The Problem of Time: Addressing challenges in spatio-temporal data integration

Nicholas Robison

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington

2018

Reading Committee:

Neil F. Abernethy, Chair

Abraham Flaxman

Ian Painter

Program Authorized to Offer Degree:

Biomedical and Health Informatics

Abstract

The Problem of Time: Addressing challenges in spatio-temporal data integration

Nicholas Robison

Chair of the Supervisory Committee:
Neil F. Abernethy
Department of Biomedical Informatics and Medical Education

Across scientific disciplines, an ever-growing proportion of data can be effectively described in spatial terms. As researchers have become comfortable with techniques for dealing with spatial data, the next progression is to not only model the data itself, but also the complexities of the dynamic environment it represents. This has led to the rise of spatio-temporal modeling and the development of robust statistical methods for effectively modeling and understanding interactions between complex and dynamic systems. Unfortunately, many of these techniques are an extension to existing spatial analysis methods and struggle to account for the data complexity introduced by the added temporal dimension; this has limited many researchers to developing statistical and visual models that assume either a static state of the world, or one modeled by a set of specific temporal snapshots.

This challenge is especially acute in the world of public health where researchers attempting to visualize historical, spatial data, often find themselves forced to ignore shifting geographic features because both the tooling and the existing data sources are insufficient. Consider, as an example, a model of vaccine coverage for the administrative regions of Sudan over the past 30 years. In wake of civil war, Sudan was partitioned into two countries, with South Sudan emerging as an independent nation in 2011. This has an immediate impact on both the visual accuracy as well as the quantitative usefulness of any data generated from aggregate spatial statistics. Or, consider epidemiological case reports that are issued from local medical facilities, how does one account for the fact that their locations may change, or that new facilities may spring up or close down as time progresses. These are real-world problems that existing GIS platforms struggle to account for.

While there have been prior attempts to develop data models and applications for managing spatio-temporal data, the growing depth and complexity of scientific research has left room for improved systems which can take advantage of the highly interconnected datasets and spatial objects, which are common in this type of research. To that end, we have developed the *Trestle* data model and application, which leverage graph-based techniques for efficiently storing and querying complex spatio-temporal data. This system simple interface to allow users to perform query operations over time-varying spatial data and return logically valid information based on specific spatial and temporal constraints. This system is applicable to a number of GIS related projects, specifically those attempting to visualize historical public health indicators such as vaccination rates, or develop complex spatio-temporal models, such as malaria risk maps.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

TABLE OF CODE

TABLE OF EQUATIONS

ACKNOWLEDGEMENTS

A dissertation is a communal achievement. An accomplish not only for the author, but also for all those have played some role in its creation over the year. These past six years have been marked by the continual support of a myriad of people around me. Family who have supported me, a Church that has prayed for me, friends who have listed to me, and advisors who have not given up on me.

The opportunity to dedicate a significant portion of one's life to scientific research is indeed a privilege and one that should not be taken lightly. I believe that we owe a debt of gratitude to those who have sacrificed in order to see us succeed. The culmination of this doctoral work is merely the first step in a life of service and faithfulness towards that sacrifice. The list of those individuals includes, of course, family, who helped develop me into a seeker of knowledge and who gave me the safety and encouragement to pursue new things. My parents, Mark and Ingrid Robison have always encouraged in me in whatever crazy scheme I managed to dream up, and helped bring me back down to Earth into the reality of what's possible. I can never thank them enough.

I will always be eternally indebted to the congregation at *Emmanuel Anglican Church*, who have been my spiritual home and family during my time in Seattle. Perhaps no other community has had such a profound impact on my life. Many of those individuals have become deep friends and mentors, some became roommates, and one became my wife. Their constant prayers, encouragements, and acts of support have been truly overwhelming.

The other major community in my life is the department of *Biomedical Informatics and Health Education*. They are the ones who took a chance on me, provided the funding through the *National Library of Medicine*, and gave me opportunities to grow, learn, and thrive. All of the faculty and staff have been tremendously supportive, and I would not be where I am today without them. My advisor Neil Abernethy helped bring this project into fruition through his thoughtful guidance and razor-sharp focus on the core issues at hand. My committee members, Abraham Flaxman and Ian Painter have been incredibly generous with their time and expertise and I thank them for that.

Finally, I would like to acknowledge by department colleagues and the 2012 student cohort. Specifically, Nikhil Gopal who has served enumerable roles in both personal and professional contexts. Not only have a learned much from him, but I'm also deeply honored to call him my friend.

During my time in graduate school, I often had a copy of Raphael's *Saint George and the Dragon* on my desk. This was inspired by an conversation with a friend who introduced me to Steven Pressfield's *The War of Art* [1], in which he describes the process of writing his first screenplay, a process he describes as *slaying the dragon*. For him, the creation of his first work was the dragon that loomed over his life, the thing that terrified him, the great challenge that held him back and blocked the future. The only way forward was to struggle until it was defeated, and the future opened. The publication of this dissertation is the slaying of my own dragon.
Soli Deo Gloria.

[2]

DEDICATION

To Callie, Blythe and the ones yet to come.

Chapter 1. INTRODUCTION

In the beginning was the map. From the earliest points in recorded history, even before devising systems of language and numbering, humans have created methods of understanding the world around them and of placing themselves within the larger framework of existence [3]. As artifacts, maps tell us two major things about the world around us. How do we get where we mean to go, and what is our place in the world?

The fields of geography and *Geographic Information Science* (GIScience) have always been in tension with these two competing questions. The first question is about understanding the spatial dimensions of the world around us, and the second is about extracting some meaning and insight about ourselves and our relationships with others in the world around us. The first question is focused on data and its collection, integration, manipulation and presentation. This is the domain of technical geography and has crossed into other fields such as mathematics, computer science and ontology. Here, geographies are lines, points, and pixels, where geographic relationships are expressed in terms of computational topologies and categories [4]. The second is focused on people and narratives; not merely concerned with the data itself but with taking a higher-level view towards understanding the value and lessons from information to help further our understandings of social dynamics and environmental pressures. Here, geographies take on a more expansive, and at times, elusive, meaning, and serve as a lens by which human activity and interaction can be understood [5].

These two narratives, though at times standing in tension with one another [6], have contributed towards expanding the capacity and necessity of GIScience and related fields. As each new technical advance brings with it more available information and enables a deeper level of theoretical analysis; so too does the state of the theoretical geography advance further and ask questions that look beyond what is currently possible. In recent years, GIScience has turned its attention towards more fully integrating temporal information as a first-class citizen in information systems and data models in order to enable deeper insight into how communities, populations and environments change in the light of different environmental and social pressures. This desire is acutely manifest in the fields of public health and global epidemiology, where the emphasis on long term studies and complex interactions of health effects drives both theory and technology to the limits. But this new lens for viewing the world has brought with it a host of new challenges related to our ability to organize and understand time-varying spatial data. A challenge which existing GIScience applications struggle to account for.

The dawn of the computer age has served to fundamentally alter the field of technical geography and dramatically expand our ability to map and model the world. Beginning with the *Canada Geographic Information System* developed by a team lead by Roger Tomlinson the 1960s [7] cartography has largely shifted into the digital realm and brought with it paradigm altering ways of creating spatial products [8], [9]. In general, three major developments have occurred which have contributed to this shift. The first is the dramatic increase in both the types and quantity of available data. Traditionally, map generation has been the domain of governments and specialist organizations. Large observation teams have been assigned to manually survey and catalogue an area and then bring their findings to be analyzed and integrated at a later date. Limited in their production and level of detail, users have been forced to work with the available sets of maps created based on the experiences and observations of the survey teams.

Whereas available information was once made up of data observed second-hand by the map creators, new spatial datasets have been developed which gather information not merely from the one-time observers, but from the participants and residents in the areas themselves. Large projects such as OpenStreetMap[1] are built on the input and contributions of volunteer curators and are designed to be dynamic representations of the current state of the world, as observed and experienced by its participants.

Beyond improvements in map generation, new *Location Aware Technologies* (LAT) such as GPS transmitters and localizable radio signals have resulted in new types of available data that can be actively gathered from individuals in a near real-time manner [10]. Indeed, an irreducible facet of traditional cartography is that it is built on data that has been observed and collected prior to being loaded into digital systems; these new LATs allow for new methods of data collection that is accessible to users as soon as the information is generated [11]. This fundamentally represents a new way of observing and modeling the physical world and has helped drive towards new methods of environmental research termed *Geographic Information Observatories* [12].

The second major development is the increased level of availability of *Geographic Information Systems* (GIS). Massive advances in computing power and data storage has resulted in access to spatial data that no longer requires specialized computing hardware or expertise in software development in order to integrate a spatial lens into traditional scientific research. Likewise, open-source geospatial systems such as QGIS[2] and GRASS[3] have reduced the effective cost of entry to near zero, while efforts such as R-Spatial[4] and PySAL[5] have moved spatial analysis out of specialty applications directly into the scientific computing environments used by domain researchers.

The last major development is in data sharing and exchange. Developed in the age of the internet and ubiquitous data connectivity, technology projects such as MapServer[6] and commercial ventures like Mapbox[7] have developed easy and cost-effective methods of disseminating map products and datasets. This has helped accelerate by groups near the center of the spatial research community, the data curators. Organizations such as the *Florida Spatial Data Library* have introduced new data curation and sharing processes which has both enabled easier access to their datasets as well as facilitated faster updating and maintenance of their existing datasets in order to account for information changes [13].

These developments have put a tremendous amount of strain on traditional GIS applications and infrastructure. The *volume*, *type*, and *diversity* of data available for spatial researchers is, at best, an uneasy fit for traditional spatial applications. This is due, in part, to the fact that all of these new types of data and interactions are inherently temporal, whereas most GIS applications have been designed in light of the static map paradigm in which a single view of the world, at a single point in time, is augmented with additional layers of information. This has limited many researchers to developing statistical and visual models that assume either a static state of the world, or one

[1] https://www.openstreetmap.org/
[2] https://qgis.org/en/site/
[3] https://grass.osgeo.org
[4] http://r-spatial.org
[5] http://pysal.org/
[6] http://mapserver.org
[7] https://www.mapbox.com

modeled by a set of specific temporal snapshots [14]. This intersection of space and time represents the cutting edge of technical GIScience research and will be the focus of this dissertation going forward.

1.1 Space and Time Representations in GIScience

After the initial effort of conceptualizing the visible world, comes the desire and necessity to ensure this information remains current and useful. With our deepening understanding of the connectedness of the world and our ever-increasing access to information, traditional methods of drawing and distributing maps soon become insufficient. This is due either to the limited amount of information that can be disseminated in a single image (which will be discussed further in Chapter 3), or due to the speed at which these maps can be updated to reflect to the state of the world around them. One dramatic example of this later challenge is recorded by the historian Marc Bloch during his experience in the French army at the outbreak of World War II. The glacial pace of information exchange between the various military units, combined with the rapid onslaught of the German army, which was far in excess of the speed of movement from previous conflicts, rendered their maps, spatial awareness and methods of information dissemination nearly obsolete and so contributed to the rapid collapse of the Grand Army [15].

An additional argument for the importance of integrating time and space, is that social geography has largely focused on uncovering the dynamics and influences of the human environment, which inherently requires understanding the immediate context of the point of observation, an inherently temporal problem. As is often the case, theoretical geography has been a driving force behind technical change [12]; in this case the integration of space and time within the field of GIScience finds itself rooted in the theory of *time-geography*. Initially described by Hägerstrand in his seminal essay *The Two Vistas*, our deepening understanding of the universe and its prevailing laws has resulted in a picture of the universe framed as a serious of detached observations and loosely coupled connections between physical processes, but with little to no grounded framework for understanding the dynamic nature of the Universe [16].

This creates a challenge when trying to understand the deeper dynamics that drive processes and interactions and can dramatically cripple the usefulness of spatial analysis when applied to areas of policymaking or evaluation [17]. Thus, time geography aims to develop a general foundation for theory building in a spatial context through recognizing the crucial importance of time-based analysis [18], [19]. From this theoretical outlook originates domain-specific distillations; each scientific domain attempts to apply this geographic framework to its own field of inquiry, reckoning with the ideas that place and nearness matter and that the impacts of these concepts changes over time with dramatic impacts on the question being answered [18].

Despite how glaring the gap in philosophical methods appears, and regardless of the depth of desire to bridge the space between observations; until recently, it has remained beyond the reach of the general research community. Now however, time geography, and its applied descendant spatio-temporal analysis, have come to the forefront of the geographic research agenda and work is actively being done to develop both the theories and implementation models for effectively working with large volumes of time varying spatial data, in a number of scientific domains.

While spatio-temporal analysis is a general problem with implications across multiple scientific and policy domains, for the remainder of this dissertation we will focus on one domain in particular which represents a number of opportunities for designing, testing and implementing time geographic techniques, Public Health.

1.2 THE RELEVANCE OF GISCIENCE IN PUBLIC HEALTH

From its earliest days, the field of public health has been acutely aware of its spatial dimensions. In 1905, the US Supreme Court decided what is considered to be the foundational legal precedent for public health law. *Jacobson vs Massachusetts* [20] involved a personal objection to recently passed law enacting compulsory vaccination against Small Pox. While the US has a strong tradition of prohibiting intrusion into the private domain of the individual[8] both the law and the resulting judicial holding enshrined the idea that living in proximity to other individuals carries with it both the risk of infection and a duty to prevent harm that can be enacted by the state, even at the expense of other personal privacy protects [21]. In a phrase space trumps self. But this consideration of environment is not merely limited to legal proceedings. The field of epidemiology contains its own founding narrative where John Snow (considered the father of modern epidemiology) traced the epicenter of the 1854 cholera epidemic in the city of London to a community water pump on Broad Street. The evidence for this discovery (which resulted in the infamous removal of the handle of the pump [22]) was based on in-home visits of every cholera death in South London and supported the assertion that not only was cholera water born, but that the spatial layout of the deaths showed the use of a common water source [23], [24].

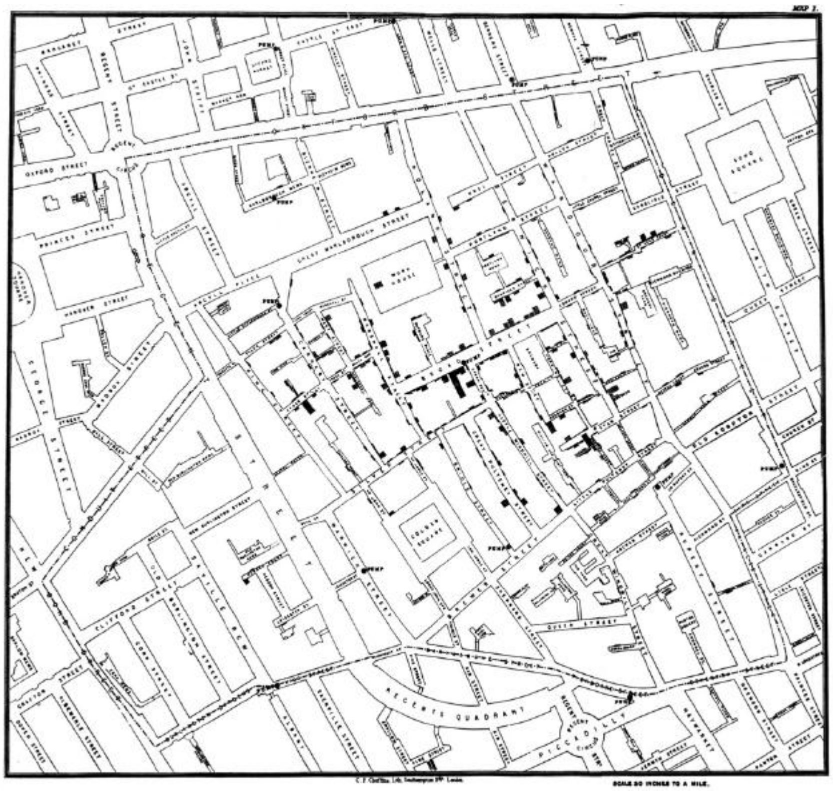


Figure 1: John Snow's map of the London Cholera Epidemic [Public Domain Image].

[8] Restatement (Second) of Torts § 652D (1977) cited in [191].

More broadly, public health has carried an understanding of the significance of place and environment, with the idea that an individual's external environment strongly influences their susceptibility to disease [25]. Encoding this additional context is challenging, but of the utmost importance for public health. As opposed to other medical disciplines, the unique nature of public health is that its broad view of health and society includes a concern for the factors and events that occurred in an individual's past, due to the long-tail impacts of environmental and sociological factors [26]. It is then no surprise that a large amount of attention has been paid to improving the use of, and access to, robust spatial datasets that give researchers and policy makers deep insight into the context and behaviors of the populations under their stewardship [27].

But that is not the only concern. An additional uniqueness of Public Health is that it is largely a function of governments and administrative entities. Given this lens, it becomes difficult to divorce the scientific side of public health from the policy and administrative component. Indeed, it is these two pieces together that have led to a focus on *evidence-based policy* making [28], [29]. Given this policy bent, public health science has often carried an interest in monitoring and evaluating the effectiveness of health policies and implementations over time. These policies and interventions are often focused around administrative districts that are subject to delineating forces outside of the health domain [30]. Understanding the shifts in these human delineations is critical if we are to accurately evaluate and understand the policy implications and outcomes of various health interventions.

Consider, as an example, a model of vaccine coverage for the administrative regions of Sudan over the past 30 years, a research frame common in public health analysis. In attempting to develop this model a researcher is confronted with the issue that in the midst of violence civil war, Sudan is partitioned into two new countries. The northern part retains the name Sudan, while the southern part becomes South Sudan. This leads to the question as to how to map and aggregate data from this region, which has a corollary impact on both the visual accuracy (is a user supposed to trust the validity of a model that shows incorrect geography?) as well as the quantitative usefulness of any data generated from aggregate spatial statistics.

Additionally, consider epidemiological case reports that are issued from local medical facilities, how does one account for the fact that their locations may change, new facilities may spring up or close down, or that populations may dramatically shift as time progresses. These are common research problems that cut to the core of the issue at hand. Robust policy evaluation and health research requires multiple years' worth of information. Within that time frame, human institutions and organizations can undergo tremendous amounts of change which may dramatically affect the environmental and policy context of the individuals. In order to gain actionable insights, researchers must find ways to understand and account for this temporal variation.

1.3 DISSERTATION GOALS

This project aims to improve upon this area of study by developing a unified system for storing, integrating, and querying complex geospatial data, specifically focused on international administrative boundaries. The goal of this system is to develop a data model and management application which will allow users to perform a specific set of query operations over a historically integrated set of global administrative boundaries and return logically valid data based on specific spatial and temporal constraints. This system will be applicable to a number of GIS applications, specifically applications attempting to visualize historical public health indicators such as vaccination rates, or develop complex spatio-temporal models, such as malaria risk maps [31].

1.4 OUTLINE AND CONTRIBUTIONS

This dissertation will approach the outlined problem in three major phases. First, a we review an evaluation of existing spatio-temporal data modeling efforts in this field. Next, we describe the design and implementation of the *Trestle* data storage and query application which serves as a reference implementation for the proposed data model. Finally, we perform series of evaluations to demonstrate both the usability of the underlying data model, as well as the ability to design domain specific applications on top of the generic data model. These three phases serve to fulfill the research aims of this dissertation:

- *Aim 1: Design and implement a unified method for automatically building and managing spatial objects from complex spatio-temporal data.*
- *Aim 2: Develop a graph-based approach for integrating spatial objects from historical records using a global dataset.*
- *Aim 3: Demonstrate using the query interface for solving a common challenge of generating custom research geographies for public health research.*

The outlines of the individual chapters are given here (excluding Chapter 1, which serves as the overall introduction):

Chapter 2 provides an introduction into some specific challenges related to spatio-temporal research within the field of global health, specifically, issues related to the spatial and temporal context of the geographic area being studied. We also introduce a set of desiderata for any proposed spatio-temporal data model or GIS application.

Chapter 3 introduces existing spatio-temporal data models and provides a technical evaluation framework for determining both the strengths of the existing methods, as well as potential areas for improvement. In addition, each data model is evaluated using the framework outlined in Chapter 2.

Chapter 4 details *Trestle*, the graph-based spatio-temporal data model and management application at the core of this dissertation project. This chapter details the spatio-temporal ontology and semantic reasoner approach that was chosen in order to address some of the limitations of existing data models. In addition, we evaluate the proposed data model using the framework outlined in Chapter 3. Finally, we describe a prototype data management application which enables usage of the data model. This chapter fulfills *Aim 1*.

Chapter 5 begins the evaluation component of the dissertation by focusing on the first of two evaluation processes. The first is to validate the ability of the data model and management application to effectively represent relationships between spatial objects with a focus on their change over time. This will be accomplished through the design and implementation of a prototype algorithm to automatically identify spatial changes in a global administrative unit database. A challenging issue in public health research. This chapter fulfills *Aim 2*.

Chapter 6 concludes the evaluation phase of the dissertation by testing the ability of the data model to effectively represent the internal state of time-varying spatial objects as required in order to address common challenges in public health research. This is achieved by describing several common algorithms for generating custom research geographies, which require valid data at the point in time that the geographies are generated. This chapter describes how these types of algorithms can be effectively implemented within the data model, as well as detailing algorithm improvements that are enabled by access to the types of data contained in the spatial data model. This chapter fulfills *Aim 3*.

Chapter 7 provides the dissertation conclusion and describes a summary of the results and contributions, as well as potential next steps for the research work.

Overall, this project has several scientific contributions:

- A spatio-temporal data model designed to leverage graph data layout and storage systems to represent complex temporal changes in objects in a unified framework.
- A data management and query application which abstracts the complexities of underlying temporal logic to present a unified method of data interaction for the end user. This application is designed to support real-world research projects within existing data workflows.
- A graph-based algorithm for analyzing spatial and temporal interactions between administrative districts.

- A temporally integrated dataset of a subset of county level districts in sub-Saharan Africa
- Description of potential methods for improving existing algorithms designed to aid researchers in developing custom geographies for specific research outcomes.

1.5 AUDIENCE

The primary audience for this dissertation are designers and implementers of geographic information systems looking to more fully integrate a temporal component either into existing spatial data management applications or new systems designed from the ground up to support time as a first-class citizen. In addition, the use cases and final aim deliverables are applicable to global health researchers, specifically those involved in longitudinal studies of public health interventions or large-scale spatial epidemiology.

Chapter 2. TEMPORAL CHALLENGES IN GIS

In the previous chapter, we outlined broad currents pertaining to the integration of time within spatial and health research. In this chapter, we will take a deeper look at existing challenges facing public health researchers in their day to day work. Specifically, we will look through the lens of public health research in both domestic and global contexts.

Section 2.1 will discuss a major methodological challenge related to integrating time in spatial analysis, namely the *uncertain geographic context* problem, which traditional GIS applications struggle to effectively account for. Section 2.2 will introduce an extension to this problem, which we have termed the *uncertain temporal context* problem. This will be illustrated through two major motivating use cases, epidemiological disease surveillance, and population movement patterns in natural disasters. Section 2.3 will conclude the chapter by summarizing these challenges into a set of concrete desiderata for new GIS applications aiming to improve upon the state of the art, which will form the foundation for the remainder of the work in this dissertation.

While the core of this dissertation is focused on global epidemiology and its related challenges, this chapter has a significant focus on the types of data common in domestic (United States) and behavioral public health. The reason for this is that the amount of data available in a domestic setting is both *broad* (large amounts of data available for numerous indicators) and *deep* (available at multiple spatial granularities, down to city block and below). This enables us to illustrate methodology challenges that are both common in domestic public health, as well as emerging within the global space. There is nothing here that is unique to domestic public health but the types of studies being performed may not always be possible in a global context, due to a lack of necessary data. As more and more information is available on a global scale the issues described here will become more apparent in non-domestic contexts.

2.1 UNCERTAIN GEOGRAPHIC CONTEXT

Advances in spatial analysis within public health research have further understanding of the causal factors contributing to various health conditions [27]. Beyond simply detailing observations and extrapolating trends, researchers are interested in understanding what contributes to various health states and outcomes, and how these contributions differ across populations.

In 2012, Kwan was the first to detail a subtle, but profound, methodological challenge related to achieving these research goals and labeled it *The Uncertain Geographic Context Problem* (UGCoP) [32]. The question is simple, what defines an individual's environment? But the answer is remarkably difficult. A 2011 systematic review of environmental studies of cardio-metabolic risk factors found that nearly 90% of them defined environment as exclusively the residential neighborhood of the individual [33], that is, where their home address was. But while researchers have developed fairly robust theoretical support for using neighborhoods as the focal point of health research [34], other work has suggested that delineating by neighborhoods or other administrative districts (such as zip codes or census blocks) is not the ideal method for determining exposure risks [35]–[37][9].

[9] This challenge identifying appropriate geographic delineations for research purposes will be discussed further in Chapter 6.

One reason for this is that individuals exhibit a high degree of spatial freedom; they tend to frequent not just their homes and but also their schools, work environments, coffee shops, church and parks. While some of these locations may be within the home neighborhood, others may be a significant distance away. A 2008 study of movement patterns in adolescents found that over 20% of their time was spent more than 1km away from home [38]. Likewise, Basta, Richmond, and Wiebe found that the majority of study participants defined their neighborhood as being more expansive than what would normally be delineated using the census tract of their primary residence. Correspondingly individuals spent nearly 92% of their time outside those traditional boundaries [39]. But while researchers have been able to determine the limits of existing methodologies, proposing alternatives has proven to be more difficult and requires significantly different approaches and data requirements than previous approaches.

Within this same vein, researchers have developed new methods for measuring an individual's environment exposure by using their actual movement patterns to construct an exposure map [33], [40]. In short, an individual's environment is defined as where they are, for a specific duration of time; environment has thus become an inherently fluid concept and requires researchers to account for not only a single individual, but any other individuals and entities that they may have come in contact with during the duration of the study. In light of this, the concept of environment shifts from a point-exposure problem to one of network analysis, in which an individual is represented as a node in a continually evolving relational graph of interactions and influences that changes over time.

While network analysis has shown tremendous promise in fields such as epidemiology [41]–[43], when applied to other health domains it requires both new conceptual models [44] as well as new methods for integrating different types of data into GIS applications [45]. These new methods require applications which have the ability, at each time interval, to answer the questions; *what exposures exist at these locations*? And, *what other individuals might be in contact with me?* In addition, these new methods are affected by traditional statistical challenges, such as the *modifiable areal unit problem* (which will be covered in Section 0), and limited data resolution. These challenges will be discussed in more detail later on but, at this point, suffice it to say that all of these issues make time series analysis an un-easy fit for existing GIS applications which are not only limited in their ability to manage large amounts of temporal data, but also in modeling changes in an individual's state over time.

2.2 Uncertain Temporal Context

While UGCoP includes a temporal component when considering the effects of exposure and location, what is missing is the temporal variation of other entities within the window of analysis. That is, not only where in the world an individual finds themselves, but also what state the world is in when they find it. In this section, we will look at two research efforts which acutely illustrate both the challenge itself and various efforts to mitigate its effects.

2.2.1 *Disease Surveillance*

While Section 2.1 focused on the necessity of developing a detailed understanding of the specific environment of a single population, there are many situations in which such an analysis is not possible. For environments in which researchers have limited access to fine grained location observations or survey data, they are often forced to resort to high level groups and aggregations;

in the global health context, it is common to utilize administrative districts as the unit of analysis. One reason for this, is that the previously mentioned lack of data may prohibit analysis at any finer resolution. When attempting to study long-term trends in public health indicators, researchers are often limited to national surveys undertaken at multi-year intervals, which have a minimum level of spatial resolution. As examples, for their 2009 study of improvements in bed net distribution in Africa, Noor, et. al used province and state level districts, which was the lowest level enabled by the sampling methods of the survey instruments being utilized [46]. Likewise, a 2011 study by van Eijk, et. al, regarding changes in malaria protection in pregnant women, utilized the same level of administrative delineation for the majority of the study, except for three countries (Nigeria, Tanzania, and Madagascar) in which they were able to use county level district [47]. Two other studies reviewed made direct mention of preferring the lowest-level (most granular) data available, but largely settled for higher level aggregations (such as at the state level) due to the dearth of sub-national sources [48], [49]. One challenge with this approach, is that it runs headlong into a statistical complication known as the *Change of support problem (CoSP).*

2.2.1.1 The Change of Support Problem

The *CoSP* is a general statistical problem related to the gathering and interpretation of multiple data sources, which may have been collected, and then analyzed, at different spatial or temporal resolutions [50]. This describes the phenomenon that when the spatial or temporal frame (scope) of a given research project increases (e.g. to cover a larger geographic space or a longer span of time) the *support* associated with each data value decreases. This term s*upport* refers to the size or volume associated with each data value, in relation to the research frame as a whole [51]. For example, consider an air quality monitoring station that records data at hourly intervals. Each value may have a high degree of support as representing the *true* air quality for that point in space and time, and perhaps for the immediately surrounding area; but when considering both the larger geographic area (such as a city or county) and broader temporal scope (such a monthly or annual values) the level of support that each value represents the true value of the air quality diminishes as each value effectively describes less and less of the total spatial and temporal frame. Within spatial research, this issue manifests itself in three major ways: when combining data with differing spatial and temporal resolutions, when collecting data in one spatial format (e.g. points, lines, images) and analyzing it in another format, and finally when aggregating data from one level of resolution into another.

The first way in which the CoSP manifests itself is when multiple datasets, collected a different spatial and temporal scales, are combined together to produce a final result. This is extremely common in public health which is often concerned with complex interactions between biological and environmental factors. An excellent example discussed further in Section 2.2.1.2, comes from the field of epidemiology specifically in attempting to develop maps of malaria risk through sub-Saharan Africa. In order to do so, multiple climate, wildlife, and topographic datasets are combined together to produce the final output. One challenge with this approach is that since the data is gathered at different spatial and temporal resolutions, each dataset provides a differing level of *support* for its data values. This means that the datasets cannot be accurately combined without some mechanism for accounting for these different data scales and levels of support [50].

The second way in which the CoSP comes into effect is when data is collected in one spatial format (e.g. as points, areas, or images) and then analyzed in a different format. In the air quality example, given above, each recorded value describes that specific point in space and time; however, researchers often desire to (or are required to) perform analysis over a geographic area. For example, a researcher attempting to determine a correlation between area quality and health outcomes may attempt to do so using data collected at the city neighborhood level which may encompass one or more air quality measurement stations (or none at all). Given that these two datasets encompass different spatial and temporal scales they cannot be directly combined together without making a decision as to how to compensate for the differing levels of support. This issue is particularly acute in public health research due to the fact that disease is specific to an individual, but environment varies over a continuum [50]; meaning that the CoSP will be a factor in nearly all geospatial public health research. Table 1 gives a list of common geospatial research goals which require mixing data collection and analysis formats.

Spatial format of data collection	Spatial format of data analysis	Examples
Point	Point	Point kriging; prediction of under sampled variables
Area	Point	Ecological inference; quadrat counts
Point	Line	Contouring
Point	Area	Use of areal centroids; spatial smoothing; block kriging
Area	Area	The MAUP; areal interpretation; incompatible/misaligned zones.
Point	Surface	Trend surface analysis; environmental monitoring; exposure assessment
Area	Surface	Remote sensing; multiresolution images; image analysis

Table 1: Examples geospatial research requiring use of different spatial data formats.

This table lists examples of *Change of Support Problems* (CoSPs) encountered when analyzing spatial data. The first column corresponds to the spatial format of the data values being collected, while the second column shows the spatial format of the data analysis, which may be different from collection formation. The third column gives examples of geospatial research which necessitates mixing collection and analysis formats. (Reproduced from [50]).

The final way in which the *CoSP* occur is when data that is collected in the same spatial format is aggregated, potentially with other data sources, to produce a new dataset which may cover a larger spatial or temporal frame, but at a lower level of granularity. This is known as the *Modifiable Areal Unit Problem* (MAUP) and relates to the fact that the sensitivity of analytical results is directly related to the spatial scale and layout in which data the data is collected and analyzed [52] [10]. While this issue is additionally present in other statistical fields [53], from a geographic perspective, it manifests in two primary ways [54]:

1. *Scale effect* –variation in numerical results resulting strictly from the number of areal units used in the analysis of a given area.
2. *Zonation effect* –changes in numerical results resulting strictly from the manner in which a larger number of smaller areal units are grouped into a smaller number of larger areal units.

These two effects are illustrated in Figure 2 and will be described in the remainder of this subsection. The *zonation effect* refers to the fact that when multiple geographic units are aggregated together, variability in the data tends to be masked. This results from that idea that for many phenomena there is a natural *scale* at which the effects are observed and increasing or

[10] The converse of this effect, where is often referred to as the *ecological fallacy* and is common when data collected about a group of individuals (or a geographic area) is used to infer patterns in individuals (or a smaller geographic area) [50].

decreasing that scale obscures the effect either by sampling at too granular a level, and thus not observing the phenomena; or aggregating these observations into larger units and obscuring its true effects. This is a common challenge in public health research in that health effects of locality (such as proximity to quality produce, interactions of air pollution and neighborhood activity level) may only be observable at a neighborhood level and when aggregated to a city or state level, the interactions may no longer be apparent. This challenge will become more and more critical as researchers gain the ability to generate data at the neighborhood and individual level but are required to report and analysis data at the level of the city or the county.

The second effect, the *scale effect,* manifests when utilizing various methods for aggregating data into larger units and integrating data collected at different scales which can result in changing conclusions based on different combinations of aggregations [52], [54], [55]. While this issue is germane to multiple geographic research domains, it is particularly acute when the methods of data collection and aggregation are unrelated to each other; such as is the case with census data which may be gathered via a number of different sampling methods (such as vital record reports or door-to-door sampling) but then aggregated based on political delineations such a census tracts or administrative districts [56], [57]. This is often experienced when performing longitudinal studies of public health indicators, or when data is not available at a fined-grained level, requiring the researcher to choose existing aggregations with may not be suitable for the task at hand. The solution to this problem is manual integration and normalization by the researcher and is challenging to address in an effective manner. This discrepancy between collection and analysis will be discussed further in Chapter 6.

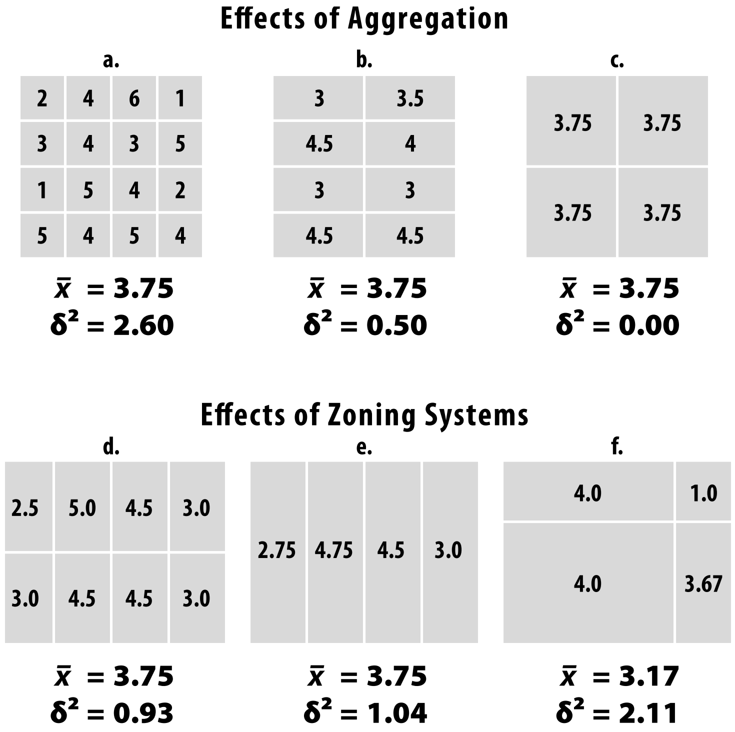


Figure 2: The effects of zonation on analysis results (Reproduced from [55]).

