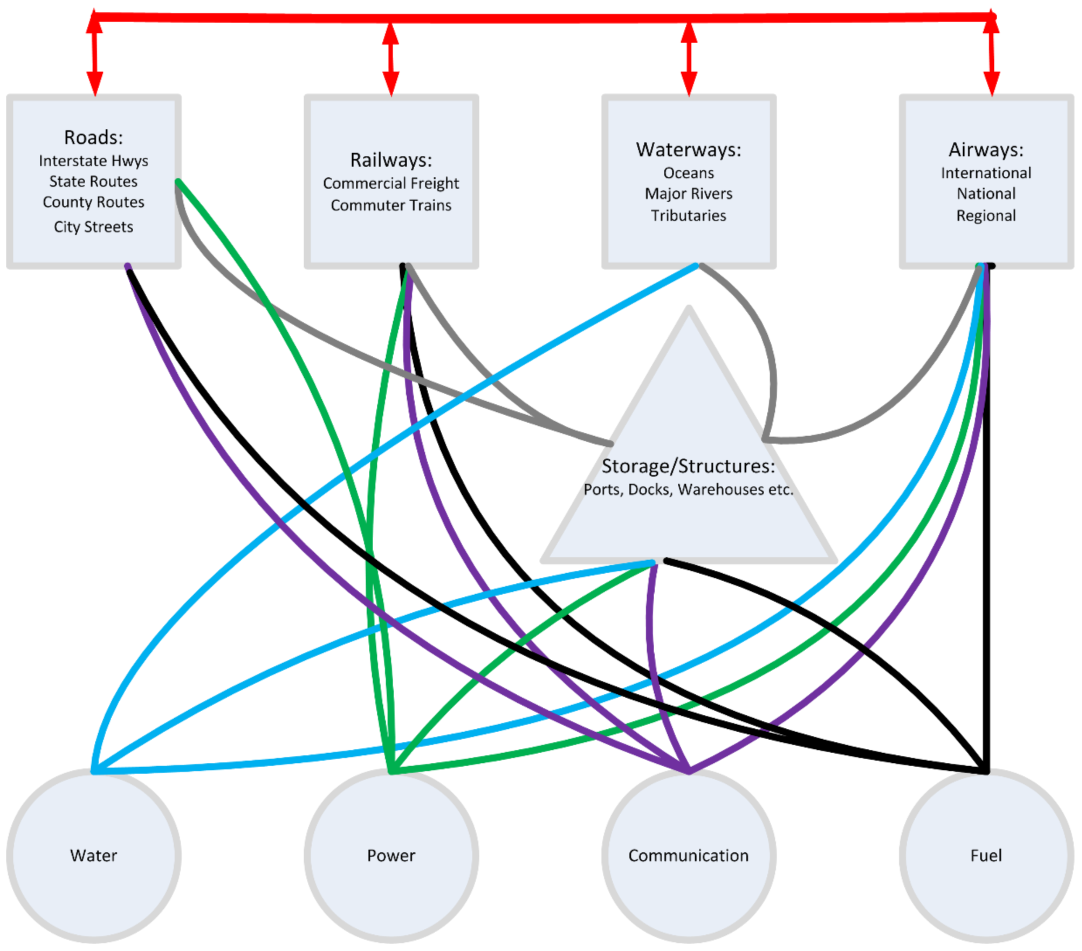


Figure 2 Modified Standard Supply Chain Model where different modes of transportation are represented by squares and utilities are represented by circles. Light blue line represents dependency of the water utility, green lines are used for power utility, black lines for fuel, and purple lines for communication, and the red lines with arrows shows inter-modal transportation for transportation of goods, e.g. rail-road, road-air, rail-water, water-road-air etc.

a. Geospatially-Located Infrastructure Elements

Geospatial data form the base upon which all other data elements are positioned. These data were largely collected from two public sector databases, *The National Map* of the U. S. Geological Survey (Sugarbaker and Carswell, 2011) and the Kansas Department of Transportation (KDOT, 2012). The U. S. Geological Survey (USGS) data included 522 tiles of 0.3 m resolution orthoimagery, the roads, bridges (road and rail), hydrography (streams, lakes and dams) shown in Figure 3. The orthoimagery were available in raster format, whereas transportation, hydrography and structure data were supplied as vector data. The high-resolution orthoimagery allowed electric grid data to be extracted. The process for electric grid extraction involved the digitization of electric poles (picked where pole shadows connected with the pole base) and electric substation locations. At this resolution the electric lines running between individual poles are also commonly visible, but visible or not the electric lines are assumed to be strung from pole to pole along a straight line. These data were further divided into lines carrying high-, moderate- and low-voltage electricity based upon pole design. These estimates were checked against KDOT schematics for high- and moderate-voltage lines in the northeast Kansas area and the agreement was excellent. While this method of data extraction has the virtue of being completely within the public domain, it is hampered by the time-consuming nature of acquisition and the inability to sample underground systems. However, underground systems remain largely undamaged by tornadic activity.

The KDOT data include communication lines (mainly cell phone towers and above ground land lines), water and sewer lines and additional structures information. The cell phone tower data were spot checked for quality control based upon select Federal Communication Commission (FCC) datasets and the USGS orthoimagery, again with excellent agreement. In general, these various data sets were easily integrated, but where there was a discrepancy, data that best fit the orthoimagery were preferred.

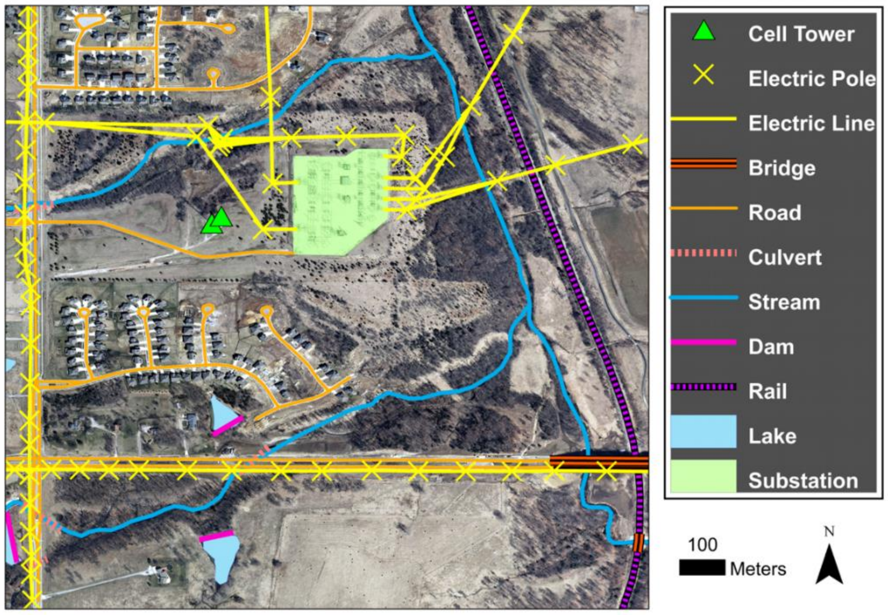


Figure 3 Orthoimagery from *The National Map* overlain by key SCSI geospatial elements (symbols defined in legend). This particular tile shows 1.5 km^2 of Johnson County, KS near the intersection of W 111th Street (running E-W near the southern boundary of the tile) and Woodland Ave (running N-S along the western boundary of the tile). The scale on this image is 1:3700.

b. Hazard Simulation (Damage Data)

The main objective of a hazard simulation is to generate the extent of damage within a given area. In an actual disaster, the level of damage to the SCSI would be direct input data. For the purpose of testing the model, however, it is necessary to produce the damage data through a hazard simulation of some sort. For this study, a hazard simulation is chosen that invokes an EF-5 tornado. Hazard data were acquired from the National Climatic Data Center (NCDC) of the National Oceanic and Atmospheric Administration (NOAA, 2004).

In this research the tornado path was chosen to cause the maximum amount of damage. The width of a tornado's path varies with intensity, and can range from 100 to 4000 meters wide and it is generally no more than 19.3 kilometers long (Rind, 1994, for example, reports that the May 31st, 2013, El Reno, OK tornado was 4-kilometer wide and lasted 26 kilometers on the ground). Tornadoes generally travel from the southwest to northeast with an average speed of 48 km/hr, and wind speeds associated with an EF-5 tornado range from 347 to 420 km/hr (Bluestein, 2006) and cause large-scale damage along and adjacent to their paths. These winds are powerful enough to lift frame houses off their foundations, toss automobile-sized missiles through the air, uproot trees, and badly damage steel re-enforced concrete structures (Grazulis, 2001). EF-4 and EF-5 tornadoes account for less than one percent of all occurrences, but are responsible for 80 percent of all tornado destruction within the United States (Bluestein, 2006).

c. Restoration Data & Supply Chain Strategic Infrastructure Elements

Restoration data includes levels of skilled workers needed, collaboration between agencies at the local state and federal level, raw material availabilities, and transportation modes for essential goods. For large scale disasters, utility and other skilled laborers are brought in from neighboring states. Collaboration entails recognition of the interdependent actions of state and local organizations and other stakeholders for plan implementation (Berke et al., 2012) and which directly affects resilience time. Restoration data for this research were acquired through the after-action reports and interviews with subject matter experts from the aftermath of tornadoes in Alabama (Gordon et al., 2011) and Maryland (Gailey, 2002).

SCSI element data identify critical elements of the infrastructure and also replacement rates necessary to keep the network viable. The National Infrastructure Protection Plan contains seventeen sectors that comprise the critical infrastructure of the United States (DHS, 2006). In this pilot study a subset was chosen for compatibility based on the eleven-system interdependent infrastructure as postulated by McDaniels et al. (2007). Table 1 shows the SCSI elements that were considered, the type of damage inflicted upon each element, and also the data source for each.

Table 1 SCSI Elements considered in the Model and damage estimates to each of the elements

Category	**Supply Chain Element**	**Type of damage**	**Geospatial Location source**
Transportation	Roads	Debris	The National Map (2011)
Transportation	Bridges	Minor structural damage, debris	The National Map (2011)
Transportation	Rail	Debris	The National Map (2011)
Electric	Lines, power poles, and substations	Lines down, structural damage to poles and substations	The National Map (2011)

Table 1 SCSI Elements considered in the Model and damage estimates to each of the elements (cont.)

Communications	Cell towers and Land lines	Lines down, structural damage to towers	MARC (2010)
Water	Pipeline and Pumping stations	Debris impact to pumping stations	MARC (2010)

4. Methodology

a. Combinatorial Graph

The data described above provide a basis for building a framework of the SCSI elements for a given region. One modeling method is by utilizing combinatorial graph theory. Graph theory is a means of showing connectivity of elements in a model (West, 2000). The different elements of SCSI can be modeled into a graph by representing them as vertices or edges. The different interconnections and interfacing between the elements can be explicitly shown on a graph. A combinatorial graph or Graph (G) is an ordered pair $G = (V,E)$ comprising of a set V, of vertices or nodes together with a set E of edges or lines which are 2-element subsets of V (i.e., an edge is related with two vertices, and the relation is represented as an unordered pair of the vertices with respect to the particular edge (West, 2000). For example, let graph, G_1 represent a road transportation network (Figure 4). All bridge and road intersections are represented as vertices (circles in Figure 4) and the roads connecting them as edges (lines connecting the circles). A path is a sequence of edges connecting a series of vertices. A graph is said to be connected if one or more paths exists between every pair of vertices. For this research each utility is represented by its own graph and prior to an extreme event each graph is connected. Note that the graph is considered connected because a pathway exists such that it is possible to go from any vertex to any other vertex.

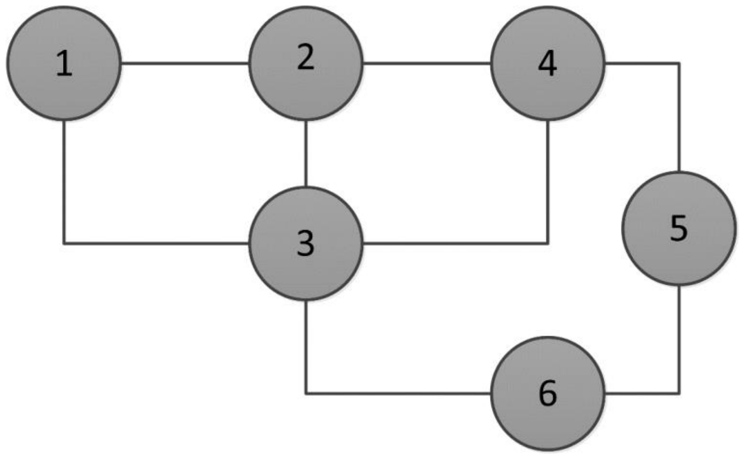


Figure 4 Graph representation for G_1

The SCSI elements identified in Table 1 were divided into edges and vertices that form the combinatorial graph (Table 2).

Table 2 vertices and edges for Graph

Vertices	**Edges**
Bridges & Intersections	Roads
Electric Poles and Substations	Electric Lines
Communication Towers	Communication Lines
Pumping stations	Water pipelines

Figure 5 shows a graph of the SCSI elements idealized as edges and vertices overlain on orthoimagery for a subset of the study area. A majority of the SCSI elements that are being considered in this research are shown in Figure 5 (some are not because they are not present in this area). Even though this figure is overlain on orthoimagery a connected graph can be seen with vertices and edges representing the SCSI elements. This graph is a representation of the entire infrastructure that is necessary for normal functioning of the city; it shows the connectivity between the different elements and also interconnections of the network.

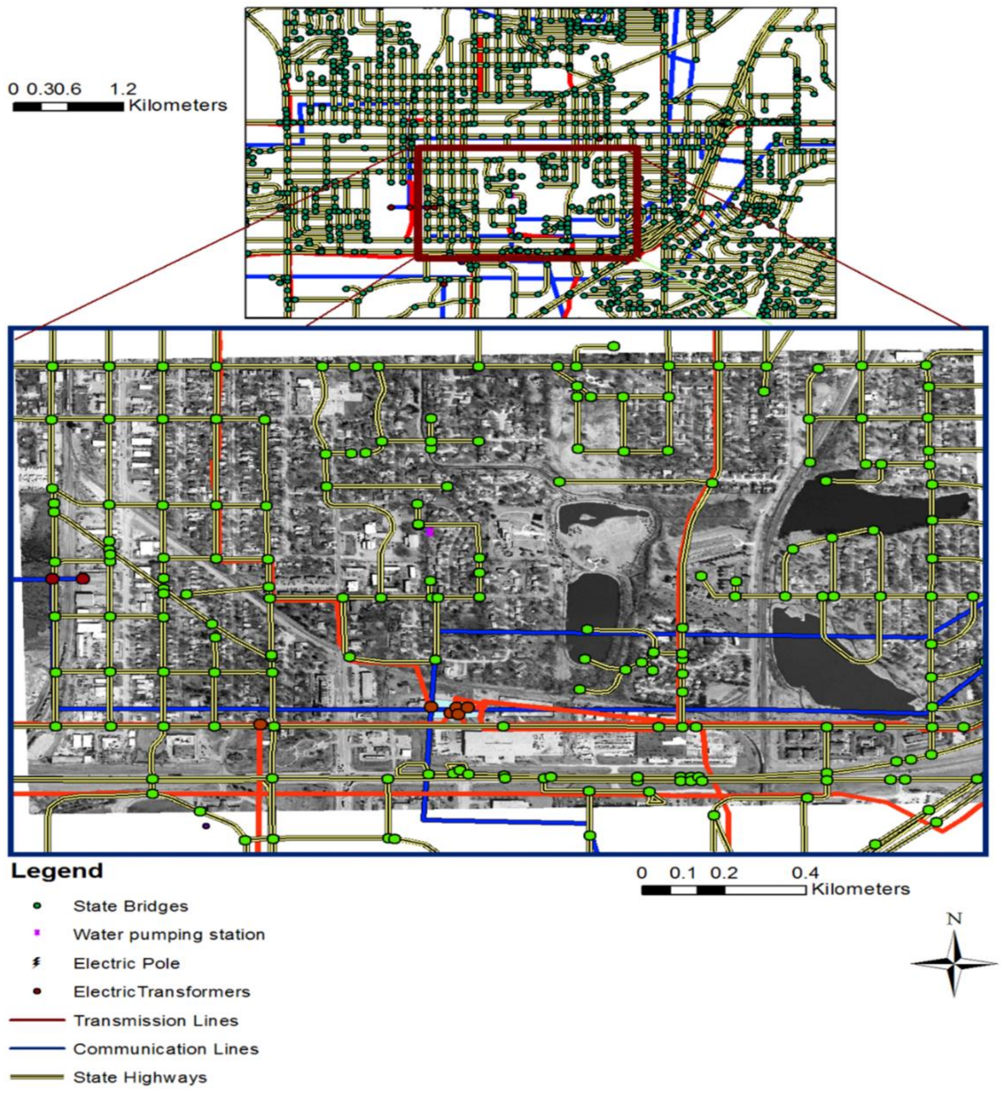


Figure 5 Subgraph within Johnson County, Kansas. Combinatorial graph shows the relevant SCSI elements, composed of electric grid, communication lines, state road transportation lines and water pumping station

b. Tornado Simulation

An EF-5 tornado is simulated within the model. The simulation path through Overland Park is 12 kilometers (7.5 miles) in length and 2.5 kilometers (1.5 miles) wide heading due north. The simulation invokes a zone of maximum damage, 100% of all edges and 90% of vertices are damaged or unusable, within the tornado’s path. A zone of reduced damage extends outward radially an additional 700 m from the edge of the previous zone. Within this 700 m swath, edges connected to vertices inside of the maximum damage zone were destroyed.

c. Construction of the Priority Matrix

A hierarchy of the order of restoration for SCSI elements was constructed. This hierarchy is represented as a priority matrix of weights and priorities (see Table 3). Priorities define the preferred order of restoration and are assigned numbers ranging from two to five, two being lowest priority. Weights are also assigned across each priority element signifying the importance of any particular vertex. The first step in restoration deals with clearing the large amount of debris on roads, so that the restoration and damage assessment crews can reach the area and decide on the course of action. The SCSI element with highest priority and largest weight establishes the initiation of restoration. This priority matrix is constructed in consultation with subject matter experts. Transportation is given the highest priority due to the fact that crews need access to the damaged areas before any other repair work can start. Electricity is the next utility to be restored, followed by communications and then water. Table 3 shows the order of restoration which was employed for this study.

Table 3 Priority matrix

	Weights		
Priorities	**5**	**3**	**1**
5 (Transportation)	Federal Highways, Federal Bridges	State Highways, State Bridges, Rail	Local Roads
4 (Electricity)	Substations & High voltage lines	Intermediate voltage lines	Low voltage lines
3 (Communication)	Communication Towers	Communication Lines	
2 (Water)	Water pumps	Water pipelines (Note: Clean and Waste Water are separate)	

d. Nearest-neighbor Algorithm

A nearest neighbor algorithm is used to restore SCSI elements. In a nearest-neighbor method, each vertex controls an area corresponding to a fraction of the distance to its neighbor in any given direction (Bondy and Murty, 1976). Increasing the weight increases the area controlled by a given vertex. Figure 6 gives an example of how the nearest neighbor algorithm would operate. Element 1 and element 8 must be reconnected, while elements two through seven are available infrastructure elements between them. The dashed lines represent possible connections that are not part of the solution, while the solid lines represent the path found by the algorithm. The priority matrix identifies the initial SCSI element that needs to be restored (represented by vertex 1 in Figure 6). The nearest neighbor algorithm finds the closest vertex (in this case, vertex 2) and joins the two by an edge. This edge represents restoration of functionality of the elements represented by these two vertices. In the next iteration, the algorithm finds the nearest unvisited vertex (vertex

7) and joins them by an edge, again representing functionality. This is repeated until a path exists that connects vertex 1 to vertex 8. For the simulation we define the restoration time as the time until 80% of the pre-event system functionality is restored. Hence, in our simulation these iterations continue until 80% of SCSI connectivity is restored.

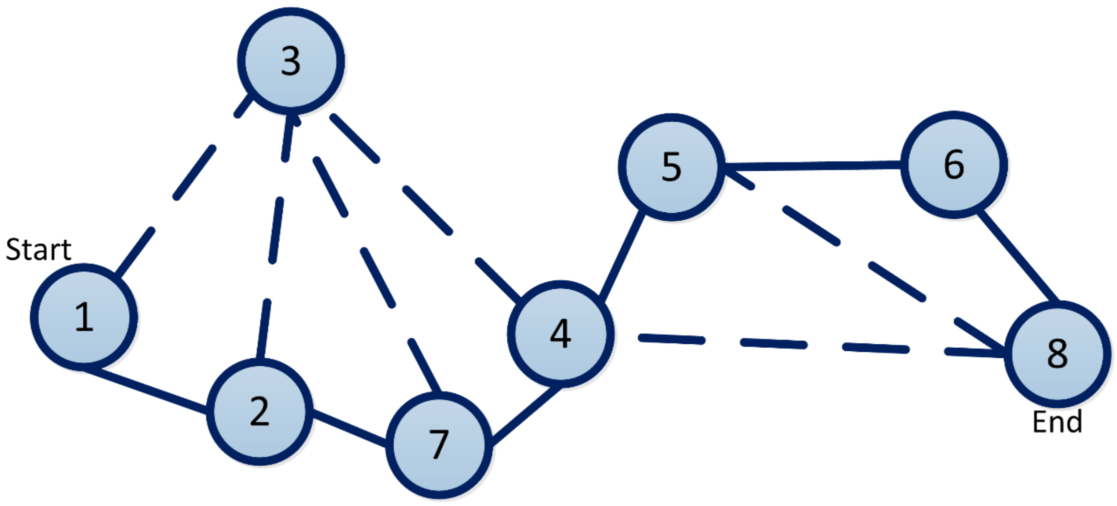


Figure 6 Nearest-neighbor Algorithm. Solid lines show shortest path to neighboring vertex, dashed lines show unused (longer) paths. Note that connectivity is established in the vertex sequence 1, 2, 7, 4, 5, 6, 8 and vertex 3 is no longer part of the grid system after restoration.

e. Critical Paths

The Critical Path Method (CPM) is one of the several algorithms for scheduling a set of project activities. It is made for projects that are have a number of smaller individual “activities” and it determines the longest path of planned activities to the end of the project

(Kelley, 1963). If some of the activities require other activities to finish before they can start, then that leads to the project becoming complex. The main advantage of using CPM is that it calculates the optimal sequence of planned activities for any project and it can help figure out how much time the whole project will take to complete. It may also have activities that are critical meaning that they have to be completed within the specified time or the project will get delayed. In this research the total resilience time is not a matter of simply adding all reconstruction times since some of the activities require other activities to either partially or completely finish before they can start and hence are on the critical path of restoration.

5. Results

a. Calculation of Resilience Time

Resilience time calculation is based on the time dependence of infrastructure's repair function. The infrastructure repair function includes: number of utility workers, raw material availability, and collaboration of the different federal, state, and local groups. The restoration data (Table 4) is mainly derived from reports by various agencies and also personal communication with engineers and workers who have worked on different tornadoes in the past. The rate of restoration varies from case to case but a mean value was used in this case.

Table 4 Restoration data

SCSI Element	**Rate of Restoration**	**Data Source**
Highways/Bridges	30 tons/truck/day	(Miller, 2007, p. 8-12); (Felknor, 1992); (Jackson, 2012)
Electric Lines	5 meters repaired/day/worker	(Personal Communication, Ameren UE); (Gordon et al. 2011, p. 38-39);
Communication Lines	10 meters repaired/day/worker	(FEMA, 2012 p. iv, 8-2); (Gordon et al. 2011, p. 65-66)
Water	6 workers repair 1 pump/day	(Personal Communication, St. Louis Sewer District); (FEMA, 2012 p. *iii*, 8-3);

Figure 7 shows pre-destruction, post-destruction, and post-restoration schematics. The image on the left shows the pre-destruction representation where all the vertices are connected. The image in the middle is the post-destruction representation wherein 90% of all the vertices and 100% of all the edges have been destroyed. In this simulation, infrastructure within the affected area has collapsed and all elements of the SCSI are impacted. Electric lines are down, highways are clogged with debris, water and sewage lines are non-functional, and communications are sporadic. The resilience time depends on how quickly network connectivity can be restored. The image on the right is post-restoration representation with 80% restoration by using the proposed framework.

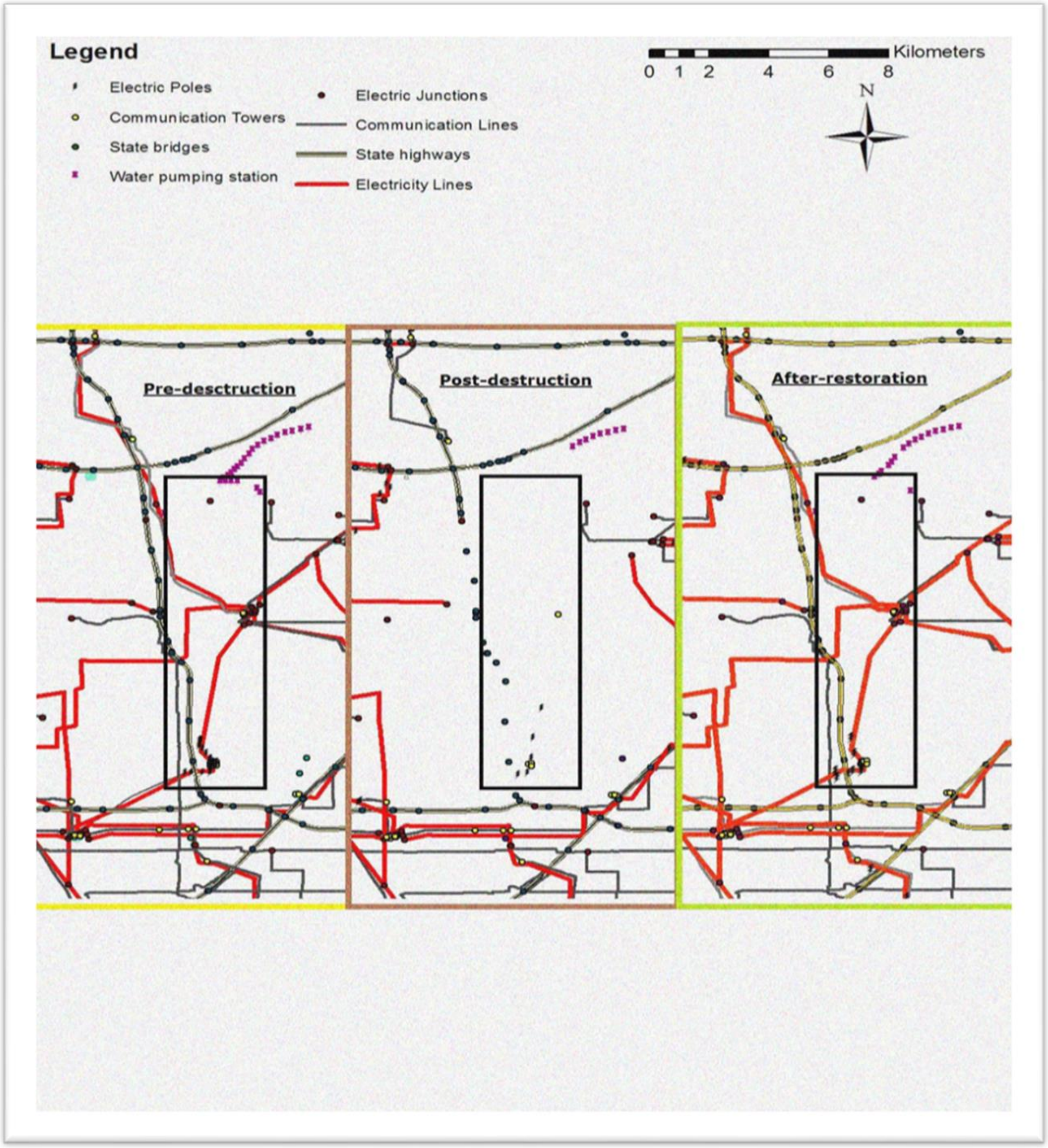


Figure 7 Pre-destruction, Post-destruction, and Post-restoration schematics

In this simulation there were 80,000 tons of debris on the road, 20,000 meters of destroyed electric lines, 12,500 meters of communication lines down, and 625 water pumps off-line. Resilience time for each independent SCSI element was calculated as shown in Table 5.

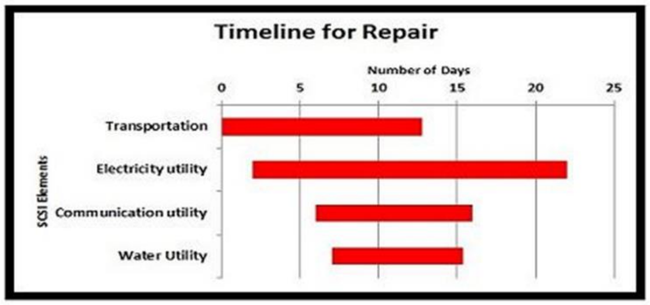
Table 5 Resilience Time per Element

SCSI Element	**Total Damage**	**80% of total damage**	**Time-dependence factors**	**Time to Repair**
Highways & Bridges	80,000 tons of debris	64,000 tons	5000 tons cleared in a day	12.8 days
Electric Lines	20,000 meters to be replaced	16,000 meters	160 workers	20 days
Communication Lines	12,500 meters to be replaced	10,000 meters	100 workers	10 days
Water Pumping Stations	625 water pumps damaged	500 water pumps	60 pumps repaired in a day	8.4 days

Figure 8a shows a Gantt chart of the timeline for restoration. The restoration work on select SCSI element depends on the prior restoration of other elements as reflected in assigned priorities. Since the repair work is interdependent, all four of the model elements are on the critical path. For example, the debris on the road network must be cleared to a certain extent before utility workers can reach damaged structures. Electric lines and junction repair starts after some of the major roads have been at least partially cleared of debris. Communication and water are dependent on the initial restoration of both roadway and electric lines. The time required for 80% restoration of the supply chain elements is equal to the completion time of the last activity on the critical path. The total time taken to reach 80% restoration of the supply chain elements is 22 days as calculated using the CPM method.

