This Dissertation

entitled

CREATING, UPDATING AND VALIDATING SIMULATIONS

IN A DYNAMIC DATA-DRIVEN APPLICATION SYSTEM

typeset with NDdiss2$_\varepsilon$ v1.0 (2004/06/15) on July 20, 2007 for

Timothy W. Schoenharl

This LaTeX 2$_\varepsilon$ classfile conforms to the University of Notre Dame style guidelines established in Spring 2004. However it is still possible to generate a non-conformant document if the published instructions are not followed! Be sure to refer to the published Graduate School guidelines at `http://graduateschool.nd.edu` as well.

It is YOUR resposnsibility to ensure that the Chapter titles and Table caption titles are put in CAPS LETTERS. This classfile does *NOT* do that! This way, you have total control over how you want the symbols and sub-/superscripts in titles and captions look like.

This summary page can be disabled by specifying the nosummary *option to the class invocation.* (i.e., `\documentclass[...,nosummary,...]{nddiss2e}`)

THIS PAGE IS *NOT* PART OF THE THESIS, BUT MUST BE TURNED IN TO THE PROOFREADER!

NDdiss2$_\varepsilon$ documentation can be found at these locations:

`http://www.gsu.nd.edu`
`http://graduateschool.nd.edu`

CREATING, UPDATING AND VALIDATING SIMULATIONS IN A DYNAMIC DATA-DRIVEN APPLICATION SYSTEM

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Timothy W. Schoenharl, B.S., M.S.

Greg Madey, Director

Graduate Program in Computer Science and Engineering
Notre Dame, Indiana
July 2007

CREATING, UPDATING AND VALIDATING SIMULATIONS IN A DYNAMIC DATA-DRIVEN APPLICATION SYSTEM

Abstract

by

Timothy W. Schoenharl

This work addresses research questions important to the Dynamic, Data-Driven Application Systems (DDDAS) community: specifically how to create, update and validate simulations instantiated from streaming sensor data. The use of Agent-Based Modeling simulations in an online context presents several challenges: simulations must demonstrate good runtime characteristics, yet in order to fit in our validation framework, the simulations must be modular and allow for alternative models of human behavior. We present a simulation of pedestrian movement we have developed according to our revised simulation design criteria, built using techniques from Design Patterns and Pattern Oriented Modeling. We present a thorough evaluation of the simulation in terms of model validation, simulation design and runtime characteristics. We present and evaluate methods for online validation of Agent-Based Models (ABM). We introduce an aggregate method for online creation and updating of ABM simulations and evaluate the approach against alternatives. We have developed answers to these questions through the design and implementation of a DDDAS application, the WIPER system. The Wireless Integrated Phone-based Emergency Response system is designed to provide emergency responders with timely information on the status of a

city or region, as well as the capability to detect, follow and possibly predict crisis events. WIPER uses the DDDAS approach to process streaming information from a cellular phone service provider to detect and predict crisis events.

We demonstrate that real time simulation of pedestrian crowds is possible with our Agent-Based Modeling simulation and present upper bounds on the population size that can be simulated in real time. We show that for certain validation measures, our validation approach yields 100% accuracy at selecting model type based on simulation output. The results of our updating approach demonstrates that aggregate updating outperforms one-to-one agent-to-referent reparameterization under certain conditions and provides empirical evidence suggesting the effects of naive realism.

DEDICATION

For Ping and Marcus.

ACKNOWLEDGMENTS

This work has been supported by a Fellowship from the Arthur J. Schmitt Foundation. The research on WIPER is supported by an NSF Grant, CISE/CNS-DDDAS, Award #0540348.

I would also like to thank my advisor, Prof. Greg Madey, for his immense support, encouragement and confidence in my abilities.

CONTENTS

FIGURES

TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION AND MOTIVATION

Creating, updating and validating Agent-Based Modeling simulations against streaming data changes the rules of simulation development and requires modelers to reconsider conventional notions about data collection, the modeling process and how much human interaction should be required to judge the results of a simulation. In this Dissertation we challenge the conventional usage scenario for simulations and consider how to address the challenges of designing simulations that interface with streaming data.

Simulation has become an important paradigm for conducting research in science and engineering. Traditionally, a scientist will observe a phenomenon in the real world, either through direct observation or via some type of sensors. Data is collected, processed and analyzed. The scientist then will pose a hypothesis, often a model that explains the phenomenon, and will design and implement a simulation that can test the model, where the simulation is instantiated and validated based on observations from the real world [44]. At every step in the process, the scientist is an active participant. This approach needs to be augmented to address situations where data collection is continuous and simulations are "in the loop" with the data collection process. Under such circumstances it is not possible to

continute to rely on human decision makers to interpret intermediate simulation results.

Recently, the NSF Blue Ribbon Panel on Simulation-Based Engineering Science identified key challenges to the advancement of simulation-based research [75]. Among these challenges was the ability to close the loop between observation and simulation, integrating simulations with measurement systems. The DDDAS approach seeks to address that challenge by incorporating real time sensor data into running simulations[95]. Dynamic, Data-Driven Application Systems is an approach to developing systems incorporating sensors and simulations where the simulations receive streaming data from the sensors and the sensors receive control information from the simulations.

We present work that addresses several open research questions relating to DDDAS: how to create, update and validate Agent-Based Modeling simulations against streaming sensor data. These questions have been addresses through the design and implementation of the Wireless Phone-based Emergency Response (WIPER) system, a DDDAS application. WIPER works to detect and predict the course of crisis events and advise emergency response planners with up to date information on crisis scenarios. WIPER works by monitoring a stream of cell phone activity data to detect anomalies, then uses Agent-Based Modeling simulations, created from and updated against the streaming data, to provide more thorough understanding of crisis events. Finally, the information regarding potential anomalies, call information and simulation output is presented to users in a web-based console.

1.2 RESEARCH QUESTIONS

In order to successfully develop the WIPER project, several challenges from the DDDAS approach needed to be addressed. Here we lay out the research questions and outline our approach to answering them.

1.2.1 CREATING AND UPDATING OF SIMULATIONS FROM STREAMING DATA

The primary research question addressed by this work is how to periodically update Agent-Based Simulations with real time streaming sensor data. This is an open question. It is possible to update an equation-based simulation, as new information simply leads to a re-parameterization of the equations. In an equation-based simulation, the parameters of the equations define the state of the phenomenon to be studied. Researchers in the area of discrete event simulations as far back as 1998 have recognized the challenges posed to updating simulations with streaming data [29]. The problem becomes potentially more challenging in an Agent-Based Modeling simulation. Here, the phenomenon to be studied is an emergent property of the system [44]. As in chaotic systems, these systems often display sensitivity to initial conditions and path dependency, where slight changes in input parameters can cause large fluctuations in future system states. Simply creating an agent to represent each individual in the sensor stream, then updating the individual's parameters (location, call activity, movement, etc) may not be sufficient. To do so may emphasize unimportant aspects of agent behavior, and fails to recognize that it is the aggregate behavior of all of the agents that defines the phenomenon.

What we propose instead is a novel approach that draws its inspiration from

empirical science (specifically the paradigm of hypothesis generation and testing against observation)[112] and pattern-oriented modeling[44]. We create ensembles of Agent-Based Simulations, covering several variations of crisis events and with a range of parameters. These simulations run for a short period of time and are then validated against streaming data from the actual crisis location. Simulations that are within an acceptable threshold given our validation framework are carried over to the next round and the ensemble is filled out with simulations that are variations on the validated simulations. The details of this approach and an evaluation of its implementation in the context of the WIPER project are given in Chapter 5.

1.3 CONTRIBUTIONS

The design and implementation of the WIPER system has required solving important research problems that can be generalized and transmitted to the community at large. Below we briefly outline these research contributions and direct interested readers to the appropriate chapters. Figure 1.1 shows the research contributions that are presented in this dissertation.

1.3.1 WIPER: OVERVIEW OF A DDDAS SYSTEM

The WIPER system is a proof-of-concept DDDAS system. An overview of the system and its contributions to the DDDAS community are given in Chapter 3. The work presented there is an introduction to the goals of the project and its contributions to the Emergency Response and Crisis Management field.

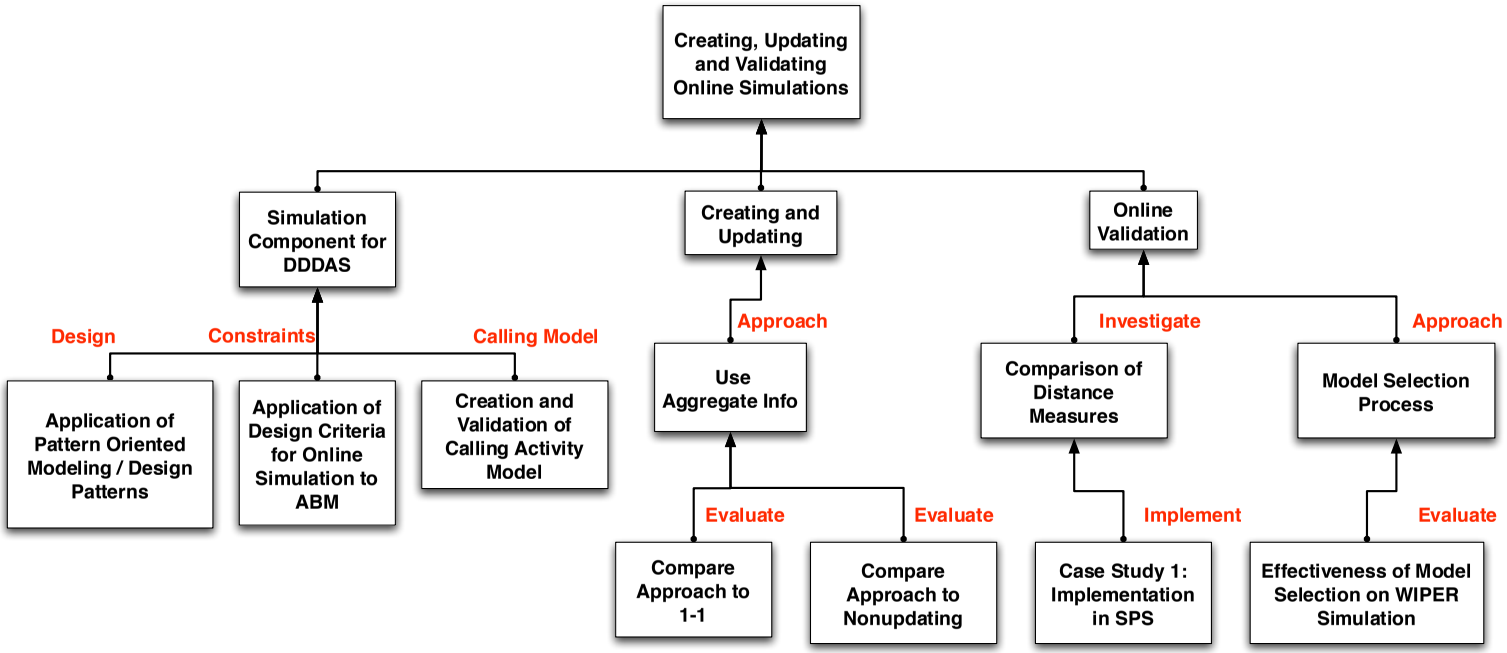

Figure 1.1: An Overview of the Research Contributions in this Dissertation.

1.3.2 DESIGN AND IMPLEMENTATION OF THE WIPER SIMULATION

A simulation is an important component of any DDDAS system. We have created a simulation using the Agent-Based Modeling paradigm that is designed to simulate cell phone-using pedestrians. As the simulation will be used in the WIPER project, it was important to design it to be created from and updated with streaming data. A thorough description of the design and implementation of the WIPER simulation is given in Chapter 4. The description includes an overview of the traditional model development process, including the application of verification and validation techniques.

1.3.3 UPDATING RUNNING SIMULATIONS WITH STREAMING DATA

The primary contribution of this research is to demonstrate our approach, in the context of the WIPER system, for adaptively updating Agent-Based Simulations with real time, streaming sensor data. As outlined above, the criteria for success are stringent: The simulations cannot be constructed with "naive realism" [70] (as cited in [44]), they should model the phenomenon but not attempt to accurately predict the movements of every individual. The method we use, ensembles of simulations with updating based on streaming data, is presented and analyzed in Chapter 5.

1.3.4 EVALUATION OF APPROACHES FOR ONLINE VALIDATION OF AGENT-BASED MODELS

In Chapter 6 we present an evaluation of several approaches to online validation of Agent-Based Models in the context of the WIPER project. The process of validating simulations is iterative and often involves subjective judgments made

by domain experts [7]. The need for human interpretation is a serious limitation of traditional validation approaches and limits their usefulness in the context of online validation. Simulation researchers have defined a need for online validation but recognize the challenges to the approach [29]. The development of techniques for online validation is an open research question in the DDDAS community [26].

1.3.5 PRESENTATIONS AND PUBLICATIONS

The WIPER system is intended to be a proof-of-concept of a system to aid Emergency Response professionals. As credence to its importance to that field, a description of the WIPER system has been presented at the premier conference for Emergency Response research: the Information Systems for Crisis Response and Management (ISCRAM) 2006 conference [91]. Additionally, a full length version of the system description, the work that comprises Chapter 3, is an invited paper to the International Journal of Intelligent Control and Systems [90]. Work from Chapter 4 is under review at the Journal of Defense Modeling and Simulation Special Issue on Modeling and Simulation in Homeland Security. Work from Chapters 6 and 5 is being revised and adapted for submission to appropriate simulation journals.

1.4 CONCLUSION

This Chapter outlines work towards completion of the Ph.D. degree in Computer Science and Engineering. This work addresses areas that the NSF considers important research directions: the ability to create and update running simulations with dynamic, streaming sensor input and the creation of a proof-of-concept DDDAS system. The research has been conducted in the context of the WIPER

system, an NSF-funded research project under the DDDAS initiative, grant award number CISE/CNS-DDDAS #0540348.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 INTRODUCTION

In this Chapter we present relevant background for the dissertation research. This research has all been conducted in the context of the WIPER project, a DDDAS system for emergency response and management. First, we introduce the Dynamic Data-Driven Application Systems (DDDAS) concept, which is a novel approach for tightly coupling sensors into the simulation process. There are many open research questions related to DDDAS, especially related to Agent-Based Modeling and online verification and validation. Then we introduce Agent-Based Modeling and Simulation, our approach to modeling the behavior of cell phone users in the WIPER project. Finally we introduce the field of verification and validation, the process of quantifying the applicability of a simulation. There are some challenges when attempting to apply accepted verification and validation techniques to Agent-Based Models. These issues are explained and prior approaches are presented.

2.2 THE WIPER SIMULATION: RESEARCH CONTEXT

The WIPER system is a software system that provides emergency responders and planners with information on unfolding crisis events [60]. WIPER monitors a

realtime stream of cell phone activity information, monitors the stream for potential crisis events, evaluates crises using Agent-Based Models that are created and updated using streaming data and finally presents all these results to emergency responders and planners through a web-based console. The WIPER system is an example of a Dynamic-Data Driven Application System, where simulations are tightly coupled to sensors. For more information about the WIPER system, see Chapter 3.

2.3 DDDAS

Dynamic Data-Driven Application Systems are systems that incorporate streaming data into running simulations and allow the simulations to influence the measurement process [26]. The applications for DDDAS systems include weather monitoring and prediction, fault monitoring, a variety of sensor network applications, biological and medical applications and homeland security/emergency response systems[19, 28]. The DDDAS approach seeks to put simulations into close contact with the data collection process, enabling simulations to be validated against streaming data and refining their predictive ability.

2.3.1 DDDAS GOALS AND CHALLENGES

In a traditional simulation, input parameters can be generated offline either by empirical data or via a researcher's educated guess. When using empirical data in this fashion there is always a manual step of processing the data so that it is appropriate to the simulation, adjusting the time or space of the data collection, removal of outliers, etc. However DDDAS systems are more time-critical and must be parameterized online from streaming data, without human interaction.

Moreover, simulations must be designed in such a way that they can be run in faster than real time and be updated with live, streaming data.

Darema lists the benefits and challenges posed by DDDAS systems in [27]. The two main goals of DDDAS stated succinctly are:

- The ability to incorporate additional data into an executing application.
- The ability of applications to dynamically steer the measurement process.

The thesis is that by incorporating streaming data into a running simulation the output from the simulation will yield "more accurate analysis, more accurate predictions and more precise controls". Similarly, by directing the data collection, "steering", simulations can reduce the overall number of measurements required by focusing on only a relevant subset of measurements. This may reduce the overall costs, the time required to collect measurements or improve the overall quality of the data collected.

2.3.2 DDDAS AND MODELING AND SIMULATION

There exist several other DDDAS research projects that use Agent-Based Modeling and Simulation. Among these are SAMAS [23] and the work of Michopoulos et al [68]. The SAMAS project is focused on multi-scale agent-based simulations in DDDAS systems, where the critical challenge is solving temporal multi-scale challenges. In [68], Michopoulos et al focus on using DDDAS concepts for updating the environment in an ABM, but no mention was made about updating the agents themselves, which is the primary challenge in the WIPER system.

In [1], the authors describe a DDDAS for "characterizing the three-dimensional geological structure and mechanical properties of individual sites and complete

basins in earthquake-prone regions." The system takes inputs from seismic sensors and uses this information along with simulations to infer the structure of a region.

In [62], the authors present a DDDAS for real-time modeling of the spread of wildfires. The simulations take streaming data from a variety of sensors, including weather (wind speed and direction, precipitation, etc) and geographic sensors (terrain, vegetation, moisture content) and feed the results into simulations.

2.4 AGENT BASED MODELING AND SIMULATION

Agent-Based Models are well suited to simulating the behavior of complex systems, such as ecological systems, biological simulations and simulations of human behavior and movement [11, 24, 82, 88, 92]. These simulations cover a vast range of application domains, but there are similarities in the systems that make them amenable to the ABM approach. First, the system is composed of heterogeneous actors, which are called agents when simulated. Next, the desired output of the system is an aggregate over all the agents in the system. This can be some measure of the state of the agents, such as location, emotional state, etc. Finally, the simulation output is generated as a result of the interactions of the agents with themselves and the environment.

Agent-Based Modeling techniques grew out of a merging of the Discrete Event Simulation and Cellular Automata paradigms [9, 73]. Cellular Automata are simulations that exist in an environmental space, such as a grid, and the cells have certain behaviors or properties that develop over time and in relation to neighboring cells [104]. A common example of a cellular automaton is Conway's Game of Life, where grid cells live or die depending on the state of neighboring cells [39].

According to [9], a Discrete Event Simulations is a simulation where the "state variables change only at a discrete set of points in time". Examples include event-based simulations that model processes, including many important to industry, such as scheduling the processing of parts in an assembly plant [20]. In these simulations events occur on a timeline, a schedule of events ordered by time. Simulations that use a timeline are considered "discrete" due to the discrete nature of time, as the simulation can produce a snapshot of system state at any given time during the execution of the schedule.

An Agent-Based Model is comprised of a set of actors, the agents, which can be heterogenous and have individual movements, behaviors and goals and an environment (often spatially explicit, as on a cellular automata grid) where the agents exists. The behavior of the simulation is a result of the interactions among agents and of the agents with their environment.

Agent-Based Modeling (ABM) is an approach to simulating social, ecological and biological systems where the system-level behavior is an emergent property of the interaction of many independent individuals (agents) with each other and the environment. ABM is an approach often used to model complex systems [48]. The approach is well suited to social systems where the agents may display complex behaviors and interactions. Agent-Based Models are often contrasted with equation-based approaches for modeling systems, such as coupled differential equations. In an equation-based model, the output from the equations is the system-level property. Consider the following example taken from ecology: The populations of predator and prey species can be modeled very well using equation-based models (such as the Lotka-Volterra equation[113]). The population levels can also be simulated in an Agent-Based Model by creating populations

of predator and prey agents, giving them appropriate behaviors, e.g. the prey consume resources from the environment and the predators consume the prey, and allowing them to interact. (For an example of the predator-prey simulation in an Agent-Based Model, see [119].) A case study comparing Agent-Based and Equation-Based models can be found in [106].

Agent-Based Modeling is characterized by creating an encapsulated unit, an agent, defining behaviors and interactions for the agents and placing them in an environment. An agent is a self-contained entity with basic actions and decision making, including interaction with an environment and the other agents around it.

The term agent is used in many different contexts to mean many different things. Unlike Artificial Intelligence agents, ABM agents tend to be less sophisticated in their ability to react and influence the surrounding environment and normally do not encompass behavior such as planning, learning or reasoning [61]. Mobile agents are self-contained executing bundles of code that can move autonomously around a network[110]. Multi-Agent Systems are composed of software agents that can operate independently, are distributed and communicate over a network [34, 121].

2.4.1 THE MODELING PROCESS

2.4.2 EVALUATION OF THE AGENT-BASED MODELING APPROACH

Developing simulations intended to describe complex systems is a challenging task. One advantage of Agent-Based Models over equation-based models is that constructing the simulation is often more intuitive for the modeler. A comparative study of Agent-Based Models and System Dynamics models is given in [106]. In

that paper, Wakeland give a case study where the same system was modeled with an Agent-Based Modeling tool (StarLogo[96]) and with a Systems Dynamics tool (STELLA[97]). The study demonstrated that the Agent-Based approach was attractive because it more closely resembled a traditional (physical) experiment.

2.4.2.1 ADVANTAGES OF THE AGENT-BASED MODELING APPROACH

In [8], Bankes argues that Agent-Based Modeling is a revolutionary tool for advancing social science for the following reasons:

- Alternative modeling formalisms are not able to address problems of social science
- Agents form a "natural ontology" for social problems
- ABM can display emergence

In the context of the WIPER project, our agents will be a person carrying a cell phone. WIPER agents move around on a simulated GIS landscape, make calls based on sampling from an empirical call distribution and are designed to react to various crisis scenarios.

2.4.2.2 CHALLENGES IN THE DEVELOPMENT OF AGENT-BASED MODELS

One of the challenges when designing Agent-Based Models is deciding how much complexity, in terms of agent behaviors and observed characteristics of agents or environment, to include in the model. Modelers must resist the desire to attempt to accurately represent every aspect of the system. This approach is often referred to as "naive realism" [44] and is counterproductive to the modeling

process. An approach to provide some guidance in the development of Agent-Based Models is the Pattern-Oriented Modeling (POM) concept[45]. POM is a model development heuristic where a modeler first seeks to identify motifs (patterns) that are observed in the real world, such as foraging behavior under certain conditions, response to stresses, seasonal patterns of movement, etc. The model is then constructed by the composition of these patterns. Patterns are added incrementally until the model displays the desired system-level properties.

2.4.3 AGENT-BASED MODELING FOR EMERGENCY RESPONSE AND PLANNING

The emergency response community has recently become aware of the applicability of Agent-Based Modeling for the simulation of disaster scenarios. Using an ABM, it is possible to evaluate policies and procedures for dealing with natural and man-made disasters.

Takahashi et al describe an Agent-Based Simulation for disaster rescue in [98]. The system, now in use as the platform for the annual RoboCup Rescue competition, uses a GIS space and information from the 1995 Kobe, Japan earthquake to create a realistic disaster. Teams compete by creating fire, rescue and police agents that attempt to subdue fires, control crowds and move civilians to safety.

2.4.4 CROWD MODELING

Agent-Based Models are well suited to representing crowd/pedestrian models. Pedestrians are autonomous, goal-directed agents whose spatial properties (slow movement, relatively constrained behavior) are amenable to representation in an ABM. The WIPER simulation is an Agent-Based pedestrian movement model.

Simulating the movements of individuals and groups is vitally important to the usefulness of the WIPER simulation.

In [71] a method for modeling crowds is presented. This approach is designed to model crowds in a visually convincing manner. The intended application is for visual simulations and video games and thus may not be able to provide useful data for emergency responders and planners.

Reynolds provides a basic movement model for flocking bird in [86]. The advantages of this approach are that the decision making process for the group is distributed, there is no leader agent that directs the group, and that group movement is an emergent phenomenon. This approach to modeling movement may be useful for the WIPER project, but the approach will need to be validated against empirical movement studies and adapted to take into account the complexity of human intentions.

2.4.5 TRAFFIC MODELING

Several Agent-Based Models of traffic flow currently exist [11] [47]. These models are useful in that they can produce useful information over a variety of traffic conditions, from traffic jams to steady state traffic flow, which is a limitation of earlier approaches [42, 43].

2.4.6 AGENT-BASED MODELING AND GIS

In GIS Data Sources, Decker covers GIS data types, sources, common issues with data sets and attendant solutions [30]. GIS is a powerful paradigm for representing geographic data because it allows users to store and access multiple layers of data on top of one another. These layers can contain information on roads,

structures, elevation, property lines, aerial photos, etc. GIS layers can come in either raster or vector format and usually a presentation of a data set contains a raster data layer on the bottom with one or more vector data layers above it. One important data layer is a digital elevation model (DEM). This is a 2D raster that stores elevation information at every *(x,y)* coordinate. Vector data represents important features with lines, curves or areas. Several important types of vector data are Digital Line Graphs (DLG), where each file represents one feature, such as transportation, elevation or water features, and Digital Chart of the World (DCW), which is an older format that provides data on the entire globe. GIS files contain metadata that describe the contents of the file. These attributes may include scale, author/creator, time (usually when file was created), subject/theme (highways, census, hydrology, etc) and Projection. Decker provides a large listing of GIS data repositories, but these seem to be quite outdated [30].

GIS, Environmental Modelling and Engineering [18] deals with the use of GIS data in environmental simulations. In order for GIS layers to be stacked or superimposed upon one another, "they must all conform to the same coordinate system and map projection". GIS data can be stored as layers/records in a relational database system. The data can then be referenced by coordinate and layers constructed from the database as needed. Issues of data quality are address, as well as specific issues to the use of GIS in simulations as well as several Case Studies.

Batty and Jiang discuss the idea of placing simulations on top of GIS spaces [14]. The authors give several examples, such as the design of pedestrian friendly spaces and land use simulations.

Hare and Deadman survey the field of Agent-Based Modeling specifically related to environmental modeling [46]. The authors collect and attempt to

classify the disparate terms surrounding Agent-Based Modeling and Simulation: Individual-Based Modeling, Agent-Based Modeling, Agent-Based Simulation Modeling, Multi-Agent Simulation, Multi-Agent-Based Simulation, Agent-Based Social Simulation and Individual-Based Configuration Modeling.

Itami and Gimblett present two examples of agent-based simulations that are built on top of GIS information[50]. (In their paper, Itami and Gimblett refer to the simulations as Multi-Agent Simulations.) The two examples are the Grand Canyon River Trip Simulator (GCRTS) to simulate rafting trips at Grand Canyon National Park and the Recreation Behavior Simulation (RMSim 2), a general simulation package for building GIS-based recreation simulations. The simulations are intended to allow recreation managers to determine usage for different areas of parks and recreation locations. The agents are goal-directed (with goals such as exercise, appreciation of the local ecology, social interaction, etc) and use fuzzy reasoning to execute plans to achieve these goals. The simulation uses transportation networks (footpaths, roads), elevation information, vegetation/ecology and facilities to represent the park or recreation area. The agents move about within the park, seeking to achieve their goals. The simulation outputs a time-ordered visualization of agents' movements, as well as plots of agent population size at various locations of interest over time.

2.5 AGENT-BASED MODELING TOOLKITS

The standard practice when developing Agent-Based Models is to build a model with an existing toolkit. Tools such as RePast [74], Swarm [69] and NetLogo [119] provide modelers with APIs to reduce the amount of programming required when creating and developing models. Each toolkit provides access to a

pseudo-random number generator, spaces for agents to inhabit (commonly used spaces include 2D grids and network spaces) and controls for starting, stopping and probing the simulation. Toolkits may represent time as time steps, integer units of time, or may implement a timeline, a discrete time structure where events can be scheduled at times down to a given resolution (such as a `double`).

Tobias and Hofmann present a study of four popular agent-based modeling toolkits used for social science simulationas in [102]. The authors examine RePast [74], Swarm [69], Quicksilver [21] and VSEit [17]. There is a proliferation of agent-based simulation tools, so the authors began by selecting the four packages by dropping from consideration any package which did not meet the following criteria:

- The simulation tool must allow models to be built on social science theories and observation. An agent in the tool should be able to represent a real human being or an institution, as opposed to an abstract or aggregate representation.
- The simulation tool must be freely available.
- The simulation tool must be able to run simulations created in Java. (The authors use the phrase "implemented in Java", but this must apply to the simulation and not the base toolkit, as Swarm uses Objective C in its runtime.)

The simulation tools were scored on a scale of 1 to 6 on the several questions in the following areas:

- General features (Licensing, support, documentation, etc)
- Modeling and Experimentation (Ease of use, programming support)

- Modeling Options (Size of simulations, representation ability, creating and managing agents)

The ratings were based on reading available documentation, user feedback and experiences with developing simulations and using demos from the tool. The study concluded that RePast and Swarm were the best suited toolkits for Social Science modeling, due to their stable user communities, good documentation and rich feature sets. The WIPER simulation has been developed using RePast due to its comprehensive feature set and ease of use.

2.6 OPTIMIZATION VIA SIMULATION

Our ensemble-based approach to updating simulations with streaming sensor data can be compared to existing techniques called "optimization via simulation" from the Discrete Event Simulation field. These approaches are related to canonical optimization techniques where the objective function is a simulation and the input space is over the parameters to the simulation.

In [3], the author presents techniques for optimizing the performance of simulations. Performance is defined as the effectiveness of the simulated system at its intended task, such as the production of widgets in a factory simulation. Approaches are put into two categories based on whether the simulation input parameters are discrete or continuous. For simulations with continuous input parameters, the author suggests the use of gradient-based methods. For simulations with discrete input parameters, the author presents approaches using random search on the input space.

Pichitlamken and Nelson describe a combined procedure for Optimization via Simulation in [79]. The authors approach has three components: "a global guid-

ance system, a selection-of-the-best procedure and local improvement". This approach uses the Nested Partition method of Shi and Olafsson [93] for global guidance and a hill climbing approach for local improvement.

2.7 ONLINE MODELING AND SIMULATION

In [29], Davis examines the state of discrete event simulators and the research directions (circa 1998) and concludes that research was moving towards the study of online simulations, but that many challenges existed. One challenge is the initialization of simulations based on a measured system state. At that time, Discrete Event Simulations were designed to simulate the steady-state behavior of systems, and as such were not properly designed to be initialized from real-world data. Davis also defines a need for online validation but presents the challenges to the approach. The author claims that online validation may be unobtainable due to the difficulty in implementing changes to a model, as opposed to varying input parameters, in an online scenario.

2.8 VERIFICATION AND VALIDATION

Verification is the process of assessing whether the conceptual model is correctly translated into the computer program and validation is the process of comparing the program's behavior with observations from the real world [9]. The verification and validation process is an important part of the development of a simulation and is essential for the acceptance of a model and important in determining how much trust to assign to a model's predictions.

2.8.1 METHODS FOR VERIFICATION AND VALIDATION

There exist numerous approaches for verification and validation. For example, Balci lists 75 commonly used techniques for verification and validation [7]. Balci presents a taxonomy of the approaches, dividing them into 4 categories: Informal, Static, Dynamic and Formal. For our purposes, we consider a classification posed by Kennedy [54] [122] for verification and validation techniques for Agent-Based Models. Kennedy divides the approaches into two types: Subjective methods and Quantitative methods. Subjective methods often rely on the judgement of experts or require human interpretation in order to be useful. These methods are commonly used during the initial development of simulations. Quantitative methods use statistical measures to compare simulation outputs to observed data or to output from other simulations. These methods are often used when calibrating a simulation or in the final stages of simulation development.

2.8.1.1 SUBJECTIVE METHODS

Subjective methods are often used during the initial phases of model development. These techniques require a human to judge the results. Some examples of subjective methods are listed below. (Adapted from [54].)

- Black-box Testing - This technique involves feeding a set of inputs to a model and judging whether the output is reasonable.
- Face Validation - A subject matter expert inspects the conceptual model and determines whether the model is based on valid assumptions and is grounded on accepted theory.
- Internal Validity - In Agent-Based Models there is often a stochastic ele-

ment. The internal validity test involves running the model repeatedly with different random seeds.

- Turing Test - The Turing test in this context refers to showing the model output to a subject matter expert. The test refers to whether the subject matter expert can determine whether the output comes from empirical observation or the simulation.

All of these approaches are useful during the development of Agent-Based Models. In Chapter 4 we apply several subjective methods of simulation validation and verification to the WIPER simulation.

Although these techniques are useful in the model development stage, all require human interpretation. For that reason none of these techniques can be used for online validation of the WIPER simulations.

2.8.1.2 QUANTITATIVE METHODS

The quantitative methods for validation and verification of Agent-Based Models consist of statistical tests such as the Chi-square test and Kolmogorov-Smirnov test, as well as other approaches that quantify the applicability of the model, such as sensitivity analysis and docking. Examples of quantitative methods are listed below. (Adapted from [54].)

- Statistical tests, such as Chi-square or K-S test, are used to determine whether model output is sufficiently similar to empirical data.

- Predictive validation - Observed data is used to parameterize the model and the model's predictions about future outcomes are compared to empirical data.

- Sensitivity analysis - Model parameters are adjusted incrementally over a range and the results are observed. Inconsistent model behavior may indicate an incorrect model.

- Docking - If two models of the same system exist, with different conceptual models, they can be used to validate each other. Both models are fed the same set of inputs and the outputs are compared.

For dynamic validation in DDDAS systems such as WIPER we focus on methods that are automated and yield unambiguous results. We present a comprehensive evaluation of several quantitative approaches in Chapter 6.

2.8.2 APPLICATION OF EXISTING VERIFICATION AND VALIDATION APPROACHES TO AGENT-BASED MODELING AND SIMULATION

Xiang [122] and Kennedy [54] both address the need for developing verification and validation techniques tailored to Agent-Based Models. The results are preliminary and the authors suggest that the best approaches involve the subjective and quantitate methods described above. Often when researchers develop ABMs they apply face validation to the model, but do not apply any other, more rigorous methods. Thus the application of even a small amount of additional validation techniques are beneficial to the ABM community.

2.9 SUMMARY AND CONCLUSION

This Chapter has presented a survey of related work and background necessary for the understanding of the Dissertation research. Dynamic, Data-Driven Application Systems is an emerging research area and an important application

domain for the modeling community. Creating Agent-Based Modeling simulations to be used in the context of DDDAS will require a more agile approach than the standard modeling practice. In this Dissertation we build upon this background work and place our research in the context of DDDAS, specifically the WIPER system, a DDDAS for Emergency Response and Crisis Management.

CHAPTER 3

WIPER: LEVERAGING THE CELL PHONE NETWORK FOR EMERGENCY RESPONSE

3.1 ABSTRACT

This chapter describes the Wireless Phone-based Emergency Response (WIPER) system. WIPER is designed to provide emergency planners and responders with an integrated system that will help to detect crisis events, as well as to suggest and evaluate possible courses of action to deal with the emergency. The system is designed as a distributed system using web services and the service oriented architecture. WIPER is designed to evaluate potential plans of action using a series of GIS-enabled Agent-Based simulations that are grounded on realtime data from cell phone network providers. The system relies on the DDDAS concept [26], the interactive use of partial aggregate and detailed realtime data to continuously update the system, which ensures that simulations always generate timely and pertinent data. WIPER presents information to users through a web-based interface of several overlaid layers of information, allowing users rich detail and flexibility. [1]

[1]Preliminary versions of this research were published in the Proceedings of ISCRAM 2006 [91] and in the International Journal of Intelligent Control and Systems [90] [2]

3.2 INTRODUCTION

Emergency responders often first learn of crisis events through eyewitness accounts from civilians calling 911. Although this information is timely, bystanders lack the perspective to convey the wider scope of a crisis and may not be able to provide reliable, actionable data. The WIPER can compliment first-person accounts by offering emergency responders a holistic view of the crisis area in terms of overall human activity, as sensed through the cell phone network and presented on top of relevant satellite and GIS representations of the area. In addition to presenting the current state of the crisis area, WIPER can employ agent-based simulations for short-term prediction and the evaluation of response strategies.

Numerous software tools have been developed to aid emergency responders. Several recent examples are EVResponse and the COMBINED project [99, 101]. These tools provide methods of gathering information on the current status of crisis situations. They provide emergency response planners with detailed, high-quality information, but require a high cost in terms of personnel and deployment. (PDAs and wireless infrastructure must be purchased, personnel trained and both need to be sent to crisis sites.) WIPER would act as a low-cost, highly available monitoring system. Its deployment would be automatic, as anyone with a cell phone in the area is a participant. No special training would be required for phone users, but balancing this, the quality of information from each person is low. Limited to location and activity information, it may not be clear what type of crisis is occurring. We use machine learning techniques to infer information about the state of the area (i.e., to distinguish a fire from a traffic jam) from the location and call activity information that we collect. WIPER would convey three distinct and useful pieces of information to emergency responders via the

web-based console:

- It provides near-real time information on the location of cell phone users in an area, plotted on a GIS-based map of the area.
- It detects potential anomalies, such as traffic jams, roving crowds and call patterns indicative of a crisis.
- It can evaluate mitigation strategies, such as potential evacuation routes or barricade placement, through the use of computer simulations.

The WIPER system is designed to address specific needs in the Emergency Response community, specifically the ability to view the development of a crisis in realtime, the ability to propose and evaluate response in near-real time and the ability to collect and analyze streaming information from a cell phone-based sensor network. This capability positions WIPER as an important component is an overall emergency response workflow. The WIPER system uses dynamic data from cellphones and analyzes the data in realtime, providing the ability to detect a crisis as it emerges. An online classification system is designed to identify crises by recognizing familiar patterns in group behavior. Responding to events from the anomaly detection system, GIS-based simulations of the region are launched and results collated and made available to the console. Finally, the web-based console allows Emergency Planners to quickly examine the current state of the environment, see possible predicted outcomes from the simulations and evaluate courses of action.

WIPER is designed to work with the current level of data available from the cell phone network (activity and rudimentary location data). The system utilizes dynamic streaming information from cell phone providers to monitor and detect

anomalies and crisis events. A more thorough discussion of the cell phone data is presented in Section 3.5.1. The most common type of potential crisis events would be traffic disturbances, but by utilizing historical knowledge of crisis events and call patterns and realtime social network algorithms, WIPER should be able to predict, detect and evaluate responses to a wide range of emergency situations. WIPER could detect crowds and demonstrators at public events, monitoring such events to determine if they are degenerating into riots.

3.3 BACKGROUND

In this section we describe relevant background to the WIPER project and related work in the Emergency Management field.

3.3.1 AGENT-BASED MODELING AND SIMULATION

Agent-Based Modeling and Simulation is a modeling paradigm that is well established for studying complex systems with emergent behavior. Examples of this type of system are biological, physical and social systems where the principal actors (agents), their surrounding environment and the modes of interaction form the basis for the emergent behavior. Agent-Based Simulations are closely related to Cellular Automata, which are often used in modeling spatial phenomena, such as traffic flow[107]. An example of the application of Agent-Based Modelling and Simulation to the area of crisis response are the TranSims and EpiSims projects [12, 72]. The TranSims project was created to accurately model the transportation system of an entire city, including personal automobiles, pedestrians, public transportation and commercial vehicles. The system is used to provide city planners with a way of accurately gauging the impact of infrastructure changes on

a city's transportation system. The EpiSims system was an outgrowth of TranSims and is able to model the transmission of infectious diseases through a city. EpiSims makes it possible to empirically evaluate methods of inhibiting the spread of biological warfare agents in an urban setting.

3.3.2 EMERGENCY MANAGEMENT

The use of Information Systems in the Emergency Management field is well established [57, 99, 101]. If designed and implemented properly, Information Systems can enable Emergency Management professionals to deal with increasingly complex crisis scenarios and coordinate effective inter-organizational response [103]. However, in order to be useful certain design considerations must be met[22].

In [51], the authors introduce the concept of a generic information processor (GIP), which is an information system that publishes content related to a crisis event for a variety of uses (incident report, emergency planning, training, etc). The content published by the GIP can be self-generated and/or a product of processed information from other GIPs. Of paramount importance in the sharing of information between GIPs is the use of well-defined document types and data formats. In order to make the WIPER system compatible with the GIP concept, information from the WIPER system is published in an XML format and is accessible via web services.

Much work has been done exploring the use of cellular phones for emergency communication, especially related to large scale targeted warnings [52, 120]. The cellular network provides a unique capability to infer the position of people in an affected area and to provide them with specific and relevant instructions. The WIPER system is designed to be complementary to these approaches, providing

information on the location and activity of cell phone handsets. Although not designed to directly interact with a system like CellAlert, WIPER can provide information to an emergency planner that would inform the use of the CellAlert or a similar system.

3.3.3 GIS ENABLED SIMULATIONS

Geographic Information Systems can be used to provide added realism in Agent-Based Simulations [41]. Agents can interact with terrain and roads representative of the real world, enhancing the credibility of such simulations. GIS systems have been successfully integrated with simulations in scenarios where an explicit spatial representation is important to the validity of the simulation[14, 53]

3.3.4 REAL-TIME SENSING IN URBAN ENVIRONMENTS

Several systems already use cell phone activity and location information to sense population density in urban settings. The most important project is MIT's SENSEable City[83]. The aim of the SENSEable City project is to allow city officials, urban planners and people at large the ability to follow the trends in population movement and activity around the city. Initially the project mapped the real time activity in the city of Graz, Austria, but now it has been expanded to cover Rome, Italy as well [84].

3.3.5 DDDAS

Recently the National Science Foundation has created a program to spur the development of Dynamic-Data Driven Application Systems[95]. A DDDAS is a software system that tightly couples simulations with sensors and data collection

devices, a process that enables simulations to more quickly adapt to changing data and even control the collection of data[26, 28]. One aspect of DDDAS is the dynamic injection of data into the system, as shown in Figure 3.1.