This figure illustrates the *scale effect* (figures a-c) where based on the number of regions being aggregated the variance (δ^2) of the result changes, whereas the mean value ($\bar{x}$) remains the same. The *zonation effect* is illustrated in figures d-f, where different orientations are used to aggregate the regions resulting in both the mean and the variance changing for each orientation.

One common method for managing the MAUP issues, is to utilize a surveillance system focused around stable sentinel sites located within specific geographic regions. These sites can then undergo periodic sampling in order to develop a longitudinal view of the population. This is the approach taken by the *DHS program*[11] from the *US Agency for International Development.* In brief, the DHS program routinely executes a series of surveys at various sentinel sites across the globe and compiles the collected results, which are then made available on their public website. These surveillance sites provide sets of historical observations that can easily be normalized in the face of any changes in population dynamics or administrative redistricting, since the survey is focused on a single population catchment area and can normalize for any changes in population size or composition. The DHS program is a common way to gain detailed survey information about populations and provides reliable sentinel sites for evaluating different public health interventions. Its data has been used by several research projects [58]–[60]; however, one challenge with this approach is that it may suffer from limited generalizability [60]. The environments of the individual sentinel sites may be so unique, even within a relatively constrained geographic scope such as an administrative province, that any resulting data may not be applicable outside that context.

Given these challenges, many research projects continue be focused on analyzing data sampled from administrative districts, and not only for data quality reasons. As was briefly mentioned in Chapter 1, administrative districts serve a number of functions, not just for the purposes of census or population delineation, but also as centers of governance, policy, and resource allocation. Political systems must make decisions on the basis of some type of delineation and this delineation is most commonly some type of district resembling a county or state. These districts can then serve as natural experiments for different policy implementations or evaluations.

But this introduces a new set of challenges, what happens when districts are changed during the course of research study? This is the central, motivating use case for the *Trestle* system, which was born out of the *Scalable Data Integration for Disease Surveillance* (SDIDS) project [61]. A joint effort between the University of Washington and McGill University, SDIDS was a project which aimed to integrate disparate health related data sources for decision making around malaria modeling. One major challenge for the SDIDS project was the fact that over time, administrative districts were changed, renamed, created, dissolved and otherwise modified. This presented a unique challenge when trying to integrate historical data from different sources. This issue is clearly illustrated in Figure 3. Here, we have two views of the *Demographic Republic of Congo* (DRC) as delineated by the UN *Global Administrative Units Layers* dataset[12] [62], one for the year 2009 and one for the year 2014. In 2009 there were 48 administrative districts at the 2nd administrative level[13], while in 2014 that number had grown to 89. *How should a researcher compensate for these changes*? *How many of these changes represent simple boundary movements, how many from county dividing into multiple new ones, and how many changes were more complex, with one county being split apart into pieces of other counties?* And perhaps most

[11] https://dhsprogram.com

[12] Details on this specific dataset are given in Chapter 5.

[13] This level corresponds to the county or parish unit in the United States.

pressingly, *how to make temporally valid inferences about health indicator changes for a spatial area?*

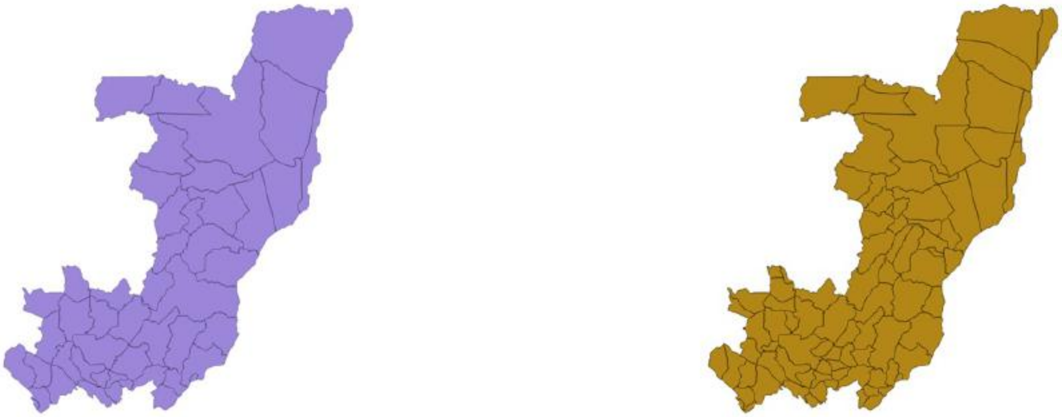


Figure 3: DRC districts in 1990 vs 2014.

This illustrates the amount of district reorganization that can occur within a single country over a period of time.

2.2.1.2 Rasterization in disease surveillance

One approach for addressing this issue is *rasterization*. In this process, a polygon representing the affected area of a given indicator (e.g. population count, vaccinate rates, income distribution, etc.) is divided into a number of gridded *cells* and each cell is assigned a portion of the total indicator value. As an example, imagine an administrative district as perfect square of 3 km x 3 km, with a population count of 90,000 people and that the desired output is a rasterized population map with a resolution of 1 km x 1 km. The simplest approach is to divide the district into 9 squares and distribute the total population count evenly amongst each square. With this approach, we can generate a rasterized population image in which each cell represents 10,000 people. This methodology has been used by projects such as the *Gridded Population of the World* [63] and variations of the same approach, using different value distribution methods, has been used by other efforts such as the *Global Burden of Disease* [64], *LandScan* [65], *WorldPop* [66], and the *Malaria Atlas Project* [67].

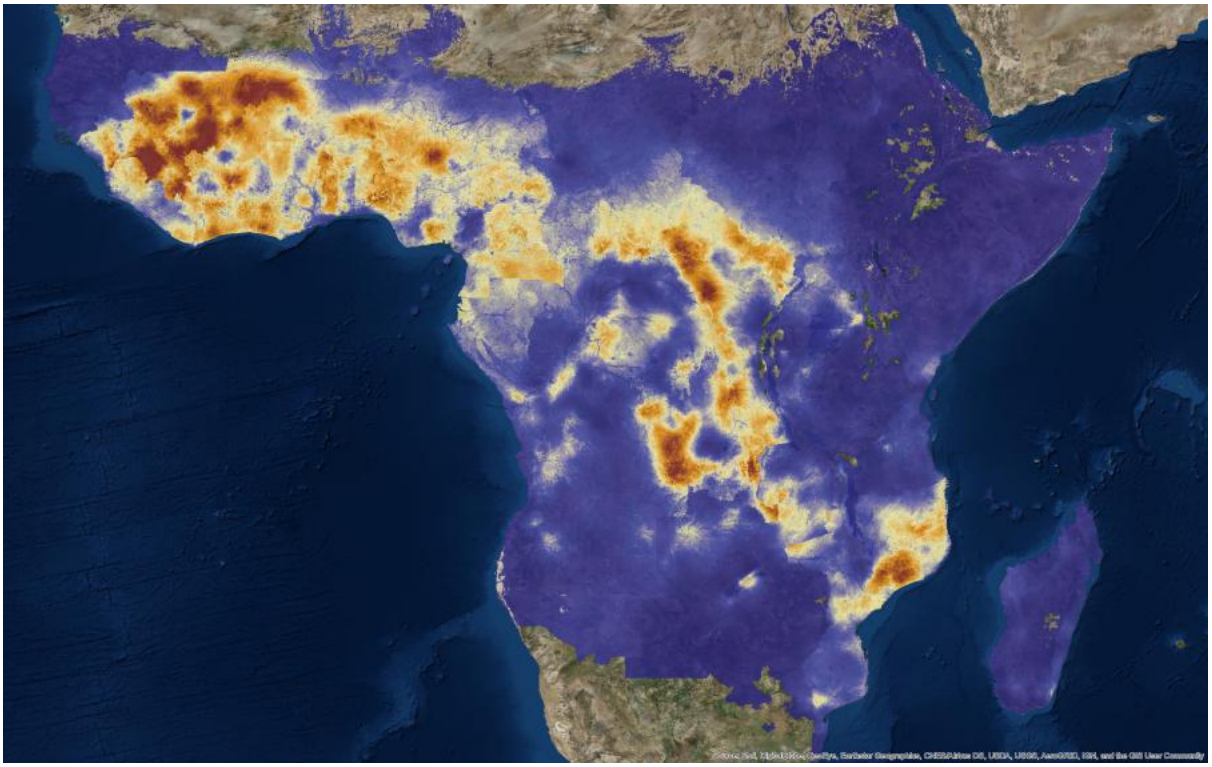

Source: Esri, DigitalGlobe, GeoEye, Earthstar Geographics, CNES/Airbus DS, USDA, USGS, AeroGRID, IGN, and the GIS User Community

Figure 4: Rasterized view of malaria risk for Africa in 2015. Generated from data provided by [31].

This approach has the benefit of being able to directly compare indicator changes over time and normalize temporal changes amongst geographic features such as rivers and census tracts. Through various automated and semi-automated techniques, the accuracy of these approaches can be very high [68]. While the resulting spatial products can be of tremendous utility, they suffer from two primary drawbacks which are of particular interest when viewed through the lens of a temporal GIS.

The first drawback to rasterization is that generating these products is a complex and time-consuming process that requires the integration of a number of distinct raster and vector data sources which must be manually collected and integrated. An initial review of 11 papers focused on malaria risk mapping revealed the use of 28 distinct data sources (details given in Table 2). These data sources can be broadly categorized into 6 major groups: climate (rainfall, cloud cover, mean temperature, etc.), malaria prevalence, map features (roads, rivers, cities, etc.), population, geography (land cover, elevation, etc.), and wildlife (biodiversity estimates). While each of the papers cited a different combination of sources, they were all drawn from these major categories. Within these categories, each dataset features a unique combination of temporal and spatial resolution. These differing resolutions often require the research to make a judgement call as to how to aggregate the various data sources into a common spatial and temporal extant and to apply some additional type of adjustment to compensate for the differences in information resolution. This decision may vary greatly between researchers and projects.

For example, consider a malaria model attempting to utilize the datasets listed in Table 2; the USGS *Digital Elevation Model (DEM)* [69] provides global elevation coverage at a spatial resolution of 1km^2, but with a varying temporal validity based on the orbital patterns of the collecting satellites (which can be on the order of weeks or months). In contrast, the *Famine Early Warning System* [70] monitors daily rainfall numbers, but on a coarser 8km^2 scale. Integrating these data sources requires deciding on a common temporal and spatial resolution to perform the analysis (for example, aggregating the DEM information into an 8-km grid, and the *Famine* data into monthly units).

While the relative ease of access to these and other datasets has led to a huge growth in GIS modeling for disease risk, it has placed a tremendous burden on the individual researcher to cope with the varying temporal/spatial scales, and each model, while perhaps sharing some common datasets, is largely an individual data curation effort and even something as simple as updating the model for a new time period requires significant background work to source new datasets that cover the new temporal interval. This approach has been shown to be quite effective but precludes any automated risk mapping or even easy cross-validation of other modeling approaches.

Data source[14]	Type	Temporal Resolution	Spatial Resolution	Used by:
Topographic and Climate Data Base for Africa[71]	Climate	Monthly	0.05-degree grid	[72]–[75]
Normalized Difference Vegetation Index[76]	Climate	Monthly	0.25 * 0.25 km^2	[72], [74], [75], [77]–[81]
MARA/ARMA	Prevalence	Annual	Varies	[72], [78]
MODIS Landcover[82]	Climate	-	1 * 1 km^2	[72], [75], [77]
African Data Sampler[83]	Features	-	-	[72], [74], [75], [78]
African Population Database[84]	Population	-	3.7 * 4.8 km^2	[72], [73]
Land Surface Temperature	Climate	8 days	1 * 1 km^2	[77]–[81]
Famine Early Warning System Network, Daily Rainfall[70]	Climate	Daily	8 * 8 km^2	[77], [79]–[81]
Digital Elevation Model[69]	Geography	Static	1 * 1 km^2	[77]–[81]
Landscan[85]	Population	Static	0.5 * 0.5 km^2	[77], [86]
HealthMapper Geographic Features[87]	Features		-	[77], [79]–[81]
Malaria Season Length[72]	Composite Model	-	-	[74], [75]
Global Resource Information Database[88]	Features	-	Varies	[78]
Cold Cloud Duration[89]	Climate	10 day	8 * 8 km^2	[78]
Africover[90]	Geography	Static	1 * 1 km^2	[78]
Malaria Atlas Project[31]	Prevalence			[86]
Walter Reed Biosystematic Unit[91]	Wildlife	Static	-	[86]
Global Biodiversity Information Facility[92]	Wildlife	Static	-	[86]

Table 2: Summary of datasets utilized in Malaria Risk Assessment studies.

This table lists the various datasets used by the malaria studies reviewed for this dissertation. Each dataset is listed along with the *type* of the dataset as determined by our classification. Likewise, the spatial and temporal resolution is given for each dataset, as well as the reference to the study it is utilized in.

The second drawback to rasterization is that building raster products still requires accounting for the temporal variation of the spatial entities. Consider, as an example, an attempt to utilize a population estimate for determining vaccine coverage rates for a given district. This estimate may have been collected 5 years ago and simply noted as *collected in District A*, how is a researcher to know if what we call District A today, is identical to what was called District A 5 years ago. The previous example of the DRC (Figure 3) shows that even 5 years can be a long period of time in the world of political redistricting. So, while the end product of rasterization may help researchers compensative for the tremendous amounts of temporal variability in spatial datasets, building these products requires addressing the problem head on.

2.2.2 *Population Movements During Disasters*

In 2010 a massive earthquake devastated the country of Haiti and left nearly 1.8 million people homeless [93]. In the immediate aftermath of the disaster, aid agencies struggled to account for the tremendous upheaval of the population and identify new clusters of individuals which congregated around functioning resource centers, as the situation on the ground evolved [94]. After

[14] Several of the papers utilized their own survey instruments to determine the prevalence information which is not cited in this table. Only references to external surveys or datasets are listed here.

the initial response, a number of white papers and research projects sprang up which proposed using cellphone call records to model population movements and extract actionable insights for relief workers [95], [96]. One project, undertaken in 2011 as a collaborative effort between the Karolinska Institute, Columbia University and Digicel Haiti collected cell phone records for 2.8 million individuals from 6 weeks leading up to the earthquake to 5 months after [93]. Their goal was to determine if pre-disaster movement patterns could be used to predict population movements in the immediate aftermath of a large-scale natural disaster. For their study, each call record (of which nearly 280 million were recorded) was linked to the geographic location of the cell tower servicing the connection. From there, the researchers compared the predictability of an individual's movement pattern before and after the earthquake. They found that movement patterns remained fairly predictable and were highly correlated with patterns observed before the disaster [97]. Thus they concluded that call record data could have tremendous utility in improving the efficiency of aid agencies responding to a natural disaster.

While the usefulness of this analysis should not be underestimated, the limitations of the underlying dataset present some issues with attempting to generalize the findings. In the study, the researchers used a map of cell towers, supplied by the telecom provider, in order to geolocate the call records. This immediately raises the questions, *were all towers functioning in the immediate aftermath of the earthquake? Did any reach the point of network saturation and offload calls to another tower? Where any disabled due to physical damage or loss of power?* These are critical to answer, especially when using a coarse grain spatial unit such as cell tower coverage area, which the study authors state can range from less than 100 meters in urban areas, to greater than 10 kilometers in more rural regions [97]. In order to robustly fulfill the stated research goal, what is needed is an accurate dataset which describes the state of the cell network at each time interval, without this, researchers are left to speculate as to the true state of the world, beyond what their single, static snapshot described.

2.2.2.1 Defining the Uncertain Temporal Context

Both of the use cases described in this section illustrate a methodological problem that goes beyond what has been described in the UGCoP literature; which is the problem of modeling the state of multiple spatial entities, which may undergo significant amounts of change over the course of a research study. We refer to this problem as the *uncertain temporal context problem* (UTCoP); which describes the uncertainty that develops when utilizing snapshot datasets (such as a fixed map of cellphone towers, or a single list of administrative districts) to perform spatio-temporal analysis. Like the contextual challenges mentioned in the preceding section, UTCoP has the potential for dramatically influencing any spatial analysis that occurs over an appreciable length of time. As will be further discussed in Chapter 5, the GAUL dataset recorded that nearly 71% of county level districts in Nigeria were re-organized at some point between 1990 and 2014.

In the future, this challenge will only grow more acute. The explosion of available data, such a geo-located social media posts [98], [99] and the relative ease of analysis means that researchers are continually finding new ways of integrating disparate datasets in order to provide a more comprehensive view of the world and its inhabitants. Unless these datasets are built and curated with a specific view towards historical accuracy, such as carefully versioning any changes or creating mappings between old and new identifiers, researchers will have to bear the full burden of fully accounting for these changes and normalizing any discrepancies or intrinsic biases.

Because of these limitations, researchers are often forced to either ignore these temporal shifts [100], or manually reconcile the data to the best of their abilities. This manual reconciliation not only raises the specter of errors introduced in the process, but also presents a challenge around reproducibility. If each dataset is manually constructed, corrected, and integrated, that process becomes a part of the scientific record and any omissions to the record threatens to reduce the ability to validate results using other methodologies, or update the study using data sources that may only be available at a later date.

One final point on this issue relates not merely to changes in the spatial context that have occurred in the past, but also changes that occur due to our ability to continually generate more and more accurate spatial maps [101]. Tremendous advances in spatial measurement and imaging technologies means that any retrospective study (or attempt at reproducing prior work) has to account for the fact that new spatial datasets may differ considerably from previous products merely due to improvements in technology. Thus, the question of determining the appropriate spatial and temporal context is not limited to merely asking *what was the state of the world on June 9th, 2011?* but also *what was known about June 9th, 2011 on May 16th, 2017?*

2.3 Desiderata of a Temporal GIS

Given these challenges and research agendas, we can begin to construct a series of desiderata for any potential temporal GIS (TGIS) that would be of use to existing spatial research projects. Based on the issues outlined above and building on work by Richardson and Goodchild [11], [102] we have identified 4 major requirements for any proposed work within this area.

2.3.1 *Enable spatio-temporal object modeling*

While traditional GIS applications have provided robust support for managing data in the form of records, or observations, the added temporal dimension presents an added interaction challenge in that users often desire to interact with a single identified object at a given temporal state. An example would be the object *King County, Washington*, while it is true that King County has undergone a significant amount of spatial change since its original formation, a user may simply be interested in interacting not with the individual boundaries, but with the object itself, asking questions such as *what is the difference in total area of King County in 1997 vs 2003?* Or at a more complex level, *what was the population density of King County in 2000?* This last question requires gathering at least two different properties of the given spatial entity and performing some computation with them. While this is possible in a traditional GIS application there is a mismatch between the user interaction (on the level of asking questions about the given state of an object) and the data storage model which is largely focused on data records. This problem is magnified as the complexity of the queries increases and as more and more spatial objects are included. As any potential TGIS would treat time as a first-class citizen, the ability to gather individual records into unified temporal objects is of great importance.

2.3.2 *Support spatial and temporal relationships between spatio-temporal objects*

After constructing spatial objects, it is often desirable to express relationships between these various entities. These relationships can take a number of different forms. Spatial, such as *contained in*, *touches, overlaps with*, etc. Temporal, *begins, comes after*, *occurs during*. Or semantic, such as *member of*, *related to*, etc. While spatial and temporal relationships are simple to express from a computational perspective, the power of object modeling comes from being able to express higher-level associations and connections between objects. An example might be a map of hospital catchment areas, which may include multiple administrative districts, or portions thereof. Aggregating these individual objects into larger set relationships is difficult to manage in traditional applications but a potential TGIS should support modeling these types of relationships between spatial objects.

2.3.3 *Annotate events that occur in an object's lifetime*

In addition to relationships, objects can also experience *events* during and around their lifetimes. These events may be as simple as *created* or *destroyed*, which mark the beginning or end points of an object, or they might be more complex and denote domain specific knowledge attached to the individual objects. As an example, consider an object which contains a series of GPS points marking the track of a survey team performing door-to-door questionnaires. Not every home will be able to participate in the survey process and so the teams might annotate various GPS points with *survey_administered* events. These events can then be used to not only rapidly identify which homes were given surveys, but also by other research teams which might use these events to determine the optimal time for administering surveys in the future, in order to maximize participation.

2.3.4 *Support multiple time states*

While most temporal queries are concerned with time in the real world, some are also interested in viewing the world from the perspective of the database at a given point in time. A TGIS should support reasoning over both real-world time, and database time. Modern applications need to account for the fact that information gathering is a continually evolving process. Improvements in measurement, error correction, and access mean that data gathered five years ago, or even last week, exists today in a different information context. This has critical implications for issues such as a scientific reproducibility and retrospective analysis.

2.4 CONCLUSION

In this chapter, we have looked at some common challenges limiting the use of robust spatio-temporal analysis within the field of public health. We have outlined some mechanisms for dealing with these challenges, and the various ways in which traditional GIS applications are either insufficient or struggle to adequately support these new techniques. In the final section, we outlined some desiderata for a TGIS which both can address these existing challenges, as well as support new models of research. In the next chapter, we will look at prior theoretical models for managing spatio-temporal data, as well as evaluate some existing approaches which provide the building blocks for the work in this dissertation.

Chapter 3. PRIOR DATA MODELING APPROACHES

The previous chapters have outlined some existing challenges in spatio-temporal analysis as well as a set of desiderata for a future temporal GIS design that aims to address these issues. This chapter will shift directions and attempt to describe some previous data models and implementations that have been developed. In addition to providing historical context, these prior efforts have laid the foundation for the approach taken in this dissertation, which will be outlined further in the following chapters.

In this chapter, Section 3.1 will describe some basic concepts and terminology that will be used for the remainder of dissertation. Section 3.2 will outline four major approaches to modeling spatio-temporal data, Time slicing, ST-Composite, ST-Object and 3-Domain, and evaluate each of them against a set of criteria derived from the desiderata in Section 2.3. Section 3.3 look at alternative database systems that extend the traditional relational model, which will be used further in the Trestle system. Finally, Section 3.4 concludes with a brief description of software implementations that build upon the modelling approaches described here.

3.1 TERMINOLOGY

An important component of any interdisciplinary project is effective communication of ideas and concepts between the various stakeholder groups. This challenge becomes especially acute when dealing with concepts which have divergent meanings between disciplines. In order to be clear and consistent with the terminology used within this dissertation, this section will clarify some of these terms and detail their usage and meaning going forward.

3.1.1 *Objects and Records*

Within the field of Computer Science, an *object* refers to a value in memory which maintains some knowledge about its own internal state and responds to operations from other objects and functions [103]. Objects form the basis of *Object-Oriented Programming Languages* (such Java[15]) and nearly every piece of information or data value is modeled as some type of object. Thus, the use of the term *object* from a computer science perspective refers not only to named entities which are interacted with by users, but also internal program representations of various datum which may be invisible to the end user.

This is markedly different from the way the term *object* is used in other fields which may utilize a looser definition, such as the one proposed by May Yuan in 1996. Here, an *object* is a semantic concept which describes a collection of properties that hold true for a particular entity [104]. The key distinction being that objects refer to named entities (such as *King County,* or *Harborview Hospital)* which have some spatial component in the physical world.

[15] https://java.com/en/

Individuals interact with objects on a continual basis. The *coffee table* holding my personal *laptop* and *phone* are all identifiers that describe discreet objects that contain intrinsic properties about themselves; such as *my phone has a screen,* and temporal properties, such as *my laptop is currently in this coffee shop*. Individuals interact with, and reason about, the world around them at the object level. Questions such as *where is my phone?* or *is my laptop currently charging?* ask questions about the spatial and temporal states of an object at a given point in time. While this is an intuitive concept, the challenge comes in mapping between these objects and the values of their associated properties, for a given temporal instant.

Perhaps the most common method for representing large amounts of structured information is the *tabular* layout found in relational databases and structured text files, in which data is oriented as *rows* and *columns*. As an example, consider a selection of data for the *American Community Survey (ACS)* dataset (which will be detailed further in Chapter 6). Table 3 lays out various properties for *King County, Washington* in a traditional data layout. Each *row* represents a year in which data is available, and each *column* the value of that data property for the given year. In this dissertation, we refer to each *column* as a *data property* and each *row* as a *record*. Each *cell* (the value of a given *data property* for a specific *record)* is referred to as a *fact*. A fact is simply the value associated with a *data property* for a given time point (or range). In Table 3 the individual cells for the total *population estimate, male population estimate, and female population estimate* columns are the *facts* for the given county with the temporal range being derived from the *year* column. The *year* column itself is not modeled as a fact itself, it merely provides the temporal context to distinguish between the different value states of the data properties. *Objects* are collections of *facts* associated for a given entity (such as a county, cell phone subscriber, survey team, etc.) that hold true for a specific temporal period (e.g. the location of a cell phone subscriber last Wednesday, or the population count of a given county two years ago). These objects are also assigned unique identifiers (which will be discussed in 3.1.2) which are generally derived from a column in the tabular data. In this example, the *id* column provides the unique identifier for the individual objects, with only one row of value needing to be used as the identifier.

ID	**Geography**	**Total population estimate**	**Male population estimate**	**Female population estimate**	**Year**
0500000US53033	King County, WA	2117125	1057544	1059581	2015
0500000US53033	King County, WA	2079967	1039852	1040115	2014
0500000US53033	King County, WA	2044449	1020603	1023846	2013

Table 3: 2013-2015 ACS data for King County, Washington.

This table shows a selection of demographic indicators for three years of ACS data. The *ID* column corresponds to the ACS identifier for King County. The *Year* column represents the ACS year for which that row applies. The *Year* field is not found in the original dataset but is commonly generated by users when combining multiple years' worth of data into a single file.

This still leaves some ambiguity in distinguishing between *objects* as collections of *facts* (such as named spatial entities used within Geography), and *objects* as described separately in computer science in *Object-Oriented Programming.* Since our definition of *facts* inherently includes a temporal component, we can refer to these types of objects as *temporal-objects*. For this dissertation, we assume that all *temporal-objects* also contain a spatial component and will thus refer to these types of objects (objects representing a collection of *facts* for a given entity) as *spatio-*

temporal objects (ST-Objects). Any use of the term *object* (with the spatial or temporal modifier) will refer to the computer science definition and understanding of objects.

3.1.2 *Object Identity*

Given that we now have a consistent definition for what constitutes an ST-Object, the next question becomes how to construct these entities and consistently refer them as they change over time. While there are numerous methods for developing a system of referring to named entities [105], in the end, the appropriate solution is largely dependent on the domain use of that object. For example, the boundaries of an administrative district might change due to political restructuring, or they might change due to improvements in measurement technology or manual refinement of previously overlapping boundaries with other districts. The question then becomes, *if a district boundary is changed due to measurement errors, does it still describe the same object?* What if a district is divided into two districts, with one district retaining the original name (such as when King County was split into Kitsap County and King County in 1857)? Solving this issue is beyond the scope of a temporal GIS, in that it requires domain knowledge to be able to make decisions on how to delineate ST-Objects; instead, a TGIS should endeavor to provide the necessary tools for collecting *facts* into ST-Objects based on the criteria required by the domain application.

3.2 PREVIOUS MODELING APPROACHES

At the core of any data management system, is the underlying data model which describes the organization of information and its relationship to other members in the database. At its simplest level, a data model provides 3 major functions [106]:

1. A set of object types which define a set of basic building blocks.
2. A set of operations which provide a means for manipulating objects types in a database
3. A set of integrity rules which constrain the valid states of the database and ensure they conform to the data model.

This section will focus on discussing several prominent spatio-temporal modeling approaches that provide background efforts for the implementation developed for this dissertation. While there are numerous approaches to choose from (see [107], [108] for a more expansive discussion), the four listed here are unique due to both their technical innovation as well as their impact on the field as a whole:

Approach 1:	Time slicing
Approach 2:	Space-Time Composite
Approach 3:	Space-Time Object
Approach 4:	3-Domain

The importance of the core data model cannot be overstressed; a poorly designed modeling approach represents a view of the world which can severely hamper the ability of the user to effectively achieve their end goals [109]. An example of this, which will be revisited later in the chapter, is the ability for a given data model to describe the temporal bounds of an ST-Object. Meaning, that the data model can represent spatial entities which may have existed for a period of time in which no information about the entity is known to the database. Without the ability to represent the temporal boundaries of ST-Objects, the data model may not be able to correctly answer queries which rely on the *before/after* state of the given entity.

In order to effectively evaluate the utility of these data models in supporting spatio-temporal public health research, we have developed an evaluation approach, which will be applied to each of the four data models discussed in this dissertation. This approach closely follows the one utilized by May Yuan in her 1999 paper describing the 3-Domain model [110].

3.2.1 *Evaluation Approach and Criteria*

3.2.1.1 Evaluation Overview

The evaluation for each data model will consist of three major components.

1. An overview of the data model design and implementation.
2. A description of the model's fulfillment of the desiderata listed in Section 2.3, as well as the *design and performance requirements* described later in this section.
3. A summary of the ability of the data model to support the *evaluation queries* outlined in Section 3.2.1.3.

At the conclusion of this section (page 57), Table 9 will provide a summary of each data model's fulfillment of the various desiderata. In addition, Table 10 will summarize support for the evaluation queries.

3.2.1.2 Technical details for desiderata outlined in Section 2.3

While the desiderata have been outlined previously in Section 2.3, they will be restated here, with more technical details to follow.

D1. Enable spatio-temporal (ST-Object) modeling
D2. Support spatial and temporal relationships between ST-Objects
D3. Annotate object events
D4. Support multiple time states

D1: Enable Spatio-Temporal Object (ST-Object) modeling

While there are many types of ST-Objects, this dissertation focuses on *real-world* objects which are in common use in public health research and practice. That means objects which have occurred in and around the modern era (BCE time points are not supported, nor are dates beyond the year 3000), on or near the earth's surface and which have human reasonable temporal granularity. This excludes modeling objects such as protein structures or solar entities, as well as attempting to distinguish between events which have occurred with sub-nanosecond granularity. In addition, these models only support a linear view of time (e.g. no temporal branches). While there are valuable reasons to exceed all of these restrictions, they are beyond the scope of the type of problems being addressed here.

D2: Support spatial and temporal relationships between ST-Objects

The most trivial function of a GIS is to support querying for relationships between spatial entities. These relationships have been outlined in the literature by Egenhofer [111] and Clementini [112] and define binary relationship between two spatial *vector* entities (polygons, lines, etc.) such as *intersects*, *contains*, and *overlaps.* Temporal relationships are those defined by Allen [113] and represent the same types of binary relationships, only in the temporal domain (e.g. *during, before, begins,* etc). These basic relationship types generally have robust support within existing data management applications; but the technical challenge comes from combining these two types into unified spatio-temporal queries. Any proposed data model should not preclude supporting these simple relational types.

The important thing to note about these types of relationships is that they refer to directly computable knowledge about basic spatial and temporal types. While many spatial problems can be answered in this fashion, such as *Find all the health facilities contained within the given bounding box*, there are two primary restrictions that are excluded from this evaluation. The first, is answering queries such as, *find all the households near a given health facility,* which require more complex capabilities to answer, such as fuzzy logic systems [114], [115]. The second is any type of compensation for uncertainty due to issue such as measurement error or other types of spatial ambiguity [116]. Although these types of queries may be supported in the future, they are beyond the scope of this dissertation and will not be used in the evaluation criteria.

D3: Annotate object events

This desideratum was adequately described in the previous chapter.

D4: Support multiple time states

The simplest, and perhaps most common, concept of time is one that places some information at a specific temporal location. For example, *Population counts for US counties between 1990 and 1995.* This is known as *event time* or *valid time* and is the most basic form of temporal information. But one challenge with building integrated data repositories is that over time new information is added to the database and old information is either amended or removed. This means that asking the above query actually introduces a large amount of uncertainty in that it is possible for the database to return two different results if the query were executed in 1996 vs 2016. This additional piece of temporal context is known as *database time* or *transaction time*, which denotes when that piece of information was known to the database. There are many ways of supporting this type of information and more detail is given in [117], [118]. For the remainder of this dissertation, these types of systems will be referred to as *bi-temporal* databases, and conversely referring to both temporal types (valid and database) together will use the term *bi-temporal.*

Design and performance requirements

In addition to the desiderata listed above, I address two additional criteria that are less focused around the theoretical merits of the data model and more related to practical implementation concerns.

1. *Scale efficiently to support large volumes of data.*
   As spatial data continues to enter the realm of *Big Data*, it will encounter more and more of the *four Vs*: volume, velocity, variety, and veracity [119]. Any proposed data model should scale to support querying larger and larger spatial datasets, as well as enable performant updating. Likewise, spatial data takes a variety of forms not only in terms of vector or raster information, but also in terms of the additional pieces of information that are spatially and temporally associated with the *ST-Objects*.
2. *Integrate with existing database implementations and software development paradigms*
   While the limitations of existing spatial data systems have been detailed in previous chapters it should also be mentioned that the more a proposed data model can take advantage of existing tooling and technologies the greater its ability to leverage advances in these technologies to its own benefit. Likewise, using database systems and methods of interacting with data that are familiar to software developers can help accelerate both initial adoption as well as future expansion to support additional data modeling uses outside the original intent.

3.2.1.3 Evaluation queries

In order to evaluate how well each data model fulfills the given desiderata, we will refer to 10 example queries common to this type of research. Table 4 lists the queries in both a generic form (referring merely to abstract objects, time points, and properties) and a more specific form, using real world data and properties. Finally, the table also lists which desiderata the query fulfills. One thing to note is that some the generic queries are duplicated; for example, *Query 1* and *Query 7* have the same generic query type, but a different specific query. The reason for this is that each of the approaches in this chapter treat spatial information as a unique data type and thus *Query 7* serves to illustrate the ability of each model to represent spatial changes over time. On the other hand, *Query 1* demonstrates each model's ability to manage more generic data properties (such as population counts and region names). This distinction will be made clearer in the remainder of this chapter.

One final note, this list of example queries serves to illustrate the data modal function and suitability for the types of queries common in public health research. While the queries cover a large scope of potential spatio-temporal queries, its intention is not to be exhaustive, but sufficient for the purposes of this evaluation. Additional examples of spatio-temporal queries, suitable for a wide range of scientific domains, can be found in [104], [120].