An alternative method for depicting critical path information and connectivity between tasks or elements can be illustrated by an Activity-on-Node (AON) diagram (Figure 8b). In an AON diagram, an activity is represented by a node (box) and the dependencies among the activities are depicted using the arrows between nodes. The relationship between the different activities and the order in which the activities are performed is represented by arrows. Even though both figures in essence have the same attributes and show total resilience time, the AON diagram shows the interdependent nature of the different activities along with the activities on the critical path.

a



b

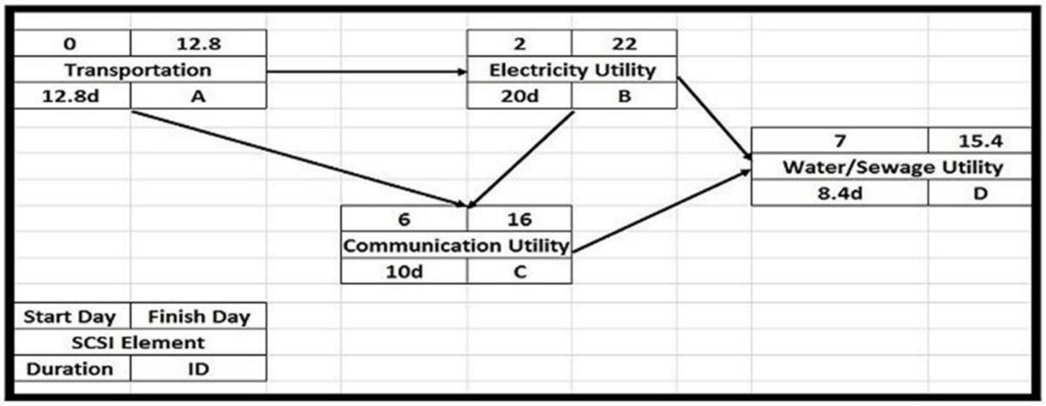


Figure 8. (a) Gantt chart for resilience time, shows when the work will start and finish on the different SCSI elements according to the priority matrix and amount of work completed. (b) Activity-on-Node Network shows total days to complete the restoration work

b. Joplin Tornado

On 22 May, 2011 an EF-5 tornado struck Joplin, Missouri, traveled 5 miles through the city center, and caused massive destruction. Approximately two hours after the tornado left the area restoration efforts began. Transportation priority was given to the reopening the Interstate 44 (I-44) corridor which runs along the southern boundary of Joplin. According to Missouri Department of Transportation (MoDOT) and City of Joplin officials, the roadways and bridges were assessed for damage by structural engineers over the span of several days, but within five days most major roads were passable with the roadways restored to 80% pre-event capacity by the tenth day. Power utility workers removed live power cables as roads became passable. Communication was initially spotty and sporadic. Water and sewage flows were also impaired. Communication and water were restored to 80% pre-event capacity in 10 days. By the end of one week approximately 50% of electric service was restored and by the eighteenth day electricity distribution systems reached 80% of pre-disaster capacity (Gregg and Lofton, 2011).

The model results have been compared with the events in Joplin to validate the described framework and not as a direct comparison of results. The restoration efforts in Joplin are representative of post-disaster recovery processes. The Joplin data has not been used in the creation of this framework, and the data to create this framework comes from reports and publication on other tornadic events.

The resilience time calculations from the model were consistent with data from Joplin, Missouri tornado (Table 6). The model yielded a resilience time of 22 days compared with 24 to 25 days for Joplin. According to the model, communication lines were re-established within 10 days, water pipelines 8-9 days. The later finished relatively quickly even though water restoration was given a lower priority, which implies minimal damage. Major roads and bridges became operational within 5 to 6 days, while the electricity took 18 days to be restored to 80% capacity. The difference between the model and the ground truth can be attributed to any number of factors, including simplicity of the model. These results are encouraging and imply that integrating different elements of the SCSI with geospatial data into a single model is practicable. This model does not reflect the real-world system, but if the data that has been identified in the paper is available then it is possible to create a framework for restoration which can mimic the real world scenarios.

Table 6 Model comparison against Joplin Restoration

Critical Infrastructure	**Model**	**Joplin**	**Difference between Model and Joplin**
Local Transportation Network (Major Roads)	6 days	5 days	1 day
Electricity	20 days	18 days	2 days
Communication Lines	10 days	10 days	0 days
Water Pipelines	8.4 days	10 days	1.6 days
Resilience Time	**22 days**	**24 to 25 days**	**2 to 3 days**

6. Discussion and Conclusion

Disaster restoration of SCSI is a complex system and as such, it is essential that system elements, connectivity, and sources of complexity be addressed in any decision framework designed to reconnect an urban environment to the larger economic infrastructure. A decision framework is best developed through the use of a systems approach. This approach identifies interface points between infrastructures. The components consist of lifeline systems including, but not limited to: transportation, power, communication, and water. The interactions between these SCSI lifeline systems provide value-added detail within a decision framework. These interactions must be considered as part of an extreme-event restoration plan.

Public access to SCSI data is a major challenge to restoration modeling. These data are often proprietary or restricted, however, results show that adequate data can be identified or derived from publically available data sets. The integration of these data is challenging due to their complexity and variety. These data provide a good snapshot of what resources would be required to restore the SCSI in an optimized timeframe and they include geospatial and technical information used to populate the model framework. Combinatorial graph theory was used to determine the efficacy of this approach and as a proof-of-concept to evaluate whether the data integration tasks can be completed at a sufficiently robust level to provide meaningful results. Despite the simplicity of the approach, results are promising.

The model simulation calculated a resilience time of 22 days which compares favorably with 24 to 25 days reported by city and state officials for the initial recovery from the EF-5 tornado that devastated Joplin, Missouri, in 2011. In the model simulation, communication lines were re-established to within 80% of pre-existing capacity in 10 days, water pipelines in 8-9 days, provided that roads and electrical infrastructure were first sufficiently repaired for access and functionality. Major roads and bridges became operational in 5-6 days, and the electric junctions took 18 days to be restored to 80% capacity. Variance of the model with the Joplin ground truth data can be attributed to any number of reasons such as: the level of collaboration between the local, state and federal agencies, the availability of raw materials for restoring such items as electric poles or communication towers, the number of utility workers that can be called into the area, the time taken to assess the extent of damage, the accessibility of the area in the aftermath of the disaster, and the simplicity of the model. Even though the demographics of Johnson County and Joplin have differences, the validation results imply that integrating different elements of the SCSI together with geospatial data into a single model can aid in determining extreme-event management decision frameworks.

7. Future Work

Future work will increase the quantity and complexity of real-world SCSI data. Further, more sophisticated modelling techniques such as agent-based modeling and complex adaptive system approaches will create a more realistic and robust analysis. Such techniques will confirm that the methods developed in this research scale across regions and hazard type. Ultimately, a straightforward user-interface will be developed that can input local data for community planners to develop restoration strategies. The ability to manage data from either public or private sources is also a challenge when attempting to collect the necessary data to create such a framework. Most of these databases are proprietary or private and not available in open source. Some of the data that does exist is static and outdated. This makes determining which database is best suited for certain analysis a strategic step in constructing this framework.

The priority matrix also requires increased sophistication in its design. Currently there is no prioritization among the SCSI elements. For example, any set of electric poles can be connected to any other set of electric poles regardless of whether they serve the same function (low, moderate, or high voltage) and this should be addressed in the model.

Delphi Studies are also planned in order to better understand the prioritization necessary for different supply chain elements (Hasson et al., 2000). This is an iterative process, where experts answer questionnaires in two or more rounds. After each round, a facilitator provides a summary of the experts' answers from the previous round as well as the reasons for their answers. The questions are asked again and the experts are encouraged to revise their earlier answers.

8. Acknowledgments

Partial funding for this research is provided through US Geological Survey award number G13AC00028. We thank Mr. George Mues, Ameren UE and Mr. Tom Broaders, Metropolitan St. Louis Sewer District, for providing important information on the prioritization of SCSI elements and the repair rate of damaged elements. We also wish to thank Mr. Mike Middleton, Missouri Department of Transportation (MoDOT), and Mr. Jack Shaller, former Assistant public works director of the city of Joplin, Missouri, for the reconstruction statistics for Joplin, Missouri. Finally, we thank Mr. Michael Finn and Dr. E. Lynn Usery, U.S. Geological Survey, and Dr. Ruwen Qin, Missouri University of Science and Technology, for their reviews of the manuscript.

9. References

Altay, N., and Green III, W. G. (2006). “OR/MS research in disaster operations management”. European Journal of Operational Research, 175(1), 475-493.

Ballantyne, D. B., Berg, E., Kennedy, J., Reneau, R., and Wu, D. (1990). “Earthquake loss estimation modeling of the Seattle water system.”*Tech. Rep.*, Kennedy/Jenks/Chilton, Federal Way, Wash.

Berke, P., Smith, G., and Lyles, W. (2012). “Planning for Resiliency: Evaluation of State Hazard Mitigation Plans under the Disaster Mitigation Act”. Nat. Hazards Rev., 13(2), pp. 139–149.

Bluestein, H. B. (2006). “Tornado alley: Monster storms of the Great Plains.” New York: Oxford University Press

Boin, A., Kelle, P., and Whybark, C. (2010). “Resilient supply chains for extreme situations: Outlining a new field of study”. International Journal of Production Economics, 126(1), pp. 1-6, 2010.

Bondy, J. A., and Murty, U. S. R. (1976). *Graph theory with applications* (Vol. 290). London: Macmillan.

Cagnan, Z. and Davidson, R. (2003). “Post-earthquake lifeline service restoration modeling”. Sixth U.S. conference and workshop on Lifeline earthquake engineering, August 10-13. Ed. J. E. Beavers. Long Beach, California: American Society of Civil Engineers, pp. 255-264.

Chopra, S. and Meindl, P. (2007). “*Supply chain management: strategy, planning, and operation”*. Pearson Prentice Hall, New Jersey, 2007.

Comfort, L. K., Sungu, Y., Johnson, D., and Dunn, M. (2001). “Complex systems in crisis: anticipation and resilience in dynamic environments”. Journal of Contingencies and Crisis Management, *9*(3), pp.144-158.

Cook, R. I., and Nemeth, C. (2006). “Taking things in one’s stride: Cognitive features of two resilient performances”. In E. Hollnagel, D.D. Woods, and N. Leveson, (Eds.), *Resilience engineering: Concepts and precepts* (pp. 205). England: Ashgate Publishing Limited.

Dekker, S., Hollnagel, E., Woods. D., and Cook, R. (2008). “Resilience engineering: New directions for

measuring and maintaining safety in complex systems”, Final report, Sweden: Lund University, School of Aviation, pp. 10.

Department of Homeland Security, (DHS), (2006). “National Infrastructure Protection Plan”. Web: *www.dhs.gov/nipp*.

Federal Emergency Management Agency (FEMA). (2012). “Mitigation Assessment Team Report. Spring 2011 Tornadoes: April 25-28 and May 22, Building Performance Observations, Recommendations, and Technical Guidance”. pp. 908 Washington, D.C.: Federal Emergency Management Agency.

Felknor, P. S., 1992: “The Tri-State Tornado: The Story of America’s Greatest Tornado Disaster”. Iowa State University Press, pp. 131

Feng, C., and Weng, C., (2005). “A bi-level programming model for allocating private and emergency vehicle flows in seismic disaster areas”. In *Proceedings of the Eastern Asia Society for Transportation Studies* (Vol. 5, pp. 1408-1423).

Gailey D., (2002), “Tornado Damage Assessment Report, forest Service – Southern Region Department of Natural Reserves”. Annapolis, Maryland. Website: http://www.dnr.state.md.us/forests

Grazulis, T.P., (2001). “The Tornado: Nature's Ultimate Windstorm”, University of Oklahoma Press, Oklahoma, pp. 324.

Gregg C., and Lofton L., (2011). “The Response to the 2011 Joplin, Missouri, Tornado Lesson Learned. Study Federal Emergency and Management Agency (FEMA)”, Kansas City, Kansas, USA.

Gordon T., Parks D., Rada J., and Weaver K., (2011), “Cultivating a State of Readiness. Tornado Recovery and Action Council”. Alabama, USA.

Hasson F., Keeney, S., and McKenna, H. (2000), “Research guidelines for the Delphi survey technique”. Journal of Advanced Nursing 32, pp. 1008–1015

Hale, T., and Moberg., C.R., (2005), “Improving Supply Chain Disaster Preparedness: A Decision Process for Secure Site Location”. International Journal of Physical Distribution & Logistics Management, 35.3, pp. 195-207.

Hernandez-Fajardo, I., & Dueñas-Osorio, L. (2011). “Sequential propagation of seismic fragility across interdependent lifeline systems”. *Earthquake Spectra*,*27*(1), pp. 23-43.

Hollnagel, E., Woods, D.D. and Leveson, N. (2006). “Resilience engineering: concepts and precepts”.

England: Ashgate Publishing Limited.

Horne, J. F., III, and Orr, J. E., (1998). “Assessing behaviors that create resilient organizations”.

Employment Relations Today **24** (4), pp. 29–39.

Horner, M. W., and Widener. M.J., (2011). “The Effects of Transportation Network Failure on People's Accessibility to Hurricane Disaster Relief Goods: A Modeling Approach and Application to a Florida Case Study”, Natural Hazards Review, 59, pp. 1619-634.

Jackson T. (2012) “Emergency Relief Program, Missouri Department of Transportation. Branson, MO.

Johnson County Government (2010). *Government of Johnson County, Kansas Website*, Web: http://jocogov.org/, last visited (Nov 2012)

Kansas Department of Transportation, (KDOT). (2012). *Kansas Department of Transportation*. Web: http://www.ksdot.org

Kelley, J. E. (1963). “The critical-path method: Resources planning and scheduling”. Industrial scheduling, 347-365.

Kondaveti, R., and Ganz, A. (2009). “Decision support system for resource allocation in disaster management”. In Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE (pp. 3425-3428). IEEE.

Lindell, M. K., and Prater, C. S., (2003). “Accessing Community Impact of Natural Disasters”, Natural Hazards Review 4.4 pp. 176-85

McDaniels, T., Chang, S., Peterson, K., Mikawoz, J., and Reed, D. (2007). “Empirical framework for characterizing infrastructure failure interdependencies”. *Journal of Infrastructure Systems*, *13*(3), pp. 175-184.

Mid-American Regional Council (MARC). (2010). *GIS Data Center*. Web: http://marc.org/gis/gisdata.htm last visited (Nov 2012)

Miller, D. (2007). "KDOT Response to Tornado Heroic", p. 8-12. Translines. Website http://www.ksdot.org/bureaus/offTransInfo/TRANSLIN/June/June07.pdf last visited (Nov 2012)

National Oceanic and Atmospheric Administration (NOAA). (2004). "General circulation model output data set for Paleoclimatology Data Contribution Series #1994-012". Website: http://www.ncdc.noaa.gov/oa/climate/severeweather/tornadoes.html (Feb. 2012)

O'Rourke, T. D. (2007). "Critical infrastructure, interdependencies, and resilience". *Bridge, Washington-National Academy of Engineering*, *37*(1), pp. 22.

Ouyang, M., Dueñas-Osorio, L., and Min, X. (2012). "A three-stage resilience analysis framework for urban infrastructure systems." *Structural safety*, *36*, pp. 23-31.

Pant, R., Barker, K., and Zobel, C. W. (2013). "Static and dynamic metrics of economic resilience for interdependent infrastructure and industry sectors". *Reliability Engineering & System Safety*.

Reed, D., Kapur, K.C., and Christie, R.D., 2009. Methodology for assessing the resilience of network infrastructure, IEEE Systems Journal, 3, 174-180.

Rind, D. (1994), "General Circulation Model Output Data Set". IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series #1994-012. NOAA/NCDC Paleoclimatology Program, Boulder, Colorado, USA.

Rosenthal, U., Charles, M.T., and Hart, P. (1989). "Coping with Crises: The Management of Disasters, Riots and Terrorism." Charles C. Thomas Publishers, Springfield, IL.

Sugarbaker, L. J., and Carswell, Jr., W. J. (2011). "*The National Map*." U.S. Geological Survey Fact Sheet, pp. 2011–3042.

Tamvakis, P., and Xenidis, Y. (2013). "Comparative Evaluation of Resilience Quantification Methods for Infrastructure Systems". *Procedia-Social and Behavioral Sciences, 74,* pp. 261-270.

Ulieru, M. (2007). "Design for resilience of networked critical infrastructures. In Digital EcoSystems and Technologies Conference". *Inaugural IEEE-IES,* pp. 540-545.

United States Census Bureau. (CENSUS) (2010). *2010 Census*. Web. Accessed on 25 January 2013.

Veras, J. H., and Jaller, M. (2011). "Immediate Resource Requirements after Hurricane Katrina". Natural Hazards Review, 13.2, pp. 117-31.

West, D. B., (2000). "Introduction to Graph Theory", 2nd ed. pper Saddle River, NJ:Prentice Hall.

Zhang, R.H., 1992. Lifeline interaction in post-earthquake urban system reconstruction, Earthquake Engineering, 10th World Conference, Balkema Press, Rotterdam, 5475-5480.

II. Identifying Geographical Interdependency in Critical Infrastructure Systems Using Publically Available Geospatial Data in Order to Model Restoration Strategies in the Aftermath of a Large-Scale Disaster

Varun Ramachandran, Tom Shoberg, Suzanna Long, Steven M. Corns, and Hector Carlo

1. Abstract

In the wake of a large-scale disaster, strategies for emergency search and rescue, short-term recovery, and medium- to long-term restoration are needed. While considerable effort is geared to developing strategies for the former two options, little comprehensive guidance exists on the latter. However, medium- to long-term restoration has a significant effect on local, regional and national economies and is essential to community vitality. In part, the deficit of robust strategies can be linked to the complexity in the data acquisition and limited methodologies to understand the interconnectedness of the relevant systems elements. This research utilizes geospatial data for Supply Chain Interdependent Critical Infrastructure (SCICI) such as transportation, energy, communications, or water, obtained or derived through publically available sources (such as The National Map of the U.S. Geological Survey) to identify, understand, and map the interdependencies between these system elements to enable restoration planning. Specifically, internal geographical relationships (herein called the 'geographical interdependency') of SCICI are mapped. These interdependencies highlight the stress points on the larger SCICI where failures occur and are not included in current built environment models. The mapping of these interdependencies is a key step forward in attempts to optimally restore an urban center's supply chain in the wake of an extreme event.

2. Introduction

The U.S. socioeconomic structure is heavily dependent on its network of critical infrastructures. These infrastructures are complex, interdependent and include numerous interface points; a disturbance in one can quickly cause cascading failure in the others. These infrastructures and their importance are defined as (DHS, 1996):

Certain national infrastructures are so vital that their incapacity or destruction would have a debilitating impact on the defense or economic security of the United States. These critical infrastructures include telecommunications, electrical power systems, gas and oil storage and transportation, banking and finance, transportation, water supply systems, emergency services (including medical, police, fire, and rescue), and continuity of government…

The restoration of supply chain networks following a natural or man-made disaster is a pervasive challenge for decision makers responsible for the reintegration of regional or national supply networks after emergency response phases have ended. Although most disaster response models include cursory socioeconomic recovery plans, there is no comprehensive model capable of using data and decision variables in real time (Ramachandran, et al, 2014). This research models critical infrastructures in terms of their connectivity to the U.S. supply chain system and identifies geographic interdependencies associated with this system. The term supply chain interdependent critical infrastructure (SCICI) is used to define interdependent supply chain components. These include transportation, power, communications, and water (Figure 1). Understanding interdependency is a data-intensive process ranging from data acquisition and integration to data simulation.

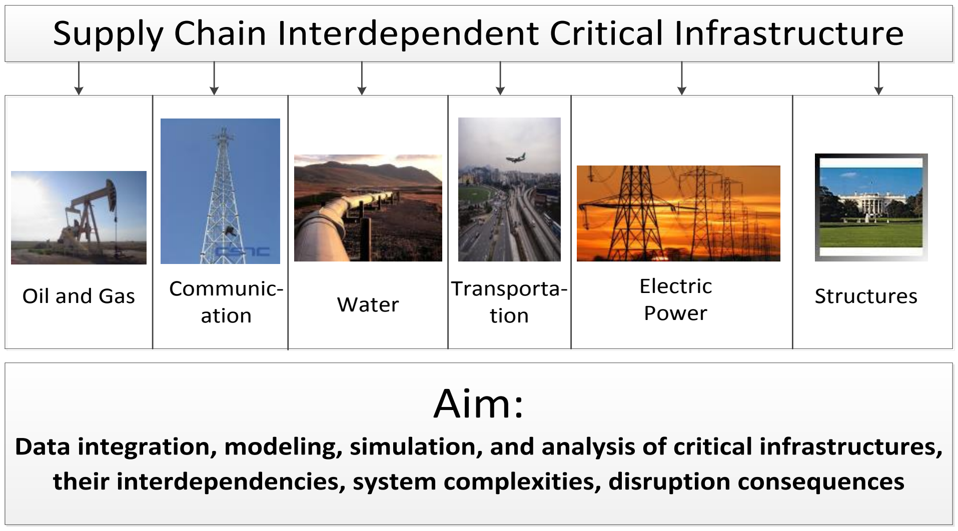

Supply Chain Interdependent Critical Infrastructure
Oil and Gas
Communic-ation
Water
Transporta-tion
Electric Power
Structures
Aim:
Data integration, modeling, simulation, and analysis of critical infrastructures, their interdependencies, system complexities, disruption consequences

Figure 1. Supply Chain Interdependent Critical Infrastructure

Events over the past decade highlight the vulnerability of critical infrastructures and also showcase interdependency among elements. During the East coast blackout (August 14, 2003) the initial problem impacted the electrical generation and distribution network, but cascade effects on other systems from water to transportation disrupted the daily lives of fifty million people across North Eastern United States and parts of Canada (Talukdar, 2003). Similarly for both Hurricane Katrina (August, 2005) and Super Storm Sandy (October, 2012), 4.8 million people were impacted. Following Super Storm Sandy, people were without power in 15 states, there was shortage of petroleum in many cities due to supply chain disruption, and the cost of repair in New Jersey alone was $36.8 billion dollars (Blake et al., 2013).

Figure 2 represents the number of global natural disasters reported between 2000 and 2011 (EM-DAT). The sheer volume of these occurrences is further evidence of the need for an effective restoration process for damaged SCICI. This restoration sequence must be based on an understanding of the interdependence of SCICI to be effective.

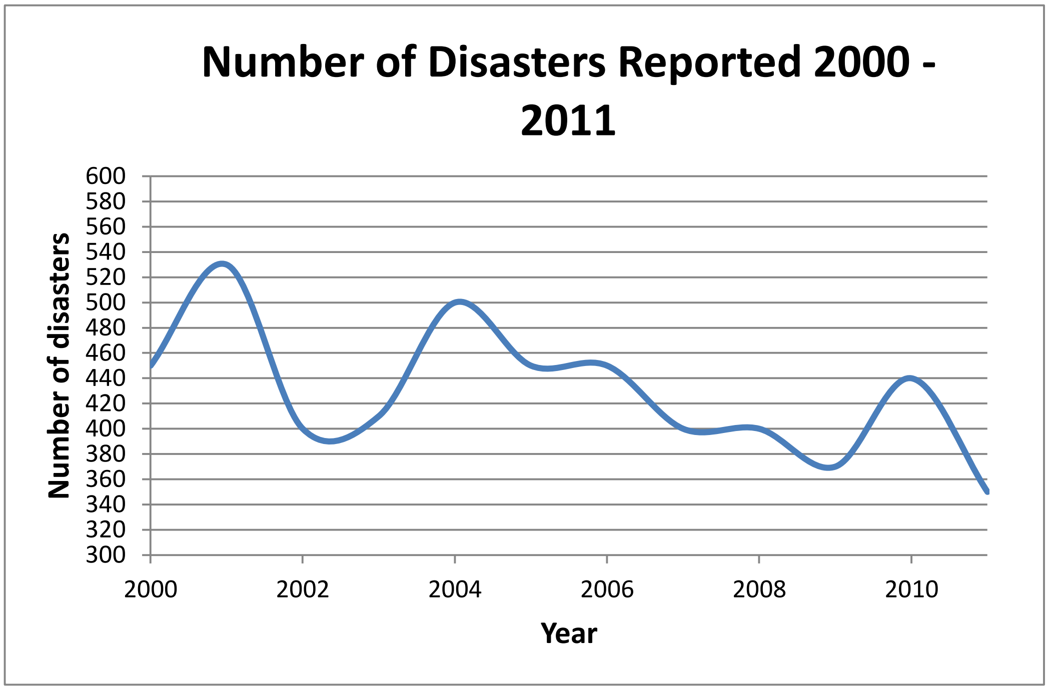


Figure 2. Natural Hazards Reported between 2000 and 2011 (EM-DAT)

This research creates a model that identifies the interdependency between SCICI and develops a restoration sequence based on data inputs. Presented in this manuscript are: the steps required for the integration of the data, the methodology for determining the interdependencies among the SCICI with a numerical example, and a preliminary restoration model using geographic interdependency inputs.

3. Literature Review

A number of approaches demonstrating the importance of disaster restoration are evident in the literature. Existing models are highly idealized and inadequate to encompass the complexities of an actual urban environment. Moreover, current models do not consider the problem from a systems view and solutions are incremental rather than inclusive of required model elements and data components. In short, existing models lack complexity, do not identify model elements from a systems perspective, and do not have a robust data identification process (Veras and Jaller, 2012; Hale and Moberg, 2005; Horner and Widener, 2011; Ramachandran, et al., 2014) Currently, there is no method which looks at the problem from a holistic view, and every approach is based on different simplifications of a mathematical model (Moteff, and Parfomak, 2004). A sampling of current methods and their limitations for restoring SCICI is below.

Qualitative models based on stochastic processes (Patton, Gray, and Schoelles, 2003; Baldick et al., 2008; North and Macal, 2007) provide useful means to identify and analyze dependencies at a higher level, but qualitative approaches cannot scale across community size or system complexity. Input-output models (Leontief, 1987) have been used to predict economic losses due to non-availability of critical infrastructure (Rigole, and Deconinck, 2006), but do not account for interdependence. System dynamics models use a top-down approach and are generally used to study the behavior of systems (Simonsen, 2007, Sterman, 2002), but they are cumbersome and lack the fine detail required for robust solutions. Continuous and discrete modeling techniques are based on mathematical designs (Liu, 1999). These models can be used to develop restoration strategies for individual damaged infrastructures, but quickly prove ineffective with complex systems. Topological and complex network models identify system structures, but fail to identify system characteristics in a useable manner (Schläpfer, Kessler, and Kroger, 2008). Simulation models are often used when analytical solutions are not possible, but simulation cannot identify all possible states (Pederson et al., 2006).

The vast majority of research after an extreme event focuses on facility location, inventory management, resource distribution strategies, or on estimation of short-term resource requirements for emergency response (see, for example, Veras and Jaller (2011); Akkihal (2006); Duran et al. (2011); Balcik and Beamon (2008); Ozbay and Ozguven (2007); Jaller et al. (2007); Altay and Green (2006); Rawls and Turnquist (2010); Mete and Zabinsky (2010); Simpson and Hancock (2009)).

Geospatial data is used in many hazard studies to detail changes between pre- and post-disaster imagery. Tornado damage assessment studies include Wagner, Myint, and Cerveny, 2012; Yuan, Dickens-Micozzi, Magsig, 2002; Myint, Yuan, and Cerveny, 2008; Jedlovec, Nair, and Haines, 2006. Post-disaster damage assessments resulting from wildfires, hurricanes, tsunami include Splinter, Strauss, and Thomlinson, 2011; Rodgers III, Kessler, and Kroger, 2012; Barnes, Fritz, and Yoo, 2007. Disaster impacts using a normalized difference vegetation index include Bentley, Mote, and Thebepanya, 2002; Wilkinson, and Crosby, 2010.

Virtually all the data used in previous studies are static and out-of-date in terms of future disasters, or synthetic. Yet, accurate, real-time data are essential for creating the level of complexity and interdependencies maps that are necessary to construct the models. Local, regional and national planners would have access to their own restricted data sets, but lack tools that can ingest these data and then provide restoration strategies. This research discusses the creation of models for SCICI restoration that can ingest real-time, publically available data, and then presents a methodology for identifying and analyzing the interdependency between SCICI. Specifically, internal geographical relationships (herein called the ‘geographical interdependency’) of SCICI are mapped. These interdependencies highlight the stress points on the larger SCICI where failures occur and are not included in current built environment models. The mapping of these interdependencies is a key step forward in attempts to optimally restore an urban center’s supply chain in the wake of an extreme event.

4. Data Requirements

Previous critical infrastructure modeling falls into one of three categories: the modeling of a single infrastructure system, such as transportation, electricity, communications or water (Gillette et al., 2002; NISAC, 2011; Shih et al., 2009); the assumption that all necessary data are hypothetically available at the time needed (Lee et al., 2005; Tolone et al., 2004); or the generation of synthetic data on which a model is built (Adachi and Ellingwood, 2008; Lewis et al., 1979). While each of these approaches have strengths and avoid the difficult task of large-scale data integration of SCICI component data, they each have significant limitations. The single system approach may have a complete real-world data set of its own system, but it does not properly define this system’s interaction with other systems. Whereas models that assume either all data is available at the time required or that generate synthetic data have also implicitly assumed knowledge of all interactive properties that exist between systems. Necessarily, they do not have the ability to evolve or adapt to changing circumstances, and therefore lack an understanding of the complex and interconnected nature of the SCICI.

The complex and interconnected nature of SCICI are coined as the ‘interdependencies’ within the SCICI. Figure 3 shows a cartoon representation of some supply chain network elements and their interdependencies. An illustrative example of such interdependencies and a cascading failure might involve: a failure in a communication relay leads to the overheating and failure of a water pump providing coolant to a power plant that destroys a boiler, shutting down the plant, and overtaxing the electrical grid. This could lead to a widespread blackout, communications shutdowns, transportation strictures, financial distress and civil unrest. While the initial failure here is on a micro-scale, the illustrative point of the importance of understanding the interconnectivity of the various SCICI is made.

Rinaldi et al. (2001) categorize interdependencies among infrastructure systems into one of four types: *Physical interdependency,* physical reliance on material flow from one infrastructure to another, *Cyber interdependency*, the existence of information transfer between infrastructures, *Logical interdependency*, any other type in interdependency that exists between infrastructures that do not fall in one of the other categories, and *Geographic interdependency,* infrastructures that are located in close proximity with each other. In this study, SCICI data are used to map the latter interdependency between SCICI into a viable Supply Chain Network (SCN) model.

In order to create a SCICI model with sophistication sufficient to illuminate the various interdependencies across systems, a large amount of real-world data needs to be acquired, integrated and analyzed.

An example of some of the types of data needed is shown for the transportation SCICI in Table 1. The components of this table are extensively discussed by Ramachandran et al. (2015). A satisfactory model of SCICI would require data such as these.