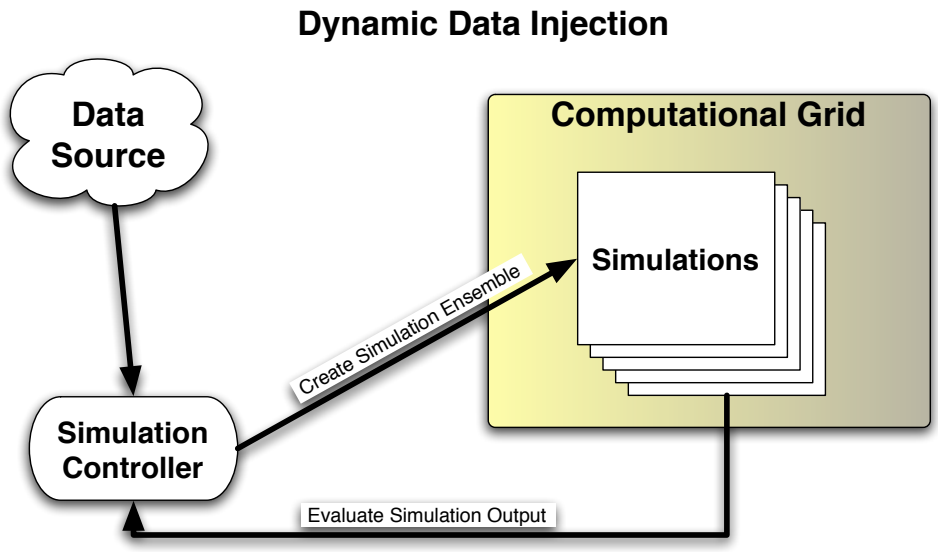

Figure 3.1. A fundamental concept of DDDAS systems: Integrating simulations with the sensors. Here we see that simulations receive a stream of real-time sensor information.

The DDDAS approach has been implemented in narrowly-focused crisis management platforms, such as weather monitoring [19] and fire monitoring [68] applications. These examples demonstrate how the DDDAS approach is beneficial in crisis scenarios, as simulations are constantly being updated and refined based on streams of incoming data.

3.4 WIPER SYSTEM OVERVIEW

As proposed in several previous projects, the existing cell phone network can be used both as a tool for detecting the state of the environment [4, 83] as well as communicating directly with those affected by crisis events [25, 120, 125]. WIPER is intended to push the boundary of crisis detection and monitoring with the current cell phone network. A visual description of the WIPER scenario is presented in Figure 3.2. The WIPER system will receive a feed of realtime information from cell phone providers. This is expected to be a sample of the incoming data, as the full data stream would be prohibitively difficult to transmit. The incoming data would be monitored for anomalies, which include the obvious spatial and temporal aggregation, as well as call patterns and movement discrepancies that can signal the impending onset of a crisis event.

Figure 3.3 shows the overall system architecture of WIPER. The WIPER system is a distributed system combining traditional methods of network communication (UDP sockets over IP) with with implementation independent and robust composition protocols (Web Services and Service Oriented Architecture). WIPER is composed of three layers:

- Data Source and Measurement
- Detection, Simulation and Prediction
- Decision Support

The Data Source and Measurement layer handles the acquisition of realtime cell phone data, as well as deterministic transformations on the data, such as aggregation of the data or the calculation of triangulation information. The Detection, Simulation and Prediction layer analyzes incoming data for anomalies, attempts

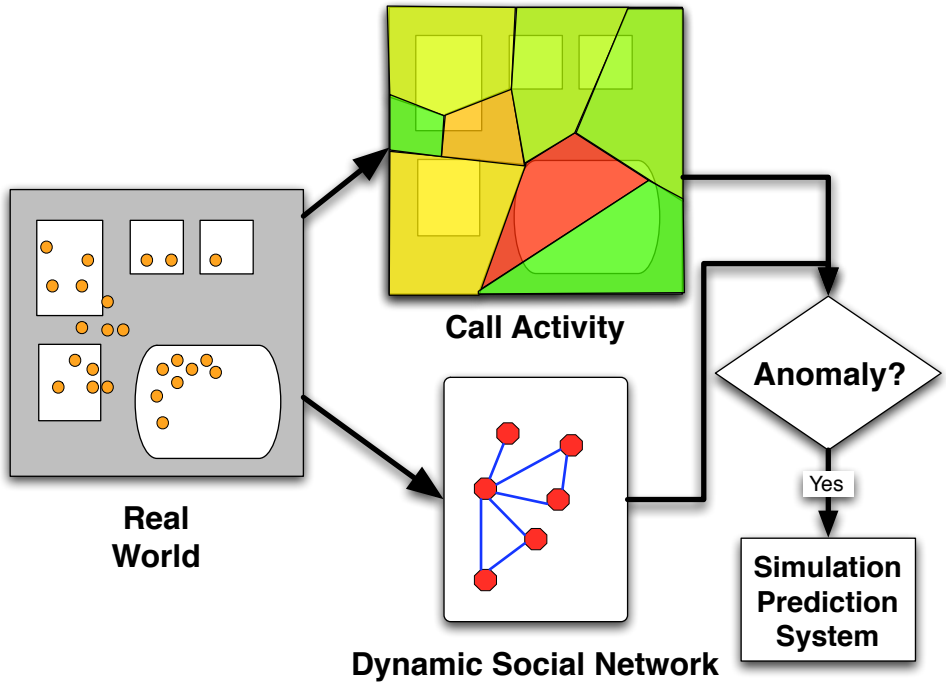

Figure 3.2. A visual representation of the WIPER scenario. As real world data streams into the system, we examine call activity by location and social network of the users to detect potential anomalies. In the image, orange circles represent cell phone users.

to simulate the anomaly to predict possible outcomes and suggests actions to mitigate the event. Finally, the Decision Support layer presents the information from the other layers to end-users, in terms of summaries of traffic information for commuters, real time maps and simulations on the anomaly to first responders and potential plans for crisis planners.

These layers are further divided into components that handle highly specific functions, as described in the following sections.

3.4.1 DATA SOURCE AND MEASUREMENT LAYER

This layer contains three modules, all of which have functionality related to the management of the real time cell phone data. The Real Time Data Source (RTDS) collects information from one cell phone provider, performs filtering and aggregation as necessary and redirects the data stream into components in the Detection, Simulation and Prediction (DSP) layer. The RTDS is composed of several mobile software agents that are dispatched to the cell phone provider. The software agent removes personalized information such as phone number and customer id and replaces it with a coded value that is internally consistent within the WIPER system but cannot be used to identify the user. For training purposes, snapshots of the data are occasionally stored on a server and become part of the Historical Data (HIS) module. The HIS streams historical data in the same format as the RTDS for training and testing the Detection and Simulation modules in the DSP layer. A Triangulation Information module (not pictured) handles converting the rough location information associated with a cell phone into a more precise location which is needed by the Simulation and Prediction System. On newer handsets, GPS sensors can provide the cell phone provider with precise location

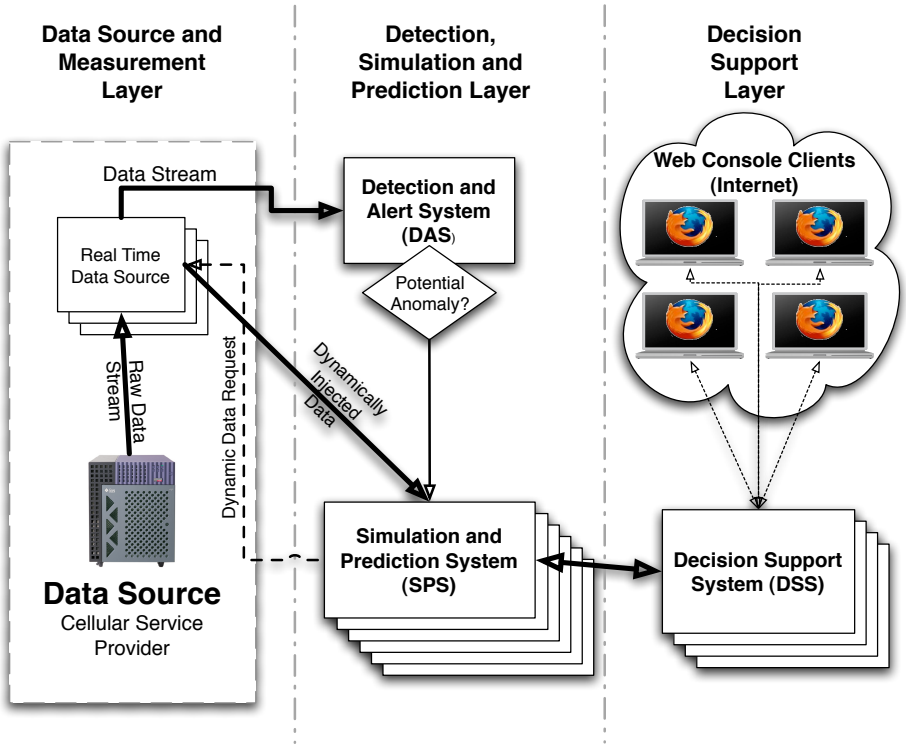

Figure 3.3. An overview of the layered structure of the WIPER system. The Data Source and Measurement layer physically resides on the cellular service providers network and handles collection, storage and preprocessing of the data. The Detection, Simulation and Prediction layer services reside on the WIPER network (e.g., at the emergency management data center) and process the streaming data to detect anomalies and run simulations for prediction and mitigation. The Decision Support layer services run on the WIPER network but access is provided to end-users across the internet through a web-based console.

information, but only if the feature is enabled and the cell phone provider is equipped to monitor it.

3.4.2 DETECTION, SIMULATION AND PREDICTION LAYER

The Detection, Simulation and Prediction (DSP) layer contains modules that monitor the streaming data, and generates computer simulations to determine whether perceived anomalies represent potential crisis events and what actions can be taken to mitigate these events. The Detection and Alert System (DAS) uses a combination of established techniques for detecting anomalous patterns of spatial activity, such as Statistical Process Control [78] and Markov-Modulated Poisson Processes [123].Upon detection of a potential anomaly, the DAS sends a message about the event to the Simulation and Prediction System (SPS). The SPS uses the information to create a GIS-based computer simulation that will attempt to model the outcome of the event. The SPS creates an ensemble of Agent-Based simulations that are run on a computational grid. The simulations are monitored by the SPS and ranked according to their ability to correctly predict the progression of the actual event. The SPS and each of the simulations interact with the RTDS to acquire more detailed information concerning the potential anomaly area. For more information on the SPS see Chapter 5. For a thorough description of the Agent-Based Simulations, see Chapter 4.

3.4.3 DECISION SUPPORT SYSTEM LAYER

The Decision Support System (DSS) acts as a front end for the WIPER system. It is the main portal for disseminating the information from WIPER to crisis planners and responders, public safety personal and the general public. A picture

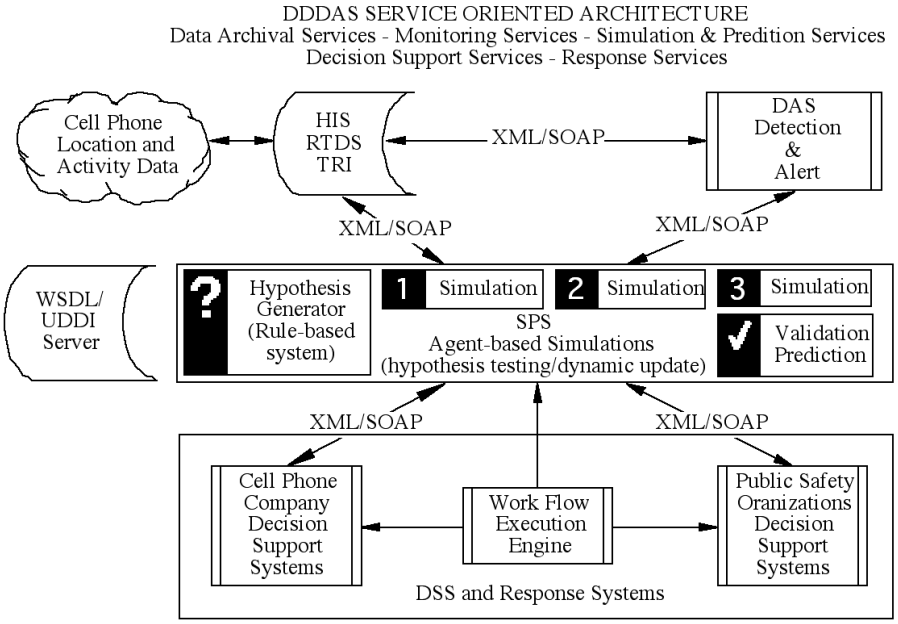

Figure 3.4. The Service Oriented Architecture of the WIPER system. The use of SOA allows WIPER components to expose their services to clients outside the WIPER system, allowing flexible composition of WIPER services into a heterogeneous emergency response workflow.

of the web-based console is shown in Figure 3.5. The DSS aggregates information from the SPS and presents the real time system status and any predicted anomaly information in a web based interface. There are options for crisis planners to specify and evaluate mitigation plans through the web interface. These plans will subsequently be evaluated with Agent-Based simulations and the results accessible from the web based interface. Crisis planners can monitor crisis areas using satellite maps and GIS images overlaid with activity data, as well as viewing the raw data entering the system and comparisons against normal activity and historical data trends.

Given the huge amount of raw data and processed information in various formats, users of the DSS will want to reduce the overall complexity of the system to address their specific needs. The DSS has been designed and implemented with that flexibility in mind, using Web Services and AJAX to implement the specific components. Users can customize the view using standards compliant web browsers, selecting which services they wish to see, adding tabbed views for different services and saving configurations for later use.

This web interface uses SSL encryption and authentication to prevent snooping and restrict access to authorized users. The DSS may also be configured to allow certain information to be publicly accessible, such as providing a near-real time picture of the traffic situation or predictions of traffic congestion.

3.4.4 TECHNOLOGIES

3.4.4.1 WEB SERVICES

The use of Web Services and the Service Oriented Architecture allows WIPER to be composed of standards-compliant modules running on heterogeneous hard-

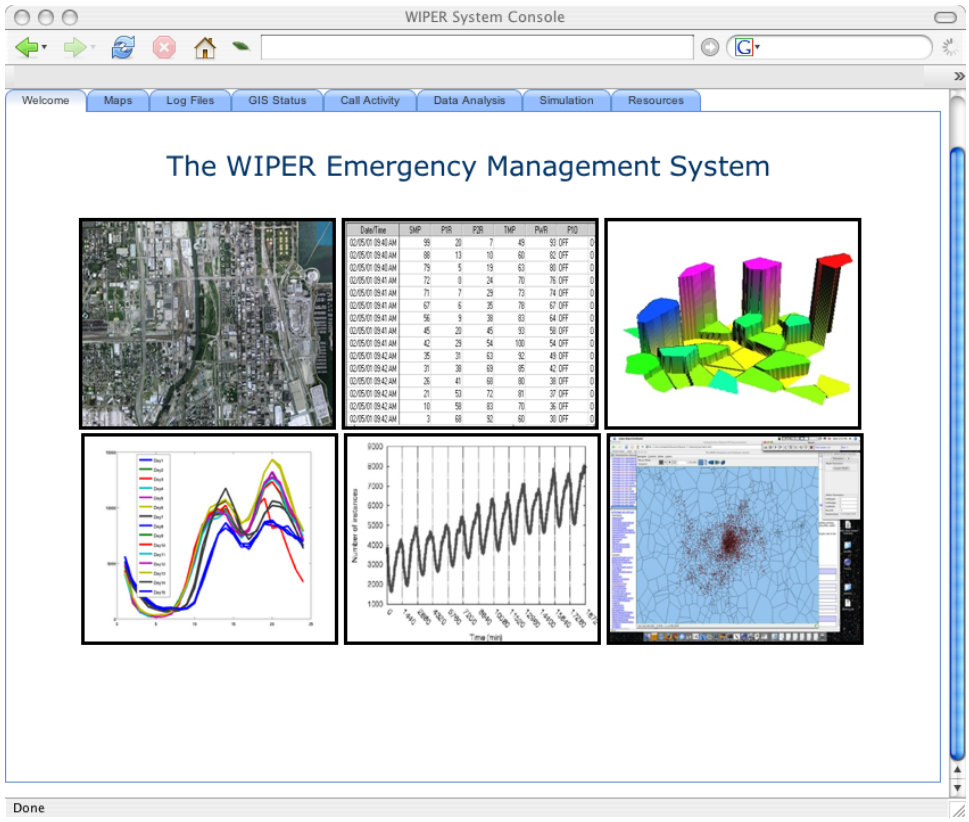

Figure 3.5. The WIPER DSS web-based console. The console provides easy, standards-compliant access to all of the components of the WIPER system, allowing emergency planners access to the real time data, both overall activity and spatially aggregated, simulation output and information on system status. The components of the system seen here are (clockwise, beginning in the upper left corner): satellite map of the affected area, raw data from cellular service provider, 3D activity intensity map, 2D plot of city-scale network activity, historical trend of activity and 2D visualization of the city simulation.

ware and operating systems and simplifies the integration of the system. The SOA for the WIPER system is demonstrated in Figure 3.4. The flexibility and ease of composition that come with Web Services allows the WIPER system to easily incorporate new data sources, such as weather and news feeds. This ease of composition also makes it possible to use WIPER components as data sources for other applications in an Emergency Response workflow.

3.4.4.2 MAPPING AND VISUALIZATION

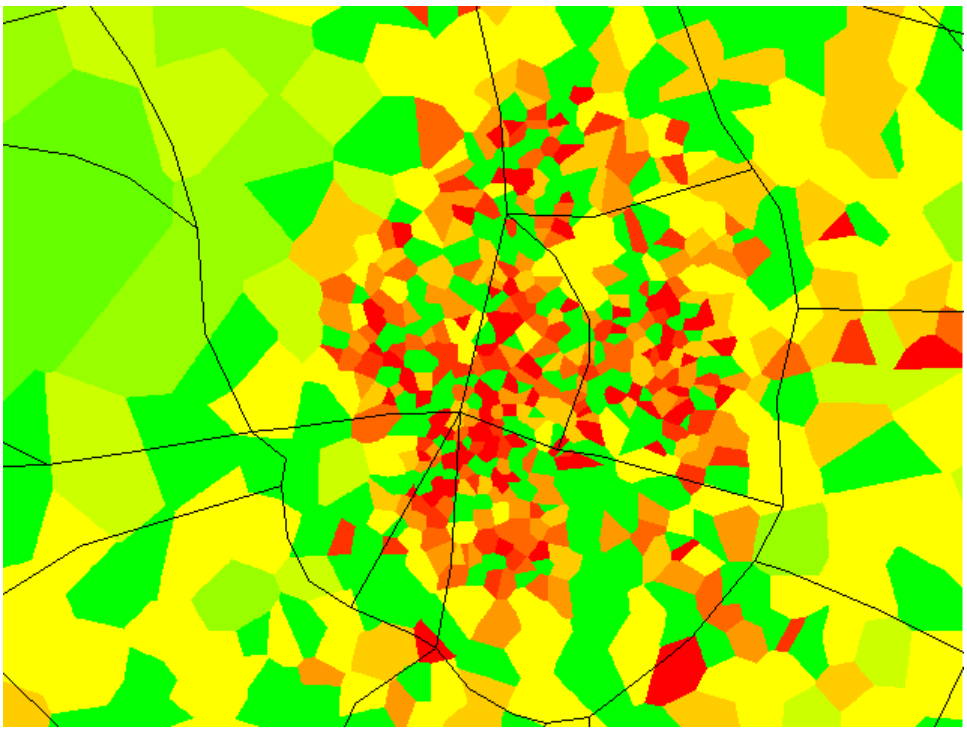

Figure 3.6. A 2D view of activity in the cellular network. Each polygon represents the spatial area serviced by one tower. The cells are colored green (low activity) to red (high activity) based on the amount of active cell phone users in that cell.

Accurate, informative visualizations are crucial to the WIPER system. A properly designed visualization system can present geographic information more clearly and coherently than a textual description. In the WIPER system, we present location data from the cell phone provider, representing a recent snapshot of the activity and location of individuals in the affected area, as well as GIS-based simulations which can be used to provide various scenarios about the development and outcome of certain crisis events.

Our data source currently provides us with data on user locations and activity at a cell-sized level of resolution. The size of a cell can vary widely and depends on many factors, but these can be generalized in a simple way using a Voronoi diagram [114] (also called Thiessen polygons). A Voronoi lattice is a tiling of polygons in the plane constructed in the following manner: Given a set of points P (in our case, a set of towers) construct a polygon around each point in P such that for all points in the polygon around p_0, the point is closer to p_0 than to any other point in P. Thus we can construct a tiling of a GIS space into cells around our towers, as shown with activity in Figure 3.6.

We currently have two methods for visualizing the location data. The first method is to color the Voronoi cells in the area of interest based on the level of activity. This method is demonstrated in Figure 3.6. In this image the color scale ranges from green (low activity) to red (high activity). Alternately, we can build a 3D image based on the activity at the site of interest, as shown in Figure 3.7. This 3D view gives a better conceptual picture of the comparative activity levels in the cells. However, viewing the activity in this manner may not enable Emergency Response planners to evaluate the current activity levels or compare them to historic activity information. We are currently considering other methods

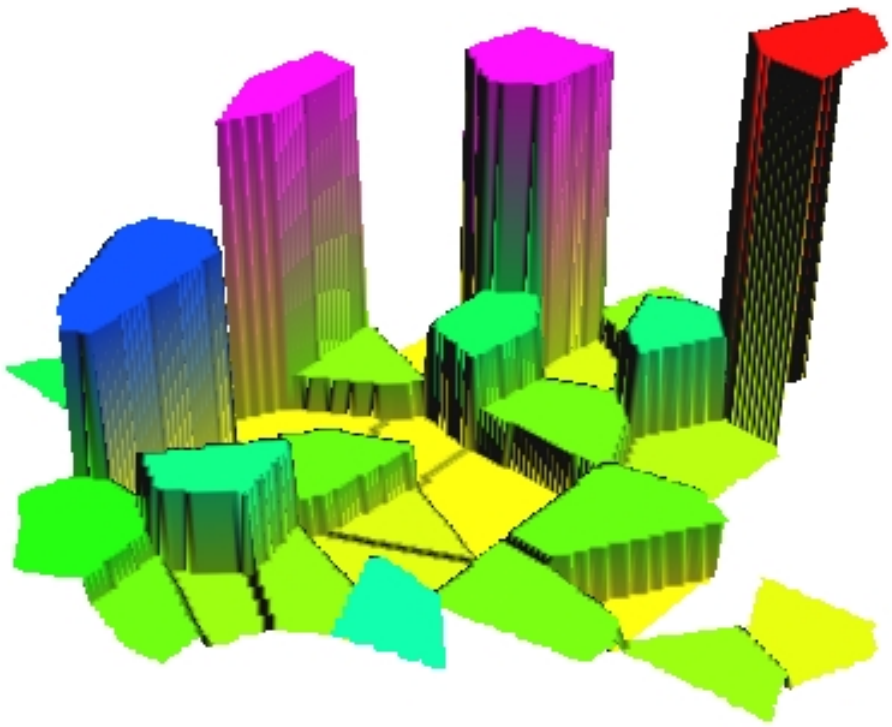

Figure 3.7. A 3D view of activity in the cell system. Each polygon represents the spatial area serviced by one tower. Cell color and height are proportional to the amount of active cell phone users in that cell.

of attenuating cell heights to account for the varying size of the cells, as shown in Figure 3.8, or normalizing the cell activities to historical values for this area at similar times.

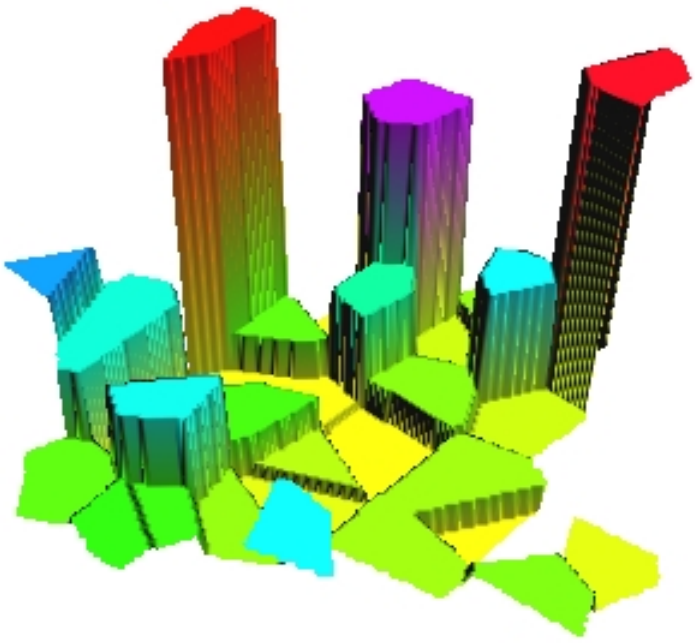

Figure 3.8. A transformed view of the activity over an urban area. The activity values are normalized by the area of the cell.

3.5 IMPLEMENTATION DETAILS

3.5.1 CELL PHONE DATA PROCESSING

Cellular service providers use a data format called "Call Data Record" (CDR) to record subscribers' activities for billing purposes. The CDRs for a cellular network provide a reasonably accurate method of sensing the activity and location of users. An example of the type of information contained in CDRs is shown in Table 3.1. In the WIPER system, we use a stream of information aggregated from CDRs. Using raw CDRs, even when the identifying characteristics of a user are encrypted or obscured can present problems relating to user privacy[5]. In order to safeguard user privacy we have chosen to use only aggregate data, where CDR information is aggregated by tower and by a time interval. Empirical results have show that a time interval of 15 minutes provides a good tradeoff between smoothing the noise of activity while preserving overall trends in the change of activity levels.

Aggregating by tower location can lead to some issues related to the resolution of users' locations, most notably the large variability in the size of the area covered by a tower. In urban areas, due to the dense population, each tower covers a comparatively small area. However, in rural regions, the size of a cell is significantly larger. These limitations may no longer be an issue as cellular providers roll out 3G networks and users begin to adopt 3G handsets. In these advanced networks there are several methods of refining users locations to a level well below that of the cell, such as in-handset GPS or triangulation based on multiple towers, angle, timing and signal strength information. For the WIPER system we can envision using this capability to define grid cells of arbitrary size for aggregating users.

TABLE 3.1

EXAMPLE CDR DATA. TOWERS ARE UNIQUELY IDENTIFIABLE BY THE TOWER NUMBER. THE ID IS A CODE THAT DESCRIBES WHETHER THE CDR IS FOR CALLER OR RECEIVER AND THE SERVICE USED (VOICE, DATA, SMS, DATA, ETC). IN THIS EXAMPLE, 1 = CALLER, VOICE, 2 = RECEIVER, VOICE, 3 = CALLER, SMS, 4 = RECEIVER, SMS.

Date	Time	Caller	Receiver	Tower	ID
2007-01-01	00:00:02	888-555-1212	888-555-4763	4303472	1
2007-01-01	00:00:02	888-555-1100	888-555-8421	4303857	2
2007-01-01	00:00:03	888-555-1100	888-555-1212	4303857	1
2007-01-01	00:00:03	888-555-1634	888-555-2158	4303205	4
2007-01-01	00:00:04	888-555-8593	888-555-8745	4303765	2
2007-01-01	00:00:04	888-555-4564	888-555-2689	4303456	2
2007-01-01	00:00:05	888-555-0245	888-555-0245	4303468	1
2007-01-01	00:00:06	888-555-8903	888-555-4575	4303115	4
2007-01-01	00:00:08	888-555-6830	888-555-2355	4303485	3
2007-01-01	00:00:09	888-555-5354	888-555-6830	4303454	2

3.5.2 GIS AND MAPPING

Figure 3.9. An example of overlaying activity information on a satellite photo. Satellite image taken from Google Earth.

In the WIPER system, it is a design goal to utilize Free and Open-Source software whenever possible. To that end we have used GRASS GIS [31], PostGIS [85], GDAL [35] and Shapelib [108] to generate images, both interactively and as part of our automated workflow. We also use OpenMap [100] and Geotools [40] to enable GIS functionality in our simulations.

In generating our images, we primarily used GRASS. First we created a spatial-relational database using PostgreSQL [80] and PostGIS. This database contains both reference information on our area of interest, including geographic features,

political boundaries (cities, counties, zip codes, etc) and some information on major roads. Using GRASS we can combine the cellular phone users' activity data, aggregated at a particular time scale (images in this paper are aggregated at 10 minute intervals) with the historic information in the PostGIS database, allowing us to view several layers of information in one image. We also use GRASS to generate images that show the change in phone activity in an area over the course of a day.

3.5.2.1 ANOMALY DETECTION ON STREAMING DATA

We are currently developing our anomaly detection system to deal with multiple types of potential anomalies. A full treatment of this topic is beyond the scope of this paper. Those interested should read Pawling et al [78] and Yan et al [123].

3.6 PRIVACY AND ETHICAL CONCERNS

Concern about government monitoring of cell phone location and call activity may present a challenge for the deployment of the WIPER system [87, 94]. In order to address any concerns about privacy, the WIPER system is designed so that all personally identifiable data is removed from the data stream before it leaves the cell provider's network, ensuring that there is no potential for sensitive data to be abused. The software agents that handle the preprocessing reside on the servers of the cellular service provider and ensure that all data that is streamed across the internet is anonymized and encrypted. The WIPER system itself uses only aggregate data from the data streams and is not designed to allow the monitoring or tracking of individual handsets. We will continue to examine the potential

impacts of such systems on personal privacy, especially in the context of location-aware systems such as WIPER that utilize GIS systems and technologies[5, 6]

3.7 CONTRIBUTIONS

We have presented the architecture of the WIPER system. It is designed as a distributed system built on open standards to address crisis events in the real world. WIPER brings cutting edge social network analysis algorithms, anomaly detection on streaming data, sophisticated GIS-enabled Agent-Based Simulations and web-based interaction and visualization tools together in one package to enhance the decision making process of Emergency Management professionals. The system can interface with the existing cellular telephone network to allow cell phone activity to be monitored in aggregate, allowing the network to be used like a large scale, ad-hoc sensor network. The stream of incoming data is monitored by an anomaly detection algorithm, flagging potential crisis events for further automated investigation. Agent-Based simulations attempt to predict the course of events and suggest potential mitigation plans. Finally the system displays output at every level to human planners that can then monitor the current situation, react to changing events and evaluate mitigation strategies. The WIPER system is designed to integrate into a crisis response workflow, adding an important component to the toolbox of Emergency Response professionals.

3.8 FUTURE WORK

For further information on the WIPER system and up to date descriptions of the system and its components, visit `http://www.nd.edu/~dddas/`.

CHAPTER 4

DESIGN AND IMPLEMENTATION OF AN AGENT-BASED SIMULATION FOR EMERGENCY RESPONSE MANAGEMENT

4.1 ABSTRACT

This Chapter introduces the simulation component of the WIPER system [90]. The simulation is an Agent-Based Model of human activity, parameterized with agent location data taken from a cell phone network and run on GIS maps of the area. The simulations model both normal behavior and crisis events. Output of the simulations is in the form of call activity and agent locations, similar to that generated by the observation of cell phone users in an urban setting. A taxonomy of crisis events is presented, which has simplified the development of the simulation. The approach to the overall simulation design draws upon ideas from Pattern Oriented Modeling and agile techniques from Software Engineering. The simulation structure is presented, along with validation and verification of the system and an evaluation of runtime characteristics.

4.2 INTRODUCTION

Fires, riots, traffic jams and natural disasters are crisis events that can impact our lives. When possible, prevention is best. However, these events cannot be entirely eliminated, and so it is advantageous to spend time and effort developing

techniques to mitigate their impact. Simulations of these events can help us to understand the development of these events, as well as evaluating strategies for dealing with these crises.

Agent-Based Simulation is an accepted paradigm for simulating human behavior in realistic environments. The canonical examples of this are in traffic simulation, a relatively mature field, and in simulating the spread of infectious disease through urban areas. Important to both of these approaches is capturing the movements of individuals in the area. Simulating events like traffic jams or outbreaks of Avian Flu is advantageous for several reasons. First, a well developed simulation can provide insight into the spread of the disease or the conditions and behaviors that can lead to traffic jams. Second, computer simulations give planners the ability to evaluate scenarios to deal with the crisis event. Planners using the EpiSims simulation [12] can evaluate strategies for preventing the spread of Smallpox in a large urban area.

4.3 BACKGROUND

Agent-Based Models are well suited to simulating the behavior of complex systems, such as ecological systems, biological simulations and simulations of human behavior and movement [11, 24, 82, 88, 92]. These simulations cover a vast range of application domains, but there are similarities in the systems that make them amenable to the ABM approach. First, the system is composed of heterogeneous actors, which are called agents when simulated. Next, the desired output of the system is an aggregate over all the agents in the system. This can be some measure of the state of the agents, such as location, emotional state, etc. Finally, the simulation output is generated as a result of the interactions of the agents with

themselves and the environment.

4.3.1 AGENT-BASED MODELING

Agent-Based Modeling techniques grew out of a merging of the Discrete Event Simulation and Cellular Automata paradigms [9, 73]. Cellular Automata are simulations that exist in an environmental space, such as a grid, and the cells have certain behaviors or properties that develop over time and in relation to neighboring cells [104].

According to [9], a Discrete Event Simulation is a simulation where the "state variable changes only at a discrete set of points in time". Examples include time-stepped or event-based simulations that model processes, including many important to industry, such as scheduling the processing of parts in an assembly plant [20]. These simulations may be time-stepped, in which actions take place at integer intervals, or events can occur on a timeline, a schedule of events ordered by time. Simulations that use a timeline are still considered "discrete" due to the discrete nature of time, usually addressable at the level of a `double`.

An Agent-Based Model is comprised of a set of actors, the agents, which can be heterogenous and have individual movements, behaviors and goals and an environment (often spatially explicit, as on a cellular automata grid) where the agents exists. The behavior of the simulation is a result of the interactions among agents and of the agents with their environment.

4.3.2 TRAFFIC SIMULATIONS

Several Agent-Based Models of traffic flow currently exist [11] [47]. These models are useful in that they can produce useful information over a variety of

traffic conditions, from traffic jams to steady state traffic flow, which is a limitation of earlier approaches [42, 43].

4.3.3 INTEGRATING GIS WITH AGENT-BASED MODELS

Several research groups have explored the integration of GIS data sources with Agent-Based Models [14, 15, 50]. The advantages of this approach are that agents can interact with realistic environments, improving the validity of the simulation's predictions, as well as allowing planners and architects to evaluate the effectiveness of designs before they are built.

Using GIS data sources to represent the environment in Agent-Based Models requires rethinking the approach to simulation. Often the environment in an ABM is a grid structure, with agent movement delineated in discrete units. However, movement on a GIS is continuous and simulations must take this into account. Often land-use simulations will use discrete parcels of land in order to avoid issues with modeling arbitrary areas [105]. In the WIPER simulation we address the issue by simulating agent movements in terms of meters per time step and avoiding discrete representations of space.

4.3.4 APPROACHES TO THE DESIGN OF AGENT-BASED MODELS

Grimm et al have suggested the use of a Pattern-Oriented approach to composing Agent-Based Models [45]. This approach is similar to the Design Patterns approach to Software Engineering, as described in Gamma et al [37]. The Pattern-Oriented approach to model development starts by identifying patterns or motifs of agent behavior. In Individual-Based Ecology, examples of these patterns could be foraging behavior, nesting, mating, etc. A simulation is then developed as

a combination of these patterns. A heuristic for the composition for these patterns is to add only as many patterns as are necessary to demonstrate the desired behavior.

4.3.5 VALIDATION OF AGENT-BASED MODELS

Validation is the process of determining whether a model is a valid representation of the real system [9]. Some researchers have argued for the value of correctness proofs as a validation approach. This approach complements the canonical model development cycle, but is only practical when the modeler is using a toolkit that supports it, such as DEVS [124]. Bankes has suggested that Agent-Based Modeling is more correctly viewed as "an example of experimental mathematics" which places the interpretation of simulation results in the purview of statistics and not formal proof [8]. Indeed, many validation techniques as presented in the literature rely on statistical testing of results and not deductive proof [7, 122].

4.4 TAXONOMY OF CRISIS SCENARIOS

In order to simulate and predict the course of crisis events, it is necessary that we have, a priori, a set of crisis scenarios that we can draw from. To make the process of developing these simulations more straightforward, we have tried to analyze various crisis events and organize the events into a taxonomy. We placed the scenarios into functional groups, focusing on the behaviors and actions of the agents that will elicit these behaviors.

4.4.1 CRISIS CATEGORIES

We divide the crisis scenarios into 3 categories based on the principal movement characteristics of the agents. These categories are not meant to be exhaustive but merely descriptive of the events that we seek to simulate. The categories are as follows:

- Flock - Agents move in roughly organized fashion, like a mob.
- Flee - Agents move away from a disturbance.
- Jam - Agents move towards their customary goals, but are constrained, as in a traffic jam.

For the Flock category, agents move as a group, but without explicit leadership in a manner similar to the BOIDS movement model [86]. The Flock category is currently composed of one movement model, the mob model. This can be used to simulate scenarios where crowds of people are causing a disturbance, such as the WTO protests that occurred in Seattle in 1999 [118].

The Flee category is a much broader category and is applicable in a wide range of crisis scenarios. The category consists of models where agents are attempting to move away from some disturbance. The models in this category can be described concisely as flee from point, flee from line (not necessarily a straight line, this can include rivers/coastlines), flee from an area and bounded flee, where the agents get a certain distance away and stop. Some examples of crisis events that fit these scenarios would be people fleeing from a burning building (either a flee/bounded flee from point or flee from area, depending on map resolution), inhabitants fleeing a chemical spill (flee an area) and residents fleeing a tsunami (flee a line).

The final category is Jam, a collection of movements that are constrained. Agents in this category are trying to reach a destination (which may be unique for each agent), but the actions of all the agents together serves to create an event where movement is restricted for the entire system. The canonical example of this type of behavior is a traffic jam. This type of crisis scenario is often not necessarily an emergency event, though it can be, as in the case of the traffic jams on North-South highways in Florida in 2005 during the Hurricane Rita evacuation [116].

4.5 DESIGN

The WIPER Agent-Based Simulation has been designed using Design Patterns [37]. The use of Design Patterns is a common approach to the development of object-oriented software and in this regard they can be applied quite well to Agent-Based Models.

4.5.1 APPLICATION OF DESIGN PATTERNS TO SIMULATION

In order to reduce the amount of time and effort spent in writing the code for these simulations, we break out the agents' movement and activity models as Strategy and Singleton Patterns [37]. This is possible because many aspects of agent state remain constant, regardless of the underlying phenomenon that we intend to simulate. We extract and encapsulate agent behaviors related to movement and activity into objects, outside of the agent itself. Agents then retain a pointer to this movement or activity model object. Although this does seem to introduce a semantic disconnect with how we expect an agent to be designed, this approach offers huge benefits in model development and allows researchers to run

simulations where it is easy to initialize a population of agents with a few models and tractable to initialize the population with a large number (100 or more) of movement or activity models. More importantly this flexibility is apparent during the running of a simulation, when agents can change their movement or activity model at runtime, easily and without adverse affects on simulation performance, as switching models can be done as easily as changing a pointer.

4.5.2 SIMULATION BASE

All simulation scenarios share important components. The GIS files representing geography, political boundaries, tower information, etc are the same for all scenarios. A centralized simulation model class handles the initialization and setup of the simulation, placing agents onto the map, setting agent movement and activity models and handling the schedule.

4.5.3 CRISIS SCENARIO COMPONENTS

The crisis scenario taxonomy is used to guide our design of the crisis movement and activity models. A standard movement model has been developed in order to provide a baseline for comparison against the crisis models. The standard model, nominally referred to as "Move and Return" is intended to represent the activity of citizens in a non-crisis scenario. Agents have a "home" location, set at the beginning of the simulation and an alternate location, which we refer to as the "work" location. Agents move from the "home" to the "work" in the morning and return "home" in the evening. In the simulation, the locations of home and work can be set for each individual agent, as well as defining the movement schedule, which should allow the model to be validated against empirical movement studies.

Such studies using cell phone calling data are currently underway.

The taxonomy of crisis scenarios provides a framework for the implementation of the movement models. All of the models in a given category share similarities and can be arranged into an inheritance hierarchy accordingly, allowing reuse of related movement code.

4.6 VALIDATION AND VERIFICATION

Model validation is a necessary step in the development of a simulation. In order to demonstrate the validity of the simulation as a whole, several steps are taken to validate the underlying theoretical model and to verify the implementation. In this section we describe the verification and validation approaches used on the WIPER simulation.

As stated in [122] and [54], the Agent-Based Modeling research community has yet to embrace formal validation and verification techniques. The authors suggest techniques from the Discrete Event Simulation community that are applicable to Agent-Based Models. Using these suggestions, we present the verification and validation of the WIPER simulation using the following techniques:

- Face validation
- Synchronization of Random Number Generator
- Input-output correlation using empirical data

4.6.1 FACE VALIDATION

The most common validation approach is face validation. In this step, a domain expert examines the theoretical model and it's underlying assumptions and

determines whether this model is a reasonable representation of the intended phenomenon. For this simulation we have conducted face validation on the conceptual model ourselves, which is a common practice in the field. We have examined the assumptions for the simulation and the design decisions that have been made in order to make the simulation process tractable. Here we briefly present the design decisions.