	Generic Query	**Specific Query**	**Desiderata**
Q1	What are the values of $\{P_1, P_2, P_n...\}$ for O T_V?	Find the values of *all data properties* for *King County, WA* valid in *June, 1991.*	D1, D2
Q2	What was the change in the value of P for O between T_V1 and T_V2?	How did the *population count* of *King County, WA* change from *1990* to *2015*?	D1, D2
Q3	Which ST-Objects have the spatial relationship R_S with O at T_V?	Which *hospitals* where *contained* in *Juba, South Sudan* in *May 2015*?	D1, D2
Q4	Which ST-Objects have the temporal relationship R_T with O?	Which ST-Objects came temporally *before Cidade de Maputo, Mozambique*?	D1, D2
Q5	What was the value of P for O at T_V and T_Q?	What value did the database maintain for *population count* of *King County, WA* in *June 1991* at the query time *March 2016*?	D1, D2, D4
Q6	Which ST-Object has R_s with G at valid time T_V?	Which ST-Object contains the point *representing the University of Washington Medical Center* in *March 2017*?	D1, D2
Q7	What was the change in the value of P for ST-Object O between T_V1 and T_V2?	How did the *spatial boundary* of *King County, WA* change over *the past 10 years*?	D1
Q8	Is ST-Object O_1 with value V_1 of P_1 equivalent to ST-Object O_2 with value V_2 of P_2?	Is the ST-Object identified by *name Manhica, Mozambique* in *1990* the same as the spatial object, identified by *name Manhica*, in *2011*?	D1
Q9	For each period T_d in temporal interval T_I what was the value of P for ST-Object O_1 vs ST-Object O_2?	How many *mosquito bed nets* were *distributed each day during 2014* in *Cidade de Maputo, Mozambique* vs *Manhica, Mozambique*	D1, D2
Q10	How many events of type E occurred during temporal interval T_I for ST-Object O?	How many *survey_administered* events did *Contact Tracing Team 1* have in *March 1991*?	D1, D3

Table 4: Example queries for desiderata fulfillment.

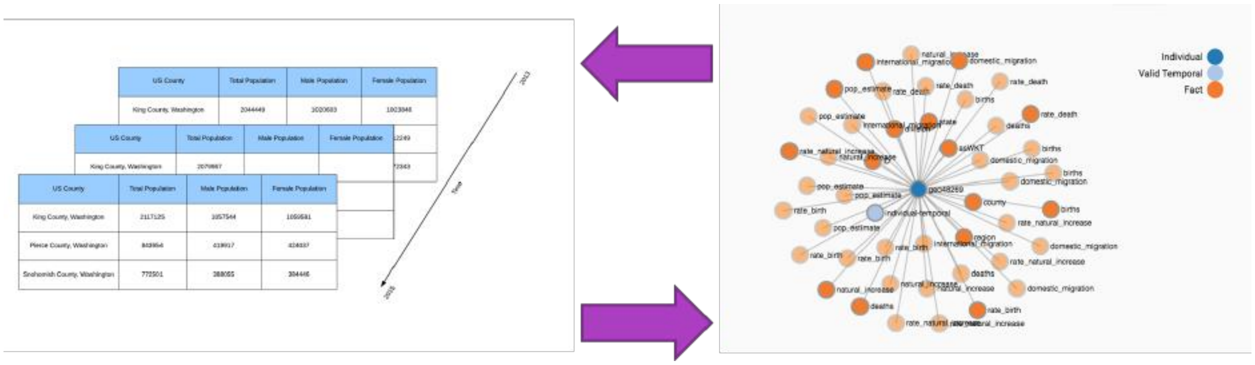
This table lists the 10 example queries used to determine how well a data modeling approach fulfills the given desiderata. Each query is listed in both a generic form, using symbolic representation, as well as in a more specific form which gives concrete examples of how each query might be used in a real-world public health use case. In addition, each query is linked to the appropriate desideratum evaluates. O represents an ST-Object, while P represents a data property associated with that object and V is the value of a given P. T_v represents *valid time* or the time that the property is true in the real world. T_q represents *database time*, or the time that the property was true within the database. T_i is a temporal interval while T_p refers to a temporal *period* within a given interval. R_s and R_t represent spatial and temporal relationships, respectively. G is a spatial entity, such as a point, polygon, or line, and E is an *event* entity.

Given the desiderata and example queries, the next question becomes, *from which potential viewpoint should we evaluate the system?* With that in mind, we will be evaluating these approaches from two distinct perspectives. The first is from that of the end user, the researcher attempting to utilize these data models to answer a domain specific research question. The second is that of a database administrator building and managing a repository of spatio-temporal data. While the first user perspective is understandable given the discussion of the previous chapters. The second perspective is perhaps more unusual, but no less important. One limitation to existing spatio-temporal research is that the required information is often scattered across a number of disparate data repositories and stored in a myriad of formats. Integrating this information into a representation usable for a given project is a tremendous cost paid by each researcher on each project. There is a tremendous amount of utility to be gained by developing a centralized data warehouse either for an individual research center or for a larger government entity [13]. The utility of a given data model lies not merely in its technical merits or theoretical simplicity, but in its ability to effectively enable the integration and management of large amounts of disparate information without requiring herculean efforts on the part of the data curators.

3.2.2 *Previous Approach 1: Time slicing*

Perhaps the simplest method for publishing data that may be recorded or updated on a periodic basis, is known as the *time slicing* approach. Here, a new copy of the data is published for each new release of information, which represents the current state of the dataset at that point in time. For instance, each year of the GAUL dataset is published as a separate Shapefile[16] and it is left to the user to determine how to integrate each new version with prior records. The same approach is taken by *ACS*, which releases updated information every year which simply supersedes any previously published data, an example of which is shown in Figure 5. Figure 6 illustrates laying out the data as individual (timestamped) files on the disk, versus loading into a relational database as timestamped tables.

[16] *ESRI Shapefile* is a common data format for storing and transmitting spatial information [192].



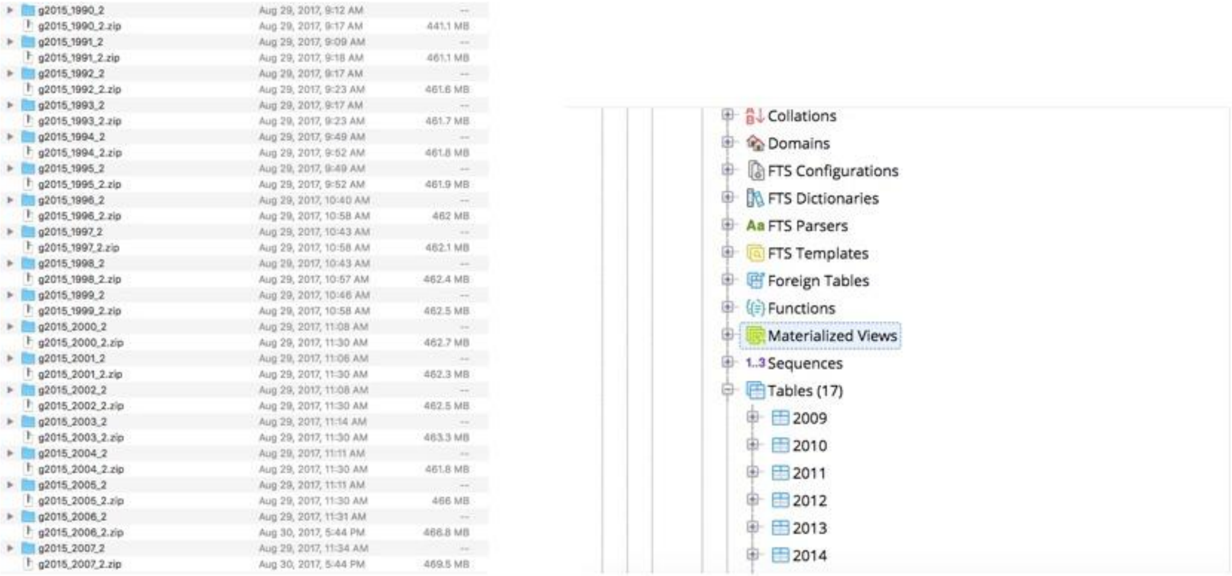
US County	Total Population	Male Population	Female Population
King County, Washington	2044449	1020683	1023846

US County	Total Population	Male Population	Female Population
King County, Washington	2079967		

US County	Total Population	Male Population	Female Population
King County, Washington	2117125	1057544	1059581
Pierce County, Washington	843954	419917	424037
Snohomish County, Washington	772501	388055	384446

Figure 5: ACS data for Washington State counties, from 2015 going back to 2013.

This shows a common *time-slicing* layout, where each new year of data is added as a separate table or file.



```
▸ g2015_1990_2        Aug 29, 2017, 9:12 AM      --
  g2015_1990_2.zip    Aug 29, 2017, 9:17 AM      441.1 MB
▸ g2015_1991_2        Aug 29, 2017, 9:09 AM      --
  g2015_1991_2.zip    Aug 29, 2017, 9:18 AM      461.1 MB
▸ g2015_1992_2        Aug 29, 2017, 9:17 AM      --
  g2015_1992_2.zip    Aug 29, 2017, 9:23 AM      461.6 MB
▸ g2015_1993_2        Aug 29, 2017, 9:17 AM      --
  g2015_1993_2.zip    Aug 29, 2017, 9:23 AM      461.7 MB
▸ g2015_1994_2        Aug 29, 2017, 9:49 AM      --
  g2015_1994_2.zip    Aug 29, 2017, 9:52 AM      461.8 MB
▸ g2015_1995_2        Aug 29, 2017, 9:49 AM      --
  g2015_1995_2.zip    Aug 29, 2017, 9:52 AM      461.9 MB
▸ g2015_1996_2        Aug 29, 2017, 10:40 AM     --
  g2015_1996_2.zip    Aug 29, 2017, 10:58 AM     462 MB
▸ g2015_1997_2        Aug 29, 2017, 10:43 AM     --
  g2015_1997_2.zip    Aug 29, 2017, 10:58 AM     462.1 MB
▸ g2015_1998_2        Aug 29, 2017, 10:43 AM     --
  g2015_1998_2.zip    Aug 29, 2017, 10:57 AM     462.4 MB
▸ g2015_1999_2        Aug 29, 2017, 10:46 AM     --
  g2015_1999_2.zip    Aug 29, 2017, 10:58 AM     462.5 MB
▸ g2015_2000_2        Aug 29, 2017, 11:08 AM     --
  g2015_2000_2.zip    Aug 29, 2017, 11:30 AM     462.7 MB
▸ g2015_2001_2        Aug 29, 2017, 11:06 AM     --
  g2015_2001_2.zip    Aug 29, 2017, 11:30 AM     462.3 MB
▸ g2015_2002_2        Aug 29, 2017, 11:08 AM     --
  g2015_2002_2.zip    Aug 29, 2017, 11:30 AM     462.5 MB
▸ g2015_2003_2        Aug 29, 2017, 11:14 AM     --
  g2015_2003_2.zip    Aug 29, 2017, 11:30 AM     463.3 MB
▸ g2015_2004_2        Aug 29, 2017, 11:11 AM     --
  g2015_2004_2.zip    Aug 29, 2017, 11:30 AM     461.8 MB
▸ g2015_2005_2        Aug 29, 2017, 11:11 AM     --
  g2015_2005_2.zip    Aug 29, 2017, 11:30 AM     466 MB
▸ g2015_2006_2        Aug 29, 2017, 11:31 AM     --
  g2015_2006_2.zip    Aug 30, 2017, 5:44 PM      466.8 MB
▸ g2015_2007_2        Aug 29, 2017, 11:34 AM     --
  g2015_2007_2.zip    Aug 30, 2017, 5:44 PM      469.5 MB
```

```
⊞ Collations
⊞ Domains
⊞ FTS Configurations
⊞ FTS Dictionaries
⊞ FTS Parsers
⊞ FTS Templates
⊞ Foreign Tables
⊞ Functions
⊞ Materialized Views
⊞ Sequences
⊟ Tables (17)
    ⊞ 2009
    ⊞ 2010
    ⊞ 2011
    ⊞ 2012
    ⊞ 2013
    ⊞ 2014
```

Figure 6: Filesystem versus database data layouts for GAUL data.

The image on the left shows the years of GAUL data laid out with separate folders for each year. Selecting the necessary data requires identifying the appropriate folder and manually loading required file. The image on the right shows the same data (only years 2009-2014) loaded into a relational database. Again, the data selection process requires identifying the appropriate table and retrieving the necessary information.

One benefit of this approach is that it is easily implemented in existing relational database systems and thus can take advantage of the tremendous research and development efforts that have gone into making these types of systems robust, scalable, and secure. Likewise, developing software against these implementations presents the developer with a technology that is well-known and reasonably understood, which allows them to focus on developing tools to solve the research problem at hand, and not on mastering the intricacies of a new research data model.

3.2.2.1 Fulfillment of Desiderata

D1: Enable Spatio-Temporal Object (ST-Object) modeling

Time slicing provides limited support for ST-Objects, in that it allows adding data properties which can be linked to a single entity by use of a consistent identifier. This enables the approach to answer *Query 1*; however, it is focused on modeling data properties and not objects, which means that each object has its properties scattered amongst various data tables and not aggregated into unified objects. This requires additional effort on the part of the user, or system implementer, to manually gather the required information and present it accordingly.

Likewise, separating the data into multiple years makes it difficult to reason about changes in ST-Object states over time. For example, answering *Query 2*, requires merging data from multiple files or database tables, which can create performance issues as query complexity grows. In addition, this approach provides no ability to identify missing data. If no data has been loaded for *2011*, the query will simply return and silently omit the missing data point, leaving it up to the user to manually verify that all the required information is present before utilizing the query results. Relatedly, this approach assumes that data is updated at the same temporal frequency. Managing data properties which might be updated more or less frequently (e.g. a population count might be updated once a year, but population migration estimates might be updated once per quarter) requires further splitting of the data into tables which represent a common temporal interval. This further compounds the query complexity as data scale increases.

D2: Support spatial and temporal relationships between ST-Objects

This approach makes it trivial to implement support for most types of spatial relationships, due to the robust support for these types of relationships within existing applications such a relational databases or GIS applications such as *ArcGIS*[17]. As each spatial property is available in each temporal layer, answering *Query 3* simply requires selecting the table with the properties for the given year, performing the appropriate spatial intersection and returning the identifiers for the matched entities.

While spatial relationships are trivial to support, calculating temporal relationships is more difficult. Consider *Query 4*, determining a temporal ordering of objects requires computing two pieces of information. 1.) The temporal interval for which each object exists 2.) the relationship between these temporal intervals. Since the time stamping model does not natively contain a concept of an object, determining existence requires computing the temporal range between the oldest and newest *fact* attributed to that individual. This presents a number of potential challenges.

[17] https://www.arcgis.com/index.html

One is that it is possible for an object to exist for a greater temporal duration than the facts associated with it. For example, Zadar, Croatia has a recorded history going back to the 9th century BCE but may only have census records going back to the 1800s [121]. Thus, determining the temporal duration for which Zadar exists may not be possible given the information available to the data model.

It should be mentioned that it is possible to partially overcome the issue of object existence by embedding that information into the data itself. This is the approach taken by the *GAUL* dataset, which contains fields which denote how long the administrative unit has been in existence. This solution though ends up being implementation dependent and not intrinsic in the modeling approach, thus presenting a situation where *Query 4* is easily answerable for some datasets and impossible to answer correctly in others.

D3: Annotate object events

Time slicing has no direct support for modeling events but could be implemented by creating an additional set of tables which either denote events at specific time points or directly reference fact rows in the tables. Again, this is an implementation specific approach and not inherent to the data model.

D4: Support multiple time states

Simple support for *validity* intervals is supported by this modeling approach, both because it can be made explicit in the data itself (e.g. each layer can contain *valid from/to* fields) and implicitly in that the data in each table is tied to the temporal range of that table. For example, data contained in the table titled *1990* can reasonably be assumed to have two meanings. First, that the data in that table is valid for the year *1990*. Second, that the data in that table is valid from the beginning of the year *1990*, until the next data snapshot. Disambiguating these two meanings is not supported by the snapshot data model and requires this information to be encoded either in the implementation, and enforced by tooling, or via cultural knowledge and enforced by error.

Returning to the example queries, satisfying *Query 1* simply requires selecting the appropriate snapshot, either the snapshot directly correlated to the temporal point, or the one closest to the temporal point (e.g. if querying for data points in 1993, the system may gather data from either the 1993 snapshot, or the 1992 snapshot, if no data exists for 1993); from there it is merely a matter of filtering the results for the appropriate object identifier and required data properties.

Answering *Query 2* is a more difficult proposition in that not only must the application perform a join of all the required data snapshots and filter for the desired properties, but it must also have some additional ability to determine if all the required data is present in order to fulfill the query. This again is a problem to be resolved by the implementation.

Regarding *database* temporal intervals, the snapshot model does not explicitly support this additional dimension but can be extended by either duplicating the snapshots or the individual data rows, as new information becomes available.

Design and performance requirements

A significant benefit of this approach is that it makes no distinction between spatial and non-spatial data properties. Each data property is stored as a column in the required table and the underlying storage engine is relied upon to manage the various data types appropriately. This means that users and administrators can manage both spatial and non-spatial data properties via the same sets of tooling. Likewise, building on well-known data storage engines (such as relational databases) allows data repositories to take advantage of the tremendous amount of engineering effort and technical improvements that have gone into make these systems scale to vast amounts of information.

But while the implementation simplicity is appealing, there are some additional challenges that should be mentioned further. The first is related to potential duplication of information. Creating a new copy of the data for each new temporal interval can result in significant duplication of information if those properties are not changed at each update interval. As an example, consider Figure 7; here we have a selection of demographic indicators for King County, Texas and King County, Washington from 2011 to 2016. For the Washington county, we can see that the demographic indicators changed for each year, but for the Texas county, the number of deaths were the same from 2014 to 2016, while the *domestic migration* rate was stable from 2013 to 2016. As the time slicing approach simply appends additional data as another table or layer, this can result in significant duplication of data, even if the majority of the indicators remain static. While this may not be an issue for smaller datasets, as both the scale of data and the complexity of the queries increases, so does the storage and computational requirements.

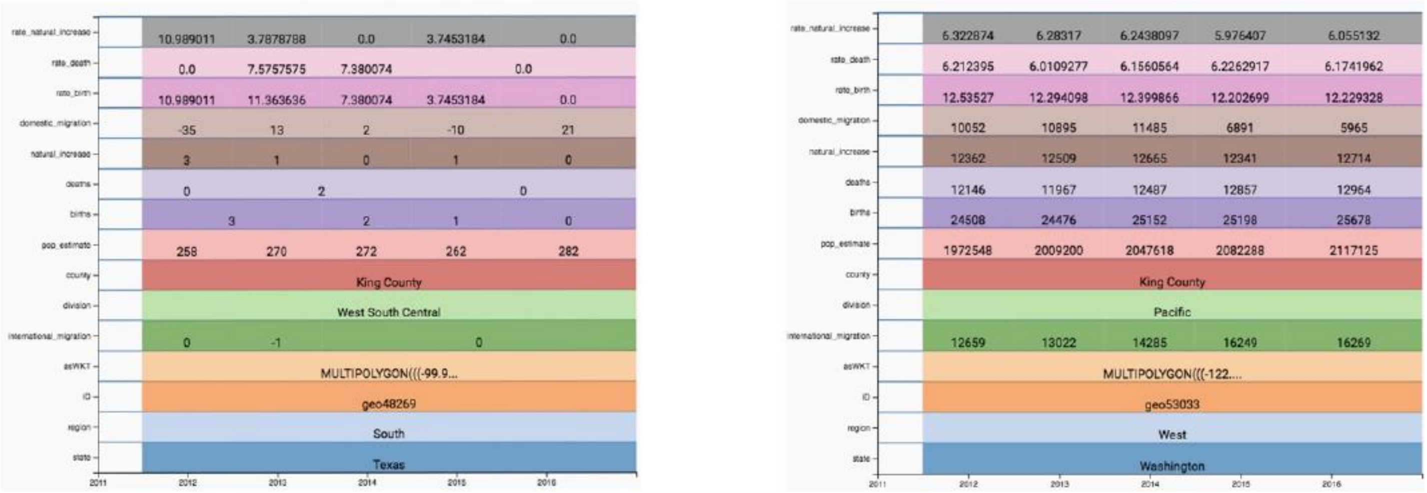


Figure 7: Comparison of ACS data properties for the years 2011-2016 for King County, Texas and King County, Washington.

This figure illustrates the amount of data property change for two Us counties over the span of 5 years. For the Texas county (on the left) the value of many properties, such as *deaths* and *births* is the same for multiple years; whereas for the Washington county (on the right) the values of nearly every property change each year. For both counties, the *county* and *division* properties undergo no change over this temporal span; however, the time-slicing approach requires duplicating data for each year, regardless of whether or not the property value is the same for preceding years.

Fulfillment of example queries

Query	Supported	Notes
Q1	✓	
Q2	✓	
Q3	✓	
Q4	✗	Implementation specific solution required
Q5	✗	Implementation specific solution required
Q6	✓	
Q7	✓	
Q8	✗	Limited object support impedes query fulfillment
Q9	✓	
Q10	✗	

Table 5: Time slicing example query fulfillment.

The time-slicing approach is able to fulfill 6 of the 10 evaluation queries; though some of the unfulfilled queries could be supported through implementation specific solutions.

3.2.3 *Previous Approach 2: ST-Composite*

First proposed by Langran and Chrisman in 1988 [122], the *ST-Composite* method improves upon the *time slicing* approach by attempting to reduce both the amount of duplicate data being stored, as well as improve the performance of spatial queries. This model beings with the most common spatial visualization method, the map, and as new information is added, at each time point, changed information is merged into the map by adding an additional layer. As an example, consider Figure

8. Here we have two views of *Cidade de Maputo, Mozambique*. One in 1990 and the other in 2014. It is easy to see that the inner counties, which make up the city (Aeroporto, Distrito Municipal 1, etc.), are merged into a new county. But around the city, nothing has changed. In the *ST-Composite* model, only the changed information is added to the new map, so all of the surrounding counties, which underwent no spatial reorganization are left as is. This reduces the amount of storage space required to maintain the spatial history of a specific location and presents an intuitive interaction model in which users are presented with a map of the state of the world at a given point in time.

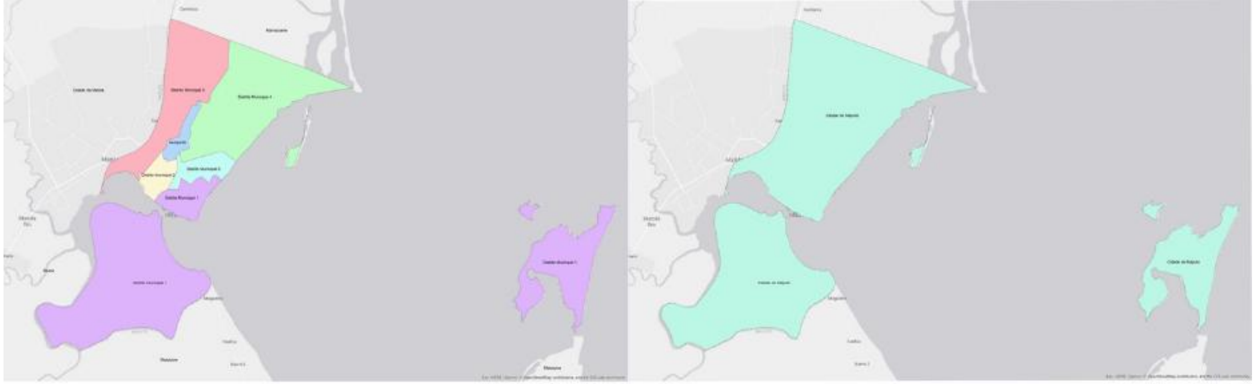


Figure 8: Cidade de Maputo, Mozambique in 1990 vs 2013.

Showing two temporal states of the capital city of Maputo, which are of interest to public health researchers and need to be compared with ease and stored efficiently.

This approach has been expanded by other researchers to incorporate techniques used in *version control* schemes employed by tools such as *git*[18] [123], which manage changes in documents by only storing the differences between file versions, rather than the entirety of the file itself. This reduces the amount of duplicate data being stored and provides a simple method for reasoning about changes in the dataset, in that each change clearly records what information was modified at each time point. Reconstructing the state of a file at a specific time point involves starting with the initial state of the file and applying each file difference up to the time period requested. This approach has been shown to scale well, even to very large datasets and the idea of replaying transformations on data, instead of simply updating the data at each time point, is the foundation for the distributed nature of computation frameworks such as Apache Spark [124].

3.2.3.1 Fulfillment of Desiderata

D1: Enable Spatio-Temporal Object (ST-Object) modeling

The *ST-Composite* model does not support object modeling. At its core, this approach is about managing changes in spatial polygons. Each changed polygon, representing a new spatial boundary for a given object, is merged into the current map state. This means that the actual boundary is discarded and what is committed to the repository is the spatial difference between the new polygon and the previous map state, thus removing any direct link between the versioned object states. Overcoming this limitation requires implementation specific solutions which link various polygon fragments back to their corresponding ST-Objects. As an example, this model

[18] https://git-scm.com

cannot answer *Query 6* because it only retains the ability to compare spatial layouts between time periods. This means that it has no ability to associate spatial boundaries with a *Manhica* ST-Object and determine which of those boundaries are valid for a given timepoint.

In addition, as this data model only has a concept of a static map, it can only present a single data property at any given point in time and any other data types are required to be managed by a different modeling paradigm and must be manually mapped back to the *ST-Composite* data store. At its simplest level, this model cannot answer *Query 1* and answering *Query 2* means it would be unable to answer the same query, but for a different data property, without constructing an entirely different map.

Another challenge with this approach, is that it has limited ability to effectively manage different types of spatial data. Consider the example detailed in Section 2.1, tracing movement patterns of disaster affected individuals, how would this model effectively manage large amounts of point data? In effect, it would devolve into the *time slicing* approach, in which each new point is merged with the previous point data, albeit with no duplication of existing points. Likewise, this model does not support representing overlapping spatial entities; if two objects claim to cover the same spatial area the *ST-Composite* model has no way to effectively represent this spatial collision.

D2: Support spatial and temporal relationships between ST-Objects

The *ST-Composite* model does provide robust support for spatial relationships, answering *Query 3* (provided the implementation supports some object linking ability) simply requires selecting the appropriate map state and performing the necessary spatial calculation. From there, the resulting spatial fragments can be linked back to their parent objects and the results returned. *Query 6* can be answered through similar means as well.

Temporal relations are more difficult for the *ST-Composite* model to directly express. Tracking historical changes to object boundaries, such as when answering *Query 7,* can be done by reconstructing the original object boundaries from the current fragments, but more complex temporal relationships, such as *Query 4*, are not supported by this model because it has no ability to track *ST-Objects* over time.

D3: Annotate object events

This model provides no support for object events.

D4: Support multiple time states

As previously mentioned, this modeling approach has a limited notion of space and time, constrained to what is representable on a static map. As such, it has no ability to represent *database time*, except through manual methods such as creating multiple versions of maps, which represent specific *database time* periods.

Design and performance requirements

One significant benefit of implementing this approach, is that it leverages the most basic spatial data representation (the map) and presents it to the user as a view into the world at a specific point in time. This allows intuitive reasoning about temporal changes such as boundary reorganization. Likewise, the implementation is fairly straightforward in that tooling exists for easily generating high-quality maps, provided the data can be simplified into a single temporal point.

That being said, one challenge with this approach, is that it splits spatial and non-spatial data into their own management paradigms, while the spatial boundaries are stored as intersection layers on the base map, the non-spatial data is retained through traditional storage methodologies. This requires each query to the data store to perform two operations. First, identify the temporal overlay of the ST-Object being queried, then look up the correct set of data properties valid at the given time point. For an ST-Object which undergoes a tremendous amount of spatial variation, there may only be a single set of data properties for a given spatial state, but for objects which remain fairly static over time (such as the national boundaries of the US) there may be a large set of data properties that need to be filtered. This asymmetry presents a challenge not only for database designers but end users as well.

Fulfillment of example queries

Query	Supported	Notes
Q1	✗	
Q2	✓	Only for a single data property
Q3	✓	
Q4	✗	
Q5	✗	
Q6	✗	
Q7	✗	
Q8	✗	
Q9	✓	Only for a single data property
Q10	✗	

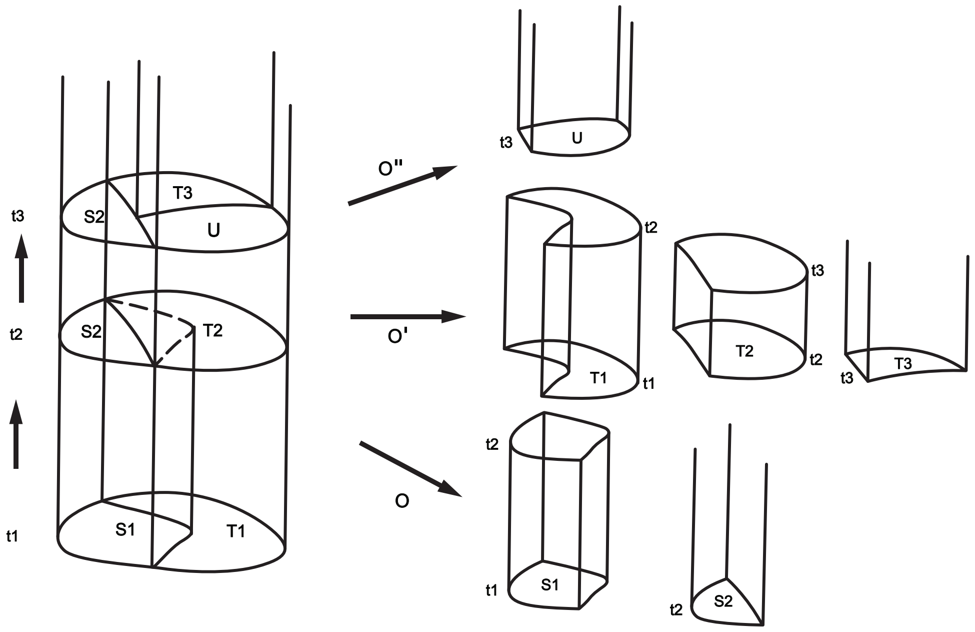
Table 6: ST-Composite example query fulfillment.

The ST-Composite model is only able to fulfil 3 of the 10 evaluation queries. For Queries 9 and 2 it is only able to fulfill them for a single data property at a time, supporting multiple data properties requires redrawing the map with the new data.

3.2.4 *Previous Approach 3: Spatio-temporal Object Model*

Thus far, the approaches we have listed have featured one prominent limitation, they have no native concept of semantic objects. Aiming to address this, Michael Worboys developed and proposed the *ST-Object object model* in 1992 and expanded it a few years later [14], [125]. At the core of this approach is the *ST-Atom*, which represents the spatial value of a given entity for a specific temporal interval. These ST-Atoms are then collected into larger ST-Objects which contain an ST-Atom for each change in the spatial value.

Given this base representation, Worboys also proposed a method for representing the spatial and temporal relationships between the various objects. Based on a simple 3D graph, each *ST-Atom* is represented as an *ST-Simplex*, which creates a right prism in the 2D plane, in which the spatial component of the atom forms the base, and the prism is then extended in the Z-axis for the entirety of the temporal period. This layout is illustrated in Figure 9. From there, *ST-Simplex* objects are collected in *ST-Complex* objects, which represent the spatial and temporal relationships between multiple *ST-Atoms*.



ST-objects modelling regional change

Decomposition of ST-objects into ST-atoms

Figure 9: ST-Object data model components.

This figure illustrates the relationships between the various data model components as developed by Worboys [125]. On the left is the *ST-Complex* which contains multiple *ST-Objects* (S, T and U). Each ST-Object is composed of multiple *ST-Atoms*. Time is shown on the Z-axis and each time an ST-Object undergoes spatial change (t1, t2 and t3), all ST-Objects are updated to account for the new change and a new ST-Atom is created for each object (which is illustrated on the right side of the image).

The benefit of this approach is that both the spatial and temporal properties of objects are represented simultaneously on a unified coordinate plane, this means that answering *Query 6* can be done in a single pass through the data, since all three coordinates (latitude, longitude, and time) are represented in a single 3D model.

3.2.4.1 Fulfillment of Desiderata

D1: Enable Spatio-Temporal Object (ST-Object) modeling

This approach is explicitly designed to support complex spatio-temporal queries over *ST-Object* states and features robust support for answering these types of queries. In this model, both spatial and temporal information are represented in the same coordinate plane and can be computed using a single intersection operation. Thus, this model can easily fulfill *Queries 3, 4, 6, and 7*.

D2: Support spatial and temporal relationships between ST-Objects

The *ST-Object* model fully supports both spatial and temporal relationships, using the same query types. This is in contrast to other approaches, such as *time slicing*, which requires two query steps in order to identify all the spatial properties valid at a specific temporal instant and finally to perform the spatial intersection on the filtered properties to fulfill the actual query. Since *ST-Simplexes* have both their spatial and temporal values represented on the same 3D coordinate plane, answering *Query 6* only requires a single range query which selects the *ST-Simplex* which contains the 3D point created by projecting Point A into the temporal space at time T. This process becomes more complicated when considering *bi-temporal* elements, which will be discussed below.

D3: Annotate object events

This model has no direct support for object events.

D4: Support multiple time states

While the original model did not feature *bi-temporal* support, Worboys later proposed an extension which included explicit support for multiple temporal types [14]. This is implemented by creating multiple sets of *ST-Complexes,* which can then unioned together at the point of temporal intersection in order to determine the valid *ST-Simplexes* to perform the actual query with.

Design and performance requirements

One significant limitation of this model is that it has a reduced ability to represent both non-spatial data properties and data properties which are updated without a corresponding change in the spatial boundaries. As an example, this data model may not be able answer *Query 2*, without an additional data storage layer. This is especially true if the spatial boundary for *King County* did not change during those 15 years, which would not trigger a new *ST-Simplex* to be added to the model.

An additional challenge with this approach is that is that it requires a fairly unique data storage method. Rather than relying on traditional storage and indexing methods, such as R-Trees [126], or QuadTrees [127]; this approach re-projects the spatial and temporal information into a single coordinate space, which may require modifying the data at load time. This results in a data layout which is not well optimized in most storage engines and may require significant effort on the part of the implementer; however, this limitation can be significantly reduced by improvements in data storage engines to support multi-dimensional indexing and query methods, such as TV-Trees [128] or space-filling curves [129]. Another point of concern, as new spatial information is added, older *ST-Simplexes* need to be modified to support the new information. This is due to a restriction in the data model in which *ST-Simplexes* cannot overlap and thus old representations need to be modified to consider the new information, this can have a dramatic effect on the ability of the system to quickly load new information as it becomes available.