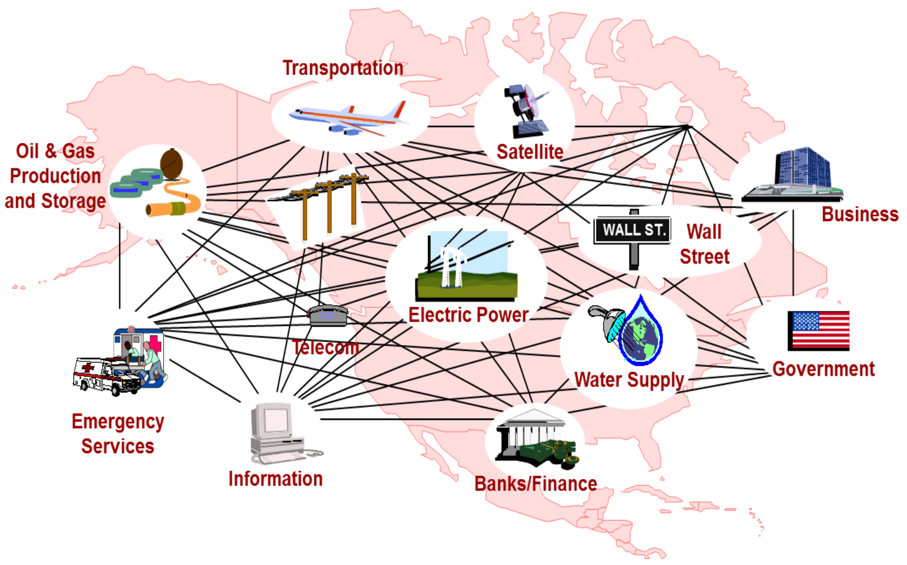


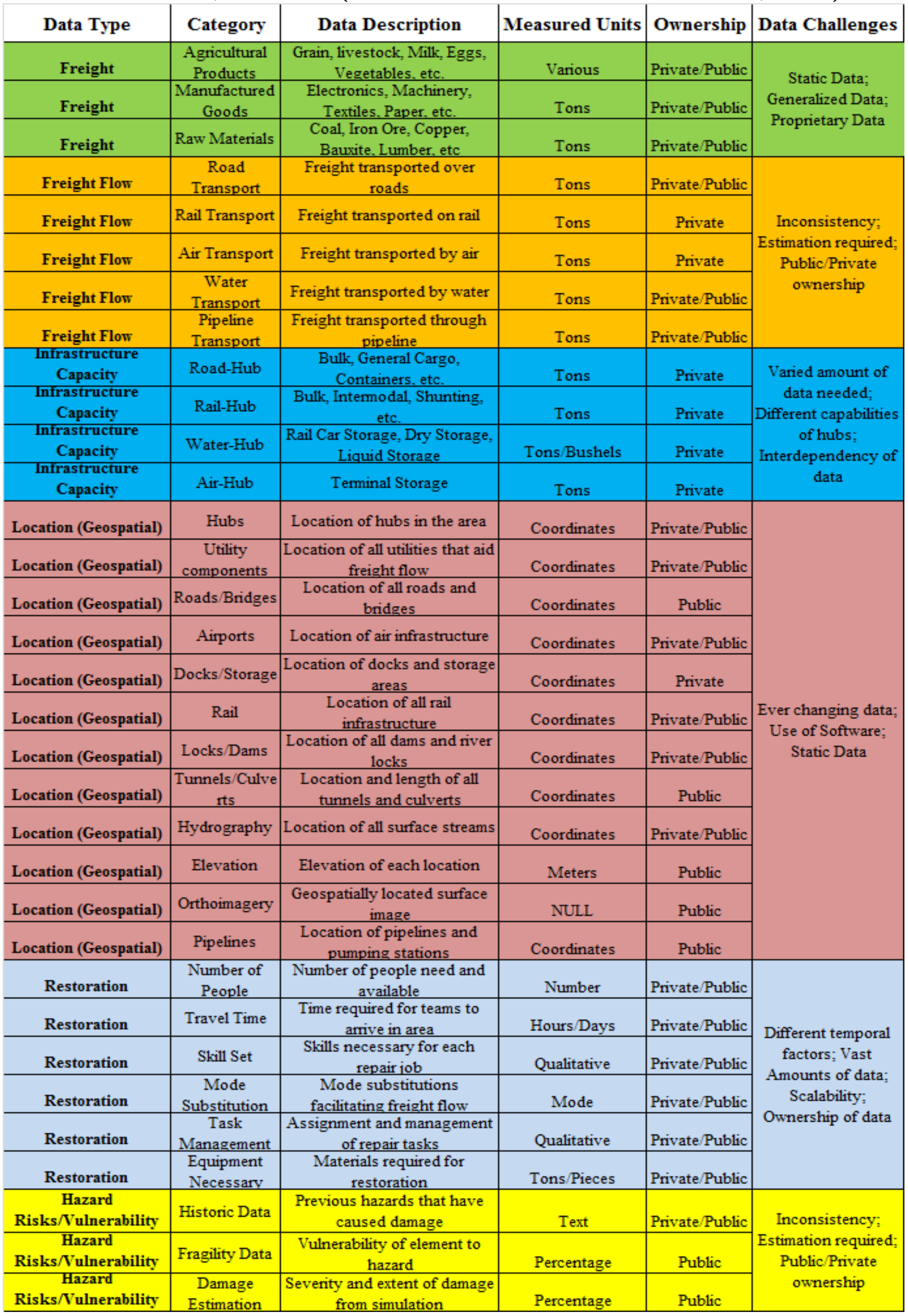
Figure 3. Cartoon illustrating the interdependent nature of critical infrastructure elements

Geographical interdependency is driven by proximity and approachability. It does not constitute a physical connection (as does physical interdependency), but does require that one element be geographically near another and that this element can be approached from the other by reasonable means. The obvious platform for the integration and analyses of these data prior to model building is geospatial. In this study, The National Map of the U.S. Geological Survey is chosen as the geospatial platform, and all other data elements are integrated onto the orthoimagery from this source. The integration is done in a Geographic Information System (GIS) environment, but it should be pointed out that both the platform and the integration environment are chosen for convenience and other systems with the same capability could perform the same service.

5. Example

The study area is represented by 29 USGS 7.5-minute quadrangles for the greater St. Louis region of Missouri and Illinois (Rogers, 2009) covering an area of 4,432 km^2 (Figure 4), and was chosen due to its proximity to the New Madrid Seismic Zone (NMSZ), in the Mississippi Embayment. This fault zone is about 240 km long and occurs from five to twenty-four kilometers beneath the earth's surface (Newman et al., 1999). The area is a source of considerable small-scale seismic activity today. Although the most recent large earthquake (estimated magnitude about 7.5) occurred in 1811 – 1812, the potential destruction due to a major earthquake in this region remains high (Tuttle et al, 2002). The area is also subject to tornadoes, particularly during the late spring through early fall months and, due to its proximity to the confluence of the Missouri and Mississippi Rivers, flooding. Indeed, should a disaster claim all the major bridges in this area, the city itself would become a virtual island in terms of transportation issues.

Table 1. Transportation data requirements for modeling SCICI in an urban environment like St. Louis, Missouri (modified from Ramachandran et al., 2015).



Data Type	Category	Data Description	Measured Units	Ownership	Data Challenges
Freight	Agricultural Products	Grain, livestock, Milk, Eggs, Vegetables, etc.	Various	Private/Public	Static Data; Generalized Data; Proprietary Data
Freight	Manufactured Goods	Electronics, Machinery, Textiles, Paper, etc.	Tons	Private/Public	
Freight	Raw Materials	Coal, Iron Ore, Copper, Bauxite, Lumber, etc	Tons	Private/Public	
Freight Flow	Road Transport	Freight transported over roads	Tons	Private/Public	Inconsistency; Estimation required; Public/Private ownership
Freight Flow	Rail Transport	Freight transported on rail	Tons	Private	
Freight Flow	Air Transport	Freight transported by air	Tons	Private	
Freight Flow	Water Transport	Freight transported by water	Tons	Private/Public	
Freight Flow	Pipeline Transport	Freight transported through pipeline	Tons	Private/Public	
Infrastructure Capacity	Road-Hub	Bulk, General Cargo, Containers, etc.	Tons	Private	Varied amount of data needed; Different capabilities of hubs; Interdependency of data
Infrastructure Capacity	Rail-Hub	Bulk, Intermodal, Shunting, etc.	Tons	Private	
Infrastructure Capacity	Water-Hub	Rail Car Storage, Dry Storage, Liquid Storage	Tons/Bushels	Private	
Infrastructure Capacity	Air-Hub	Terminal Storage	Tons	Private	
Location (Geospatial)	Hubs	Location of hubs in the area	Coordinates	Private/Public	Ever changing data; Use of Software; Static Data
Location (Geospatial)	Utility components	Location of all utilities that aid freight flow	Coordinates	Private/Public	
Location (Geospatial)	Roads/Bridges	Location of all roads and bridges	Coordinates	Public	
Location (Geospatial)	Airports	Location of air infrastructure	Coordinates	Private/Public	
Location (Geospatial)	Docks/Storage	Location of docks and storage areas	Coordinates	Private	
Location (Geospatial)	Rail	Location of all rail infrastructure	Coordinates	Private/Public	
Location (Geospatial)	Locks/Dams	Location of all dams and river locks	Coordinates	Private/Public	
Location (Geospatial)	Tunnels/Culverts	Location and length of all tunnels and culverts	Coordinates	Public	
Location (Geospatial)	Hydrography	Location of all surface streams	Coordinates	Private/Public	
Location (Geospatial)	Elevation	Elevation of each location	Meters	Public	
Location (Geospatial)	Orthoimagery	Geospatially located surface image	NULL	Public	
Location (Geospatial)	Pipelines	Location of pipelines and pumping stations	Coordinates	Public	
Restoration	Number of People	Number of people need and available	Number	Private/Public	Different temporal factors; Vast Amounts of data; Scalability; Ownership of data
Restoration	Travel Time	Time required for teams to arrive in area	Hours/Days	Private/Public	
Restoration	Skill Set	Skills necessary for each repair job	Qualitative	Private/Public	
Restoration	Mode Substitution	Mode substitutions facilitating freight flow	Mode	Private/Public	
Restoration	Task Management	Assignment and management of repair tasks	Qualitative	Private/Public	
Restoration	Equipment Necessary	Materials required for restoration	Tons/Pieces	Private/Public	
Hazard Risks/Vulnerability	Historic Data	Previous hazards that have caused damage	Text	Private/Public	Inconsistency; Estimation required; Public/Private ownership
Hazard Risks/Vulnerability	Fragility Data	Vulnerability of element to hazard	Percentage	Public	
Hazard Risks/Vulnerability	Damage Estimation	Severity and extent of damage from simulation	Percentage	Public	

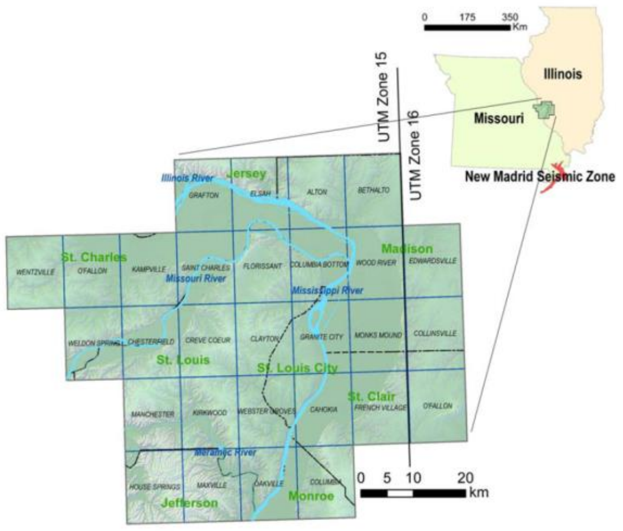


Figure 4. The study area, the greater St Louis, metropolitan area with USGS 7.5' topographic quadrangle coverage (Rogers, 2009).

6. Data Acquisition and Preprocessing

SCICI geospatial data required for the construction of public SCN models that will identify and catalog geographical interdependencies must necessarily come from several open sources. One such source is The National Map of U.S. Geological Survey which distributes fully integrated layers of orthoimagery, elevation, hydrology, transportation, place names, and land cover (Sugarbaker, and Carswell, 2011). The orthoimagery is a particularly rich data source as it consists of aerial photographs that have been mathematically corrected to remove camera distortions and flight path variations ('orthorectified'), thereby producing images of uniform scale that allow accurate determination of coordinates, distances, areas, shapes, directions, and land usages from these images (Mauck et al., 2009). In this study area 2268 orthoimagery tiles from The National Map were downloaded for total coverage. These images have resolutions ranging from 0.15 m up to 0.6 m cell-lengths. From these images it is possible to extract infrastructure elements such as bridges, culverts, docks, dams, electric poles, electric substations, fire hydrants, power plants, storm drains, water reclamation plants and more by heads up digitization. A third source of public SCICI data include the state departments of transportation, whose road and rail data best integrate with the road and rail depictions on the orthoimagery. An example of the SCICI geospatial data compiled for a section of the St. Louis metroplex is shown in Figure 5.

Elevation data is used in this study to calculate the ‘approachability’ of SCICI, and consists of the National Elevation Dataset (NED) digital elevation models (DEM) for this region (Figure 6). These are interpolated elevation grids have been based largely on topographic map contour data. The highest resolution DEMs were chosen which consisted mostly of 1/9 arc-second data (3 m cells) for the greater St. Louis county area, and 1/3 arc-second data (10 m cells) for the rest of the area.

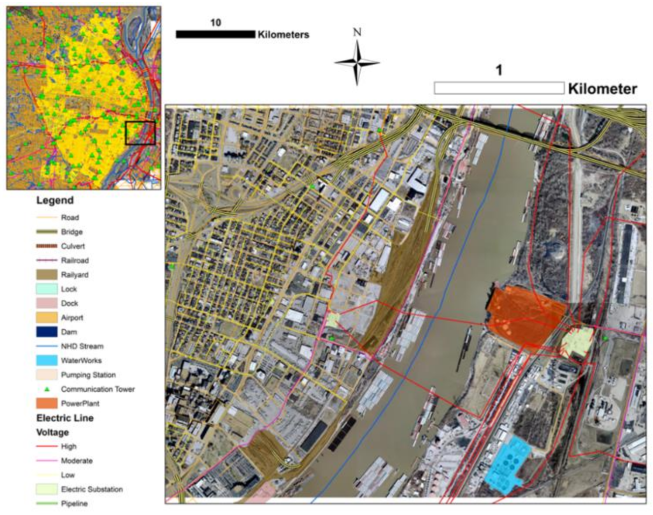


Figure 5. Orthoimagery and selected SCICI infrastructure for St. Louis, Missouri region (Ramachandran et al., 2015).

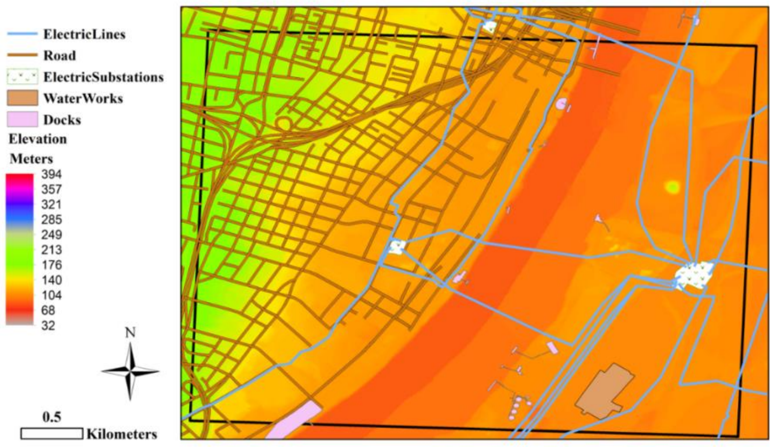


Figure 6. Elevation DEM integrated with road networks for the central United States.

While other data sources are necessary for the modeling of physical, cyber and logical interdependencies, the sources described here allow for the development and implementation of algorithms that can map geographical interdependency among SCICI.

7. Algorithm for Mapping Geographic Interdependency

Infrastructures are said to be geographically interdependent if they are within a close proximity and are able to establish a connection between each other. The elevation data can be used as a feasibility criterion to test for physical connectivity between infrastructure elements. A large-scale disaster would most likely cause a change of state to all the infrastructure elements that are close to each other. For this research, geographic interdependency is studied in two parts: calculating the nearest neighbor and calculating infrastructure element approachability. Figure 7 illustrates the procedure adopted for mapping the geographic interdependency between elements of the SCICI that are within a given threshold distance of each other using a nearest neighbor algorithm.

The process for applying the nearest neighbor algorithm is made up of three stages: Infrastructure Data Loading (IDL), Infrastructure Data Cleaning (IDC), and Geographic Interdependency Mapping Proximity (GIMP). The IDL stage (Line 1 of Figure 7) establishes connection to a SQL® database[1]. SQL, or Structured Query Language, is a special purpose programming language designed for managing and processing data. The advantage of using the SQL database is that a geospatial data add-on is available that supports geography data types and can store spatial data in tables (in the form of points, lines and polygons). Queries can then be written to analyze and manipulate the data stored within the tables. The IDC (Line 3 of Figure 7) stage is a data integration phase where all the geospatial data that are required to map the interdependencies are converted into a format that is readable by the SQL database. For this, GDAL (an open-source translator library) is used to convert the raster and vector geospatial data. Using this translator, a query is written to convert the existing spatial metadata into database readable format so that the data can be analyzed when needed. This is an important step because when the data gets transferred to a database all the attributes of the data are converted into columns and can be queried as individual items. The GIMP (Line 11 of Figure 7) specifies rules for mapping the proximity interdependency. The GIMP stage determines if the infrastructure table exists, and if not, creates it. From this table a hash map and spatial index are compiled which reduces subsequent computing time.

[1]Reference to any specific commercial product, process, or service by trade name, trademark, manufacture, or otherwise does not constitute an endorsement, a recommendation, or a favoring by the U.S. government or the U.S. Geological Survey

1:	Load all the necessary shapefiles and the descriptor ≥IDL Stage
2:	**Using Gdal ogr2ogr convert the shapefile into database format**
3:	Cleanse the data (EPSG 4269) ≥IDC Stage
4:	Create a new table with required fields and correct data types
5:	Start a new stored procedure
6:	**while** table exists
7:	Alter and update table
8:	**Set**
9:	Inner Join using geometry location
10:	Create hash table and spatial index for each table for faster join
11:	**Select** column from <tablename> **and specify join rules** ≥ GIMP Stage
12:	**Insert stored procedure for Nearest Neighbour algorithm**
13:	Compute distance for each infrastructure element
14:	Compute nearest infrastructure within threshold and store results
15:	**End set**
16:	Continues till all elements of SCICI are traversed
17:	**End while**
18:	**Update** table
	Do the same for other infrastructure and create spatial results

Figure 7. Steps for systemic geographic interdependency mapping for finding nearest neighbors

A spatial index is a type of extended index containing data of a single spatial type (such as geometry or geography). In this implementation the spatial index is built using R-trees. R-trees span a 2-dimensional space, which in this case decompose the data into a four-level grid hierarchy, thereby creating the spatial index. Hence, all the data are stored in an overlapping grid hierarchy making it easier to query or retrieve. In this manner, a spatial index for each infrastructure element can be created to speed the recovery of interdependency information.

A nearest neighbor algorithm is then implemented to find the nearest infrastructure elements to every element in the database. For example, consider bridges, the nearest neighbor algorithm calculates the distance from a particular bridge element to the nearest communication tower, electricity substation, electric grid line, dock and so forth for all infrastructure elements using geographic locations. A threshold radius within which to perform the search is chosen based on the type of infrastructure element queried for the area. For example, a one kilometer radius would be reasonable for selecting the nearest road, electric grid line or water main, whereas ten kilometers would be more reasonable for electric substations, water pumping stations, docks, and so forth. With this threshold each infrastructure element is traversed and if an element is found it is updated in the corresponding table. This process of identification of elements continues until all elements have been traversed.

Table 2 shows an example of the output after implementing the algorithm in Figure 7. The *location* entries give the geographic position of every bridge within the search area. The remaining columns identify the nearest infrastructure element (it this case, electric substations, docks, and communication towers) to each bridge.

The procedure used to map geographic interdependencies using elevation data is described in Figure 8. This method modifies the nearest neighbor method shown in Figure 7 to include elevation data to determine the feasibility of a road connecting two infrastructure elements.

Table 2. Results of finding nearest neighbor

Location	**Nearest Electric Substation**	**Nearest Dock**	**Nearest Cell Tower**
0xAD1000000114E2B	Substation_023	Dock_011	Cell_012
0xAD10000001140EE	Substation_023	Dock_011	Cell_023
0xAD1000000114B48	Substation_102	Dock_023	Cell_024
0xAD10000001142D	Substation_024	Dock_023	Cell_025
0xAD100000011497	Substation_024	Dock_023	Cell_026
0xAD100000010404	Substation_024	Dock_056	Cell_045
0xAD1000000114F4	Substation_051	Dock_057	Cell_028
0xAD1000000114AC	Substation_051	NULL	Cell_029
0xAD100000011404	Substation_051	Dock_052	Cell_032

The new algorithm, referred to as an ‘approachability function’, is defined as a maximum slope beyond which one SCICI element cannot be reached from the other SCICI element in its vicinity. In essence this criterion gives insight into whether a SCICI element can be connected to or repaired from another.

1:	Overlay the SCICI data from TNM and also the extracted from TNM over the DEM data
2:	Find elevation of each point using a 3D profile and extract the metadata ≥IED Stage
3:	Load all the necessary shapefiles and the descriptors ≥IDL Stage
4:	Using Gdal ogr2ogr convert the shapefile into database format
5:	Cleanse the data by georefrencing it, and making them into the same geometry (EPSG 4269)
6:	Create a new table with required fields and correct data types ≥IDC Stage
7:	Start a new stored procedure
8:	**while** table exists
9:	Alter and update table
10:	**Set**
11:	Inner Join using geometry location
12:	Create hash table and spatial index for each table for faster join
13:	**Insert Stored Procedure:** Algorithm 1 find nearest neighbor
14:	**If** neighbor found, **Then** check slope value
15:	**If** slope<300m/km checkbox **YES**
16:	**Else** Checkbox **NO**
17:	**Else** ≥GIMA Stage
18:	**End set**
19:	Continues till all elements of SCICI are traversed
20:	**End while**
21:	**Update** table
	Do the same for other infrastructure and create spatial results

Figure 8. Steps for systemic geographic interdependency mapping for finding an approachability function

One straightforward illustration of this connection is whether utilities crews who need to reach an electric grid line for repair can access the line from a particular road segment after a disaster. Infrastructure elements may be in the same vicinity (proximity), but this does not mean that they are approachable (approachability).

The approachability determination algorithm has four steps: Infrastructure Elevation Data (IED), Infrastructure Data Loading (IDL), Infrastructure Data Cleaning (IDC), and Geographic Interdependency Mapping Accessibility (GIMA). Two of these, IDL (Line 3 of Figure 8) and IDC (Line 6 of Figure 8) are similar to the method described in Figure 7. The IED (Line 2 of Figure 8) step is a geospatial data integration stage, where elevation data (derived from the NED) is overlain on SCICI location data from TNM and projected into the same Universal Transverse Mercator projection as the TNM orthoimagery. The next step is to find the elevation for each feature representing a SCICI element. A profile is then created by adding surface information to find the elevation change (Z-value), and the slopes associated with each. Once, these values are obtained, a connection is established in the database. The GIMA (Line 17 of Figure 8) step calculates '*approachability function'*. The data are loaded into the database and nearest neighbors are found to each SCICI element. For each neighbor that is found, its slope and elevation values are checked to determine if it is above a given threshold.

Table 3 shows an example of the elevation and the slope information for a sample of the roads in the area under consideration. The slope is found by splitting the lines at vertices (starting and ending) and determining its length in kilometers (km) and creating a surface profile to find the Z-value (elevation change in meters, m). The slope then will be recorded in m/km.

Table 4 gives example results of the method when the approachability function is implemented showing how the interdependencies can be mapped. In this table, the Approachable column is a binary which shows 'Yes' if the element of infrastructure is approachable from that particular road, which means it satisfies that criteria for the slope threshold, and it shows 'No' if the element does not satisfy the slope threshold criteria. The maximum slope threshold for an interstate road in the United States should not exceed 8% grade (about 80m/km, Aashto, 2001) irrespective of the speed limit. Since, this research looks at all the different types of roads (interstates, US highway, State Routes) the slope threshold value is set higher at 30% grade (about 300 m/km) which is closer to the maximum slope of a known road in the United States (370m/km, Aashto, 2001). To calculate the *'approachability function'* the horizontal distance from the nearest road to a particular SCICI element is found. Then, the □ value (elevation change) is used to calculate the slope from the road to the SCICI element, and if this slope is less than 300m/km (threshold value), then the SCICI elements is deemed approachable from that road. The '*location*' column in the table gives the location of the roads in the area under consideration. The nearest electric substation ('Nearest_ESub'), nearest Dock ('Nearest_Dock'), and the nearest communication tower ('Nearest_CellT') are calculated for each road and a snippet is shown Table 4.

The methods developed and proposed here are robust and flexible. This is an important result because different terrains, features and modes of disaster will require different types of interdependency modeling.

Table 3. Elevation and Slope data for Road infrastructure of SCICI

Road Type	**Length (km)**	**Z Value (m)**	**Slope (m/km)**
US ROUTE_32	3.65	195.57	53.6
STATE ROUTE_21	36.00	210.07	5.8
INTERSTATE_23	0.70	212.28	303.3
STATE ROUTE_038	0.58	234.04	403.5
INTERSTATE_23	2.43	641.73	263.9
STATE ROUTE_12	3.78	175.31	46.4

Table 4. Example Results of applying 'approachability function'

Location	**Nearest Substation**	**Approach-able**	**Nearest Dock**	**Approachable**	**Cell Tower**	**Approachable**
114E2BCFE8B8F5	Sub_023	Yes	Dock_011	Yes	Cell_012	Yes
1140EED38156C5	Sub_023	Yes	Dock_011	Yes	Cell_023	Yes
114B48420F0E75	Sub_102	Yes	Dock_023	Yes	Cell_024	Yes

Table 4. Example Results of applying ‘approachability function’ (cont.)

1142D5E7836965	Sub_024	Yes	Dock_023	Yes	Cell_025	Yes
114973E5D282E5	Sub_024	Yes	Dock_023	Yes	Cell_026	No
10404000000ECC	Sub_024	Yes	Dock_056	Yes	Cell_045	No
114F44DE0F2805	Sub_051	Yes	Dock_057	Yes	Cell_028	No
114ACBE1D60965	Sub_051	Yes	NULL	No	Cell_029	No
1140484A684995	Sub_051	Yes	Dock_052	No	Cell_032	No
1149C98112BC85	NULL	Yes	Dock_006	No	Cell_031	Yes

8. Conclusion and Future Works

There are two major findings of this research. This work shows that there is sufficient data publically available to create a near real-world modeling scenario for infrastructure elements. Some limitations exist, such as the static nature of the geospatial data and the amount of estimation required while interpreting transportation data. Understanding these bounds, the results of this research show that there is sufficient data available in public domain to create a model with sufficient complexity to assist with decision making with periodic updates as infrastructure changes.

The second finding of this research is the demonstration of methods for interdependency mapping. This research focuses on geographic interdependency, but similar types of algorithm can be implemented to determine other types of interdependency (physical, cyber, and logical). Integrating geospatial data with freight flow and infrastructure, and combining these with restoration and hazard data is a complex task. This complexity arises mainly due to the interdependent nature of infrastructure systems. Most of the modeling techniques previously studied have either ignored the interdependent nature of critical infrastructure and have looked at only one infrastructure, or have assumed that the use of synthetic data can mimic real-world scenarios sufficiently, which is not the case.

The modeling technique presented here utilizes data provided by The National Map (orthoimagery, elevation etc.) to identify the location of the SCICI, find the proximity between them, and also develop an approachability function. The spatial information is used to identify relationships that exist between the elements of SCICI; this paves the way to understand the complex nature of these systems. Mapping and understanding geographic interdependency is essential when trying to model real-world scenarios.

This research is an important step in understanding the restoration of critical infrastructure after damage due to natural or man-made disaster. Protecting critical infrastructures remains a difficult and an open problem, made more complex as there is not a clear understanding of the interdependencies that exists among the infrastructures. Better understanding of these interdependencies will lead to a heuristic for the restoration process.

The advantage of this methodology is that it is scalable, and flexible, i.e. the same model can be used for different regions and different infrastructure elements if the data are available. The methodology proposed in this work contributes to the literature by its explicit combination of modeling infrastructure elements using real data and mapping interdependencies between them. Previous research considers a synthetic area, only looks at a particular infrastructure, or does not provide a comprehensive framework to model and map the interdependencies.

The next step in this research will address limitations of the research with respect to the data. Future work will increase the level of detail and robustness of the data. LiDAR and elevation data can be used to better approximate real-world scenarios. The use of GPS data alone can create problems when looking at something that is not at the same height. Moreover, the use of semantic ontology should aid with the integration of data. The current data are from different sources in different formats. Semantics will greatly help with understanding the data and finding trends within the data. Because of the size and variety of the data, future work will also consider reducing the computing time. The 29 size block area that was considered for this research required more than 7 Tb of data to describe it properly. Big-data analytics and parallel processing techniques will likely prove useful in the development of required datasets and usable restoration tools.

9. References

Adachi, T., B. Ellingwood. (2008). "Serviceability of earthquake-damaged systems: effects of electrical power availability and back-up systems on system vulnerability". *Reliability Engineering System Safety. 93(1), 78–88.*

Akkihal, A. R. (2006). "Inventory pre-positioning for humanitarian operations", *Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA.*

Altay, N., W. G. Green III. (2006). "OR/MS research in disaster operations management". *European Journal of Operational Research*, 175(1), 475-493.

Balcik, B., B. M. Beamon. (2008). Facility location in humanitarian relief. *International Journal of Logistics*, *11*(2), 101-121.

Baldick, R., B. Chowdhury, I. Dobson, Z. Dong, B. Gou, D. Hawkins, and Zhang, X. (2008). "Initial review of methods for cascading failure analysis in electric power transmission systems IEEE PES CAMS task force on understanding, prediction, mitigation and restoration of cascading failures". *Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE* (pp. 1-8). IEEE.

Barnes, C. F., H. Fritz, and J. Yoo. (2007). "Hurricane disaster assessments with image-driven data mining in high resolution satellite imagery," *IEEE Transaction on Geoscience and Remote Sensing*, vol. 45, no. 6, pp. 1631–1640.

Bentley, M. L., T. L. Mote, and P. Thebepanya. (2002). "Using Landsat to identify thunderstorm damage in agricultural regions," *Bulletin of American Meteorological Society*, vol. 83, no. 3, pp. 363–376.

Blake, E. S., T. B. Kimberlain, R. J. Berg, J. P. Cangialosi, and J. L. Beven II. (2013). "Tropical Cyclone Report: Hurricane Sandy". *National Hurricane Center, 12*.

Department of Homeland Security (DHS). (1996). *The White House, United States Government.1996. Executive Order EO 13010*. Critical Infrastructure Protection Retrieved from: http://fas.org/irp/offdocs/eo13010.htm. Last accessed on 12/2/2014.

Duran, S., M. A. Gutierrez, and P. Keskinocak. (2011). "Pre-positioning of emergency items worldwide for CARE International". *Interfaces, 41(3), 223–237.*

EM-DAT: The OFDA/CRED International Disaster Database. Université Catholique de Louvain. Brussels, Belgium. Available at: http://www.cred.be/emdat.