For the simulation, we choose to model human behavior in 1 minute increments. The choice of time step was dictated by our need to generate calling activity at no more than 1 minute intervals. Agent movement simulations may benefit from smaller time steps, but there is a tradeoff between time resolution and simulation runtime. For the purposes of agent movement simulation, 1 minute intervals provides adequate accuracy to model agent behavior. Agents must move on a simulated represention of the world. We chose to implement this using a GIS, which provides an accurate model of the world. We chose to model cell tower coverage areas using Voronoi cells. This is an accepted technique for representing coverage areas, as stated in [66].

4.6.2 SYNCHRONIZATION OF RANDOM NUMBER GENERATOR

This verification technique tests the simulation to determine whether the random number generator is initialized and used properly and ensures that simulations results can be replicated. In this procedure, we ran multiple instances of the WIPER simulation with the same input parameters and with identical random seeds. This is vitally important in Agent-Based Models, as there are multiple stochastic processes and synchronization of the random number generator ensures that results can be replicated. The WIPER simulation uses a random number

generator to generate Normal and Uniform distributions, both from the Colt API [49].

A test of 4 replications each of 6 different random seeds was conducted. For each level of random seed, all simulations generated identical output. This is an excellent result for the verification of the simulation and confirms that the model implementation meets our criteria for replicability.

4.6.3 INPUT-OUTPUT CORRELATION WITH EMPIRICAL DATA

The WIPER simulation is designed to be a component in the WIPER emergency response system. The intended purpose of the simulation is to be used in a DDDAS system where the simulations are updated with and validated against streaming data from a cell phone network. With this in mind, the WIPER simulation was designed to generate simulation output similar to empirical data captured from a cellular service provider and furnished to the WIPER group. Here we present a detailed examination of the simulation output in terms of cell phone activity, comparing it to the empirical data taken from the CDR data. Call Data Record (CDR) data is the data used by cellular service providers for billing purposes. CDR records contain transaction records for the initiation and termination of calls, SMS messages and other services. For a further introduction to CDR data, see Section 3.5.1.

In order to perform these tests, we started by running 10 simulations, all with identical input parameters but varying the random seed for each simulation. We later increased this to 100 simulations. We parameterized the simulations with the number of active users for the simulated area, as taken from the empirical CDR data. We set the movement model to null, which causes the cell phone

user agents to remain at fixed locations throughout the simulation. We set the activity model to be a Distribution-based model, as described in Section 4.7.1. The Distribution-based model is designed to generate call activity in a way that is similar to that found in the empirical CDR data.

We present the call activity output of the first round of simulations in Figures 4.1 and 4.2. In Figure 4.1, the output of all of the simulations are plotted on top of the empirical data, which is in blue. In these figures the empirical data is the CDR data for the region for one day. In Figure 4.2, the results from each simulation is individually plotted against the empirical data. These plots provide an interesting graphical view of the output. A more useful plot of the output is the empirical data plotted over the range of the simulated data, shown in Figure 4.3. This plot demonstrates that the simulated data falls in a range around the empirical data.

A common test to determine if two data sets come from the same generating distribution is the Kolmogorov-Smirnov test[117]. This test is performed pairwise on the output of each simulation against the empirical data. In Table 4.1, we present the results of the K-S test on the initial round of 10 simulations, including the D value (the K-S test statistic) and whether the result indicates acceptance at the $\alpha = 0.10$, 0.05 and 0.01 levels. As shown in the Table, output from the 10 simulations all pass at the $\alpha = 0.01$ level of significance, 6 of 10 pass at the $\alpha = 0.05$ level of significance and 3 of 10 pass at the $\alpha = 0.10$ level of significance. It should be noted that the D-values are more significant as the value approaches 0.

According to Banks, goodness of fit tests, such as the K-S test, are sensitive to sample size and have a tendency to reject candidate distributions for large sample sizes[9]. Our simulations generate a sample of size 144, as the data is aggregated in

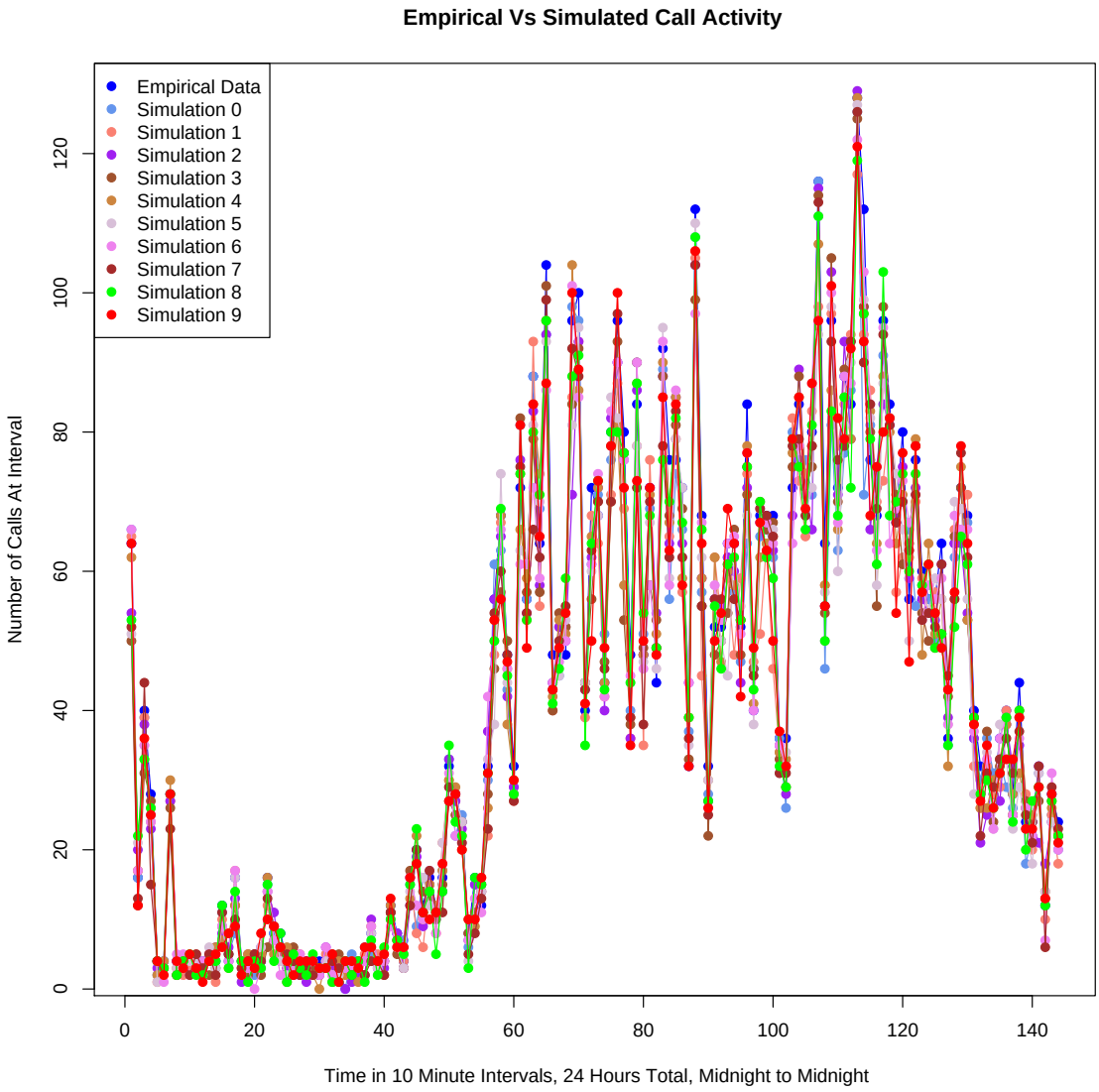

Figure 4.1. Plot of cell phone call activity from various simulation runs and empirical data for one day of actual and simulated time.

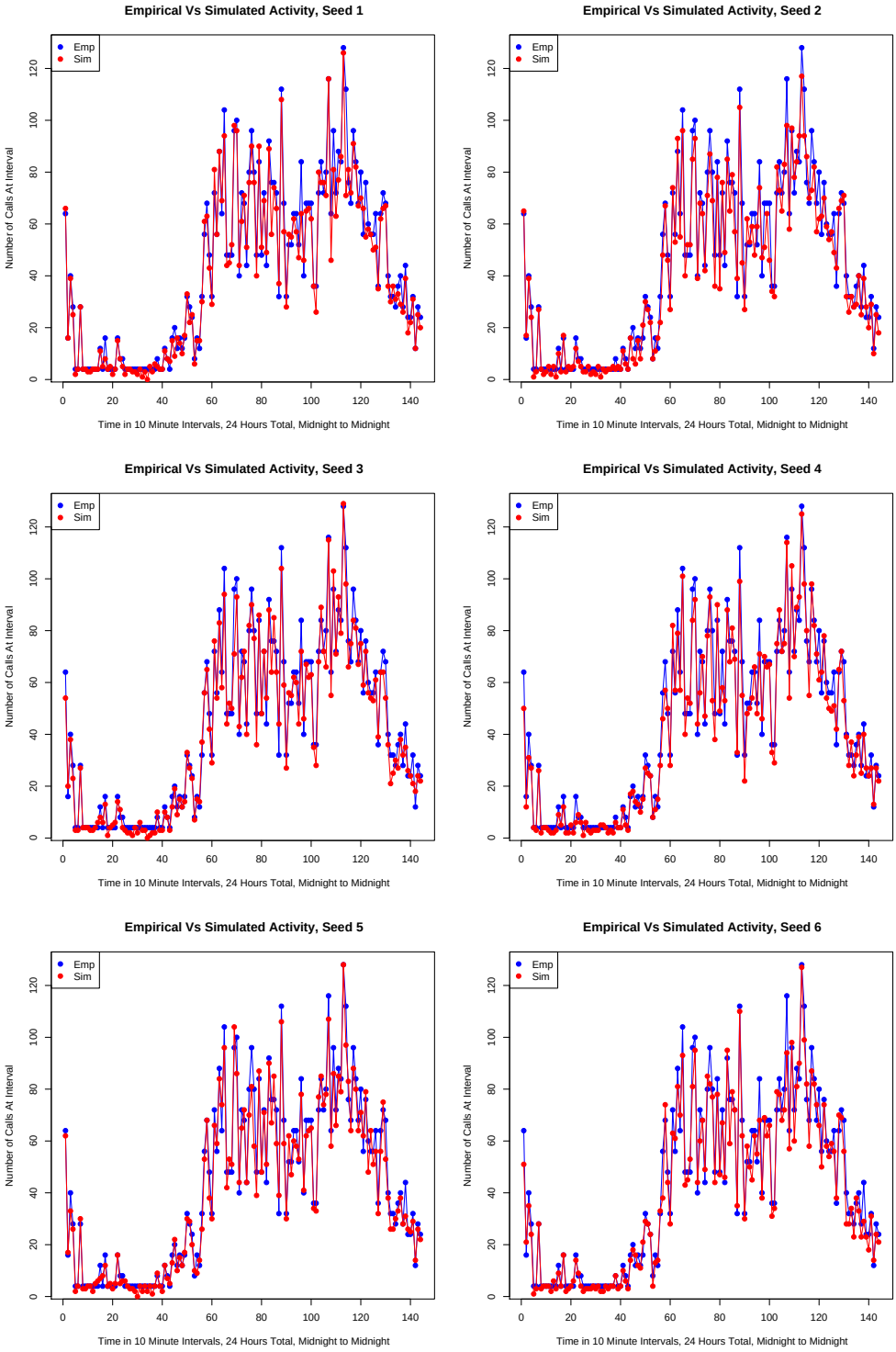

Figure 4.2. Plots of cell phone call activity from various simulation runs and empirical data for one day of actual and simulated time.

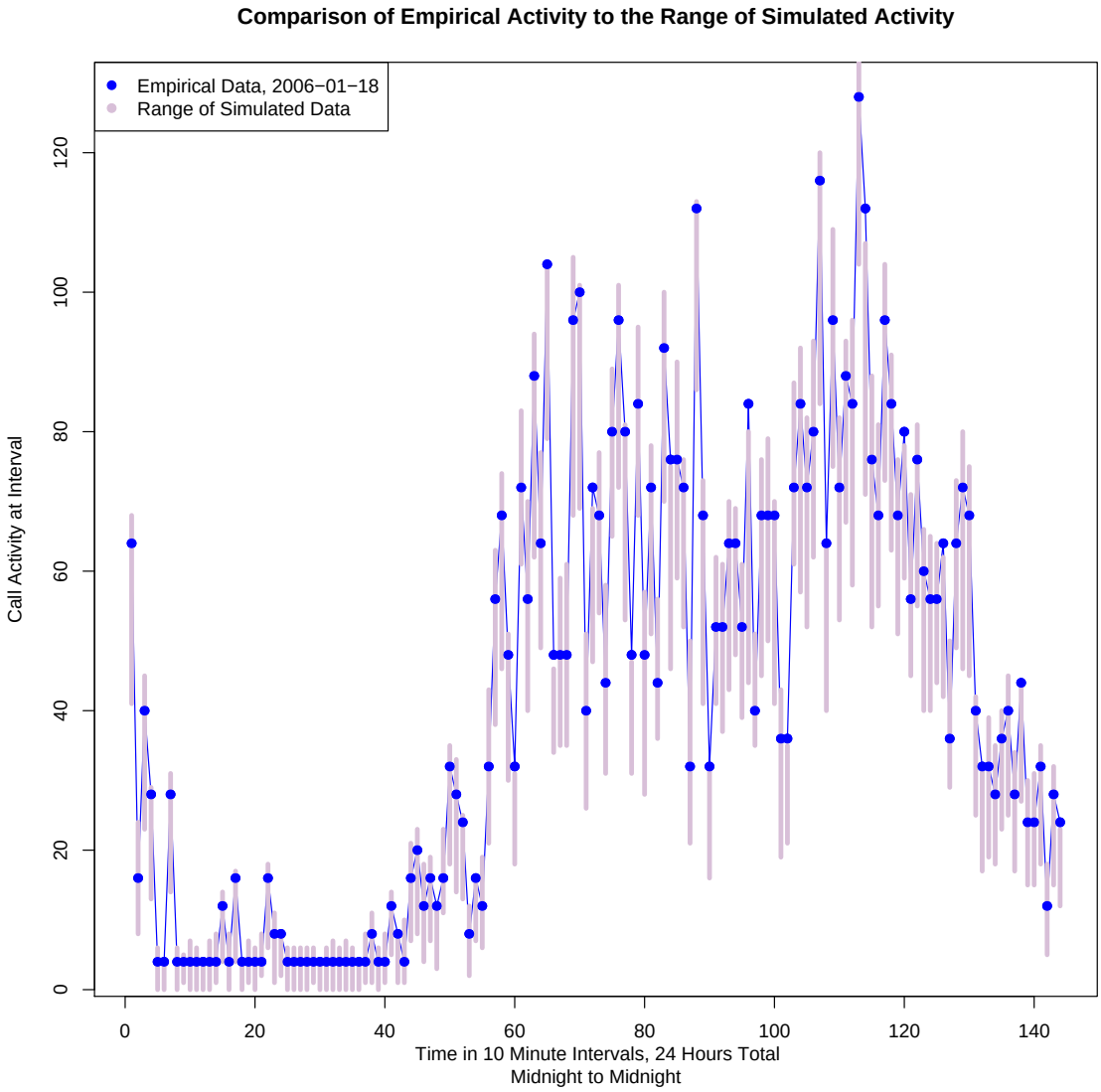

Figure 4.3. Plot of empirical call activity over the range of simulated call activity, demonstrating that the simulation generates activity without bias.

TABLE 4.1

EXAMINATION OF KS TEST STATISTICS FOR COMPARING SIMULATED CALL ACTIVITY TO EMPIRICAL DATA FROM TEN REPLICATIONS OF THE SIMULATION. GIVEN ARE THE D TEST STATISTIC AND A NOTATION AS TO WHETHER THE TEST FOR THAT SIMULATION IS ACCEPTED FOR $\alpha = 0.10$, $\alpha = 0.05$ AND $\alpha = 0.01$.

Simulation	D Statistic	Accept at $\alpha = 0.10$	$\alpha = 0.05$	$\alpha = 0.01$
1	0.903	YES	YES	YES
2	0.1042	NO	YES	YES
3	0.1319	NO	NO	YES
4	0.1319	NO	NO	YES
5	0.0903	YES	YES	YES
6	0.1181	NO	NO	YES
7	0.1111	NO	YES	YES
8	0.1042	NO	YES	YES
9	0.1250	NO	NO	YES
10	0.0764	YES	YES	YES

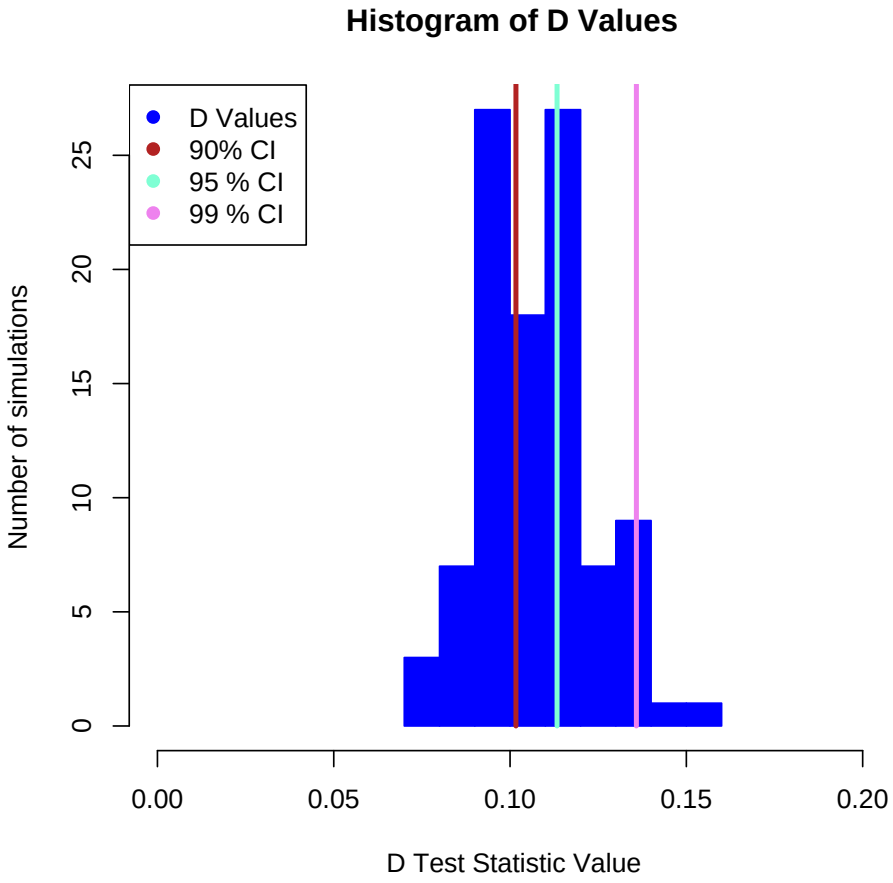

Figure 4.4. Histogram of the Kolmogorov-Smirnov test statistics for 100 runs of the simulation. The acceptance values for $\alpha = 0.10$, $\alpha = 0.05$ and $\alpha = 0.01$ are plotted as vertical bars. Lower test statistic values indicate higher confidence.

10-minute intervals over 24 hours. In Figure 4.4 we present a look at a histogram of the K-S test statistic on the output of 100 simulations. This was done to visualize how well the simulations fit the empirical data over a larger range of simulation runs. The histogram demonstrates that an appreciable number of the simulations rank above the $\alpha = 0.10$ level of significance and the vast majority rank above the $\alpha = 0.01$ level of significance. This result, on a much larger set of simulation output, demonstrates significant evidence that the WIPER simulation generates call activity data similar to that seen in the empirical data.

4.7 IMPLEMENTATION

The WIPER simulations are written in Java using the RePast Agent Modeling Framework [74]. The WIPER simulation also uses a number of Java APIs that come bundled with the RePast distribution, including the Colt High Performance Scientific Library [49] and GeoTools [40] and OpenMap [100] for GIS. A UML diagram of the WIPER simulation is shown in Figure 4.5.

4.7.1 ACTIVITY MODELS

The activity models define the calling behavior of the agents. There are 3 activity models used in the simulation: the `NullActivity` model, the `AlwaysCall` model and the `DistributionBased` model. All of the `ActivityModel` types are subclasses of `ActivityModel`. `ActivityModel` is designed as a Singleton class (as described in [37]), recognizing that all of the relevant state used to determine an agent's calling activity (Time, Date, Agent location) is maintained in the WIPER agent and in `WiperSimModel`. `ActivityModel` is an abstract base class with three methods:

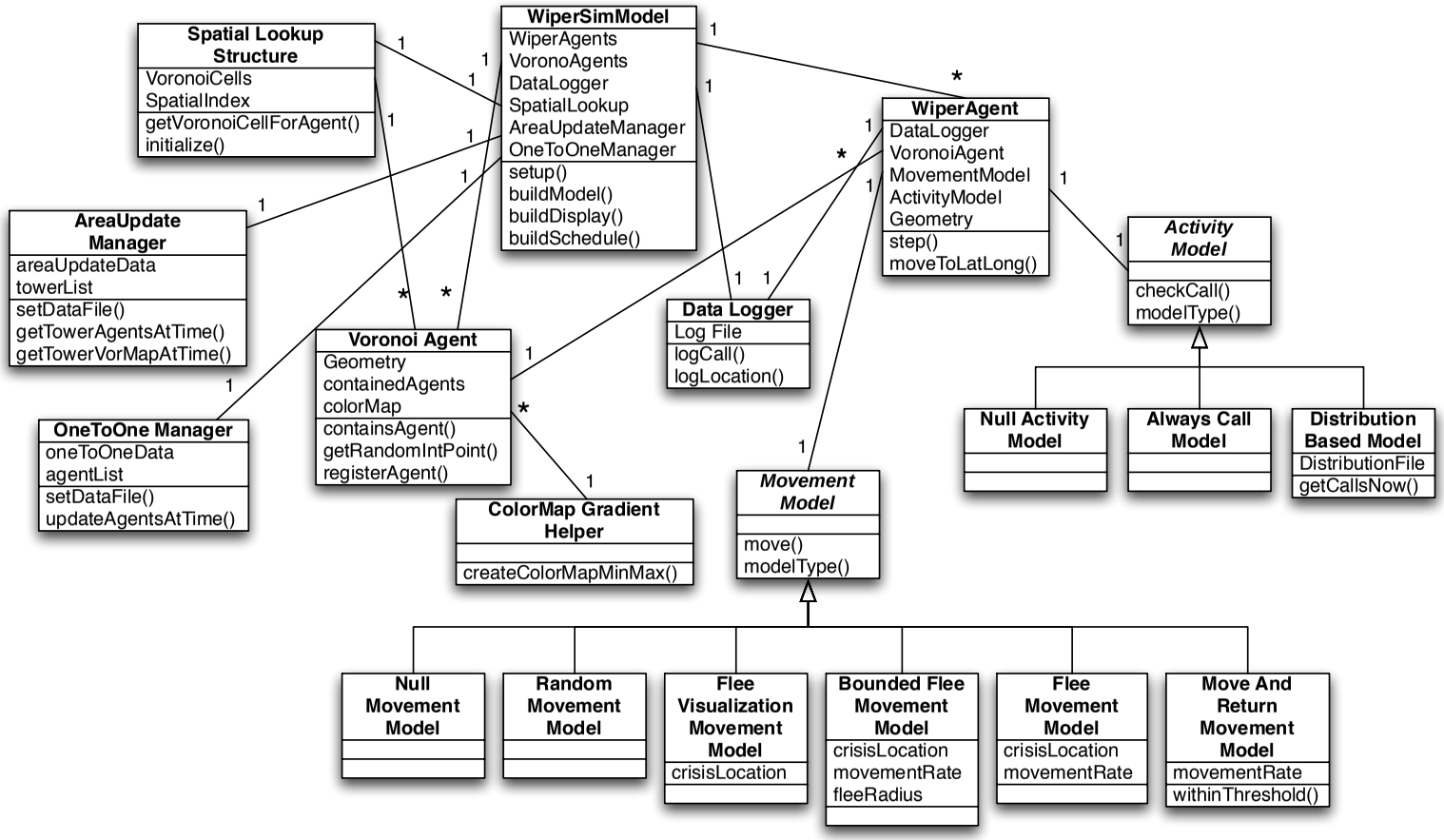

Figure 4.5: UML diagram of the WIPER simulation. Shown are contributed components and relevant methods and fields. Not shown are classes from outside packages such as the RePast API, Colt and Openmap.

`getInstance()` Singleton Accessor method.

`checkCall(WiperAgent)` Method for determining whether agent should make a call and if so, placing the call.

`modelType()` Returns the integer value of this model, using the model type definitions from `WiperSimModel`.

WIPER Agents use the `ActivityModel` objects in a Strategy Pattern [37]. This makes it possible to set the agent's behavior type as a parameter and provides fine-grained control over the behavior of individual agents in the simulation. The control over agent behavior allows large scale heterogeneity in agent movement models, an important consideration when attempting to model a large population of human agents.

The `NullActivity` model is a place-holder class. It overrides the abstract methods from `ActivityModel` but does not produce any behavior. This class is intended to be used for testing the Movement models, as it will not alter agent behavior.

The `AlwaysCall` model is another class used for testing. In this class, the `checkCall` method always causes the agent to make a call. This class can be used to test how agents call each other and to test other attributes of calling behavior.

The `DistributionBased` model is the primary call activity model in the simulation. This model requires initialization, it must be given an empirical call distribution as input. When the `checkCall` method is called, the method gets the (simulation) time and date from the `WiperSimModel` and then uses the empirical call distribution to determine the expected call activity at the current interval. A further discussion of the method of generating agent calling activity is given in Section 4.7.7.

4.7.2 MOVEMENT MODELS

The movement models define the manner in which WIPER agents move on the map. There are 5 different movement models: `NullMovement`, `RandomMovement`, `MoveAndReturnMovement`, `FleeMovement` and `BoundedFleeMovement`. As with `ActivityModel`, `MovementModel` is an abstract base class and is designed to be a Singleton. The abstract methods are:

`getInstance()` Singleton Accessor method.

`move(WiperAgent)` Moves agent on the map.

`modelType()` Returns the integer value of this model, using the model type definitions from `WiperSimModel`.

As with `ActivityModel`, WIPER agents use `MovementModel`s in the Strategy pattern. In this case, the agent encapsulates the location and any pertinent characteristics (movement speed, location of work or home, etc) and the `MovementModel` accesses these through the agent when calculating the destination.

The `NullMovement` class is a placeholder, implementing the `move` method but without causing the calling agent to actually move. This class is useful when testing `ActivityModel`s.

The `RandomMovement` class is used to move agents in a random fashion on the map. When the `move()` method is called, a random direction is chosen and the agent is advanced along that direction. The distance traveled depends on the agent's movement speed and the length of a simulation time step. The agents do not continue traveling along this path, at each time step a new direction is chosen.

The `MoveAndReturnMovement` class defines a "daily routine", where agents travel from a home location to a work location and back. This model works in con-

junction with events scheduled in the `WiperSimModel` and with state maintained in the WIPER agents, the home and work locations, information on whether they are traveling to home or work, etc. The `sendAgentsToWork()` and `sendAgentsHome()` methods are placed on the schedule at initialization by the `WiperSimModel` and scheduled to execute at a particular time in the simulation. For example, the `sendAgentsToWork()` method may be scheduled for 8am in simulation time and the `sendAgentsHome()` method scheduled for 5pm, corresponding to a typical American workday. These methods work by setting the `MovementModel` of the WIPER agents to the `MoveAndReturnMovement`, which will cause the agents to begin moving towards work or home, respectively, at the next time step. The WIPER agent maintains the location of its own work and home locations, and the `MoveAndReturnMovement` uses these to calculate the next position of the agent.

The `FleeMovement` class is an implementation of a crisis movement class. The `move()` method moves each WIPER agent in a straight line away from a disaster location. The location of the crisis is initialized in the `WiperSimModel` at the start of the simulation. In this class, WIPER agents always move away from the crisis and continue moving until the simulation ends. This type of behavior is similar to what is expected in a major disaster scenario, such as the fleeing from Manhattan on September 11. Due to the relatively short duration of the WIPER simulations, minutes and hours, rather than days, this type of behavior is a good first-order approximation.

The `BoundedFleeMovement` class is a refined version of `FleeMovement`. In this `move` method, each WIPER agent moves directly away from the crisis location until it reaches a threshold, then stops. The crisis location and the flee radius are initialize in `WiperSimModel` at the start of the simulation. This type of crisis

behavior is more consistent with what would be seen in a building fire or small-scale crisis, where people flee the crisis until they reach a safe distance.

4.7.3 THE WIPER SIMULATION MODEL CLASS

In the WIPER simulation, the `WiperSimModel` class extends `SimModelImpl` and is responsible for the creation, initialization and management of the simulation. The `SimModelImpl` class is a RePast class that partially implements the `SimModel` interface, which is designed to control the schedule, run the simulation and respond to input from the RePast GUI console. In the WIPER simulation, `WiperSimModel` is the largest and most complex class.

When a WIPER simulation is started, `WiperSimModel` receives the initial parameters either from the RePast GUI (for interactive simulations) or from the command line (when used as part of the WIPER Simulation Prediction System or when dispatching simulations to a computational grid). The initial simulation parameters are used to specify which components will be visualized on the graphical display, configure the GIS map with voronoi cells and geographical features, initialize the `WiperAgent`s (either from a file or via a string of tower IDs and number of agents at each tower), schedule actions (these can include generating and saving GIS snapshots, logging of calls and agent locations, scheduling each `WiperAgent`'s `step()` method, etc) and handle proper clean up at the end of the simulation (closing output files, etc).

4.7.4 THE WIPER AGENT CLASS

The primary purpose of the WIPER simulation is to model the behavior of cell phone carrying inhabitants of an urban area. The `WiperAgent` class is the imple-

mentation of the cell phone agent, encapsulating state information such as current location, home and work locations, current cell tower / voronoi cell. As described above, `WiperAgent`s use the Strategy pattern [37] for holding their movement and activity models, which allows unprecedented flexibility across the population of agents. This means that movement or activity models can be assigned individually to agents and that the behavior of an agent can be changed easily by replacing its movement or activity models.

A `WiperAgent` is initialized with a current location, links to the `WiperSimModel` and the centralized logger, given a starting location, home and work locations and movement and activity models. At each time step, the agent's `step()` method is called. In this method the agent calls the `checkCall()` method in it's `ActivityModel` and the `move()` method in it's `MovementModel`. If the agent makes a call, it records this information in the log. Similarly, an agent can be configured to record its movement in the log, however agent movements are usually recorded by `WiperSimModel` at some interval (usually a multiple of the time step), in order to reduce the size of the logs.

4.7.5 CENTRALIZED LOGGING ARCHITECTURE

Agent movements and calling activity are recorded to the WIPER simulation log using a centralized logging architecture. The `DataLogger` class acts as a central entry point to the log, collecting information from agents and sending the results to a file. The `DataLogger` produces two files for each simulation run, an activity file with all call activity and a location file with agent locations. The activity file records agent calling activity in a format identical to the CDR data from our cellular service provider. For a more thorough introduction to the CDR file

format, see Section 3.5.1. This compatibility makes it possible to use simulation-generated activity files in place of empirical CDR data in the RTDS component of the WIPER system (see Section 3.4.1).

4.7.6 GIS

Various GIS data sources are used in the WIPER simulation: roads, political boundaries, cellular tower locations, voronoi cells. A voronoi diagram is a tiling on a surface around a set of points $p \in P$ where each point p_i occupies a unique cell and all points within that cell are closer to p_i than to any other $p \in P$[114]. The voronoi cells in the WIPER simulation are a tiling of the map made from the cellular tower locations. These cells provide a good first order approximation of the coverage area around each tower, allowing the simulation to read in agent activity at a tower and translate this to approximate agent locations on a map. Political boundaries, such as postal codes, city limits, etc are useful when attempting to paramterize agents based on census data or other demographics linked to location.

4.7.7 IMPLEMENTING SIMULATED CALL ACTIVITY FROM EMPIRICAL DATA

In order to create a valid activity model, we base our activity model on empirical data taken from the historical data collected for the WIPER project. This empirical data is a trace of Call Data Records (CDR) taken over a 15-day period in early 2006. Each entry in the CDR file is a transaction corresponding to the beginning or end of a call or SMS message, containing the date, time, caller, receiver, tower number and transaction ID (either call begin/call end or SMS begin/end). The CDR files contain 1,113,222,456 records, with a breakdown of the transaction

types shown in Table 4.2.

TABLE 4.2

BREAKDOWN OF CALL TRANSACTION TYPES IN THE CDR FILES.

Transaction	Count
Call Begin	375,363,655
Call End	347,705,509
SMS Begin	127,172,484
SMS End	262,980,808
Total	1,113,222,456

4.8 CONTRIBUTIONS AND RESULTS

In this section we present a thorough examination of the characteristics of the WIPER simulation, including runtime performance, scalability and offline characteristics. Output of the simulation, including screen shots of the visual components, is also provided.

4.8.1 GUI DISPLAY AND VISUAL COMPONENTS

A sample screen shot of the WIPER simulation is shown in Figure 4.6. In this image, the WIPER simulation is being run in GUI mode, using the standard

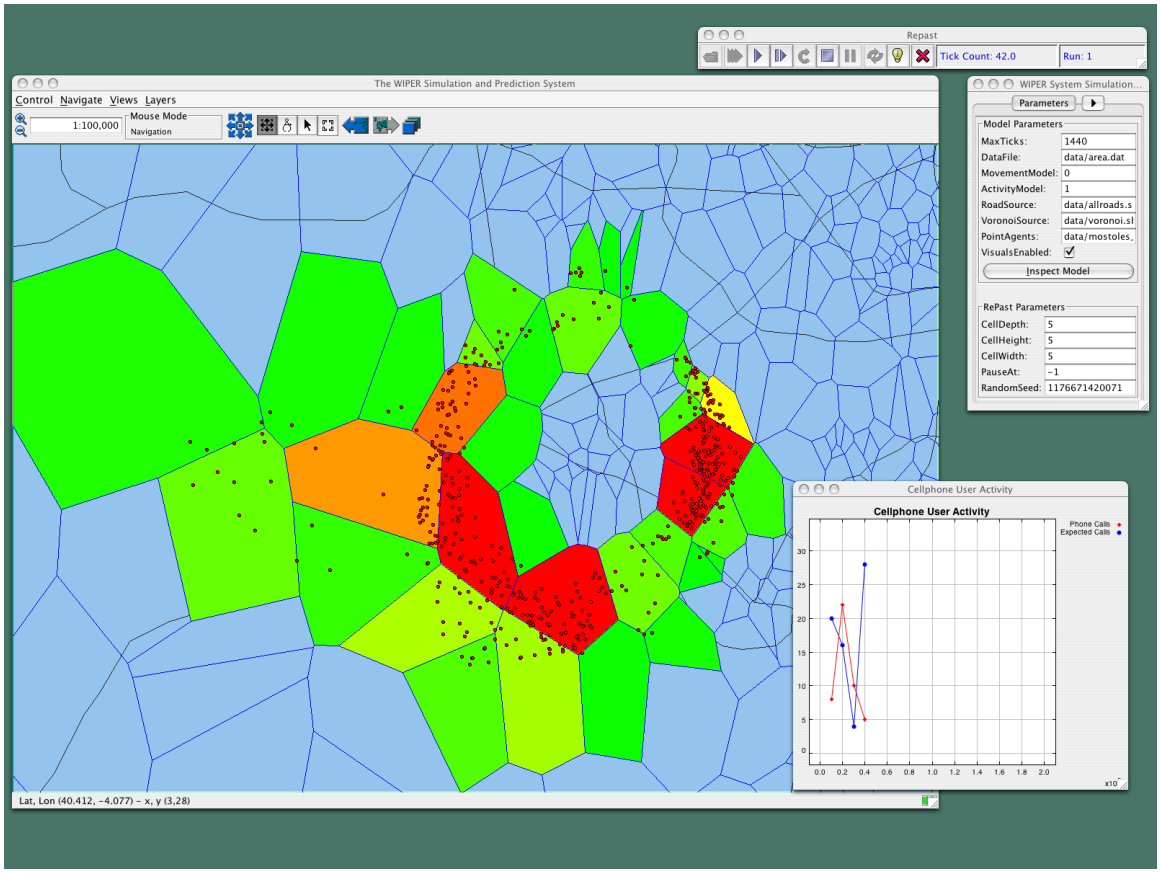

Figure 4.6. Screen shot of the WIPER simulation. This simulation is a Flee movement model with distribution-based activity model. Agents are represented as red dots with the Voronoi cells colored by the number of contained agents.

RePast toolbar to control the simulation and adjust input parameters. The map is an OpenMap display showing agent locations and the voronoi cell boundaries, with the voronoi cells colored by the number of agents within each cell's boundaries. The call activity window shows a running display of the call activity of all agents in the simulation, plotted against the empirical call activity for this time and day.

When the WIPER simulation is run interactively, a researcher can observe via the GUI display the state of the system and the recent agent behaviors. In order to improve runtime performance, the plotting of agent locations can be disabled. The overall agent locations can still be determined through the coloring of the voronoi cells but with a lower degree of accuracy.

4.8.2 RUNTIME PERFORMANCE

The runtime performance of the WIPER simulation in response to changing levels of graphical output is presented in Table 4.8.2. All simulations for this analysis are started with 500 agents, using the `MoveAndReturnMovement` and `DistributionBased` models. The simulation runs for one simulated day, 1440 time steps, with each time step representing one minute. The simulation can be run without any graphical display (the default when running simulations on a computational grid), with a GIS display that shows information such as cell tower locations, the surrounding Voronoi cell for a tower and agent locations and finally the simulation can generate snapshots of the GIS display, which can be displayed on the graphical console of the WIPER system or compiled together to form a movie.

The results shown in Table 4.8.2 are the total cumulative running time for 20 runs of the simulation. The results clearly demonstrate that there is a cost

TABLE 4.3

CUMULATIVE RUNTIME FOR 20 RUNS OF THE SIMULATION WITH VARYING LEVELS OF GRAPHICAL OUTPUT. GIS - GIS DISPLAY, SHOWING TOWER LOCATIONS AND VORONOI CELLS COLORED BY NUMBER OF CONTAINED AGENTS. AGENT LOCATIONS - THE LOCATION OF ALL AGENTS ARE ADDED TO THE GIS DISPLAY. SNAPSHOTS - EVERY 10 TIME STEPS THE SIMULATION MAKES A SNAPSHOT OF THE GIS DISPLAY.

GIS	AGENT LOCATIONS	SNAPSHOTS	TIME
NO	NO	NO	240.34s
YES	NO	NO	354.70s
YES	YES	NO	363.44s
YES	YES	YES	35200.29s

associated with running the graphical display and generating snapshots of the GIS. It is interesting to note that the penalty for visualizing the agents on the GIS display is negligible, compared with the cost of the GIS. Generating the snapshots only once every 10 time steps requires an approximately 2 orders of magnitude cost.

The scaling of the simulation with regards to number of agents, area fixed, is shown in Figure 4.7. In the simulations, all `WiperAgents` are started with `MoveAndReturnMovement` and `DistributionBased` models and are initialized into the same geographical area. For each level of agent population size, 25 replications with unique random seeds were run. The agent population varied from 10 to 10,000

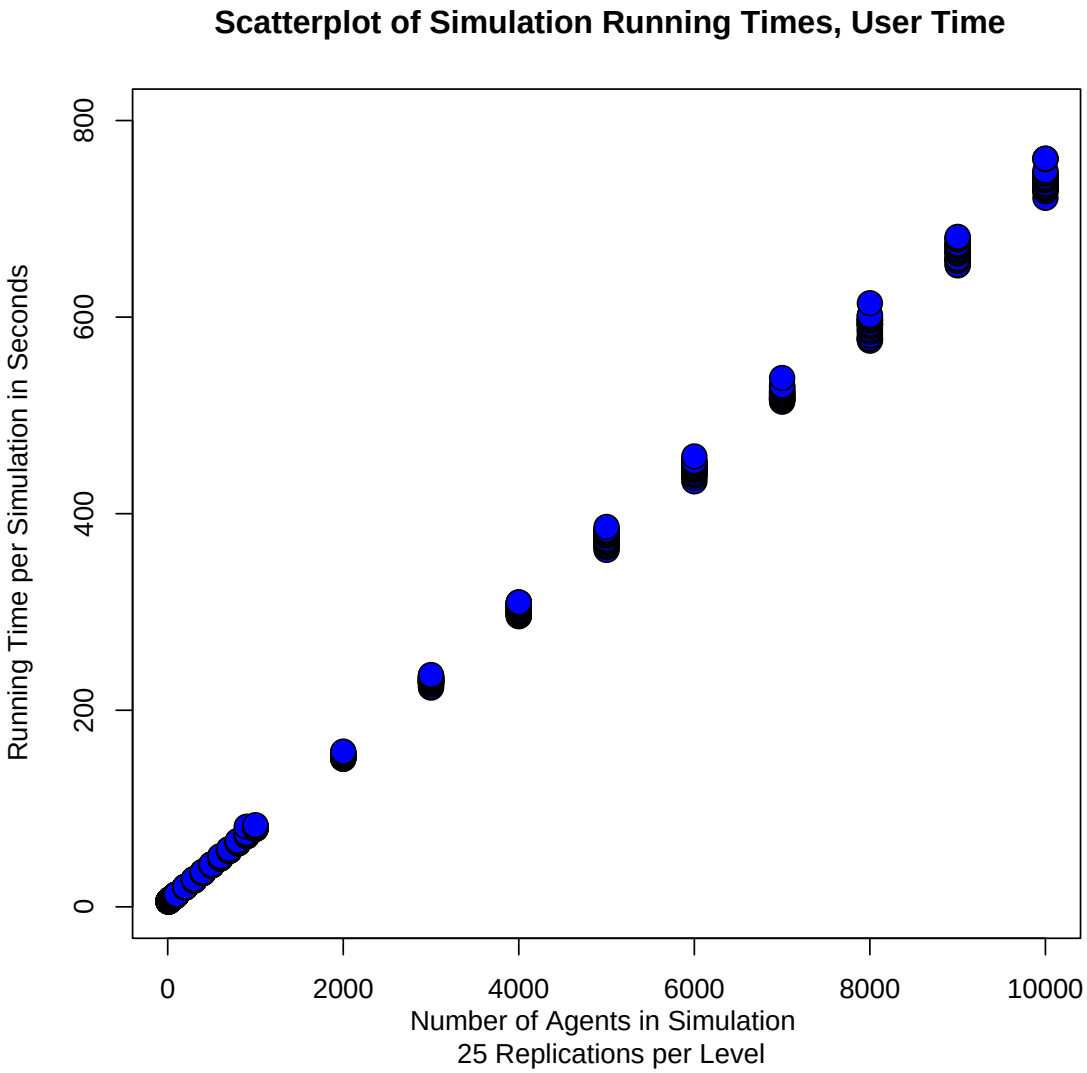

Figure 4.7. Simulation scalability results showing variations in running time. Visual analysis indicates that simulations scale linearly with respect to agent population size.

in the following manner: 10-100 by increments of 10, 100-1000 by increments of 100 and 1000-10,000 by increments of 1000.