Fulfillment of example queries

Query	Supported	Notes

Q1	✓	
Q2	✓	
Q3	✓	
Q4	✓	
Q5	✓	
Q6	✓	
Q7	✓	
Q8	✓	
Q9	✓	
Q10	✗	

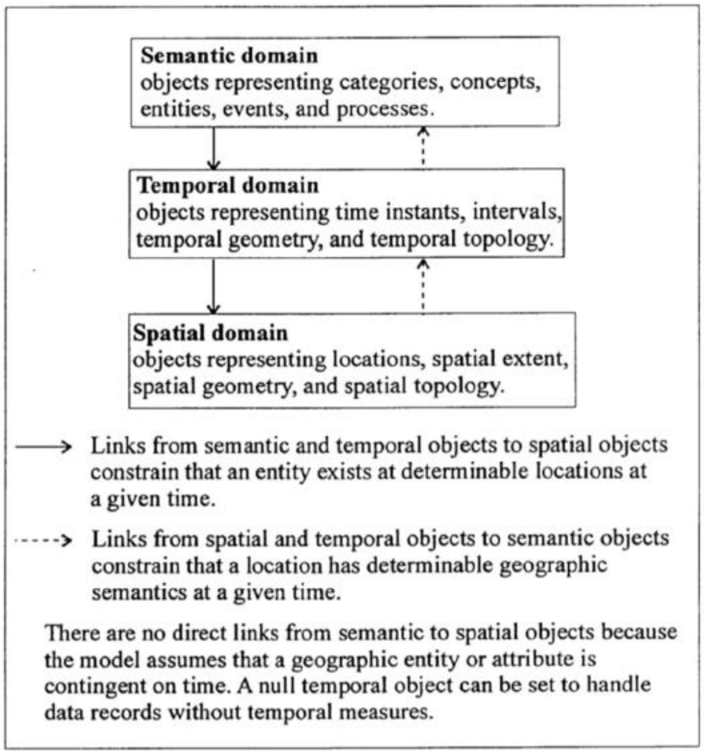
Table 7: ST-Object example query fulfillment.

The ST-Object model is able to fulfill 9 of the 10 evaluation queries. The only unsupported query is due to missing *event* support.

3.2.5 *Previous Approach 4: 3-domain Model*

One limitation that has been identified with the previous approaches is the bifurcation of spatial and non-spatial information. Both the *ST-Composite* and *ST-Object* models focus on managing and reasoning about changes in spatial information, leaving non-spatial information to be dealt with through external data management solutions. In 1996, May Yuan proposed a new solution to this problem that focused on the fact that fundamentally, there are three types of data related to spatio-temporal research. *Spatial, temporal,* and *semantical* [104]. While the first two datatypes have been described earlier, the *semantical* data type refers to thematic properties and classifications which are associated to a given ST-Object. These semantic types generally refer to data properties and are necessary for effectively answering a broad range of spatio-temporal queries. Failing to fully account for them in a given data system significantly limits both its utility, as well as its potential adoption. Thus, she proposed the *3-domain* model which aims to not only address limitations in prior approaches (specifically the three mentioned previously in this chapter), but also to integrate the three fundamental data types into a unified model.

A significant innovation of this approach is the realization that information can be modeled in such a way that there can be a one-to-one mapping from semantical and temporal objects to spatial objects, and from spatial and temporal objects to semantical objects, as illustrated in Figure 10. This means that an appropriate data model can effectively represent all three types of data and efficiently link between them as the query requires. This is accomplished by modeling each data domain separately and creating lookup mechanisms for linking in either direction.



Semantic domain
objects representing categories, concepts, entities, events, and processes.

Temporal domain
objects representing time instants, intervals, temporal geometry, and temporal topology.

Spatial domain
objects representing locations, spatial extent, spatial geometry, and spatial topology.

——> Links from semantic and temporal objects to spatial objects constrain that an entity exists at determinable locations at a given time.

- - - -> Links from spatial and temporal objects to semantic objects constrain that a location has determinable geographic semantics at a given time.

There are no direct links from semantic to spatial objects because the model assumes that a geographic entity or attribute is contingent on time. A null temporal object can be set to handle data records without temporal measures.

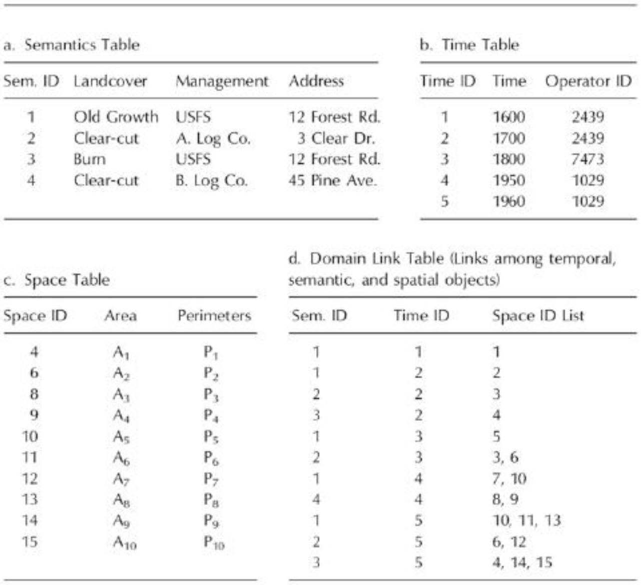
Figure 10: Linking between spatial, temporal, and semantic information (Reproduced from [110]).

Yuan points out that this model can be implemented either via a traditional relational database layout, or through more complex object modeling approach. Here, we will describe the relational model, as it is conceptually simpler and easier to implement in existing data storage systems.

In the relational approach, this model defines 3 primary tables (illustrated in Figure 11):

- *Semantics Table*: Which maintains the stable object identifiers and any additional semantic information that is associated with the ST-Object (e.g. population estimate, hospital bed count, etc.).
- *Time Table*: Maintains a list of temporal events which are known to the database. Each change to a piece of information (e.g. adding a new object, changing a boundary, updating a data property) results in a new row added to the table denoting when the event occurred, in real-world time.
- *Domain-Link Table*: This final table links object identifiers and temporal events to a list of spatial identifiers which are valid for that object at the given temporal instant.

In addition to these tables, the *3-domain* model also maintains a *spatial graph*, which tracks changes in spatial values over time. This is required because one of its supported performance enhancements is that it only stores the most recent set of spatial information, rather than the entire history of past values. We refer to these pieces of information as *spatial fragments*. The spatial graph is a powerful optimization and is constructed in such a way that the spatial value of an object at any given point in time can be re-computed using both the graph and the *Domain-Link Table*, which determines the fragments associated with the ST-Object at the given time point (see Figure 12). This is similar to the approach taken by the *ST-Composite* model but extended to include an additional table which tracks the spatial fragments contributing to the ST-Object at any point in time.



a. Semantics Table

Sem. ID	Landcover	Management	Address
1	Old Growth	USFS	12 Forest Rd.
2	Clear-cut	A. Log Co.	3 Clear Dr.
3	Burn	USFS	12 Forest Rd.
4	Clear-cut	B. Log Co.	45 Pine Ave.

b. Time Table

Time ID	Time	Operator ID
1	1600	2439
2	1700	2439
3	1800	7473
4	1950	1029
5	1960	1029

c. Space Table

Space ID	Area	Perimeters
4	A_1	P_1
6	A_2	P_2
8	A_3	P_3
9	A_4	P_4
10	A_5	P_5
11	A_6	P_6
12	A_7	P_7
13	A_8	P_8
14	A_9	P_9
15	A_{10}	P_{10}

d. Domain Link Table (Links among temporal, semantic, and spatial objects)

Sem. ID	Time ID	Space ID List
1	1	1
1	2	2
2	2	3
3	2	4
1	3	5
2	3	3, 6
1	4	7, 10
4	4	8, 9
1	5	10, 11, 13
2	5	6, 12
3	5	4, 14, 15

Figure 11: 3-domain table layout (Reproduced from [110]).

The shows the three primary tables required for linking between all three types of data. The *space table* is the table representation of the *spatial graph* described above.

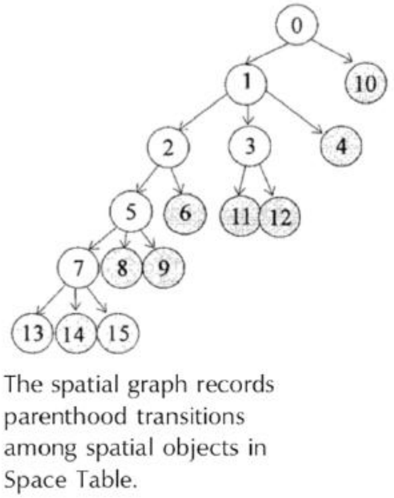

0
10
1
2
3
4
5
6
11
12
7
8
9
13
14
15
The spatial graph records parenthood transitions among spatial objects in Space Table.

Figure 12: Graph layout of the *space table* shown in Figure 10 (Reproduced from [110]).

To illustrate the data model, consider *Query 7*; in order to calculate an answer, the model selects all the time elements for *Object A* that are within the query window (e.g. 10 years from the current date). Next, for each time element it looks up the list of spatial fragments valid for that object at the given time element. For each fragment is missing from the *spatial graph*, it traverses the graph to select all the child fragments that are currently valid, and then aggregates them into two objects, representing the initial spatial state and the final spatial state, which can then be compared.

3.2.5.1 Fulfillment of Desiderata

D1: Enable Spatio-Temporal Object (ST-Object) modeling

This data model provides robust support for maintaining consistent identifiers for ST-Objects as they undergo state changes over time; however, one limitation is that the ST-Objects, by default, derive their existence from the presence (or absence) of data properties. This means that this model is limited in its ability to answer *Query 4*, because it would only be able to determine the existence duration by what information is has in the database. Adding this information would require additional information to be added to the *Semantics Table*.

D2: Support spatial and temporal relationships between ST-Objects

This model provides full support for reasoning about spatial relationships between objects. In addition to simple binary relationships, this model provides the ability to answer *Query 6,* since it maintains a list of which spatial entities contribute to an *ST-Object* over time. One limitation of this approach is that since only the latest spatial states are stored in the model, performing historical relation queries requires traversing the spatial graph and re-computing the original spatial state, which may be challenging as the number of states increases.

Likewise, for temporal relationships, aside from the afore mentioned limitation of being unable to fully express object existence, this model provides robust support for temporal relationships that can be derived from the temporal range of the object facts.

D3: Annotate object events

By default, this model provides no support for object events, but could be adapted to do so by two approaches. First, events could be modeled as additional pieces of semantic data and stored in the appropriate table. Second, additional information could be added to the *time table* to mark temporal events as having some additional identifiers which can be queried separately. While both of these approaches are feasible, they require implementation specific solutions as the data model provides no native concept of object events.

D4: Support multiple time states

This model provides no native concept of multiple temporal types but could be added by simply extending the *Time Table* to include additional *database* temporal columns. This, of course, would be left to the implementers to ensure.

Design and performance requirements

One benefit of this approach is that it is directly amenable to implementation within existing data modeling paradigms, such as relational databases. Each domain type (spatial, temporal, semantic) can be represented as tables in a relational model, with key lookups supported between the tables.

One limitation is that all semantic information (which includes both object identifiers as well as data properties) is constrained to the *Semantics Table*. This means that each new object type added to the database requires a new *Semantics Table* to be added, in order to include the properties for the new object. This significantly improves the complexity of answering spatio-temporal queries that require information from multiple types of ST-Objects. For example. *Query 1*, might require referencing information from multiple datasets, which could present an issue as a data sizes grow.

Fulfillment of example queries

Query	Supported	Notes
Q1	✓	
Q2	✓	
Q3	✓	
Q4	✗	Implementation specific solution required
Q5	✗	Implementation specific solution required
Q6	✓	
Q7	✓	
Q8	✓	
Q9	✓	
Q10	✗	

Table 8: 3-Domain example query fulfillment.

The 3-domain approach is able to fulfill 7 of the 10 evaluation queries. Queries 4 and 5 both require implementation specific solutions, due to the model's limited support for object existence and *bi-temporal* queries.

3.2.6 *Evaluation Summary*

In concluding the data model evaluation, we can see that the four models described thus far present unique and innovative methods for managing spatio-temporal data. Taken as a whole, several major themes have emerged. The first, is that spatial relationships are essentially *par for the course*, these models, especially when implemented within traditional data backends, provide robust support for different types of spatial relationships between object states, with little to no additional effort required. The second theme, is that the models provide varying degrees of support for temporal relationships, but largely struggle when dealing with the temporal boundaries of object existence, especially object existence which extends beyond the amount of time for which an object has record facts.

The third theme is only the *ST-Object* model directly supports bi-temporal implementations, though it may be possible for other approaches to be modified to support these types of data properties, but only for specific implementations. Fourth, it remains an open question as to how these models scale to the types of datasets common in public health data. Namely, multiple datasets, linked together, to form a final analytical model (as described in Section 2.2.1.2). This challenge is especially acute for ST-Objects such as US counties, which have a large number of data properties, spread across multiple datasets. This is further compounded by the fact that the spatial area of these regions may have little to no spatial change, but which have other data properties which change every year. This magnifies the requirement for being able to manage the internal states of ST-Objects.

The final theme, is that none of the above models provide a native concept of events, but some may allow them to be implemented as an additional data property type. Going forward, there remains space for a new data model that leverages the advances of the previous models, such as support for all fundamental types of spatio-temporal information, maintaining a consistent object identifier, and presenting an intuitive interaction model to the end user; but improves upon the state of the art to support events, large-scale interlinked datasets, and true object existence.

Approach	Object Modeling	Object Relationships	Events	Time States	Development
Time slicing	Not directly, only if implemented by the database administrator	Full support for spatial relationships, limited support for temporal relationships	No event support	Supports bi-temporal time	Simple to implement, inefficient with large datasets
ST-Composite	No support for ST-Objects	Limited support for spatial relationships, no support for temporal relationship	No event support	No direct support for database time	Requires custom data backend and relies on unique indexing techniques
ST-Object	Full support for ST-Objects	Full support for both spatial and temporal relationships	No event support	Supports bi-temporal time	Requires custom data backend and relies on unique indexing techniques
3-domain	Full support for ST-Objects	Full support for both spatial and temporal relationships	Some support for events	No direct support for database time	Builds on existing relational database technologies, along with custom spatial support

Table 9: Summary of data modal desiderata fulfillment.

This table compares the various modeling approaches in their ability to satisify both the evaluation desiderata and the design and performance requirements. The main themes are limited to no support for *bi-temporal* queries, *events* or object existence.

Query	Time slicing	ST-Composite	ST-Object	3-Domain	Total
Q1	✓	✗	✓	✓	3
Q2	✓	✓	✓	✓	4
Q3	✓	✓	✓	✓	4
Q4	✗	✗	✓	✗	1
Q5	✗	✗	✓	✗	1
Q6	✓	✗	✓	✓	3
Q7	✓	✗	✓	✓	3
Q8	✗	✗	✓	✓	3
Q9	✓	✓	✓	✓	4
Q10	✗	✗	✗	✗	0
Total	6	3	9	7	

Table 10: Summary of data model example query fulfillment.

The final counting of the evaluation query fulfillment for the various approaches. The columns and rows are aggregated to give a view both the number of queries fulfilled by a given approach, as well as how well supported a given query is.

3.3 Alternative Data Storage Approaches

For the most part, all of the previously described approaches have been designed with an eye towards the traditional relational data storage model; in part, due to its ubiquity, but also because the relational table model provides an intuitive means for interacting with basic data types. But new advances in computer science have opened the door towards more unique methods for organizing the underlying data.

Recently, there has emerged a new trend in database engineering that aims to move beyond the traditional relational database model towards new methods for storing and interacting with data. Termed NoSQL, which stands for *Not only SQL*, the development of which has been driven by a need to solve inherent limitations in relational databases at *web-scale* (scale in which user count is measured in the millions and billions), but has also understood that a *one-size fits all* approach to data storage is not only sub-optimal for many domains, but may in fact be a limiting factor against developing richer and more interactive applications.

While the relational database model has traditionally held the row as the base unit of interaction (meaning query operations are oriented around filtering out unneeded rows in relational tables), new databases have been developed that support new interaction models. Some such as MongoDB[19] utilize *documents* as their basic data model, which are collections of semi-structured properties on a single object; this has the benefits of improved storage performance and enables flexible data integration because the documents themselves can have an arbitrary arrangement of properties, with the schema being enforced by the application and not by the database. Its shortcomings are that it forces the application developers to carry the burden of data management and integrity in their application as the data store itself provides no mechanisms for supporting semantic integration. Similar approaches have been taken with more traditional object databases such as Caché[20].

Likewise, databases such as Neo4J[21] and Blazegraph[22] have implemented graph-based layouts where data is modeled not as documents or rows, but as nodes on a relational graph. This provides the flexibility of schema-less databases, as well as additional support for modeling additional types of relationships beyond associating an object to its data properties. Given these recent advances, it is worth considering whether the existing data models can be modified to support these new storage paradigms or whether new data models can be developed to further utilize these advances. This will be addressed in the following chapter.

3.4 APPLICATION-SPECIFIC IMPLEMENTATIONS

The data models discussed in the preceding sections have been largely limited to theoretical discussions and have seen little to no development or implementation efforts. However, prior research efforts have led to the development of several temporal GIS applications. Projects such as *STARS*, *STIS*, and *BoundarySeer* have resulted in domain specific applications which aim to provide tools for working with spatio-temporal data to solve specific research challenges (such as reconciling boundary changes over time) [130]–[132].

TGRASS and *EDGIS* are more general-purpose temporal GIS applications. *TGRASS* builds upon the well-known *Grass* GIS environment[23] and leverages components of the data models described in this chapter. While it does provide some *Application Programming Interface (API)* support for interacting with other research environments (such as the *R* statistical programming

[19] https://www.mongodb.com
[20] https://www.intersystems.com/products/cache/
[21] https://neo4j.com
[22] http://blazegraph.com
[23] https://grass.osgeo.org

environment[24]), it is largely focused on enabling temporal support for existing *Grass* application modules [133]. *EDGIS* is a new type of temporal GIS which implements Worboys *ST-Object* model and enables rich interaction with spatio-temporal data [134].

What all these projects have in common is that they represent new systems specifically designed for working with spatio-temporal data (with the exception of *TGRASS* which leverages the existing API framework of *Grass*). This means that researchers are required to move their existing research workflows into these custom systems. While that does allow for optimizing a software environment for the specific task at hand, it not only limits the utility, but also presents a potential limitation as the size and scope of the datasets grow. Coupled with the proven value of large-scale spatial data warehouses, there exists a desire to develop a data management platform that is removed from a specific application environment that allows users to access the necessary spatio-temporal data within the existing software tools already in use by domain researchers.

3.5 Conclusion

In conclusion, this chapter evaluated four common spatio-temporal data modeling approaches, *Time slicing, ST-Composite, ST-Object, and 3-domain*. Each data model provided a unique perspective by which to model and interact with spatial data, but with several limitations that may impact their utility in public health research. Likewise, we have seen that there have been limited attempts at integrating temporal data support into existing spatial applications, and even early work in designing new types of GIS applications that treat time as a first-class citizen. However, there remains room for expanding upon existing modeling approach, as well as developing an open data storage environment which can be accessed by multiple researchers in the software environments of their choice. In the next chapter, we will outline our data modeling approach which aims to address these issues and utilize fundamentally new types of data storage approaches in order to improve upon the state of the art.

[24] https://www.r-project.org/

Chapter 4. DESIGN AND ARCHITECTURE OF TRESTLE

In the previous chapter, we outlined a number of prior data modeling approaches that have aimed to improve support for temporal data within GIScience applications. In this chapter, we will introduce the *Trestle* system and data model which takes a novel approach for managing spatio-temporal data through a graph-based approach. In brief, our approach extends existing efforts (specifically the 3-Domain model) and aims to not only improve upon limitations, but also introduce new methods for integrating disparate datasets. Section 4.1 introduces the idea of data integration and curation as a key component of spatio-temporal data modeling. Section 4.2 describes the design goals for the Trestle system as well as the overall project architecture. Section 4.2.1 gives a brief introduction to knowledge representation approaches for spatio-temporal data management. Section 4.2.2 outlines the Trestle data model and representation. Section 4.2.3 describes the implementation of the Trestle data management application through an example of loading data from the GAUL dataset. Section 4.3 evaluates the Trestle system against the desiderata described in Section 3.2.

4.1 Data Integration and Data Curation

The approaches described in Chapter 3 all have a number of distinct strengths, as well as some limitations; but beyond the nuances of the various approaches one additional limitation remains. Each approach, only addresses *one* of the challenges faced by public health researchers, namely accessing historical data of a given dataset. Consider the example given in Section 2.2.1; when building a map of disease risk, researches find themselves needing to leverage multiple datasets that each contain a piece of information necessary in constructing their map of the world. Integrating these multiple datasets is not something directly taken into consideration by the approaches described in the previous chapter and is largely left as an exercise for the user.

While it is possible to merely extend the spatial objects with new data properties gathered from multiple sources, this approach struggles as the sheer volume of information increases. Indeed, for the ST-Composite and ST-Object approaches, if the number of data properties for a given object is significantly greater than the amount of spatial change of the object they are effectively required to implement an entirely separate data storage solution, in order to account for the non-spatial data properties. Given these two challenges, the sheer volume of data and the multiplicity of datasets required for a given research problem, it is desirable to have a more flexible method for managing complex data relationships within and between ST-Objects. Indeed, it becomes clear that it is impossible to separate the challenges faced in object modeling from those faced in spatial data curation and an effective modeling solution should aim to address both sets of challenges.

While data curation encompasses a number of distinct issues; including, but not limited to, data provenance, discovery, context metadata, etc. [135]; one in particular is of significant relevance to this domain, which is the challenge of data heterogeneity. Most curation efforts face 3 major types of data heterogeneity [136]:

- *Syntactic heterogeneity*: Where data is stored in a number of different formats (e.g. Shapefiles, GeoJSON, database tables, object databases).

- *Structural heterogeneity:* Where the data is structured in different formats with common data properties annotated or stored in different ways (e.g. the boundary of a county may be referred to as *geom*, or *boundary*, in different datasets).
- *Semantic heterogeneity:* Where the meaning of terms and properties varies between the datasets (e.g. a population count indicator might be computed using multiple methods in different collection contexts).

When considering modeling data as ST-Objects, we can extend this typology to add an additional classification, which we term *object heterogeneity*. This is the idea that as both the temporal scale of object lifetimes and the amount of available information increases, so does the complexity of managing the internal states of ST-Objects. Meaning, which data properties are valid at time given time point for a specific ST-Object. What is needed is a flexible approach for defining a minimal set of concepts that allow mapping spatial object states and relationships into an internal representation that can easily be extended as more concepts and data repositories become available.

4.2 TRESTLE

With this in mind, we now introduce the technical design and implementation of the *Trestle* system, a high-level overview of which is given in Figure 14. The key innovation of Trestle is similar to the one developed in May Yuan's *3-Domain* model and described in Section 3.2.5. In her model, Yuan was able to effectively categorize data into three major groups, *temporal, spatial,* and *semantic* and architected her data model towards cleanly representing these three data types. For Trestle, the key categorization, pertains not to data types, but relationship types. Here, we can classify relationships into two major categories. *External* relationships between two ST-Objects (such as contains, before, after, etc.) and *internal* relationships between an ST-Object and its associated data properties. While this second relationship type may not be immediately apparent, its importance increases as the temporal scope of ST-Objects increases. Meaning, as more data properties become associated with an ST-Object these internal relationships become more critical in reasoning about the internal state of the given object. Likewise, as more datasets are linked together, the relational *graph* layout becomes more natural and intuitive as individual ST-Objects can link between various associated properties from disparate datasets.

Modeling data properties as individual entities linked to a larger object is similar to the approach developed by IBM's *Clio* system [137], and further solidified in various incarnations of *graph databases*. The benefit is performance, ease of implementation, and flexibility in extending the system with more complex relationships and concepts as they become available. Taking this approach, however, introduces two major challenges. The first is in translating between the data formats and layouts common geospatial research (such as ESRI shapefiles) and the underlying graph layout. Graph systems require unique interaction models and are not necessarily intuitive to users familiar with traditional data layouts and formats. In order to address this, the Trestle project contains a Java based *management application* which performs the data translation and management, presenting the user with a simple and unified interface for accessing the necessary data and performing complex spatio-temporal queries. The management application and its functions will be described in Section 4.2.3. An example of this data translation is given in Figure 13 which shows a selection of time-series data for the ACS dataset translated into the graph-layout utilized by Trestle. The example graph shows only a single ST-Object, rather than the entire example dataset.

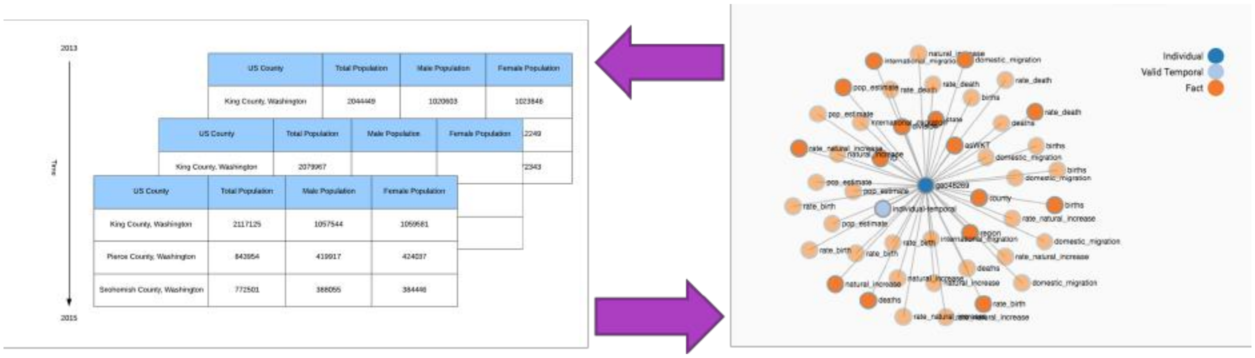


Figure 13: Translating between table and graph data layout.

This figure shows one of the primary utilities of the Trestle application, translating between traditional data formats and the graph-based layout that underpins the system. This ability to smoothly transition between the two formats is transparent to the user enables both Trestle's flexibility and ease of use.

The second challenge pertains to classifying the various relationships and maintaining the necessary context of the individual data properties. Each dataset contains unique context and metadata which influences the use and integration of the dataset members; this additional context often contains the necessary information for interpreting the data property and utilizing it correctly. What is needed is a flexible way of linking datasets together, which may contain their own meaning and terminology for the various data indicators. To address this challenge, we have taken an approach which builds off of ideas developed for the *semantic web*; namely, data sharing ontologies. The design of the Trestle ontology will be given in Section 4.2.2 but before doing so, Section 4.2.1 will give a brief overview of how ontologies have been used to organize data in both spatio-temporal data modeling as well as other semantic domains.

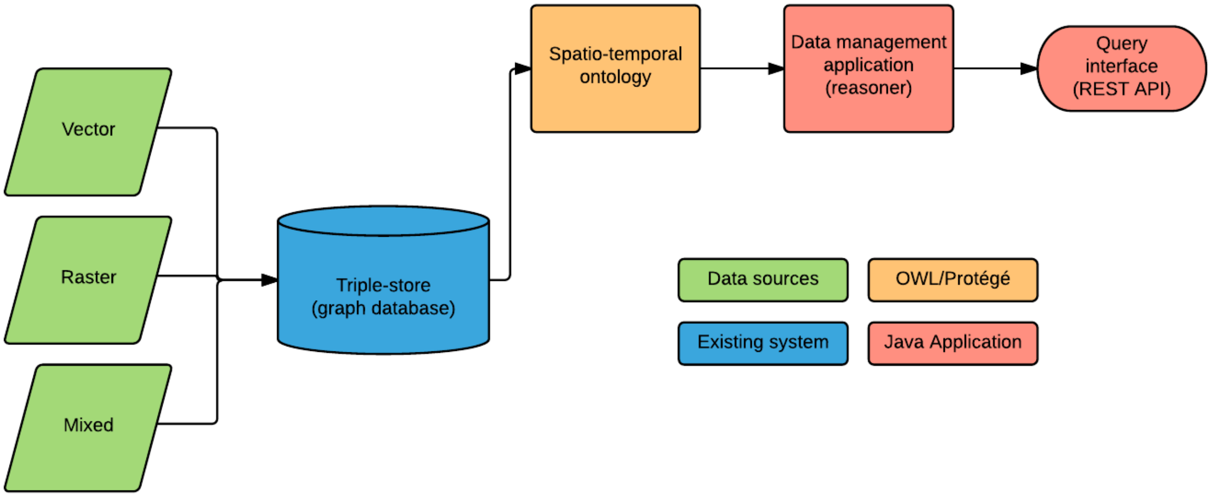


Figure 14: Trestle application design and architecture.

In this architecture diagram, the green components represent data sources that are loaded into Trestle. The blue component is the third-party database used by Trestle, coupled with the custom ontology (in orange). The red components are the Trestle management application and query interface.

4.2.1 *Organizing data with ontologies*

One approach to effectively integrate and manage the disparate information sources, made use of in Trestle, is found in the field of *ontologies*. Originating within the philosophy and mathematics disciplines an ontology defines, in a philosophical sense, an explicit, but partial, account of a certain conceptualization [138]. In short, an ontology attempts to make explicit a certain way of viewing and reasoning about a given knowledge domain. With the advent of modern information systems and the rise of the Internet, ontologies have taken on a new meaning within information science. Beyond just a specific conceptualization, what constitutes an ontology cannot be divorced from the idea of sharable knowledge [139]. Indeed, as Smith and Mark have made clear, much of the value derived from modern ontologies is from their ability to facilitate the reusability of data and their enablement of interconnected systems based on agreed upon conceptualizations [140].

It is this data sharing feature that has so intrigued computer scientists and data curators. Ontologies provide the ability to share not only raw pieces of information, but also the necessary meaning and context to make use of that information within different applications. Ontologies have been successfully used to manage and integrate disparate data sources across a significant number of scientific domains. Ranging from history [141], biology [142], public health[143], [144], and computer security [145], [146], researchers and practitioners have successfully developed knowledge representation systems that enable domain knowledge to be encoded in a computable format and structured in such a way as to be easily stored and integrated between multiple, discrete data sources.

It is important to mention that this dissertation is not an ontology research project. The innovation of this endeavor is not in developing a particularly novel knowledge representation that more accurately models the geophysical world; instead, this project utilizes a simple spatio-temporal ontology to help surface integration points between heterogeneous datasets and to model the relationship between user defined entities and the various properties and relationships that make up those objects. Think of it as less of a rigid structuring of the world, and more of a loose typology that defines points of interaction between datasets that aids in surfacing latent semantic concepts within the user defined data classes. In the end, the underlying ontology used by Trestle is an artifact of the design goals, not a goal in and of itself. It is through this feature that Trestle is able to effectively organization information and convey both data and context to the user, something existing approaches are unable to do. In the end it will be hidden from the user, which will interact with the system through a defined query interface, provided by the management application, but without being forced to deal with the vagaries of categories, ologs, or existential questions about the true nature of rivers.

When attempting to model these shared conceptualizations of domain knowledge (such as geography or disease surveillance), there are a number of different approaches to take. Frank (2003) has identified five *levels* of ontology development ranging from the modeling of physical reality to understanding cognitive agents [147]. At each ascending layer, the level of abstraction increases, as does the utility towards modeling domain specific problems. For this project, our ontology lives firmly in Frank's 3rd tier, in that it attempts to construct spatial concepts from sets of underlying properties and objects. This means that it provides a layer of abstraction above traditional geospatial ontologies that attempt to define things such as geometries or geographic features; the goal is that the ontology can be expanded to support existing spatial datasets and definitions by providing a simple set of geographic and temporal relations that provide the building blocks for constructing unified spatial objects from underlying facts and attributes, either by direct assertion or via logical inference.

An example of the potential for ontologies for data integration is the SDIDS project, previously described in Section 0, and designed to support the integration of malaria surveillance data into a structured and normalized format [61]. This ontology provides an expansive conceptualization of entities related to disease control and surveillance; such as different health facility types, census measurement methods, biological tests, etc. These information sources can be combined to provide a robust spatio-temporal platform for supporting disease surveillance modeling, with the Trestle ontology providing the spatio-temporal primitives and the SDIDS ontology providing the contextual information for the underlying data properties. The project described in this dissertation was originally conceived as an extension to the SDIDS platform to handle the various mapping tasks required.

Within the field of geography there has been a significant amount of preceding work in the area of spatio-temporal ontologies; however, these efforts have largely been focused on either the temporal, or spatial side of the equation [148], [149]. In general, most of the data science field has focused on developing temporal data models, while GIS researchers have held spatial data as their primary domain [110]. In part because of this cultural divide, only a few concerted efforts have been undertaken to integrate both dimensions into a unified ontology [110], [150]–[152].

While often conflated together, it is important to make a distinction between ontologies, as a categorical and knowledge representation topic, and ontologies as they are commonly found in *semantic web* projects. The *semantic web* is a collection of technologies that were developed to improve the interoperability of internet-based data repositories. In this conception, each piece of information contains a *Universal Resource Identifier* (URI)[25] which references the web accessible location that maintains both the data point, as well as the necessary context for interpreting that piece of information.

To illustrate this through a spatial example consider that a county in the Topologically Integrated Geographic Encoding and Referencing (TIGER) dataset, the boundary data provided by the US Census Bureau[26], would contain an *area* data property and might also contain an *average population count* data property which includes a URI that indicates the information came the ACS dataset. This means that the TIGER dataset does not need to maintain information about the population count measure (such as how it was collected or what the base units are), even though that information is critical when using the measure in spatio-temporal research. When a user performs a query such as *Find the 10 largest US counties which have the highest population density,* the system contains all the required information about the *area* property, but not the *population count*, so at query time, it fetches the additional context from the ACS system (which maintains the knowledge about the *population count* measure) and incorporates that into the logic required to answer the query. This means, that instead of requiring each repository to maintain the required information for all related datasets, it allows each repository to maintain the information for its respective datasets and communicate that knowledge, as needed, to external users.

Beyond the remote data linking capabilities, the semantic web also includes a data representation format that expresses information as sets of *triples*. A triple is a data tuple of three properties: *Subject, Property, Object*. Which indicates the value of a given property attributed to specific entity. An example of a triple assertion is: *<gaul:Manhica,gaul:administrativeLevel,gaul:Level2>*. This triple is a uniform way of stating that the entity *Manhica* (a county in Mozambique) is a Level 2 administrative unit[27]. Or more precisely, the subject *Manhica* is known to have an attributed property *administrativeLevel* which has the value *Level2*, as described by the *GAUL* dataset. The identifiers are given in *short-form* notation, in which rather than writing down the entire URI, only a short prefix is used, and the implementation maintains a mapping from prefixes to full URIs.