Gillette, J., R. Fisher, J. Peerenboom, R. Whitfield. (2002). "Analyzing Water/Wastewater Infrastructure Interdependencies". *6th International Conference on Probabilistic Safety Assessment and Management, San Juan (PR),* June 23-28. Available on-line: http://www.dis.anl.gov/pubs/42598.pdf.

Holguín-Veras, J., and M. Jaller. (2011). "Immediate resource requirements after hurricane Katrina". *Natural Hazards Review, 13*(2), 117-131. DOI 10.1061/(ASCE)NH.1527-6996.0000068.

Jaller, M., S. Ukkusuri, J. Holguín-Veras. (2007). "A stochastic humanitarian inventory model for fixed lifetime goods for disaster planning. *INFORMS Annual Meeting, Hanover, MD.*

Jedlovec, G. J., U. Nair, and S. L. Haines. (2006). "Detection of storm damage tracks with EOS data," *Weather Forecasting*, vol. 21, no. 3, pp. 249–267.

Lee, E., A. Wallace, J.E. Mitchell and D. Mendonça. (2005). "Decision technologies for protection of critical infrastructures". *In Proceedings of Working Together: R&D Partnerships in Homeland Security, Boston*. Available via: http://www.rpi.edu/~mitchj/papers/decisiontechnologies.html.

Leontief, W. (1987). "Input-output analysis". *The new palgrave. A dictionary of economics, 2*, 860-64.

Lewis, H.W., R. J. Budnitz, W. D. Rowe, H. J. C. Kouts, F. von Hippel, W. B. Loewenstien. (1979). “Risk assessment review group report to the US Nuclear Regulatory Commission”. *IEEE Transactions Nuclear Science 1979.* NS-26: 4686–90.

Liu, J., X. Liu, T. K. J. Koo, B. Sinopoli, S. Sastry, and E. A. Lee. (1999). “A hierarchical hybrid system model and its simulation”. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on* (Vol. 4, pp. 3508-3513). IEEE.

Long, S., T. Shoberg, V. Ramachandran, S.M. Corns, and H.J. Carlo. (2013). “Integrating complexity into data-driven multi-hazard supply chain network strategies”. *Proceedings of the ASPRS\CaGIS 2013 Specialty Conference*, San Antonio,TX, Available at: http://info.asprs.org/publications/proceedings/SanAntonio2013/index.htm.

Mauck, J., K. Brown, W. J. Carswell Jr. (2009). *The National Map*—Orthoimagery: U.S. Geological Survey Fact Sheet 2009-3055, 4 p.

Mete, H. O., and Z. B. Zabinsky. (2010). “Stochastic optimization of medical supply location and distribution in disaster management”. *International Journal of Production Economics*, *126*(1), 76-84.

Moteff, J., and P. Parfomak. (2004). Critical infrastructure and key assets: definition and identification. Technical report, CRS, Report for Congress.

Myint, S., M. Yuan, R. Cerveny, and C. Giri. (2008). “Comparison of remote sensing image processing techniques to identify tornado damage areas from Landsat TM data,” *Sensors*, vol. 8, no. 2, pp. 1128–1156.

National Infrastructure Simulation & Analysis Center. Available via: http://www.sandia.gov/nisac/. Accessed on April 29, 2014

Network Optimization Models. National Infrastructure Simulation & Analysis Center. Available via: http://www.sandia.gov/nisac/net_op.html [accessed November 14, 2014].

Newman, A., S. Stein, J. Weber, J. Engeln, A. Mao, A.and T. Dixon. (1999). “Slow deformation and lower seismic hazard at the New Madrid seismic zone”. *Science, 284*(5414), 619-621.

North, M. J., and C. M. Macal, (2007). “Managing business complexity: discovering strategic solutions with agent-based modeling and simulation”. *Oxford University Press.* New York, NY, USA, 39.

Ozbay, K., and E. Ozguven. (2007). “Stochastic humanitarian inventory control model for disaster planning”. *Transportation Research Record No. 2022, Transportation Research Board, Washington, DC, 63–75.*

Patton, E. W., W. D. Gray, and M. J. Schoelles. (2009). “SANLab-CM–The Stochastic Activity Network Laboratory for Cognitive Modeling”. Schläpfer (Vol. 53, No. 21, pp. 1654-1658). SAGE Publications.

Pederson, P., D. Dudenhoeffer, S. Hartley, and M. Permann. (2006). "Critical infrastructure interdependency modeling: a survey of US and international research". *Idaho National Laboratory,* 1-20.

Ramachandran, Varun, Long, Suzanna, Shoberg, Tom, Corns, Steven M. and Carlo, Hector J. (2015) “Post-disaster Supply Chain Interdependent Critical Infrastructure System Restoration Modeling: A Review of Necessary Data” Computers, Environment and Urban Systems, Submitted February 2015.

Rawls, C. G., and M. A. Turnquist. (2010). “Pre-positioning of emergency supplies for disaster response”. *Transportation research part B: Methodological, 44*(4), 521-534.

Rigole, T., and G. Deconinck. (2006). “A survey on modeling and simulation of interdependent critical infrastructures”. In *Proceedings of 3rd IEEE Benelux Young Researchers Symposium in Electrical Power Engineering* (p. 9).

Rinaldi, S. M., J. P. Peerenboom, and T. K. Kelly. (2001). “Identifying, understanding, and analyzing critical infrastructure interdependencies”. *Control Systems, IEEE, 21*(6), 11-25.

Rodgers III, J. C., A. W. Murrah, and W. H. Cooke. (2009). “The impact of Hurricane Katrina on the coastal vegetation of the Weeks Bay Reserve, Alabama from NDVI data,” *Estuaries Coasts*, vol. 32, no. 3, pp. 496–507.

Rogers, J. D. (2009). “Overview of Likely Consequences of a Magnitude 6.5+ Earthquake in the Central United States: New Madrid Seismic Zone Conference: Preparing for a Significant Central U.S. Earthquake – August 2008”. *U.S. Geological Survey Open File Report 2009-Xxxx*

Schläpfer, M., T. Kessler, and W. Kröger. (2012). “Reliability analysis of electric power systems using an object-oriented hybrid modeling approach”. In *16th Power Systems Computation Conference, Glasgow.* arXiv preprint arXiv:1201.0552.

Shih, C.Y., C.D. Scown, L. Soibelman, H.S. Matthews, J.H. Garrett Jr., K. Dodrill, and S. McSurdy. (2009). “Data Management for Geospatial Vulnerability Assessment of Interdependencies in U.S. Power Generation”. *Journal of Infrastructure Systems,* Vol. 15, No. 3, September. 179–189.

Simonsen, I., L. Buzna, K. Peters, S. Bornholdt, S., and D. Helbing. (2007). “Dynamic Effects Increasing Network Vulnerability to Cascading Failures”. *Physics Review Letters,* 100, 17-21.

Simpson, N. C., and P. G. Hancock. (2009). “Fifty years of operational research and emergency response”. *Journal of the Operational Research Society, S126-S139.*

Splinter, K. D., D. R. Strauss, and R. B. Thomlinson. (2011). “Assessment of post-storm recovery of beaches using video imaging techniques: A case study at Gold Coast Australia,” *IEEE Transaction Geoscience Remote Sensing,* vol. 49, no. 12, pp. 4704–4716.

Sterman, J. D. (2002). “Systems dynamics modeling: tools for learning in a complex world”. *Engineering Management Review, IEEE, 30*(1), 42.

Sugarbaker, L., and W. J. Carswell. (2011). “The National Map®”. US Department of the Interior, US Geological Survey.

Talukdar, S. N., J. Apt, M. Ilic, L. B. Lave, and M. G. Morgan. (2003). “Cascading failures: survival versus prevention”. *The Electricity Journal*, 16(9), 25-31. doi:10.1016/j.tej.2003.09.003.

Tolone, W. J., D. Wilson, A. Raja, W. N. Xiang, H. Hao, S. Phelps, E. W. Johnson. (2004). “Critical infrastructure integration modeling and simulation”. In *Intelligence and Security Informatics* (pp. 214-225). Springer Berlin Heidelberg.

Tuttle, M. P., E. S. Schweig, J. D. Sims, R. H. Lafferty, L. W. Wolf, and M. L. Haynes. (2002). “The earthquake potential of the New Madrid seismic zone”. *Bulletin of the Seismological Society of America*, *92*(6), 2080-2089.

U.S. Census Bureau & Bureau of Transportation Statistics. (2013). Commodity Freight Survey (CFS). Available at http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/commodity_flow_survey/index.html. Last accessed on 6/19/2014.

U.S. Bureau of Transportation Statistics. (2010). National Transportation Atlas Database (NTAD). *Research and Innovative Technology Administration., U.S. Bureau. of Transportation Statistics, U.S. Department of Transportation., Washington, D.C.* Available at http://www.bts.gov/publications/national_transportation_atlas_database/2010. Last accessed on 8/23/2014.

Wagner, M. A., S. W. Myint, and R. S. Cerveny, R. S. (2012). “Geospatial Assessment of Recovery Rates Following a Tornado Disaster”. *Geoscience and Remote Sensing, IEEE Transactions on*, *50*(11), 4313-4322.

Wilkinson, D. W., and M. K. Crosby. (2010). “Rapid assessment of forest damage from tornadoes in Mississippi,” *Photogrammetry Engineering and Remote Sensing*, vol. 76, no. 12, pp. 1298–1301.

Yuan, M., M. Dickens-Micozzi, and M. A. Magsig. (2002). “Analysis of tornado damage tracks from the 3 May tornado outbreak using multispectral satellite imagery,” *Weather Forecasting*, vol. 17, no. 3, pp. 382–398.

III. Post Disaster Supply Chain Interdependent Critical Infrastructure System Restoration Modeling: A Review of the Necessary Data

1. Abstract

The majority of restoration strategies in the wake of large-scale disasters have focused on short-term emergency response solutions. Few consider medium- to long-term restoration strategies to reconnect urban areas to national *supply chain interdependent critical infrastructure systems* (SCICI). These SCICI promote the effective flow of goods, services, and information vital to the economic vitality of an urban environment. To re-establish the connectivity between the different SCICI, relationships between these systems must be identified, formulated, and added to a common framework to form a system-level restoration plan. The aim of this paper is to review the data required for model construction, its accessibility, and integration. A review of publically available data reveals a paramount need for real-time data and information to assist urban planners with recovery following an extreme event. A particular SCICI (transportation) is used to highlight the interdependencies and challenges of creating models capable of describing the complexity of an urban environment. Integrating geospatial data derived from public domain sources such as The National Map of the U.S. Geological Survey (USGS) with supply chain data allows for the creation of more accurate models of urban center transportation networks. This review indicates that geospatial infrastructure data are the most abundant of these data. While much of it can be acquired through public sources, an effort is required to gather, develop, and integrate data from multiple sources to build a complete model. Therefore, continued availability of high quality, public information is essential.

Keywords: GIS, Extreme Events, Critical Infrastructure, Supply Chain, Logistics Network

Authors:

Varun Ramachandran

PhD Student, Department of Engineering Management & Systems Engineering, Missouri University of Science and Technology, 600 W, 14th Street, Rolla MO 65409, USA

Email address: vrnq5@mst.edu, Phone # 281/468-6769, Address: 112, East 16th Street, Rolla, MO 65401

Dr. Suzanna Long*

Associate Professor, Department of Engineering Management & Systems Engineering, Missouri University of Science and Technology, 600 W, 14th Street, Rolla MO 65409, USA

Email address: longsuz@mst.edu, Phone # 573/341-7621, Address: 230, Engineering Management Building, Missouri University of Science and Technology, Rolla, MO 65409

Dr. Tom Shoberg

CEGIS, U.S. Geological Survey, Rolla MO 65401, USA

Email address: tshoberg@usgs.gov , Phone # 572/308-3582, Address: 1400, Independence Drive, Rolla, MO 65401

Dr. Steven Corns

Associate Professor, Department of Engineering Management & Systems Engineering, Missouri University of Science and Technology, 600 W, 14th Street, Rolla MO 65409, USA

Email address: cornss@mst.edu, Phone # 573/341-6367, Address: 213, Engineering Management Building, Missouri University of Science and Technology, Rolla, MO 65409

Dr. Héctor J. Carlo

Associate Professor, Department of Industrial Engineering, University of Puerto Rico at Mayaguez, Call Box 9000 Mayaguez, PR 00681

Email address: hector.carlo@upr.edu, Phone # 787/832-404 x-3105, Address: University of Puerto Rico at Mayaguez, Call Box 9000 Mayaguez, PR 00681

Acknowledgments

This work was supported by the US Geological Survey under Grant G13AC00028.

2. Introduction

Critical infrastructure systems provide the backbone for socioeconomic vitality and security of urban areas. These systems are defined by the US Department of Homeland Security (DHS) as follows:

> *Critical infrastructure are the assets, systems, and networks, whether physical or virtual, so vital to the United States that their incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety, or any combination thereof* (DHS, 2014).

A supply chain interdependent critical infrastructure system (SCICI) is composed of many systems, including but not limited to: transportation, power, communications, and water, which are interdisciplinary in nature. In addition, these SCICI exhibit complex interdependencies that must be captured to create models that are representative of the true system conditions.

Effective modeling of critical infrastructure restoration must incorporate ideas and tools from a wide spectrum of research areas including: simulation-based optimization, structural engineering, human behavior modeling, geographic information systems (GIS), and supply chain management. In general, recent disaster management studies use either a qualitative (Carlson & Doyle, 1999; Haimes 2005; Amin & Wollenberg, 2005) or quantitative methodology (MacKenzie et al., 2014; Adams & Stewart, 2014). These efforts fail to capture full system complexity by not combining qualitative and quantitative methodologies and ignoring the interdependencies that lead to emergent behaviors. In addition, the majority of restoration strategies in the wake of large-scale disasters have focused on short-term emergency rescue and recovery methodologies (Holguín -Veras and Jaller, 2011; Hale and Moberg, 2005; Widener and Horner, 2011). Few consider medium- to long-term restoration strategies that reconnect urban areas to the national SCICI. The medium- to long-term restoration of these systems requires longer time lines and larger financial investments than short-term emergency response, and so a methodology specific for these phases is necessary.

A survey paper by Altay and Green (2006) found that of 110 articles relating to disaster operations management research, 43.6% relate to the mitigation phase, 21.8% focus on preparedness, 23.6% relate to response, and only 10.9% are related to recovery (12 articles). Further, most previous studies focus only on a single aspect of one system within the SCICI (Shinozuka et al., 2007; Ouyang and Dueñas-Osorio, 2011; Rosato et al., 2008), or on emergency response processes (Bruneau et al., 2006; Vugrin et al., 2010; Reed et al., 2010). A review of disaster recovery studies categorized by disaster management lifecycle do not build a comprehensive framework that identifies the data required to build such a model but assume that the data is available (Altay and Green, 2006; Kondaveti and Ganz, 2009; Feng and Weng, 2005; Miller-Hooks et al., 2012). Operations research-style quantitative research typically focuses on game theory or inventory/sourcing models (MacKenzie, et al., 2014).

To map restoration strategies of the SCICI in the aftermath of a disaster one must first build a comprehensive framework that realistically models the SCICI in a normal environment. This requires a large amount of data be integrated across many disciplines. One tool that is useful for this research is geographic information systems (GIS) technology. GIS can be used to examine the interdependency among critical infrastructure systems (Sinton, 1992) or depict geographic correlations within critical infrastructure elements (Burrough, 1990; Goodchild and Haining 2004; Greene, 2002). But a multi-dimensional approach to this modeling has yet to be considered (Mitchell, 2005; Zeiler, 2010; Openshaw, 1994).

Models required for planning the restoration of SCICI systems must capture real-world complexities and use real-time data to be useful to decision-makers. Geospatial data plays a key role in SCICI restoration; thus, there exists the need to understand accessibility issues and inherent uncertainties associated with such data. While Federal, state, and local entities routinely use GIS technology with subsets of SCICI data in disaster planning activities, using these data to map infrastructure elements, their interdependencies, and their restoration in the aftermath of an extreme event has seldom been done (Fletcher, 2002). As an important first step, this article documents the use of publically available data for the creation of complex SCICI models.

3. Method

The emphasis in modeling critical infrastructure systems has been on developing methodologies and algorithms, rather than on incorporating real-world data. Most studies have taken a one-dimensional approach wherein it is either assumed that the required data is hypothetically complete and available, or synthetic data is generated for analyses when needed. It is difficult to understand all the complex interactions that exist between infrastructure elements and systems based on such approaches. In this study, the transportation infrastructure system within SCICI is used as an example to illustrate its complex interactions with other SCICI systems and categorize, integrate, and analyze the data required to properly model this system. The transportation (logistics) infrastructure system presented here includes the transport mode (road, rail, air, and water) infrastructure, the freight that is moved through these modes, and the storage of that freight.

As with any system that forms a component of the larger SCICI system, a model of this component system must be created with the understanding that it be integrated into a larger SCICI modeling framework. The construction of a restoration model of any element of SCICI damaged due to a large-scale disaster can be divided into five work-flow phases: *acquisition and integration of data, SCICI system modeling, SCICI interdependency determinations, hazard damage simulation, and restoration modeling.* A work-flow diagram for the transportation infrastructure system is shown in Figure 1. Each phase requires different types of input data, typically in diverse formats (including non-digital formats) and stored in different databases on different computers. While this presents a challenge to the modeling effort, the identification and integration of these data are essential for creating realistic SCICI system models.

The *acquisition and integration of data phase* incorporates all data necessary to make a realistic model of the pre-disaster SCICI system for the region under consideration. For the transportation infrastructure system this would consist of: (1) freight data - storage/distribution facilities data, modes of transport and their capacity, and flow data, and (2) infrastructure data - with respect to the capacity the infrastructure can sustain and the location of each infrastructure element. Typically these data are not readily available in digital databases, may be proprietary, and/or come from multiple sources, making its integration daunting.

The *SCICI system modeling phase* combines the data from the previous phase to construct a model of the SCICI system and how it operates to perform the tasks necessary to accomplish the overall SCICI goals. The transportation infrastructure system model incorporates the freight data, system capacities, and the available transportation network from the acquisition and how it works together to move goods throughout the region being considered.

.

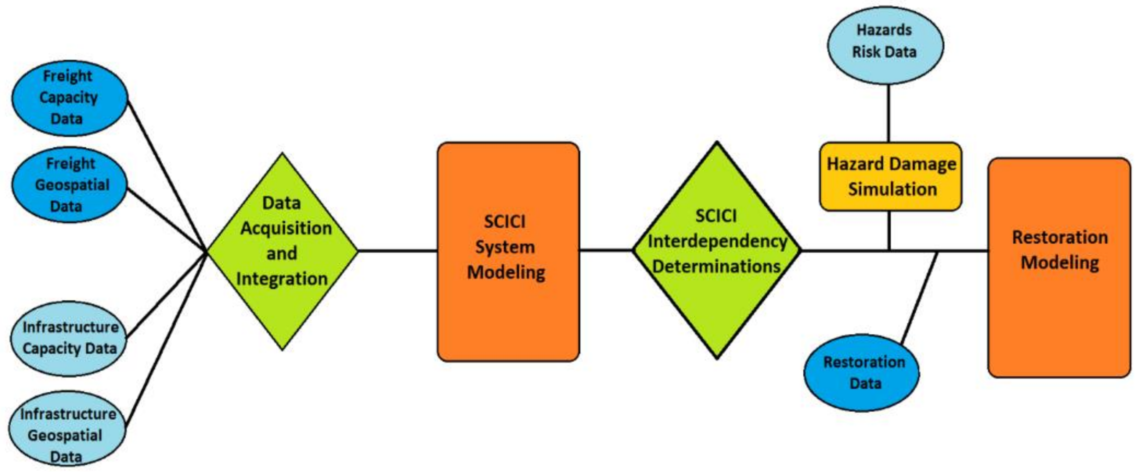


Figure 1. Schematic work flow pattern for transportation infrastructure system restoration modeling.

In the *SCICI interdependency determinations phase*, the interdependencies are mapped between SCICI systems both internally and to the external regional, national, and global supply chain elements. This is crucial to any restoration efforts. Through these interdependencies it becomes possible to detect critical points of failure that can cause a cascade effect damaging many elements upon the failure of a single element.

The *hazard damage simulation phase* gathers information related to the critical points and determines how potential hazards might affect these weak points in the SCICI. This allows for the testing of restoration modeling before the onset of a large-scale disaster. In the event of a disaster, the actual damage itself would be the input data for the restoration optimization model rather than simulated damage.

Finally in the *restoration modeling phase* scheduling and work flows are created to return the SCICI system back to the pre-event capabilities. Optimization techniques are applied here to develop plans that allows for the reassembly of the transportation system in a relatively efficient manner. In the case of the transportation system at hand, this would involve both reconnecting the transportation modes and restoring the capacity of those connections to pre-event levels.

After identifying the data required to model the SCICI systems it is necessary to acquire these data. Given the amount of data that must be collected there are several challenges. Table 1 shows data requirements for mapping the transportation system of SCICI and also identifies several difficulties in acquiring these data. Transportation is restricted to the transportation of physical goods (as opposed to information, services, electricity, or the like). This is accomplished through one or more modes of transportation (air, rail, pipeline, water, or road). Hence, the data required for these different transportation modes include, but is not limited to: capacity, location, and freight forwarding capabilities. Further, much of the data required to model the transportation of goods is owned by private companies who are generally unwilling to share such information. As a result, acquiring the necessary datasets or resources can be time-consuming and introduce many uncertainties. To account for this, no proprietary data is represented in the following discussion of the different data types.

Table 1. Data Requirements for Transportation Sector (modified from Long et al., 2013)

Category	**Data Description**	**Data Type**	**Ownership**	**Data Challenges**
Freight Data				
Commodity Freight	Food, Paper, Wine, etc.	Tons	Public	Static data; Generalized data; Proprietary data
Manufactured Goods	Electronics, Machinery, Textiles, etc.	Tons	Private/ Public	
Raw Materials	Coal, Iron Ore, Bauxite, etc.	Tons	Private/ Public	
Freight Flow Data				
Road Transport	Goods transported over roads	Tons	Private/ Public	Inconsistency Estimation required Public/Private ownership
Rail Transport	Goods transported on rail	Tons	Private	
Air Transport	Goods transported by air	Tons	Private	
Water Transport	Goods transported by water	Tons	Private/ Public	
Pipeline Transport	Goods transported through pipeline	Tons	Private/ Public	
Infrastructure Capacity Data				
Road-Hub	Bulk, General Cargo, Containers, etc.	Tons	Private	Varied amount of data needed Different capabilities of hubs Interdependency of data
Rail-Hub	Bulk, Intermodal, Shunting, etc.	Tons	Private	
Water-Hub	Rail Car Storage, Dry Storage, Liquid Storage	Tons/ Bushels	Private	
Infrastructure Location Data (Geospatial Data)				

Table 1. Data Requirements for Transportation Sector (modified from Long et al., 2013) (cont.)

Hub Location	Number of hubs in the area	Number	Private	Ever changing data; Use of Software; Static data
Utility Location	Location of all utilities that aid freight flow	Number	Private/ Public	
Road/ Bridge Location	Location of all roads and bridges	Number	Public	
Airport Location	Location of air infrastructure	Number	Private	
Pipeline Location	Location of pipelines and pumping stations	Number	Private	
River Location	Location of docks and storage areas	Number	Private	
Rail Location	Location of all rail infrastructure	Number	Private	
Restoration Data				
Number of People	Number of people need and available	Number	Private/ Public	Different temporal factors; Vast amount of data; Scalability; Ownership of data
Travel Time	Time required for teams to arrive in area	Hours/ Days	Private/ Public	
Skill Set	Skills necessary for each repair job	Qualitative	Private/ Public	
Mode Substitution	Mode substitutions facilitating freight flow	Mode	Private/ Public	
Task Management	Assignment and management of repair tasks	Qualitative	Private/ Public	
Equipment Necessary	Materials require for restoration	Tons/ Pieces	Private/ Public	
Hazard Risk and Vulnerability Data				

Table 1. Data Requirements for Transportation Sector (modified from Long et al., 2013) (cont.)

Historic Data	Previous hazards that have caused damage	Text	Private/ Public	Inconsistency Generalized data Proprietary data
Fragility Data	Vulnerability of element to hazard	Percentage	Public	
Damage Estimation	Severity and extent of damage from simulation	Percentage	Public	

a. Fright/Freight Flow Data

Freight data include information about commodities shipped, their weight, manufactured goods versus raw materials, and the value of materials that are transported. In addition, the mode of transportation (rail, road, air, water, or pipeline) used to ship the goods and the holding capacities of each mode for a given area are included in this data. Freight flow data are typically measured in tons of goods transported and recorded as tons/commodity/mode by the National Transportation Atlas (NTAD, 2010). The primary source for freight data is the Commodity Flow Survey (CFS) of 2013 (U.S. Census Bureau, 2013). It is a public database that contains information on domestic interstate freight. Data are fed into this database through a variety of sources, but the primary problem with these data is their resolution and completeness (LeBeau, 2006). Data gaps can, in part, be removed by estimating values for a commodity using a gravity model of spatial interactions, which can be used as a method for determining facility locations (Holguin-Veras and Jaller, 2011; Nan Liu and Vilain, 2004). Origins, destinations, and modes also require estimation due to the gaps in freight data. In general, these data provide enough information to form estimates for missing data (Transportation Research Board, 2003). More accurate data likely exists, but it is proprietary in nature. Since most freight transportation companies are privately owned; the modes used, commodities shipped, routing (including transshipment facilities), and tonnage are either under-reported or the data is not available to the public. In these cases it is necessary to estimate the missing data based on the publically available data. The data regarding commodities passing through a state are generally available, and from this information the flow of commodities through a particular area can be estimated. The tonnage transported can be a major factor in assigning priorities within restoration models (e.g. the greater the tonnage transported, the higher the priority that mode of transport has during the restoration process).

b. Transportation Infrastructure Capacity Data

Infrastructure capacity data incorporates holding capacities of infrastructure facilities that aid freight flow such as cargo hubs. When considering hubs that store goods and commodities, the multimodal nature of modern cargo transportation systems is important. Goods may arrive by river or sea, be stored in a water-hub, be picked up by a truck and subsequently stored in a road-hub. There are four main types of hubs considered here: Water-hubs, Rail-hubs, Road-hubs, and Air-hubs. 1) *Water-Hubs* form the largest and most diversified hubs in the transportation system. They facilitate transportation services for many types of products via barge or ship. They are also multimodal hubs that act as transfer points for many types of products from water modes to other modes such as rail, pipeline, air, or truck. An inherent problem with the data associated with water-hubs is that a variety of information unique to that hub is needed. 2) *Rail-Hubs* are most commonly rail freight yards. These hubs require a great deal of space for multiple tracks and are therefore most likely to be located on greenfield sites within or near major industrial zones. Rail-hubs generally have very large holding capacities and also act a multimodal hub. 3) *Road-Hubs* usually store freight which is very diverse and bulky. They also act as multimodal hubs, shipping and receiving goods from road, rail, air, and water. Road-hubs are generally located just off major interstates to reduce transportation time. *4) Air-Hubs* are typically located at airports connected to major road networks which allows for the rapid flow of people or cargo. These constitute the smallest hub connection due to the relatively high costs involved with air transport.

The data required for these hubs include freight handling data (what equipment is required for loading), information about the facilities required to accommodate ships, trucks and trains (berths, loading bays and freight yards respectively), total capacity data according to type of goods they can store (cold storage, hot storage, hazardous material etc.), and freight flow. Most of the transportation data for road and rail is obtained from National Transportation Atlas Database (NTAD, 2010) or the CFS of the U.S. Census Bureau (2013) which is a public resource.

c. Geospatial Data

Geospatial datasets contains the location information associated with various types of data and as such forms the base into which other data are integrated. The geospatial data include the locations of hubs, warehouses, utilities, infrastructure, and all other objects or materials that could be damaged and in need of repair or replacement from the impact of a large-scale disaster. Most of these data are available or can be derived from geospatial-centered websites like The National Map (TNM) of the U.S. Geological Survey. A shortfall of these data is their static nature. Most geospatial data are updated yearly or over the course of several years, so as new warehouses and hubs are built, the geospatial data will not convey these new sites until the next update cycle. Also, the extraction of these data from such geospatially located sources as orthoimagery can be quite time consuming and require specialized personnel for the process. The advantage of these data is their free availability, large area coverage, and accurate overview of ground features. Figure 2 illustrates some SCICI element examples for the St. Louis, Missouri region.