Table 4.4 gives the summary statistics for a subset of the running times. The simulations demonstrate a low standard deviation about the mean indicating that the runtime results should be a good predictor of simulation runtimes when deployed in the WIPER system.

The plot of the mean user time for each level of agent population are shown in Figure 4.8. In this figure, the scaling of simulation time with respect to agent population is nearly perfectly linear. Linear regression on the data generates a line of $m = 0.07352$, $b = 5.91087$, with an adjusted R-Square value of 1 and a p-value of $< 2.2 * 10^{-16}$.

Figure 4.9 shows the scalability of the simulations with respect to agent population out to a population size of 1,000,000 agents. These simulations run for 60 simulated minutes and population size is sampled from 1,000 to 10,000 in steps of 1,000, from 10,000 - 100,000 in steps of 10,000 and from 100,000 to 1,000,000 in steps of 100,000. In these simulations the initial starting position of the agents is constrained to a given geographical area, but as the simulation progresses the agent population diffuses, so that larger agent populations will occupy a larger geographical area.

The figures clearly show linear scaling with respect to varying numbers of agents in the WIPER simulation for agent population up to 10,000 agents and respectable but above linear scaling when simulations have more than 200,000 agents. This result is confirmation that the simulation demonstrates excellent runtime characteristics. It is unlikely that an improvement can be made on the scaling, as the nature of the simulation requires that each agent be visited at each

TABLE 4.4

AVERAGE RUNNING TIME AND STANDARD DEVIATION FOR SIMULATIONS WITH 10-10,000 AGENTS. TIMES ARE IN SECONDS. SIMULATIONS DISPLAY LOW STANDARD DEVIATION THAT SCALES APPROPRIATELY WITH RUNTIME.

Number of Agents	Mean Running Time	Std. Dev.
10	5.6172	0.14025334
20	6.4272	0.09688997
⋮	⋮	⋮
90	11.7548	0.14338526
100	12.4444	0.15564597
200	20.0124	0.19751118
⋮	⋮	⋮
900	73.4960	1.88104138
1000	80.6840	0.86200348
2000	154.2744	1.94242563
⋮	⋮	⋮
9000	667.6448	8.62510054
10000	738.6588	10.71719814

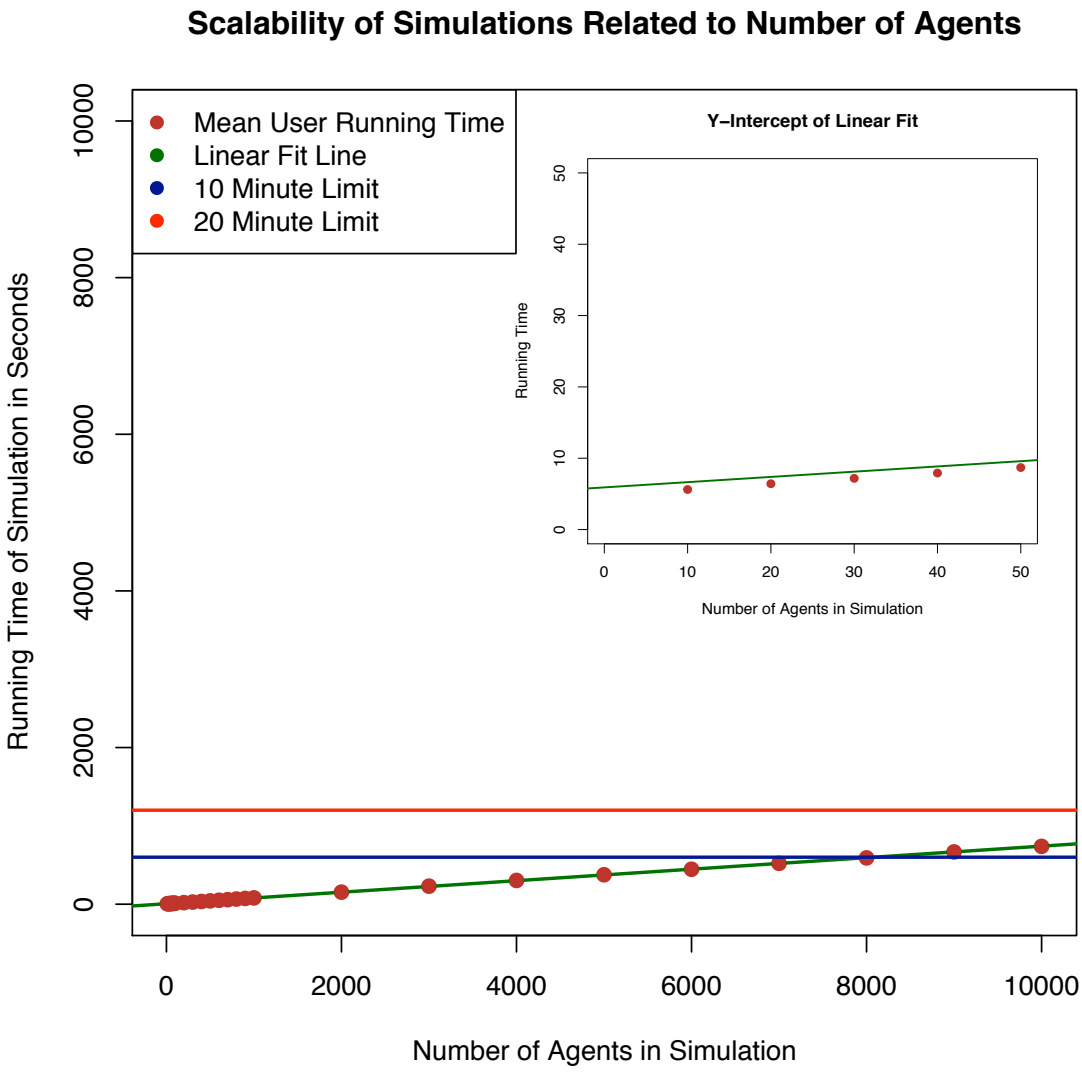

Figure 4.8. Simulation scalability with respect to agent population size. Results shown with linear fit line.

time step.

4.8.3 OFFLINE CHARACTERISTICS

The WIPER simulation, in a code snapshot from April, 2007, contains 3595 lines of code and an additional 1355 lines of unit testing code. When packaged as an executable `jar` file, along with supporting libraries, the simulation is 7.7MB. Additional data files require approximately 10MB of space.

There are three types of output files from the simulation: a .txt file containing CDR data, a .loc file with agent locations at regular intervals and (optionally) .png screenshots. The generated CDR data file is in ASCII text and is about 500KB uncompressed for 1650 agents with a duration of 1 day. The location file for the same scenario, also ASCII text, is 6.5 MB uncompressed. The screenshots consist of images of the GIS map covering the initial area of the simulation and are 40KB each.

4.8.4 DESIGN CONTRIBUTIONS

The development of an Agent-Based Modeling simulation, as with software in general, is an iterative and ongoing process. A simulation is never truly "finished", as there is always some area that can be improved upon: speed of execution, ease of use, memory footprint, variety of crisis scenarios, accuracy / validity of model results, etc. With this in mind, the WIPER simulation has been designed to be easy to maintain and extend. The simulation is designed using conventional software engineering techniques such as Design Patterns [37]. Components of the simulation are tested and verified using an extensive and comprehensive Unit Testing suite. The simulation has been developed using an Agile approach, with the

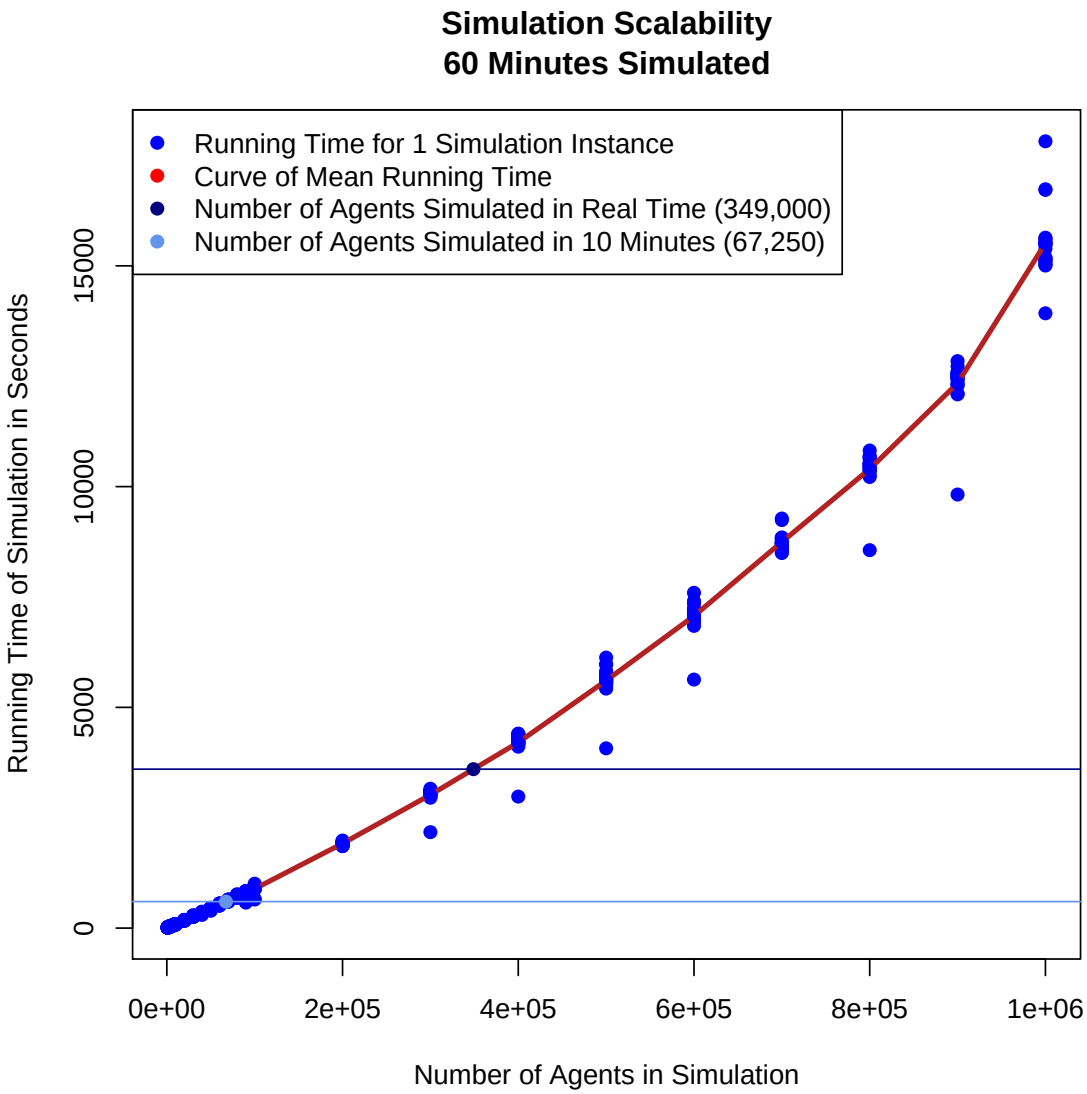

Figure 4.9. Simulation scalability with respect to number of agents, 20 replications at each level. In this instance we run simulations for 60 minutes of simulated time. In this graph we examine agent simulations out to a population size of 1,000,000 agents.

design of the simulation revised several times using the concept of Refactoring as described by Fowler [36]. Although these characteristics of the WIPER simulation are difficult to measure quantitatively it is clear that they certainly contribute to the extensibility and maintainability of the simulation.

Developer and user documentation is provided in the form of annotated JavaDoc pages, with extensive comments from the developer and references to relevant classes in the various APIs used by the simulation, e.g. RePast, OpenMap, etc. Additional documentation for end users is provided on a research wiki and will be made available upon request.

4.9 SUMMARY

In this chapter we have presented the WIPER simulation, an Agent-Based Model for simulating the movement and calling behavior of cell phone users. The simulation has a calling activity model based on the activity observed in empirical data taken from cellular service provider records. The activity generated by the simulation is indistinguishable from the empirical data according to statistical tests. The movement models demonstrate various behaviors that are important when attempting to simulate crisis behavior. Also, as the simulation is designed to be a part of the WIPER system for emergency response, where simulations are updated with streaming data, runtime performance is an important development goal. We have demonstrated through a scalability exploration the runtime characteristics of the simulation, showing that it displays adequate performance for use in the time-critical WIPER system.

4.10 FUTURE WORK

The current cell phone activity models represent usage under normal scenarios. We would like to study the existing CDR data and examine behavior of people during crisis events. We will use this information to guide the creation of crisis activity models and will use our existing validation framework to evaluate this approach.

As shown, the simulation does not implement all of the crisis scenarios described in the crisis taxonomy. We are in the process of implementing a more comprehensive set of crisis scenarios. As the simulation is designed for use in simulating crisis events, we would like to make the scheduling of crisis events easier. Currently this requires changes to the source code.

In order to better model human behavior and the spread of information as it relates to crisis events, it may be useful to examine creating a social network structure on the agents. The could allow us to study information diffusion across a population, as well as examining the interplay between physical location and distance in a social network.

CHAPTER 5

CREATING AND UPDATING AGENT-BASED MODELING SIMULATIONS FROM STREAMING REAL-TIME SENSOR DATA

5.1 ABSTRACT

According to the DDDAS approach, the accuracy of simulations can be improved when they are created with recently acquired data [26]. Extending this idea, the predictive ability of simulations can be further improved when they are updated in an online fashion with streaming data. However, the process of creating and updating simulations with streaming sensor data presents several challenges: How should parameters detected in the real world be applied to the simulation? Is it worthwhile to pursue a one-to-one mapping between agents in the simulation and their referent in the real world? Are the costs of maintaining complex, one-to-one relationships between agents and referents justified with improved model payoff? In this Chapter we explore the challenges to creating and updating simulations with streaming sensor data, develop an approach to solving the challenges and evaluate the approach in the context of the WIPER project, an Emergency Response system that is designed according to the principles of DDDAS [90].

5.2 INTRODUCTION

Emergencies strike without warning. In order to model floods, crowd evacuations, traffic jams and other time-critical events, it is important to be able to create simulations that reflect our knowledge of the current state of the world. Sensor networks, real-time traffic data and cell phone networks provide simulation modelers and governmental agencies with an unprecedented level of information regarding the state of the world. This information can be used to parameterize simulations that attempt to model the world, but only if simulations are designed appropriately and a framework exists for updating running simulations with streaming data. This chapter proposes an approach to improving the effectiveness of simulations with streaming data by stating model design assumptions that make it possible to create and update simulations with streaming data and by providing a framework for updating simulations with streaming data.

Simulation is a well-accepted approach to planning for emergencies and testing approaches to mitigating crises. Government agencies currently use agent-based simulations to test out different scenarios for combatting the spread of biological organisms, such as smallpox or avian flu[12, 33]. In the Emergency Response and Management field, the term "simulation" can represent live-action re-enactments of crisis events, pen and paper simulations or computer simulations. In this Chapter we take the term simulation to only refer to computer simulations.

Simulations can vary in their predictive ability, but are greatly improved when they can be grounded with real-world information [26]. We consider the real-world data in two different classes. The first class of data is relatively unchanging data, such as GIS data that faithfully represent topography, transit networks and land use. This data varies on a long time scale, that of years or decades, and can be

safely created and updated offline. The second kind of real-world data varies on a short time scale, that of minutes or hours, such as the locations of cell phone users or the location, speed and direction of traffic. In the simulations that we use to evaluate our approach, environmental data is considered in the first class of data, is created offline and is constant over all simulation instances. Data on agent locations and behaviors is used online and varies over time (though the data stream itself is re-used for all simulation instances).

5.3 PROBLEM STATEMENT

Researchers agree that streaming data has the potential to improve the predictive ability of simulations [26] [29]. However, there is no consensus about how to approach the challenges of creating and updating simulations with streaming data. In response to this confusion and recognizing the potential contribution to the field of simulation, the NSF has created the DDDAS initiative to address these challenges.

5.4 BACKGROUND

Validation is described as "getting the right model" according to [7]. By this, Balci means that validation is the process of determining whether the conceptual model is a reasonable representation of the system it is designed to simulate. Various validation techniques exist that are appropriate for use with Agent-Based Models. Kennedy describes several approaches in [54].

The Optimization via Simulation approach seeks to determine input parameters for a model that yield optimal output. This approach merely seeks to make optimal choices on selecting input parameters while keeping the underlying model

the same. Pichitlamken and Nelson describe a combined procedure for Optimization via Simulation in [79]. The authors approach has three components: "a global guidance system, a selection-of-the-best procedure and local improvement". This approach uses the Nested Partition method of Shi and Olafsson [93] for global guidance and a hill climbing approach for local improvement. Although the authors offer an interesting framework, certain drawbacks make utilization of the entire approach impossible under our constraints. First, the selection of samples from input space to initialize the global search can be NP-Hard. Second, the hill climbing approach is susceptible to becoming trapped in local optima and relies on randomization and the global search procedure to provide guidance. In the WIPER system it is acceptable to use a local search algorithm and forgo attempts at optimal prediction, as the simulation component is continually updated with streaming data and long-term defects are corrected by the introduction of sensor data.

5.5 APPROACH

The benefits of updating simulations with streaming sensor data are manifestly obvious. Consider Figure 5.1 (adapted from [29]). Here we see the results (in output space) of updating simulations with streaming sensor data. Sensors collect information about the progression of an event in the real world, starting with the event at t_{-1}

Our approach to addressing the challenges of creating and updating Agent-Based simulations in a DDDAS is to aggregate sensor data to a larger level, in the case of WIPER to the cell-tower level, and to introduce random variation in the data. A naive approach would suggest that it is important to maintain as

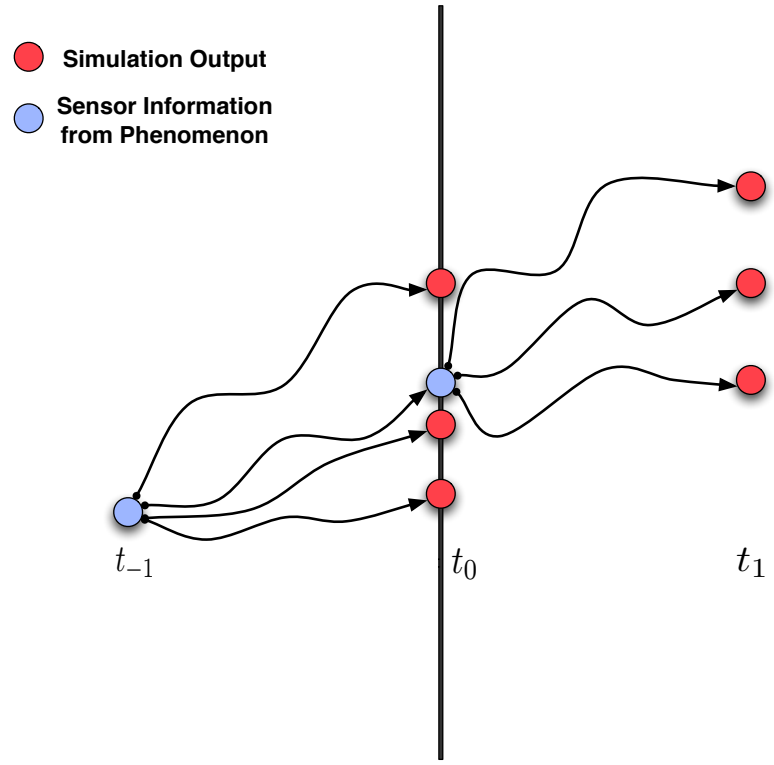

Figure 5.1. A visual representation of online updating of simulations. The figure shows various simulation trajectories in output space and compares them to an event as detected by sensors. Figure adapted from [29].

much data as possible about each individual in the system. Our streaming cell phone data could potentially yield location information on every individual down to a resolution of a few meters. However, canonical model development practice suggests that such an approach would be counter-productive, leading to naive realism, unduly adding complexity to the model without a corresponding increase in model accuracy or usefulness [9, 45].

5.5.1 UPDATING AND RE-PARAMETERIZING

Figure 5.2 shows a graphical comparison of two approaches to revising simulations with streaming data. Simulations can be updated or reparameterized. We define updating to refer to the process of restarting simulations with approximate information on agent locations (and other parameters). In the Figure, this corresponds to receiving information on the number of agents in a voronoi cell, but without specific location information on each agent. We define re-parameterizing as the process of maintaining a 1 to 1 correspondence between human beings in the real world and agents in the simulation. As information streams in about the corresponding referent in the real world, we modify the state (parameters) of each agent to reflect the rich information about the corresponding human, such as precise location, movement trends, etc.

There are tradeoffs involved in creating more detailed models with larger parameter sets. The practice of trying to capture all possible information about a system in a model is a common problem in the modeling community and is called Naive Realism [45]. Banks describes the phenomenon as the tradeoff between model complexity and "payoff", where payoff is defined to be usefulness of the model and combines various measures such as fitness for its intended purpose,

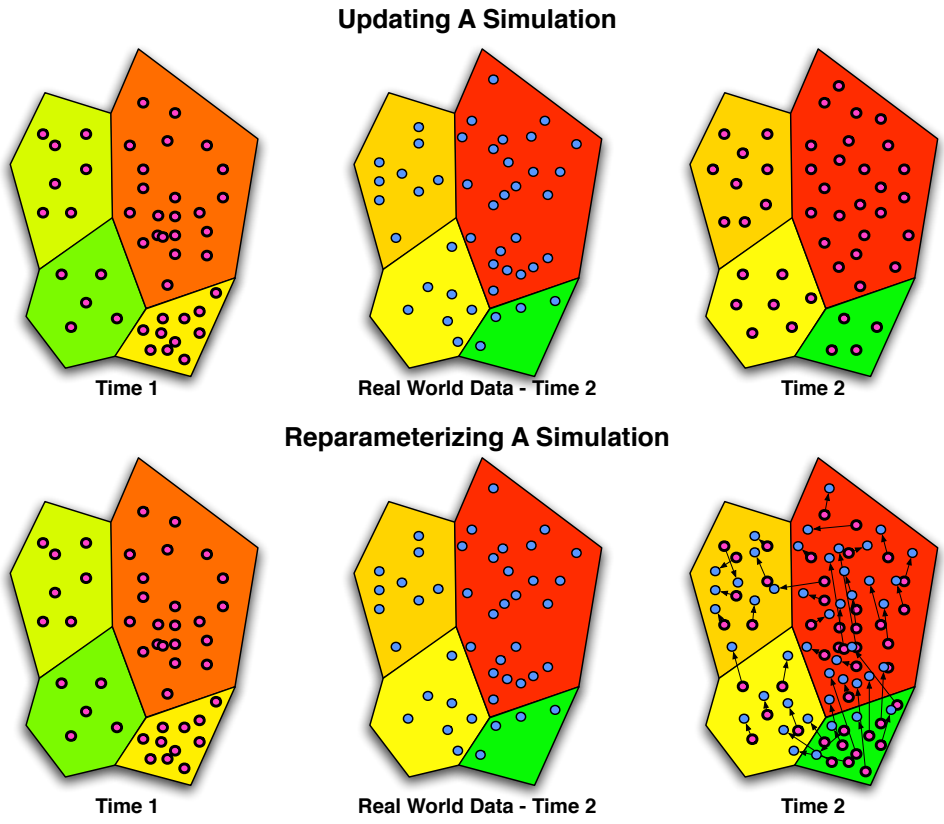

Figure 5.2. A graphical comparison between updating simulations and reparameterizing simulations from streaming data. When reparameterizing a simulation, agent locations and parameters are changed to conform to the streaming data. Updating a simulation causes larger-scale properties, such as numbers of agents in a cell, to be reset. Note that agent locations on an update become approximate.

ease of use, runtime characteristics, etc [9].

5.6 EXPERIMENTAL SETUP AND RESULTS

In order to demonstrate the effectiveness of our approach, we present two experiments. In the initial experiment we examine the differences between updating simulations and re-parameterizing them, exploring the effects of these two techniques in terms of how well the resulting simulations track a given event. In the second experiment we present a case study on updating simulations. We examine the effectiveness of using streaming data to update simulations and demonstrate its effectiveness for tracking and predicting the progress of a simulated event.

For both experiments we adopt Euclidean distance as the metric for measuring the simulation differences. We utilize one run of the WIPER simulation as our simulated event and consider agent population at a fixed set of towers as the vector of locations that we will measure against. We present agent locations at 1 minute intervals and plot the updated simulations based on their Euclidean distance at each interval.

For the updated simulations, agent locations are aggregated to the tower level and simulations are updated every 10 minutes (simulation time) with streaming data. For the re-parameterized simulations, we maintain a 1 for 1 correspondence between agents in the target simulation and agents in the tracking simulations. Every 10 minutes (simulation time) we adjust the parameters of agents in the tracking simulation to correspond to agent parameters in the target simulation.

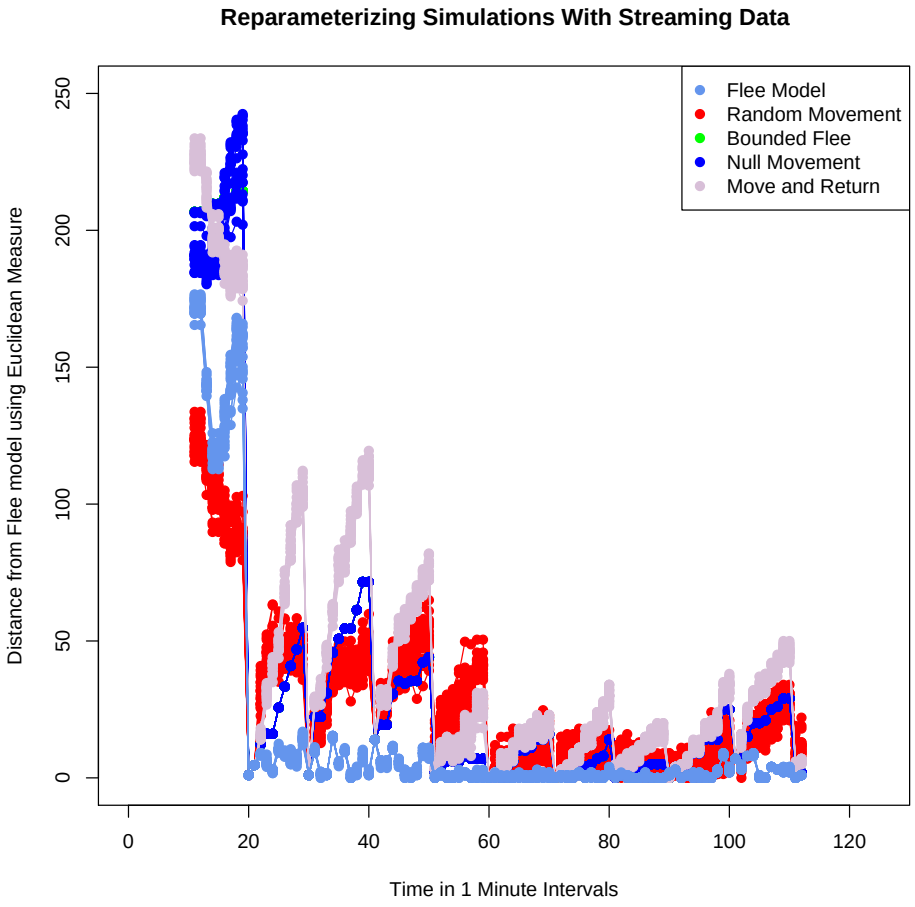

Figure 5.3. Effects of re-parameterizing simulations. Tracking a Flee movement.

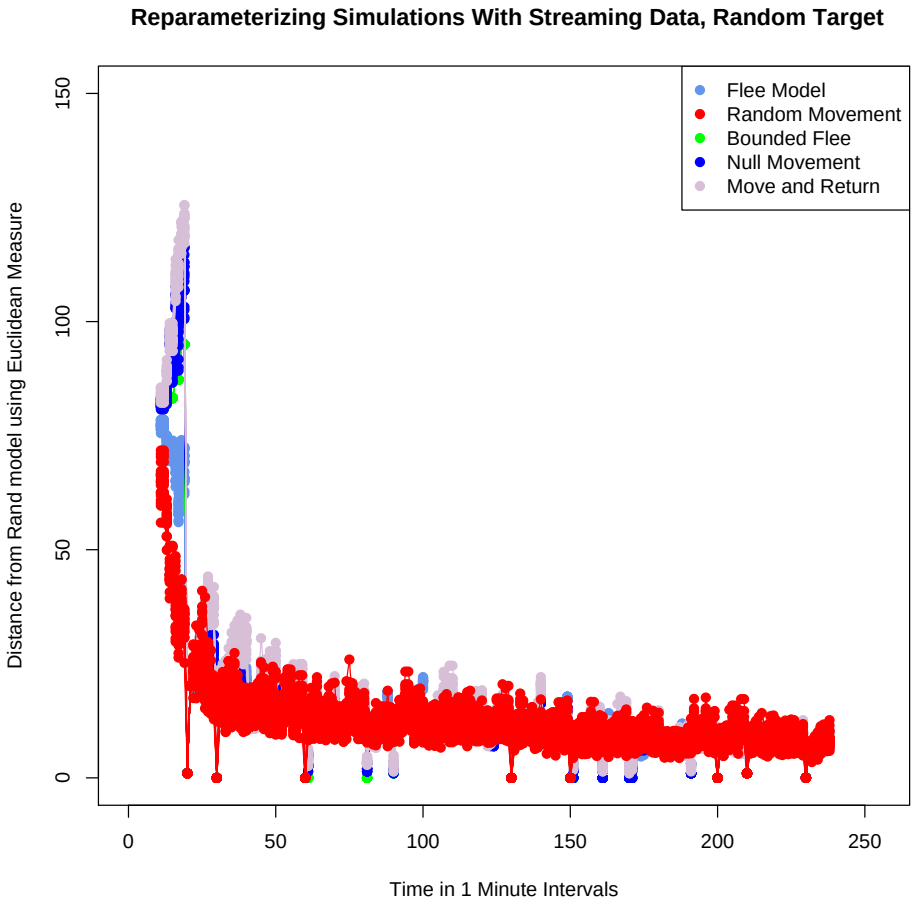

Figure 5.4. Effects of re-parameterizing simulations. Tracking a Random movement.

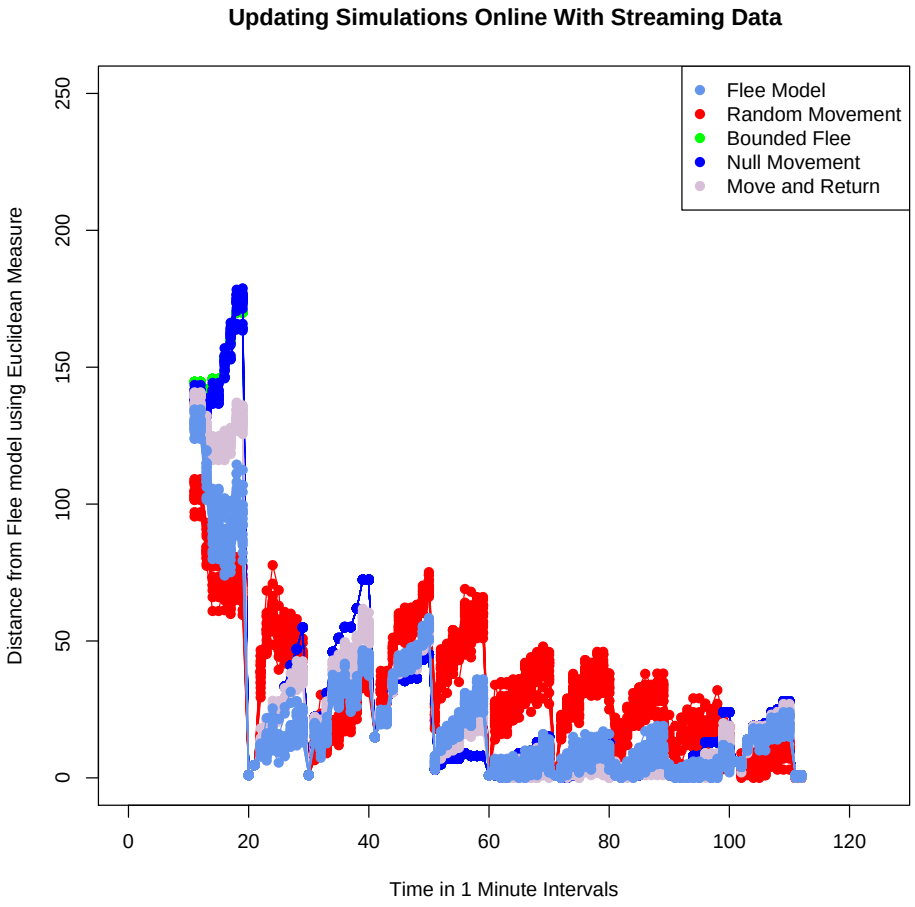

Figure 5.5. Effects of online updating simulations. Tracking a Flee movement.

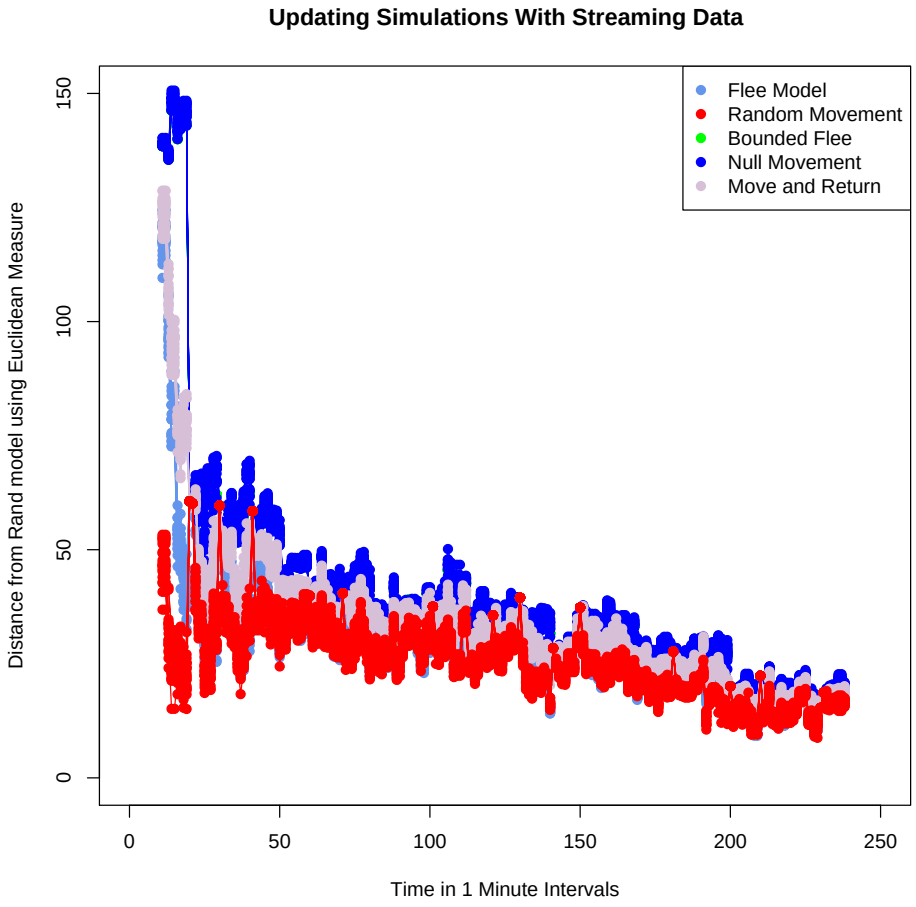

Figure 5.6. Effects of online updating simulations. Tracking a Random movement.

5.6.1 UPDATING AND RE-PARAMETERIZING SIMULATIONS

Results of re-parameterizing simulations are shown in Figures 5.3 and 5.4. Results of updating simulations are shown in Figures 5.5 and 5.6. In each of these Figures, 20 simulations of each movement model type are run. At every 1 minute interval, the simulation outputs a vector with agent locations which is compared against the target simulation, either a Flee movement model or a random movement model.

In Figures 5.7 and 5.8 we compare the effects of updating against reparameterizing. As clearly shown in the Figure, in the initial steps of the simulation, updating generates output that yields a lower distance to the target simulation. As the desire is to minimize the distance between the simulation and target, this is a significant result. Beyond the initial 10 time steps the reparameterized simulations begin to outperform the updated simulations, however, in the context of the WIPER system, the initial 10 time steps are the most important.

The results of this experiment show that the updating approach is a competitive alternative to 1-1 reparameterization. Although we can parameterize and re-parameterize the simulations with precise agent locations, this information does not necessarily improve the performance of the model. Also, the reparameterization process has significant drawbacks, as it is more difficult to manage detailed information about a population of people (in this case precise individual locations) than to maintain aggregate information (number of agents at each tower).

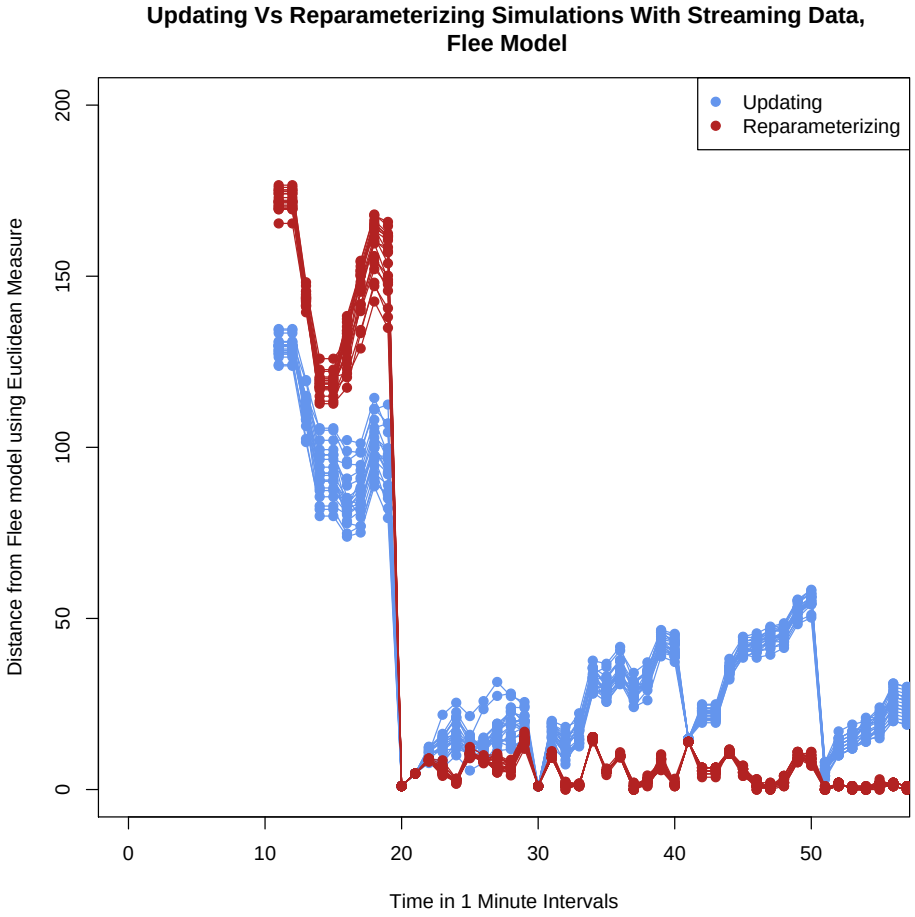

Figure 5.7. Comparison of Updating to Reparameterization, 20 simulations each, using Flee model.