Given this representation of triples, the next challenge comes with transferring these triples between repositories, over the web. For that, the semantic web introduces a structured data format, based largely on XML (Extensible Markup Language)[28], which allows for interoperability between systems. These representations vary in their verbosity and level of expressivity. The simplest is *Resource Description Framework* (RDF) which defines the base triple format, as well as a limited set of specific data relationships relationships. RDF allows arbitrary extension of relationships that can be defined by the various repositories, such as *gaul:administrativeLevel*, which is a relationship specific to the GAUL dataset. At the other end of the expressivity spectrum is *OWL* (Web Ontology Language). OWL builds on RDF and adds a number of additional relationships (such as *sameAs*, which specifies that two identifiers, though distinct, describe the same entity) and restrictions (such as *someValuesFrom*, which specifies that the objects of a given property, must also be members of a specific class).

[25] A URI refers to a globally accessible, and unique, identifier which can be used to reference a unique piece of information, or location across the global internet. An example of a URI would be: http://www.w3.org/XML/1998/namespace. Which refers to the web page found at the *w3.org* website.

[26] https://www.census.gov/geo/maps-data/data/tiger.html

[27] The GAUL dataset defines country boundaries as Level 0, state (or province) boundaries as Level 1, and county (or district or parish) boundaries as Level 2.

[28] There are additional formats such as *Turtle, JSON-LD* and *N-Triples*, which are more succinct ways of representing RDF triples, but these are largely derivatives of the base RDF-XML and OWL-XML formats.

One powerful feature of OWL that is missing in RDF, is *inference*. Inference is the ability to build additional knowledge from an initial dataset, based on a specified set of rules. OWL supports a number of different inference profiles [153], [154] each of which provides different sets of rules that cover the spectrum between maximum logical expressivity and inference performance. An example of an OWL rule is *owl:transitiveProperty*[29]. This rule specifies that a given object relationship is transitive for all members in that chain. An example in the spatial domain would be the spatial relationship *contains*. The definition of *contains* derives from Egenhofer [111] and specifies that for Object A to be fully contained within Object B, no part of Object A can fall outside of Object B.

This means, we can specify the relationship as transitive and apply the inference engine (also referred to as the *reasoner*) to build additional relationships. This means, that for the TIGER dataset, we can specify that *King County* is contained within *Washington State*, and that *Washington State* is contained with the *United States* and allow the reasoner to discover that *King County* is contained within the *United States*. While this is a trivial example, it shows the flexibility of applying an inference engine to extract larger sets of relationships from smaller, partially specified input datasets. An example of specifying these relationships in OWL-XML is given in Code 1. More details on the semantic web and inference engines can be found in [155].

```
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://nickrobison.com/dissertation/trestle.owl"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI="http://nickrobison.com/dissertation/trestle.owl"
     versionIRI="http://nickrobison.com/dissertation/trestle.owl/0.9.5">
    <Prefix name="" IRI="http://nickrobison.com/dissertation/trestle.owl"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Import>http://www.opengis.net/ont/geosparql</Import>
   <ClassAssertion>
      <Class IRI="#GAUL"/>
      <NamedIndividual IRI="#aeroporto:1990:2013"/>
   </ClassAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#has_component"/>
    <NamedIndividual IRI="#ManhicaUnion"/>
    <NamedIndividual IRI="#manhica1:2009:2013"/>
  </ObjectPropertyAssertion>
  <ObjectPropertyAssertion>
    <ObjectProperty IRI="#has_component"/>
    <NamedIndividual IRI="#ManhicaUnion"/>
    <NamedIndividual IRI="#manhica2:2009:2013"/>
  </ObjectPropertyAssertion>
</Ontology>
```

Code 1: Example of OWL-XML for Cidade de Maputo from the GAUL dataset.

[29] https://www.w3.org/TR/owl-ref/#TransitiveProperty-def

4.2.2 *Trestle ontology design*

Given the previous introduction to ontologies and knowledge representation, this section details the design of the Trestle ontology. The ontology itself is implemented using the *Protégé* editor and framework [156] and defines a small set of basic classes and primitive relationships, which can be augmented in domain specific applications. A visual overview is given in Figure 15 and will be described further in this section.

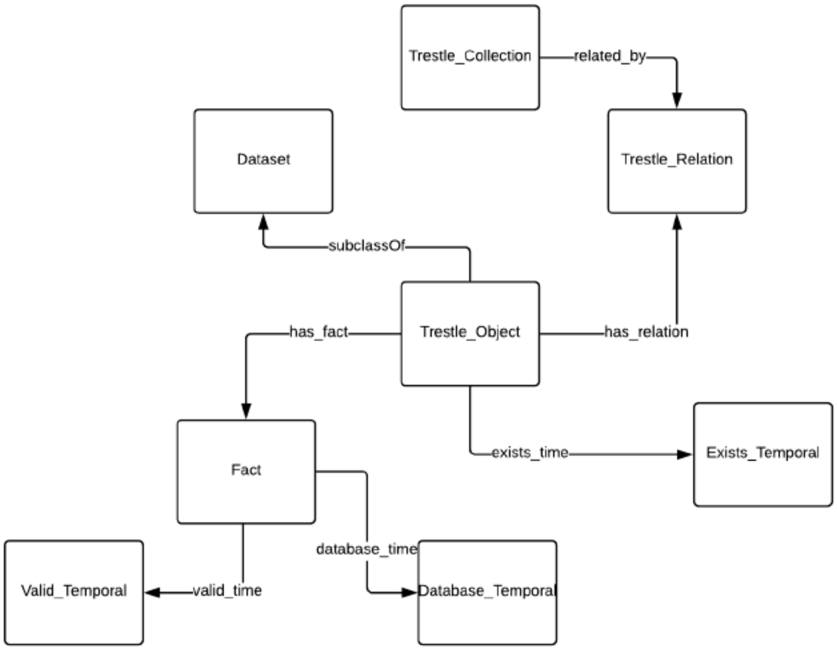


Figure 15: Trestle ontology layout.

This figure shows the layout of the Trestle ontology and how the various classes interact with each other.

4.2.2.1 Trestle_Object

At the core of the ontology is the Trestle_Object class, which provides the central mechanism for defining spatio-temporal objects and their associated properties and relationships. A Trestle_Object roughly corresponds to an ST-Object in the Worboys model [125], or a *semantic object* in Yuan's [110]. A Trestle_Object contains two data properties, *exists_from* and *exists_to*, which defines the temporal bounds for which that object exists. This is required by the ontology and management application avoids the ambiguous existence problem described in Chapter 3 in which prior data models have no native concept of object lifetimes. While a Trestle_Object can have an unbounded existence (and thus no *exists_to*) property, it must have an *exists_from* property. It should be noted that all temporal intervals in Trestle are represented as *open-closed* intervals, which means that the start of the interval is inclusive of the initial timepoint, but *exclusive* of the ending timepoint. So, a Trestle_Object which exists only for the year 2013, would be represented with the *exists_temporal* of [01-01-2013, 01-01-2014), which includes January 1st, 2013, up to the last day of the of 2013 (or the last instant of the year 2013 supported by the temporal data type).

A Trestle_Object also contains zero or more *has_fact* relationships, which point to the Trestle_Fact entities associated with the object. A Trestle_Fact represents the value of specific data property at a given temporal intersection[30]. Modeling spatial objects as collections of related facts is one of the key innovations of Trestle and a core component of its flexible data layout model. Whereas it is intuitive to model relationships between objects as a graph layout (e.g. *Object A is contained within Object B*); in reality, the primary relationship type associated with a Trestle_Object is between the object and its data properties, especially as the amount of available information for a given object increases. By capitalizing on this relational nature, we can utilize the same computational primitives to model relationships between objects as within objects. In short, from the perspective of the underlying datastore, there should be no distinction between a spatial relationship (such as *contains*) and an intra-object relationship such as *has_fact*, even though one describes a relationship between two *ST_Objects* and the other describes a relationship between an object and its data properties. They are both represented as *edges* between *nodes*, in a relational graph, that only differ in their logical categorization by the reasoner.

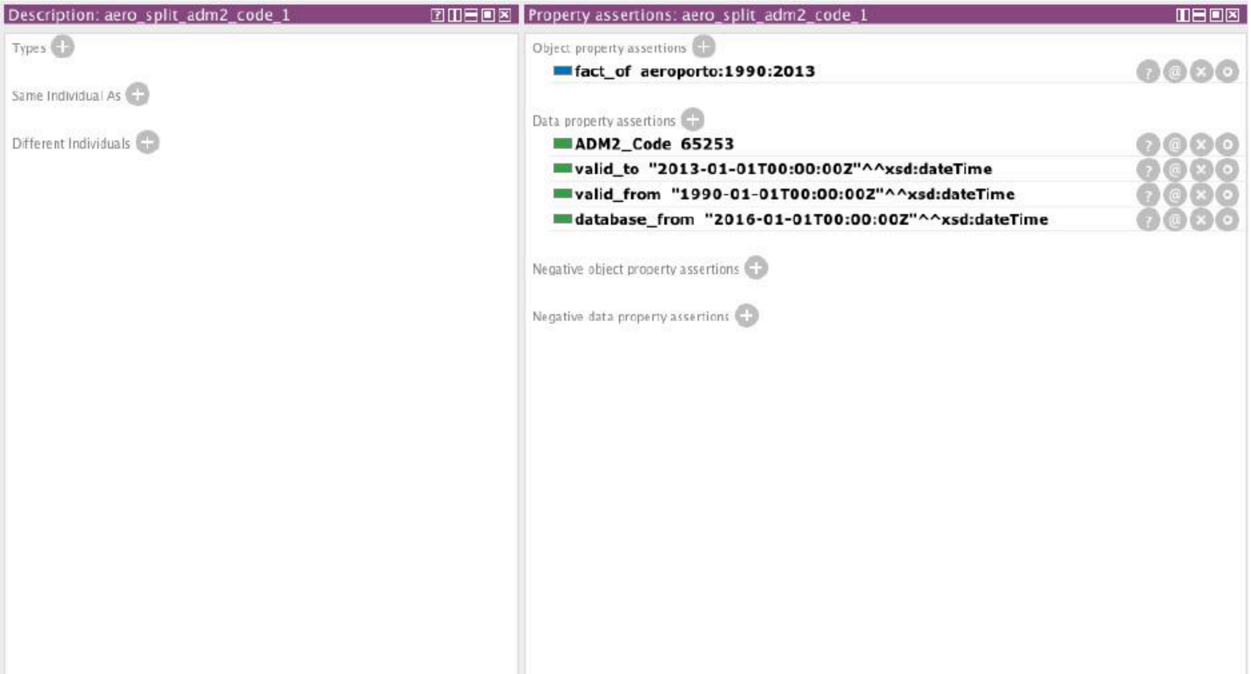
A Trestle_Fact entity contains three pieces of information: a *valid temporal*, a *database temporal*, and a *dataproperty*. The valid temporal represents the real-world temporal range of the given fact. It can be represented as either a point (e.g. valid only on March 1st, 2001) or an interval (e.g. valid from March 1st, 2001 to March 1st, 2002). The *databse temporal* is the mechanism by which Trestle achieves *bi-temporal* support. This defines the range of time (from the perspective of the datastore) which the fact is the most recent value of that data property. While *valid temporals* may be represented as either a point or an interval, *database temporals* must be represented as intervals (following the same rules as the *exists temporal*).

Beyond storing temporal data, facts also serve the function of conveying data values as well. But this is where the dynamism and flexibility of the data model comes into play. The facts do not have *fact* values, instead they have data properties that are named for the actual data properties, and values that match. This is illustrated in Figure 16, which shows the data property assertions for a fact entity, associated with *Aeroporto, Mozambique* and conveying the *ADM2_Code* data property. Here we can see the three temporal assertions (*valid_from, valid_to, database_from*) as well as the *ADM2_Code* assertion which contains the actual fact value (65253).

This is an important technical point because of what it enables. Namely, it means users can directly query for the data properties they are interested in, and not indirectly for fact nodes which may or may not contain the correct data property. This allows the Trestle_Facts to be implemented in a generic fashion and relies on the underlying datastore to disambiguate which Trestle_Fact contains the necessary piece of information. This means that queries such as *What was the population count of King County, Washington in 2015*, can be directly mapped to the underlying graph nodes, without any intermediate translation. This also means that answering queries such as *Which US counties had the largest populations in 2015*, can be quickly answered by the application, as it

[30] As our ontology natively supports *Bi-temporal* time, we will use the term *temporal intersection* to denote the combination of *valid time* and *database time* when applied to a specific fact. As an example, consider the phrase "The population count of King County for the year 2000 is 2 million, as recorded in the database on March 1st, 2001", the *temporal intersection* of this fact would be: *valid time*: [1/1/2000-1/1/2001) *database time:* [3/1/2001,).

maintains the ability to directly query for fact nodes with the given data property name and from there, filter down to only the facts valid at the given temporal point.



Description: aero_split_adm2_code_1
Types
Same Individual As
Different Individuals
Property assertions: aero_split_adm2_code_1
Object property assertions

fact_of aeroporto:1990:2013

ADM2_Code 65253
valid_to "2013-01-01T00:00:00Z"^^xsd:dateTime
valid_from "1990-01-01T00:00:00Z"^^xsd:dateTime
database_from "2016-01-01T00:00:00Z"^^xsd:dateTime

Negative object property assertions
Negative data property assertions

Figure 16: Data property assertions for a single *fact* entity in Protégé editor.

The five *data property assertions,* for this entity from the GAUL dataset, in this figure that this fact represents the value of *65254* for the property *ADM2_Code.* The valid interval for this fact is from 1990 to the last date of 2012. This is the most recent version of this fact value, since there is a *database_from* assertion, but no *database_to* which indicates this is an open-ended *database temporal* interval.

An additional feature of each data property is that it contains a unique datatype. This type has two properties. This first is that it corresponds to a concrete type representable by most computing environments (e.g. integer, string, floating point number, etc.). The second is that it also contains a unique URI which further specifies both the application dependent type (e.g. that this datatype corresponds to a population count) and the repository which contains additional context about that specific piece of information. One example is the representation of spatial information. By convention, spatial data is stored in the *Well-Known Text* (WKT) format, which defines a specified layout of point information pertaining to a spatial representation. Some examples of data in WKT format are given in Table 11. In our data model, WKT values are stored as strings, but identified with the datatype URI *http://www.opengis.net/ont/geosparql#wktLiteral*. This differentiates them from other types of string values, such as place names, and allows the application to make decisions about how it handles the data. In the case of WKT values, this additional context allows the underlying datastore to optimize for the spatial values and apply the appropriating indexing mechanisms required for performant spatial queries.

Location	WKT Representation
Authors' Academic Office Seattle, WA	POINT(-122.3402250, 47.6241320)
Yosemite National Park (Boundary Box)	POLYGON((-119.885 38.0832, -119.2172 38.0832, -119.2172 37.6234, -119.885 37.6234, -119.885 38.0832))

Table 11: Examples of WKT representations of various US geographic entities.

This is an important component of the Trestle system. The data model makes no effort to understand what type of information is being stored, beyond the concrete type which it needs in order to represent that data (e.g. that the given value is represented as a floating point, a string, or an integer). The semantic meaning of the information is pushed up in the user application level. This means, that the data model requires no special knowledge in order to expand to support multiple types of data. If a user application requires a distinction between two population counts, which may have different sampling methods, both can be stored in the database, but with different URIs. This capacity is leveraged to enable additional data types, such as multi-language support where a data property, such as name, might have multiple representations in different languages. The data model supports representing and disambiguating these datatypes, a unique feature of the Trestle system.

4.2.2.2 Relationships Types in Trestle

At its core, Trestle is an ontology of relationships. The basic primitive, relating a Trestle_Object to its associated facts, is a type of relationship, albeit one that is contained within the semantic boundaries of the object itself. In addition to these types of relationships, Trestle also contains the notion of spatial and temporal relationships, based on the well-known relations described in the literature [111], [113], and a selection of domain specific relationships that describe the relationship of a given object before and after its existence. A summary of the relationships supported by Trestle is given in Table 12 on page 72.

Spatial and Temporal Relationships

The basic spatial and temporal relationships supported by Trestle largely mirror those found in traditional spatial databases but differ in two specific ways. First, these relationships are not automatically computed, but must be either manually asserted, by the user, or logically inferred, by the reasoner. Determining spatial and temporal relationships at query time is supported by a combination of the underlying data store and the management application. This means the application fully supports queries such as *Find all the households that fall within a given hospital catchment area*; likewise, the management application allows persisting these relationships into the ontology, which can then be used to support query answering, rather than requiring relation computations for each individual query.

Second, many of these relationships have additional logical properties which are not often reflected in traditional data store. The first of these properties is *transitivity*, which indicates that a given logical relationship expressed between two entities (A and B) is also true for a third entity (C) if the original relationships is held between B and C. For example, *Seattle* is *inside King County,* which is *inside Washington State*. Thus, through *transitivity* we can infer that *Seattle* is *inside Washington State*. This holds for temporal relationships as well; if *breakfast* comes *before lunch* and *lunch* comes *before dinner*, then we know that *breakfast* comes *before dinner*. The second logical extension is *symmetry* (and its logical converse *asymmetry*). This specifies that a relationship expressed in a specific direction (e.g. A to B) also holds true in the inverse direction (B to A). For example, consider the statement *King County, WA* is spatially disjoint from *King County, TX*. By the nature of the symmetric property, we can also infer that *King County, TX* is *disjoint* from King County, WA, without being required to compute any type of spatial calculation.

The final logical extension is *inverse*, this allows specifying that a relationship expressed in a specific direction (A to B) has a corresponding relationship that holds true in the opposite direction (B to A). An example is the previous illustration of *Seattle* being *inside King County*. The *inverse* logical property implies that *King County contains Seattle*. Since the *contains* relationship is also expressed as *transitive*, we can automatically infer that *Washington State contains Seattle*. A list of spatial and temporal relationships known to Trestle is given in Table 12, along with their corresponding well-known counterparts and any *additional* logical properties maintained by *Trestle*.

The power of these additional logical assertions becomes apparent as both the data size and query complexity scales beyond trivial use cases. Consider an example dataset of household survey data that was collected in the field and annotated without a direct spatial location, merely names of cities where the households were located. The question then becomes, how to answer the query *Find all the household survey responses for a given province*? The traditional answer would be to assign some location information to each entry, in order to place it within the given city; but Trestle allows for a different approach. Each entry in the data can be marked with an *inside* relationship to the city mentioned in the entry. If the cities themselves have location data associated with them (from a different dataset such as GAUL), then Trestle can automatically infer that each survey response is *inside* (and conversely *contained_by*) the province the city is located in. These relationships are directly expressed in the database and are not required to be computed at query time.

Additional relationship types

Beyond spatial and temporal relationships, Trestle also supports two additional relationship types. The first, is *event* relationships. These relationships define temporal points which correspond to semantic events. Meaning, something occurred in the lifetime of the object which from the perspective of the domain use case, is worth marking. While the management application allows for marking arbitrary object events, the ontology natively supports three event types: c*reated, destroyed,* and *split/merge*.

The first two are fairly straight forward and define the bounds of the temporal scope for which a *Trestle_Obiect* exists. *Split/Merge* events are specific to geographic boundaries and denote the spatial history of a given object over time. These events contain some additional semantic content which stipulates that a split/merge event only occurs in two specific cases. First, when a given boundary *splits* into multiple new boundaries, such that all the area of the original object is accounted for, with no additional area included or left out. Second, when multiple boundaries are merged into a single, new boundary such that all of the area of the original boundaries are accounted for, with no additional area included or left out. This means that when a *split/merge* event is present, the spatial area covered by the input object(s) before the event, exactly corresponds to the spatial area of the output object(s) merely in a different configuration. This allows for answering queries such as *What administrative boundaries were merged into Cidade de Maputo in 2014?* This will be discussed further in Chapter 5.

The final relationship type supported by Trestle is object relationships, while this largely falls under the category of *collections* (which is described in the following sub-section) at this point it should be mentioned that the type of object relationships supported by the ontology are true object relationships, in that a Trestle_Object inherently contains the temporal period for which the object exists. Thus, Trestle can accurately answer the question *Which objects came before Cidade de Maputo* because *Cidade de Maputo* is temporally bounded. This is possible even if no facts exist for the object over the entirety of its lifetime. This means that Trestle can answer queries such as *For what percentage of years is census data missing for Zadar, Croatia?* This has powerful implications for research tasks such as data quality assurance or identifying possible scopes of projects given the amount of available data and is something existing data models struggle to account for.

Trestle Relationship	Type	Corresponding Relationship	Transitive	Symmetric	Inverse	Reference
Covers	Spatial	Covers	No	No	Covered_by	[111]
Contains	Spatial	Contains	Yes	No	Inside	[111]
Disjoint	Spatial	Disjoint	No	Yes		[111]
Inside	Spatial	Inside	Yes	No	Contains	[111]
Covered_by	Spatial	CoveredBy	No	No	Covers	[111]
Spatial_Equals	Spatial	Equals	Yes	Yes		[111]
Spatial_Meets	Spatial	Meet	No	Yes		[111]
Spatial_Overlaps	Spatial	Overlap	No			[111]
After	Temporal	After	Yes	No	Before	[113]
Before	Temporal	Before	Yes	No	After	[113]
During	Temporal	During	Yes	Yes		[113]
Finishes	Temporal	Finishes	No	No	Finished_by	[113]
Starts	Temporal	Starts	No	No	Started_by	[113]
Started_by	Temporal	Started-by	No	No	Starts	[113]
Finished_by	Temporal	Finished-by	No	No	Finishes	[113]
Temporal_Meets	Temporal	Meets	No	Yes		[113]
Temporal_Overlaps	Temporal	Overlaps	No	No		[113]
Created	Event	Created	No	No		[157]
Destroyed	Event	Destroyed	No	No		[157]
Merged	Event		No	No	Merged_of	
Merged_from	Event		No	No	Merged_into	
Merged_into	Event		No	No	Merged_from	
Split	Event		No	No	Split_of	
Split_of	Event		No	No	Split	
Split_from	Event		No	No	Split_of	

Table 12: Summary of Trestle relationships and their well-defined counterparts

This table lists the fundamental relationships support natively by the Trestle ontology. The *Corresponding Relationship* column maps between the name of the relationship in Trestle and the name of the relationship in the original research paper which the relationship is based on. In addition, this table lists any additional logical properties intrinsic to the relationship, as well as the corresponding *inverse* relationship. The *split/merge* relationships do not have a related research paper as they are unique to the Trestle system.

4.2.2.3 Collections: Sets of spatio-temporal objects in Trestle

The previous sub-section gave a brief introduction to object relationships in *Trestle* and provided some additional information as to their flexibility in supporting spatial and temporal relationships between objects. An additional use of these object relationships is in supporting groupings of Trestle_Objects known as *Trestle_Collections*. A Trestle_Collection is a grouping of Trestle_Objects, which contain some type of association, either temporal, spatial, or semantic. It can be thought of as a sub-graph of related Trestle_Objects within the larger relational graph of the database. These collections can be added to or subtracted from on-demand and can thus support a fluid concept of associativity, which is necessary for supporting spatial algorithms which require some type of stateful computation (such as the algorithms described in Chapter 5 and Chapter 6).

An example of a Trestle_Collection might be a custom base map created by an organization for use by their research teams. In order to obtain global map coverage, organizations often use datasets such as GAUL or the *Database of Global Administrative Areas (GADM)*[31] which feature extensive global coverage but may not be as current or detailed as what is publicly available from individual countries. For instance, the *GAUL* dataset maintains county level data across the globe, and sub-county data for 45 countries. In some case it is desirable, even necessary, to utilize more granular data such as the *TIGER* data; however, integrating this information with other datasets is challenging to manage and often requires duplicating the data to create custom datasets. This data fragmentation is difficult to maintain and necessitates manual updating if the underlying information changes (such as new measurements or boundary changes). In contrast, Trestle_Collections do not require this type of duplication since the relational nature of the data model means that the collection associations are always pointing to the original data fact, including both the most recent version and historical versions as well. This significantly reduces the amount of data duplication and administrative burden for updating and maintaining datasets. Likewise, an individual Trestle_Object can be a member of multiple collections and can be added to or removed from membership as necessary. This means, that if *Cidade de Maputo* has an updated spatial boundary, this change would be reflected across all collections it is a member of.

Collections also support a *strength* parameter, which allows representing object associations which may have an association with varying degrees of strength or certainty. This allows, at query time, for users to specify a restriction to only return objects with a relationships strength above a certain threshold. The benefit of this is that different applications will require differing levels of data certainty and in some cases, such as a data verification task, returning even the most ancillary piece of information is desirable, as it allows errors to be corrected and evaluated. One example of the importance of this capability is that automated methods for spatial matching objects inherently result in probabilistic matches; having the ability to support this uncertainty and present the information to the user, in order to make the final decision, is an important strength of the Trestle ontology. It should be noted, that the ontology itself provides no mechanisms for reconciling this uncertainty, nor does it infer or compute any probabilistic matches, it merely supports the ability to convey the uncertainty context to the user application where the appropriate decisions can be made. This *collection* relationship is a unique feature of Trestle that is not present in other data models; use of this feature will be illustrated in greater detail in Chapter 5.

4.2.3 *Trestle Application implementation*

Trestle is implemented as a *Java* application, building on top of a number of open-source components to create a management application that removes the need for individual users to directly interact with the ontology or the underlying graph database. Instead, interaction with the

[31] https://gadm.org

application occurs through either a set of *Application Programming Interfaces* (APIs) or a prototype web interface.

As illustrated in Figure 14, the management application is built on top of an existing *triple-store*, which is a special type of graph database [158] optimized for storing RDF triples and performing automated inference. Trestle is designed to support multiple triple-stores, but requires that they feature the following capabilities:

1. Support for the OWL 2 inference profile [153], specifically, the *owl:PropertyChainingAxiom* rule[32]. This rule is largely relegated to use in Trestle_Collections and is thus not critical to the core operation of *Trestle,* but the support greatly simplifies the implementation of more complex inference and association capabilities.
2. GeoSPARQL semantics [159], which provides spatial query support for RDF data; or, a comparable implementation which allows for performing at least a basic set of spatial operations, which can be augmented by additional code in the management application.

We evaluated a number of different triple-stores over the course of this project and implemented a selection of those systems as backends for *Trestle*. A summary is provided in Table 13, but in the end we utilized the *GraphDB* engine developed by *Ontotext*[33], which was originally known as the *OWLIM* project. All results reported in this dissertation are utilizing the *GraphDB* backend.

[32] https://www.w3.org/TR/owl2-primer/#Property_Chains

[33] https://ontotext.com/products/graphdb/

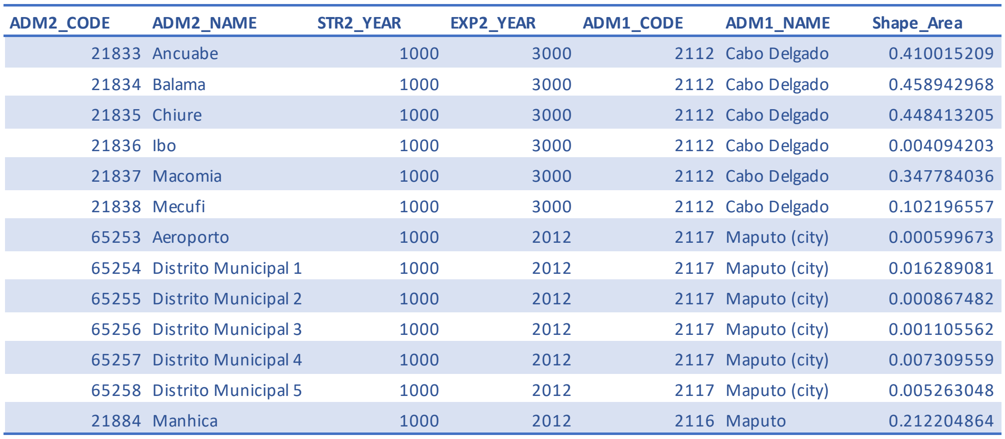
Database	Open source	OWL2	GeoSPARQL	Implemented
Oracle Spatial	No	Yes	Yes	Yes
Virtuoso	Yes	No[a]	Custom[b]	Yes
GraphDB	No[c]	Yes	Yes	Yes
Jena TDB	Yes	No	No[d]	Yes
Stardog	No[c]	Yes	Partial[e]	No
Blazegraph	Yes	Extensible[f]	No	No
Marklogic	No	Extensible[f]	Custom[b]	No

[a] Virtuoso 7.0 (which is not open source) provides the ability to specify custom inference rules.
[b] Provides a custom spatial functionality which closely mirrors the GeoSPARQL specification.
[c] Provides a free license with some limitations.
[d] Provides support for simple spatial distance and intersection calculations.
[e] Partial support for GeoSPARQL specification.
[f] Supports custom rulesets.

Table 13: Summary of triple stores evaluated for Trestle implementation.

This table lists the various triple-stores that were evaluated for the Trestle project. Some of the triple-stores were only evaluated based on reviewing the feature sets and documentation, while others were actually developed in code and evaluating using both synthetic and real-world data. The primary finding of this table is that combining spatial features and advanced inference capabilities is difficult to find in existing projects.

In order to illustrate the operation of the application, we will use as an example, loading and integrating the GAUL dataset of Administrative Level 2 (county level) boundaries. To refresh, the GAUL dataset contains information of the administrative districts of global countries, at varying levels of granularity, from 1990 to 2015, delivered as a set of individual shapefiles, one for each year. Each district in the dataset is assigned a unique identifier, which remains constant for its lifetime. If a district is reorganized, such as merged with another district, or dramatically resized, it is given a new identifier, which breaks the link between the old district and the new one. An example of the GAUL data is given in Table 14.



ADM2_CODE	ADM2_NAME	STR2_YEAR	EXP2_YEAR	ADM1_CODE	ADM1_NAME	Shape_Area
21833	Ancuabe	1000	3000	2112	Cabo Delgado	0.410015209
21834	Balama	1000	3000	2112	Cabo Delgado	0.458942968
21835	Chiure	1000	3000	2112	Cabo Delgado	0.448413205
21836	Ibo	1000	3000	2112	Cabo Delgado	0.004094203
21837	Macomia	1000	3000	2112	Cabo Delgado	0.347784036
21838	Mecufi	1000	3000	2112	Cabo Delgado	0.102196557
65253	Aeroporto	1000	2012	2117	Maputo (city)	0.000599673
65254	Distrito Municipal 1	1000	2012	2117	Maputo (city)	0.016289081
65255	Distrito Municipal 2	1000	2012	2117	Maputo (city)	0.000867482
65256	Distrito Municipal 3	1000	2012	2117	Maputo (city)	0.001105562
65257	Distrito Municipal 4	1000	2012	2117	Maputo (city)	0.007309559
65258	Distrito Municipal 5	1000	2012	2117	Maputo (city)	0.005263048
21884	Manhica	1000	2012	2116	Maputo	0.212204864

Table 14: Sample of GAUL dataset properties for the year 1990.

A small subset of data from the GAUL dataset, showing a selection of available data properties, their names and values.

One major challenge with working with type of information is transforming the dataset from its original *time-slice* layout, into the unified Trestle_Objects supported by *Trestle*. In order to accomplish this, data is loaded one year at a time, starting with the earliest year (1990) and continuing to the most recent (2015). This means that the Trestle_Objects will be built progressively as each new year of data is added; Trestle provides various strategies for reconciling changes when adding new data.

Before loading the data, the decision must be made as to how to model the input datasets as Trestle_Objects. These objects represent the changing state of a semantically unified object as it evolves over time. Trestle largely delegates the object modeling decision to the domain application and instead focuses on providing the basic primitives for managing and querying these objects. That being said, these primitives require each Trestle_Object to fulfill the following constraints:

1. Objects must have a spatial component.
   At this time, Trestle does not support non-spatial objects. Any information which does not describe a spatial extant, must be modeled as a Fact on a given Trestle_Object. This spatial value may change over the lifetime of the object, but each object must have at least spatial component for the duration of its lifetime. This is a temporary limitation of the management application is not an intrinsic limitation of the underlying data model.
2. Objects must exist for a temporal interval, not a single temporal point.
   While individual facts may be valid only for a specific temporal instant, objects must exist for a temporal interval, regardless of how short that interval actually is.
3. Facts cannot temporally overlap
   Trestle does not support multiple values of the same fact at the same temporal point. This avoids the need for the application to disambiguate the *correct* values for a given user queries. Modeling similar facts (such as reporting population counts via multiple sampling methods) can be accomplished by the use of unique URIs for each data property.

These constraints not only allow Trestle to effectively manage and query the underlying data, but also serve to provide a conceptual framework for thinking about how to transform data from traditional table layouts into more complex ST-Objects. In this example, the individual administrative districts are represented as Trestle_Objects with each new year of data being merged into the object as its loaded (the merging process will be described below).

Trestle provides a set of tools to easily define the mapping of input data in the form of tables, into unified Trestle_Objects. These mappings are referred to as *data definitions* and specify multiple properties of the Trestle_Objects including specifying stable object identifiers, listing data properties, determining relationships and constraining object existence.

An example of the data definition used for the GAUL dataset is given in Code 2. Currently, these definitions are constructed using the *Java* programming language, utilizing specific language features such as *annotations*. This is merely an artifact of the current prototype implementation and additional methods for creating these definitions can be implemented using data interchange formats such as *Cap'n Proto*[34], or through additional user tooling.

These *data definitions* play a crucial role in enabling support for both missing data and for retrieving subsets of available information. Multiple *data definitions* can be created for a single *dataset*, each of which is specialized for a given research project. An example would be *ACS* data linked to a given *TIGER* county; while *ACS* provides a multitude of data properties, only a few may be necessary for a given research problem. With Trestle*,* a user could create a *data definition* which only specifies the data properties required for their specific question thus reducing both the complexity of user interaction, as well as improving performance by only requiring the management application to deal with a reduced subset of the available information.

Likewise, methods for dealing with missing data become more focused on the individual objects, rather than the dataset as a whole. This means that while a given dataset may be specified to have a *population count* fact, Trestle does not enforce that constrain at load time. Instead, it allows the data to be added and then at query time, will raise an error if the user attempts to query for a fact that is not valid for that Trestle_Object at the given time point. The basic principle is that data completeness is as much a function of the domain application as is semantic interpretation of the individual data points. This means that the underlying Trestle datastore is not required to ensure that individual data properties are present and in the expected data format, instead, that decision is left to the management application and enforced by the constraints specified in the *data definitions*. If data is missing or incorrect for certain Trestle_Objects that issues should only affect the user if they actually need those specific properties.