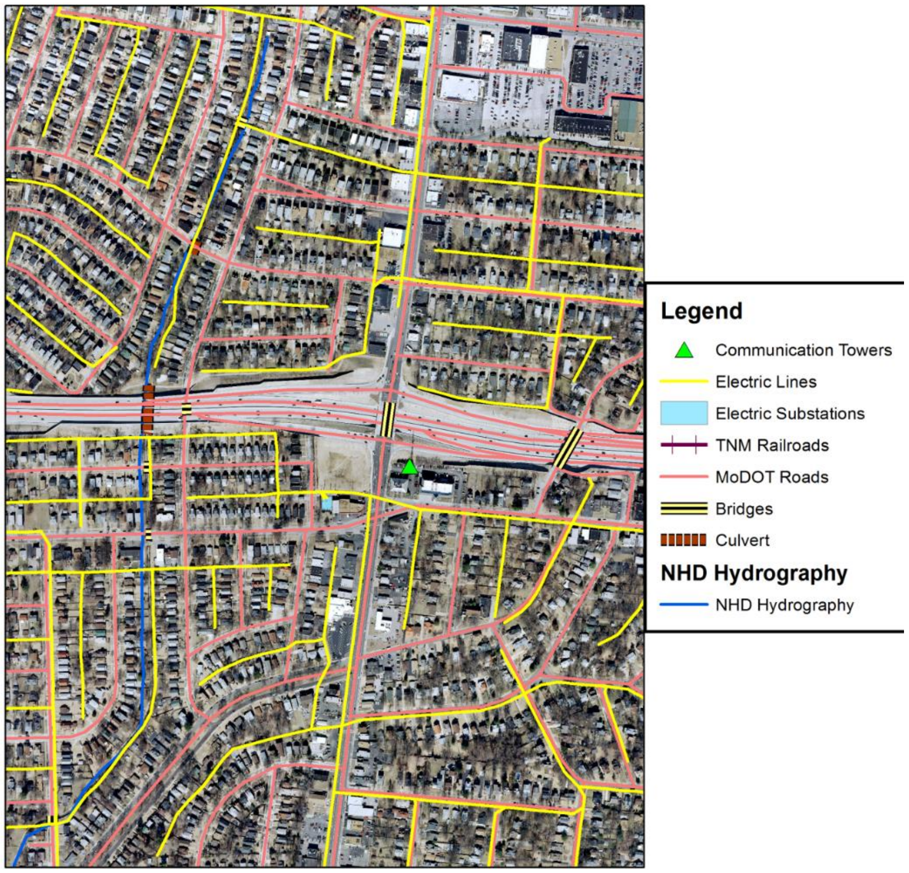


Figure 2 Orthoimagery, hydrography (National Hydrography Dataset, NHD) and rail data for St. Louis, Missouri region from The National Map of the U.S. Geological Survey. Road data is from the Missouri Department of Transportation. Data from the U.S. Geological Survey and the Missouri Department of Transportation are in the public domain and freely available for download. Other elements (communications, electric power) are derived from the public domain orthoimagery.

d. Restoration Data

Restoration data are records containing information on rebuilding or recovery activity rates. These data include the number of skilled workers available for restoration activities, raw material stockpiles, necessary equipment accessibility, the time required for teams to assemble within a given area, and collaborations between invested agencies: federal, state, and local. These data come, in part, from personal interviews with people experienced in disaster reconstruction and from published agency reports on restoration activities. Typically these data are not available in electronic format and, for the most part, integrating these can be time consuming. Much of these data are specific to the type of disaster experienced. Nevertheless, elements are often generalizable and can be used in developing restoration estimates for most damage estimates.

e. Hazard Data

The damage experienced by the transportation sector will, of course, depend on such variables as the type of disaster, its severity, duration, the vulnerability of the infrastructure, and the like. The actual damage experienced must ultimately be input data into any reconstruction optimization model, nevertheless, for the purpose of testing such a model a damage estimate can be simulated. Such a simulation requires hazard risk evaluation data as well as SCICI survivability estimates. Much progress with such simulations has already been made by FEMA (2003) and can be accessed in the HAZUS-MH software which provides simulations of some network vulnerabilities to different hazards

f. Role of GIS in Data Acquisition and Integration

GIS offers tools that make the acquisition and integration of SCICI system data more tractable. Data layers from The National Map of the U.S. Geological Survey include orthoimagery, elevation, hydrography, transportation, place names, and land cover, and can be downloaded directly into a GIS database (Sugarbaker and Carswell, 2011). The orthoimagery serves as an excellent, if rather memory-extensive, base map from which to hang existing data sets and to extract further SCICI data. The orthoimagery projection is used as the default coordinate system into which all other data will be projected. Anything that is visible in the orthoimagery can be extracted by digitization as new SCICI data features (e.g., the locations of culverts, cell towers, electric power lines, bridges, pumping stations, etc.). Further analyses of the orthoimagery also provide the ability to estimate capacity of these infrastructure elements as well (e.g., number of road lanes, number of rail tracks, dock lengths, electric line voltages, etc.). In addition, many local and regional government agencies (state departments of transportation, state departments of commerce, city utility districts, etc.) have data that can be integrated into a GIS database. To create the transportation system network, GIS is used to represent real-world features that are populated by discrete identifiable objects to build network analysis models based on graph theory representing transportation elements as vertices and edges.

g. SCICI Interdependencies

One of the main characteristics of SCICI elements is the multiplicity of interdependencies between them. For example, a water pumping station, in order to function, requires electricity to run the pumps, communication to control how much water needs moved, water lines through which the water will move, and roads to access the station. In any attempt to return functionality to a pumping station after a large-scale disaster, it is necessary to know the local interdependencies such as which electrical lines ran into the power station, what roads access it, what cell tower communicates with it, and through which lines water moves into and out of the station. Less obvious, but equally important, it is necessary to understand that these connected elements are interdependent on far field elements such as which power station feeds electricity to the sector of the pumping station, which substations transform the power into usable voltages, what communication path moves from the controller to the pumping station, what bridges are available to move material and manpower to the pumping station for repair, where are there any damaged water lines between this pumping station and those before and after it. The main contribution in the acquisition of all these data and their integration into a GIS is the ultimate ability to map out these interdependencies through the SCICI model.

4. Results

The modeling techniques presented here make use of the high resolution imagery provided by The National Map to identify both the location of system elements and their proximity to one another. This spatial information identifies the interfaces between the systems and captures the interrelationships that give rise to complex responses. The interrelationships are driven by the system specific information, in the case of the transportation infrastructure system this is the freight and infrastructure data.

In order to test the efficacy of the integration of these data into the proposed modeling techniques, the St. Louis, Missouri metropolitan region was chosen as a test area. This area is covered by 2268 orthoimagery tiles from The National Map with cell-lengths ranging from 0.15 m to 0.6 m. These tiles constitute the base map onto which other data layers are projected.

Considerable transportation data (particularly roads and rail lines) are available from state (in this case, Missouri and Illinois) departments of transportation. Much of the rest of the data is extracted from orthoimagery by heads-up digitization or other sources as shown in Table 2.

Table 2. Data acquired and integrated for SCICI modeling for St. Louis metro area.

Title	**Description**	**Source**	**Restrictions**	**Data Processing**
Geospatial Data	*Several layers of TNM data serve as the base to all data integration processes: orthoimagery, elevation, hydrograph, place-names and land use*	The National Map of the U.S. Geological Survey	Open access/public data	None
Infrastructure Data	*Infrastructure data such as airports, electric grid, bridges, overpasses, tunnels, culverts, dams, docks, pumping stations…*	Extrapolate from The National Map	Open access/public data	Digitization
Transportation Data	*Road and rail lines*	State departments of transportation	Varies from state to state	Re-projection to desired coordinates
Communication Data	*Cell Towers*	Federal Communications Commission	Open access/public data	Re-projection to desired coordinates

Table 2. Data acquired and integrated for SCICI modeling for St. Louis metro area (cont.)

Supply Chain Data	*Rates of flow of commodities*	U.S. Department of Commerce and Private Industry sources	Public/Private	Integration with geospatial data
Restoration Data	*Rate and manner in which supply chain elements are repaired after a large-scale disaster*	Federal, State and Local Governments	Open access/public data	Integration with geospatial data
Hazard Data	*The nature of destruction of specific supply chain elements by any large-scale disaster*	Federal, State and Local Governments	Open access/public data	Integration with geospatial data

Many of the features that need to be digitized have a three-dimensional structure (e.g. cell tower, electric poles, etc.). To reduce the effects of parallax, features extracted from the orthoimagery are preferentially digitized at their base (for example, where a pole and its shadow meet). It should be noted, however, that since these data are extracted from the orthoimagery, only elements that are visible from the air can be digitized. Some elements (such as sewer lines and water mains) can be interpolated based on surface features in high-resolution orthoimagery (man-hole covers or fire hydrants), whereas others (buried telephone lines, electric lines and fiber optic cable, and gas mains) are best obtained from other sources which are often more difficult to obtain. In addition, where high resolution imagery is not available (typically outside of urban settings) the level of detail would correspondingly decrease.

In spite of the being relatively few SCICI databases available to the general public that can be used for realistic models of disaster restoration, a considerable amount of infrastructure data can be gleaned from public sources, as shown in Figure 3 for South St. Louis, Missouri. This indicates that a large amount of data applicable to SCICI systems is available from public datasets alone. To date, 640 Gigabytes of data have been acquired for review. While this is rather large for real-time processing and model manipulation, the size of the data needed to describe actual infrastructure elements such as bridges, culverts, road networks, electric grid, communication networks, dams, locks, rail networks, water facilities and docks for the St. Louis metroplex is less than 100 Megabytes. This presents a complicated tangle of infrastructure elements, but with preprocessing it can now be fit into a model that will begin to piece together the interdependencies of the SCICI which is crucial to their restoration in the wake of a large scale disaster.

However, even with this rich source of SCICI data, severe limitations still remain. One of these is that orthoimagery data must, by its very nature, be considered a static data source. It is a picture of the SCICI environment at the time of the flyovers, and these are not updated until the next flight cycle occurs which is generally between 3 to 5 years. Changes made to SCICI between data cycles cannot be incorporated into the model by this method.

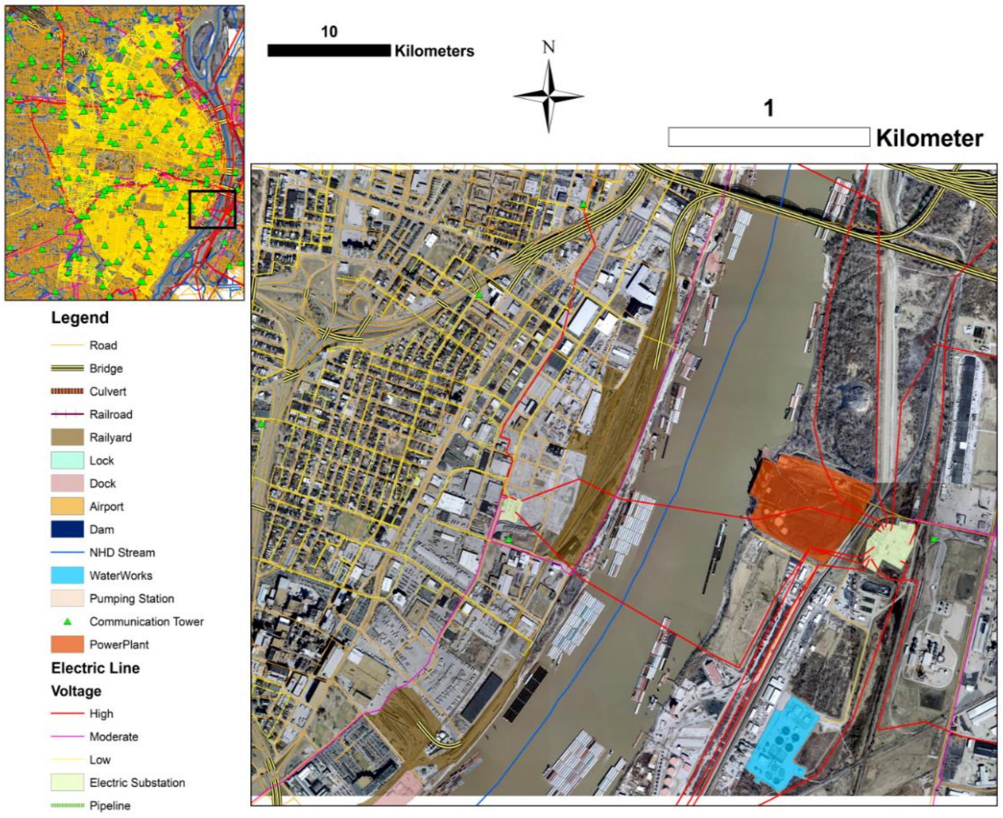


Figure 3 SCICI elements for a section of southern St. Louis, Missouri. Upper left map shows infrastructure in the St. Louis metropolitan area, inset black box shows the expanded area in the larger low right box.

Also, at least in regards to the SCICI data derived from orthoimagery, only what can be seen from the air can be acquired. As discussed earlier very little information is available from orthoimagery on pipelines, buried electrical wiring, water mains (although these can be tracked by fire hydrants), fiber optic cable or gas lines. These must be acquired from other sources not all of which lies in the public domain. Further, the labor intensive digitization on such a massive scale introduces many human errors into the data including features that are missed, erroneously added, misinterpreted, or digitized inaccurately. While this is potentially serious in individual cases, the sheer quantity of the data should permit the proper interdependencies to emerge; which is the ultimate objective. Again like the damage assessment simulation, the input of the infrastructure, once the techniques for mapping of the interdependencies is complete, will be input by individual communities. As electrical grids, water distribution systems, gas lines, etc., become more ‘smart’, (Amin and Wollenberg, 2005; Gao et al., 2012; Gungor et al., 2011) data can be fed directly into the model from sensors, giving a dynamic, real-time dimension to the analysis.

5. Conclusion and Future Work

Integrating hazard, human intervention, restoration, geospatial, freight flow, and infrastructure data for each SCICI element helps create a complex model of SCICI. This complexity arises not from the data itself, but in the interaction of SCICI processes which these data map (for example, an electric pole is not complex, but what happens to a water pumping station, a warehouse refrigeration unit, and several traffic lights if that pole were to be destroyed can lead to complexity). While separately these processes are complicated, in essence it is their interaction and interdependence that generates nonlinear behavior (complexity). However, all SCICI elements have a common property: they all have complex components which interact with each other. The larger the scale of the SCICI, the more complex its systems, and the more it starts to display unexpected and nonlinear behavior. It is this behavior that can lead to cascading failures throughout many of the SCICI elements when a single unit fails. A major goal for modeling and optimization techniques is to see such failures in the system and rapidly repair and even improve complex infrastructure.

This research addresses a gap that exists in literature associated with the acquisition and integration of the different types of data which must be brought together in order to build complex and robust models of supply chain systems. Geospatial infrastructure data is the most abundant of these data, and while much of it is acquirable through public sources, a serious effort is required to gather, develop and integrate these data. Continued availability of public geospatial data is of paramount importance because no single utility or private firm has access to the various sources of data necessary to model supply chains that feed their own function. Further, much of the modeling is done in academic communities outside government circles which preclude access to restricted or classified data.

The bulk of the freight flow transportation data are proprietary, this requires that reasonable assumptions be made regarding data that are not accessible. Nevertheless, this research suggests that there is sufficient data available in public domain to create a realistic model of the transportation system, and that this model is scalable to the other elements of SCICI.

Future work will increase the quantity and diversity of real-world data to expand through the other SCICI elements. Mapping the interdependency between SCICI elements is essential to the construction of Supply Chain modelling. These interdependencies are important due to the complexity of the systems. Further, sophisticated modelling and optimization techniques need to be created to explore the efficiency of restoration schema.

The next step in this research is to increase the level of detail and robustness of the data used in this research so far. LiDAR, and elevation data can be used to get further close to real-world. Only using GPS data can create problems when looking at something which is not on the same height. Or, the extrapolating and identifying underground pipes is extremely difficult with the data that is available right now. Making use of semantics and ontology is another step forward as this will help with the integration of data. Since, the data right now comes from different sources and is not available in the same format semantics can play a huge role in understanding the data and finding trends within the data, which will be beneficial when looking at mapping interdependencies. Since, this research involves working with large, and varied amount of data, there is also a need to look at reducing the computing time. The 29 size block area that was considered in this research needed more than 7 Tb of data, and this will only increase as the research becomes more robust, hence there is a need to look at big-data analytics and parallel processing techniques.

6. References

Adams, T. M., and L. D. Stewart, 2014. “Chaos theory and organizational crisis: A theoretical analysis of the challenges faced by the New Orleans Police Department during Hurricane Katrina”. *Public Organization Review,* 14(4): DOI 10.1007/s11115-014-0284-9.

Altay, N., and W. G. Green, 2006. “OR/MS research in disaster operations management”. *European Journal of Operational Research,* 175(1):475–93: DOI 10.1016/j.ejor.2005.05.016

Amin, S. M., and B. F. Wollenberg. 2005. “Toward a smart grid: power delivery for the 21st century”. *Power and Energy Magazine, IEEE, 3*(5), 34-41. DOI 10.1109/MPAE.2005.1507024

Bruneau, M., S. E. Chang, R. T. Eguchi, G. C. Lee, T. D. O’Rourke, A. M. Reinhorn, M. Shinozuka, K. Tierney, W. A. Wallace and D. von Winterfeldt. 2003. “A framework to quantitatively assess and enhance the seismic resilience of communities”. *Earthquake Spectra*, 19 (4):737–8. DOI http://dx.doi.org/10.1193/1.1623497

Burrough, P. A. 1990. “Methods of spatial analysis in GIS”. *International Journal of Geographical Information Systems*, *4*(3), 221-223.

Carlson, J. M., and J. Doyle. 1999. “Highly optimized tolerance: A mechanism for power laws in designed systems”. *Physical Review E*, *60*(2), 1412. DOI http://dx.doi.org/10.1103/PhysRevE.60.1412

Department of Homeland Security (DHS). 2014 *The White House, United States Government. 2014. Presidential Proclamation - Critical Infrastructure Security and Resilience Month*. Retrieved from: http://www.whitehouse.gov/the-press-office/2014/10/31/presidential-proclamation-critical-infrastructure-security-and-resilienc. Last accessed on 6/19/2014.

Federal Emergency Management Agency (FEMA). 2003. *HAZUS-MH MR4 technical manual*. Washington, DC.

Feng, C., and C. Weng. 2005. “A bi-level programming model for allocating private and emergency vehicle flows in seismic disaster areas”. *Proceedings of the Eastern Asia Society for Transportation Studies* 2005:1408–23.

Fletcher, D. R. 2002. “The Role of Geospatial Technology in Critical Transportation Infrastructure Protection: A Research Agenda”. *National Consortium on Remote Sensing in Transportation-Infrastructure Management*.

Gao, J., Y. Xiao, J. Liu, W. Liang, and C. L. Chen. 2012. “A survey of communication/networking in Smart Grids”. *Future Generation Computer Systems*, *28*(2), 391-404. DOI 10.1016/j.future.2011.04.014

Godschalk, D.R., 1991. “Disaster mitigation and hazard management E.D. Thomas, G.J. Hoetmer (Eds.), Emergency Management: Principles and Practice for Local Government”. *International City Management Association, Washington, DC (1991)*, pp. 131–160

Goodchild, M. F., and R. P. Haining. 2004. “GIS and spatial data analysis: Converging perspectives”. *Papers in Regional Science, 83*(1), 363-385. DOI 10.1007/s10110-003-0190-y

Greene, R. W. 2002. “Confronting catastrophe: A GIS handbook”. Redlands: page 140. ESRI press.

Gungor, V. C., D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke. 2011. “Smart grid technologies: communication technologies and standards”. *Industrial Informatics, IEEE Transactions on, 7*(4), 529-539. DOI 10.1109/TII.2011.2166794

Hale, T., and C. R. Moberg. 2005. “Improving supply chain disaster preparedness: a decision process for secure site location”. *International Journal of Physical Distribution & Logistics Management*, *35*(3), 195-207. DOI http://dx.doi.org/10.1108/09600030510594576

Haimes, Y. Y. 2005. “Infrastructure interdependencies and homeland security”. *Journal of Infrastructure Systems, 11*(2), 65-66. DOI 10.1061/(ASCE)1076-0342(2005)11:2(65)

Holguín-Veras, J., and M. Jaller. 2011. “Immediate resource requirements after hurricane Katrina”. *Natural Hazards Review*, *13*(2), 117-131. DOI 10.1061/(ASCE)NH.1527-6996.0000068

Kondaveti, R, and R. Ganz. 2009. “Decision support system for resource allocation in disaster management”. In: *Proceedings of the 31st annual international conference of the IEEE EMBS*, Minneapolis, Minnesota; pp. 3425–8. DOI 10.1109/IEMBS.2009.5332498

LeBeau, J. 2006. “A Critical Review and Integration of GIS-Based Spatial Databases for Multi Commodity and Multi-Mode Freight Movement Modeling and Security Analysis in USA”, Paper presented at GIS for Transportation Symposium, Des Moines, Iowa. April 19-22, 2007.

Long, S., T. Shoberg, V. Ramachandran, S.M. Corns, and H.J. Carlo. 2013. “Integrating complexity into data-driven multi-hazard supply chain network strategies”. *Proceedings of the ASPRS\CaGIS 2013 Specialty Conference*, San Antonio,TX, URL: http://info.asprs.org/publications/proceedings/SanAntonio2013/index.htm,

MacKenzie, C.A., K. Barker, and J. R. Santos. 2014. “Modeling a severe supply chain disruption and post-disaster decision making with application the the Japanese earthquake and tsunami”. *IIE Transactions*, 46(12): 1243-1260. DOI 10.1080/0740817X.2013.876241

Miller-Hooks, E., X. Zhang, and R. Faturechi. 2012. "Measuring and maximizing resilience of freight transportation networks". *Computers & Operations Research,* 39(7): 1633-1643. DOI 10.1016/j.cor.2011.09.017

Mitchell, A. 2005. "The ESRI Guide to GIS Analysis: Spatial Measurements and Statistics. Vol 2". Redlands. ESRI Guide to GIS analysis.

Nan Liu. L., and P. Vilain. 2004. "Estimating Commodity Inflows to a Substate Region Using Input-Output Data: Commodity Flow Survey Accuracy Tests". *Journal of Transportation and Statistics*. 7: 1-8.

Openshaw, S., 1994. "Two exploratory space-time-attribute pattern analysers relevant to GIS". *Spatial Analysis and GIS,* 83-104.

Ouyang, M., and L. Dueñas-Osorio. 2011. "Efficient Approach to Compute Generalized Interdependent Effects between Infrastructure Systems". *Journal of Computing in Civil Engineering, 25(5), 394-406*. DOI 10.1061/(ASCE)CP.1943-5487.0000103

Reed D. A., K. C. Kapur, and R. D. Christie. 2007. "Methodology for assessing the resilience of networked infrastructure". *IEEE Systems Journal,* 3(2):174–80. DOI 10.1109/JSYST.2009.2017396

Rosato V., L. Issacharoff, F. Tiriticco, and S. Meloni. 2008. "Modeling interdependent infrastructures using interacting dynamical models". *International Journal of Critical Infrastructure,* 4(1–2):63–79. DOI 10.1504/IJCIS.2008.016092

Sinton, D. F. (1992). "Reflections on 25 years of GIS". *GIS World, 5*(2), 1-8.

Shinozuka, M., X. Dong, T. C. Chen, and X. Jin. 2007. "Seismic performance of electric transmission network under components failures". *Earthquake Engineering Structural Dynamics.,* 36:224–7. DOI: 10.1002/eqe.627

Sugarbaker, L., and W. J. Carswell. 2011. "The National Map$^{®}$". US Department of the Interior, US Geological Survey.

Transportation Research Board (TRB). 2003. A Concept for a National Freight Data Program. *Special Report 276. Transportation Research Board*: Washington, D.C.

U.S. Bureau of Transportation Statistics, 2010. National Transportation Atlas Database (NTAD). *Research and Innovative Technology Administration., U.S. Bureau. of Transportation Statistics, U.S. Department of Transportation., Washington, D.C.* Available at http://www.bts.gov/publications/national_transportation_atlas_database/2010. Last accessed on 8/23/2014.

U.S. Census Bureau & Bureau of Transportation Statistics. 2013. Commodity Freight Survey (CFS). Available at http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/publications/commodity_flow_survey/index.html. Last accessed on 6/19/2014.

Vugrin, E. D., D. E. Warren, M. A. Ehlen, and R. C. Camphouse. (2010). "A framework for assessing the resilience of infrastructure and economic systems". In *Sustainable and Resilient Critical Infrastructure Systems* (pp. 77-116). Springer Berlin Heidelberg. DOI 10.1007/978-3-642-11405-2_3

Widener, M. J., and M. W. Horner. 2011. "A hierarchical approach to modeling hurricane disaster relief goods distribution". *Journal of Transport Geography*, *19*(4), 821-828. DOI 10.1016/j.jtrangeo.2010.10.006

Zeiler, M. 2010. "Modeling Our World: The ESRI Guide to Geodatabase Concepts". ESRI press. ISBN: 9781589482784

SECTION

2. CONCLUSION AND FUTURE WORKS

This research discussed a systems based methodology for modeling supply chain interdependent critical infrastructure systems and mapping the interdependencies that exists among the infrastructure. The process considers the different types of data required to model the SCICI elements (transportation, water, power, communication) and uses publically available data to create the model as close to real-time as possible. The research identifies problems with the data, but proves that with the data that is available publically it is possible to model the infrastructure. To map the interdependencies between infrastructures an agent based model is used where agents interact with each other first haphazardly and from this emergent behaviors emerge that help in mapping interdependency. A test case is developed for the St. Louis metro region, where data is obtained, and analyzed to model the infrastructure of that area. Then, all of this data integrated and geographic and physical interdependencies is mapped.

The advantage of this methodology is that it is scalable, and flexible i.e. the same model can be used for different regions and different infrastructure elements if the data is available. It can work as a plug-and-play model where the end-user plugs in data and the interdependencies are mapped, and this can be used when dealing with restoration after damage to infrastructure due to natural or man-made events. The methodology proposed in this work contributes to the literature by its explicit combination of modeling infrastructure elements using real data and mapping interdependencies between them.

The disruption of SCICI on a metropolitan scale drastically impacts economic stability on a regional, national, and international scale. Without a comprehensive, data-driven model of SCICI, communities cannot re-establish the level of vibrant connectivity required for a timely restoration of goods and services. Results from this work will establish a decision framework capable of optimizing disaster restoration. The research presented here evaluates several key components in the complex and complicated problem of supply chain network restoration in the aftermath of large-scale disasters. There were three main areas of research when looking to develop a methodology that can be utilized to look at restoration after an extreme event. The first is to develop a framework that can be used to model critical infrastructure. This was achieved using an interdisciplinary approach to develop a comprehensive framework for resiliency modeling. The approach includes: graph theory, geospatial data, supply chain assessment, and hazards risk analysis, as well as the integration of these diverse data sets. The second area of focus was to understand and acquire the data required for this research. A review of publically available data revealed a paramount need for real-time data and information to assist urban planners with recovery following an extreme event. The third area of focus was the integration and system interdependency mapping of dozens of diverse, complicated and heterogeneous data sets. The simulation-optimization models will be of a general-purpose nature and the main paradigms that will result from this research will be useful for modeling any post-disaster environment. These models have key broader impacts in terms of the influence on society and public policy.

The other important contribution of this research is the interdependency mapping. This is extremely essential when looking at cascading failure or damage, and also when looking at restoration after damage. Modeling this can be done at different levels of abstraction from high level to detailed level as well as on a physical, logical, or cyber level. Depending on the type of results this will vary. Most of the research currently available does not consider multiple infrastructures, and this is an extremely important aspect when looking at restoring an urban environment.

This research is another step towards looking at restoration of critical infrastructure after damage due to natural or man-made disaster. Protecting critical infrastructures remains a difficult and an open problem, especially due to there not being a clear understanding of the complex interdependencies that exists among the infrastructures. Once, that has been studied a heuristic can be developed to look at the restoration process, and putting the urban area back on the grid or getting the economy back on track. Most of the methods that are present do not address the problem with a comprehensive method but look at only parts of the problem, but this research can act as basis of making a data-driven model that evaluates methods and priorities for doing the restoration process. The broader impacts and how this research can be taken forward is given below:

- A computer-based model is created to simulate and analyze the data which uses ABMS for mapping the interdependencies. Since the number of natural hazards and calamities is increasing, there is a need to look at the economic aspect of restoration. This will include cost of restoration along with the cost of rebuilding. This research looks to implement the Leontief input/output model to calculate the damage and also considers interdependency between infrastructures. Going forward, this can be expanded to look at feasibility of changing the infrastructure, and how cost-effective that would be. Either, building sustainably or building it better rather than just rebuilding.

- This research also considers efforts to reduce public risk after an extreme event. This is done by removing the uncertainties as to how the restoration is to be done, or by looking at limiting the damage by pre-planning. Though modeling human behavior is a complex task, optimizing the restoration means that they whole process should run seamlessly with all the components working in tandem with each other. This includes public-private handoffs. This work provides a platform for future research to be done in this area. Improvements need to be made to look at the data sharing capabilities, responsibilities, authorization etc.
- One of the main areas of concern is the dearth of publically available data and the problems associated with this data. There needs to be an improvement in this area, in regards to what data is available, what data is necessary for such a research, and the inherent problems (static, private, inconsistent, proprietary data etc.). There is not a lot of literature available which deals with only the data aspect, as they generally 'make-up' this data. This research builds a database that can be built upon, to include more data which is required for this research.
- Coupling GIS and modeling system has not been very beneficial in the past amd the best way forward is to integrate the required functionality of either the GIS or simulation into the other. In this research GIS software works as a part of the modeling software and is integrated using a middleware. Since the tool is open-source, additions to this can be made easily. One of the aims of this research is to give this framework to decision-makers who can then make decisions regarding restoration based on real-world like scenario. But, this tool can also be used for pre-event preparedness, so that the city planners and the city itself can be ready.

Since, all the infrastructure of the city is mapped in this tool, the vulnerabilities of the city can be found out and it also allows planners and private companies to preview how their buildings would react to an extreme event. This research can further be extended to include more facilities and become more detailed so as to improve the efficiency of the results. Using ABMS provides advantages like reduced computational times and increase in efficiency of computation due to the fact the system can be divided be divided into sub-systems, and also it helps when the size of the problem is large. Integrating GIS data for ABM is a difficult process and many considerations are needed such as what data to use, how the data has to manipulated, or how the agents should react. But, this does lead to increased research in this area, like this work.