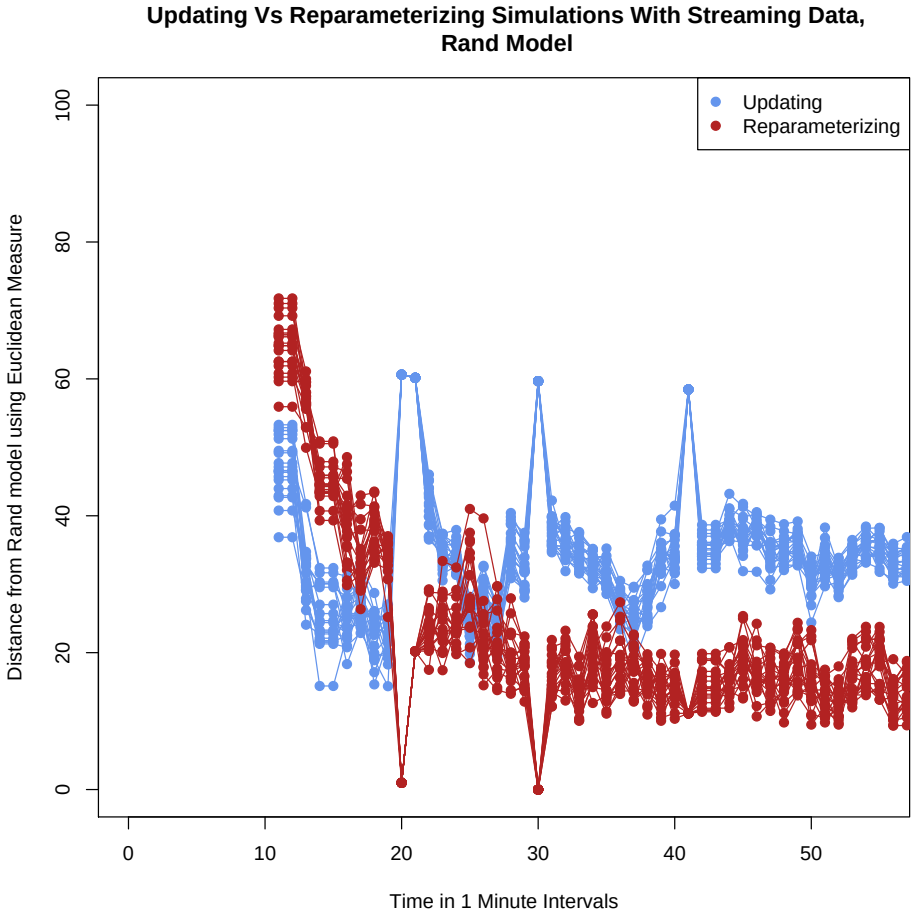

Figure 5.8. Comparison of Updating to Reparameterization, 20 simulations each, using Rand model.

5.6.2 CASE STUDY: EFFECTIVENESS OF UPDATING SIMULATIONS FOR TRACKING CRISIS EVENTS

We present a case study demonstrating the effectiveness of updating simulations for tracking and predicting normal behavior and a crisis event. Simulations generate agent locations at 1 minute intervals. The vector of agent locations is then measured using the Euclidean distance metric from a baseline simulation. In these experiments we visualize two separate baseline simulations, a simulation with random movement and another simulation displaying the Flee behavior. The Flee behavior was chosen to highlight a crisis behavior situation. The random movement scenario was selected as it demonstrates behavior that will be difficult to track and predict.

The results of the case study are presented in Figures 5.9 and 5.10. The simulations are updated at 10 minute intervals with agent location information from the target simulation. The track of the target simulation is not plotted (it would lie along the x-axis). As seen in the Figures, simulations track the Flee movement with varying degrees of success, with successive time steps leading to lower accuracy. Simulations have less success tracking the random movement simulation.

5.7 CONCLUSION

We have presented an approach to updating Agent-Based Models with streaming sensor data, using aggregate information with random variation of agent initialization. Our approach addresses specific needs of a DDDAS application, demonstrating a feasible, scalable approach to simulation updating. We have presented two experiments demonstrating the effectiveness of our technique, one in

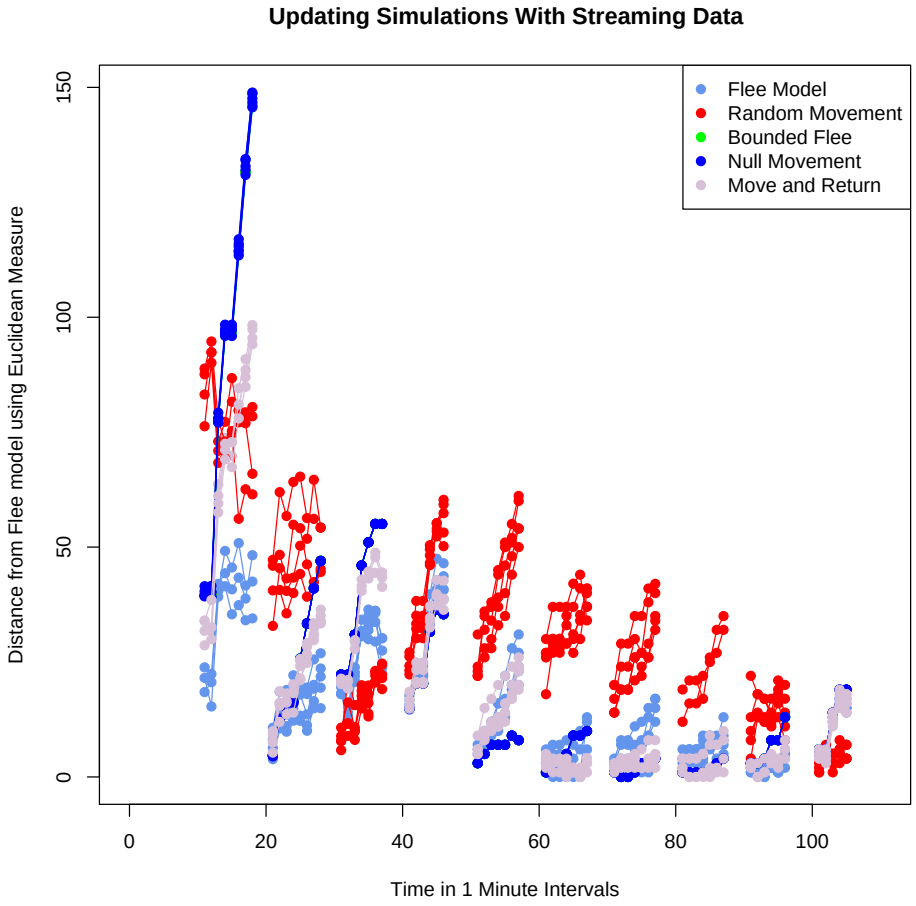

Figure 5.9. Effects of updating simulations. Tracking a Flee movement.

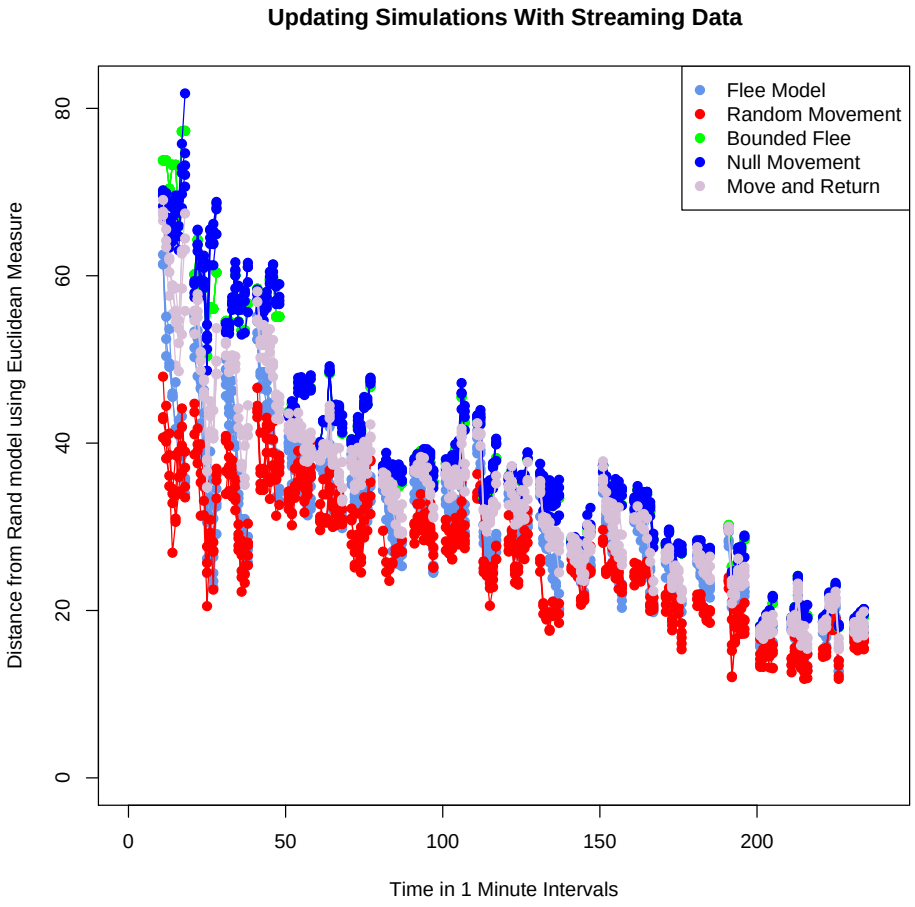

Figure 5.10. Effects of updating simulations. Tracking a Random movement.

isolation and the other comparing it to a more intensive approach.

The results of our experiment comparing updating to reparameterization demonstrate the effectiveness of our approach and provide experimental evidence to support the simulation community's view on the dangers of naive realism. In the context of the WIPER system we have shown that updating simulations is competitive with reparameterization and that the random variations in agent initialization, when spread across the entire population, have a surprisingly small effect on simulation usefulness.

5.8 FUTURE WORK

In order to extend our work and make it more useful to the simulation community at large, we would like to conduct a more thorough evaluation of the continuum of model complexity, examining successively more complex models of agents and agent behaviors and the resulting effectiveness of those models in tracking and predicting the outcomes of real events, both normal activities and crisis situations. Similarly it is interesting to consider the ideal level of aggregation granularity, examining simulations using very large cell sizes to the ideal case where the aggregate area corresponds individual agent locations.

CHAPTER 6

EVALUATION OF MEASUREMENT TECHNIQUES FOR THE VALIDATION OF AGENT-BASED SIMULATIONS AGAINST STREAMING DATA

6.1 ABSTRACT

This Chapter presents a study evaluating the applicability of several different measures to the validation of Agent-Based Modeling simulations against streaming data. We evaluate the various measurements and validation techniques using pedestrian movement simulations used in the WIPER system. These simulations generate output on the calling activity of agents, as well as movement data. Here we consider techniques for online validation of the simulation movement models.

6.2 INTRODUCTION

The WIPER system uses streaming cell phone activity data to detect, track and predict crisis events [90]. The Agent-Based Modeling simulations in the WIPER system are intended to model the movement and cell phone activity of pedestrians. These simulations model crisis events and are intended to be validated in a online fashion against streaming data.

In the WIPER system, ensembles of simulations are created, with each simulation parameterized with a particular crisis scenario and initialized from the

streaming data. When the all of the simulations in the ensemble have finished running, the results are validated against streaming data from the cell phone network. Thus the validation technique must provide a method of discriminating between various simulations. In the context of the WIPER project, this means determining which crisis model is the best fit for the phenomenon detected in the streaming data.

6.3 BACKGROUND

Validation is described as the process of determining whether a given model is an appropriate choice for the phenomenon being modeled. In the model development cycle, validation is normally considered in the context of simulation creation and development, and is done nearly exclusively in an offline fashion. However, the process of selecting an appropriate model for a given phenomenon is precisely what is needed in the dynamic context of the WIPER system.

There exists a large body of work on the topic of simulation validation. A survey of techniques and approaches to offline validation of discrete event simulations can be found in Balci [7]. This work is an essential reference for validation, but many of the techniques are suited to offline validation only, as the interpretation requires human judgement.

This section is divided into three subsections related to the provenance of the techniques we intend to evaluate. The first section deals with canonical offline validation techniques from simulation, the second section presents distance measures and the third section presents work that has been done with online simulations.

6.3.1 OFFLINE SIMULATION VALIDATION

Balci presents a thorough evaluation of techniques for validation of models in the context of model and simulation development [7]. The intent for these techniques was to aid in the validation and verification of simulations prior to deployment. Some techniques mentioned by Balci that are useful to our current discussion are predictive validation (also called input-output correlation) and blackbox testing.

Kennedy and Xiang describe the application of several techniques to the validation of Agent-Based Models [54, 122]. The authors separate techniques into two categories: subjective, which require human interpretation, and objective, for which success criteria can be determined *a priori*. We focus on objective techniques, as the requirements of a DDDAS system make it impossible to place human decision makers "in the loop".

6.3.2 ONLINE SIMULATIONS

Researchers in the area of discrete event simulations recognize the challenges posed to updating simulations online from streaming data [29]. The need for human interpretation is a serious limitation of traditional validation approaches and limits their usefulness in the context of online validation. Simulation researchers have defined a need for online validation but recognize the challenges to the approach. Davis claims that online validation may be unobtainable due to the difficulty in implementing changes to a model in an online scenario. We present a limited solution to this problem by offering multiple models simultaneously and using validation to select among the best, rather than using online validation to drive a search through model space.

It is important to distinguish between the model, the conceptual understanding of factors driving the phenomenon, and the parameters used to initialize the model. Optimization via simulation is a technique that is similar to canonical optimization and seeks to make optimal choices on selecting input parameters while keeping the underlying model the same and uses a simulation in place of the objective function. These techniques are usually grouped by whether they are appropriate for discrete or continuous input spaces [3]. For simulations with continuous input parameters, the author suggests the use of gradient-based methods. For simulations with discrete input parameters, the author presents approaches using random search on the input space.

Pichitlamken and Nelson describe a combined procedure for Optimization via Simulation in [79]. The authors approach has three components: "a global guidance system, a selection-of-the-best procedure and local improvement". For the work presented here, we are most interested in their selection procedure.

6.4 MEASURES

We evaluate the following measures in the context of ranking simulations:

- Euclidean distance
- Manhattan distance
- Chebyshev distance
- Canberra distance
- Binary distance

The distance measures are used to evaluate the output of the WIPER simulations in the context of agent movement. Agents move on a GIS space and agent locations are generalized to the cell tower that they communicate with. The space is tiled with Voronoi cells [114] that represent the coverage area of each cell tower. Empirical data from the cellular service provider aggregates user locations to the cell tower and the WIPER simulations do the same. Thus we can evaluate the distance measures using a well-defined vector of cell towers where the value for each position in the vector is the number of agents at that tower at each time step.

6.4.1 DISTANCE MEASURES

Euclidean distance is the well-known distance metric from Euclidean geometry. The distance measure can be generalized to n dimensions from the common 2 dimensional case. The formula for Euclidean distance in n dimensions is given in Equation 6.1

$$d(\overline{p}, \overline{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{6.1}$$

where

$$\overline{p} = (p_1, p_2, \cdots p_n) \tag{6.2}$$

$$\overline{q} = (q_1, q_2, \cdots q_n) \tag{6.3}$$

Manhattan distance, also known as the taxicab metric, is another metric for

measuring distance, similar to Euclidean distance. The difference is that Manhattan distance is computed by summing the absolute value of the difference of the individual terms, unlike Euclidean distance which squares the difference, sums over all the differences and takes the square root. From a computational perspective Manhattan distance is significantly less costly to calculate than Euclidean distance, as it does not require taking a square root. The formula for Manhattan distance in n dimensions is given in Equation 6.4.

$$d(\overline{p}, \overline{q}) = \sum_{i=1}^{n} |p_i - q_i| \tag{6.4}$$

Chebyshev distance, also called the L_∞ metric, is a distance metric related to Euclidean and Manhattan distances [115]. The formula for the Chebyshev distance is given in Equation 6.5. The Chebyshev distance returns the maximum distance between elements in the position vectors. For this reason the metric seems appropriate to try on the WIPER simulations, as certain models may produce an output vector with one cell having a large variation from the norm.

$$d(\overline{p}, \overline{q}) = \max_i(|p_i - q_i|) = \lim_{k \to \infty} \left(\sum_{i=1}^{n} |p_i - q_i|^k \right)^{1/k} \tag{6.5}$$

The Canberra distance metric is used in situations where elements in the vector are always non-negative. In the case of the WIPER simulations, the output vector is composed of the number of agents in each Voronoi cell, which is always non-negative. The formula for Canberra distance is given in Equation 6.6. As defined, individual elements in the distance calculation could have zero for the numerator

or denominator. Thus in cases where $|p_i| = |q_i|$, the element is omitted from the result.

$$d(\overline{p}, \overline{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i + q_i|} \tag{6.6}$$

The binary distance measure, also called the asymmetric binary measure, compresses a vector of real values into binary values and does a pairwise comparison of elements. Binary distance is calculated by considering the vector as binary, where nonzero elements are considered 1 and zero elements zero, then examining every position in which at least one vector has a nonzero element. The measure is then the ratio of the number of positions where strictly one element is nonzero over the number of positions with nonzero elements. The binary distance seemed well suited to the validation of the WIPER simulation output, as our baseline model, the Flee crisis, tends to output a vector where a Voronoi cell has either a large number of agents or no agents at all. The formula for the binary distance measure is given in Equation 6.7.

$$d(\overline{p}, \overline{q}) = \frac{\sum_{i=1}^{n} p_i \text{ xor } q_i}{\sum_{i=1}^{n} p_i \text{ or } q_i} \tag{6.7}$$

where

$$p_i = \left\{ \begin{array}{l} 0 \text{ if } p_i = 0 \\ 1 \text{ otherwise} \end{array} \right\} \tag{6.8}$$

6.5 EXPERIMENTAL SETUP

In order to evaluate the feasibility of our approach, we present three experiments that demonstrate the effectiveness of the measures on validating agent movement models. The first experiment uses output from a particular run of the WIPER simulation as the synthetic data that will be tested against. This output is considered a "target" simulation. For the second movement model experiment we want to examine the effectiveness of measures in ranking models over all model types. Finally, we present a CMC curve showing the results of using distance measures for model verification.

The purpose of these tests are not to demonstrate the effectiveness of the simulation to match the synthetic data but to demonstrate the ability of the measure to differentiate between simulation movement and activity model types. In a DDDAS system models of human behavior will be created and, according to the traditional model development approach, be validated offline. From this set of pre-validated models the system must be able to select, while the system is running, the model that best matches the data.

For the initial movement model experiment, we examine the effectiveness of the various statistical tests and measures in their ability to rank simulations in their closeness to the baseline simulation. The baseline simulation models a crisis scenario where people are fleeing a disaster. All simulations, including the baseline, are started with 900 agents distributed among 20 Voronoi cells. The distribution of agents to Voronoi cells is fixed over all of the simulations. For each of the 5 movement models, 100 replications of the simulation using different random seeds are run. Our evaluation approach is to examine the effectiveness of each measure in ranking output against the baseline. In this experiment, the desired results will

show that instances of the Flee model are closer to the target simulation than other model types.

The second movement model experiment considers all of the movement models simultaneously. We use the data from the 500 simulation runs and create a matrix of distance values between every pair of values. Each position m_{ij} in the 500x500 matrix is the value of the distance metric between row i and column j. For consistency we present both the upper and lower halves of the matrix, as well as the diagonal, which is always equal to 0. We create this distance matrix for each of the distance metrics we consider. The outcome of the experiment is determined by examining the matrix and determining if the distance metric used shows low distance for simulation runs of the same model and high distance between simulation runs with differing models.

Finally, we use an approach from the data mining community for measuring the effectiveness of algorithms at recognition tasks. The Cumulative Match Characteristic (CMC) curve is a graphical summary of the ability of an approach to correctly identify examples. We generate the CMC curve by measuring a simulation instance against the other 499 instances and ordering the values from low to high. The first example that is of the same model type as the instance is the rank match for that instance. The CMC curve shows, over the 500 probe simulations, what accuracy the measure yields for a given rank.

6.6 RESULTS

Results of using the Euclidean distance metric to measure the differences in agent locations between movement models is shown in Figure 6.1. At the first time interval the Euclidean metric does an excellent job of ranking the Flee model

instances as being the best match to the baseline. All of the Flee model instances have low distance to the baseline and are all lower than any instance of the other models. Interestingly, as the simulations progress, the Euclidean distance of each simulation's output from the Flee model baseline seems to yield good results for classifying the models, as they demonstrate low inter-class distance and high intra-class distance. The exception is the Null and Bounded Flee models. This result is discussed below in the analysis.

Results of using the Manhattan distance metric to measure the differences in agent locations between movement models is shown in Figure 6.2. The Manhattan metric produces similar results to the Euclidean metric, with good results beginning at the first time interval.

Results of using the Chebyshev distance metric to measure the differences in agent locations between movement models is shown in Figure 6.3. As with the other measures in the L family, the Chebyshev distance metric does a good job of differentiating between model types in the early stages of the simulation run and in the late stages.

Results of using the Canberra distance metric to measure the differences in agent locations between movement models is shown in Figure 6.4. The Canberra metric appropriately ranks the Flee model simulations as closest, but as the simulations progress the results appear to be unstable and beyond the 11th sampling interval the Canberra metric fails to return valid values for Flee model simulations. Also, unlike the Euclidean and Manhattan metrics, the Canberra metric displays some overlap in the distance results for different model types.

Results of using the binary distance metric to measure the differences in agent locations between movement models is shown in Figure 6.5. The binary metric is

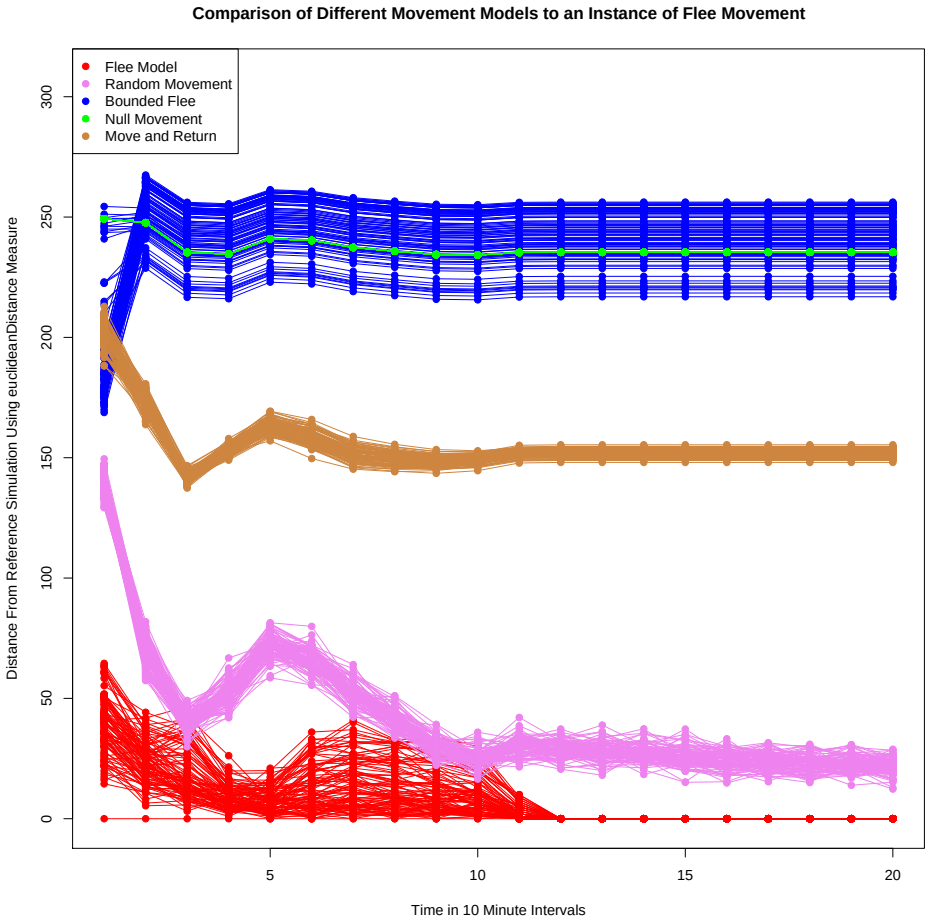

Figure 6.1. Comparing agent movement in various movement models to flee movement using euclidean distance as the measure.

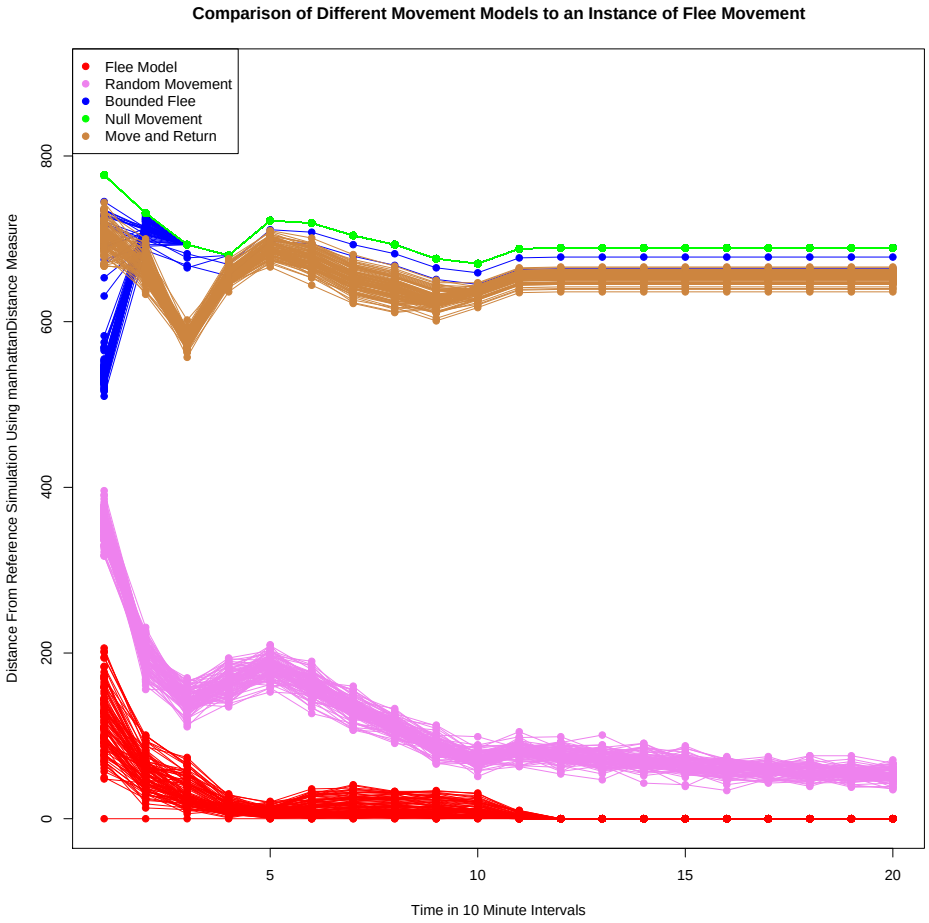

Figure 6.2. Comparing agent movement in various movement models to flee movement using manhattan distance as the measure.

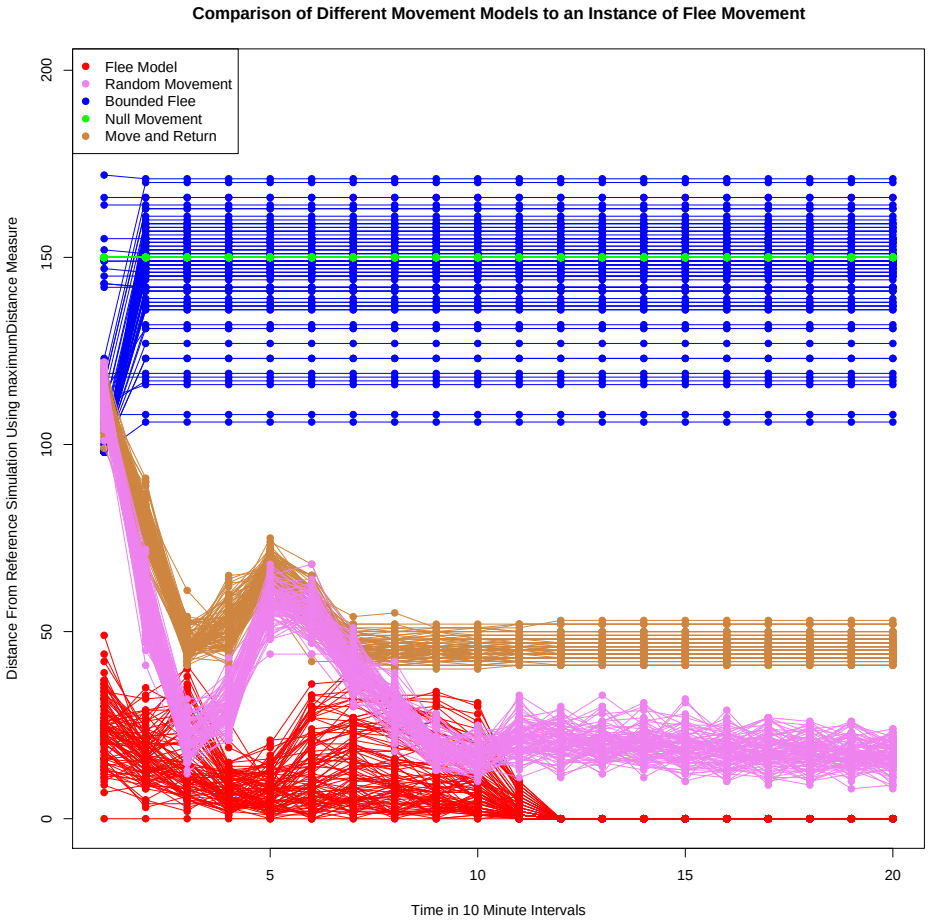

Figure 6.3. Comparing agent movement in various movement models to flee movement using Chebyshev distance as the measure.

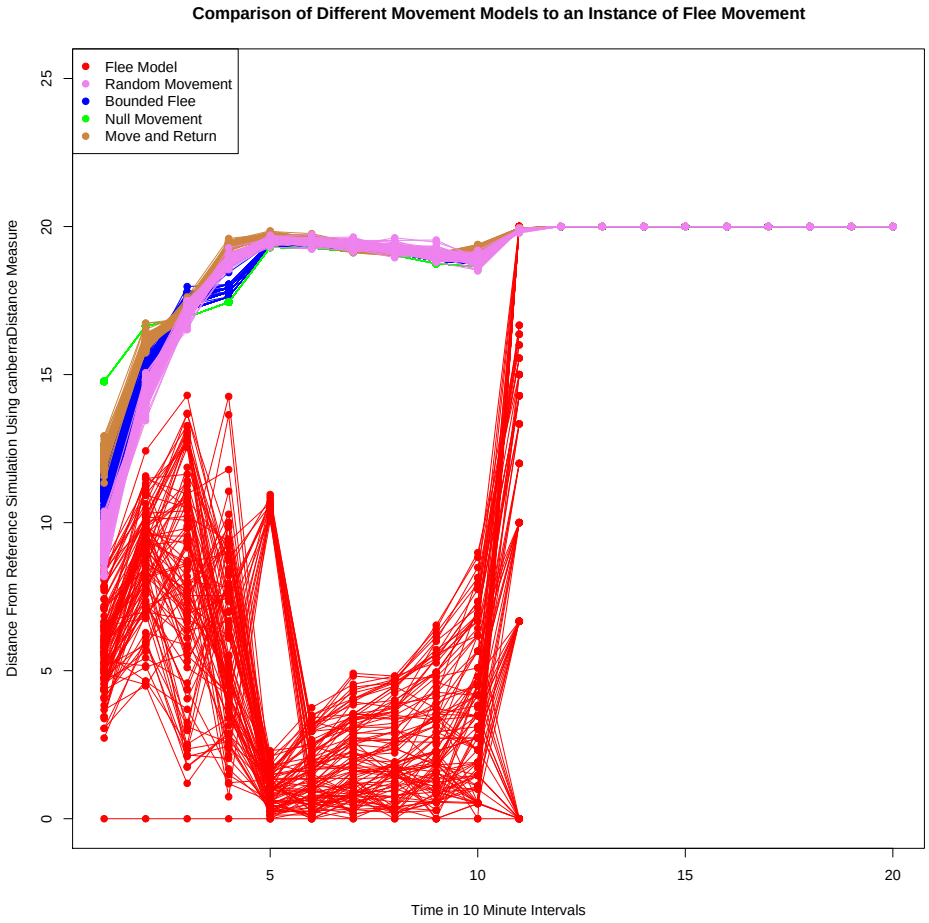

Figure 6.4. Comparing agent movement in various movement models to flee movement using Canberra distance as the measure.

unable to correctly rank the Flee model simulations in the initial time interval. This is problematic considering the context of the WIPER system, as a good measure for the ensembles must provide the ability to discriminate between model types early in the simulation run.

Results of plotting the distance of the output from 500 simulations, 100 runs of each of the 5 movement models, is shown in Figures 6.6, 6.7, 6.8 and 6.9. These results provide a more thorough analysis of the usefulness of the metrics than simply comparing to one run of a simulation. In the ideal scenario the matrix will display low distance between simulations of the same model type (in the figures this would be a bright green square on the diagonal) and high distance when measured against simulations of a different model type (orange or red squares in the remainder of the matrix).

The figures measure the distance in the respective metric of the first time interval of the simulation. The simulations are grouped according to model type in the order, left to right (and top to bottom), Flee Movement, Random Movement, Null Movement, Bounded Flee Movement and Move and Return Movement. Each figure is colored from green to red, with green representing low distance and red high, with the colors scaled to the range of the distance values for the respective metrics.

Figures 6.6, 6.7 and 6.10 present the results for the Euclidean, Manhattan and Chebyshev metrics, respectively. Each of these metrics presents fairly good results in giving simulations of the same model type low distances and simulations with different model types high distance.

The results of the Canberra and binary metrics are less clear. The Canberra metric, Figure 6.8 appears to produce high distance values for Flee model simu-

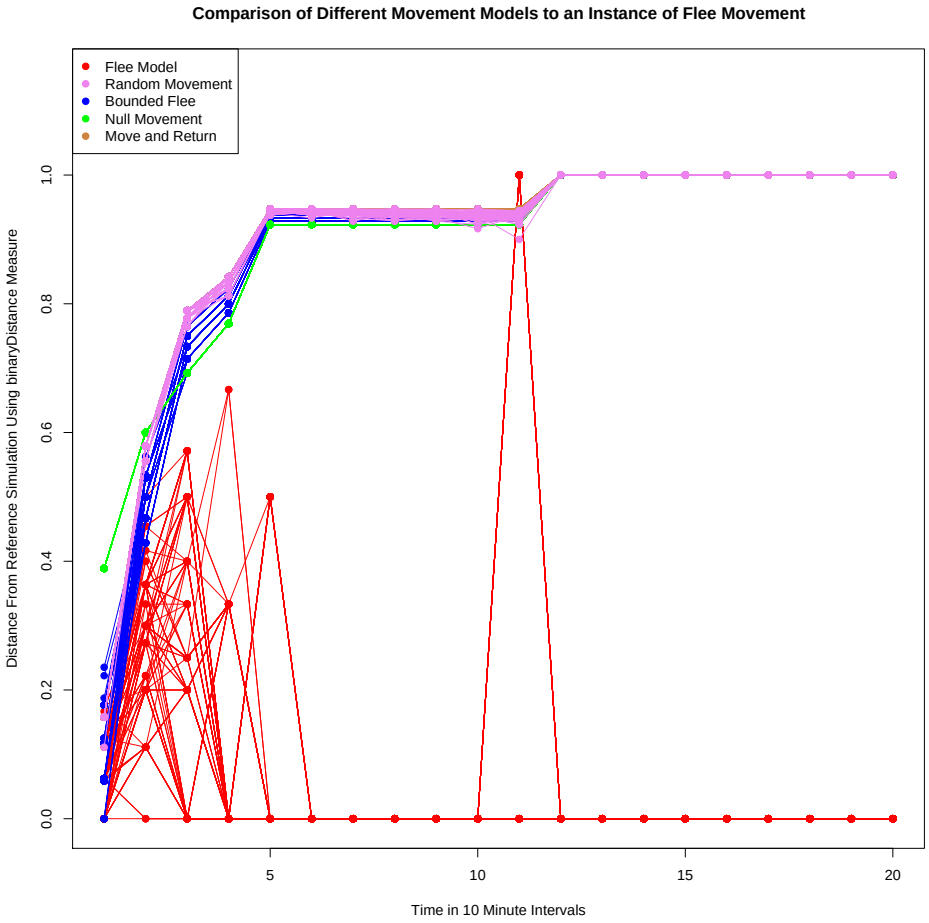

Figure 6.5. Comparing agent movement in various movement models to flee movement using binary distance as the measure.

lations against other Flee model simulations and likewise for the Bounded Flee model. The binary metric, Figure 6.9 is unable to differentiate Random model simulations from Move and Return model simulations and gives high distances when matching Bounded Flee model simulations to other Bounded Flee simulations, similarly for Flee simulations.

6.6.1 USING MEASURES FOR RANKING

In Figure 6.11 we present a CMC curve that demonstrates the effectiveness of the various distance measures at selecting the correct model of a simulation. For the curve we use a probe size of 500 (500 simulation instances) and a gallery size of 5 (since there are 5 distinct model types that we would like to differentiate between). All of the measures in the L family achieve a rank one recognition rate of 100%. Table 6.6.1 displays more in depth information on the match characteristics. Simply registering the rank one rate does not give an indication of the order of the rest of the examples. Instead we must look at the distances to the first true positive and to the first false positive. The separation between these distances gives a relative indication on the range of acceptable values for the recognition threshold.

6.7 CONCLUSIONS

Using a distance metric for model selection has several advantages. An experimental study, like that presented in this Chapter, allows users to calibrate the selection threshold, which makes it possible for the DDDAS to classify a phenomenon based on the distance from the validated model to the event. Alternately, should no model meet the threshold, the system can determine that none of the

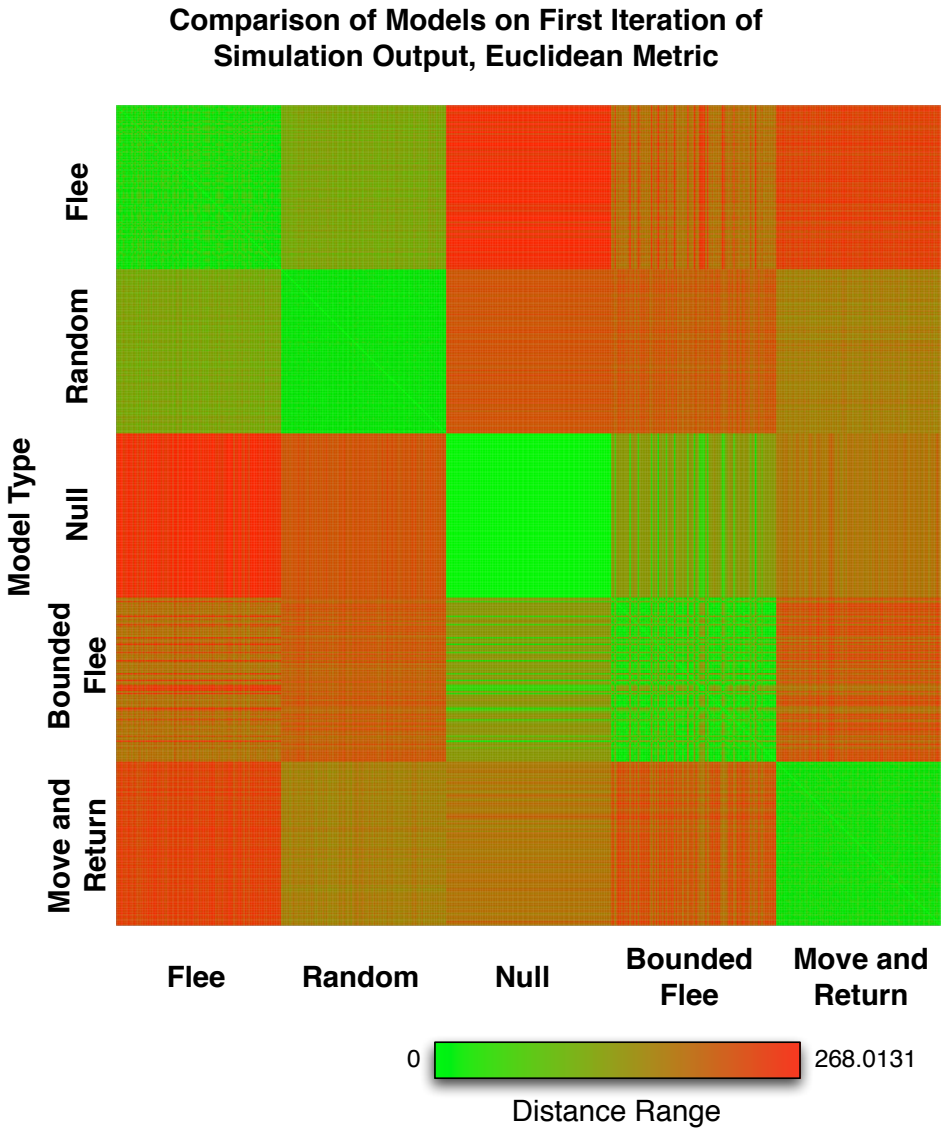

Figure 6.6. Plot of the euclidean distance between simulation output. Simulations are grouped along x- and y-axis according to movement model in the order Flee movement, Random movement, Null movement, Bounded Flee movement and Move and Return movement.