[34] https://capnproto.org

```
@DatasetClass(name = "GAUL")
public class GAULObject {
    private final String objectID;
    private final long gaulCode;
    private final String objectName;
    private final byte[] validRange;
    private final Polygon shapePolygon;
    private final long adm1Code;
    private final String adm1Name;
    private final String status;
    private final boolean dispArea;
    private final long adm0Code;
    private final String adm0Name;

    public GAULObject(String id, long adm2_code, String adm2_name,
LocalDate startDate, LocalDate endDate, String wkt, long adm1_code,
String adm1_name, String status, boolean disp_area, long adm0_code,
String adm0_name) {
        this.objectID = id;
        this.gaulCode = adm2_code;
        this.objectName = adm2_name;
        this.validRange = DateFieldUtils.writeDateField(startDate,
endDate);
        this.shapePolygon = (Polygon) GeometryEngine.geometryFromWkt(wkt,
0, Geometry.Type.Polygon);
        this.adm0Code = adm0_code;
        this.adm0Name = adm0_name;
        this.adm1Code = adm1_code;
        this.adm1Name = adm1_name;
        this.status = status;
        this.dispArea = disp_area;
    }

    @Fact(name = "id")
    public String getObjectIDAsString() {
        return this.objectID;
    }

    @IndividualIdentifier
    @Ignore
    public String getID() {
        return String.format("%s-%s-%s-%s", this.gaulCode,
this.objectName.replace(' ', '_'), this.getStartDate().getYear(),
this.getEndDate().getYear());
    }

    @Fact(name = "adm2_name")
    public String getObjectName() {
        return objectName;
    }

    @Ignore
    public Polygon getShapePolygon() {
        return shapePolygon;
    }

    @Fact(name = "adm2_code")
    public long getGaulCode() {
        return gaulCode;
    }
    @Spatial(name = "wkt")
    public String getPolygonAsWKT() {
        return GeometryEngine.geometryToWkt(shapePolygon, 0);
    }

    @StartTemporal
    public LocalDate getStartDate() {
        return DateFieldUtils.readStartDate(this.validRange);
    }

    @EndTemporal
    public LocalDate getEndDate() {
        return DateFieldUtils.readExpirationDate(this.validRange);
    }

    @Fact(name = "adm1_code")
    public long getAdm1Code() {
        return adm1Code;
    }

    @Fact(name = "adm1_name")
    public String getAdm1Name() {
        return adm1Name;
    }

    @Fact(name = "status")
    public String getStatus() {
        return status;
    }

    @Fact(name = "disp_area")
    public boolean getDispArea() {
        return dispArea;
    }

    @Fact(name = "adm0_code")
    public long getAdm0Code() {
        return adm0Code;
    }

    @Fact(name = "adm0_name")
    public String getAdm0Name() {
        return adm0Name;
    }
}
```

Code 2: GAUL dataset definition in *Trestle*

This is the dataset definition used for the GAUL dataset utilized in Chapter 5. It consists of a single Java *class* which contains the required dataset component *annotations*. An *@IndividualIdentifier* which specifies how to uniquely identify this Trestle_Object. An *@DatasetClass* annotation which indicates which dataset this object is a member of, and a pair of *@StartTemporal* and *@EndTemporal* annotations which denote the *existence* interval of the object. Finally, the *@Spatial* and *@Fact* annotations specify the various data properties that are associated with this object.

Once the Trestle_Objects have been identified and their dataset definitions created, each object must next be assigned a temporal existence interval before it can be loaded into the datastore. This existence can be specified one of two ways. Manually, or automatically based on updated object facts. The automatic method will be covered in a later sub-section, but the manual method will be described here.

When loading an object into *Trestle*, the *data definition* requires specifying the temporal extant (either an interval or point) of the data being loaded. This is implicitly translated into both the valid temporal of the facts as well as the existence interval for the given object. While this presents a conflation between the two concepts of validity and existence, it is an artifact of the prototype implementation and will be addressed in the future. It also speaks to the difficulty in mapping being row-based and graph-based data layouts in that each new piece of information being added to an object may or may not have an impact on the temporal bounds of the object itself.

For each year of data added to *Trestle*, originally stored in a single shapefile, the following actions need to be taken:

1. Transform each record in the shapefile from the standard row-based layout into the graph database format.
2. Determine whether to add the record to a new object or merge it with an existing object and reconcile any data conflicts that may occur.
3. Optionally, compute spatial and temporal relationships for each new object and all others that might interact with it.

Each of these steps will be detailed in the remainder of this section.

4.2.3.1 Transform the data

Each record is transformed from the table layout to the graph layout, which models the data as a set of *nodes* in the graph database with *edges* between the nodes that the reasoner uses to both classify the nodes, based on the rules in the ontology, as well as to determine the relationship types between the various facts and objects. This is illustrated in Figure 17, which shows an example transformation of the GAUL record for *Cidade de Maputo, Mozambique* for the year 2013. Here, the object identifier and existence temporals are stored in the *object header* node, which is shown in blue, with the associated *Trestle_Facts* shown in yellow. The facts are linked to the object header via *has_fact* relationships.

Each graph node contains a number of *data property assertions*[35], the purpose of which is to denote the temporal scope of the given fact or object. The *object header* contains an *exists_temporal* assertion, while the *fact* nodes, contain both a *valid_temporal,* as well as a *database_temporal*, which provides the *bi-temporal* support for each fact. By default, the *database_temporal* is created as an open-ended interval that begins when the data is first loaded into Trestle and continues into the future; but this can be manually specified by the user.

[35] These data property assertions are different from the data properties described by the dataset being loaded (e.g. population count, name, etc.) and are an RDF term which describes information triples attributed to a given entity. These two types of properties will be disambiguated by always referring to RDF data properties as *assertions*.

In addition to the temporal assertions, each fact node also contains a data property assertion of the value of the data property being modeled. By using the name of the data property as the name of the data property assertion, this allows individual fact nodes to be addressed by their property name, without requiring the database or the management application to maintain a list of which fact nodes are associated with which data property. This also means that enforcing the information schema and datatypes is the domain of the management application, which handles returning the correct data as required by the *data definition* used in the query.

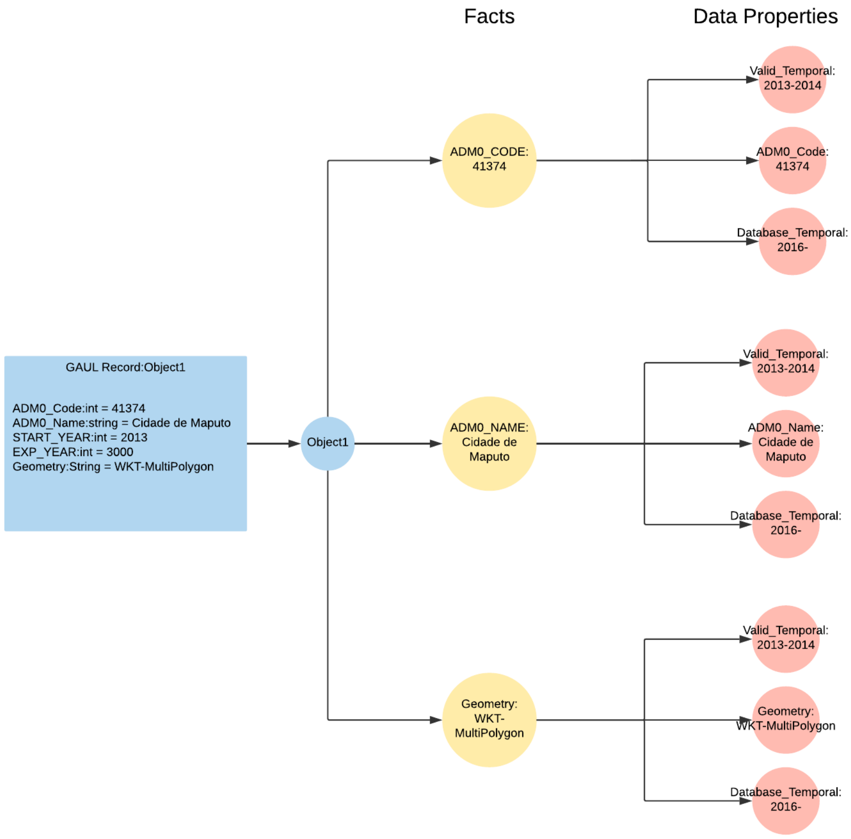

Facts
Data Properties
GAUL Record:Object1
ADM0_Code:int = 41374
ADM0_Name:string = Cidade de Maputo
START_YEAR:int = 2013
EXP_YEAR:int = 3000
Geometry:String = WKT-MultiPolygon
Object1
ADM0_CODE: 41374
ADM0_NAME: Cidade de Maputo
Geometry: WKT-MultiPolygon
Valid_Temporal: 2013-2014
ADM0_Code: 41374
Database_Temporal: 2016-
Valid_Temporal: 2013-2014
ADM0_Name: Cidade de Maputo
Database_Temporal: 2016-
Valid_Temporal: 2013-2014
Geometry: WKT-MultiPolygon
Database_Temporal: 2016-

Figure 17: GAUL record transformation

This figure illustrates transforming a single *record* in the GAUL dataset, corresponding to a single region for a single year, in the Trestle graph layout. The blue circle represents the *object header node*, which contains the identifier and the *exists temporal* information. The yellow circles are the associated *fact nodes*, with their data properties shown in the red circles.

4.2.3.2 Reconcile data conflicts

As each year of data is added to Trestle, the management application looks at the object identifiers and decides to either create new a Trestle_Object or merge the updated information into an existing object. Currently, Trestle supports three strategies for reconciling any conflicts that may occur between facts currently associated with a Trestle_Object and facts associated with the new data being loaded. It is important to note that these merge strategies do not change the values of the data being stored (e.g. they do not attempt to re-compute data properties such as *average population growth* based on new data) they only attempt to reconcile the temporal ranges of existing facts with the new ones being added.

Merging occurs on fact-by-fact basis, and for each fact Trestle asks two questions, 1) *does this new fact fall within the temporal existence of the given Trestle_Object?* 2) *is there an existing fact that is valid for at least a portion of the temporal period described by this new fact?*

Question 1: Does this new fact fall within the temporal existence of the given Trestle_Object?

In order to answer the first question, *Trestle* first considers the existence interval of the object and provides three methods for determining how and when to update facts.

1. *Ignore* –Trestle ignores the existence interval of the object and proceeds to merge the facts using the strategy described below. In this mode, it is possible for Trestle to result in a logically inconsistent state in which an object exists for a smaller temporal interval than the validity interval of is facts. For example, *Object A* might exist for the temporal interval [1990,2013), but have a *population count* fact valid for [1991,2015). The benefit of this mode is that it improves data loading performance in cases when object existence is either unnecessary to the domain use case, or when it is known that the object existence always exceeds the validity period of the object *facts* (e.g. when an object has an open-ended existence interval).
2. *During* – Trestle will only merge facts which occur entirely within the existence interval of the object. Attempting to load data which falls outside of this range will result in an error being raised.
3. *Extend* –Trestle will extend an object's existence interval if it encounters facts that fall outside this given range. For example, *Object A*, described earlier exists for the interval [1990,2013), when attempting to add a population count fact with the validity interval [1992,2015), *Trestle* would automatically extend the existence interval of *Object A* to encompass the new information. Thus, after loading the new fact, the existence of *Object A* would be [1990,2015).

Question 2: Is there an existing fact that is valid for at least a portion of the temporal period described by this new fact?

After considering object existence, if no errors are raised, Trestle proceeds to load the new facts using one of three merge strategies:

1. *NoMerge* – If Trestle sees that a newly added fact temporally overlaps with an existing fact an error is raised and the data is not loaded.
2. *ContinuingFacts* – When merging a fact, Trestle looks at the validity interval of the existing fact and only merges if the existing fact is *continuing*, meaning it does not have a specified ending temporal. If the existing fact is not *continuing*, an error is raised, and the data is not loaded.
3. *ExistingFacts* –Trestle will always merge new facts which overlap with existing facts.

It should be noted that <u>this process only occurs if Trestle determines that the user is attempting to add a new fact which temporally conflicts with an existing fact</u>. If no conflicts occur, the new information is simply added to the Trestle_Object without any updates to the temporal values of the previously loaded facts.

To illustrate the merging process, consider an existing GAUL object, *Cidade de Maputo*, which has been first loaded as part of the *2013* GAUL year[36]. This means, that the Trestle_Object has an existence interval of [2013, 2014) and the *ADM2_Name* fact has a value of *Cidade de Maputo* for the *continuing* validity interval of [2013,). In 2014, assume that the value of the *ADM2_Name* fact changed to *Cidade de Newputo.* In this case, Trestle performs the following actions:

1. Updates the Trestle_Object existence interval to encompass the new fact (e.g. *from* [2013, 2014) *to* [2013, 2015)).
2. Updates the *database* temporal of the existing fact to extend up to the temporal point at which the new data is being added (e.g. the current system timestamp).
3. Creates a new fact for the *ADM2_Name* property with the existing fact value (Cidade de Maputo), with a *continuing database* temporal starting at the temporal point at which the new data is being added (e.g. the current system timestamp). The *valid* temporal written to the new fact is a new interval created from the start point of the original fact temporal, but with a new ending point at the start temporal of the *valid* temporal of the new fact. (e.g. from [2013,) to [2013, 2014)).
4. Creates a new fact for the *ADM2_Name* property with the new fact value (Cidade de Newputo), with a *continuing database* temporal starting at the temporal point at which the new data is being added (e.g. the current system timestamp). The *valid* temporal that is written to the new fact is a new *continuing* interval with the correct start point of the new fact (e.g. [2014,)).

This process is repeated for each new fact being merged and is illustrated in Figure 18, which shows the fact relationships for the object before and after merging the new data. This process of *versioning* the fact nodes is the method by which Trestle achieves *bi-temporal* support and

[36] To simplify the example, we will only be showing the merging process for the *ADM2_Name* data property and assume that the existence interval of the Trestle_Object is determined based on the validity interval of the loaded facts. Likewise, we will assume that the validity interval of the fact is *continuing*, in order to identify the most complex merge case.

leverages the strengths of the underlying graph database to, at query time, filter nodes which fall outside the temporal range of the query. At no point is information removed from the database; the only modification that occurs to existing data is to update the *database* temporals of the original facts to indicate a more recent version now exists. It should also be noted that the *existence* interval of Trestle_Objects is not versioned; once updated, it is impossible to determine what the existence of the object was in previous database state.

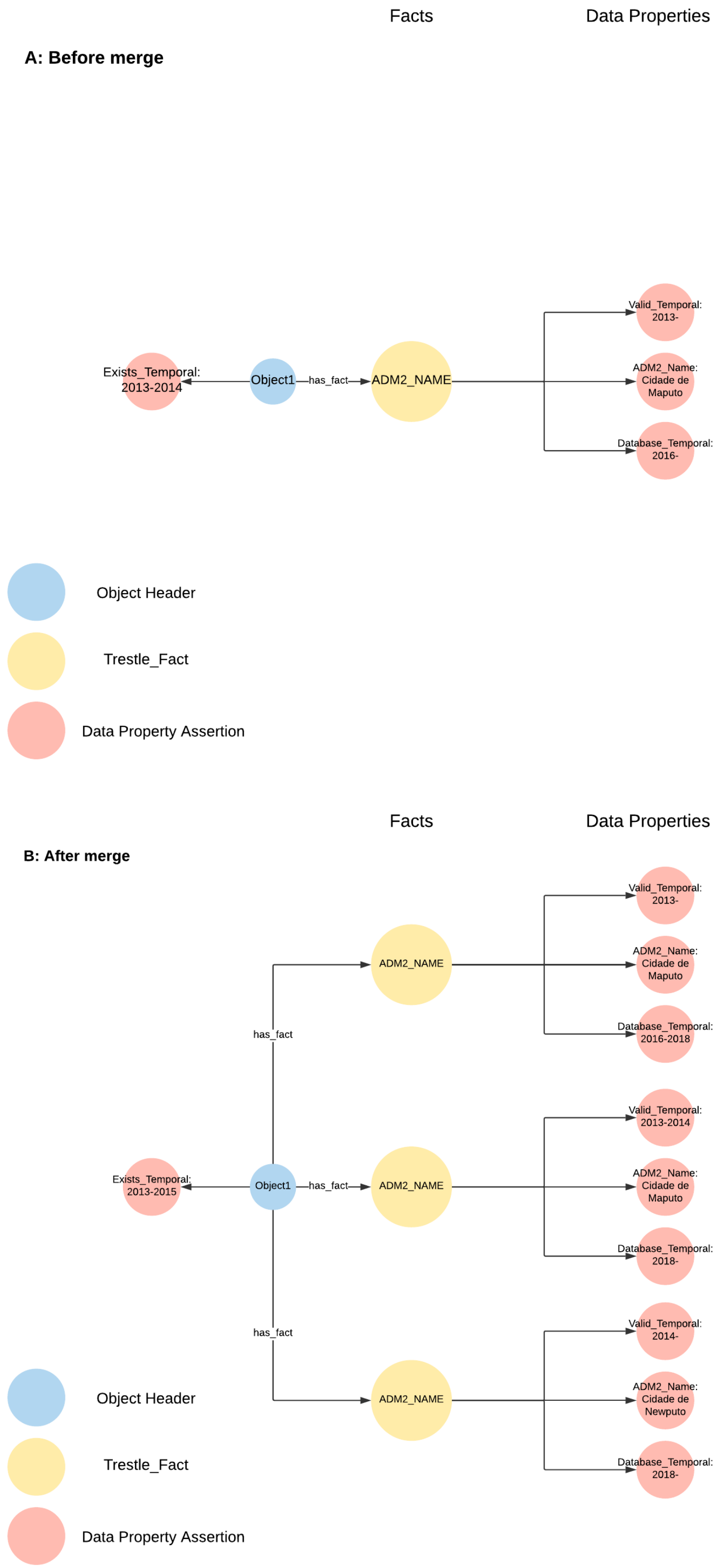


Figure 18: Data layout for GAUL object before (A) and after (B) a new record is merged.

4.2.3.3 Compute spatial and temporal relationships

The final, optional, step when loading data into Trestle is to compute spatial and temporal properties for the various objects. While Trestle provides robust support for arbitrary spatial and temporal queries, which can be dynamically retrieved at query time, it also provides a framework for pre-computing these relationships. These two approaches differ in several ways. The first, is that performing runtime queries is more flexible, in that it allows users to arbitrarily determine the relationships between various Trestle_Objects at the point in which they need the actual information. This is the same approach taken by traditional spatial databases but without the ability to leverage the additional logical properties of the relationships supported by the Trestle_Ontology, which is a unique feature of the Trestle system. Pre-computing the object relationships at load time, improves individual query performance, but at the cost of additional time required to load the datasets in the first place. The benefit of this later approach is that the pre-computed relationships can take advantage of the additional logical inference in the Trestle ontology. This is a feature optimized for data warehousing and *read-heavy* tasks, where data is loaded infrequently but queried and accessed often. These two approaches can be used in tandem to optimize for different patterns of data usage and storage.

Whereas previous steps in the data loading process were accomplished automatically by *Trestle*, this final step is done manually in order to avoid a performance penalty when ingesting large amounts of data. Instead of performing the computation at data load Trestle provides the basic primitives to compute these various relationships and provides a simple API for automatically generating a complete list of spatial and temporal relationships for any set of Trestle_Objects given as inputs, which can then be persisted in the database. The process for computing these relationships will be detailed further in Chapter 5.

4.3 FULFILLMENT OF DESIDERATA AND COMPARISON WITH PREVIOUS METHODS

Before concluding this chapter, we will discuss how Trestle fulfills the desiderata outlined in Section 2.3 and compares to the modeling approaches described in Chapter 3.

4.3.1 *Fulfillment of Desiderata*

D1: Enable Spatio-Temporal Object (ST-Object) modeling

Trestle provides robust support for building and managing complex ST-Objects by both enabling consistent identifiers for objects as they change over time, but also by explicitly supporting lifetimes of ST-Objects, regardless of the temporal interval of the underlying data properties.

D2: Support spatial and temporal relationships between ST-Objects

Trestle provides full support for reasoning about relationships between objects. In addition, it supports querying about relationships between various object states over time (such as which facts are valid for a given temporal range). An added benefit of using a reasoner, is that it allows additional logical information to be included in the relational algebra, which extends the ability of existing binary relationships to automatically encompass new information as it is added to the database.

D3: Annotate object events

Trestle supports object events including those explicitly defined by the ontology and arbitrary events that are specific to individual domain applications.

D4: Support multiple time states

Bi-temporal support is accomplished by maintaining both *valid* and *database* temporal intervals for individual facts and requiring queries to account for both temporal dimensions.

Design and Performance Requirements

In Chapter 3, we discussed the importance of a proposed data model being able to leverage existing programming and data storage paradigms so as to both reduce the difficulty of working with the data model, as well as being able to take advantage of ongoing engineering work and improvements. With that in mind, the focus of the Trestle development work has been on abstracting away the underlying ontology and triple-store so as not to introduce a completely different data paradigm that most users are unfamiliar with and to present data, to the user, in formats compatible with existing research environments. This is similar to the approach taken by *object-relational mapping* (ORM) tools such as Hibernate[37] for the Java language but optimized for interacting with ontologies and graph data. These tools allow the users to work with traditional programming language objects and concepts, and handling mapping to and from the underlying relational database. Translating between these two data layouts is non-trivial and has required a significant amount of development effort in order to achieve these goals.

The end result of this process is two primary methods for interacting with the management application that closely match existing programming and interaction paradigms. The first method of interaction is through the core API which allows users to integrate Trestle into their existing research environments and utilize the API to directly add and retrieve the data they need. These APIs are exposed both within the Java programming environment, as well as remotely as standard web accessible *Representational State Transfer* (REST) APIs. The REST implementation is what enables Trestle to interact with other non-Java programming environments as well as interoperate with other data analysis environments that can support data exported in traditional formats such as ESRI Shapefiles or *JavaScript Object Notation* (JSON) files.

The second method of interaction is through the prototype web application, which builds on top of the REST APIs to demonstrate the flexibility of the Trestle data retrieval environment. Figure 19 shows the *dataset viewer* prototype, which allows the user to select a geographic area of interest

[37] http://hibernate.org/orm/

and visualize the spatial change in the data over time. The data for a specific timepoint can then be exported in a number of different formats for user in existing spatial analysis tooling. This enables the use of Trestle as a spatio-temporal data warehouse which can support existing data analysis workflows where in users simply retrieve the temporally correct data from Trestle and then input that data into their existing tooling.

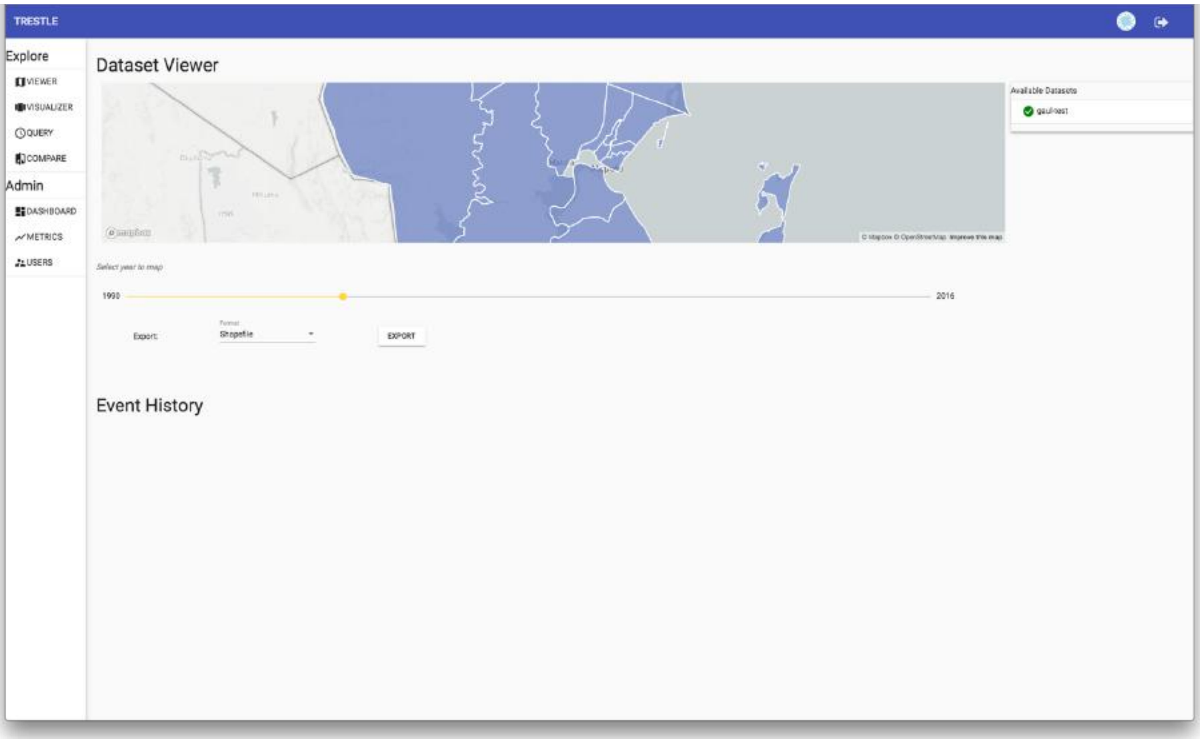


Figure 19: Dataset viewer from Trestle’s prototype web application.

This figure shows the main page of the prototype web application developed for this dissertation. While more details will be given in Appendix A, this gives the user the ability to select a specific dataset and perform simple temporal queries against it. Specifically, the ability to move between various time points and visualize the change in the spatial area. Once the desired spatial and temporal scope is selected, the results can be exported to a common spatial data format.

Once data has been persisted to the triple-sore, from the management application, we rely heavily on the built-in query engine and spatial functionality to support the more complex operations in Trestle. Overall, the functionality and performance of the triple-stores have been quite good, but some time has been spent attempting to address two major performance limitations.

The first limitation revolves around inference. Since the core of the application includes an ontology, each piece of information added to the application as to go through the logical inference engine (also known as the reasoner). Most triple-stores are optimized for query performance and because of this, each time a triple is added to the repository, the reasoner needs to execute and update the inference profile (the triples in the datastore that are logically inferred rather than directly asserted) for all potentially affected triples. This set of potentially affected triples is referred to as the *transitive closure* and grows progressively based on both the size of the database (the number of triples being stored) and the number of rules in the inference profile[38]. This can present a performance bottleneck for write heavy workloads, where most of the operations in the database involve adding new data rather than querying existing data. In addition, this inference step is additional work that is not required for traditional graph or relational databases and while it enables added functionality not found in other systems (such as transitive spatial relationships), it does present a performance limitation when directly comparing against existing systems. While there has been some progress in improving the performance of semantic reasoning, through methods such as parallel computation [160], [161], it remains an ongoing area of research.

The second major performance limitation revolves around data caching, which a key performance optimization for databases. Traditional caching and indexing methods, such as B-Trees [162] struggle to effectively manage ST-Objects due to their temporal nature. To illustrate, consider two user queries to retrieve data for *Cidade de Maputo* on August 1st, 2014 and again on August 12th, 2014. Since the GAUL dataset is only updated on an annual basis, those two queries should return identical objects, but how is the database to understand that no properties of *Cidade de Maputo* changed between those two temporal instants? In order to work around this limitation, and significantly improve query performance, we implemented a novel temporal indexing method known as a TD-Tree [163] which allows us to cache an object and specify the temporal bounds for which that object is valid. Using this method, the Trestle management application can correctly fulfill these two queries with only a single read from the database.

These types of performance optimizations and usability tweaks are critical to the operation of Trestle and have enabled the application to support a wide range of user queries and integrations, but at the cost of significant development work in order to reach this point.

4.3.2 *Comparison with Previous Methods*

When compared to the previous approaches outlined in Chapter 3, Trestle provides a number of distinct advantages and unique features not found in other data models. A summary of the comparison between Trestle and prior data models is given in Table 15, which includes the same information in Table 9, but with the addition of the Trestle system. While all of these features and details have been outlined previously in this chapter, this subsection will briefly summarize all the following major features of Trestle:

[38] By default, the Oracle database only updates the inference profile when manually instructed to. This improves data loading performance at the cost of not computing the inferred relationships until a later point in time. This means that the transitive property of the spatial *contains* relationships would not be compute (or updated) when new data is loaded. It should be mentioned that Oracle does support an optional mode of performing the inference when completing each database transaction.

1. *Direct access to fact values*:
   Because each fact is modeled as a node in a relational graph, each fact can be directly accessed by either the user or the application at any point in time. This allows Trestle to efficiently support *fact first* data queries. For example, queries such as *Find all the US counties which had a population count greater than 10,000 in 2013*, can be optimized by the underlying graph-database and be made extremely performant. This is difficult to achieve with traditional object-modeling approaches as all the data properties are stored within the object record which can inhibit performance as object complexity increases.

2. *Support for referencing data between datasets*:
   Given that the *has_fact* relationship is merely an edge between two nodes (one of which is an object header and the other is a fact node) this dramatically increases Trestle's ability to link across different datasets. There is no technical requirement that all of an object's facts be *owned* by a given Trestle_Objects (meaning that the facts are only linked to the given object in a single dataset). Instead, these objects can have *has_fact* relationships to facts associated with other dataset members. This means that a given data repository could have two datasets, TIGER and ACS, with the members of the TIGER dataset referencing the facts in the ACS dataset. Thus, whenever the ACS dataset is updated, the TIGER objects will immediately see the most up-to-date values, but with access to any previous versions. This not only simplifies data management and curation, but also increases performance and reduces data sizes as a single fact value is only stored in the database once but referenced multiple times by the objects which need it. The ability to link between datasets is not feature directly supported by existing modeling approaches.

3. *Manages all datatypes through the same mechanisms:*
   One challenge with previous approaches is that they make a technical distinction between spatial and no-spatial data properties and manage these two categories through entirely different paradigms (e.g. the spatial graph in the *3-Domain* model). This presents a challenge as data scale increases, especially due to the fact that a large amount of available information is intrinsically non-spatial, such as population counts or census data, and is instead merely linked to a spatially referenced entity. By contrast, Trestle maintains no distinction between spatial and non-spatial data, instead relying on the underlying datastore to optimize and manage the various data properties. This not only simplifies implementation and interaction, but also allows Trestle to take advantage of performance and query improvements as the datastores are further developed.

4. *Maintains data context:*
   Through the use of datatype URIs, Trestle maintains a distinction between the concrete type of a data property (e.g. string, floating point number, etc.) and its user-defined type (such as differentiating between population count sampling methods). While Trestle remains agnostic as to the user-defined data type, this additional information can provide the necessary context to the user in order to correctly interpret the data value. This data context is a unique feature of Trestle, derived from the underlying semantic web technologies, and is not found in other modeling approaches.

5. *Supports pre-computed and logical extensions to spatial and temporal relationships*:

Through the use of the reasoner supported by the ontology and the underlying graph-database, Trestle supports additional logical extensions to the common spatial and temporal relationships supported by other modeling approaches. These extensions (such as symmetry and transitivity) may not be immediately useful for smaller projects, but for larger data warehouses, this can provide a powerful foundation for deeper linking between datasets in that it allows relationships, expressed within a single dataset, to be logically extended to encompass previously disparate datasets as well.

6. *Extensible to support additional event and data types:*
   While Trestle defines a small subset of datatypes and supported events, applications can easily extend the Trestle ontology to add additional concepts necessary to their specific use case. As an example, the SDIDS project can create its own ontology which includes application specific concepts (such as biological indicators or health facility type) and import the Trestle ontology to take advantage of the underlying spatial features.

7. *Supports custom collections of related objects:*
   Trestle provides the ability to create groupings of related objects into custom collections. This further illustrates Trestle's ability to not only manage objects, but datasets as well. Collections can feature objects from multiple datasets and thus serve to improve data management and curation by grouping together the data necessary for a given research study. Previous modeling approaches are limited to managing ST-Objects and are not designed to support higher-level aggregations and associations.

8. *Provides a platform for data access and curation:*
   Through the use of open APIs and standard web technologies, the Trestle application provides a data platform that can enable researchers to access the required information within their existing research environments. Rather than requiring users to adapt to a custom workflow, Trestle interoperates with existing applications and handles the translation between the underlying data model and the common spatial data formats in use by researchers.

Approach	Object Modeling	Object Relationships	Events	Time States	Development
Time slicing	Not directly, only if implemented by the database administrator	Full support for spatial relationships, limited support for temporal relationships	No event support	Supports bi-temporal time	Simple to implement, inefficient with large datasets
ST-Composite	No support for ST-Objects	Limited support for spatial relationships, no support for temporal relationship	No event support	No direct support for database time	Requires custom data backend and relies on unique indexing techniques
ST-Object	Full support for ST-Objects	Full support for both spatial and temporal relationships	No event support	Supports bi-temporal time	Requires custom data backend and relies on unique indexing techniques
3-domain	Full support for ST-Objects	Full support for both spatial and temporal relationships	Some support for events	No direct support for database time	Builds on existing relational database technologies, along with custom spatial support
Trestle	Full support for ST-Objects	Full support for both spatial and temporal relationships, including logical extensions	Supports built-in and user defined events	Supports bi-temporal time	Leverages semantic web and graph-database technologies for custom data model

Table 15: Updated summary of data modal desiderata fulfillment, including Trestle.

This is an update to the data originally presented in Table 9, but with the addition of the Trestle system. Here we can see that Trestle fulfills of the desiderata outlined in Section 2.3.

Query	Time slicing	ST-Composite	ST-Object	3-Domain	Trestle	Total
Q1	✓	✗	✓	✓	✓	4
Q2	✓	✓	✓	✓	✓	5
Q3	✓	✓	✓	✓	✓	5
Q4	✗	✗	✓	✗	✓	2
Q5	✗	✗	✓	✗	✓	2
Q6	✓	✗	✓	✓	✓	4
Q7	✓	✗	✓	✓	✓	4
Q8	✗	✗	✓	✓	✓	4
Q9	✓	✓	✓	✓	✓	5
Q10	✗	✗	✗	✗	✓	1
Total	6	3	9	7	10	

Table 16: Updated summary of data model example query fulfillment, including Trestle.

This is an update to the data originally presented in Table 10, but with the addition of the Trestle system. Here we can see that Trestle supports all of the example queries described in Section 3.2.1.3.

4.4 Conclusion

In conclusion, the data model and management application encompassed in the Trestle project provides a robust foundation for managing complex spatio-temporal data and integrating disparate datasets into a unified repository. The following chapters will serve to evaluate the utility and correctness of Trestle by applying the application towards addressing a series of common challenges facing public health researchers.

Chapter 5. EVALUTION 1: DATASET INTEGRATION

5.1 INTRODUCTION

The previous chapter introduced the *Trestle* data model and management application, which provides a foundation for integrating and reasoning about time-varying spatial datasets. This chapter begins the evaluation process of the Trestle system.

As previously discussed, a significant innovation of Trestle is in extending the relational model from solely focusing on relationships between objects (such as contains, before, etc.) to also including relationships within objects (e.g. has_fact). By utilizing the same primitives and data layouts for each relationship type, Trestle is able to scale to support not only complex object relationships, but to also effectively manage the internal data state of ST-Objects even as the amount of information within in each object increases. This means that evaluating the Trestle project involves two distinct phases:

1. Determining Trestle's ability to create and query spatial and temporal relationships between ST-Objects.
2. Evaluating its ability to effectively represent internal ST-Object states as relationships of an ST-Object to its associated facts.

This chapter explores the first phase of the evaluation process, while the second phase will be the focus of Chapter 6. In order to evaluate Trestle's object-relationship abilities, we will attempt to utilize Trestle to address a common challenge in longitudinal public health research, namely the difficulty in maintaining a consistent view of the spatial state of the world for the duration of the study period. By implementing an algorithm, on top of Trestle, to automatically detect split/merge events within the GAUL dataset, we will leverage the flexibility of both Trestle and the underlying graph data layout and clearly demonstrate its object-relationship capabilities.