APPENDIX
AGENT BASED MODELING CODE

```
/*©vrnq5

 *

 *

 *

 *

 * */


package scsi.agent;


import java.util.Iterator;

import java.util.logging.Logger;


import com.vividsolutions.jts.geom.Geometry;


import repast.simphony.context.Context;

import scsi.environment.Airport;

import scsi.environment.GISFunctions;

import scsi.environment.SpatialIndexManager;

import scsi.exceptions.AgentCreationException;

import scsi.main.ContextManager;

import scsi.main.GlobalVars;


/**

 * Create agents. There are three methods that can be used to create agents: randomly
create a number of agents, create
```

```
 * agents from a point shapefile or create a certain number of agents per neighborhood
specified in an area shapefile.
 *
 * <P>
 * The method to use is specified by the 'agent_definition' parameter in
<code>parameters.xml</code>. The parameter
 * takes the following form:
 * </P>
 *
 * <pre>
 * {@code
 * <method>:<definition>
 * }
 * </pre>
 *
 * <P>
 * where method and can be one of the following:
 * </P>
 *
 * <ul>
 * <li>
 *
 * <pre>
 * {@code random:<num_agents>}
 * </pre>
 *
 * Create 'num_agents' agents in randomly chosen houses. The agents are of type
<code>DefaultAgent</code>. For example,

 * this will create 10 agents in randomly chosen houses: '<code>random:1</code>'. See
the

 * <code>createRandomAgents</code> function for implementation details.</li>

 *

 * <li>

 *

 * <pre>

 * {@code point:<filename>%<agent_class>}

 * </pre>

 *

 * Create agents from the given point shapefile (one agent per point). If a point in the
agent shapefile is within a

 * building object then the agent's home will be set to that building. The type of the
agent can be given in two ways:

 * <ol>

 * <li>The 'agent_class' parameter can be used - this is the fully qualified (e.g. including
package) name of a class

 * that will be used to create all the agents. For example the following will create
instances of <code>MyAgent</code>

 * at each point in the shapefile
'<code>point:data/my_shapefile.shp$my_package.agents.MyAgent</code>'.</li>

 * <li>A String column in the input shapefile called 'agent_type' provides the class of the
agents. IIn this manner

 * agents of different types can be created from the same input. For example, the
following will read the shapefile and

 * look at the values in the 'agent_type' column to create agents:
'<code>point:data/my_shapefile.shp</code>' (note that
 * unlike the previous method there is no '$').</li>

 * </ol>

 *

 * See the <code>createPointAgents</code> function for implementation details.

 *

 * <li>

 *

 * <pre>

 * {@code area:<filename>$BglrC1%<agent_class1>$ .. $BglrC5%<agent_class5>}

 * </pre>

 *

 * Create agents from the given areas shapefile. Up to five different types of agents can
be created. Columns in the

 * shapefile specify how many agents of each type to create per area and the agents
created are randomly assigned to

 * houses withing their area. The columns names must follow the format 'BglrCX' where
1 <= X <= 5. For example the

 * following string:<br>

 *

 * <pre>

 * {@code area:area.shp$BglrC1%BurglarAgent$BglrC2%EmployedAgent}

 * </pre>

 *

 * will read the <code>area.shp</code> and, for each area, create a number of
<code>BurglarAgent</code> and

 * <code>EmployedAgent</code> agents in each area, the number being specied by
columns called <code>BglrC1</code> and
```

```
 * <code>BglrC2</code> respectively. See the <code>createAreaAgents</code> function
for implementation details.</li>
 * </ul>
 *
 * @author vrnq5
 * @see DefaultAgent
 */
public class AgentFactory {

	private static Logger LOGGER = Logger.getLogger(AgentFactory.class.getName());

	/** The method to use when creating agents (determined in constructor). */
	private AGENT_FACTORY_METHODS methodToUse;

	/** The definition of the agents - specific to the method being used */
	private String definition;

	/**
	 * Create a new agent factory from the given definition.
	 *
	 * @param agentDefinition
	 */
	public AgentFactory(String agentDefinition) throws AgentCreationException {

		// First try to parse the definition
		String[] split = agentDefinition.split(":");
```

```
			if (split.length != 2) {
				throw new AgentCreationException("Problem parsin the
definition string '" + agentDefinition
						+ "': it split into " + split.length + " parts but should
split into 2.");
			}
			String method = split[0]; // The method to create agents
			String defn = split[1]; // Information about the agents themselves

			if (method.equals(AGENT_FACTORY_METHODS.RANDOM.toString())) {
				this.methodToUse = AGENT_FACTORY_METHODS.RANDOM;

			} else if
(method.equals(AGENT_FACTORY_METHODS.POINT_FILE.toString())) {
				this.methodToUse = AGENT_FACTORY_METHODS.POINT_FILE;
			}

			else if
(method.equals(AGENT_FACTORY_METHODS.AREA_FILE.toString())) {
				this.methodToUse = AGENT_FACTORY_METHODS.AREA_FILE;
			}

			else {
				throw new AgentCreationException("Unrecognised method of
creating agents: '" + method
						+ "'. Method must be one of " +
AGENT_FACTORY_METHODS.RANDOM.toString() + ", "
```

```
                                                +
AGENT_FACTORY_METHODS.POINT_FILE.toString() + " or " +
AGENT_FACTORY_METHODS.AREA_FILE.toString());
                    }


                    this.definition = defn; // Method is OK, save the definition for creating
agents later.


                    // Check the rest of the definition is also correct (passing false means
don't create agents)
                    // An exception will be thrown if it doesn't work.
                    this.methodToUse.createAgMeth().createagents(false, this);
          }


          public void createAgents(Context<? extends IAgent> context) throws
AgentCreationException {
                    this.methodToUse.createAgMeth().createagents(true, this);
          }


          /**
           * Create a number of in randomly chosen houses. If there are more agents than
houses then some houses will have
           * more than one agent in them.
           *
           * @param dummy
           *          Whether or not to actually create agents. If this is false then just check
that the definition can be
           *          parsed.
```

```
	 * @throws AgentCreationException
	 */
	private void createRandomAgents(boolean dummy) throws 
AgentCreationException {
		// Check the definition is as expected, in this case it should be a number
		int numAgents = -1;
		try {
			numAgents = Integer.parseInt(this.definition);
		} catch (NumberFormatException ex) {
			throw new AgentCreationException("Using " + this.methodToUse 
+ " method to create "
					+ "agents but cannot convert " + this.definition + " 
into an integer.");
		}
		// The definition has been parsed OK, no can either stop or create the 
agents
		if (dummy) {
			return;
		}

		// Create agents in randomly chosen houses. Use two while loops in case 
there are more agents
		// than houses, so that houses have to be looped over twice.
		LOGGER.info("Creating " + numAgents + " agents using " + 
this.methodToUse + " method.");
		int agentsCreated = 0;
		while (agentsCreated < numAgents) {
				Iterator<Airport> i =
ContextManager.AirportContext.getRandomObjects(Airport.class, numAgents)
						.iterator();
				while (i.hasNext() && agentsCreated < numAgents) {
					Airport b = i.next(); // Find a building
					IAgent a = new DefaultAgent(); // Create a new agent
					a.setHome(b); // Tell the agent where it lives
					b.addAgent(a); // Tell the building that the agent lives
there
					ContextManager.addAgentToContext(a); // Add the agent
to the context
					// Finally move the agent to the place where it lives.
					ContextManager.moveAgent(a,
ContextManager.AirportProjection.getGeometry(b).getCentroid());
					agentsCreated++;
				}
			}
		}

		/**
		 * Read a shapefile and create an agent at each location. If there is a column
called
		 *
		 * @param dummy
		 *         Whether or not to actually create agents. If this is false then just check
that the definition can be
		 *         parsed.
		 * @throws AgentCreationException
```

```
         */
        @SuppressWarnings("unchecked")
        private void createPointAgents(boolean dummy) throws AgentCreationException
{

                // See if there is a single type of agent to create or should read a colum in
shapefile
                boolean singleType = this.definition.contains("$");

                String fileName;
                String className;
                Class<IAgent> clazz;
                if (singleType) {
                        // Agent class provided, can use the Simphony Shapefile loader to
load agents of the given class

                        // Work out the file and class names from the agent definition
                        String[] split = this.definition.split("\\$");
                        if (split.length != 2) {
                                throw new AgentCreationException("There is a problem
with the agent definition, I should be "
                                                + "able to split the definition into two parts
on '$', but only split it into " + split.length
                                                + ". The definition is: '" + this.definition +
"'");
                        }
                         // (Need to append root data directory to the filename).
                        fileName =
ContextManager.getProperty(GlobalVars.GISDataDirectory)+split[0];
                        className = split[1];
                        // Try to create a class from the given name.
                        try {
                                clazz = (Class<IAgent>) Class.forName(className);
                                GISFunctions.readAgentShapefile(clazz, fileName,
ContextManager.getAgentGeography(), ContextManager
                                                        .getAgentContext());
                        } catch (Exception e) {
                                throw new AgentCreationException(e);
                        }
                } else {
                        // TODO Implement agent creation from shapefile value;
                        throw new AgentCreationException("Have not implemented the
method of reading agent classes from a "
                                                + "shapefile yet.");
                }


                // Assign agents to houses
                int numAgents = 0;
                for (IAgent a : ContextManager.getAllAgents()) {
                        numAgents++;
                        Geometry g = ContextManager.getAgentGeometry(a);
                        for (Airport b :
SpatialIndexManager.search(ContextManager.AirportProjection, g)) {
                                if
(ContextManager.AirportProjection.getGeometry(b).contains(g)) {
						b.addAgent(a);
						a.setHome(b);
					}
				}
			}

			if (singleType) {
				LOGGER.info("Have created " + numAgents + " of type " +
clazz.getName().toString() + " from file "
						+ fileName);
			} else {
				// (NOTE: at the moment this will never happen because not
implemented yet.)
				LOGGER.info("Have created " + numAgents + " of different types
from file " + fileName);
			}

		}

		private void createAreaAgents(boolean dummy) throws AgentCreationException
{
			throw new AgentCreationException("Have not implemented the
createAreaAgents method yet.");
		}

		/**
		 * The methods that can be used to create agents. The CreateAgentMethod stuff
is just a long-winded way of
```

```
	 * hard-coding the specific method to use for creating agents into the enum
(much simpler in python).
	 *
	 * @author Nick Malleson
	 */
	private enum AGENT_FACTORY_METHODS {
		/** Default: create a number of agents randomly assigned to buildings */
		RANDOM("random", new CreateAgentMethod() {
			@Override
			public void createagents(boolean b, AgentFactory af) throws
AgentCreationException {
				af.createRandomAgents(b);
			}
		}),
		/** Specify an agent shapefile, one agent will be created per point */
		POINT_FILE("point", new CreateAgentMethod() {
			@Override
			public void createagents(boolean b, AgentFactory af) throws
AgentCreationException {
				af.createPointAgents(b);
			}
		}),
		/**
		 * Specify the number of agents per area as a shaefile. Agents will be
randomly assigned to houses within the
		 * area.
		 */
```

```
        AREA_FILE("area", new CreateAgentMethod() {

            @Override

            public void createagents(boolean b, AgentFactory af) throws
AgentCreationException {

                af.createAreaAgents(b);

            }

        });


        String stringVal;

        CreateAgentMethod meth;


        /**

         * @param val

         *          The string representation of the enum which must match the
method given in the 'agent_definition'

         *          parameter in parameters.xml.

         * @param f

         */

        AGENT_FACTORY_METHODS(String val, CreateAgentMethod f) {

            this.stringVal = val;

            this.meth = f;

        }


        public String toString() {

            return this.stringVal;

        }
```

```
                    public CreateAgentMethod createAgMeth() {

                              return this.meth;

                    }


                    interface CreateAgentMethod {

                              void createagents(boolean dummy, AgentFactory af) throws
AgentCreationException;

                    }

          }


}

/*

©vrnq5


*/


package scsi.environment;


import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Hashtable;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.Map;
import java.util.Vector;

import org.apache.commons.lang3.ArrayUtils;
//import java.util.;
import org.geotools.referencing.GeodeticCalculator;

import cern.colt.Arrays;

import com.vividsolutions.jts.geom.Coordinate;
import com.vividsolutions.jts.geom.Envelope;
import com.vividsolutions.jts.geom.Geometry;
import com.vividsolutions.jts.geom.GeometryFactory;
import com.vividsolutions.jts.geom.LineString;
import com.vividsolutions.jts.geom.Point;
import com.vividsolutions.jts.operation.distance.DistanceOp;

import repast.simphony.space.gis.Geography;
import repast.simphony.space.graph.RepastEdge;
import repast.simphony.space.graph.ShortestPath;
import scsi.environment.Junction;

import scsi.environment.NetworkEdge;

import scsi.agent.IAgent;

import scsi.exceptions.RoutingException;

import scsi.main.ContextManager;

import scsi.main.GlobalVars;


/**

 * Create routes around a GIS road network. The <code>setRoute</code> function
actually finds the route and can be

 * overridden by subclasses to create different types of Route. See the method
documentation for details of how routes

 * are calculated.

 *

 */

public class Route implements Cacheable {


        private static Logger LOGGER = Logger.getLogger(Route.class.getName());


        static {

                // Route.routeCache = new Hashtable<CachedRoute, CachedRoute>();

        }


        private IAgent agent;

        private Coordinate destination;

        private Airport destinationAirport;
	/*
	 * The route consists of a list of coordinates which describe how to get to the destination. Each coordinate might
	 * have an attached 'speed' which acts as a multiplier and is used to indicate whether or not the agent is
	 * traveling along a transport route (i.e. if a coordinate has an attached speed of '2' the agent will be able to
	 * get to the next coordinate twice as fast as they would do if they were walking). The current position indicate
	 * where in the lists of coords the agent is up to. Other attribute information about the route can be included as
	 * separate arrays with indices that match those of the 'route' array below.
	 */
	private int currentPosition;
	private List<Coordinate> routeX;
	//private List<Double> routeSpeedsX;
	/*
	 * This maps route coordinates to their containing Road, used so that when traveling we know which road/community
	 * the agent is on. private
	 */
	private List<Road> roadsX;

	// Record which function has added each coord, useful for debugging
	private List<String> routeDescriptionX;

	/*
```

```
	 * Cache every coordinate which forms a road so that Route.onRoad() is quicker.
Also save the Road(s) they are part
	 * of, useful for the agent's awareness space (see getRoadFromCoordCache()).
	 */
	private static volatile Map<Coordinate, List<Road>> coordCache;
	/*
	 * Cache the nearest road Coordinate to every building for efficiency (agents
usually/always need to get from the
	 * centroids of houses to/from the nearest road).
	 */
	private static volatile NearestRoadCoordCache nearestRoadCoordCache;
	/*
	 * Store which road every building is closest to. This is used to efficiently add
buildings to the agent's awareness
	 * space
	 */
	private static volatile AirportOnRoadCache airportOnRoadCache;
	// To stop threads competing for the cache:
	private static Object airportOnRoadCacheLock = new Object();


	/*
	 * Store a route once it has been created, might be used later (note that the
same object acts as key and value).
	 */
	// TODO Re-think route caching, would be better to cache the whole Route
object
	// private static volatile Map<CachedRoute, CachedRoute> routeCache;
	// /** Store a route distance once it has been created */
	// private static volatile Map<CachedRouteDistance, Double>
routeDistanceCache;


	/*
	 * Keep a record of the last community and road passed so that the same
buildings/communities aren't added to the
	 * cognitive map multiple times (the agent could spend a number of iterations on
the same road or community).
	 */
	private Road previousRoad;
	private Area previousArea;


	/**
	 * Creates a new Route object.
	 *
	 * @param destination
	 *          The agent's destination.
	 *
	 * @param destinationBuilding
	 *          The (optional) building they're heading to.
	 *
	 * @param type
	 *          The (optional) type of route, used by burglars who want to search.
	 */
	public Route(IAgent agent, Coordinate destination, Airport destinationAirport) {
		this.destination = destination;
		this.agent = agent;
		this.destinationAirport = destinationAirport;
	}

	/**
	 * Find a route from the origin to the destination. A route is a list of Coordinates
which describe the route to a
	 * destination restricted to a road network. The algorithm consists of three major
parts:
	 * <ol>
	 * <li>Find out if the agent is on a road already, if not then move to the nearest
road segment</li>
	 * <li>Get from the current location (probably mid-point on a road) to the
nearest junction</li>
	 * <li>Travel to the junction which is closest to our destination (using Dijkstra's
shortest path)</li>
	 * <li>Get from the final junction to the road which is nearest to the destination
	 * <li>
	 * <li>Move from the road to the destination</li>
	 * </ol>
	 *
	 * @throws Exception
	 */
	protected void setRoute() throws Exception {
		long time = System.nanoTime();
		// this.routeX = new ArrayList<Coordinate>();
		// this.roadsX = new ArrayList<Road>();
		// this.routeDescriptionX = new ArrayList<String>();
		// this.routeSpeedsX = new ArrayList<Double>();
		this.routeX = new Vector<Coordinate>();
		this.roadsX = new Vector<Road>();
		this.routeDescriptionX = new Vector<String>();
		//this.routeSpeedsX = new Vector<Double>();

		LOGGER.log(Level.FINER, "Planning route for: "
				+ this.agent.toString()
				+ " to: "
				+ this.destinationAirport.toString());
				//+ ((this.agent.getTransportAvailable() == null) ? "" :
"using transport: "
				//			+
this.agent.getTransportAvailable().toString()));
		if (atDestination()) {
			LOGGER.log(Level.WARNING, "Already at destination, cannot
create a route for " + this.agent.toString());
			return;
		}

		Coordinate currentCoord =
ContextManager.getAgentGeometry(this.agent).getCoordinate();
		Coordinate destCoord = this.destination;

		// See if a route has already been cached.
		// CachedRoute cachedRoute = new CachedRoute(currentCoord,
destCoord, this.agent.getTransportAvailable());
		// synchronized (Route.routeCache) {
		// if (Route.routeCache.containsKey(cachedRoute)) {
			// TempLogger.out("Route.setRoute, found a cached route from " +
currentCoord + " to " + destCoord
			// + " using available transport " + this.agent.getTransportAvailable() + ",
returning it.");
			// // Return a clone of the route that is stored in the cache
			// // TODO do we need clones here? I don't think so...
			// CachedRoute cr = Route.routeCache.get(cachedRoute);
			// // this.routeX = Cloning.copy(cr.getRoute());
			// // this.roadsX = new ArrayList<Road>(cr.getRoads());
			// // this.routeSpeedsX = new ArrayList<Double>(cr.getRouteSpeeds());
			// // this.routeDescriptionX = new ArrayList<String>(cr.getDescriptions());
			// this.routeX = new Vector<Coordinate>(cr.getRoute());
			// this.roadsX = new Vector<Road>(cr.getRoads());
			// this.routeSpeedsX = new Vector<Double>(cr.getRouteSpeeds());
			// this.routeDescriptionX = new Vector<String>(cr.getDescriptions());
			//
			// return;
			// }
			// } // synchronized

			// No route cached, have to create a new one (and cache it at the end).
			try {
				/*
				 * See if the current position and the destination are on road
segments. If the destination is not on a road
				 * segment we have to move to the closest road segment, then
onto the destination.
```

```
             */
            boolean destinationOnRoad = true;
            Coordinate finalDestination = null;
            if (!coordOnRoad(currentCoord)) {
                /*
                 * Not on a road so the first coordinate to add to the route
is the point on the closest road segment.
                 */
                currentCoord = getNearestRoadCoord(currentCoord);
                addToRoute(currentCoord, Road.nullRoad, "setRoute()
initial");
            }
            if (!coordOnRoad(destCoord)) {
                /*
                 * Not on a road, so need to set the destination to be the
closest point on a road, and set the
                 * destinationOnRoad boolean to false so we know to add
the final dest coord at the end of the route
                 */
                destinationOnRoad = false;
                finalDestination = destCoord; // Added to route at end of
alg.
                destCoord = getNearestRoadCoord(destCoord);
            }

            /*
             * Find the nearest junctions to our current position (road
endpoints)
```

```
                         */

                        // Start by Finding the road that this coordinate is on
                        /*
                         * TODO EFFICIENCY: often the agent will be creating a new route
from a building so will always find the
                         * same road, could use a cache. Even better, could implement a
cache in FindNearestObject() method!
                         */
                        Road currentRoad = Route.findNearestObject(currentCoord,
ContextManager.roadProjection, null,

        GlobalVars.GEOGRAPHY_PARAMS.BUFFER_DISTANCE.LARGE);
                        // Find which Junction is closest to us on the road.
                        List<Junction> currentJunctions = currentRoad.getJunctions();

                        /* Find the nearest Junctions to our destination (road endpoints)
*/

                        // Find the road that this coordinate is on
                        Road destRoad = Route.findNearestObject(destCoord,
ContextManager.roadProjection, null,

        GlobalVars.GEOGRAPHY_PARAMS.BUFFER_DISTANCE.SMALL);
                        // Find which Junction connected to the edge is closest to the
coordinate.
                        List<Junction> destJunctions = destRoad.getJunctions();
                        /*
```

```
                     * Now have four possible routes (2 origin junctions, 2 destination
junctions) need to pick which junctions
                     * form shortest route
                     */
                    Junction[] routeEndpoints = new Junction[2];
                    List<RepastEdge<Junction>> shortestPath =
getShortestRoute(currentJunctions, destJunctions, routeEndpoints);
                    // NetworkEdge<Junction> temp = (NetworkEdge<Junction>)
                    // shortestPath.get(0);
                    Junction currentJunction = routeEndpoints[0];
                    Junction destJunction = routeEndpoints[1];

                    /* Add the coordinates describing how to get to the nearest
junction */
                    List<Coordinate> tempCoordList = new Vector<Coordinate>();
                    this.getCoordsAlongRoad(currentCoord,
currentJunction.getCoords(), currentRoad, true, tempCoordList);
                    addToRoute(tempCoordList, currentRoad, 1,
"getCoordsAlongRoad (toJunction)");

                    /*
                     * Add the coordinates and speeds etc which describe how to
move along the chosen path
                     */
                    this.getRouteBetweenJunctions(shortestPath, currentJunction);

                    /*
```

```
                         * Add the coordinates describing how to get from the final
junction to the destination.
                         */

                        tempCoordList.clear();

        this.getCoordsAlongRoad(ContextManager.junctionGeography.getGeometry(des
tJunction).getCoordinate(),
                                        destCoord, destRoad, false, tempCoordList);
                        addToRoute(tempCoordList, destRoad, 1, "getCoordsAlongRoad
(fromJunction)");

                        if (!destinationOnRoad) {
                                addToRoute(finalDestination, Road.nullRoad, "setRoute
final");
                        }

                        // Check that a route has actually been created
                        checkListSizes();

                        // If the algorithm was better no coordinates would have been
duplicated
                        // removePairs();

                        // Check lists are still the same size.
                        checkListSizes();

                } catch (RoutingException e) {
```

```
				LOGGER.log(Level.SEVERE, "Route.setRoute(): Problem creating
route for " + this.agent.toString()
						+ " going from " + currentCoord.toString() + " to " +
this.destination.toString() + "("
						+ (this.destinationAirport == null ? "" :
this.destinationAirport.toString())
						+ ") See earlier messages error messages for more
info.");
				throw e;
			}
			// Cache the route and route speeds
			// List<Coordinate> routeClone = Cloning.copy(theRoute);
			// LinkedHashMap<Coordinate, Double> routeSpeedsClone =
Cloning.copy(this.routeSpeeds);
			// cachedRoute.setRoute(routeClone);
			// cachedRoute.setRouteSpeeds(routeSpeedsClone);

			// cachedRoute.setRoute(this.routeX, this.roadsX, this.routeSpeedsX,
this.routeDescriptionX);
			// synchronized (Route.routeCache) {
			// // Same cached route is both value and key
			// Route.routeCache.put(cachedRoute, cachedRoute);
			// }
			// TempLogger.out("...Route cacheing new route with unique id " +
cachedRoute.hashCode());

			LOGGER.log(Level.FINER, "Route Finished planning route for " +
this.agent.toString() + "with "
```

```
				+ this.routeX.size() + " coords in " + (0.000001 *
(System.nanoTime() - time)) + "ms.");


		// Finished, just check that the route arrays are all in sync

		assert this.roadsX.size() == this.routeX.size()

				//&& this.routeDescriptionX.size() ==
this.routeSpeedsX.size()

				&& this.roadsX.size() == this.routeDescriptionX.size();

	}


	private void checkListSizes() {

		assert this.roadsX.size() > 0 && this.roadsX.size() == this.routeX.size()

				//&& this.routeDescriptionX.size() ==
this.routeSpeedsX.size()

				&& this.roadsX.size() == this.routeDescriptionX.size() :
this.routeX.size() + "," + this.roadsX.size()

				+ "," + this.routeDescriptionX.size();// + "," +
this.routeSpeedsX.size();


	}


	/**

	 * Convenience function that can be used to add details to the route. This should
be used rather than updating

	 * individual lists because it makes sure that all lists stay in sync

	 *

	 * @param coord

	 *          The coordinate to add to the route
 * @param road
 *          The road that the coordinate is part of
 * @param speed
 *          The speed that the road can be travelled along
 * @param description
 *          A description of why the coordinate has been added
 */
private void addToRoute(Coordinate coord, Road road, String description) {
        this.routeX.add(coord);
        this.roadsX.add(road);
        //this.routeSpeedsX.add(speed);
        this.routeDescriptionX.add(description);
}


/**
 * A convenience for adding to the route that will add a number of coordinates
with the same description, road and
 * speed.
 *
 * @param coord
 *          A list of coordinates to add to the route
 * @param road
 *          The road that the coordinates are part of
 * @param speed
 *          The speed that the road can be travelled along
 * @param description
```

```
	 *            A description of why the coordinates have been added
	 */
	private void addToRoute(List<Coordinate> coords, Road road, double speed,
String description) {
		for (Coordinate c : coords) {
			this.routeX.add(c);
			this.roadsX.add(road);
			//this.routeSpeedsX.add(speed);
			this.routeDescriptionX.add(description);
		}
	}

	/**
	 * Travel towards our destination, as far as we can go this turn.
	 * <p>
	 * Also adds houses to the agent's cognitive environment. This is done by saving
each coordinate the person passes,
	 * creating a polygon with a radius given by the "cognitive_map_search_radius"
and adding all houses which touch the
	 * polygon.
	 * <p>
	 * 
	 * @param housesPassed
	 *            If not null then the buildings which the agent passed during their travels
this iteration will be
	 *            calculated and stored in this array. This can be useful if a agent needs to
know which houses it has
```

```
	 *	just passed and, therefore, which are possible victims. This isn't done by
default because it's quite
	 *	an expensive operation (lots of geographic tests which must be carried
out in each iteration). If the
	 *	array is null then the houses passed are not calculated.
	 * @return null or the buildings passed during this iteration if housesPassed
boolean is true
	 * @throws Exception
	 */
	public void travel() throws Exception {
		// Check that the route has been created
		if (this.routeX == null) {
			this.setRoute();
		}
		try {
			if (this.atDestination()) {
				return;
			}
			double time = System.nanoTime();

			// Store the roads the agent walks along (used to populate the
awareness space)
			// List<Road> roadsPassed = new ArrayList<Road>();
			double distTravelled = 0; // The distance travelled so far
			Coordinate currentCoord = null; // Current location
			Coordinate target = null; // Target coordinate we're heading for
(in route list)
```

```
			boolean travelledMaxDist = false; // True when travelled
maximum distance this iteration
			//double speed; // The speed to travel to next coord
			GeometryFactory geomFac = new GeometryFactory();
			currentCoord =
ContextManager.getAgentGeometry(this.agent).getCoordinate();

			while (!travelledMaxDist && !this.atDestination()) {
				target = this.routeX.get(this.currentPosition);
				//speed = this.routeSpeedsX.get(this.currentPosition);
				/*
				 * TODO Remember which roads have been passed, used
to work out what should be added to cognitive map.
				 * Only add roads once the agent has moved all the way
down them
				 */
				//
roadsPassed.add(this.roads.get(this.previousRouteCoord()));
				// Work out the distance and angle to the next coordinate
				double[] distAndAngle = new double[2];
				Route.distance(currentCoord, target, distAndAngle);
				// divide by speed because distance might effectively be
shorter

				double distToTarget = distAndAngle[0];// / speed;
				// If we can get all the way to the next coords on the route
then just go there
				if (distTravelled + distToTarget <
GlobalVars.GEOGRAPHY_PARAMS.TRAVEL_PER_TURN) {
```

```
                        distTravelled += distToTarget;
                        currentCoord = target;

                        // See if agent has reached the end of the route.
                        if (this.currentPosition == (this.routeX.size() - 1)) {
                            ContextManager.moveAgent(this.agent,
geomFac.createPoint(currentCoord));
                            //
ContextManager.agentGeography.move(this.agent,
geomFac.createPoint(currentCoord));
                            break; // Break out of while loop, have
reached end of route.
                        }
                        // Haven't reached end of route, increment the
counter
                        this.currentPosition++;
                    } // if can get all way to next coord

                    // Check if dist to next coordinate is exactly same as
maximum
                    // distance allowed to travel (unlikely but possible)
                    else if (distTravelled + distToTarget ==
GlobalVars.GEOGRAPHY_PARAMS.TRAVEL_PER_TURN) {
                        travelledMaxDist = true;
                        ContextManager.moveAgent(agent,
geomFac.createPoint(target));
                        // ContextManager.agentGeography.move(agent,
geomFac.createPoint(target));
```

```
                                        this.currentPosition++;
                                        LOGGER.log(Level.WARNING, "Travel(): UNUSUAL
CONDITION HAS OCCURED!");
                                } else {
                                        // Otherwise move as far as we can towards the
target along the road we're on.
                                        // Move along the vector the maximum distance
we're allowed this turn (take into account relative
                                        // speed)
                                        double distToTravel =
(GlobalVars.GEOGRAPHY_PARAMS.TRAVEL_PER_TURN - distTravelled);
                                        // Move the agent, first move them to the current
coord (the first part of the while loop doesn't do
                                        // this for efficiency)
                                        //
ContextManager.agentGeography.move(this.agent,
geomFac.createPoint(currentCoord));
                                        ContextManager.moveAgent(this.agent,
geomFac.createPoint(currentCoord));
                                        // Now move by vector towards target (calculated
angle earlier).
                                        ContextManager.moveAgentByVector(this.agent,
distToTravel, distAndAngle[1]);
                                        //
ContextManager.agentGeography.moveByVector(this.agent, distToTravel,
distAndAngle[1]);


                                        travelledMaxDist = true;
                                } // else
                        } // while
```

```
//                              this.printRoute();


                                /*
                                 * **TODO** Agent has finished moving, now just add all the buildings
and communities passed to their awareness
                                 * space (unless they're on a transport route). Note also that if on
a transport route without an associated
                                 * road no roads are added to the 'roads' map so even if the check
wasn't made here no buildings would be
                                 * added anyway.
                                 */
                                // Community c = null;
                                // if (!this.onTransportRoute) {
                                // String outputString = "Route.travel() adding following to
awareness space for '"
                                // + this.agent.toString() + "':";
                                // // roadsPassed will have duplicates, this is used to ignore them
                                // Road current = roadsPassed.get(0);
                                // // **TODO** The next stuff is a mess when it comes to adding
communities to the memory. Need to go
                                // // through and make sure communities aren't added too many
times (i.e. more than once for each journey)
                                // // and that they are always added when they should be.
                                //
                                // for (Road r : roadsPassed) { // last road in list is the one the
                                // // agent finishes iteration on
```

```
				// if (r != null && roadsPassed.get(0) != null && !current.equals(r))
{
				// // Check road isn't null () and that buildings on road haven't
already been added
				// // (road can be null when coords that aren't part of a road are
added to the route)
				// current = r;
				// if (r.equals(this.previousRoad)) {
				// // The agent has just passed over this road, don't add the
buildings or communities again
				// } else {
				// outputString += "\n\t" + r.toString() + ": ";
				// List<Building> passedBuildings = getBuildingsOnRoad(r);
				// List<Community> passedCommunities = new
ArrayList<Community>();
				// if (passedBuildings != null) { // There might not be any buildings