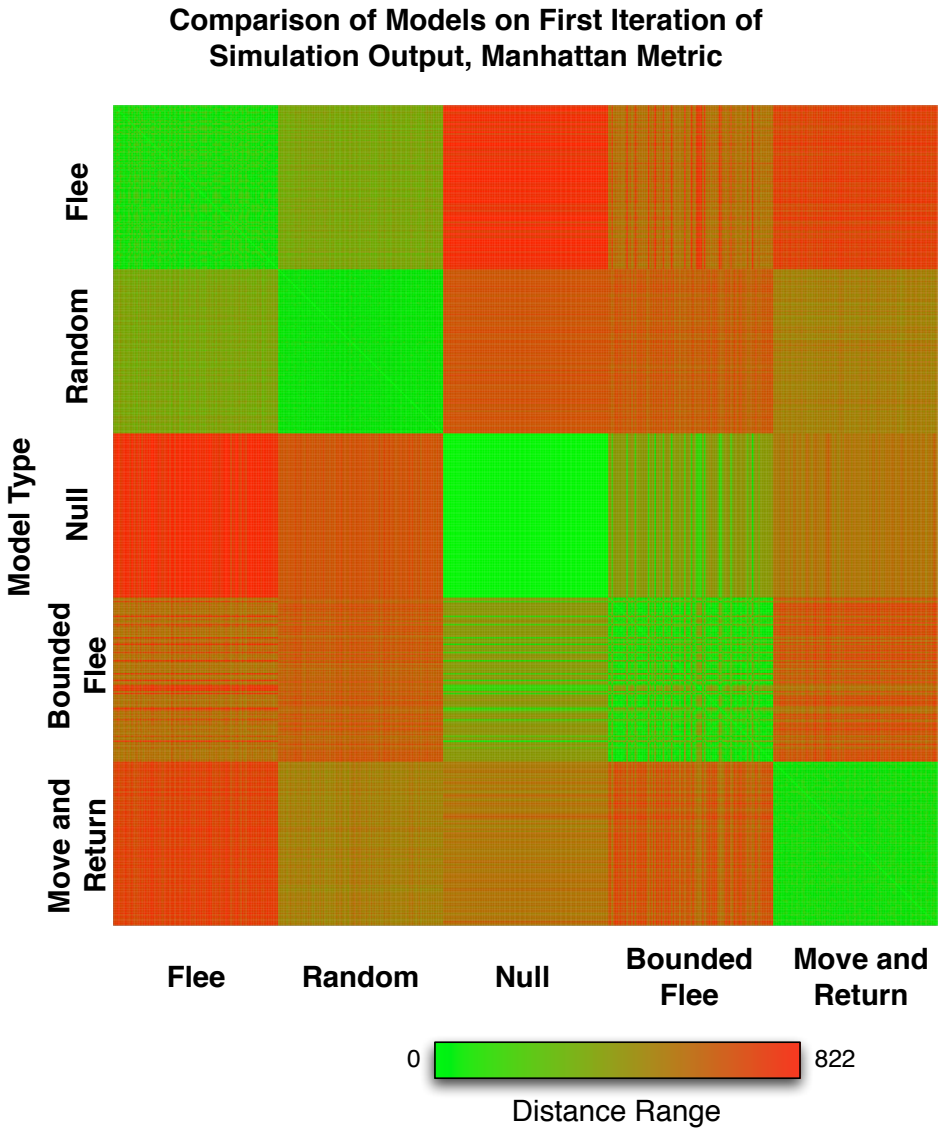

Figure 6.7. Plot of the manhattan distance between simulation output. Simulations are grouped along x- and y-axis according to movement model in the order Flee movement, Random movement, Null movement, Bounded Flee movement and Move and Return movement.

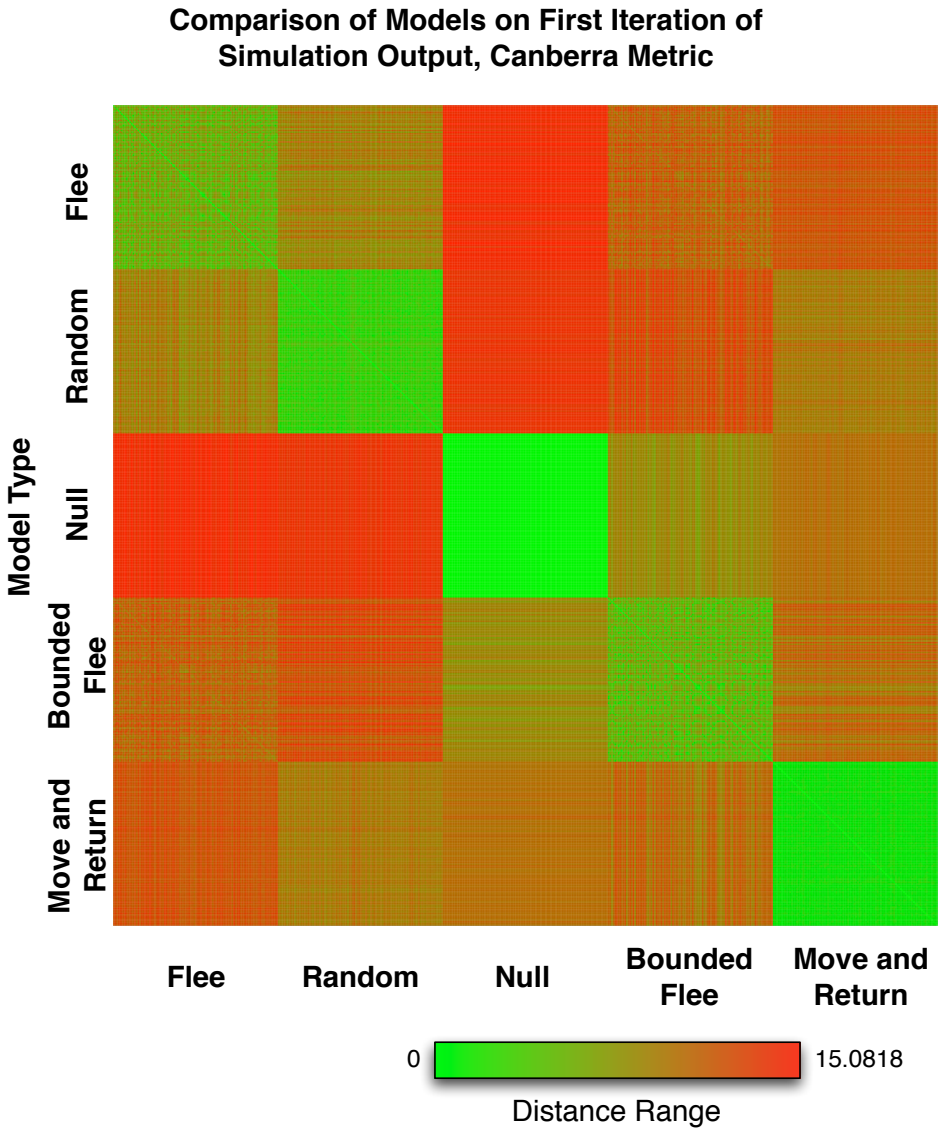

Figure 6.8. Plot of the canberra distance between simulation output. Simulations are grouped along x- and y-axis according to movement model in the order Flee movement, Random movement, Null movement, Bounded Flee movement and Move and Return movement.

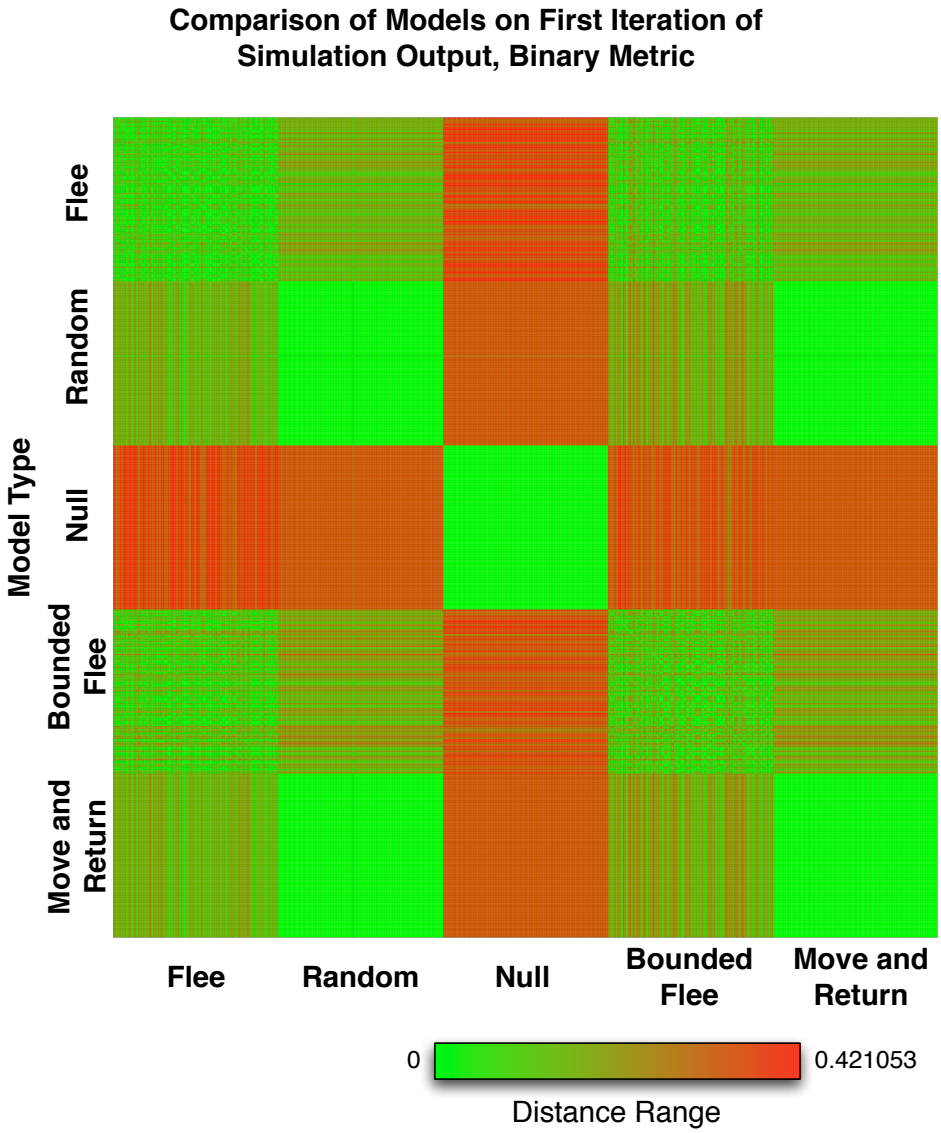

Figure 6.9. Plot of the binary distance between simulation output. Simulations are grouped along x- and y-axis according to movement model in the order Flee Movement, Random movement, Null movement, Bounded Flee movement and Move and Return movement.

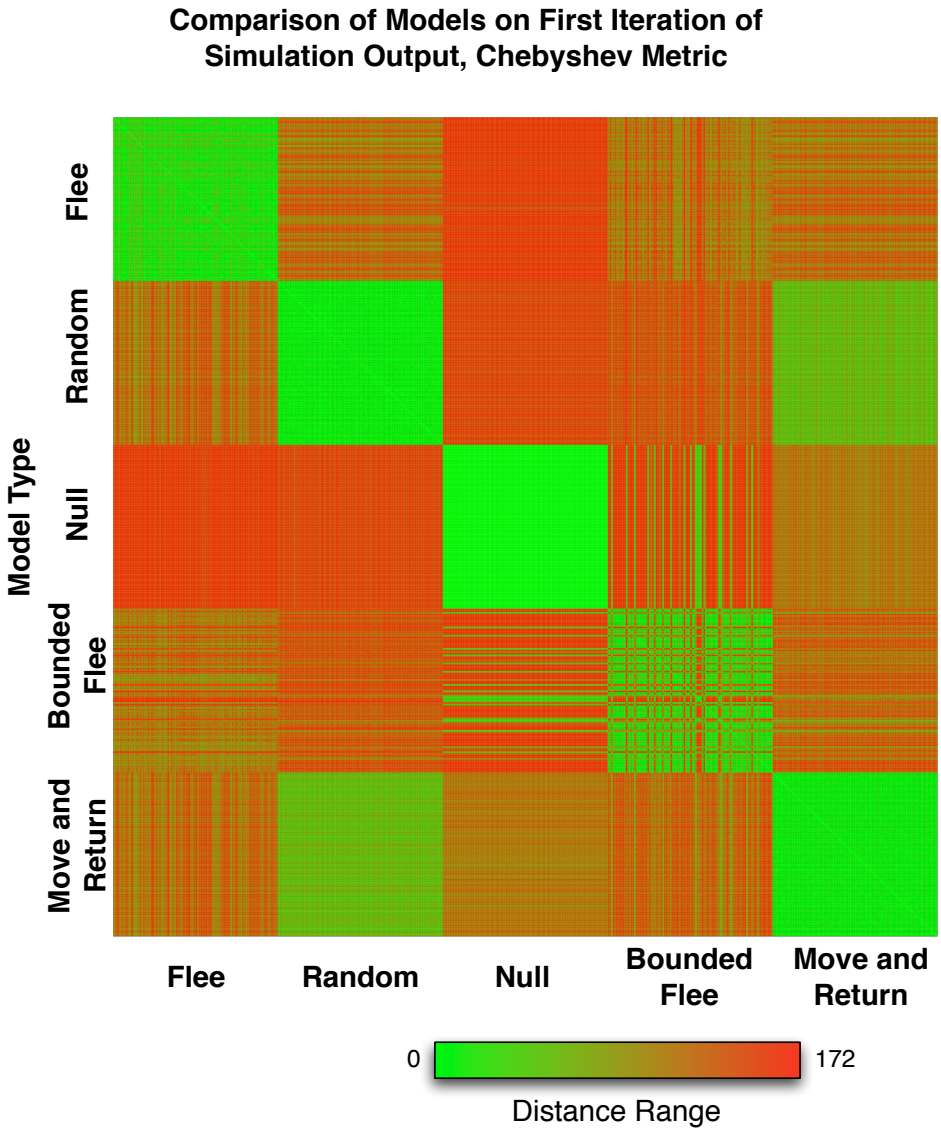

Figure 6.10. Plot of the Chebyshev distance between simulation output. Simulations are grouped along x- and y-axis according to movement model in the order Flee movement, Random movement, Null movement, Bounded Flee movement and Move and Return movement.

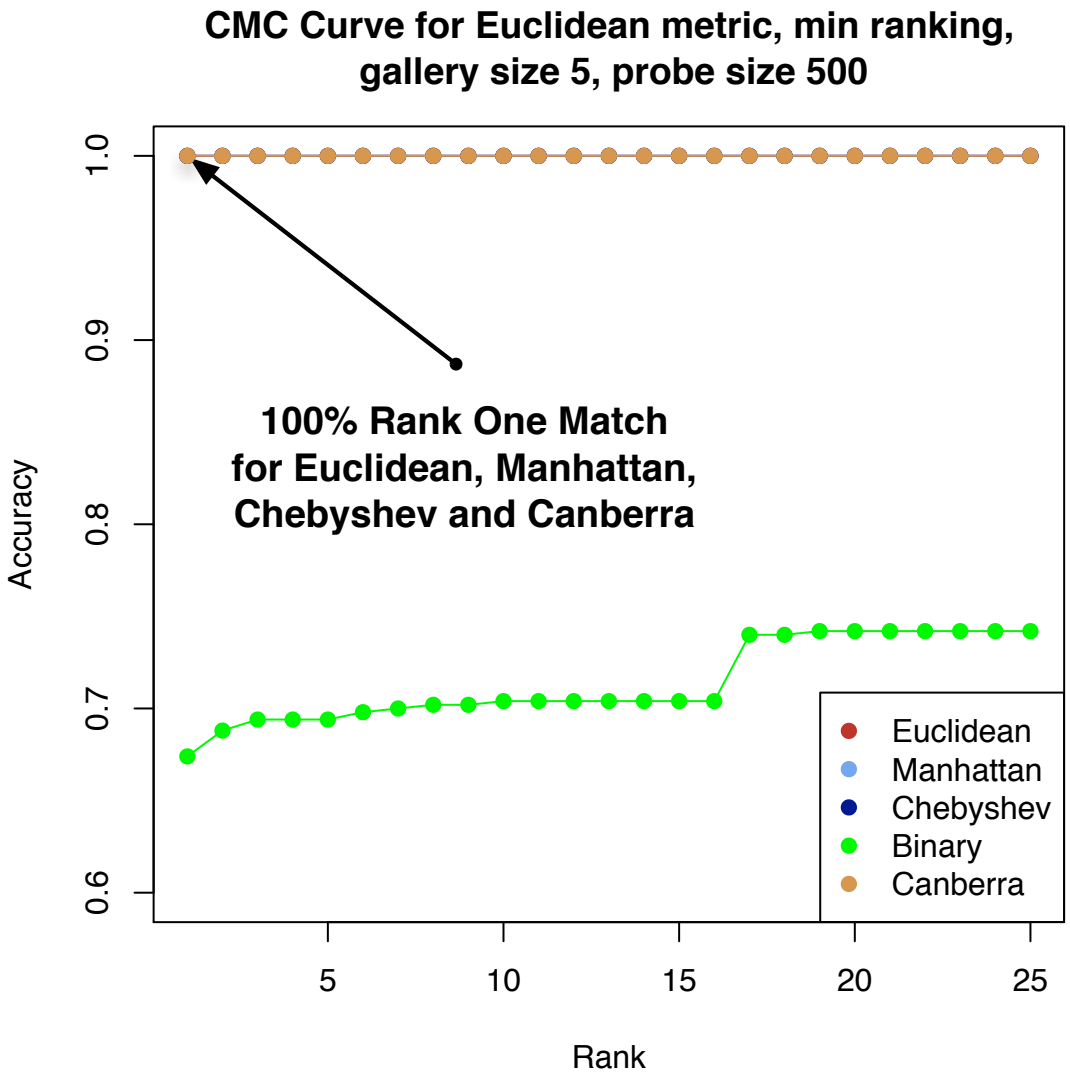

Figure 6.11. CMC Curve displaying Rank 1-25 matches for all 5 distance metrics, gallery size 5, probe size 500.

TABLE 6.1

SUMMARY OF AVERAGE DISTANCES TO FIRST TRUE AND FALSE MATCHES, DEMONSTRATING THE FITNESS OF DISTANCE MEASURES FOR CLASSIFICATION. ALL OF THE MEASURES FROM THE L FAMILY DISPLAY GOOD CHARACTERISTICS.

Measure	**Avg Distance to First True Match**	**Avg Distance to First False Match**	**Avg Distance From False to True**
Manhattan	40.372	294.12	253.748
Euclidean	12.78490	110.2979	97.51297
Chebyshev	6.41	58.604	52.194
Canberra	1.358531	7.132292	5.773761
Binary	0.002380637	0.04760387	0.04522324

models are appropriate. In that case the measure may give a suggestion for the "closest fit" model and provides a scenario for new model creation.

In this Chapter we have presented an evaluation of various tests and measurements for online validation of Agent-Based Models. We have shown that the Euclidean and Manhattan distance metrics work well for validating movement models, however binary and Canberra distance are significantly less useful.

The Manhattan, Euclidean and Chebyshev metrics produce favorable results when used for measuring the similarity of simulations. Under the conditions we have tested, they produce low inter-model distances with high intra-model distance. Any of these metrics is adequate for an application such as the WIPER project.

The Canberra metric is useful under certain circumstances, but the poor performance measuring Flee model simulations against other Flee model simulations make it less than desirable for use in the WIPER project.

The binary metric displays very poor performance in the experiments described here. This is likely a result of the formula for calculating the binary metric, as it discards elements with zero values, in this case Voronoi cells with no agents. The output vector of many of the simulations have towers with no agents, some models in excess of 30% of towers with no agents. This result could be attributed at least partially to the input parameters, but it demonstrates a weakness of the binary metric.

Figures 6.1, 6.2 and 6.7 show the distance metrics failing to differentiate between the Bounded Flee model and the Null Movement model. This result is an artifact of the way movement is measured in the simulations. Since agent locations are aggregated to the level of the Voronoi cell, agent movements below this

resolution do not appear in the output. In the Bounded Flee model, agents move 1000 meters from the crisis and then stop moving. Thus, if the crisis is centered in a Voronoi cell that is approximately 2000 meters across, agents in the crisis cell will not appear to have moved at all.

A caveat concerning the use of simple thresholds for model selection in online validation: In the WIPER project, where mislabeling a crisis event as benign could have dire consequences, it is important to factor in to the system the cost of false negatives. Crisis event models should be weighted so that when confronted with a crisis event, the chance of labeling it as normal behavior is minimized.

6.8 FUTURE WORK

The work in this chapter has focused on measurements for online validation of agent movement models, where validation is selection from among a set of alternatives. Agent behavior in the WIPER simulation is composed of both movement and activity models. It is important for the online validation procedure to treat both movement and activity. In the future we would like to examine measurements for online validation of agent activity models, perhaps in conjunction with work being done to characterize crisis behavior as seen in cell phone activity data [123]. In keeping with our framework, we will need to create not only different input parameters for the activity models, but new models that describe agent behavior under different scenarios (normal activity, crisis, etc). Such work on generating additional agent activity models is currently under way.

6.9 ACKNOWLEDGEMENTS

The graphs in this Chapter were produced with the R statistics and visualization program and the *plotrix* package for R [81][59].

CHAPTER 7

SUMMARY

7.1 INTRODUCTION

This Dissertation has presented the research completed towards the Ph.D. degree. This research proposes a series of novel approaches to problems of using streaming data in a DDDAS. The research addresses open research questions in the DDDAS community, demonstrates the feasibility of our approach through the application in a proof-of-concept system and presents a study of validation techniques for Agent-Based Simulations against streaming data.

Utilizing streaming data in conjunction with simulations has many potential benefits. Simulations parameterized with recent streaming data offer the best possibility of predicting outcomes of events. Similarly, for longer running simulations or longer duration phenomena, streaming data can be used to update the simulations. However, this use of streaming data poses challenges to the simulation modeler and simulation system builder, as the integration of streaming data is not a straightforward challenge. This dissertation has presented our approach to the use of streaming data in a DDDAS system: aggregate updating and validation as model selection.

7.2 RESEARCH CONTEXT

The Dynamic, Data-Driven Application Systems approach offers considerable benefits to researchers studying dynamic, complex systems [95]. However, there are a large set of challenges that go along with this approach, including many open areas of research. The feasibility of the DDDAS approach hinges on the ability of simulations to be tightly integrated into the data collection process. Two aspects of this that we address are the creation and updating of simulations with streaming data, and online validation of models.

7.3 RESEARCH GOALS

The goals of this research project have been as follows:

- To develop the Simulation Prediction System component of the WIPER system
- To create the Agent-Based SImulation for modeling human movement in normal and crisis events for the WIPER system
- Developing and implementing a method to create and update Agent-Based Simulations from streaming sensor data
- To analyze various techniques for validating simulations against a stream of data

These goals address important research topics as described in the NSF DDDAS program solicitation [95] and the NSF Blue Ribbon Panel on Simulation-Based Engineering Science [75]. Those sources describe the challenges posed by the use of streaming sensor data in simulations.

7.4 COMPLETED RESEARCH

This section outlines the research that has been completed towards the Ph.D. degree. The contributions have been organized to correspond with the listing given in Section 7.3.

7.4.1 THE WIPER SYSTEM

The WIPER system is a proof-of-concept DDDAS simulation. It has been developed to test several approaches to solving questions relevant to the DDDAS community, specifically the initialization and online updating of Agent-Based Simulations with streaming sensor data.

The WIPER system is in development by a large interdisciplinary group, including Computer Scientists, Physicists and Sociologists. The work presented here, towards the completion of the Ph.D. degree, is comprised of an academic description of the WIPER system, the design and implementation of the Simulation Prediction System and its attendant applicability to the DDDAS community.

7.4.2 THE AGENT-BASED SIMULATION FOR THE WIPER PROJECT

In order to test our approach to initializing and updating simulations, it was necessary to first create an example simulation. The WIPER simulation models pedestrians carrying cell phones under several normal and crisis scenarios. The development of the simulation was aided by the creation of a taxonomy of crisis scenarios which categorizes crises by the representative agent behaviors. The simulation has been designed using the Pattern Oriented Modeling approach of Grimm and Railsbeck, which places emphasis on generating recognizable patterns through the composition of behavioral primitives [45]. The simulations are built

using the Agent-Based Modeling approach and have been designed iteratively using agile programming techniques such as unit testing, refactoring and design patterns.

7.4.3 CREATING AND UPDATING AGENT-BASED SIMULATIONS FROM STREAMING DATA

The creation and updating of simulations from streaming sensor data is a core requirement of DDDAS systems [26]. In the WIPER project, we have addressed this issue by demonstrating an aggregate approach, where detailed sensor information is intentionally aggregated and simulations induce an amount of random variation on inputs. We have presented experiments demonstrating the effectiveness of the approach, compared to no updating and compared to a more sophisticated method that maintains a 1-1 agent to human representation, and a case study involving the implementation of the approach in the WIPER project.

7.4.4 MEASUREMENTS FOR ONLINE VALIDATION OF AGENT-BASED SIMULATIONS FROM STREAMING DATA

As mentioned in [54], the validation of Agent-Based simulations is a relatively new area, and there has been even less work done on online validation of these simulations. The contributions of this chapter are a first step towards a comprehensive understanding of how to conduct online validation of Agent-Based simulations, including framing the problem as a model selection process, the evaluation of several measurement techniques and a case study of using online validation in the WIPER project.

7.5 SUMMARY

Dynamic, Data-Driven Application Systems represent an exciting new approach that will advance the field of simulation. However, many open questions about the feasibility of DDDAS concepts remain. This research has addressed some of the challenges in DDDAS systems, creating and updating simulations from streaming sensor data, techniques for validating simulations against streaming data and provided insight into the design of DDDAS systems through the development of a proof of concept DDDAS, the WIPER project. The knowledge gained from this research can be leveraged to develop more advanced DDDAS systems.

APPENDIX A

DATA PREPARATION AND IMPLEMENTATION OF HIGH PERFORMANCE DATA WAREHOUSE

A.1 OVERVIEW

In this section we describe the steps taken to create the data warehouse for the WIPER project. The data warehouse was created in accordance with standard principles of software engineering and database design, while balancing the need for performance with flexibility of data representation.

A.2 INTRODUCTION

A data warehouse is a repository of historical data, often kept in a relational database and used for knowledge extraction and data mining. The creation and maintenance of a data warehouse is a challenging task and often requires breaking relational database normal forms, as data warehouses are optimized for particular types of access patterns and are usually expected to be static repositories of data.

A.3 INTERNATIONALIZATION ISSUES

Our initial data dumps included several international characters which unfortunately were not recognized as Unicode by PostGreSQL. The easiest fix was to

use sed to replace all of the offending characters in the data files with ASCII characters. It appears that the original encoding for the characters was a non-Unicode compliant standard.

A.4 SCHEMA

The schema for the data went through several iterations before a reasonable format was found. We began by using text fields for all of the attributes, then analyzing the fields to determine the smallest data type able to represent the attribute. Later, we decided to recode the attributes offline, outside of the database in order to both anonymize the data and reduce the size of the attribute representation inside the database.

A.5 DATA CLEANSING SCRIPTS

A.5.1 INTERNATIONAL CHARACTER REMOVAL

We used this sed command to remove the international characters from the file and format all null fields in the same format:

```
cat $1 | sed s//i/g | sed s//o/g | sed s//n/g | sed s//N/g > ${1}.cleansed
```

A.5.2 NULL FIELDS AND VARYING COLUMN WIDTHS

There appear to be at least two methods of representing `NULL` data in the database dumps. In some circumstances, missing data was represented as the following character string: `##########`, in other fields, missing data was simply an empty field. In order to use the `\copy` command to load the data, we had

to unify these methods so that only one character sequence was recognized as representing `NULL`.

In the Call Duration field, it was found that at least one call had a duration of 33780 (not sure of units, seconds?). This was surprising and broke our assumption of using a 2-byte integer for storing duration, as it was only able to hold values up to 32768.

A.6 LOADING

Loading the data into the database was a nontrivial task. The table schema for the different tables needed to be adjusted several times to accommodate service types and phone numbers that were larger than expected. Initially, it was believed that `call_to` numbers would all be standard phone numbers, however this was not a valid assumption. There were a significant number of calls to special service numbers with 5 digit phone numbers, as well as several international calls that were larger than the data type originally meant to hold the phone number. Also, the service type varied widely, from the standard service which was only 9 characters wide, to other services that had names ranging from 18 to 24 characters.

Our approach for loading the data was to find a schema that was permissive enough to allow all the data to be loaded into the database. Once this schema was found, we will use database operations to shrink the schema to the minimum that will hold all of the data fields. On several occasions we started with fixed width character fields, only to find that there were a few records that violated the character length constraint, often requiring double or more storage space. For these fields we chose to use variable length characters in order to save space in the database. The initial schema is shown in Table A.1. Once the data has been

Column	Type
call_from	text
call_to	text
service	text
dest_code	var_char(16)
time_init	timestamp
duration	int
cost	double
unknown	char(4)

TABLE A.1

INITIAL SCHEMA FOR DETAILED TABLE

imported into the database, we will run some performance tests to determine the best data storage format.

A.7 ANALYSIS OF COLUMN ATTRIBUTES

The first step in setting the draft schema was to analyze the size and space requirements for all relevant columns. Table A.2 shows the results of a study of the detailed table. The first 5 columns are all character fields in the database. The duration field is an integer in the table, the study being done to see if we could reduce the number of bits necessary to represent the values.

The same tests were run on the two week aggregate data tables. Those results are presented in Table A.7.

Armed with the knowledge of the maximum and average field size for rows

Column Name	Max Width	Avg Width	Std. Dev.
call_from	18	9.00	0.34
call_to	30	8.26	1.90
service	24	9.38	1.60
dest_code	5	3.16	0.55
unknown	3	2.55	0.50
duration	45436	74.0	202.

TABLE A.2

ANALYSIS OF THE COLUMNS OF THE DETAILED TABLE

Column Name	Max Width	Avg Width	Std. Dev.
call_from_id	18	8.996	0.1856
call_to_id	30	8.248	1.942
dest_code	5	2.668	1.218
call_type	4	1.015	0

TABLE A.3

ANALYSIS OF THE COLUMNS OF THE TWO-WEEK TABLE.

in the table, we experimented with different implementations to determine the performance tradeoffs inherent in our decisions on data representation.

APPENDIX B

MAPPING OF TEMPORAL CELL PHONE ACTIVITY DATA

B.1 ABSTRACT

This section covers the steps taken to create the maps and animations showing cell phone activity over time, plotted on a map of the region of interest. The location information is taken from the originating tower of the cell phone call.

B.2 GOALS

The final goal of this effort is to create animations and maps of call activity showing the spatial behaviors of calls over time. We seek to present this information accurately using a GIS, with political boundaries (state, provincial, city) and postal codes.

B.3 FIRST STEPS

We began by examining the data. The data was transfered as flat files in ASCII format with the fields seperated by semicolons. The data, as usual, required several levels of cleaning and formatting before it could be used in a mapping applications. First, it was necessary to extract just the useful information: latitude, longitude and cell number from the dataset. This was relatively straightforward, but there were several instances where towers had missing or incomplete values.

Any tower with a mangled latitude or longitude was excluded from study. Next, the latitude and longitude values needed to be corrected. The values from the cellular service provider were given as a string such as "N0435959". Also, there is a convention for referencing all longitude values by positive or negative values, but the service provider left all of the values as positive values and only provided an 'E' or 'W' to indicate east or west longitude. This was an issue because the prime meridian runs through our area of interest.

B.3.1 LATITUDE AND LONGITUDE

Modern GIS systems use latitude and longitude values that are formatted as a floating point number, with up to two digits before the decimal place representing the degrees and an arbitrary number of digits after the decimal. However, in legacy systems, latitude and longitude values are represented as 6 digits, two for each of Degrees, Minutes, Seconds. In our data file, the latitude and longitude values had no decimal place and there was no indication as to which format they were in. At first glance, we mapped the values as if they were in degrees with floating point precision. The resulting image is shown in Figure B.1. After looking at the image, we were convinced that something was wrong with the data. However, we searched on towers by postal code and determined that indeed, the towers that we needed were in fact represented. After some deliberation, the following idea was proposed: since they data providers had used E and W longitude to represent the data, was it not possible that they had given the data in DD'MM'SS instead of decimal format? A quick change to the perl script determined this to be true. Once the latitude and longitude values were correctly formatted, the resulting image looked more like what we expected. This is shown in Figure B.2

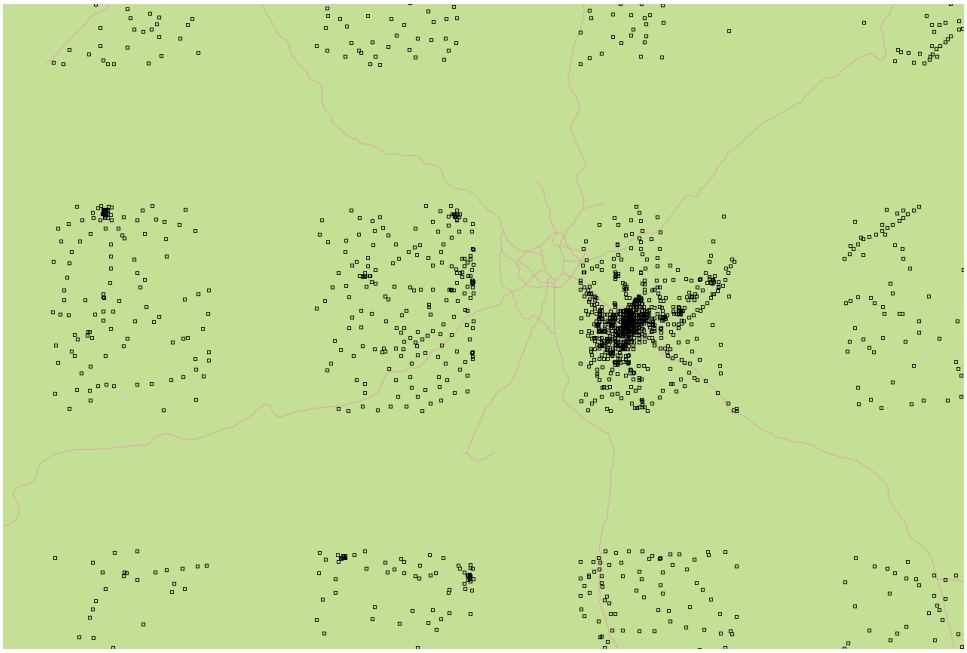

Figure B.1. First attempt to map towers.

B.3.2 DATATYPE CONVERSION

In order to visualize the data layers in the GIS system, it was necessary to convert the list of values from a flat ASCII file to the ESRI Shapefile format. There exist several open source tools to aid in the conversion and manipulation of GIS data. For this project so far we have used FWTools (v. 1.0.0b2), shapelib (v 1.2.8) and gen2shp (v 0.3.1). FWTools includes the GDAL (raster) and OGR (vector) tools, which are used for manipulating and converting GIS files. The program gen2shp takes an ASCII flat file with location information (points, lines, polygons) and converts it to the ESRI Shapefile format. This requires the shapelib libraries for various utilities.

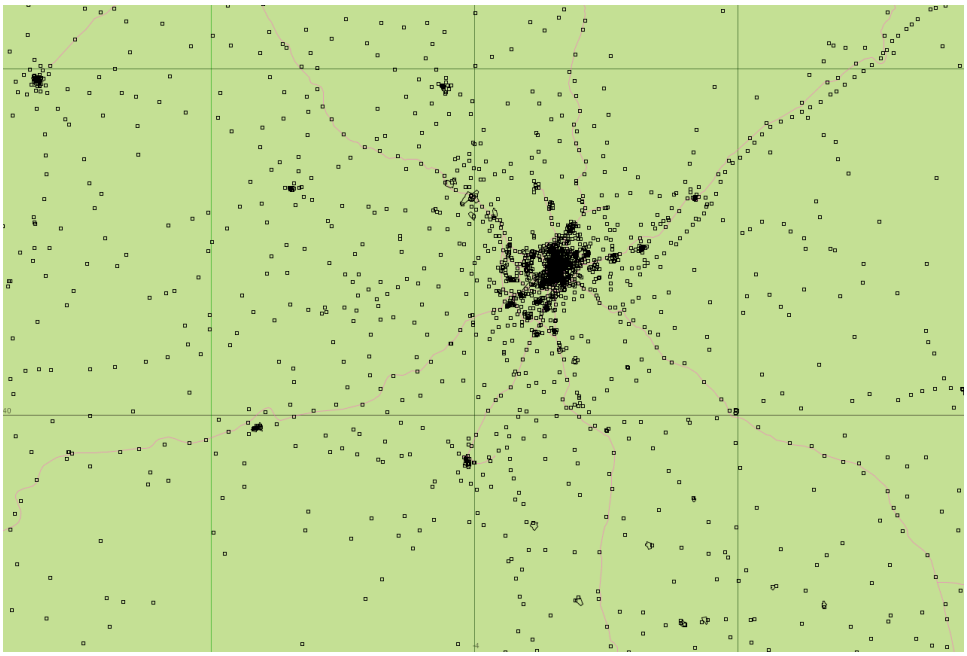

Figure B.2. Second attempt to map towers using corrected latitude and longitude values.

B.4 CREATION OF VORONOI DIAGRAM FOR SPATIAL ACTIVITY REPRESENTATION

Our goal in all of this work is to build a visualization of the spatial activity in the cellular network. With the foundation, the cell tower map, finished we needed to find a way to divide up the space around each tower such that the activity at each tower could be easily represented on the GIS map. The easiest solution is to build a Voronoi diagram on top of the map of cell towers. A Voronoi diagram is a decomposition of a metric space around a given set of points such that for a given point p and associated cell S_p, all points within S_p are closer to p than any other initial point [114].

Given our set of starting towers, we used Grass GIS to build a Voronoi diagram around all of the points. However, the first attempts were without success due to the existence of three "towers" at each location. In the tower location file each line is given a different tower id, however, these are not "towers". In reality, each physical tower hosts no less than three antennas, pointing in different directions. In order to build the Voronoi diagram we had to correct the problem by allowing only one "tower" at each location. The resulting image is shown in Figure B.3.

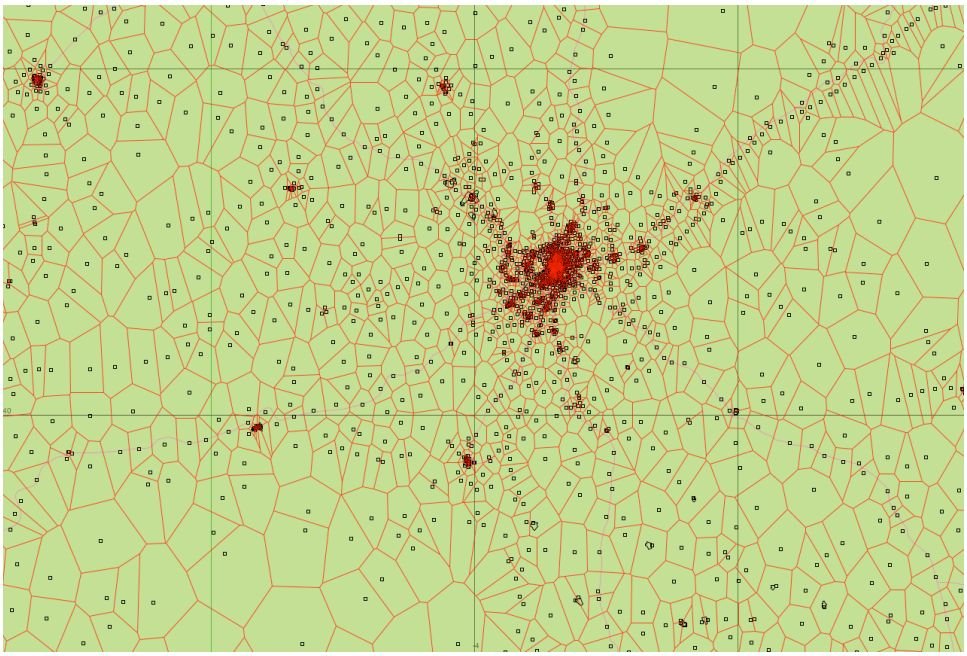

Figure B.3. Towers with an associated voronoi diagram.

APPENDIX C

CLUSTERING ANALYSIS OF SOCIAL NETWORKS

C.1 ABSTRACT

This Chapter summarizes the clustering analysis of the social network of cellular phone users. Our goal is to have a baseline understanding of the characteristics of the network so that we may create an anomaly detection system that runs in real time as part of the WIPER system. This project is focused on looking at the call activity network among prepaid customers for one cellular service provider for a 15-day period. The network is extracted from customer and activity data and is analyzed using standard tools from network analysis. Analysis of the link distribution of the network demonstrates that it falls in the "scale-free" regime, confirming previous results in this area. In light of insight gained from the first rounds of network analysis, we examine the top 10 clusters in the network in high detail. This analysis is currently in a very preliminary stage. When complete, it will be used to identity anomalies related to emergency events. [1]

C.2 INTRODUCTION

The analysis of social networks is a compelling and challenging area. So much of our modern life is measured and cataloged: cell phone communications, emails,

[1]This work was conducted under the supervision of Dr. Nitesh Chawla as part of the course requirements for his Data Mining class.

landline phone conversations (recently popularized by the NSA) and social networking websites where people compete to collect "friends". With this data collection comes an unprecedented opportunity to analyze social networks to see if they conform with expected theory.

This work examines a small social network, that of the calls made and received by prepaid customers for a mid-sized cellular service provider. Our work demonstrates that the observed social network conforms to some previous results in social networks and also shows the limitations of examining a social network from a single modality.

C.3 BACKGROUND: NETWORK MEASURES

There exist a large number of network measures that are important to the social network and network analysis communities. These measures are used to describe structural properties of the network and can be used to characterize networks. Some of the measures, such as the degree distribution, are relatively straightforward to calculate, whereas others have a high computational complexity, such as the characteristic path length. Additionally, some network measures, such as diameter, require that the graph be connected. When the graph is composed of more than one component, such calculations yield meaningless results.