Section 5.2 describes the public health problem that will be used as the motivating example for the evaluation of the Trestle system. Section 5.3 details the split/merge algorithm design and implementation, built on top of Trestle, which aims to address the motivating described in Section 5.2 and produce a spatially and temporally integrated dataset that allows users to determine the sequence of events which occurred before or after the lifetime of a Trestle_Object. Section 5.4 analyzes the algorithm results to determine whether or not the split/merge paradigm is an appropriate method for describing spatial change in administrative boundaries. Section 5.5 details some limitations of the split/merge algorithm, and some additional algorithm designs are outlined in Section 5.6. Finally, Section 5.7 evaluates Trestle's ability to effectively model spatio-temporal relationships between Trestle_Objects as seen through its ability to implement the split/merge algorithm in order to address the motivating public health problem.

5.2 TRACKING CHANGES IN ADMINISTRATIVE BOUNDARIES

To begin, we will describe a common challenge in public health research and the difficulty in addressing this issue with existing spatial tooling. As outlined in Chapter 2, public health researchers often find themselves initiating long-term longitudinal research projects in order to develop an effective understanding of complex health phenomena. But as the scope of these studies increase, both in terms of geographic scale and temporal length, researchers are forced to reckon with the fact that the spatial environment at the beginning of the study may be drastically different from the spatial state of the study environment at the end of the research period. As an example, between the 1971 and 1981 Censuses in the United Kingdom, only 44% of administrative boundaries remained the same. That number then shrunk to 32% between the 1981 and 1991 censuses [164]. While much work has been done to develop methods for accounting for this type of change [165], researchers are still left with the need to understand the degree of change in the spatial environment and how those changes relate to the new state of the map. Answers to the questions of *what came before, what came after, and how things are related*, are something that traditional GIS applications struggle to account for, due to their lack of support for ST-Objects, which leaves researchers manually searching for the before/after state of a given region. But answers to these types of questions are of critical importance in ensuring the accuracy of the studies being performed.

As a concrete example of this challenge, consider two examples from the GAUL dataset at the county (ADM2) level. Figure 20 shows a view of the capital city of Mozambique, *Cidade de Maputo*, as it looked in 2014. Figure 21 goes back in time to 1990 and reveals that between these two map states, something happens which changes the spatial conformation of Cidade de Maputo. To a researcher attempting to analyze changes in public health indicators, realizing that their current view of the world does not hold true throughout the study duration immediately begs the question. *What came before Cidade de Maputo?* And further, *what is the relationship between what is now Cidade de Maputo and what was there previously?*

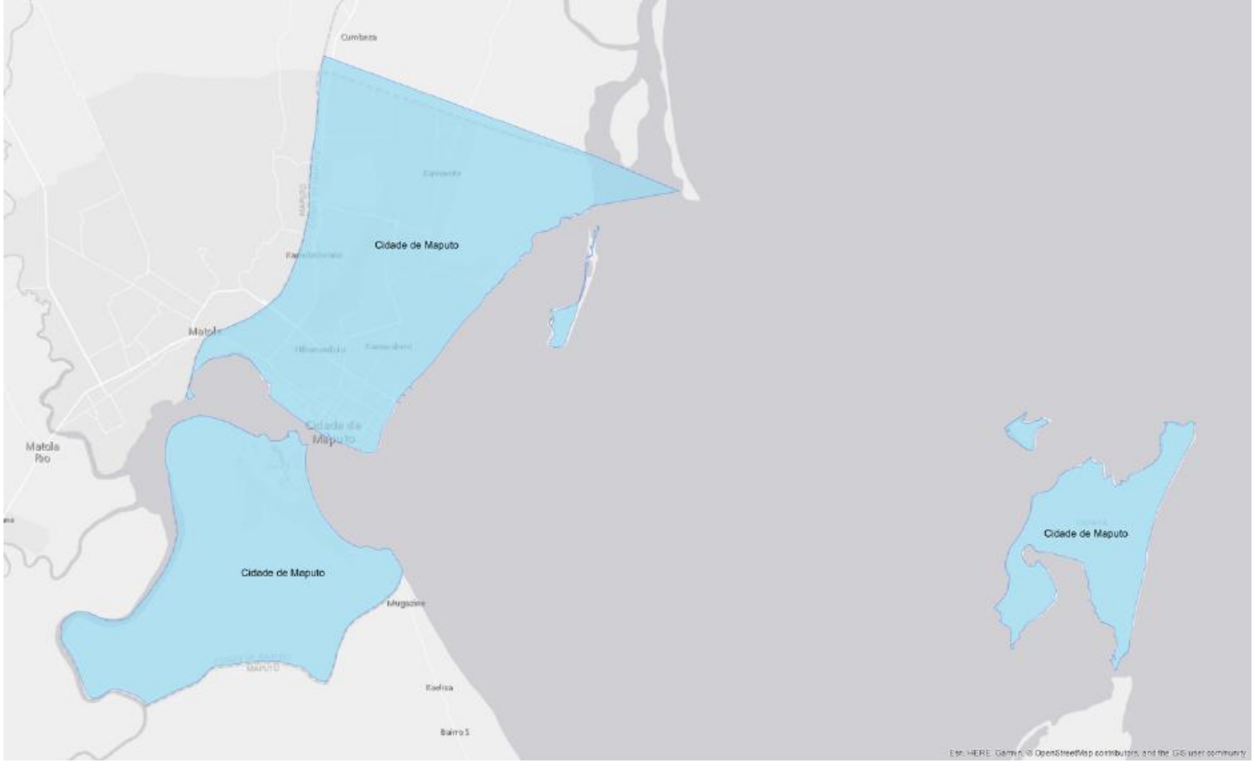


Figure 20: The city of Cidade de Maputo, in 2014.

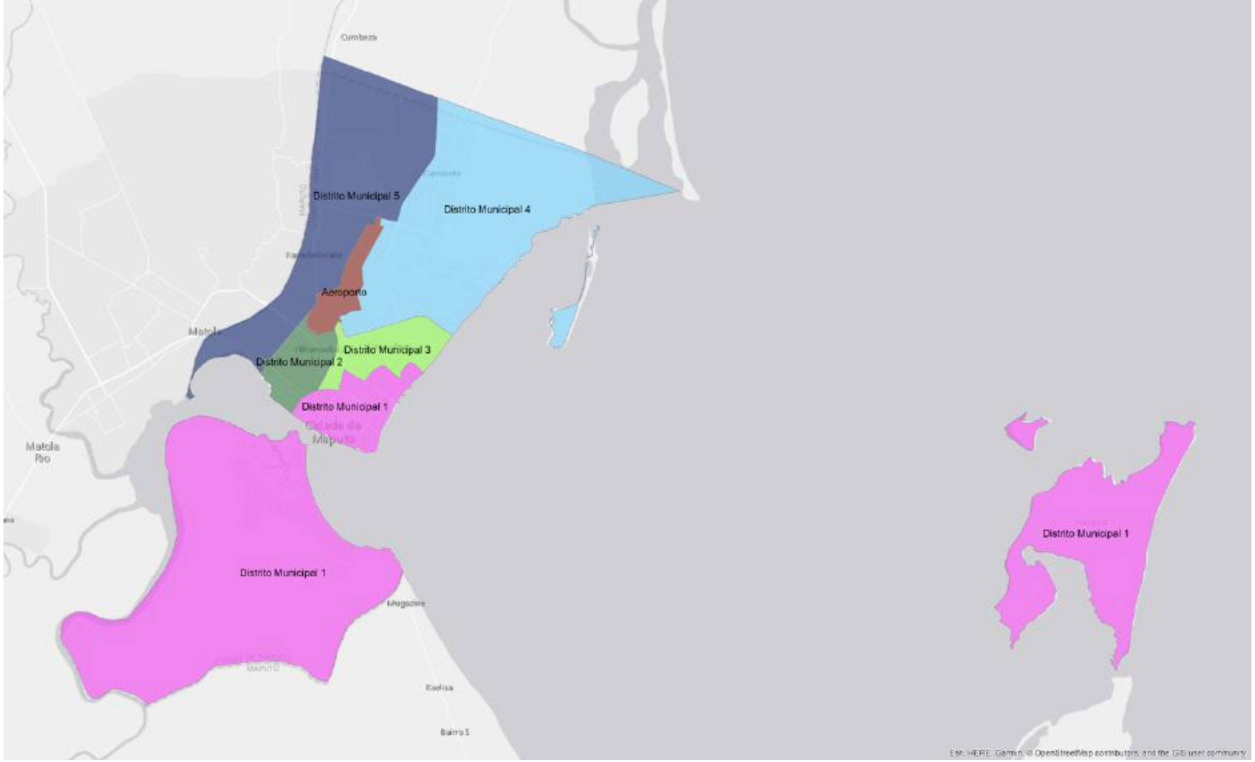
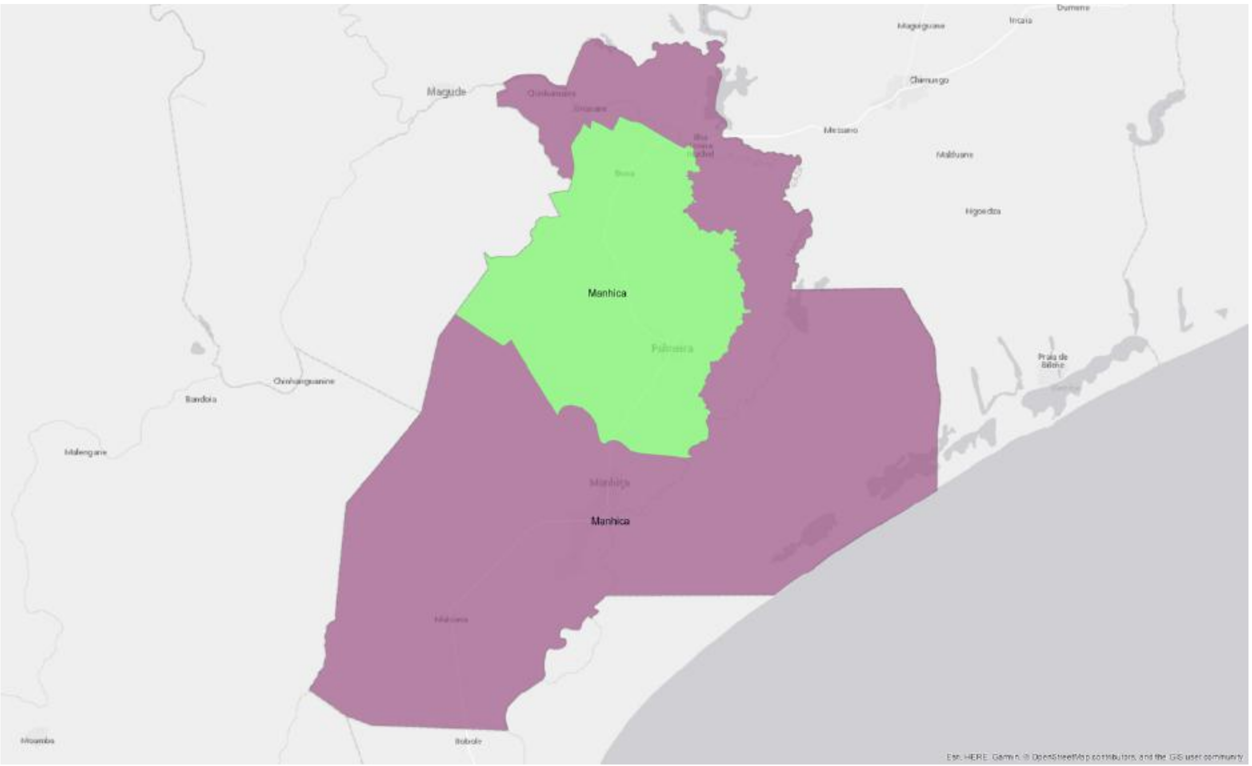


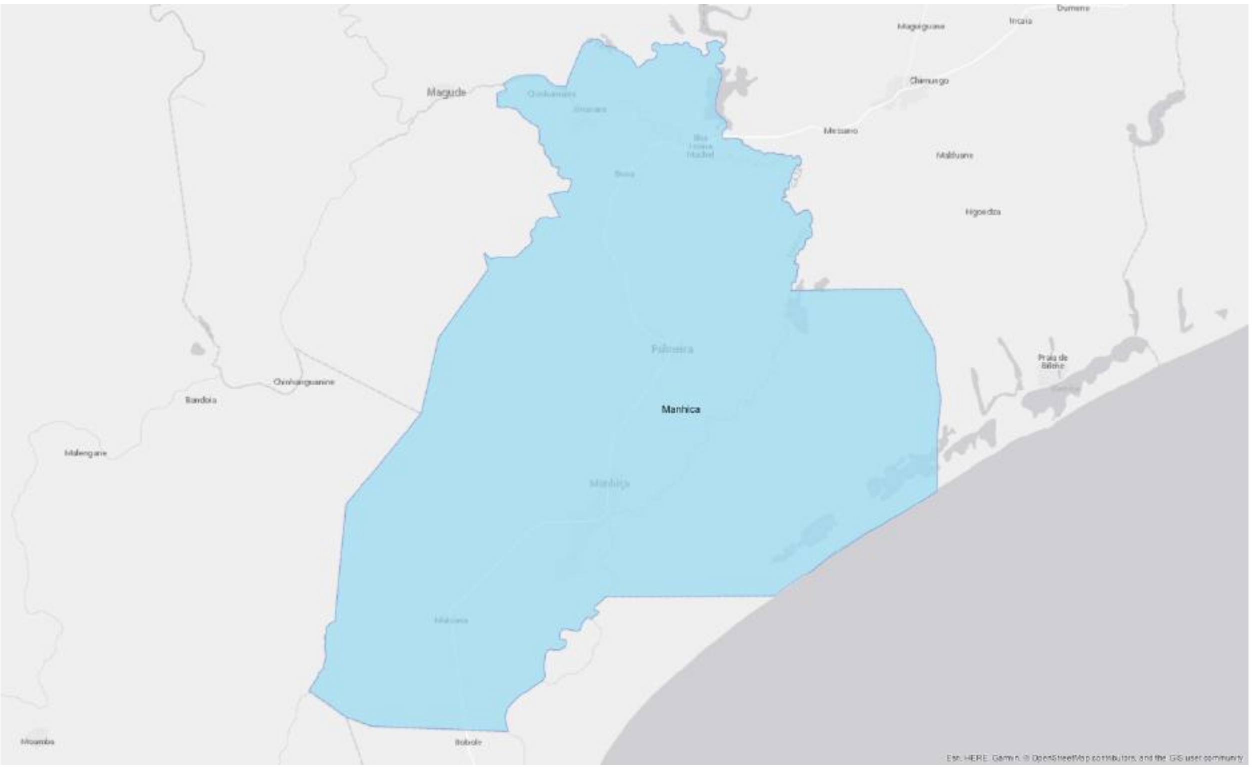
Figure 21: The same spatial area in 1990.

Another concrete example is given in Figure 22 and Figure 23, which show two views of *Manhica* (also in Mozambique) at the same time points. Two things differentiate these maps from the previous example. The first, is that instead of multiple counties apparently merging into a new entity, we have the reverse, a single county is apparently split into two new counties; however, with one additional complication. Both the initial county and the two resulting counties share the same name, *Manhica*. In order to disambiguate these entities, the GAUL dataset assigns each region a unique identifier which is never duplicated and only valid for the lifetime of the given region. However, data collected in the field is often coded not with a stable and unique identifier, but with the name of the region it is collected in, at the time it was originally recorded. Thus, begging the question, *is Manhica, as it is known in 1990, the same Manhica that exists in 2014?* The answer to this is an emphatic no, but <u>without a system like Trestle, determining this requires a manual inspection of the administrative boundaries</u>.



Esri, HERE, Garmin, © OpenStreetMap contributors, and the GIS user community

Figure 22: Two *Manhica* counties in Mozambique, in 2014.



Esri, HERE, Garmin, © OpenStreetMap contributors, and the GIS user community

Figure 23: The same spatial area in 1990.

At this point, a new question arises: *But what about counties which do not undergo a clean split or merge?* For example, counties that undergo changes such as observed in Figure 24 and Figure 25, where area from a single region in one temporal state is distributed to two or more regions in the other state. Here, we can see that the same spatial area is covered by three counties in 2014, but two different counties in 1990. This uneven distribution of area means that spatial analysis of the two temporal states is not as straightforward as in the previous examples and that researchers must manually determine how to reconcile the differences. However, knowing this requirement allows us to account for this scenario in developing and evaluating the split/merge algorithm.

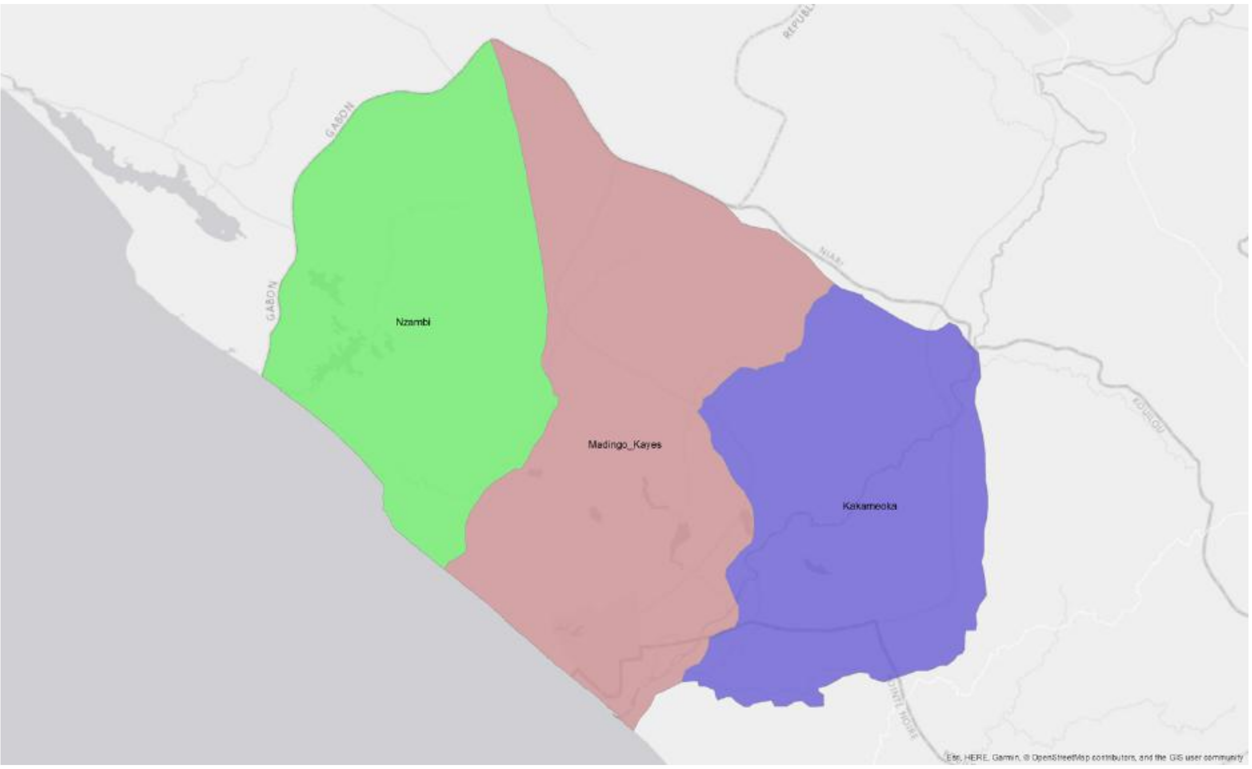

GABON
GABON
REPUBLI
NIAKI
KOUILOU
Nzambi
Madingo_Kayes
Kakamoeka
Esri, HERE, Garmin, © OpenStreetMap contributors, and the GIS user community

Figure 24: Three counties in Congo, 2014.

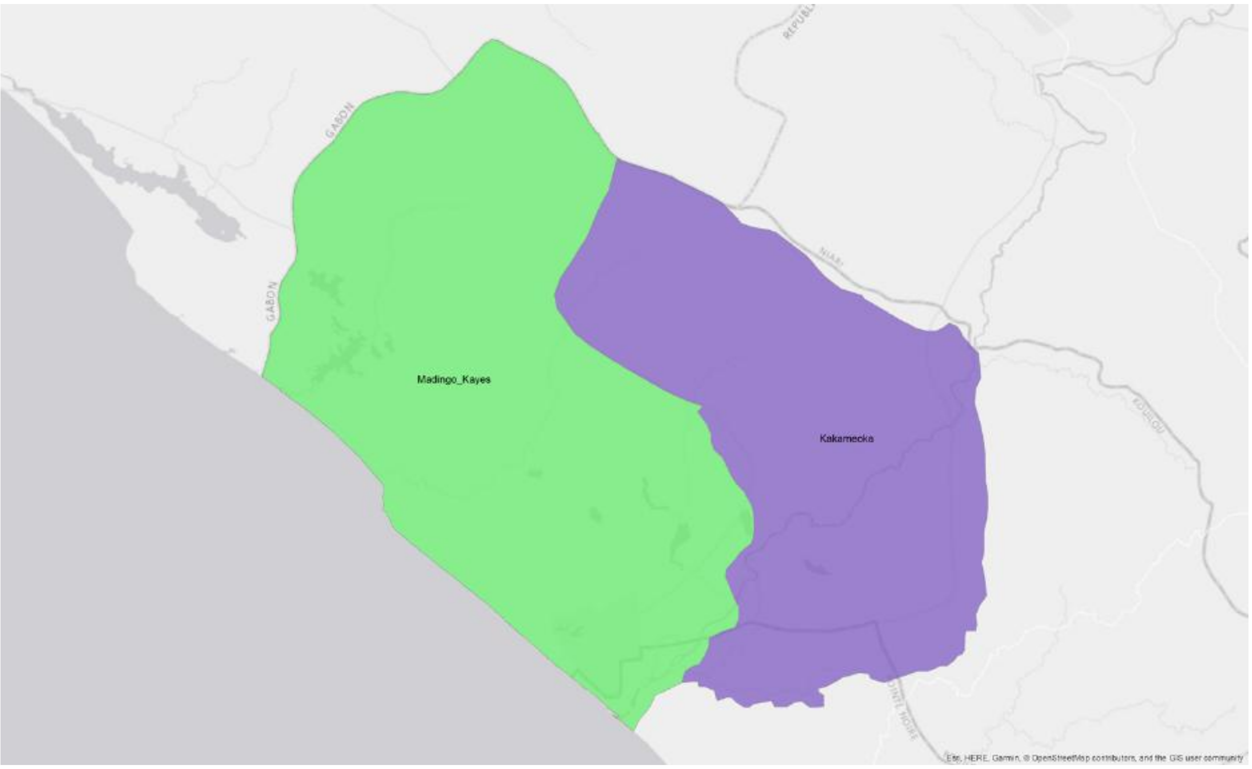

GABON
GABON
REPUBLI
NIAKI
KOUILOU
Madingo_Kayes
Kakamoeka
Esri, HERE, Garmin, © OpenStreetMap contributors, and the GIS user community

Figure 25: The same spatial area in 1990.

To address these challenges, we introduce the idea of a spatial *split/merge* event as a foundational method for describing spatial change in administrative districts and for tracking those changes over time. A *split* occurs when a spatial region is divided into two or more regions, which cover exactly the same spatial area, no more, no less (such as is the case with *Manhica* in Figure 22 and Figure 23). A *merge* is the inverse, where two or more spatial regions are unified into a single region, which covers exactly the spatial area, no more, no less (such as is the case with *Maputo* in Figure 20 and Figure 21). The final possibility (as in the case with the counties in the Congo in Figure 24 and Figure 25) is that a spatial reorganization may include a different spatial area than the before/after state. In this example, the spatial area covered by either of the counties in 1990 is not directly equivalent to any combination of the resulting 3 counties in 2014.

This illustrates a potential limitation of our algorithm design in that it only attempts to identify spatially equivalent split/merge events, not more complex changes that affect a portion of existing counties. The reason for this limitation is twofold. First, it allows a determination of the amount of global spatial change than can effectively be described through the split/merge approach. Second, it provides researchers with the assurance that the before and after state of a given spatial region accounts for identical spatial area, which simplifies any data indicator normalization that may need to be performed at a later date. Finally, the resulting output of the algorithm also includes collections of Trestle_Objects that have some spatial and temporal relationship, even if it falls short of direct spatial equivalency. This allows researchers to determine on a case-by-case basis how to handle more complex spatial changes, such as those described in Figure 24 and Figure 25. This is very similar to the current approach taken by researchers when working with historical datasets.

To summarize, this public health challenge provides an ideal problem space for performing the first part of the Trestle evaluation process. Specifically, this approach provides three major outcomes:

1. *Evaluate the ability of Trestle to effectively represent relationships between spatio-temporal objects.*
   One strength of Trestle is that it provides a number of primitives for implementing spatio-temporal algorithms that build and operate on ST-Objects. This prototype algorithm utilizes these primitives (such as collections, relationships, etc.) to answer the underlying research question as well as demonstrate the usability of the system for other types of algorithms that might attempt to achieve the same result. In addition, it provides a demonstration of the object-relationship capabilities of the Trestle system.
2. *Determine the percentage of boundary reorganizations in the GAUL dataset that can be directly associated with a split/merge event.*
   From a research perspective, it is desirable for each spatial object to *cleanly* split or merge into one or more spatial objects, with no overall change in the area being described. Of course, this is not always possible, but it remains to be seen how much spatial change can be clearly described by the split/merge approach. One goal of this implementation is to determine the feasibility of delineating splits/merges in historical datasets and quantify the amount of change that can be directly traced back to the initial spatial state.
3. *Produce a spatially and temporally integrated dataset, suitable for supporting longitudinal public health research.*

> The final output of this algorithm is a spatially and temporally integrated dataset that represents all the information available in the GAUL dataset, for all the years in which information is available, but unified into Trestle_Objects and linked together through time. We believe this dataset to be useful to longitudinal public health researchers, across the globe.

The next section will describe the design and implementation of the split/merge algorithm, which will be followed by an evaluation of its utility when executed against the GAUL dataset.

5.3 Split/Merge Algorithm Design and Implementation

5.3.1 *Algorithm Design*

Our algorithm design takes a spatially exhaustive approach, which attempts to answer the question, *for the time point immediately preceding (or following) the lifetime of a given spatial object, can I account for all the spatial area described by the object?* This approach is common not only in the geographic space, but also in other fields such as medical image segmentation [166] where multiple spatial and temporal slices of data are integrated together to build a unified image of a medical process. We build upon these existing techniques to both develop the algorithm, as well as perform an evaluation of its utility. The benefit of this approach is that it is simple to implement and easy to clearly evaluate success and failure cases. The overall goal is accomplished by progressively building up collections of spatial objects that overlap each other, at some point in time, and determining if any combination of objects, at a different time point, form a *spatial union*. A *spatial union* is defined as an equivalency between an initial object and a spatial aggregation of a given set of additional objects. To illustrate this, consider the above example of *Maputo* (Figure 20 and Figure 21). A visual inspection[39] of the two images shows that *Cidade de Maputo* is equivalent to the spatial union of *Distrito Municipal 1, Distrito Municipal 2, Distrito Municipal 3, Distrito Municipal 4, Distrito Municipal 5,* and *Aeroporto.* Likewise, we can see that the *Manhica* in 1990, is spatially equivalent to the two *Manhica* regions in 2014 (Figure 22 and Figure 23). For Figure 24 and Figure 25 there is no spatial equivalency because no individual region from 1990 can be fully accounted for by any combination of regions from 2014.

One challenge with this approach, is that there might be some percentage of area that is not accounted for between the two spatial states, either due to rounding affects in the data processing or small measurement errors in the GAUL dataset. These errors primarily arise due to the conversions between spatial data formats. The GAUL dataset is distributed as a set of ESRI shapefiles, but the Trestle data store requires converting the data in the WKT format. When performing the spatial computations, the data is converted in the Java JTS format[40]. These conversions are common in other spatial databases and not unique to the Trestle system. This issue is further compounded by the fact that spatial coordinates can often have a very large *mantissa* (the value of a floating-point number after the decimal point). The Java language makes use of the IEEE-754 numerical representation, which can result in some ambiguity for operations with very precise numbers [167].

To account for this, we implemented a *cutoff* value which defines the percent spatial similarity between both sides of the spatial union. We set this value to be 95% in our experiments, but this

[39] This visual inspection is also confirmed numerically.

[40] https://locationtech.github.io/jts/.

value is configurable both at data loading time, as well as query time. This means that a dataset can be generated using a low similarity value (and thus potentially increase the number of false positives) but queried to return only values with a high degree of spatial similarity, which may be more approach for the specific task at hand.

These unions are progressively built over time as each new region is evaluated by the algorithm. In order to accomplish this, we need some mechanism for storing some state associated with each region: namely, which regions might potentially contribute to a spatial union of the given region. This state management is accomplished, in Trestle, through the use of the *Trestle_Collections* feature. These collections allow us to create links between regions that have spatial relationship between one another, in this case a *spatial overlap*. As each new Trestle_Object is evaluated by the algorithm, it is associated with any collections in which a member spatially overlaps with the new object. Each of these collections are then evaluated to determine if any combination of their members (and the new object) form a spatial union. Since these collections are independent of each other, a given Trestle_Object might be associated with more than one collection. The details of the actual order of operations for the algorithm is given in Section 5.3.3.

5.3.2 *GAUL Data selection*

While the GAUL dataset features global coverage, for this experiment we limited the input space to 7 African countries, which have experienced varying levels of change in administrative boundaries over the past several years, largely due to internal political pressures [168]. In addition, we included the countries of Mozambique and Uganda, which were a focus of the *SDIDS* project that originally funded this work.

Table 17 gives an overview of the amount of spatial change experienced by the various countries over time. This was generated after loading the GAUL data into Trestle and combining the various years of data into unified Trestle_Objects. The amount of variation ranged from 0% (Benin and Uganda had no spatial change), to a high of nearly 71% of objects in Nigeria which were modified during the past 24 years. Figure 26, Figure 27, Figure 28 and Figure 29 show the spatial state of Congo and Nigeria at the earliest (1990) and latest (2014) years of the GAUL dataset.

Country	Number of Objects	Unchanged Objects	Changed Objects
Benin	77	77	0
Central African Republic	72	69	3
Congo	119	18	101
Democratic Republic of the Congo	52	46	6
Mozambique	152	142	10
Nigeria	1017	296	721
South Sudan	46	46	0
Sudan	87	84	3
Uganda	170	170	0

Table 17: Distribution of reorganized regions for each evaluated country.

This table shows the amount of county-level *reorganization* experienced by each of the evaluated countries from 1990-2014. The term *object* refers to the Trestle_Object which contains all the records for that given region. This shows that some countries underwent little to no reorganization, whereas *Congo* and *Nigeria* underwent a significant amount of spatial change.

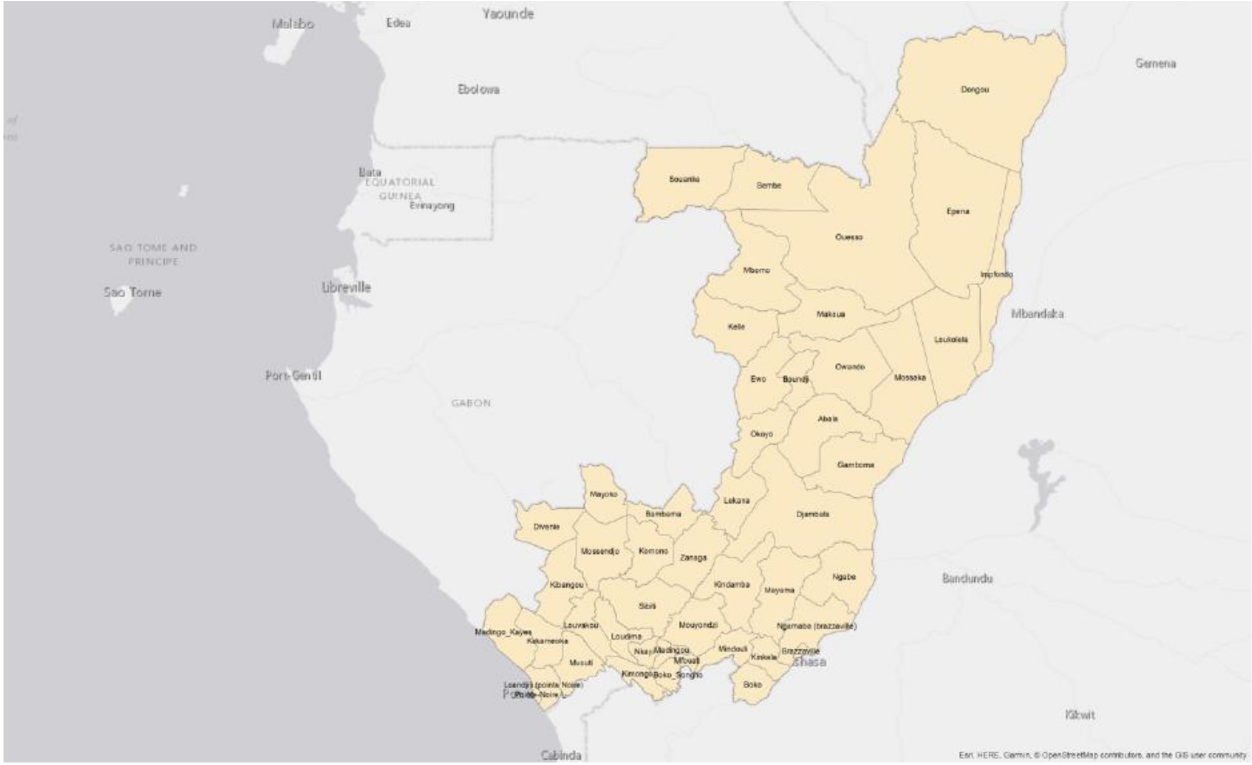


Figure 26: County-level district organization for Congo, in 1990.