close to the road (unlikely)
				// outputString += passedBuildings.toString();
				// this.passedObjects(passedBuildings, Building.class);
				// // For efficiency just find one of the building's communities and
hope no other
				// // communities were passed through - NO! I'VE CHANGED THIS
BELOW!
				// c = passedBuildings.get(0).getCommunity();
				// // Check all buildings to make sure that if the agent has passed
more than one community
				// // then they are all added.
				// for (Building b : passedBuildings) {
				// if (!passedCommunities.contains(b.getCommunity())) {
```

```
                    // passedCommunities.add(b.getCommunity());
                    // }
                    // }
                    // for (Community com : passedCommunities) {
                    // if (com != null) {
                    // this.passedObject(com, Community.class);
                    // }
                    // }
                    //
                    // } else { // Community won't have been added because no
buildings passed, use slow method
                    // c =
GlobalVars.COMMUNITY_ENVIRONMENT.getObjectAt(Community.class, currentCoord);
                    // if (c != null) {
                    // this.passedObject(c, Community.class);
                    // }
                    // // TODO I think the following line is wrong, if the agent has
made
                    // // a long move they might have passed right through a
community that doesn't
                    // // have any buildings, perhaps this should check *all* the
communities that touch
                    // // the road, not just the community the agent finished the
move in (i.e. currentCoord)
                    //
passedCommunities.add(GlobalVars.COMMUNITY_ENVIRONMENT.getObjectAt(Commu
nity.class,
                    // currentCoord));
                    // }
                    // }
                    // }
                    // } // for roadsPassed
                    // TempLogger.out(outputString + "\n");
                    // } // if !onTransportRoute
                    // else {
                    // TempLogger.out("Route.travel() not adding to burglar '" +
this.agent.toString()
                    // + "' awareness space beecause on transport route: ");
                    // }
                    //
                    // // Finally set the previousRoad and previousCommunity so that
if these haven't changed in the next
                    // iteration they're not added to
                    // // the cognitive map again.
                    // this.previousRoad = roadsPassed.get(roadsPassed.size() - 1);
                    // // this.previousCommunity = c; // This was the most recent
community passed over
                    //
                    // TempLogger.out("...Finished Travelling(" + (0.000001 *
(System.nanoTime() - time)) + "ms)");
                    // // } // synchronized
GlobalVars.TRANSPORT_PARAMS.currentBurglar
             } catch (Exception e) {
                    LOGGER.log(Level.SEVERE, "Route.trave(): Caught error travelling
for " + this.agent.toString()
                                   + " going to " + "destination "
```

```
                                                + (this.destinationAirport == null ? "" :
this.destinationAirport.toString() + ")"));
                              throw e;
                    } // catch exception
          }


          /**
           * Get the distance (on a network) between the origin and destination. Take into
account the Burglar because they
           * might be able to speed up the route by using different transport methods.
Actually calculates the distance
           * between the nearest Junctions between the source and destination. Note that
the GRID environment doesn't have any
           * transport routes in it so all distances will always be the same regardless of the
agent.
           *
           * @param agent
           * @param destination
           * @return
           */
          public double getDistance(Coordinate origin, Coordinate destination) {


                    // // See if this distance has already been calculated
                    // if (Route.routeDistanceCache == null) {
                    // Route.routeDistanceCache = new Hashtable<CachedRouteDistance,
Double>();
                    // }
```

```
			// CachedRouteDistance crd = new CachedRouteDistance(origin,
destination, theBurglar.getTransportAvailable());
			//
			// synchronized (Route.routeDistanceCache) {
			// Double dist = Route.routeDistanceCache.get(crd);
			// if (dist != null) {
			// TempLogger.out("Route.ggetDistance, found a cached route distance
from " + origin + " to "
			// + destination + " using available transport " +
theBurglar.getTransportAvailable()
			// + ", returning it.");
			// return dist;
			// }
			// }
			// No distance in the cache, calculate it

		//	synchronized (GlobalVars.TRANSPORT_PARAMS.currentBurglarLock) {
		//		GlobalVars.TRANSPORT_PARAMS.currentAgent = theBurglar;
				// Find the closest Junctions to the origin and destination
				double minOriginDist = Double.MAX_VALUE;
				double minDestDist = Double.MAX_VALUE;
				double dist;
				Junction closestOriginJunc = null;
				Junction closestDestJunc = null;
				DistanceOp distOp = null;
				GeometryFactory geomFac = new GeometryFactory();
```

```
                        // TODO EFFICIENCY: here could iterate over near junctions
instead of all?
                        for (Junction j :
ContextManager.junctionContext.getObjects(Junction.class)) {
//                              // Check that the agent can actually get to the junction (if
might be part of a transport route
//                              // that the agent doesn't have access to)
//                              boolean accessibleJunction = false;
//                              accessibleJunc: for (RepastEdge<Junction> e :
ContextManager.roadNetwork.getEdges(j)) {
//                                      NetworkEdge<Junction> edge =
(NetworkEdge<Junction>) e;
//                                      for (String s : edge.getTypes()) {
//                                              if
(theBurglar.getTransportAvailable().contains(s)) {
//                                                      accessibleJunction = true;
//                                                      break accessibleJunc;
//                                              }
//                                      } // for types
//                              }// for edges
//                              if (!accessibleJunction) { // Agent can't get to the junction,
ignore it
//                                      continue;
//                              }
                                Point juncPoint = geomFac.createPoint(j.getCoords());


                                distOp = new DistanceOp(juncPoint,
geomFac.createPoint(origin));
                                dist = distOp.distance();
```

```
                    if (dist < minOriginDist) {
                        minOriginDist = dist;
                        closestOriginJunc = j;
                    }
                    // Destination
                    distOp = new DistanceOp(juncPoint,
geomFac.createPoint(destination));
                    dist = distOp.distance();
                    if (dist < minDestDist) {
                        minDestDist = dist;
                        closestDestJunc = j;
                    }
                } // for Junctions
                    // Return the shortest path plus the distance from the
origin/destination to their junctions
                    // TODO NOTE: Bug in ShortestPath so have to make
finalize is called, otherwise following lines are
                    // neater
                    // - MAYBE THIS HAS BEEN FIXED BY REPAST NOW.
                    // return (new
ShortestPath<Junction>(EnvironmentFactory.getRoadNetwork(),
                    // closestOriginJunc)).getPathLength(closestDestJunc)+
minOriginDist + minDestDist ;
                    // TODO : using non-deprecated methods don't work on
NGS, probably need to update repast libraries
                ShortestPath<Junction> p = new
ShortestPath<Junction>(ContextManager.roadNetwork, closestOriginJunc);
                double theDist = p.getPathLength(closestDestJunc);
```

```
                    // ShortestPath<Junction> p = new

                    //
ShortestPath<Junction>(EnvironmentFactory.getRoadNetwork());

                    // double theDist =
p.getPathLength(closestOriginJunc,closestDestJunc);

                    p.finalize();

                    p = null;

                    double finalDist = theDist + minOriginDist + minDestDist;

                    // // Cache this distance

                    // synchronized (Route.routeDistanceCache) {

                    // Route.routeDistanceCache.put(crd, finalDist);

                    // }

                    return finalDist;

                //} // synchronized


        }


        /**

         * Find the nearest coordinate which is part of a Road. Returns the coordinate
which is actually the closest to the

         * given coord, not just the corner of the segment which is closest. Uses the
DistanceOp class which finds the

         * closest points between two geometries.

         * <p>

         * When first called, the function will populate the 'nearestRoadCoordCache'
which calculates where the closest road

         * coordinate is to each building. The agents will commonly start journeys from
within buildings so this will
```

```
	 * improve efficiency.
	 * </p>
	 *
	 * @param inCoord
	 *            The coordinate from which to find the nearest road coordinate
	 * @return the nearest road coordinate
	 * @throws Exception
	 */
	private synchronized Coordinate getNearestRoadCoord(Coordinate inCoord)
throws Exception {
		// double time = System.nanoTime();

		synchronized (airportOnRoadCacheLock) {
			if (nearestRoadCoordCache == null) {
				LOGGER.log(Level.FINE, "Route.getNearestRoadCoord
called for first time, "
						+ "creating cache of all roads and the
buildings which are on them ...");
				// Create a new cache object, this will be read from disk if
				// possible (which is why the getInstance() method is used
				// instead of the constructor.
				String gisDir =
ContextManager.getProperty(GlobalVars.GISDataDirectory);
				File airportFile = new File(gisDir +
ContextManager.getProperty(GlobalVars.AirportShapefile));
				File roadsFile = new File(gisDir +
ContextManager.getProperty(GlobalVars.RoadShapefile));
```

```
					File serialisedLoc = new File(gisDir + 
ContextManager.getProperty(GlobalVars.AirportRoadsCoordsCache));


					nearestRoadCoordCache = 
NearestRoadCoordCache.getInstance(ContextManager.AirportProjection,
							airportFile, 
ContextManager.roadProjection, roadsFile, serialisedLoc, new GeometryFactory());
				} // if not cached
			} // synchronized
			return nearestRoadCoordCache.get(inCoord);
	}


	/**
	 * Finds the shortest route between multiple origin and destination junctions. 
Will return the shortest path and
	 * also, via two parameters, can return the origin and destination junctions which 
make up the shortest route.
	 * 
	 * @param currentJunctions
	 *          An array of origin junctions
	 * @param destJunctions
	 *          An array of destination junctions
	 * @param routeEndpoints
	 *          An array of size 2 which can be used to store the origin (index 0) and 
destination (index 1) Junctions
	 *          which form the endpoints of the shortest route.
	 * @return the shortest route between the origin and destination junctions
	 * @throws Exception
 */
	private List<RepastEdge<Junction>> getShortestRoute(List<Junction> 
currentJunctions, List<Junction> destJunctions,
			Junction[] routeEndpoints) throws Exception {
		double time = System.nanoTime();
	//	synchronized (GlobalVars.TRANSPORT_PARAMS.currentBurglarLock) {
			// This must be set so that NetworkEdge.getWeight() can adjust 
the weight depending on how this
			// particular agent is getting around the city
		//	GlobalVars.TRANSPORT_PARAMS.currentAgent = this.agent;
			double shortestPathLength = Double.MAX_VALUE;
			double pathLength = 0;
			ShortestPath<Junction> p;
			List<RepastEdge<Junction>> shortestPath = null;
			for (Junction o : currentJunctions) {
				for (Junction d : destJunctions) {
					if (o == null || d == null) {
						LOGGER.log(Level.WARNING, 
"Route.getShortestRoute() error: either the destination or origin "
								+ "junction is null. This can 
be caused by disconnected roads. It's probably OK"
								+ "to ignore this as a route 
should still be created anyway.");
					} else {
						p = new 
ShortestPath<Junction>(ContextManager.roadNetwork);
						pathLength = p.getPathLength(o,d);
						if (pathLength < shortestPathLength) {
```

```
                                        shortestPathLength = pathLength;
                                        shortestPath = p.getPath(o,d);
//                                      ShortestPath<Junction> p2 = new
ShortestPath<Junction>(ContextManager.roadNetwork);
//                                      shortestPath = p2.getPath(o, d);
//                                      p2.finalize();
//                                      p2 = null;
                                        // shortestPath = p1.getPath(o, d);
                                        // p1.finalize(); p1 = null;
                                        routeEndpoints[0] = o;
                                        routeEndpoints[1] = d;
                                    }
                                    // TODO See if the shortestpath bug has
been fixed, would make this unnecessary
                                    p.finalize();
                                    p = null;
                                } // if junc null
                            } // for dest junctions
                        } // for origin junctions
                        if (shortestPath == null) {
                            String debugString = "Route.getShortestRoute() could not
find a route. Looking for the shortest route between :\n";
                            for (Junction j : currentJunctions)
                                debugString += "\t" + j.toString() + ", roads: " +
j.getRoads().toString() + "\n";
                            for (Junction j : destJunctions)
                                debugString += "\t" + j.toString() + ", roads: " +
j.getRoads().toString() + "\n";
```

```
                                        throw new RoutingException(debugString);
                                }
                                LOGGER.log(Level.FINER, "Route.getShortestRoute (" + (0.000001
* (System.nanoTime() - time))
                                                + "ms) found shortest path " + "(length: " +
shortestPathLength + ") from "
                                                + routeEndpoints[0].toString() + " to " +
routeEndpoints[1].toString());
                                return shortestPath;
                        //} // synchronized
                }


                /**
                 * Calculates the coordinates required to move an agent from their current
position to the destination along a given
                 * road. The algorithm to do this is as follows:
                 * <ol>
                 * <li>Starting from the destination coordinate, record each vertex and check
inside the boundary of each line
                 * segment until the destination point is found.</li>
                 * <li>Return all but the last vertex, this is the route to the destination.</li>
                 * </ol>
                 * A boolean allows for two cases: heading towards a junction (the endpoint of
the line) or heading away from the
                 * endpoint of the line (this function can't be used to go to two midpoints on a
line)
                 *
                 * @param currentCoord
```

```
        * @param destinationCoord
        * @param road
        * @param toJunction
        *          whether or not we're traveling towards or away from a Junction
        * @param coordList
        *          A list which will be populated with the coordinates that the agent should
follow to move along the
        *          road.
        * @param roadList
        *          A list of roads associated with each coordinate.
        * @throws Exception
        */
       private void getCoordsAlongRoad(Coordinate currentCoord, Coordinate
destinationCoord, Road road,
                       boolean toJunction, List<Coordinate> coordList) throws
RoutingException {

               Route.checkNotNull(currentCoord, destinationCoord, road, coordList);

               double time = System.nanoTime();
               Coordinate[] roadCoords =
ContextManager.roadProjection.getGeometry(road).getCoordinates();

               // Check that the either the destination or current coordinate are actually
part of the road
               boolean currentCorrect = false, destinationCorrect = false;
               for (int i = 0; i < roadCoords.length; i++) {
                       if (toJunction && destinationCoord.equals(roadCoords[i])) {
                           destinationCorrect = true;
                           break;
                       } else if (!toJunction && currentCoord.equals(roadCoords[i])) {
                           currentCorrect = true;
                           break;
                       }
               } // for

               if (!(destinationCorrect || currentCorrect)) {
                       String roadCoordsString = "";
                       for (Coordinate c : roadCoords)
                           roadCoordsString += c.toString() + " - ";
                       throw new RoutingException("Neigher the origin or destination
nor the current"
                            + "coordinate are part of the road '" +
road.toString() + "' (person '" + this.agent.toString()
                            + "').\n" + "Road coords: " + roadCoordsString +
"\n" + "\tOrigin: " + currentCoord.toString()
                            + "\n" + "\tDestination: " +
destinationCoord.toString() + " ( "
                            + this.destinationAirport.toString() + " )\n " +
"Heading " + (toJunction ? "to" : "away from")
                            + " a junction, so " + (toJunction ? "destination" :
"origin")
                            + " should be part of a road segment");
               }

               // Might need to reverse the order of the road coordinates
```

```
			if (toJunction &&
!destinationCoord.equals(roadCoords[roadCoords.length - 1])) {
				// If heading towards a junction, destination coordinate must be
at end of road segment
				ArrayUtils.reverse(roadCoords);
			} else if (!toJunction && !currentCoord.equals(roadCoords[0])) {
				// If heading away form junction current coord must be at
beginning of road segment
				ArrayUtils.reverse(roadCoords);
			}
			GeometryFactory geomFac = new GeometryFactory();
			Point destinationPointGeom = geomFac.createPoint(destinationCoord);
			Point currentPointGeom = geomFac.createPoint(currentCoord);
			// If still false at end then algorithm hasn't worked
			boolean foundAllCoords = false;
			search: for (int i = 0; i < roadCoords.length - 1; i++) {
				Coordinate[] segmentCoords = new Coordinate[] { roadCoords[i],
roadCoords[i + 1] };
				// Draw a small buffer around the line segment and look for the
coordinate within the buffer
				Geometry buffer =
geomFac.createLineString(segmentCoords).buffer(GlobalVars.GEOGRAPHY_PARAMS.BU
FFER_DISTANCE.SMALL.dist);
				if (!toJunction) {
					/* If heading away from a junction, keep adding road
coords until we find the destination */
					coordList.add(roadCoords[i]);
					if (destinationPointGeom.within(buffer)) {
						coordList.add(destinationCoord);
```

```
                        foundAllCoords = true;
                        break search;
                    }
                } else if (toJunction) {
                    /*
                     * If heading towards a junction: find the curent coord, add
it to the route, then add all the remaining
                     * coords which make up the road segment
                     */
                    if (currentPointGeom.within(buffer)) {
                        for (int j = i + 1; j < roadCoords.length; j++) {
                            coordList.add(roadCoords[j]);
                        }
                        coordList.add(destinationCoord);
                        foundAllCoords = true;
                        break search;
                    }
                }
            } // for
            if (foundAllCoords) {
                LOGGER.log(Level.FINER, "getCoordsAlongRoad (" + (0.000001 *
(System.nanoTime() - time)) + "ms)");
                return;
            } else { // If we get here then the route hasn't been created
                // A load of debugging info
                String error = "Route: getCoordsAlongRoad: could not find
destination coordinates "
```

```
                                        + "along the road.\n\tHeading *" + (toJunction ?
"towards" : "away from")
                                        + "* a junction.\n\t Person: " + this.agent.toString()
+ ")\n\tDestination building: "
                                        + destinationAirport.toString() + "\n\tRoad causing
problems: " + road.toString()
                                        + "\n\tRoad vertex coordinates: " +
Arrays.toString(roadCoords);
                        throw new RoutingException(error);
                        /*
                         * Hack: ignore the error, printing a message and just returning
the origin destination and coordinates.
                         * This means agent will jump to/from the junction but I can't
figure out why the fuck it occasionally
                         * doesn't work!! It's so rare that hopefully this isn't a problem.
                         */
                        // TempLogger.err("Route: getCoordsAlongRoad: error... (not
debugging).");
                        // List<Coord> coords = new ArrayList<Coord>();
                        // coords.add(currentCoord);
                        // coords.add(destinationCoord);
                        // for (Coord c : coords)
                        // this.roads.put(c, road); // Remember the roads each coord is
                        // // part of
                        // return coords;

                }
        }
```

```
	private static void checkNotNull(Object... args) throws RoutingException {
		for (Object o : args) {
			if (o == null) {
				throw new RoutingException("An input argument is null");
			}
		}
		return;
	}


	/**
	 * Returns all the coordinates that describe how to travel along a path, restricted
to road coordinates. In some
	 * cases the route wont have an associated road, this occurs if the route is part of
a transport network. In this
	 * case just the origin and destination coordinates are added to the route.
	 *
	 * @param shortestPath
	 * @param startingJunction
	 *            The junction the path starts from, this is required so that the algorithm
knows which road coordinate
	 *            to add first (could be first or last depending on the order that the road
coordinates are stored
	 *            internally).
	 * @return the coordinates as a mapping between the coord and its associated
speed (i.e. how fast the agent can
	 *         travel to the next coord) which is dependent on the type of edge and the
agent (e.g.
	 *	driving/walking/bus). LinkedHashMap is used to guarantee the insertion
order of the coords is maintained.
	 * @throws RoutingException
	 */
	private void getRouteBetweenJunctions(List<RepastEdge<Junction>>
shortestPath, Junction startingJunction)
			throws RoutingException {
		double time = System.nanoTime();
		if (shortestPath.size() < 1) {
			// This could happen if the agent's destination is on the same road
			// as the origin
			return;
		}
		// Lock the currentAgent so that NetworkEdge objects know what speed
to use (depends on transport available to
		// the specific agent).
	//	synchronized (GlobalVars.TRANSPORT_PARAMS.currentBurglarLock) {
	//		GlobalVars.TRANSPORT_PARAMS.currentAgent = this.agent;

			// Iterate over all edges in the route adding coords and weights as
appropriate
			NetworkEdge<Junction> e;
			Road r;
			// Use sourceFirst to represent whether or not the edge's source
does actually represent the start of the
			// edge (agent could be going 'forwards' or 'backwards' over edge
			boolean sourceFirst;
			for (int i = 0; i < shortestPath.size(); i++) {
```

```
                    e = (NetworkEdge<Junction>) shortestPath.get(i);
                    if (i == 0) {
                         // No coords in route yet, compare the source to
the starting junction
                         sourceFirst =
(e.getSource().equals(startingJunction)) ? true : false;
                    } else {
                         // Otherwise compare the source to the last coord
added to the list
                         sourceFirst =
(e.getSource().getCoords().equals(this.routeX.get(this.routeX.size() - 1))) ? true
                                             : false;
                    }
                    /*
                     * Now add the coordinates describing how to move along
the road. If there is no road associated with
                     * the edge (i.e. it is a transport route) then just add the
source/dest coords. Note that the shared
                     * coordinates between two edges will be added twice,
these must be removed later
                     */
                    r = e.getRoad();
                    /*
                     * Get the speed that the agent will be able to travel along
this edge (depends on the transport
                     * available to the agent and the edge). Some speeds will
be < 1 if the agent shouldn't be using this
                     * edge but doesn't have any other way of getting to the
destination. in these cases set speed to 1
```

```
                                 * (equivalent to walking).
                                 */
//                              double speed = e.getSpeed();
//                              if (speed < 1)
//                                      speed = 1;

                                if (r == null) { // No road associated with this edge (it is a
                                                                        // transport link) so
just add source
                                        if (sourceFirst) {
                                                this.addToRoute(e.getSource().getCoords(),
r, "getRouteBetweenJunctions - no road");
                                        //      this.addToRoute(e.getTarget().getCoords(),
r, "getRouteBetweenJunctions - no road");
                                                // (Note speed = -1 used because we don't
know the weight to the next
                                                // coordinate - this can be removed later)
                                        } else {
                                                this.addToRoute(e.getTarget().getCoords(),
r, "getRouteBetweenJunctions - no road");
                                        //      this.addToRoute(e.getSource().getCoords(),
r, "getRouteBetweenJunctions - no road");
                                        }
                                } else {
                                        // This edge is a road, add all the coords which
make up its geometry
                                        Coordinate[] roadCoords =
ContextManager.roadProjection.getGeometry(r).getCoordinates();
                                        if (roadCoords.length < 2)
                                        throw new
RoutingException("Route.getRouteBetweenJunctions: for some reason road " + "'"
                                                    + r.toString() + "' doesn't
have at least two coords as part of its geometry ("
                                                    + roadCoords.length + ")");
                                    // Make sure the coordinates of the road are added
in the correct order
                                    if (!sourceFirst) {
                                        ArrayUtils.reverse(roadCoords);
                                    }
                                    // Add all the road geometry's coords
                                    for (int j = 0; j < roadCoords.length; j++) {
                                        this.addToRoute(roadCoords[j], r,
"getRouteBetweenJuctions - on road");
                                        // (Note that last coord will have wrong
weight)
                                    } // for roadCoords.length
                                } // if road!=null
                            }
                            // Check all lists are still the same size.
                            assert this.roadsX.size() == this.routeX.size()
                                    //&& this.routeDescriptionX.size() ==
this.routeSpeedsX.size()
                                    && this.roadsX.size() ==
this.routeDescriptionX.size();

                            // Check all lists are still the same size.
                            assert this.roadsX.size() == this.routeX.size()
```

```
                                                //&& this.routeDescriptionX.size() ==
this.routeSpeedsX.size()
                                                && this.roadsX.size() ==
this.routeDescriptionX.size();


                        // Finished!
                        LOGGER.log(Level.FINER, "getRouteBetweenJunctions (" +
(0.000001 * (System.nanoTime() - time)) + "ms");
                        return;
//              } // synchronized
        } // getRouteBetweenJunctions


        /**
         * Determine whether or not the person associated with this Route is at their
destination. Compares their current
         * coordinates to the destination coordinates (must be an exact match).
         *
         * @return True if the person is at their destination
         */
        public boolean atDestination() {
                return
ContextManager.getAgentGeometry(this.agent).getCoordinate().equals(this.destination
);
        }


        // /**
        // * Removes any duplicate coordinates from the curent route (coordinates
which
```

```
        // * are the same *and* next to each other in the list).
        // * <p>
        // * If my route-generating algorithm was better this would't be necessary.
        // */
        // @Deprecated
        // private void removePairs() throws RoutingException {
        // if (this.routeX.size() < 1) {
        // // No coords to iterate over, probably something has gone wrong
        // throw new RoutingException("Route.removeDuplicateCoordinates():
WARNING an empty list has been "
        // + "passed to this function, something has probably gone wrong");
        // }
        // TempLogger.out("ROUTE BEFORE REMOVING PAIRS");
        // this.printRoute();
        //
        // // (setRoute() has already checked that lists are same size)
        //
        // // Iterate over the list, removing coordinates that are the same as their
neighbours.
        // // (and associated objects in other lists)
        // Iterator<Road> roadIt = this.roadsX.iterator();
        // Iterator<Coordinate> routeIt = this.routeX.iterator();
        // Iterator<Double> routeSpeedIt = this.routeSpeedsX.iterator();
        // Iterator<String> routeDescIt = this.routeDescriptionX.iterator();
        // Coordinate c1, c2;
        // Road currentRoad = roadIt.next();
        // Road nextRoad = null;
        // routeIt.next(); routeSpeedIt.next(); routeDescIt.next();
        // while ( roadIt.hasNext() ) {
        // nextRoad = roadIt.next();
        // routeIt.next();
        // routeSpeedIt.next();
        // routeDescIt.next();
        //
        // c1 = currentRoad.getCoords();
        // c2 = nextRoad.getCoords();
        //
        // if (c1.equals(c2)) {
        // // Remove objects from the lists
        // roadIt.remove();
        // routeIt.remove();
        // routeSpeedIt.remove();
        // routeDescIt.remove();
        // }
        // else {
        // currentRoad = nextRoad;
        // }
        // }
        //
        // TempLogger.out("ROUTE AFTER REMOVING PAIRS");
        // this.printRoute();
        // }
```

```
	private void printRoute() {
		StringBuilder out = new StringBuilder();
		out.append("Printing route (" + this.agent.toString() + "). Current position
in list is "
				+ this.currentPosition + " ('" +
this.routeDescriptionX.get(this.currentPosition) + "')");
		for (int i = 0; i < this.routeX.size(); i++) {
			out.append("\t(" + this.agent.toString() + ") " +
this.routeX.get(i).toString() + "\t"
					//+ this.routeSpeedsX.get(i).toString() +
					+ "\t" + this.roadsX.get(i) + "\t"
					+ this.routeDescriptionX.get(i));
		}
		LOGGER.info(out.toString());
	}



	/**
	 * Find the nearest object in the given geography to the coordinate.
	 *
	 * @param <T>
	 * @param x
	 *          The coordinate to search from
	 * @param geography
	 *          The given geography to look through
	 * @param closestPoints
	 *	An optional List that will be populated with the closest points to x (i.e.
the results of
	 *	<code>distanceOp.closestPoints()</code>.
	 * @param searchDist
	 *	The maximum distance to search for objects in. Small distances are
more efficient but larger ones are
	 *	less likely to find no objects.
	 * @return The nearest object.
	 * @throws RoutingException
	 *	 If an object cannot be found.
	 */
	public static synchronized <T> T findNearestObject(Coordinate x, Geography<T>
geography,
			List<Coordinate> closestPoints,
GlobalVars.GEOGRAPHY_PARAMS.BUFFER_DISTANCE searchDist)
			throws RoutingException {
		if (x == null) {
			throw new RoutingException("The input coordinate is null, cannot
find the nearest object");
		}


		T nearestObject = SpatialIndexManager.findNearestObject(geography, x,
closestPoints, searchDist);


		// Old way without using spatial index:
		//
		// GeometryFactory geomFac = new GeometryFactory();
		// Point point = geomFac.createPoint(x);
// // TODO Use an expanding buffer that starts small but gets bigger if no
object is found.
//
// Geometry buffer = point.buffer(searchDist.dist);
// double minDist = Double.MAX_VALUE;
// T nearestObject = null;
// for (T t : geography.getObjectsWithin(buffer.getEnvelopeInternal())) {
// DistanceOp distOp = new DistanceOp(point,
geography.getGeometry(t));
// double thisDist = distOp.distance();
// if (thisDist < minDist) {
// minDist = thisDist;
// nearestObject = t;
// // Optionally record the closest points
// if (closestPoints != null) {
// closestPoints.clear();
// // TODO clean conversion of array to List (don't have access
// // to internet!)
// Coordinate[] crds = distOp.closestPoints();
// List<Coordinate> temp = new ArrayList(crds.length);
// for (Coordinate c : crds)
// temp.add(c);
// closestPoints.addAll(temp);
// }
// } // if thisDist < minDist
// } // for nearRoads
if (nearestObject == null) {
					throw new RoutingException("Couldn't find an object close to
these coordinates:\n\t" + x.toString());

			} else {

					return nearestObject;

			}

	}


	/**

	 * Returns the angle of the vector from p0 to p1 relative to the x axis

	 * <p>

	 * The angle will be between -Pi and Pi. I got this directly from the JUMP program
source.

	 *

	 * @return the angle (in radians) that p0p1 makes with the positive x-axis.

	 */

	public static synchronized double angle(Coordinate p0, Coordinate p1) {

			double dx = p1.x - p0.x;

			double dy = p1.y - p0.y;


			return Math.atan2(dy, dx);

	}


	/**

	 * The building which this Route is targeting

	 * IMPORTANT ADD MORE HERE
MOSTLY!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

	 *
	 *
	 *
	 *
	 *
	 * @return the destinationHouse
	 */
	public Airport getDestinationAirport() {
		if (this.destinationAirport == null) {
			LOGGER.log(Level.WARNING, "Route: getDestinationAirport(),
warning, no airportbuilding has "
					+ "been set. This might be ok, the agent might be
supposed to be heading to a coordinate "
					+ "not a particular airport(?)");
			return null;
		}
		return destinationAirport;
	}

	/**
	 * The coordinate the route is targeting
	 *
	 * @return the destination
	 */
	public Coordinate getDestination() {
		return this.destination;
	}
	/**
	 * Maintain a cache of all coordinates which are part of a road segment. Store
the coords and all the road(s) they
	 * are part of.
	 *
	 * @param coord
	 *          The coordinate which should be part of a road geometry
	 * @return The road(s) which the coordinate is part of or null if the coordinate is
not part of any road
	 */
	private List<Road> getRoadFromCoordCache(Coordinate coord) {

		populateCoordCache(); // Check the cache has been populated
		return coordCache.get(coord);
	}

	/**
	 * Test if a coordinate is part of a road segment.
	 *
	 * @param coord
	 *          The coordinate which we want to test
	 * @return True if the coordinate is part of a road segment
	 */
	private boolean coordOnRoad(Coordinate coord) {