C.3.1 DEGREE DISTRIBUTION

One of the easiest measures to calculate is the degree distribution. This measure allows researchers to visualize whether the network falls into a normal, log-normal, exponential or scale-free distribution. Often networks fall into either an exponential distribution (similar to how incomes fall into a Pareto/Zipf distribu-

tion) or a scale-free distribution (as seen in the world-wide web)[2, 10, 109].

The degree distribution is computed by counting the degree of each vertex (usually the in-degree for directed graphs) and then plotting a histogram of the results. For networks that display exponential or scale-free distributions, it is necessary to plot the histogram on a log-log plot. This dampens the effects of the overwhelming number of nodes with low degree and makes visualization of the tail possible. Exponential distributions tend to have a region of linear negative slope (when plotted on log scale) with an abrupt cutoff and scale-free distributions have a similar region of linear negative slope but with a heavy tail.

C.3.2 DIAMETER, CHARACTERISTIC PATH LENGTH, GLOBAL HARMONIC MEAN DISTANCE

Diameter, Characteristic Path Length and Global Harmonic Mean Distance are three measures that are related, attempting to measure similar phenomena[2, 63, 109]. They measure, in different ways, how "closely" connected a network is, specifically, how far (usually in terms of hops) nodes are apart. These measures are often used to describe the amount of time a message takes to propagate across a network.

The diameter measure is simply the longest of all of the short paths from every vertex to every other vertex in the network. Computing the diameter of a graph involves running an all-pairs shortest path calculation on the network, and then finding the largest value.

The diameter can be sensitive to large values and may not be representative of the average distance that messages need to travel in the network. In graphs like the lollipop graph, one long path overwhelms the calculation, but represents

only the worst-case scenario and does not convey the average number of hops that an average message must travel. Watts and Strogatz developed the characteristic path length in response to this need. The characteristic path length, L, is computed by taking the median of the means of all shortest paths in the network. For each node in the network, take the mean of the length of all shortest paths to other nodes in the network. Then find the median of all of these values. There exist approximation algorithms of L, but most implementations use the naive approach, which is very time consuming. Additionally, Watts and Strogatz "normalize" L by dividing the median value by the value for a lattice of the same size. For a more detailed discussion, please see [109].

The global harmonic mean distance, D_{global} is another measure that attempts to represent the speed at which information may flow across the network. The equation for calculating D_{global} is shown in Equation C.1. In order to calculate D_{global}, add the multiplicative inverse of the distances between each pair of vertices, then divide $N(N-1)$ by that value. The global harmonic mean distance is also normalized in the same manner as L, by dividing the result of Equation C.1 by the D_{global} of a lattice of similar size. There are several features that make D_{global} attractive over L. First, D_{global} can be calculated on networks that are not connected. Second, the computation is significantly less complex than that of L.

$$D_{global}(G) = \frac{N(N-1)}{\sum_{i,j \in G} \frac{1}{d_{i,j}}} \tag{C.1}$$

C.3.3 CLUSTERING, LOCAL HARMONIC MEAN DISTANCE

Complementing the global measures described above, there are several local measures that are used to characterize networks. These measures, the clustering

coefficient and local harmonic mean distance, describe the average level of interconnectivity of each neighborhood around each node in the network. This measure can be used to characterize the "processing" power of the network, as problems can be shared between people in a neighborhood.

The clustering coefficient is a measure of the amount of interconnections among a node's neighbors, averaged over the entire graph[109]. The equation for the calculation of the clustering coefficient C is shown in Equation C.2. For each vertex i in the network, we examine the subgraph, G_i, of i's neighbors, excluding i. For every pair of vertices j, k in G_i, if there is an edge between them, add 1, otherwise add 0. Once all possible combinations of j, k have been examined, divide the result by the amount of possible edges for G_i, which is $\frac{n_i(n_i-1)}{2}$, where n is the number of edges in G. Then repeat the process over the entire graph. The clustering coefficient, like L, is normalized by dividing by the maximum possible connections among the neighbors.

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\sum_{j,k\in G_i}\epsilon_{j,k}}{\frac{n_i(n_i-1)}{2}} \text{ where } \epsilon_{j,k} = \left\{ \begin{array}{l} 1 \text{ for } e_{j,k} \in G_i \\ 0 \text{ for } e_{j,k} \notin G_i \end{array} \right\} \tag{C.2}$$

The equation for calculating local harmonic mean distance, D_{local}, is shown in Equation C.3 [63]. In this case, G_i is the subgraph of nodes connected to i, including i, and $d_{j,k}$ is the distance between nodes j and k. The calculation is very similar to the procedure for calculating C, and the description is omitted for brevity.

$$D_{local}(G) = \frac{1}{N}\sum_{i=1}^{N}\frac{N(N-1)}{\sum_{j,k\in G_i}\frac{1}{d_{j,k}}} \tag{C.3}$$

In terms of computational complexity, C and D_{local} are similar and are rela-

tively fast on real-world networks, due to their sparse nature. In the code I wrote to calculate local and global harmonic mean distance, major portions of the algorithmic code is shared between the local and global implementations due to their similarity. In practice we have found that for the D_{local} calculation, the naive approach to all-pairs shortest path outperforms Djikstra's algorithm. However, for D_{global}, where the all-pairs shortest path calculation occurs on the entire network, the Djikstra's algorithm approach is vastly superior.

C.4 SOFTWARE FOR NETWORK ANALYSIS AND VISUALIZATION

For this version of the project we used Jung [76] for calculating the network measures. Jung is a Java API for network visualization and calculation. The Jung API is very easy to use, but is not designed for the size of network we need to study. The analysis approach tries to marry object-oriented design with network algorithms, which is not efficient in terms of time or space complexity, especially when paired with the overhead from Java. For example, using the 64-bit Sun HotSpot JVM for Opteron, we ran the network analysis and filtering tool with our 777,304 node network. The program used a 7GB heap size (on a machine with 8GB of main memory), took 72 hours to run (32 hours reading in the network) and eventually ran out of memory before completing any analysis. For the final analysis we will be doing the subgraph extraction in custom built code.

We will use Pajek for network visualization, as it can scale to larger networks than Jung[13]. Pajek has been used by others in my research group to visualize networks on the order of 35,000 nodes and is more than adequate for examining and visualizing our top 10 clusters.

C.4.1 CONTRIBUTED SOFTWARE

For this project, two useful software tools were developed. First, a Java program using the Jung API was written to calculate the network metrics. This software was useful but proved to be too memory intensive for the final network. Instead we chose to develop our own network package, tentatively called "VEIN".

VEIN [89] is a network manipulation and analysis package written in Ruby. The package includes base classes for Graphs, Nodes and Edges, as well as utilities to seperate the network into components and output a visualization of the graph using the GraphViz[38] package. The VEIN package offers better performance in both memory usage and speed compared to Jung, but lacks many of the network metrics that Jung provides. We have begun to implement some of the network measures in Ruby using the GNU Scientific Library and Ruby-GSL bridge, which provides native Ruby access to GSL methods and data structures with excellent runtime performance.

C.5 DATA SOURCE

The data set that we examine contains approximately 22 million phone call records, with 4.9 million unique customers and several million users from outside the system. Cell contracts fall into two categories: prepaid and postpaid. Postpaid contracts stipulate that users pay a monthly fee for a certain amount of call time. Prepaid users pay in advance for the amount of minutes they use but do not need to pay a recurring fee. We have data on the aggregate call activity of all users in the system for 15 two-week periods from April 2004 to June 2005. Table C.1 shows the breakdown of prepaid and postpaid users in the dataset.

Out of this collection of two-week period, we have selected one period, June

Contract Type	Number of Users
Prepaid, Active	135,767
Prepaid, Expired	977,844
Postpaid, Active	4,423,236
Postpaid, Expired	503,214

TABLE C.1

BREAKDOWN OF USERS BY CONTRACT TYPE AND ACTIVITY STATUS.

1-15, 2005, for intense scrutiny. In order to limit the size of the dataset to a reasonable level, the study will be restricted to prepaid users.

The data was transfered to us in flat ASCII text files. Each file contains a line for all calls from a given number, to another number for the duration of the 5-day or 15-day period. (Initially the cellular service provider gave us 15-day datasets, then in October 2005 they switched to 5-day datasets.) Each line contains the following information: the originating number, the destination number, the number of calls for the time period, the total duration of all calls, the service type (voice, SMS, multimedia message, WAP) and the amount billed.

In addition to the call activity datasets, we have a dataset with customer billing information. These tables contain information on active users and users who have left the system. In order to select the prepaid users, we must cross-reference the two-week datasets with a customer information table.

C.6 NETWORK

Initially, we generated a network of 424,810 nodes (each corresponding to a user) from 5 days of data. For the next iteration, we used a 15-day dataset from June 1-15, 2005, to correspond with our user information which was also collected in June 2005. The resulting network contains 777,304 nodes.

The network was constructed in two steps. First, a list of the prepaid users was taken from the user data file and used to create a hash. Next, using the hash, we step through the network activity file and extract any calls that were made to or from a prepaid number.

Starting with the raw network, in order to reduce the size of the network, we drop all nodes that have no incident edges, which filters out the nodes that we are not interested in. Then we compute the degree distribution of the network, in order to compare it to the initial network. Finally, we extract the 10 largest components of the network in order to examine them more carefully.

C.7 RESULTS

For the 5-day dataset, we noticed some interesting phenomena. First, once the graph had been created, a rudimentary analysis showed that the graph consisted of multiple connected components. A connected component is a subgraph where all vertices are reachable from any other vertex. In the graph we generated, there were 135780 connected components. This number corresponds exactly to the number of prepaid users, so we believe that there are no calls between prepaid users in this 5 day period. As evidence we have computed the diameter of 90,000 of the diameter and the largest reported diameter is 2, which lends evidence to the theory that there are no calls between prepaid users.

In order to rule out effects from the short time period, we examined the 15-day period. For this period, the graph consists of 777,304 nodes and 28,117 components.

C.7.1 GRAPH MEASURES

The size of both the 5-day and 15-day datasets make it prohibitively expensive to calculate network measures, where they would yield useful values. Some measures, such as characteristic path length and diameter, are undefined on networks that are not connected. The harmonic mean distance measures yield meaningful results when networks are not connected, but the time and space complexity of the algorithms made it impossible to calculate these values for the entire network.

As mentioned above, we were able to calculate certain network measures on the 10 largest components from the 15-day network. These results are shown in Table C.2. The components from the 5-day network were trivially small and were not analyzed. Table C.2 shows only the network measures that have been implemented and tested in VEIN. It is important to point out that these components display very low global harmonic mean distance and diameter values. When we look at the graph images later, the reason for these low values will be apparent.

C.7.2 DEGREE DISTRIBUTION

The degree distribution is simply the number of adjacent edges for each node. By looking at a histogram of the degrees of all nodes in the network, we can detect whether the graph falls into any of the known distributions[2, 10]. Natural phenomena often fall into one of the following distributions: normal, log-normal, exponential, scale-free. With network data, we often see exponential and scale-

Component	*Number of Vertices*	*Diameter*	*Global HMD*
1	65	4.0	0.0502
2	61	2.0	0.0269
3	51	5.0	0.0772
4	46	2.0	0.0744
5	46	2.0	0.0599
6	45	4.0	0.0633
7	43	2.0	0.0714
8	41	3.0	0.0337
9	40	4.0	0.0915
10	39	2.0	0.0624

TABLE C.2

IMPLEMENTED NETWORK MEASURES FOR THE TOP 10 COMPONENTS.

free distributions. The first visualization of the degree distribution is shown in Figure C.1. This histogram is plotted with equal-width bins on untransformed x- and y-axes. The size of the first bin overwhelms the scale of the graph and makes it difficult to see if there are any users with more than 12 calls in the 5 day period.

A standard practice in the study of exponential and scale-free phenomena is to visualize the distribution on a log-log plot. In Figure C.2, we show the data plotted on a log-log scale, dropping the users that made no calls. This plot provides a much more accurate description of the call activity in the system. Shown on the plot is a linear fit line with slope $= -3.8733$. A linear regression on the log data is a common method of detecting exponential and scale-free distributions. If a linear fit can be made at high degree of significance, it indicates that the distribution is exponential or scale-free. The difference between the two depends on whether the tail of the distribution falls off (exponential) or is heavy (scale-free).

We also add the plots from the 15-day data. The histogram is shown in Figure C.3 and the log-log plot with linear fit line is shown in Figure C.4.

C.7.3 TOP 10 COMPONENTS

Out of 28,117 components in the 15-day network, we extracted the 10 largest for further analysis. This are shown in Figures C.5 - C.14. It is interesting to note that these components all display a similar, star-like structure. I am currently attempting to analyze more of the network components to confirm that this structure is not anomalous. Analysis of the initial network yielded several components with structure similar to Component 10 (Figure C.14), leading me to conclude that the central vertex was actually a service (911, 411, etc) and not a user. However, the prevalence of the star-like structure has made me consider that

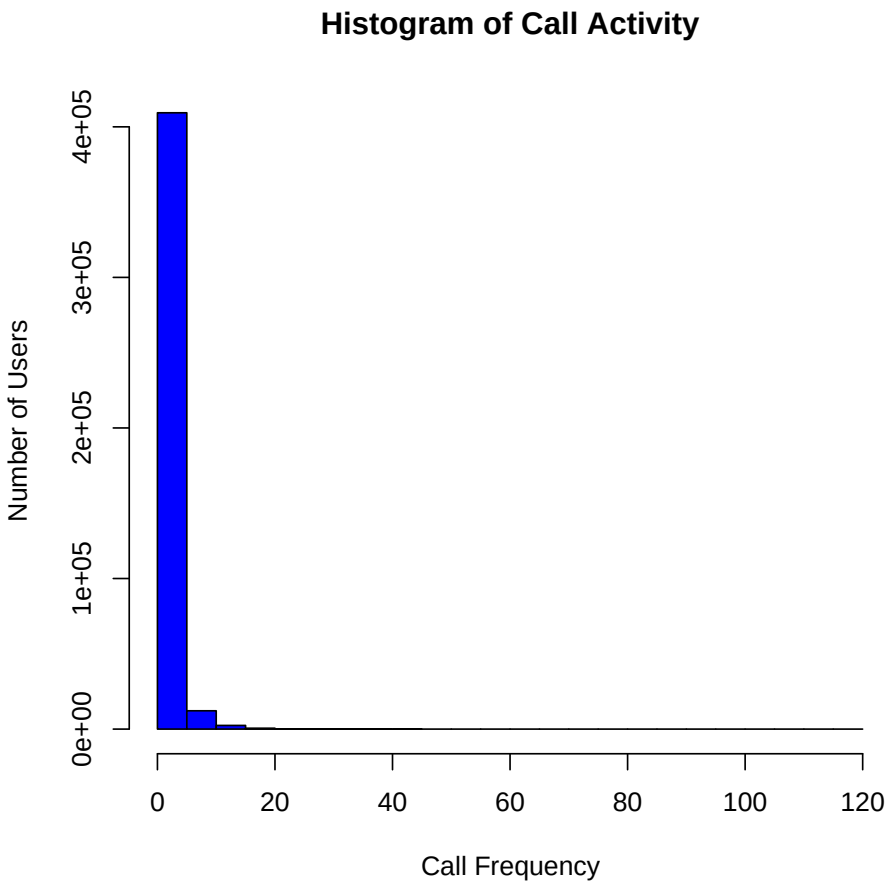

Figure C.1. Histogram of the call activity for the 5 day period.

the network of prepaid users may look like that due to out-of-band communication. Components 3 (Figure C.7), 6 (Figure C.10) and 7 (Figure C.11) display some more interesting structure, but further analysis is required.

C.8 CONCLUSIONS

At this stage, we have extracted the network we want from the call data, but we have not yet been able to fully analyze the network. Analysis at this level

exceeds the capabilities of existing tools and requires us to step back and develop our own tools for extracting and analyzing the network components.

Although the graph is composed of a high number of components, the first analysis seems to indicate that the call frequency confirms existing results in social network analysis. Further study is needed to determine whether the star-like pattern is anomalous or is a phenomenon associated with prepaid phone customers. In order to determine this we would like to examine the call network of postpaid/ service contract users and examine the differences. It may be that prepaid users treat their cell phone as secondary communication device, for caller id, receiving SMS messages, voice mail and make calls only rarely. This type of behavior would mean that much of the users' communication with members of their social network is out-of-band, and thus not visible to us.

C.9 CONTRIBUTIONS

We will briefly summarize the contributions of this project regarding the VEIN network analysis library for Ruby. VEIN was written to be a lean, clean-room implementation of the network analysis tools and measures listed above, along with the ability to read and write files in the Pajek network format and to output visualizations in png format using GraphViz.

VEIN has a working implementation of the Floyd-Warshall All-Pairs Shortest Path algorithm[111], will correctly split a network into its components, can calculated diameter and global harmonic mean distance (normalized) and is well documented. VEIN includes RDoc documentation.

C.10 FUTURE WORK

We would like to complete the analysis of the prepaid users network to compare it with prior work in the field. Of special interest is the lack of connectivity in the network and the sparse, hub-like structures. Hopefully by interpreting our results in the context of other single modality social networks (postpaid users, email correspondence, etc) we can suggest reasons for these results.

C.11 ACKNOWLEDGMENTS

Special thanks to Professor Nitesh Chawla for his advice and guidance in the development of this work. The research began as a course project for his Data Mining class.

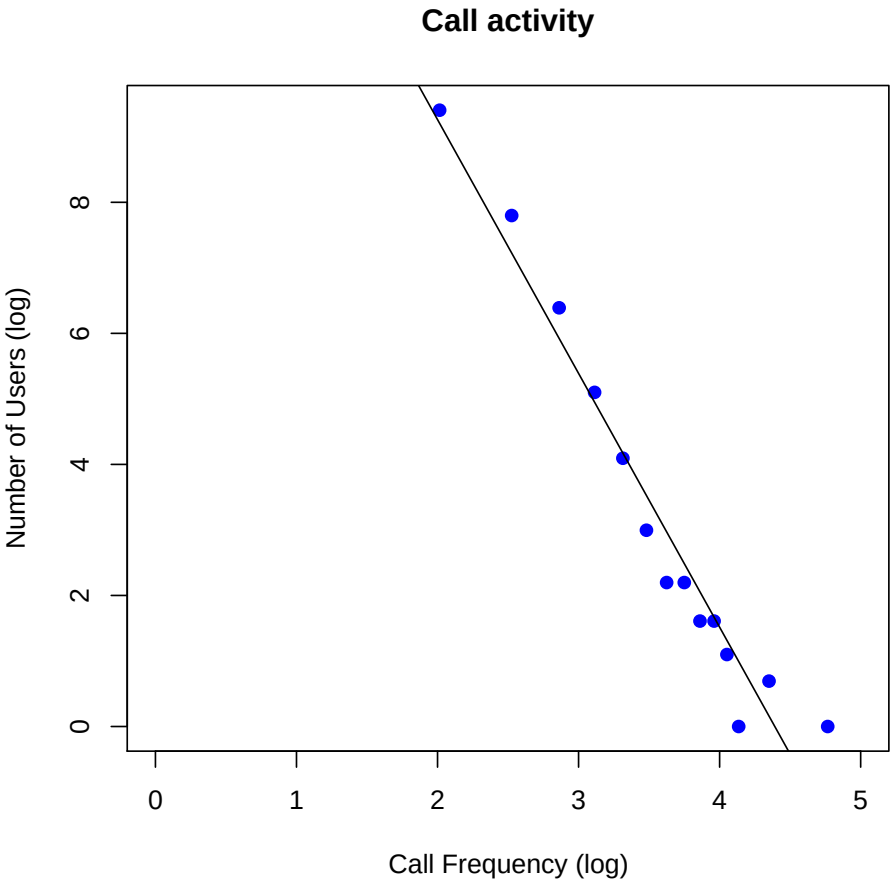

Figure C.2. Call activity for the 5 day period, excluding 0 values, log-log plot.

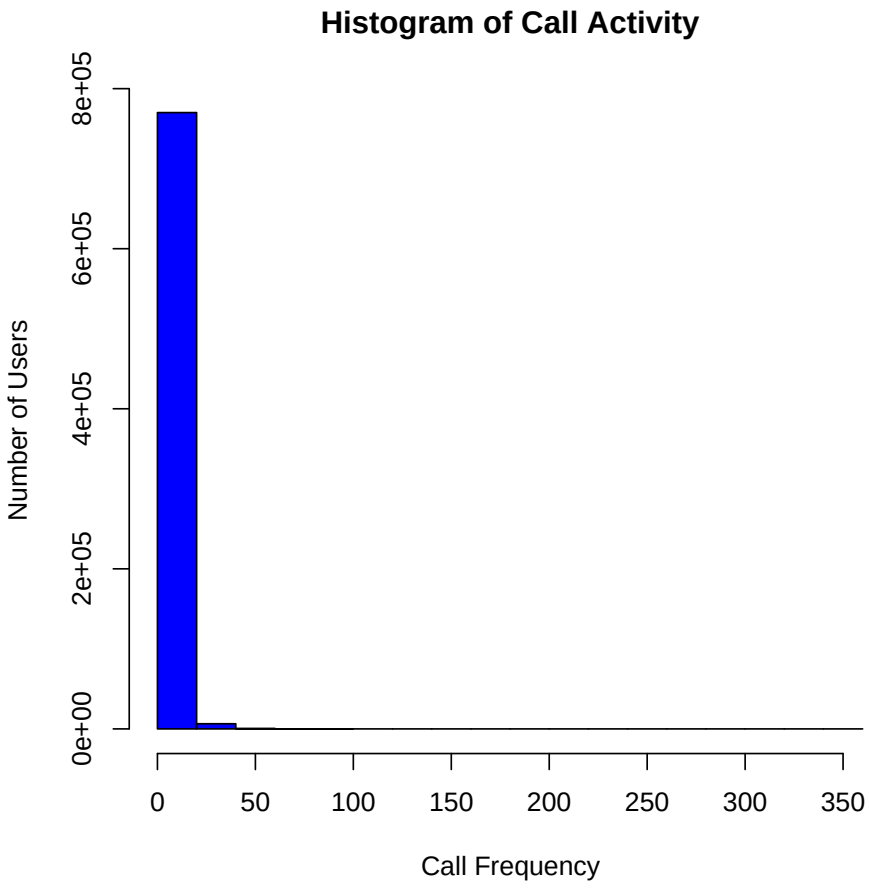

Figure C.3. Histogram of the call activity for the 15 day period.

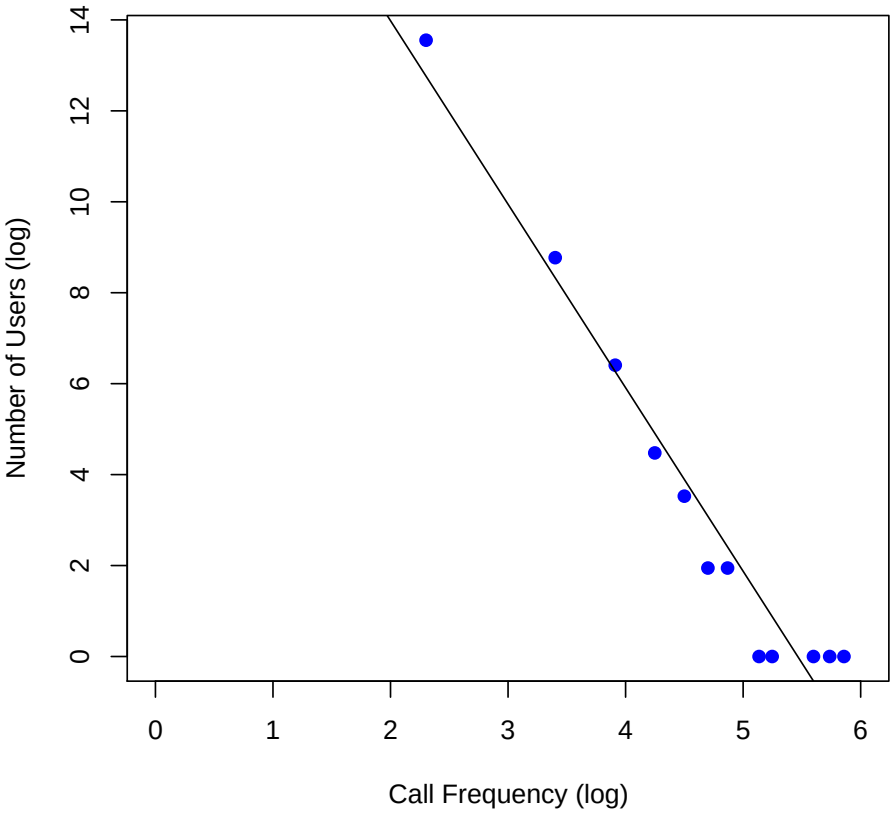

Figure C.4. Call activity for the 15 day period, excluding 0 values, log-log plot.

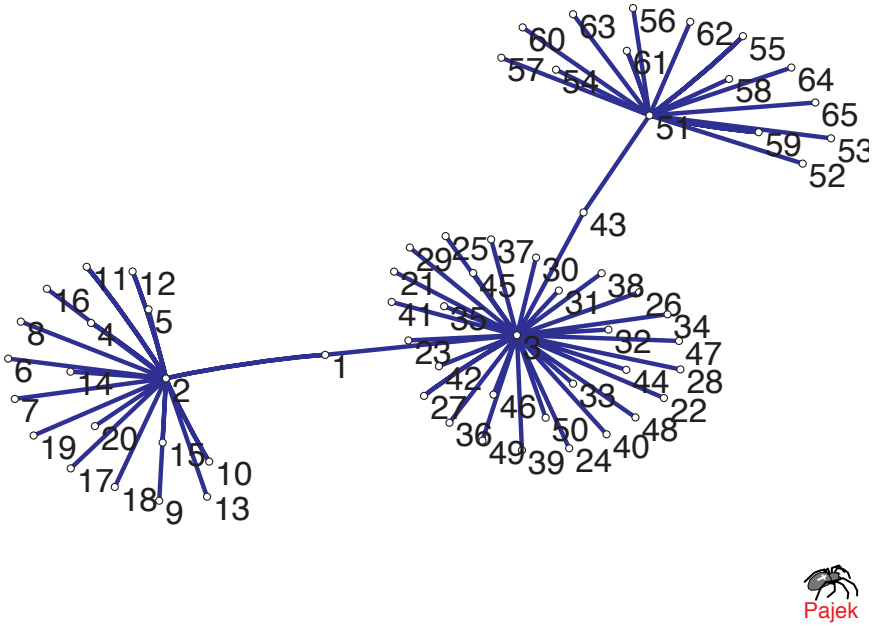

Figure C.5. Component 1

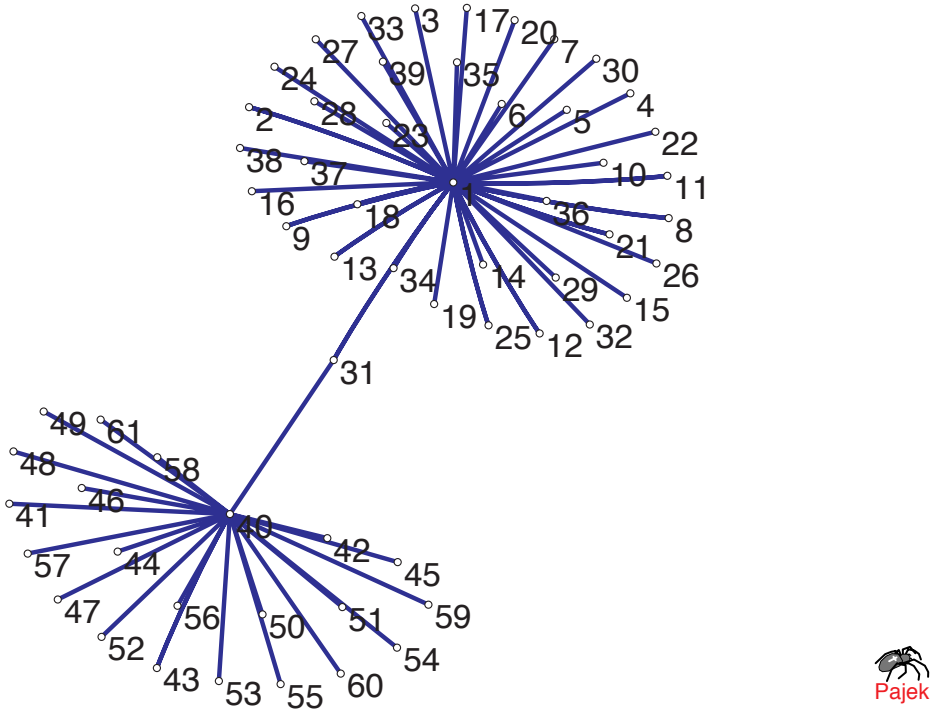

Figure C.6. Component 2

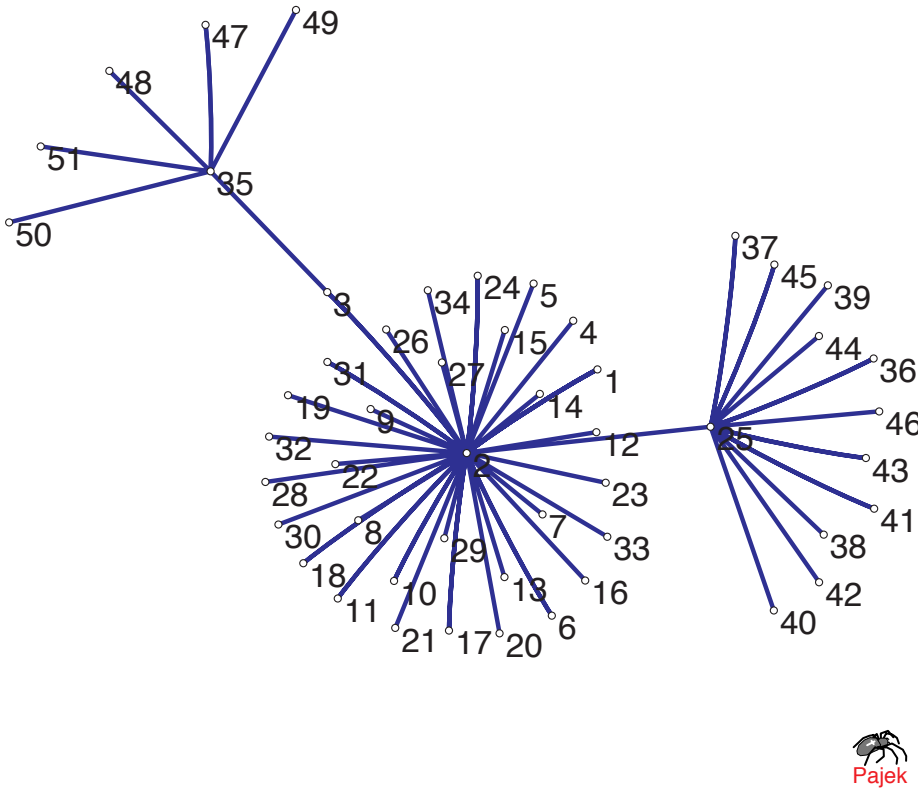

Figure C.7. Component 3

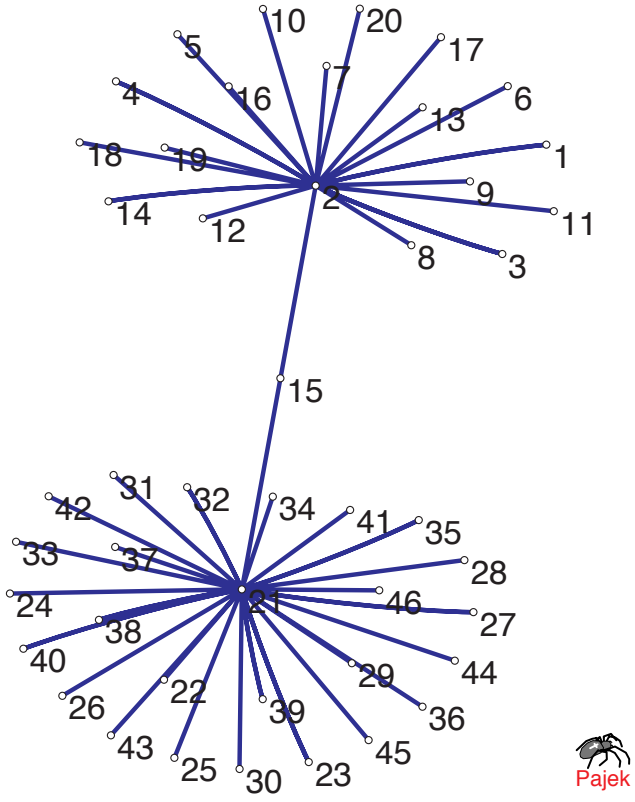

Figure C.8. Component 4

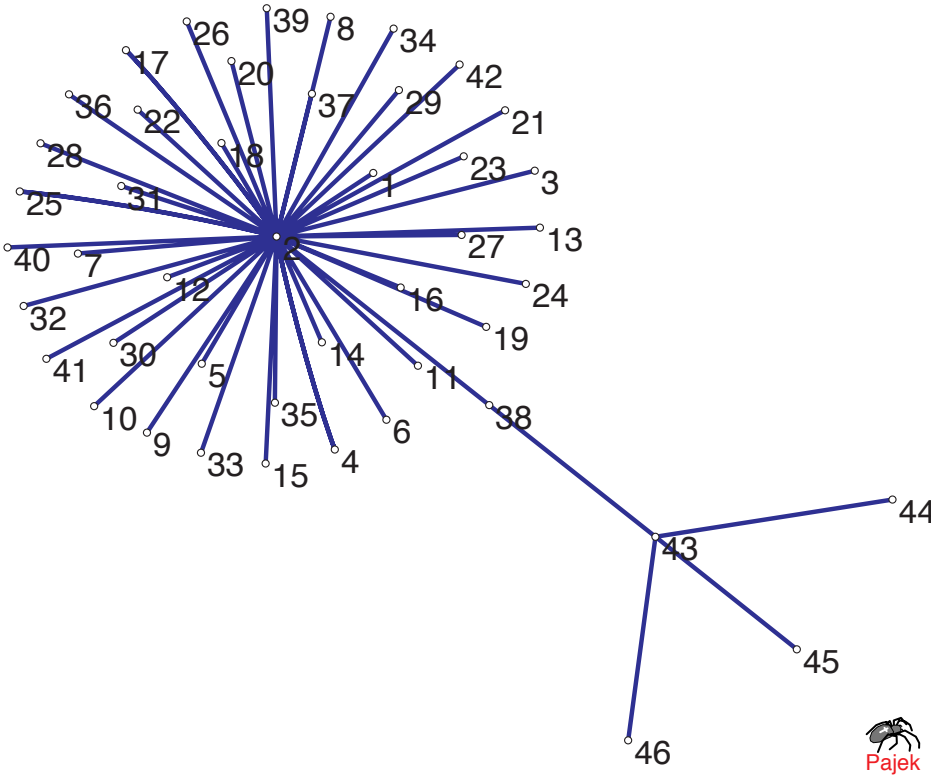

Figure C.9. Component 5

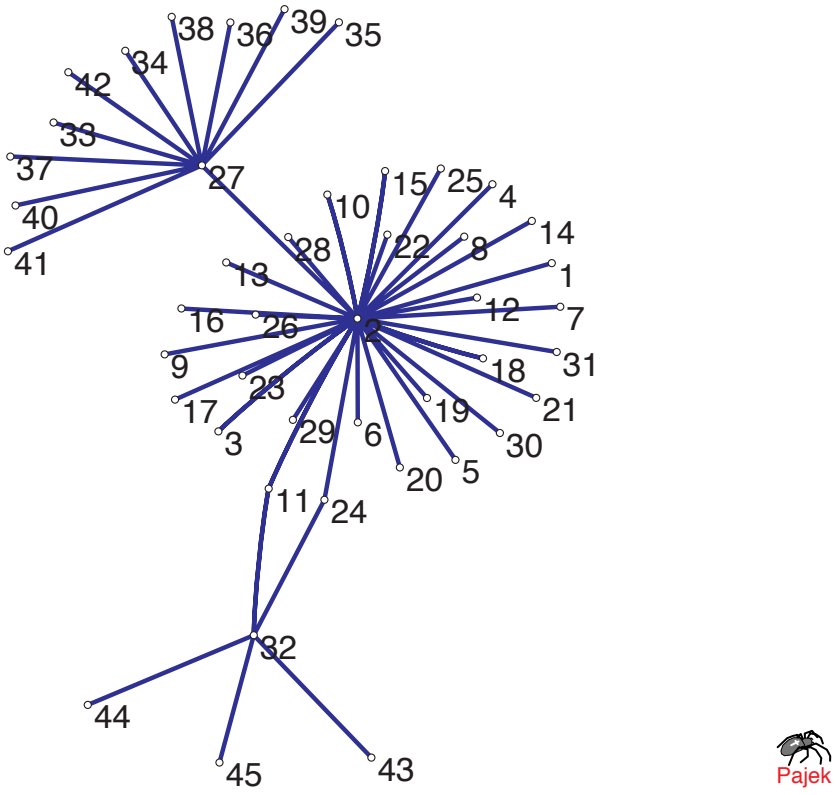

Figure C.10. Component 6

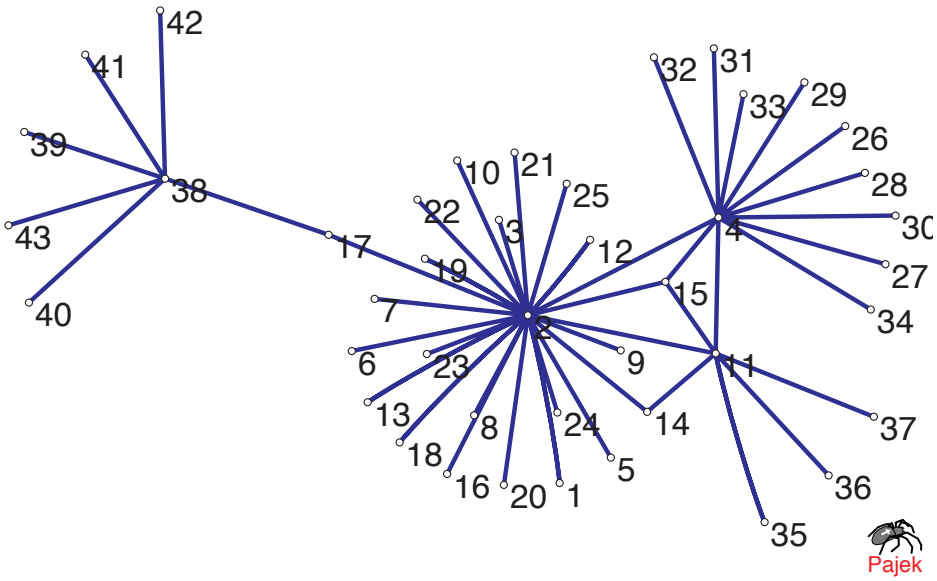

Figure C.11. Component 7

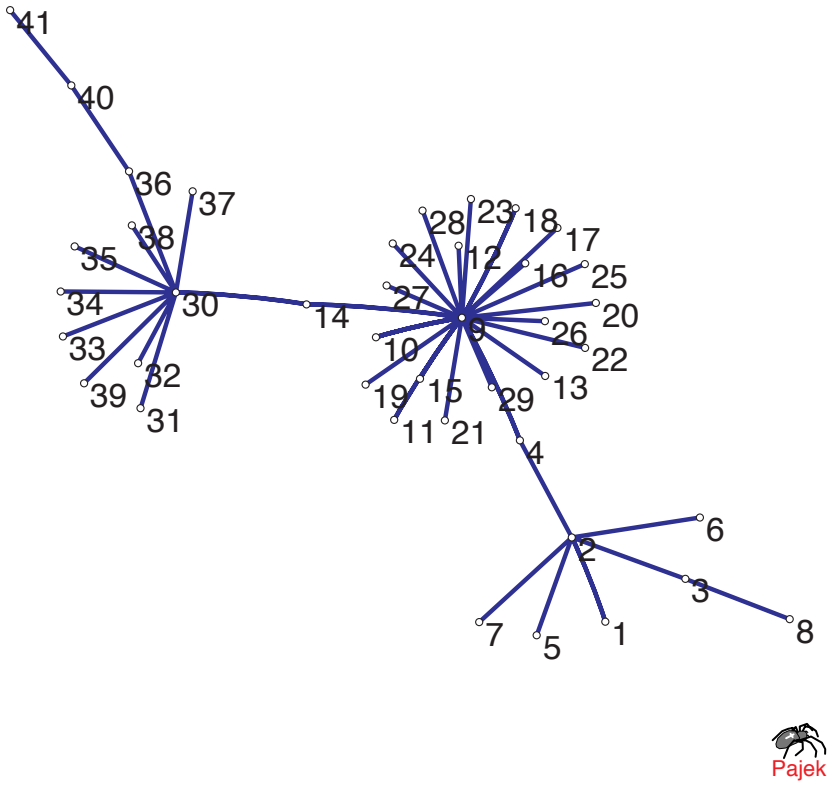

Figure C.12. Component 8

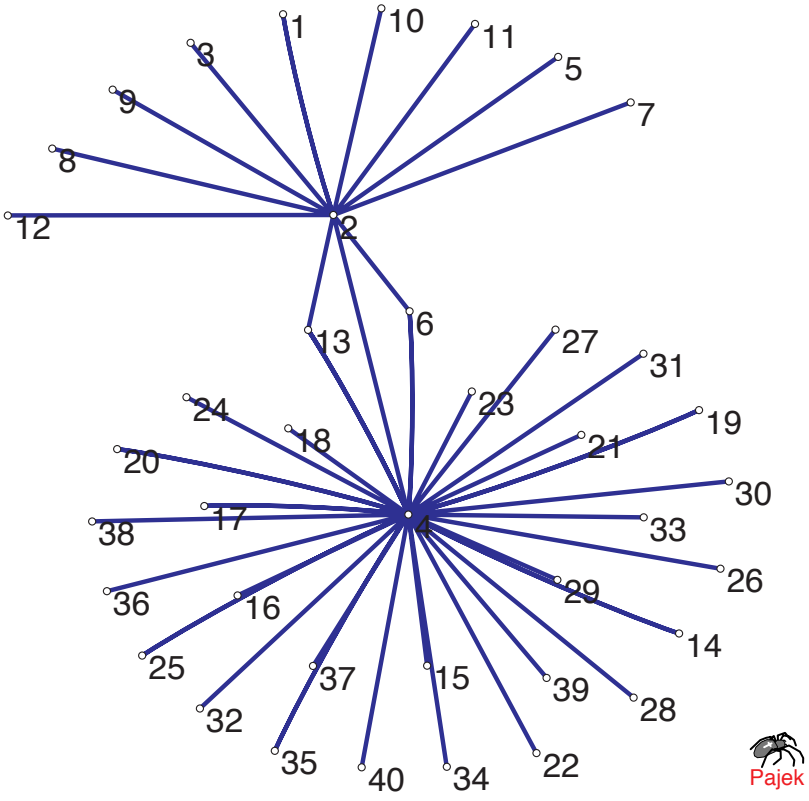

Figure C.13. Component 9

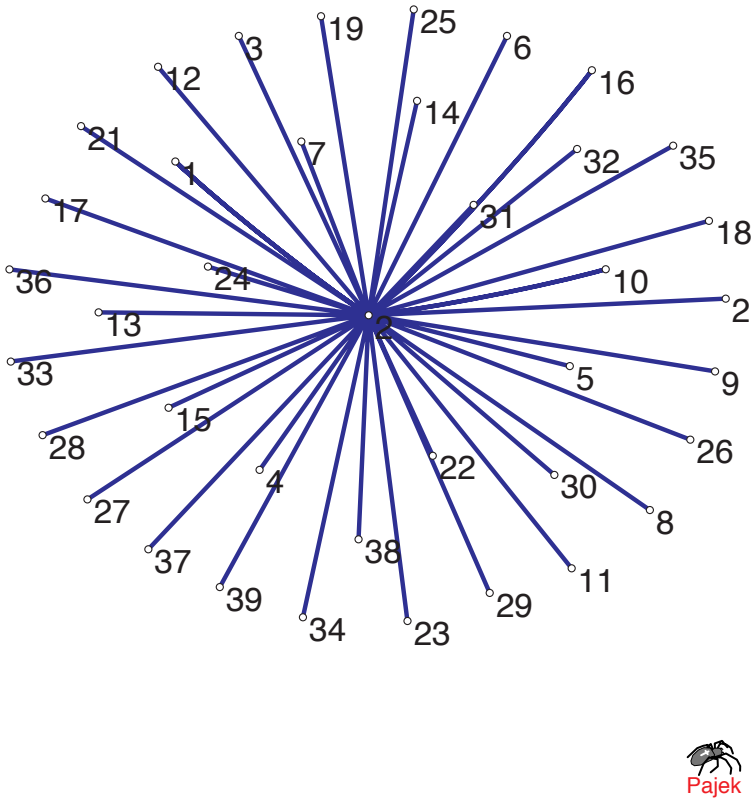

Figure C.14. Component 10

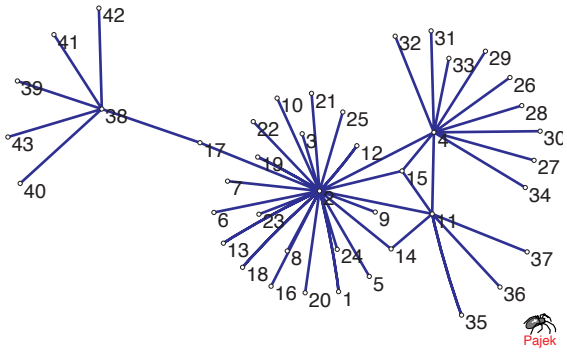

Figure C.15. Component 7, as visualized with GraphViz.

APPENDIX D

VEIN: A RUBY PACKAGE FOR SOCIAL NETWORK ANALYSIS

D.1 ABSTRACT

This paper introduces the VEIN social network analysis package. VEIN is designed to be a lightweight, scriptable alternative to packages like Jung. VEIN reads and writes files in the Pajek network file format and produces visualizations using GraphViz. [1]

Key Words: Network Analysis, Ruby

D.2 INTRODUCTION

There exist many social network analysis tools, from standalone GUI programs such as UCINET[16] and Pajek [13] to broad, comprehensive APIs like Jung [76]. However, many of these tools are difficult to use and do not scale well. There is a need for simple, extensible tools that are easy to work with, can read and write common network file formats and can handle very large networks.

In order to motivate the need for VEIN, consider the following: as part of a course project, we needed to analyze a social network derived from cellular telephone data over a period of 5 days. The resulting network, with 424,000 actors,

[1] This work was originally published in the proceedings of NAACSOS 2006 [89].

challenged the limitations of Pajek and UCINet and we initially began the analysis with a custom Java program using JUNG. However, writing the application was labor intensive, and when finished, we were surprised by the actual structure of the network. Instead of being one large component, the network consists of 135,780 small components. We examined a larger dataset, covering 15 days, however this network, consisting of 777,000 actors, was too large to even load into memory with JUNG, on a 64-bit machine with 8GB of memory. VEIN was initially created to analyze this network, which it did.

VEIN is intended to be a simple, easy to use and lightweight but powerful package for social network analysis. It is written in Ruby[64], an Object-Oriented language that is ideally suited for scripting prototype development.

D.3 BACKGROUND

The goals behind the development of VEIN are very similar to the work already found in the BOOST [56] and JUNG [76] libraries. These APIs implement graph algorithms in a modular way, making it possible to create custom utilities to analyze and visualize networks. However, in some instances, it is important to create a proof-of-concept prototype or to write a small utility to confirm a hypothesis. In these cases a heavy API can require more time and effort than a programmer is willing to commit.

The BOOST graphics library provides Python bindings, which makes it a very close neighbor to VEIN. However, BOOST provides relatively few algorithms specifically designed for social network analysis.

RGL [32] is a graph library written in Ruby and inspired by BOOST. The goals of RGL are similar to those of VEIN, namely to provide a simple, efficient

graph API. RGL, like BOOST, does not have a social network focus, leaving programmers to implement many important social network algorithms.

UCINet [16] is a comprehensive package for network understanding, but it only runs on Windows, does not support scripting and does not offer an API. This is important when developing custom applications or writing scripts to batch process a large number of networks.

Pajek [13] is another excellent tool for network visualization and analysis, but its limitations are similar to those of UCINet: it is Windows only and only supports interactive use.

D.4 VEIN OVERVIEW

VEIN does not intend to supplant packages like Pajek and UCINet, rather to compliment them and add another tool to the arsenal of the social network analyst. VEIN is intended to be easily extensible and incorporated into scripts, so that ideas can be easily tested for feasibility or exploratory analysis.

D.4.1 DESIGN

The design of VEIN at this stage is very rudimentary and we expect it to become more sophisticated as it develops. Currenly VEIN consists of Graph, Node and Edge classes, with algorithms, such as All-Pairs Shortest Path [111] as part of the Graph class. We intend to adopt the design methodology of BOOST and RGL, which is to develop a Graph interface that hides the implementation and provide simple iterators for traversing the graph, optimized for the underlying graph structure[32, 56]. This approach was first described by Dietmar Kühl and is part of an approach called graph design patterns [55].

D.4.2 FEATURES

VEIN currently supports reading and writing of networks in the Pajek network format. This format was chosen because it has nearly universal support among social network analysis packages and due to its simplicity. When analyzing social networks, visualization is an important concern. VEIN uses GraphViz [38] to visualize graphs, using a Ruby/GraphViz bridge[58]. VEIN currently provides breadth-first and depth-first search iterators on the graph, as well as the Floyd-Warshall All-Pairs Shortest Path algorithm [111], algorithms for generating the connected components, the network degree distribution and diameter.

D.4.3 VERIFICATION

VEIN is being developed with a large testing suite to validate all of its algorithms and classes. This testing is a mix of black box and white box tests that compare VEIN's algorithms to those implemented in JUNG and elsewhere. Test cases are written in Ruby using the built-in "test" unit-testing package.

D.5 CONTRIBUTION

The VEIN package for network analysis is a lightweight but powerful tool for analyzing social networks. Its simple API makes it easy to incorporate into scripts for exploratory analysis of large scale social networks.

D.6 FUTURE WORK

Development of VEIN is ongoing. We are currently in an early stage of development but look forward to providing the social network analysis community with a new lightweight tool for the rapid prototyping of social network applications.

Among the changes that need to be implemented are the development of adjacency list-based algorithms, such as the Dijkstra algorithm, which will considerably improve the scalability. We would also like to add in a large validation set of canonical social networks to the testing suite.

D.7 AVAILABILITY

VEIN is free and Open Source software and is available on the Sourceforge website at `http://sourceforge.net/projects/ruby-vein/`.

D.8 ACKNOWLEDGMENTS

Thanks to my advisor, Greg Madey, for encouragement on this project. Also thanks to Nitesh Chawla, as VEIN began as a course project for his Data Mining class. This work is supported by a fellowship from the Arthur J. Schmitt Foundation.

APPENDIX E

DESIGN AND IMPLEMENTATION OF AN EXTENSIBLE, FLEXIBLE DATA CURATION SYSTEM

E.1 ABSTRACT

This appendix describes the prototype data curation system that has been designed and implemented for the managing of data for the WIPER project. In any scientific research project, data must be carefully collected, annotated, analyzed and preserved. The recent explosion of research data has made it clear that ad-hoc and manual methods for data curation are insufficient to keep pace with the data deluge. Certain inevitable consequences of using datasets and the desire for replicability of results adds to the problem, forcing researchers to store multiple copies of the same data in various forms. This chapter explores one approach to the management of large and ever-expanding datasets, addressing issues such as determining data quality, tradeoffs between the desire for privacy and ease of use, management in the face of changing data schemas and observations from the implementation and roll-out of a data curation and management system.

E.2 INTRODUCTION

In the WIPER project we receive 10-25GB of data per month. The data comes in several (3-5) file formats and each file is partitioned into 1GB chunks. Certain

data has a useful lifetime measured in weeks, with its value decreasing daily, thus this data must be processed as quickly as possible. Processing, in this context entails the following:

1. Assemble the chunks into a large, compressed file
2. Decompress the file (resulting in a 5-10x increase in file size)
3. Run tests to determine the data quality (aggregate statistics, other measures)
4. Hash any sensitive data
5. Encrypt and securely store the original files
6. Put hashed data in an accessible location for the researchers

APPENDIX F

HIGH PERFORMANCE DISK CONFIGURATION FOR RAPID DATABASE ACCESS

F.1 OVERVIEW

In this section we present a look at the effect of disk configuration on database performance. Performance is always a concern, but it is elevated to primary importance when working with very large databases. Improper choices relating to storage and infrastructure can cause user queries to run longer, making certain types of especially-long running queries impossible. We test a large database of 100 million records on 4 different disk configurations: single disk, RAID 0, RAID 1 and RAID 5. We make our selection based on the expected load characteristics on our database server.

F.2 RAID

Recognizing the need for increased performance and reliability using commodity hard drives, Patterson et al [77] suggested several methods of treating several disks as one large drive. This concept, Redundant Array of Inexpensive Disks (RAID), as outlined in [77] allows users to combine several inexpensive commodity hard drives into a virtual disk that behaves like a large, high performance,

high reliability drive. There are several levels of RAID, but the most common seen today are levels 0, 1, 5 and 0+1.

RAID level 0 spreads writes across 2 or more disks. This is referred to as *striping*. Two disks are essentially combined into one large virtual disk and blocks are written to only one of the two disks. The idea is that writes will be approximately as fast as with a single disk, but reads will see a speedup, since blocks will be read from two disks at once. The problem with RAID 0 is that it does not offer any redundancy, so the failure of one disk can mean the loss of all data on the drives.

RAID level 1, also called *mirroring*, duplicates writes to two disks simultaneously. Ideally, this results in an exact copy of all blocks on both disks. Thus if one disk fails, there is an exact duplicate of all data, so nothing is lost. Unfortunately, this configuration requires twice the amount of storage and performs poorly in terms of writes, as it takes the maximum of the write time for both disks, in addition to any operating system overhead to duplicate the writes. Reads should, ideally, be similar to those on a single disk.

RAID level 5 spreads the writing of data blocks across several disks, similar to striping, but adds a parity block. If there are 3 disks (the minimum), a write operation sends data blocks to two of the disks and a parity block to the third disk. The parity is calculated in such a way that having just the parity block and one of the data blocks allows the second data block to be computed. This method of redundancy significantly reduces the amount of storage needed to insure data security in the face of a single disk failure. Writes suffer a performance penalty due to the calculation of the parity block, but reads approach the level of a stripe.

RAID level 0+1 is a combination of mirroring and striping. It requires at least 4 disks. The four disks are split into two mirrored groups which are treated

as two virtual disks. These two virtual disks are then used in a RAID 0 stripe configuration. This configuration supposedly offers the benefits of both RAID 0 and 1, namely fast reads and quick recovery in the even of an error. Due to the large number of disks required for level $0 + 1$, we do not examine this configuration.

F.3 TESTING METHODOLOGY

The test system is a dual Opteron 250 system with 4GB of RAM and four 300GB SATA disks, running RedHat Advanced Server 4. In each instance, the operating system is installed on one disk and the remaining 3 disks are used for the RAID array. We configured our RAID arrays using mdadm [65], a standard Linux tool for working with RAID arrays. Our database system is PostgreSQL, version 8.0.4 with the type 3 jdbc driver, version 8.0.313. The test code was written with the Sun Java 1.5 SDK for AMD64.

The database is configured with a single table. Records in the table have a serial value which also is the primary key, as well as a 32 character field (*name*) and a 16 character field (*etc*). The Java test program creates records with a random 10 character string for the name field and a 5 character string for the etc field.

We use two simple Java programs to measure the performance on two database tasks. The code runs on the same machine as the database and connects to the database using jdbc. The test code uses the java.system.currentTimeMillis method to measure execution time. The Insert test code first inserts 50 million records into a database that starts with 50 million records, then runs a Select on the resulting 100 million record database. The runtime measures the combined time to run both the Insert and Select operations. The Select test starts with the 100

million record database and runs a Select query. For both sets of test code the Select operation is chosen in such a way that the expected number of returned records is significantly less than 1. The rationale for this approach is that in previous attempts to measure performance, the query returned so many records that it filled available memory and began to compete for system resources with the database, thus affecting the results of the test. See Sections F.7 and F.8 for a detailed listing of the test code.

F.4 DISK CONFIGURATIONS

For the test we measure the performance on four different disk configurations: single disk, RAID 0 (stripe), RAID 1 (mirror) and RAID 5. We use a single disk in order to get a baseline level of performance. All three of the RAID configurations are setup to use three disks. For RAID 1 (mirror) and 0 (stripe), the third disk acts as a spare and thus should not adversely affect performance. In all cases the RAID partitions are created using the default settings in mdadm [65].

F.5 RESULTS

The results of the performance tests are shown graphically in Figures F.2 and F.3. A table of the results, with values in hours and minutes as well as milliseconds, is shown in Figure F.1. On the Insert test, the RAID 0 (stripe) configuration yields the best results, taking 3.88 hours, with the single disk being only 1.8 minutes (.66%) slower. The RAID 5 configuration is 26% slower, taking 4.87 hours, and the RAID 1 (mirror) configuration is 34% slower, taking 5.2 hours. For the Select test, the RAID 0 (stripe) configuration ran fastest, at 1.45 minutes. The RAID 5 configuration was second at 1.71 minutes (18% slower). The single

RAID	Time for insert	Time in hours	Time for select	Time in minutes
RAID 1 (Mirror)	18709186	5.20	218654	3.64
RAID 0 (Stripe)	13967807	3.88	87071	1.45
Single Disk	14059829	3.91	208011	3.47
RAID 5 (1 Parity)	17534300	4.87	102568	1.71

Figure F.1. Elapsed time for test programs, time in milliseconds, hours and minutes.

disk, surprisingly, came in third at 3.47 minutes (39% slower) and the RAID 1 (mirror) was last, at 3.64 minutes (51% slower).

F.6 CONCLUSIONS

The performance results suggest that for raw performance, there is a statistically significant advantage to arranging the disk drives in a RAID 0 (stripe) configuration. However, there are several mitigating factors that prevent us from implementing this. First, we know that the database will primarily be used in a static context, that is, once the data has been imported, it will remain constant and all database actions will be queries on the static data. Second, based on the size of the database and its critical importance to the research project, reliability must be factored in to any decision. Based on these assumptions, we can place less importance on the Insert results and also weigh data redundancy with performance. Examining solely the Select test, we see that RAID 5 has similar performance to RAID 0, and significantly better performance than RAID 1 or a single disk. Thus the best balance of performance on a Select workload and data redundancy is RAID 5.

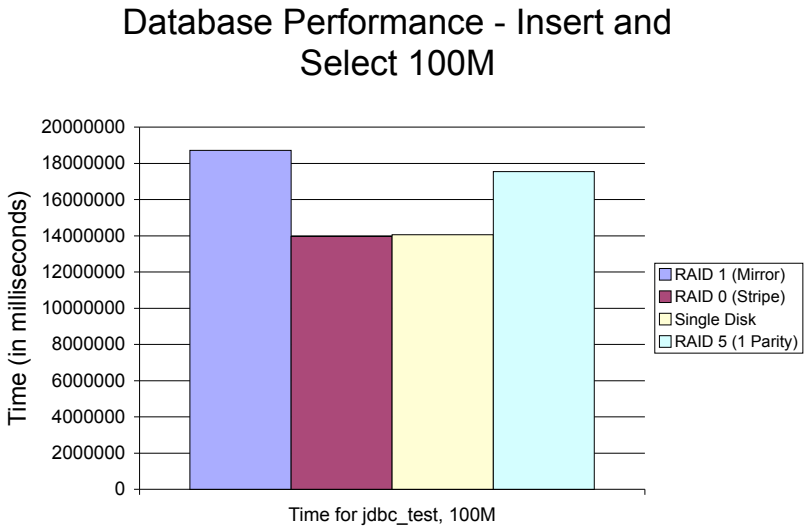

Figure F.2. Performance on insert on 50 million records, select on 100 million records.

Figure F.3. Performance on select on 100 million records.

```
import java.util.*;
import java.sql.*;
import cern.jet.random.*;

public class jdbc_test {

        private Uniform randomGen;

public jdbc_test(){
        randomGen = new Uniform(65,90,48930444);
}

public static void main(String[] args){

        try{
                Class.forName("org.postgresql.Driver");
        } catch(Exception e){
                System.out.println(e.toString());
        }
        jdbc_test jdbc = new jdbc_test();


        long begin = System.currentTimeMillis();
        jdbc.go();
        long end = System.currentTimeMillis();

        System.out.println("Runtime was " + (end - begin) + " Milliseconds");
}

public void go(){

        try{
        Connection db = DriverManager.getConnection("jdbc:postgresql:test",
         "tschoenh", "");
        Statement st = db.createStatement();

        for(int i=0; i<50000000; i++){
                st.executeUpdate("INSERT INTO testtable(title, etc)
```

```
                VALUES('" + randomName() +"','"+ randomEtc() +"')");
        }
        ResultSet rs = st.executeQuery("SELECT * FROM testtable
         WHERE title = 'AAAAAAAAAA'");

        while (rs.next()) {
                rs.getString(1);
        }
        rs.close();
        st.close();

        } catch(Exception e){
                System.out.println(e.toString());
        }

}

        public String randomName(){

                char[] name = new char[10];
                for(int i=0; i<10; i++){
                        name[i]=randomChar();
                }
                return new String(name);
        }

        public String randomEtc(){

                char[] etc = new char[5];
                for(int i=0; i<5; i++){
                        etc[i]=randomChar();
                }
                return new String(etc);
        }

        public char randomChar(){
                return (char)randomGen.nextInt();
        }

}
```

F.8 JDBC SELECT CODE

```
import java.util.*;
import java.sql.*;
import cern.jet.random.*;

public class select_test {

        private Uniform randomGen;

public select_test(){
        randomGen = new Uniform(65,90,48930444);
}

public static void main(String[] args){

        try{
                Class.forName("org.postgresql.Driver");
        } catch(Exception e){
                System.out.println(e.toString());
        }
        select_test jdbc = new select_test();


        long begin = System.currentTimeMillis();
        jdbc.go();
        long end = System.currentTimeMillis();

        System.out.println("Runtime was " + (end - begin) + " Milliseconds");
}

public void go(){

        try{
        Connection db = DriverManager.getConnection("jdbc:postgresql:test",
         "tschoenh", "");
        Statement st = db.createStatement();
        }
        ResultSet rs = st.executeQuery("SELECT * FROM testtable
         WHERE title = 'AAAAAAAAAA'");
```

```
        while (rs.next()) {
                rs.getString(1);
        }
        rs.close();
        st.close();

        } catch(Exception e){
                System.out.println(e.toString());
        }

}

        public String randomName(){

                char[] name = new char[10];
                for(int i=0; i<10; i++){
                        name[i]=randomChar();
                }
                return new String(name);
        }

        public String randomEtc(){

                char[] etc = new char[5];
                for(int i=0; i<5; i++){
                        etc[i]=randomChar();
                }
                return new String(etc);
        }

        public char randomChar(){
                return (char)randomGen.nextInt();
        }

}
```

BIBLIOGRAPHY

1. Volkan Akcelik, Jacobo Bielak, George Biros, Ioannis Epanomeritakis, Omar Ghattas, Loukas F. Kallivokas, and Eui Joong Kim. A framework for online inversion-based 3d site characterization. In M. Bubak et al, editor, *Lecture Notes in Computer Science (LNCS)*, volume 3038, pages 717–724. Springer, 2005.

2. Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

3. Sigrún Andradóttir. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, chapter Simulation Optimization. John Wiley & Sons, New York, NY, 1998.

4. AP. Tracking cell phones for real-time traffic data. http://www.wired.com/news/wireless/0,1382,69227,00.html, October 2005.

5. M. P. Armstrong. Geographical informational technologies and their potentially erosive effects on personal privacy. *Studies in the Social Sciences*, 27:19–28, 2002.

6. M. P. Armstrong and A. Ruggles. Geographical information technologies and personal privacy. *Cartographica*, 40:63–73, 2005.

7. Osman Balci. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, chapter Verification, Validation, and Testing. John Wiley & Sons, New York, NY, 1998.

8. Steven C. Bankes. Agent-based modeling: A revolution? *Proceedings of the National Academy of Sciences*, 99:7199–7200, May 2002.

9. Jerry Banks, John Carson, Barry Nelson, and David Nicol. *Discrete-Event System Simulation*. Prentice Hall, Upper Saddle River, NJ, Third edition, 2005.

10. A. Barabasi. *Linked: The New Science of Networks*. Perseus Publishing, Cambridge, MA, 2002.

11. C. L. Barrett and et al. Transportation analysis simulation system. Technical report, Los Alamos National Laboratory, 2004.

12. Chris L. Barrett, Stephen G. Eubank, and James P. Smith. If smallpox strikes portland ... *Scientific American*, March 2005.

13. Vladimir Batagelj and Andrej Mrvar. Pajek: Analysis and visualization of large networks. *University of Ljubljana Preprint Series*, 2003.

14. Michael Batty and Bin Jiang. Multi-agent simulation: New approaches to exploring space-time dynamics within gis. *Graphical Information Systems Research - UK (GISRUK) 1999*, April 1999.

15. Ling Bian. A conceptual framework for an individual-based spatially explicit epidemiological model. *Environment and Planning B*, 31:381–395, 2004.

16. Steven Borgatti, Martin Everett, and Linton Freeman. UCINET IV version 1.0 user's guide. Columbia, SC: Analytic Technologies, 1992.

17. Kai-H. Brassel. Flexible modelling with VSEit, the versatile simulation environment for the internet. *Journal of Artificial Societies and Social Simulation*, 4(3), 2001.

18. Allan Brimicombe. *GIS, Environmental Modelling and Engineering*. Taylor and Francis, 2003.

19. J. Brotzge, V. Chandresakar, K. Droegemeier, J. Kurose, D. McLaughlin, B. Philips, M. Preston, and S. Sekelsky. Distributed collaborative adaptive sensing for hazardous weather detection, tracking, and predicting. In M. Bubak et al, editor, *The Proceedings of ICCS 2004, Lecture Notes in Computer Science 3038*, pages 670–677, 2004.

20. L.E. Bruzzone, R.M. Molfino, and M. Zoppi. A discrete event simulation package for modular and adaptive assembly plants. In M.H. Hamza, editor, *Proceedings of Modelling, Identification, and Control (MIC 2003)*, February 2003.

21. Jan Burse. Quicksilver. http://quicksilver.tigris.org/, 2003.

22. Madhavi M. Chakrabarty and David Mendonca. Design considerations for information systems to support critical infrastructure management. In *Proceedings of the Second International ISCRAM Conference*, April 2005.

23. A. Chaturvedi, J. Chi, S. Mehta, and D. Dolk. SAMAS: Scalable architecture for multi-resolution agent-based simulation. In M. Bubak et al, editor, *The Proceedings of ICCS 2004, Lecture Notes in Computer Science 3038*, pages 779–788, 2004.

24. Scott Christley, Mark S. Alber, and Stuart A. Newman. Patterns of mesenchymal condensation in a multiscale, discrete stochastic model. *PLoS Computational Biology*, 3(e76), 2007.

25. Julie Clothier. Dutch trial SMS disaster alert system. `http://www.cnn.com/2005/TECH/11/09/dutch.disaster.warning/index.html`, November 2005.

26. Frederica Darema. Dynamic Data Driven Application Systems: A new paradigm for application simulations and measurements. In M. Bubak et al, editor, *The Proceedings of ICCS 2004, Lecture Notes in Computer Science 3038*, pages 662–669, 2004.

27. Frederica Darema. Grid computing and beyond: the context of dynamic data driven applications systems. In *Proceedings of the IEEE*, volume 93, pages 692–697. IEEE, March 2005.

28. Frederica Darema and et al. DDDAS workshop report. `http://www.nsf.gov/cise/cns/dddas/2006_Workshop/index.jsp`, January 2006.

29. Wayne J. Davis. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, chapter On-line Simulation: Need and Evolving Research Requirements. John Wiley & Sons, New York, NY, 1998.

30. Drew Decker. *GIS Data Sources*. John Wiley and Sons, 2001.

31. GRASS Development Team. GRASS GIS. `http://grass.itc.it`.

32. Horst Duchene. RGL: The ruby graph library. http://rubyforge.org/projects/rgl/, 2005.

33. Stephen Eubank. Scalable, efficient epidemiological simulation. *Proceedings of the 2002 ACM Symposium on Applied Computing*, pages 139–145, 2002.

34. Jacques Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

35. Open Source Geospatial Foundation. GDAL - Geospatial Data Abstraction Library. `http://www.gdal.org`.

36. Martin Fowler. *Refactoring: improving the design of existing code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

37. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

38. Emden R. Gansner and Stephen C. North. An open graph visualization system and its applications to software engineering. *Software — Practice and Experience*, 30(11):1203–1233, 2000.

39. Martin Gardner. The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, pages 120–123, October 1970.

40. The Geotools project. `http://geotools.codehaus.org`, 2006.

41. H. Randy Gimblett. *Integrating Geographic Information Systems and Agent-Based Technologies for Modeling and Simulating Social and Ecological Phenomenon*, pages 1–20. Oxford University Press, 2002.

42. Harold Greenberg. An analysis of traffic flow. *Operations Research*, 7(1):79–85, January - February 1959.

43. B. D. Greenshields, J. R. Bibbins, W. S. Channing, and H. H. Miller. A study of traffic capacity. *Highway Research Board Proceedings*, 14:448–477, 1935.

44. Volker Grimm and Steve Railsback. *Individual-based Modeling and Ecology*. Princeton University Press, 2005.

45. Volker Grimm, Eloy Revilla, Uta Berger, Florian Jeltsch, Wolf M. Mooij, Steven F. Railsback, Hans-Hermann Thulke, Jacob Weiner, Thorsten Wiegand, and Donald L. DeAngelis. Pattern-oriented modeling of agent-based complex systems: Lessons from ecology. *Science*, 310(5750):987–991, November 2005.

46. M. Hare and P. Deadman. Further towards a taxonomy of agent-based simulation models in environmental management. *Mathematics and Computers in Simulation*, 2003.

47. Peter Hidas. Modelling lane changing and merging in microscopic traffic simulation. *Transportation Research Part C: Emerging Technologies*, 10:351–371, October-December 2002.

48. John H. Holland. *Hidden order: how adaptation builds complexity*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995.

49. Wolfgang Hoschek. Colt high performance scientific library. `http://dsd.lbl.gov/~hoschek/colt/index.html`, 2004. [Online; accessed 28-March-2007].

50. R. M. Itami and H. R. Gimblett. Intelligent recreation agents in a virtual gis world. *Complexity International*, 8, 2001.

51. J. Jenvald, M. Morin, and J. Peter Kincaid. A framework for web-based dissemination of models and lessons learned from emergency-response operations. *International Journal of Emergency Management*, 1(1):82–94, 2001.

52. J. Jenvald, J. Stjernberger, A. Nygren, and H. Eriksson. Using wireless networks to provide early warning of emergency incidents. In *Proceedings of the International Emergency Management Society's 8th Annual Conference (TIEMS 2002)*, pages 523–533, 2002.

53. Bin Jiang and H. Randy Gimblett. *An Agent-Based Approach to Enviromental and Urban Systems within Geographic Information Systems*, pages 171–189. Oxford University Press, 2002.

54. Ryan C. Kennedy. Verification and validation of agent-based and equation-based simulations and bioinformatics computing: identifying transposable elements in the aedes aegypti genome. Master's thesis, University of Notre Dame, 2006.

55. Dietmar Kühl. Design patterns for the implementation of graph algorithms. Master's thesis, Technische Universität Berlin, 1996.

56. Lie-Quan Lee, Jeremy G. Siek, and Andrew Lumsdaine. The generic graph component library. In *OOPSLA '99: Proceedings of the 14th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 399–414, New York, NY, USA, 1999. ACM Press.

57. Ola Leifler and Johan Jenvald. Critique and visualization as decision support for mass-casualty emergency management. In *Proceedings of the Second International ISCRAM Conference*, April 2005.

58. Gregoire Lejeune. ruby-graphviz. `http://raa.ruby-lang.org/project/ruby-graphviz/`, 2005.

59. Jim Lemon, Ben Bolker, Sander Oom, Eduardo Klein, Barry Rowlingson, Hadley Wickham, Anupam Tyagi, Olivier Eterradossi, and Gabor Grothendieck. *plotrix: Various plotting functions*, 2007. R package version 2.2.

60. Gregory R. Madey, Albert-László Barabási, Nitesh V. Chawla, Marta Gonzalez, David Hachen, Brett Lantz, Alec Pawling, Timothy Schoenharl, Gábor Szabó, Pu Wang, and Ping Yan. Enhanced situational awareness: Application of DDDAS concepts to emergency and disaster management. In Y. Shi, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, editors, *Lecture Notes in Computer Science (LNCS 4487)*, pages 1090–1097. Springer, May 2007.

61. Pattie Maes. Agents that reduce work and information overload. *Commun. ACM*, 37(7):30–40, 1994.

62. J. Mandel, M. Chen, L.P. Franca, C. Johns, A. Puhalskii, J.L. Coen, C.C. Douglas, R. Kremens, A. Vodacek, and W. Zhao. A note on dynamic data driven wildfire modeling. In *Lecture Notes in Computer Science (LNCS)*, volume 3038, pages 725–731. Springer, 2005.

63. Massimo Marchiori and Vito Latora. Harmony in the small-world. *Physica A*, 285:539–546, 2000.

64. Yukihiro Matsumoto. The ruby programming language. *InformIT*, June 2000.

65. mdadm: Manage md devices aka software raid - linux man page. http://www.die.net/doc/linux/man/man8/mdadm.8.html.

66. Seapahn Meguerdichian, Farinaz Koushanfar, Miodrag Potkonjak, and Mani B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *INFOCOM*, pages 1380–1387, 2001.

67. Rachel Metz. Saving the world with cell phones. `http://www.wired.com/news/wireless/0,1382,68485,00.html`, August 2005.

68. J. Michopoulos, P. Tsompanopoulou, E. Houstis, and A. Joshi. Agent-based simulation of data-driven fire propagation dynamics. In M. Bubak et al, editor, *The Proceedings of ICCS 2004, Lecture Notes in Computer Science 3038*, pages 779–788, 2004.

69. N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The swarm simulation system: a toolkit for building multi-agent simulations. Technical Report 96-06-042, Santa Fe Institute, 1996.

70. D. Mollison. Modelling biological invasions: Chance, explanation, prediction. *Philosophical Transactions of the Royal Society of London B*, 314:675–693, 1986.

71. Soraia Raupp Musse and Daniel Thalmann. Hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):152–164, 2001.

72. K. Nagel, R. Beckman, and C. Barrett. Transims for urban planning. Technical Report LA-UR 984389, Los Alamos National Laboratory, Los Alamos, NM., 1999.

73. John Von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.

74. M.J. North, N.T. Collier, and J.R. Vos. Experiences creating three implementations of the Repast agent modeling toolkit. *ACM Transactions on Modeling and Computer Simulation*, 16:1–25, January 2006.

75. J. Tinsley Oden. The NSF blue ribbon panel on simulation-based engineering science: Revolutionizing engineering science through simulation, February 2006.

76. Joshua O'Madadhain, Danyel Fisher, Padhraic Smyth, Scott White, and Yan-Biao Boey. Analysis and visualization of network data using jung. *Journal of Statistical Software*, preprint, 2005.

77. David A. Patterson, Garth Gibson, and Randy H. Katz. A case for redundant arrays of inexpensive disks (RAID). In *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pages 109–116, New York, NY, USA, 1988. ACM Press.

78. Alec Pawling, Nitesh V. Chawla, and Greg Madey. Anomaly detection in a mobile communication network. In *Proceedings of the North American Association for Computational Social and Organizational Science (NAACSOS) Conference 2006*, 2006.

79. Juta Pichitlamken and Barry L. Nelson. A combined procedure for optimization via simulation. *ACM Trans. Model. Comput. Simul.*, 13(2):155–179, 2003.

80. PostgreSQL Global Development Group. PostgreSQL. `http://www.postgresql.org`.

81. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. ISBN 3-900051-07-0.

82. S. F. Railsback, B . C. Harvey, J. Hayse, and K. LaGory. Tests of theory for diel variation in salmonid feeding activity and habitat use. *Ecology*, 86:947–959, 2005.

83. Carlo Ratti and et al. SENSEable City Project. `http://senseable.mit.edu/projects/graz/`, 2005.

84. Carlo Ratti and et al. SENSEable City: Real Time Rome. `http://senseable.mit.edu/realtimerome/`, 2005.

85. Refractions Research. PostGIS. `http://postgis.refractions.net`.

86. Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, New York, NY, USA, 1987. ACM Press.

87. M. Richtel. Live tracking of mobile phones propts court fights on privacy. *The York Times, Late - Final ed.*, December 2005.

88. David A. Robalino and Robert J. Lempert. Carrots and sticks for new technology: Abating greenhouse gas emissions in a heterogeneous and uncertain world. *Integrated Assessment*, 1(1):1–19, March 2000.

89. Tim Schoenharl. VEIN: A ruby package for social network analysis. In *Proceedings of NAACSOS 2006*, 2006.

90. Tim Schoenharl, Ryan Bravo, and Greg Madey. WIPER: Leveraging the cell phone network for emergency response. *International Journal of Intelligent Control and Systems*, 11(4), December 2006.

91. Tim Schoenharl, Greg Madey, Gábor Szabó, and Albert-László Barabási. WIPER: A multi-agent system for emergency response. In *Proceedings of the Third International ISCRAM Conference*, May 2006.

92. Shahab Sheikh-Bahaei, Glen E. P. Ropella, and C. Anthony Hunt. In silico hepatocyte: Agent-based modeling of the biliary excretion of drugs. In Levent Yilmaz, editor, *Proceedings of Agent-Directed Simulation Symposium (ADS 2006)*, April 2005.

93. Leyuan Shi and Sigurdur Olafsson. Nested partitions method for global optimization. *Oper. Res.*, 48(3):390–407, 2000.

94. R. Singel. U.S. cell-phone tracking clipped. `http://www.wired.com/news/print/0,1294,69390,00.html`, 2005.

95. NSF Program Solicitation. DDDAS: Dynamic data-driven application systems. NSF Program Solicitation NSF 05-570, June 2005.

96. `http://www.media.mit.edu/starlogo`.

97. ISEE Systems. STELLA. `http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx`.

98. Tomoichi Takahashi, Satoshi Tadokoro, Masayuki Ohta, and Nobuhiro Ito. Agent based approach in disaster rescue simulation - from test-bed of multiagent system to practical application. *Lecture Notes in Computer Science*, 2377, 2002.

99. Bogdan Tatomir and Leon Rothkrantz. Crisis management using mobile adhoc wireless networks. In *Proceedings of the Second International ISCRAM Conference*, April 2005.

100. BBN Technologies. Openmap: Open systems mapping technology. http://openmap.bbn.com/.

101. Manoj Thomas, Francis Andoh-Baidoo, and Shilpa George. EVResponse - moving beyond traditional emergency response notification. In *Proceedings of the Eleventh Americas Conference on Information Systems*, 2005.

102. Robert Tobias and Carole Hofmann. Evaluation of free java-libraries for social-scientific agent based simulation. *Journal of Artificial Societies and Social Simulation*, 7(1), 2004.

103. Jiri Trnka, Michael Le Duc, and Åke Sivertun. Inter-organizational issues in ICT, GIS and GSD - mapping Swedish emergency management at the local and regional level. In *Proceedings of the Second International ISCRAM Conference*, April 2005.

104. Alan M. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society (B)*, 237(631):37–72, 1952.

105. Paul Waddell. UrbanSim: Modeling urban development for land use, transportation and environmental planning. *Journal of the American Planning Association*, 68(3):297–314, 2002.

106. Wayne W. Wakeland, Edward Gallaher, Louis Macovsky, and C. Athena Aktipis. A comparison of system dynamics and agent-based simulation applied to the study of cellular receptor dynamics. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, 2004.

107. Ruili Wang, Wensheng Zhang, and Qinghai Miao. Effects of driver behavior on traffic flow at three-lane roundabouts. *International Journal of Intelligent Control and Systems*, 10(2):123–130, June 2005.

108. Frank Warmerdam. Shapelib. `http://shapelib.maptools.org`.

109. Duncan Watts and Steven Strogatz. Collective dynamics of 'small world' networks. *Nature*, 393:440–442, 1998.

110. James E. White. *Mobile agents*, pages 437–472. MIT Press, Cambridge, MA, USA, 1997.

111. Wikipedia. Floyd-warshall algorithm — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Floyd-Warshall_algorithm&oldid=50217598`, 2006. [Online; accessed 5-May-2006].

112. Wikipedia. Karl Popper — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Karl_Popper&oldid=50312019`, 2006. [Online; accessed 25-April-2006].

113. Wikipedia. Lotka-Volterra equation — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Lotka-Volterra_equation&oldid=45321989`, 2006. [Online; accessed 25-April-2006].

114. Wikipedia. Voronoi diagram — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Voronoi_diagram&oldid=47842110`, 2006. [Online; accessed 25-April-2006].

115. Wikipedia. Chebyshev distance — wikipedia, the free encyclopedia, 2007. [Online; accessed 14-May-2007].

116. Wikipedia. Hurricane rita — wikipedia, the free encyclopedia, 2007. [Online; accessed 27-March-2007].

117. Wikipedia. Kolmogorov-smirnov test — wikipedia, the free encyclopedia, 2007. [Online; accessed 8-April-2007].

118. Wikipedia. WTO ministerial conference of 1999 protest activity — wikipedia, the free encyclopedia, 2007. [Online; accessed 27-March-2007].

119. U. Wilensky. Netlogo. `http://ccl.northwestern.edu/netlogo`, 1999. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.

120. Mark Wood. Cellalert, for government-to-citizen mass communications in emergencies. In *Proceedings of the Second International ISCRAM Conference*, April 2005.

121. Michael Woolridge and Michael J. Wooldridge. *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA, 2001.

122. Xiaorong Xiang, Ryan Kennedy, Gregory Madey, and Steve Cabaniss. Verification and validation of agent-based scientific simulation models. In Levent Yilmaz, editor, *Proceedings of the 2005 Agent-Directed Simulation Symposium*, volume 37, pages 47–55. The Society for Modeling and Simulation International, April 2005.

123. Ping Yan, Timothy Schoenharl, and Gregory R. Madey. Application of markov-modulated poisson processes to anomaly detection in a cellular telephone network. Technical report, University of Notre Dame, 2007.

124. Bernard P. Zeigler and Sankait Vahie. Devs formalism and methodology: unity of conception/diversity of application. In *WSC '93: Proceedings of the 25th conference on Winter simulation*, pages 573–579, New York, NY, USA, 1993. ACM Press.

125. Hans Zimmermann. Recent developments in emergency telecommunications. In *Proceedings of the Second International ISCRAM Conference*, April 2005.

This document was prepared & typeset with pdfLaTeX, *and formatted with* NDdiss2$_\varepsilon$ *classfile (v1.0[2004/06/15]) provided by Sameer Vijay.*