		populateCoordCache(); // check the cache has been populated
		return coordCache.containsKey(coord);
	}
	private synchronized static void populateCoordCache() {

		double time = System.nanoTime();
		if (coordCache == null) { // Fist check cache has been created
			coordCache = new HashMap<Coordinate, List<Road>>();
			LOGGER.log(Level.FINER,
					"Route.populateCoordCache called for first time,
creating new cache of all Road coordinates.");
		}
		if (coordCache.size() == 0) { // Now popualte it if it hasn't already
										// been
populated
			LOGGER.log(Level.FINER, "Route.populateCoordCache: is empty,
creating new cache of all Road coordinates.");

			for (Road r :
ContextManager.roadContext.getObjects(Road.class)) {
				for (Coordinate c :
ContextManager.roadProjection.getGeometry(r).getCoordinates()) {
					if (coordCache.containsKey(c)) {
						coordCache.get(c).add(r);
					} else {
						List<Road> l = new ArrayList<Road>();
						l.add(r);
						// TODO Need to put *new* coordinate
here? Not use
						// existing one in memory?
```

```
                                        coordCache.put(new Coordinate(c), l);
                                }
                        }
                }

                LOGGER.log(Level.FINER, "... finished caching all road coordinates
(in " + 0.000001
                                * (System.nanoTime() - time) + "ms)");
        }
    }

    /**
     * Find the buildings which can be accessed from the given road (the given road
is the closest to the buildings).
     * Uses a separate cache object which can be serialised so that the cache doesn't
need to be rebuilt every time.
     * THE OTHER BUILDINGS WILL GO HERE!!!
     *
     * !!
     * !
     * !
     * !
     * !
     * !
     *
     * @param road
     * @return
```

```
         * @throws Exception
         */
        private List<Airport> getAirportOnRoad(Road road) throws Exception {
                if (airportOnRoadCache == null) {
                        LOGGER.log(Level.FINER, "Route.getAirportOnRoad called for first
time, "
                                        + "creating cache of all roads and the buildings
which are on them ...");
                        // Create a new cache object, this will be read from disk if possible
(which is why the
                        // getInstance() method is used instead of the constructor.
                        String gisDir = GlobalVars.GISDataDirectory;
                        File airportFile = new File(gisDir + GlobalVars.AirportShapefile);
                        File roadsFile = new File(gisDir + GlobalVars.RoadShapefile);
                        File serialLoc = new File(gisDir +
ContextManager.getProperty(GlobalVars.AirportRoadsCache));
                        airportOnRoadCache =
AirportOnRoadCache.getInstance(ContextManager.AirportProjection, airportFile,
                                        ContextManager.roadProjection, roadsFile,
serialLoc, new GeometryFactory());
                } // if not cached
                return airportOnRoadCache.get(road);
        }


        /**
         * Calculate the distance (in meters) between two Coordinates, using the
coordinate reference system that the
         * roadGeography is using. For efficiency it can return the angle as well (in the
range -0 to 2PI) if returnVals
```

```
	 * passed in as a double[2] (the distance is stored in index 0 and angle stored in
index 1).
	 *
	 * @param c1
	 * @param c2
	 * @param returnVals
	 *          Used to return both the distance and the angle between the two
Coordinates. If null then the distance
	 *          is just returned, otherwise this array is populated with the distance at
index 0 and the angle at
	 *          index 1.
	 * @return The distance between Coordinates c1 and c2.
	 */
	public static synchronized double distance(Coordinate c1, Coordinate c2,
double[] returnVals) {
		// TODO check this now, might be different way of getting distance in
new Simphony
		GeodeticCalculator calculator = new
GeodeticCalculator(ContextManager.roadProjection.getCRS());
		calculator.setStartingGeographicPoint(c1.x, c1.y);
		calculator.setDestinationGeographicPoint(c2.x, c2.y);
		double distance = calculator.getOrthodromicDistance();
		if (returnVals != null && returnVals.length == 2) {
			returnVals[0] = distance;
			double angle = Math.toRadians(calculator.getAzimuth()); // Angle
in range -PI to PI
			// Need to transform azimuth (in range -180 -> 180 and where 0
points north)
			// to standard mathematical (range 0 -> 360 and 90 points north)
```

```
                if (angle > 0 && angle < 0.5 * Math.PI) { // NE Quadrant
                    angle = 0.5 * Math.PI - angle;
                } else if (angle >= 0.5 * Math.PI) { // SE Quadrant
                    angle = (-angle) + 2.5 * Math.PI;
                } else if (angle < 0 && angle > -0.5 * Math.PI) { // NW Quadrant
                    angle = (-1 * angle) + 0.5 * Math.PI;
                } else { // SW Quadrant
                    angle = -angle + 0.5 * Math.PI;
                }
                returnVals[1] = angle;
            }
            return distance;
    }


    /**
     * Converts a distance lat/long distance (e.g. returned by DistanceOp) to meters.
The calculation isn't very
     * accurate because (probably) it assumes that the distance is between two
points that lie exactly on a line of
     * longitude (i.e. one is exactly due north of the other). For this reason the value
shouldn't be used in any
     * calculations which is why it's returned as a String.
     *
     * @param dist
     *          The distance (as returned by DistanceOp) to convert to meters
     * @return The approximate distance in meters as a String (to discourage using
this approximate value in
```

```
         *       calculations).
         * @throws Exception
         * @see com.vividsolutions.jts.operation.distance.DistanceOp
         */
        public static synchronized String distanceToMeters(double dist) throws
Exception {
                // Works by creating two coords (close to a randomly chosen object)
which are a certain distance apart
                // then using similar method as other distance() function
                GeodeticCalculator calculator = new
GeodeticCalculator(ContextManager.roadProjection.getCRS());
                Coordinate c1 =
ContextManager.AirportContext.getRandomObject().getCoords();
                calculator.setStartingGeographicPoint(c1.x, c1.y);
                calculator.setDestinationGeographicPoint(c1.x, c1.y + dist);
                return String.valueOf(calculator.getOrthodromicDistance());
        }

        public void clearCaches() {
                if (coordCache != null)
                        coordCache.clear();
                if (nearestRoadCoordCache != null) {
                        nearestRoadCoordCache.clear();
                        nearestRoadCoordCache = null;
                }

                if (airportOnRoadCache != null) {
```

```
                airportOnRoadCache.clear();
                airportOnRoadCache = null;
        }
        // if (routeCache != null) {
        // routeCache.clear();
        // routeCache = null;
        // }
        // if (routeDistanceCache != null) {
        // routeDistanceCache.clear();
        // routeDistanceCache = null;
        // }
}

// /**
// * Will add the given buildings to the awareness space of the Burglar who is
// * being controlled by this Route. Also tells the burglar which buildings
// * have been passed if appropriate, this is needed for agents who are
// * currently looking for a burglary target.
// *
// * @param buildings
// * A list of buildings
// */
// @SuppressWarnings("unchecked")
// protected <T> void passedObjects(List<T> objects, Class<T> clazz) {
// this.agent.addToMemory(objects, clazz);
// if (clazz.isAssignableFrom(Building.class)) {
		// // System.out.println("Route.passedObjects(): "+objects.toString());
		// this.agent.buildingsPassed((List<Building>) objects);
		// }
		// }

		/**
		 * Will add the given buildings to the awareness space of the Burglar who is
being controlled by this Route.
		 *
		 * @param buildings
		 *            A list of buildings
		 */
		protected <T> void passedObject(T object, Class<T> clazz) {
			List<T> list = new ArrayList<T>(1);
			list.add(object);
			this.agent.addToMemory(list, clazz);
		}

}

/*
************************************************************************
*/

/**
 * Class can be used to store a cache of all roads and the buildings which can be
accessed by them (a map of
 * Road<->List<Building>. Buildings are 'accessed' by traveling to the road which is
nearest to them.
 * <p>
 * This class can be serialized so that if the GIS data doesn't change it doesn't have to be
re-calculated each time.
 * However, the Roads and Buildings themselves cannot be serialised because if they are
there will be two sets of Roads
 * and BUildings, the serialised ones and those that were created when the model was
initialised. To get round this, an
 * array which contains the road and building ids is serialised and the cache is re-built
using these caches ids after
 * reading the serialised cache. This means that the id's given to Buildings and Roads
must not change (i.e.
 * auto-increment numbers are no good because if a simulation is restarted the static
auto-increment variables will not
 * be reset to 0).
 *
 *
 */
class AirportOnRoadCache implements Serializable {

	private static Logger LOGGER =
Logger.getLogger(AirportOnRoadCache.class.getName());

	private static final long serialVersionUID = 1L;
	// The actual cache, this isn't serialised
	private static transient Hashtable<Road, ArrayList<Airport>> theCache;
	// The 'reference' cache, stores the building and road ids and can be
	// serialised
	private Hashtable<String, ArrayList<String>> referenceCache;


	// Check that the road/building data hasn't been changed since the cache was
	// last created
	private File airportFile;
	private File roadsFile;
	// The location that the serialised object might be found.
	private File serialisedLoc;
	// The time that this cache was created, can be used to check data hasn't
	// changed since
	private long createdTime;


	// Private constructor because getInstance() should be used
	private AirportOnRoadCache(Geography<Airport> airportEnvironment, File
airportFile,
				Geography<Road> roadEnvironment, File roadsFile, File
serialisedLoc, GeometryFactory geomFac)
				throws Exception {
		// this.buildingEnvironment = buildingEnvironment;
		// this.roadEnvironment = roadEnvironment;
		this.airportFile = airportFile;
		this.roadsFile = roadsFile;
		this.serialisedLoc = serialisedLoc;
		theCache = new Hashtable<Road, ArrayList<Airport>>();
		this.referenceCache = new Hashtable<String, ArrayList<String>>();
```

```
        LOGGER.log(Level.FINE, "airportOnRoadCache() creating new cache with
data (and modification date):\n\t"
                    + this.airportFile.getAbsolutePath() + " (" + new
Date(this.airportFile.lastModified()) + ")\n\t"
                    + this.roadsFile.getAbsolutePath() + " (" + new
Date(this.roadsFile.lastModified()) + ")\n\t"
                    + this.serialisedLoc.getAbsolutePath());

        populateCache(airportEnvironment, roadEnvironment, geomFac);
        this.createdTime = new Date().getTime();
        serialise();
    }

    public void clear() {
        theCache.clear();
        this.referenceCache.clear();

    }

    private void populateCache(Geography<Airport> airportEnvironment,
Geography<Road> roadEnvironment,
            GeometryFactory geomFac) throws Exception {
        double time = System.nanoTime();
        for (Airport b : airportEnvironment.getAllObjects()) {
            // Find the closest road to this building
            Geometry airportPoint = geomFac.createPoint(b.getCoords());
            double minDistance = Double.MAX_VALUE;
```

```
                        Road closestRoad = null;
                        double distance;
                        Envelope e =
airportPoint.buffer(GlobalVars.GEOGRAPHY_PARAMS.BUFFER_DISTANCE.LARGE.dist)
                                        .getEnvelopeInternal();
                        for (Road r : roadEnvironment.getObjectsWithin(e)) {
                                distance = DistanceOp.distance(airportPoint,
ContextManager.roadProjection.getGeometry(r));
                                if (distance < minDistance) {
                                        minDistance = distance;
                                        closestRoad = r;
                                }
                        } // for roads
                                // Found the closest road, add the information to the
cache
                        if (theCache.containsKey(closestRoad)) {
                                theCache.get(closestRoad).add(b);

        this.referenceCache.get(closestRoad.getIdentifier()).add(b.getIdentifier());
                        } else {
                                ArrayList<Airport> l = new ArrayList<Airport>();
                                l.add(b);
                                theCache.put(closestRoad, l);
                                ArrayList<String> l2 = new ArrayList<String>();
                                l2.add(b.getIdentifier());
                                this.referenceCache.put(closestRoad.getIdentifier(), l2);
                        }
```

```
		} // for buildings
		int numRoads = theCache.keySet().size();
		int numBuildings = 0;
		for (List<Airport> l : theCache.values())
			numBuildings += l.size();
		LOGGER.log(Level.FINER, "Finished caching roads and buildings. Cached "
+ numRoads + " roads and "
				+ numBuildings + " buildings in " + 0.000001 *
(System.nanoTime() - time) + "ms");
	}

	public List<Airport> get(Road r) {
		return theCache.get(r);
	}

	private void serialise() throws IOException {
		double time = System.nanoTime();
		FileOutputStream fos = null;
		ObjectOutputStream out = null;
		try {
			if (!this.serialisedLoc.exists())
				this.serialisedLoc.createNewFile();
			fos = new FileOutputStream(this.serialisedLoc);
			out = new ObjectOutputStream(fos);
			out.writeObject(this);
			out.close();
		} catch (IOException ex) {
```

```
                if (serialisedLoc.exists())
                    serialisedLoc.delete(); // delete to stop problems loading
incomplete file next time
                throw ex;
            }
            LOGGER.log(Level.FINER, "Serialised AirportOnRoadCache to " +
this.serialisedLoc.getAbsolutePath() + " in ("
                    + 0.000001 * (System.nanoTime() - time) + "ms)");
    }


    /**
     * Used to create a new BuildingsOnRoadCache object. This function is used
instead of the constructor directly so
     * that the class can check if there is a serialised version on disk already. If not
then a new one is created and
     * returned.
     *
     * @param buildingEnv
     * @param buildingsFile
     * @param roadEnv
     * @param roadsFile
     * @param serialisedLoc
     * @param geomFac
     * @return
     * @throws Exception
     */
    public synchronized static AirportOnRoadCache
getInstance(Geography<Airport> airportEnv, File airportFile,
			Geography<Road> roadEnv, File roadsFile, File serialisedLoc,
GeometryFactory geomFac) throws Exception {

		double time = System.nanoTime();

		// See if there is a cache object on disk.

		if (serialisedLoc.exists()) {

			FileInputStream fis = null;

			ObjectInputStream in = null;

			AirportOnRoadCache bc = null;

			try {

				fis = new FileInputStream(serialisedLoc);

				in = new ObjectInputStream(fis);

				bc = (AirportOnRoadCache) in.readObject();

				in.close();


				// Check that the cache is representing the correct data
and the

				// modification dates are ok

				// (WARNING, if this class is re-compiled the serialised
object

				// will still be read in).

				if
(!airportFile.getAbsolutePath().equals(bc.airportFile.getAbsolutePath())

						||
!roadsFile.getAbsolutePath().equals(bc.roadsFile.getAbsolutePath())

						|| airportFile.lastModified() >
bc.createdTime || roadsFile.lastModified() > bc.createdTime) {

					LOGGER.log(Level.FINER, "BuildingsOnRoadCache,
found serialised object but it doesn't match the "
```

```
                                                    + "data (or could have different
modification dates), will create a new cache.");
                              } else {
                                        // Have found a useable serialised cache. Now use
the cached
                                        // list of id's to construct a
                                        // new cache of buildings and roads.
                                        // First need to buld list of existing roads and
buildings
                                        Hashtable<String, Road> allRoads = new
Hashtable<String, Road>();
                                        for (Road r : roadEnv.getAllObjects())
                                                allRoads.put(r.getIdentifier(), r);
                                        Hashtable<String, Airport> allAirport = new
Hashtable<String, Airport>();
                                        for (Airport b : airportEnv.getAllObjects())
                                                allAirport.put(b.getIdentifier(), b);

                                        // Now create the new cache
                                        theCache = new Hashtable<Road,
ArrayList<Airport>>();

                                        for (String roadId : bc.referenceCache.keySet()) {
                                                ArrayList<Airport> buildings = new
ArrayList<Airport>();
                                                for (String airportId :
bc.referenceCache.get(roadId)) {

        buildings.add(allAirport.get(airportId));
```

```
                                                }

                                                theCache.put(allRoads.get(roadId),
buildings);

                                        }

                                        LOGGER.log(Level.FINER, "BuildingsOnRoadCache,
found serialised cache, returning it (in "

                                                        + 0.000001 * (System.nanoTime() -
time) + "ms)");

                                        return bc;

                                }

                        } catch (IOException ex) {

                                if (serialisedLoc.exists())

                                        serialisedLoc.delete(); // delete to stop problems
loading incomplete file next tinme

                                throw ex;

                        } catch (ClassNotFoundException ex) {

                                if (serialisedLoc.exists())

                                        serialisedLoc.delete();

                                throw ex;

                        }


                }


                // No serialised object, or got an error when opening it, just create a

                // new one

                return new AirportOnRoadCache(airportEnv, airportFile, roadEnv,
roadsFile, serialisedLoc, geomFac);

        }
```

```
}


/*
**********************************************************************
*/


/**
 * Caches the nearest road Coordinate to every building for efficiency (agents
usually/always need to get from the
 * centroids of houses to/from the nearest road).
 * <p>
 * This class can be serialised so that if the GIS data doesn't change it doesn't have to be
re-calculated each time.
 *
 *
 */
class NearestRoadCoordCache implements Serializable {


	private static Logger LOGGER =
Logger.getLogger(NearestRoadCoordCache.class.getName());


	private static final long serialVersionUID = 1L;
	private Hashtable<Coordinate, Coordinate> theCache; // The actual cache
	// Check that the road/building data hasn't been changed since the cache was
	// last created
	private File airportFile;
	private File roadsFile;
	// The location that the serialised object might be found.
```

```
        private File serialisedLoc;
        // The time that this cache was created, can be used to check data hasn't
        // changed since
        private long createdTime;

        private GeometryFactory geomFac;

        private NearestRoadCoordCache(Geography<Airport> airportEnvironment, File
airportFile,
                        Geography<Road> roadEnvironment, File roadsFile, File
serialisedLoc, GeometryFactory geomFac)
                        throws Exception {

                this.airportFile = airportFile;
                this.roadsFile = roadsFile;
                this.serialisedLoc = serialisedLoc;
                this.theCache = new Hashtable<Coordinate, Coordinate>();
                this.geomFac = geomFac;

                LOGGER.log(Level.FINE, "NearestRoadCoordCache() creating new cache
with data (and modification date):\n\t"
                                + this.airportFile.getAbsolutePath() + " (" + new
Date(this.airportFile.lastModified()) + ") \n\t"
                                + this.roadsFile.getAbsolutePath() + " (" + new
Date(this.roadsFile.lastModified()) + "):\n\t"
                                + this.serialisedLoc.getAbsolutePath());

                populateCache(airportEnvironment, roadEnvironment);
```

```
		this.createdTime = new Date().getTime();

		serialise();

	}


	public void clear() {

		this.theCache.clear();

	}


	private void populateCache(Geography<Airport> airportEnvironment,
Geography<Road> roadEnvironment)

			throws Exception {

		double time = System.nanoTime();

		theCache = new Hashtable<Coordinate, Coordinate>();

		// Iterate over every building and find the nearest road point

		for (Airport b : airportEnvironment.getAllObjects()) {

			List<Coordinate> nearestCoords = new ArrayList<Coordinate>();

			Route.findNearestObject(b.getCoords(), roadEnvironment,
nearestCoords,


	GlobalVars.GEOGRAPHY_PARAMS.BUFFER_DISTANCE.LARGE);

			// Two coordinates returned by closestPoints(), need to find the
one

			// which isn't the building coord

			Coordinate nearestPoint = null;

			for (Coordinate c : nearestCoords) {

				if (!c.equals(b.getCoords())) {

					nearestPoint = c;
```

```
                                        break;
                                }
                        } // for nearestCoords
                        if (nearestPoint == null) {
                                throw new Exception("Route.getNearestRoadCoord()
error: couldn't find a road coordinate which "
                                                + "is close to building " + b.toString());
                        }
                        theCache.put(b.getCoords(), nearestPoint);
                }// for Buildings
                LOGGER.log(Level.FINER, "Finished caching nearest roads (" + (0.000001 *
(System.nanoTime() - time)) + "ms)");
        } // if nearestRoadCoordCache = null;


        /**
         *
         * @param c
         * @return
         * @throws Exception
         */
        public Coordinate get(Coordinate c) throws Exception {
                if (c == null) {
                        throw new Exception("Route.NearestRoadCoordCache.get()
error: the given coordinate is null.");
                }
                double time = System.nanoTime();
                Coordinate nearestCoord = this.theCache.get(c);
```

```
		if (nearestCoord != null) {
			LOGGER.log(Level.FINER, "NearestRoadCoordCache.get() (using
cache) - ("
					+ (0.000001 * (System.nanoTime() - time)) + "ms)");
			return nearestCoord;
		}
		// If get here then the coord is not in the cache, agent not starting their
journey from a house, search for
		// it manually. Search all roads in the vicinity, looking for the point which
is nearest the person
		double minDist = Double.MAX_VALUE;
		Coordinate nearestPoint = null;
		Point coordGeom = this.geomFac.createPoint(c);

		// Note: could use an expanding envelope that starts small and gets
bigger
		double bufferDist =
GlobalVars.GEOGRAPHY_PARAMS.BUFFER_DISTANCE.LARGE.dist;
		double bufferMultiplier = 1.0;
		Envelope searchEnvelope = coordGeom.buffer(bufferDist *
bufferMultiplier).getEnvelopeInternal();
		StringBuilder debug = new StringBuilder(); // incase the operation fails

		for (Road r :
ContextManager.roadProjection.getObjectsWithin(searchEnvelope)) {

			DistanceOp distOp = new DistanceOp(coordGeom,
ContextManager.roadProjection.getGeometry(r));
			double thisDist = distOp.distance();
```

```
                        // BUG?: if an agent is on a really long road, the long road will not
be found by getObjectsWithin because
                        // it is not within the buffer
                        debug.append("\troad ").append(r.toString()).append(" is
").append(thisDist).append(
                                " distance away (at closest point). ");


                        if (thisDist < minDist) {
                            minDist = thisDist;
                            Coordinate[] closestPoints = distOp.closestPoints();
                            // Two coordinates returned by closestPoints(), need to
find the
                            // one which isn''t the coord parameter
                            debug.append("Closest points
(").append(closestPoints.length).append(") are: ").append(
                                    Arrays.toString(closestPoints));
                            nearestPoint = (c.equals(closestPoints[0])) ?
closestPoints[1] : closestPoints[0];
                            debug.append("Nearest point is
").append(nearestPoint.toString());
                            nearestPoint = (c.equals(closestPoints[0])) ?
closestPoints[1] : closestPoints[0];
                        } // if thisDist < minDist
                        debug.append("\n");


                    } // for nearRoads


                    if (nearestPoint != null) {
```

```
                LOGGER.log(Level.FINER, "NearestRoadCoordCache.get() (not
using cache) - ("
                        + (0.000001 * (System.nanoTime() - time)) + "ms)");
                return nearestPoint;
            }
            /* IF HERE THEN ERROR, PRINT DEBUGGING INFO */
            StringBuilder debugIntro = new StringBuilder(); // Some extra info for
debugging
            debugIntro.append("Route.NearestRoadCoordCache.get() error: couldn't
find a coordinate to return.\n");
            Iterable<Road> roads =
ContextManager.roadProjection.getObjectsWithin(searchEnvelope);
            debugIntro.append("Looking for nearest road coordinate around
").append(c.toString()).append(".\n");
            debugIntro.append("RoadEnvironment.getObjectsWithin() returned
").append(
                    ContextManager.sizeOfIterable(roads) + " roads, printing
debugging info:\n");
            debugIntro.append(debug);
            throw new Exception(debugIntro.toString());


        }


        private void serialise() throws IOException {
            double time = System.nanoTime();
            FileOutputStream fos = null;
            ObjectOutputStream out = null;
            try {
```

```
                        if (!this.serialisedLoc.exists())
                                this.serialisedLoc.createNewFile();
                        fos = new FileOutputStream(this.serialisedLoc);
                        out = new ObjectOutputStream(fos);
                        out.writeObject(this);
                        out.close();
                } catch (IOException ex) {
                        if (serialisedLoc.exists()) {
                                // delete to stop problems loading incomplete file next
time
                                serialisedLoc.delete();
                        }
                        throw ex;
                }
                LOGGER.log(Level.FINE, "... serialised NearestRoadCoordCache to " +
this.serialisedLoc.getAbsolutePath()
                                + " in (" + 0.000001 * (System.nanoTime() - time) + "ms)");
        }


        /**
         * Used to create a new AirportsOnRoadCache object. This function is used
instead of the constructor directly so
         * that the class can check if there is a serialised version on disk already. If not
then a new one is created and
         * returned.
         *
         * @param buildingEnv
```

```
	 * @param buildingsFile
	 * @param roadEnv
	 * @param roadsFile
	 * @param serialisedLoc
	 * @param geomFac
	 * @return
	 * @throws Exception
	 */
	public synchronized static NearestRoadCoordCache
getInstance(Geography<Airport> airportEnv, File airportFile,
			Geography<Road> roadEnv, File roadsFile, File serialisedLoc,
GeometryFactory geomFac) throws Exception {
		double time = System.nanoTime();
		// See if there is a cache object on disk.
		if (serialisedLoc.exists()) {
			FileInputStream fis = null;
			ObjectInputStream in = null;
			NearestRoadCoordCache ncc = null;
			try {

				fis = new FileInputStream(serialisedLoc);
				in = new ObjectInputStream(fis);
				ncc = (NearestRoadCoordCache) in.readObject();
				in.close();

				// Check that the cache is representing the correct data
and the
```

```
                        // modification dates are ok
                        if
(!airportFile.getAbsolutePath().equals(ncc.airportFile.getAbsolutePath())
                                    ||
!roadsFile.getAbsolutePath().equals(ncc.roadsFile.getAbsolutePath())
                                    || airportFile.lastModified() >
ncc.createdTime || roadsFile.lastModified() > ncc.createdTime) {
                              LOGGER.log(Level.FINE, "BuildingsOnRoadCache,
found serialised object but it doesn't match the "
                                          + "data (or could have different
modification dates), will create a new cache.");
                        } else {
                              LOGGER.log(Level.FINER,
"NearestRoadCoordCache, found serialised cache, returning it (in "
                                          + 0.000001 * (System.nanoTime() -
time) + "ms)");
                              return ncc;
                        }
                  } catch (IOException ex) {
                        if (serialisedLoc.exists())
                              serialisedLoc.delete(); // delete to stop problems
loading incomplete file next tinme
                        throw ex;
                  } catch (ClassNotFoundException ex) {
                        if (serialisedLoc.exists())
                              serialisedLoc.delete();
                        throw ex;
                  }
```

```
		}

		// No serialised object, or got an error when opening it, just create a new
one
		return new NearestRoadCoordCache(airportEnv, airportFile, roadEnv,
roadsFile, serialisedLoc, geomFac);
	}

}

/**
 * Used to cache routes. Saves the origin and destination coords and the transport
available to the agent (if transport
 * changes then the agent might have to create a new route.
 *
 *
 */
class CachedRoute {
	private List<Coordinate> theRoute;
	private List<Double> routeSpeeds;
	private List<String> routeDescriptions;
	private List<Road> roads;
	private Coordinate origin;
	private Coordinate destination;
	private List<String> transportAvailable;
	// Used to generate hash codes (each route must have unique ID)
	private static int uniqueRouteCacheID;
	private int uniqueID;


	public CachedRoute(Coordinate origin, Coordinate destination, List<String>
transportAvailable) {
		this.origin = origin;
		this.destination = destination;
		this.transportAvailable = transportAvailable;
		this.uniqueID = CachedRoute.uniqueRouteCacheID++;
	}


	public void setRoute(List<Coordinate> theRoute, List<Road> roads, List<Double>
routeSpeeds,
			List<String> routeDescriptions) {
		this.theRoute = theRoute;
		this.roads = roads;
		this.routeSpeeds = routeSpeeds;
		this.routeDescriptions = routeDescriptions;
	}


	public List<Coordinate> getRoute() {
		return this.theRoute;
	}


	public List<Double> getRouteSpeeds() {
		return this.routeSpeeds;
	}
	public List<Road> getRoads() {
		return this.roads;
	}

	public List<String> getDescriptions() {
		return this.routeDescriptions;
	}

	@Override
	public String toString() {
		return "CachedRoute " + this.uniqueID;
	}

	/**
	 * Returns true if input object is a CachedRoute and the the origin, destination
and transport available are the
	 * same as this CachedRoute
	 */
	@Override
	public boolean equals(Object obj) {
		if (obj instanceof CachedRoute) {
			CachedRoute r = (CachedRoute) obj;
			return (r.origin.equals(this.origin)) &&
(r.destination.equals(this.destination))
					&&
(r.transportAvailable.equals(this.transportAvailable));
		} else {
				return false;
			}
		}

		/**
		 * Returns:
<code>Float.floatToIntBits((float)(this.origin.getX()+this.origin.getY()))</code>
		 */
		@Override
		public int hashCode() {
			return Float.floatToIntBits((float) (this.origin.x + this.origin.y));
		}
}

/**
 * Used to cache route distances. Saves the origin and destination coords and the
transport available to the agent (if
 * transport changes then the agent might have to create a new route).
 *
 * @author Nick Malleson
 */
class CachedRouteDistance {
		private Coordinate origin;
		private Coordinate destination;
		private List<String> transportAvailable;
		private static int uniqueRouteCacheID; // Used to generate hash codes (each
                                                                        //
route must have unique ID)
	private int uniqueID;

	// private List<Coord> theRoute; // The actual route (a list of coords)

	public CachedRouteDistance(Coordinate origin, Coordinate destination,
List<String> transportAvailable) {
		this.origin = origin;
		this.destination = destination;
		this.transportAvailable = transportAvailable;
		this.uniqueID = CachedRouteDistance.uniqueRouteCacheID++;
	}

	@Override
	public String toString() {
		return "CachedRouteDistance " + this.uniqueID;
	}

	/**
	 * Returns true if input object is a CachedRoute and the the origin, destination
and transport available are the
	 * same as this CachedRoute. Because routes are non-directional the origins and
destinations are interchangeable.
	 */
	@Override
	public boolean equals(Object obj) {
		if (obj instanceof CachedRouteDistance) {
			CachedRouteDistance r = (CachedRouteDistance) obj;
			return ((r.origin.equals(this.origin) &&
r.destination.equals(this.destination)) || (r.origin
					.equals(this.destination) &&
r.destination.equals(this.origin)))
					&&
r.transportAvailable.equals(this.transportAvailable);
		} else {
			return false;
		}
	}

	/**
	 * Returns:
<code>Float.floatToIntBits((float)(this.origin.getX()+this.origin.getY()))</code>
	 */
	@Override
	public int hashCode() {
		return Float.floatToIntBits((float) (this.origin.x + this.origin.y));
	}
}

/**
 * Convenience class for creating deep copies of lists/maps (copies the values stored as
well). Haven't made this
 * generic because need access to constructors to create new objects (e.g. new
Coord(c))
```

```
 */
final class Cloning {

	public static List<Coordinate> copy(List<Coordinate> in) {

		List<Coordinate> out = new ArrayList<Coordinate>(in.size());
		for (Coordinate c : in) {
			// TODO Check this Coordinate constructor does what I expect it
to
			out.add(new Coordinate(c));
		}
		return out;
	}

	// Not used now that route speeds are a list, not a map
	// public static LinkedHashMap<Coordinate, Double>
	// copy(LinkedHashMap<Coordinate, Double> in) {
	//
	// LinkedHashMap<Coordinate, Double> out = new LinkedHashMap<Coordinate,
	// Double>(in.size());
	// for (Coordinate c : in.keySet()) {
	// out.put(c, in.get(c));
	// }
	// return out;
	// }
// private List<Coord> theRoute; // The actual route (a list of coords)


        public CachedRouteDistance(Coordinate origin, Coordinate destination,
List<String> transportAvailable) {
                this.origin = origin;
                this.destination = destination;
                this.transportAvailable = transportAvailable;
                this.uniqueID = CachedRouteDistance.uniqueRouteCacheID++;
        }
```

VITA

Varun Ramachandran was born in Bombay, Maharashtra in the Western part of India. He finished his schooling from New Delhi, India and graduated from Vellore Institute of Technology, Vellore, India with a Bachelor's of Science degree in Information Technology in 2011. He started his Master's degree in Engineering Management at Missouri University of Science and Technology (Rolla) in August 2011 and graduated in December 2012. From January 2013 he worked towards his PhD degree and graduated in May 2015. He has published papers in various conferences and his dissertations in various journals. He has also given many presentations on his research at conferences all over the world. He received the best Master's research award at the 2012 Engineering Management and Systems Engineering symposium held at Missouri S&T. He was a student member of ASEM, ISERC, and AAG. His primary research interests are in Supply Chain Management, Operations Research and Project Management. Varun also held a Graduate Research Assistantship under Dr. Suzanna Long where he worked on Restoration of Critical Infrastructure in the Aftermath of an Extreme Event. Varun also interned with ABB Inc., Greenville, South Carolina and held the post of Supply Chain Manager for a three month period.