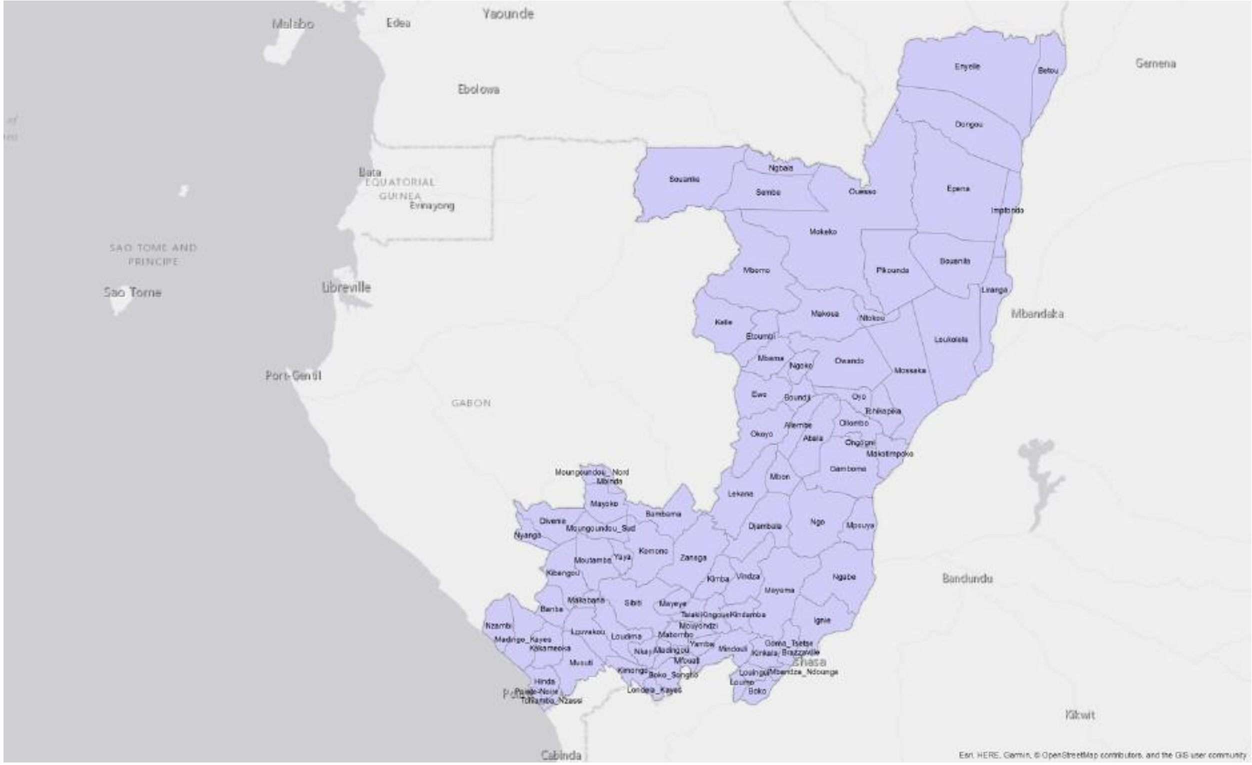


Figure 27: County-level district organization for Congo, in 2014.

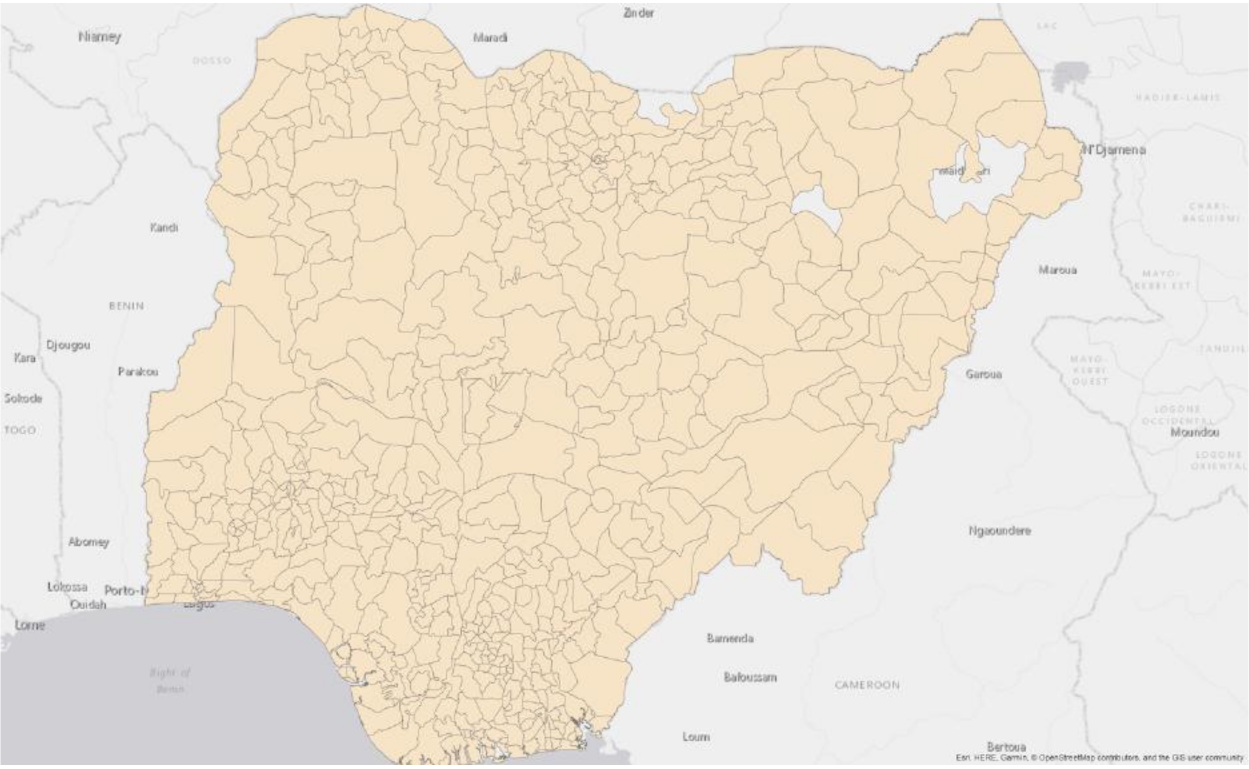


Figure 28: County-level district organization for Nigeria, in 1990

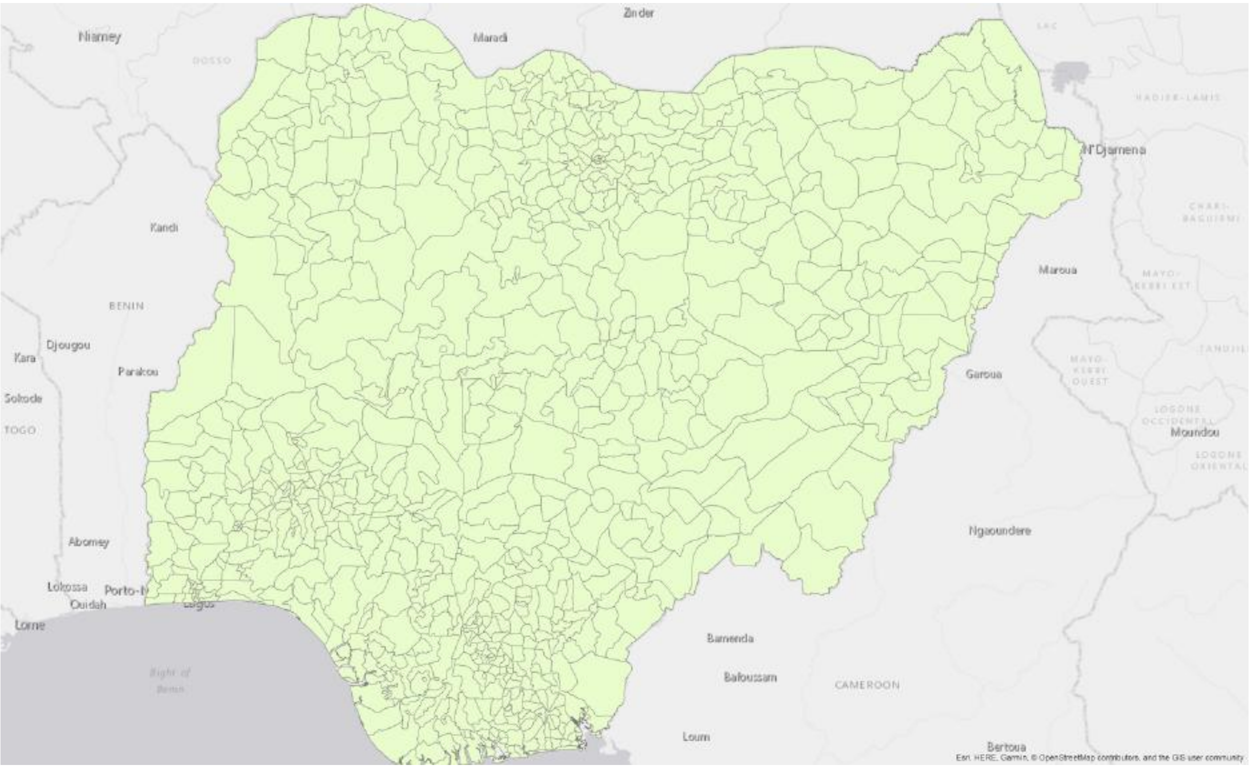


Figure 29: County-level district organization for Nigeria, in 2014

Given that the algorithm attempts to match objects based on their area, the spatial dynamics of the dataset are important and may have an impact on the overall utility of the algorithm. For example, if the regions are very large, a 95% similarity match may leave a significant amount of area unaccounted for. Conversely, small, tightly packed regions may present a challenge to the algorithm as there might be multiple combinations of regions that could contribute area to a larger aggregation. Figure 30 shows the logarithmic distribution of region area. Here we can see that the regions are clustered towards the smaller end of the distribution with 90% of the regions being less than 8,000 km^2 in size. Likewise, Figure 31 shows a log-log plot of the size of the region versus the number of points in the exterior boundary. It shows that there is no direct correlation between the size of the object and the complexity of its boundary; however, there are two distinct clusters within the dataset which warrant future exploration as to their internal dynamics and relationships.

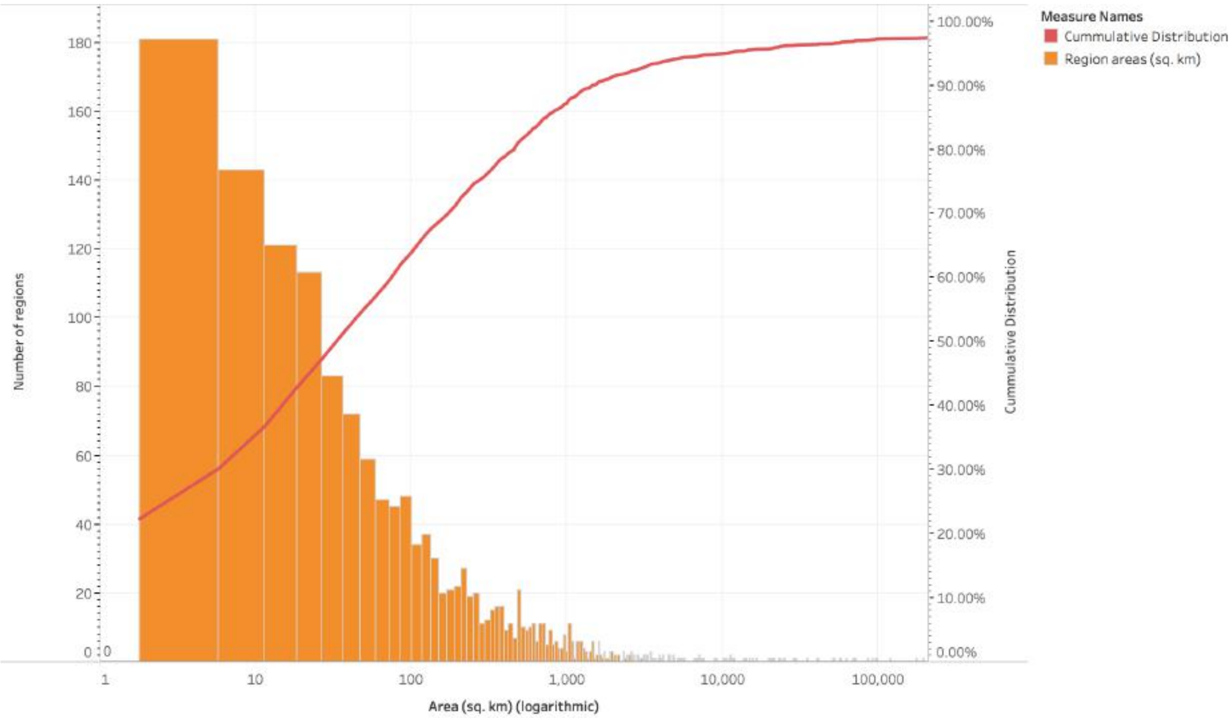


Figure 30: Distribution of region areas.

This figure presents a histogram of the spatial area (in km^2) encompassed by each region (totals shown in the left Y-axis). The red line is the cumulative probability distribution (total shown on the right Y-axis) which illustrates that the majority of the region areas are cluster towards the smaller end of the range. It should be noted that there are some very *long-tail* outliers which are several orders of magnitude larger than the smallest regions.

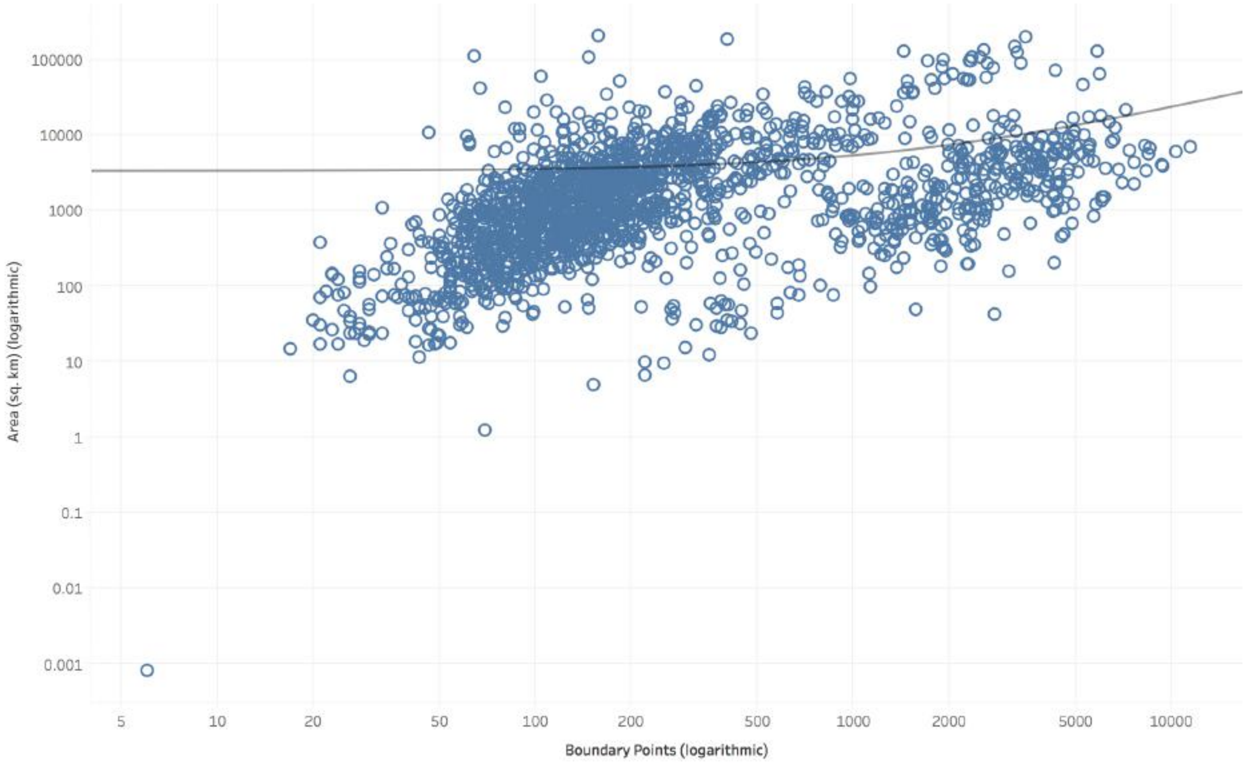


Figure 31: Region size vs. number of boundary points.

This figure illustrates the relationship between region size and complexity of the exterior boundary. Each region is plotted in logarithmic space with the total number of geographic *points* contained in the boundary on the X-axis, and the spatial area (in km^2) on the Y-axis. The red line shows that there is no correlation between the complexity of the boundary and its spatial area.

Figure 32 gives some additional context as to the temporal dynamics of the initial dataset. Figure 32(a) shows the number of regions existing for each year, which indicates major redistricting occurs in 1999 and 2011. Figure 32(b) shows the distribution of the number of years for which the regions exist, with clusters at 3 years, 9 years, 15 years, and 24 years (the latter indicating the region did not change during the course of the study).

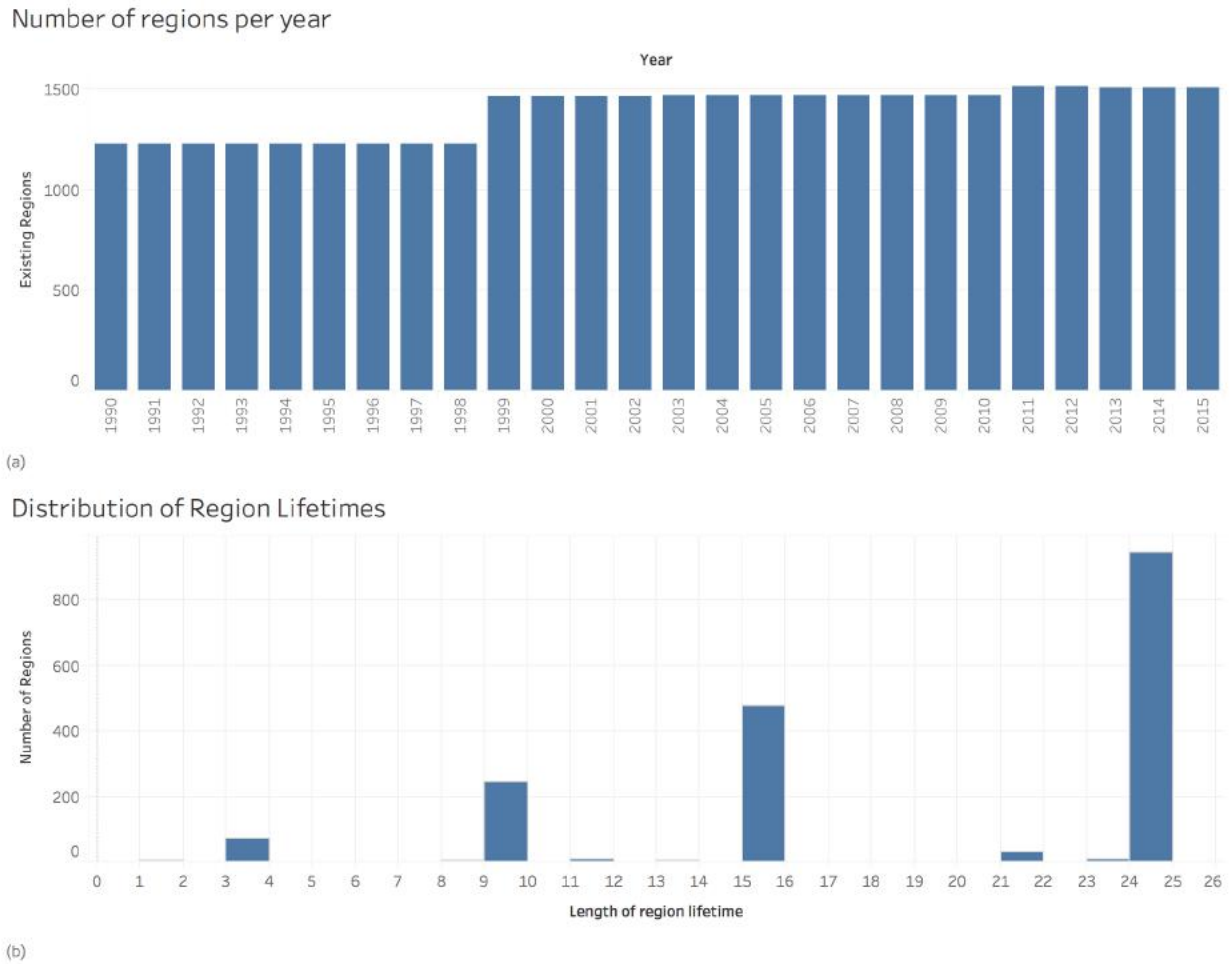


Figure 32: Temporal properties of selected countries from GAUL dataset.

These two graphs illustrate some of the temporal dynamics of the countries being evaluated. (a) plots the number of regions over time, showing two major reorganizations between the years 1998-1999 and 2010-2011. (b) Shows the *lifetime* of a given region, the number of years for which it existed. This shows clusters at 3 years, 9 years, 15 years, and 24 years (which is the entire length of the study period).

5.3.3 *Algorithm Implementation*

The implementation of the split/merge algorithm is achieved through the use of the *Hadoop* computing framework[41], which allows us to parallelize the computation across a number of separate computing nodes. For this experiment, we utilized the *Hortonworks HDP 2.6.4* installation, which bundles Hadoop 2.7.3, along with a number of additional configuration utilities. The experiment was run on a small cluster of Dell Optiplex Desktop computers, each with 2 cores and 4 GB of RAM, running Ubuntu 16.04.

[41] http://hadoop.apache.org

Hadoop is a *map/reduce* framework in which data is loaded in parallel and *mapped* into key/value sets, with a single key containing zero or more associated records. These sets are then sorted and sent to the *reducers* where the bulk of the computation is performed, and results reported. In general, Hadoop is best suited for computational problems where the input set is reduced into a single result that is reported for each input key (the canonical example is counting the occurrence of words in a set of documents or computing the average flight time for airline routes over the past year) in a stateless manner. Meaning, that each individual key/value set does not rely on the result of any other key/value set. While this creates some friction between the computing framework and the algorithm implementation, which requires some way to maintain the state of the computation (e.g. which Trestle_Objects have already been evaluated and associated together), it provides several strengths that were leveraged in the implementation.

At startup, Hadoop, in parallel, loads a set of input files from a shared location, which are then distributed to each of the cluster nodes. In this case, the input files are the ESRI Shapefiles distributed by the *United Nations Food and Agriculture Organization*. Next, the *mapper* phase processes the input files and emits a series of *key/value* pairs, which correspond to an application specific key which uniquely identifies each GAUL region within each year of the dataset. In this application, we opted to create our own keys which combine the GAUL code and the name of the region to create a unique key. While the invariants of the GAUL dataset state that each region is assigned a unique ADM2 Code, we found instances of codes being reused between several regions (an example is Nassarawa and Nassarawa Egon, two distinct regions in Nigeria which share the same GAUL code, *23049*) and thus opted to add an additional piece of information to enforce uniqueness.

The output of the mapper is the unique key for each region, and a set of values that correspond the entry of that region for each year it appears in the GAUL dataset. An example is given in Table 18. This simplifies the algorithm implementation in that all the years of a given region are processed at once, rather than individually for each year.

Key	Value	Value	Value	Value	Value	Value
Bilene	1990	1991	1992	1993	…	2014
Cidade de Maputo	2013	2014	2015			
Aeroporto	1990	1991	1992	1993	…	2012

Table 18: Key/value layout of GAUL records in the Hadoop framework.

This illustrates the *key/value* layout of the year data in the Hadoop framework. Each *key* contains the total number of records linked to the given region in the GAUL dataset.

Once the key/value sets have been generated, the data is partitioned between the nodes with all the regions for a single country sent to the same node. While this is not the most efficient parallelization approach (since some countries may contain far more regions than others while other countries may have a higher degree of spatial variance) it has the benefit of being fairly trivial to implement in Hadoop and ensures that regions for a single country are not distributed across the nodes, which improves performance and reduces the need to synchronize operations between the nodes.

Once the data mapping has been completed, the key value sets are passed to the *reduce* phase, which performs the bulk of the computation. The reducer first inspects in individual key/value set and determines whether or not the given key contains all possible records for the input space. Meaning, *does a record exist for each year of the GAUL dataset?* It is possible to do this because, as mentioned earlier, all the records for a given region are present in the same key/value set and can be processed in a single operation. If this query holds true, we know that the given region did not undergo any spatial change and can be loaded directly into Trestle. This is accomplished via the *merge* logic described in Section 4.2.3.

If the input key does not contain a record for each year of the GAUL dataset (e.g. Aeroporto only has records from 1990 to 2012), the application then proceeds through the algorithm logic described below. For each input key (labeled as the *base individual* in the process below), the algorithm performs the following actions:

1. Intersect the base individual with any other *Trestle_Collections* in the database to determine if this individual could potentially be related to any previously seen objects.
2. If no collections intersect, create a new *Trestle_Collection* for the base individual and load it, along with all its associated records, into the database. Repeat the process for the next individual.
3. If any collections do intersect, for each collection:
    a. Get the spatial value for the base individual
    b. As the spatial value may be a multi-geometry (e.g. MultiLineString, MultiPolygon, etc.), for each geometry in the spatial value, get the exterior ring as a polygon and add to an array.
    c. Create a geometry union from the array of exterior polygons and union them together.
    d. Get all the members of the matching collection.
    e. For each member in the collection:
        i. Get the spatial value for the member individual.
        ii. As the spatial value may be a multi geometry (e.g. MultiLineString, MultiPolygon, etc.) for each geometry in the spatial value, get the exterior ring as a polygon and add to an array.
        iii. Create a geometry union from the array of exterior polygons and union them together.
        iv. Add the resulting union to an array of the exterior ring polygons of all the concept members.
    f. Create a union of the array of exterior ring polygons of all concept members.
    g. Calculate the areas of both the concept union and the individual union.
    h. Compute the intersection of the concept union and the individual union and calculate the resulting area.
    i. If the area of the intersection, divided by the larger of the area of either the concept union or the individual union, is greater than 0 add all the concept members to a collection of potentially matching objects.
4. If 1 or more objects are potentially matching:
    a. Search for a spatial union using the *Spatial Union Algorithm* (described below) between any of the matching objects and the base individual. If a spatial union exists, create a union between the objects in the database.

    b. For each matching object, determine any spatial or temporal relationships between the matching object and the input individual and write them into the database.
5. If no objects match the base individual, write the individual into the database and create a new collection containing only the base individual.

This algorithm is performed for each key/value set in the GAUL dataset. Trestle handles deduplication of data, so if a key has already been associated with a spatial union, no additional data is stored in the database, even though the spatial union algorithm may be executed multiple times.

Spatial Union Algorithm:

The spatial union determination is accomplished through the following algorithm, which assumes as a starting point, an input set of the objects to evaluate for a spatial union. The following actions are performed:

1. Divide the objects into a set of *early* and *late* objects through the following process:
    a. Sort the input set using the *Temporal Comparison* algorithm (described below)
    b. For each object in the sorted set:
        i. Get the start and end temporals for the object.
        ii. If the current end date is null, set the current end date equal to the end date of the object.
        iii. Else, if the current end date is not after the end date of the object, add the object to the set of early objects.
        iv. Else, if the object start date is after the current end date or the object start date is equal to the current end date, add the object to the set of late objects.
        v. Return a new object with the sets of early/late objects.
    c. If there are no early *or* late polygons, then there cannot be a spatial union, return an empty value.
    d. Extract the geometry value from each object, keeping the geometries for the early/late objects in separate sets.
    e. Determine match direction (e.g. whether we are looking for a SPLIT or a MERGE) by the following algorithm:
        i. If we have more early objects than late objects, return *SPLIT.*
        ii. If we have more late objects than early objects, return *MERGE*.
        iii. If we have an equal number of early/late objects, return *UNKNOWN*.
    f. Given the match direction, do the following:
        i. If *MERGE*:
            1. Create a new queue the size of the number of late objects, sorted by the strength of the *Union Calculation Algorithm* result (described below), in descending order.
            2. Perform the *Union Calculate Algorithm* for each late polygon, and the set of early polygons, if the algorithm returns a value, add it to the queue.
        ii. If *SPLIT:*
            1. Create a new queue the size of the number of early objects, sorted by the strength of the *Union Calculation Algorithm* result, in descending order.

            2. Perform the *Union Calculate Algorithm* for each early polygon, and the set of late polygons, if the algorithm returns a value, add it to the queue.
        iii. If *UNKNOWN:*
            1. Perform both the *SPLIT* and *MERGE* steps of the algorithm using a combined queue the size of the total number of early/late objects.
    g. Return the first result (spatial union with the highest strength) from the queue. If the queue is empty, return an empty value, indicating no union exists.

Temporal Comparison Algorithm

This algorithm sorts Trestle_Objects to determine which object is *before* or *after* the other. One key distinction is that for an object to be considered before or after another object, it must be *entirely* before or after. Any temporal overlap between two objects results in a *during* relationship. This is illustrated in Figure 33.

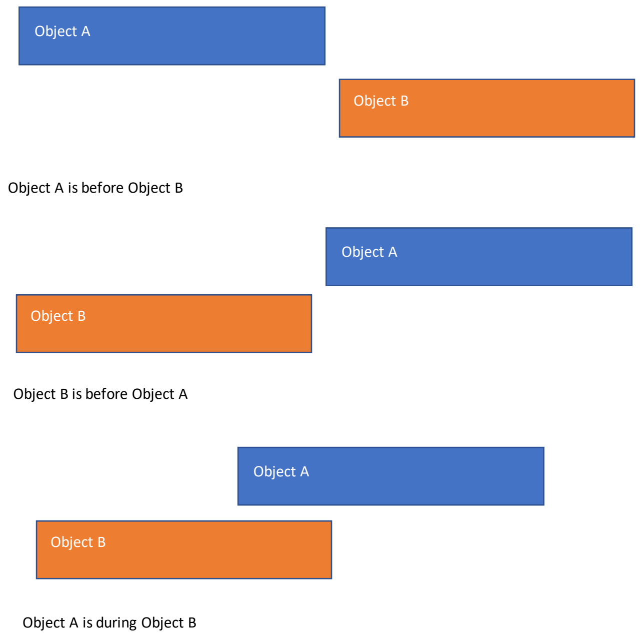

Object A
Object B
Object A is before Object B
Object A
Object B
Object B is before Object A
Object A
Object B
Object A is during Object B

Figure 33: Examples of temporal relationships in Trestle.

This figure illustrates how Trestle handles *before/after* temporal relationships. In order for Object A to be before Object B, it must be *entirely* before having *no* temporal overlap with Object B. The same holds true for the *after* relationship, it must come entirely *after* Object B. Any amount temporal overlap results in a *during* relationship.

Determining these relationships is accomplished through the following algorithm:

1. Given two objects (Object A and Object B), compare the start temporal of Object A with the ending temporal of Object B.

    a. If Object B has an ending temporal, and ending temporal comes before the start temporal of Object A, then Object A comes *after* object B, return *after*.
    b. If Object B does not have an ending temporal, then Object A cannot come *after*, but may be *during* or *before*.
        i. If the start temporal of Object B is *after* the ending temporal of Object A, then Object A occurs *before* Object B, return *before*.
        ii. If the previous conditions are not met Object A must occur *during* Object B, return *during*.
2. If Object B has an ending temporal and the ending temporal does not come *before* Object A, then Object A might be *before* or *during* Object B. Compare the start temporal of Object A, with the start Temporal of Object B:
    a. If the start temporal of Object B occurs *after* the start temporal of Object A:
        i. If the start temporal of Object B occurs *at* or *after* the ending temporal of Object A, then Object A occurs *before* Object B.
        ii. Else, Object A occurs *during* Object B.
    b. Else, Object A occurs *during* Object B.
3. If Object A does not have an ending temporal, then Object A occurs *during* Object B.

Union Calculation Algorithm

The actual determination of whether or not a spatial union exists between a given set of *input polygons* and a set of *match polygons* is handled by the following algorithm. At its simplest level, it attempts to determine if there exists a spatial equivalency between one of the input objects, and any other combination of potentially matching objects that exceeds a given *cutoff* threshold. This threshold sets the limit for the amount of difference (in terms of percentage of spatial area) allowed between the input object the potential union of matching objects.

1. Split into *input polygons* into *input polygon powerset* using the *Powerset* algorithm (described below).
2. For each *input set* in the power set:
    a. Perform a spatial union for all members of the input set.
    b. Create a powerset of the *match polygons*
    c. For each set in the *match polygon powerset*:
        i. Perform a spatial union for all member of the match input set.
        ii. Calculate the P*ercent Similarity* (described below) between the spatial union of the match input set and the spatial union of the input set.
        iii. If the percent similarity exceeds the cutoff threshold, return the potential match set.
    d. If a match is returned for a set in the *match polygon powerset*, return that value.
3. If no match is found for any *input set* return an empty value.

Powerset calculation

The *powerset*, for a given set *S*, is the set of all subsets of *S*, including the empty set and *S* itself. This allows the algorithm to try each possible combination of input polygons. It is accomplished through the following steps:

1. Create a sorted *output set* of sets of polygons, which sorts those sets based on the number of polygons they contain.
2. Create a new queue of the all the input polygons.
3. If the queue is empty, return the sorted *output set* of polygons.
4. If the queue is not empty, get the first polygon in the queue.
5. Create a new set with all the remaining polygons in the queue.
6. Compute the *powerset* for the remaining polygons
7. For each set in the newly computed *powerset* of remaining polygons:
    a. Create a new set and add the first polygon from the queue.
    b. Add all the polygons from the *powerset* set to the newly created set.
    c. Add the newly created set to the sorted *output set*.
8. Return the *output set*.

Percent similarity calculation

Calculating the percent spatial equality between two polygons (*inputPolygon* and *matchPolygon)* is accomplished through the following process:

1. Determine the largest spatial area between the two polygons.
2. Compute the spatial intersection between the *inputPolygon* and the *matchPolygon*.
3. Return the area of the spatial intersection divided by the largest spatial area of the two polygons.

5.4 ANALYZING THE ALGORITHM RESULTS

Once the integrated dataset was built, the algorithm results were compared to determine the overall utility of this split/merge approach to describing spatial change. For the evaluation, we randomly select 100 regions from the integrated dataset and manually compared them against a gold standard to determine which of the following categories each random region belongs to:

True negative – The algorithm correctly identified the region as not a part of a spatial split/merge.
False negative – The algorithm incorrectly identified the region as not a part of a spatial split/merge.
False positive – The algorithm incorrectly identified the region as a part of a spatial split/merge.
True positive – The algorithm correctly identified the region as a part of a spatial split/merge.

The gold standard comparison was done through the use of custom tooling developed using Trestle and the prototype web application, which is shown in Figure 34. More details will be given in Appendix A, but will be briefly described here. The map at the top of the image allows for loading the Trestle dataset in three dimensions, with the various regions laid out on the Z-axis based on their temporal existence. The map can then be rotated, zoomed, and the various regions can be moved up and down in the Z-axis to better understand the spatial interaction of the various regions. A given region (target region) can then be spatially intersected with other regions and any overlapping areas are added to the map and their percent contribution to the target region is shown on the screen. In this example, we can see a comparison for the *Ankpa* region (in blue) alongside *Omala* (dark red) and *Ankpa* (light red), each of which contributes a portion of the area for the target region. This clearly shows that the target region (Ankpa) splits apart into *Omala* and *Ankpa*[42].

Once the gold standard values were generated, they were compared against the algorithm results. This experiment was run using spatial cutoff values of 90%, 95%, 97.5% and 99%. The full results are given in

[42] The temporal existence of the GAUL regions is encoded in their unique identifiers. In this case, *22901-Ankpa-1000-1999* indicates that the *Ankpa* region (GAUL code 22901) exists from the year 1000 to the year 1999. This region is drawn on top of *Ankpa (1999)* and *Omala* due to manual map changes by the user. By default, *Ankpa* would be drawn underneath the two later regions.

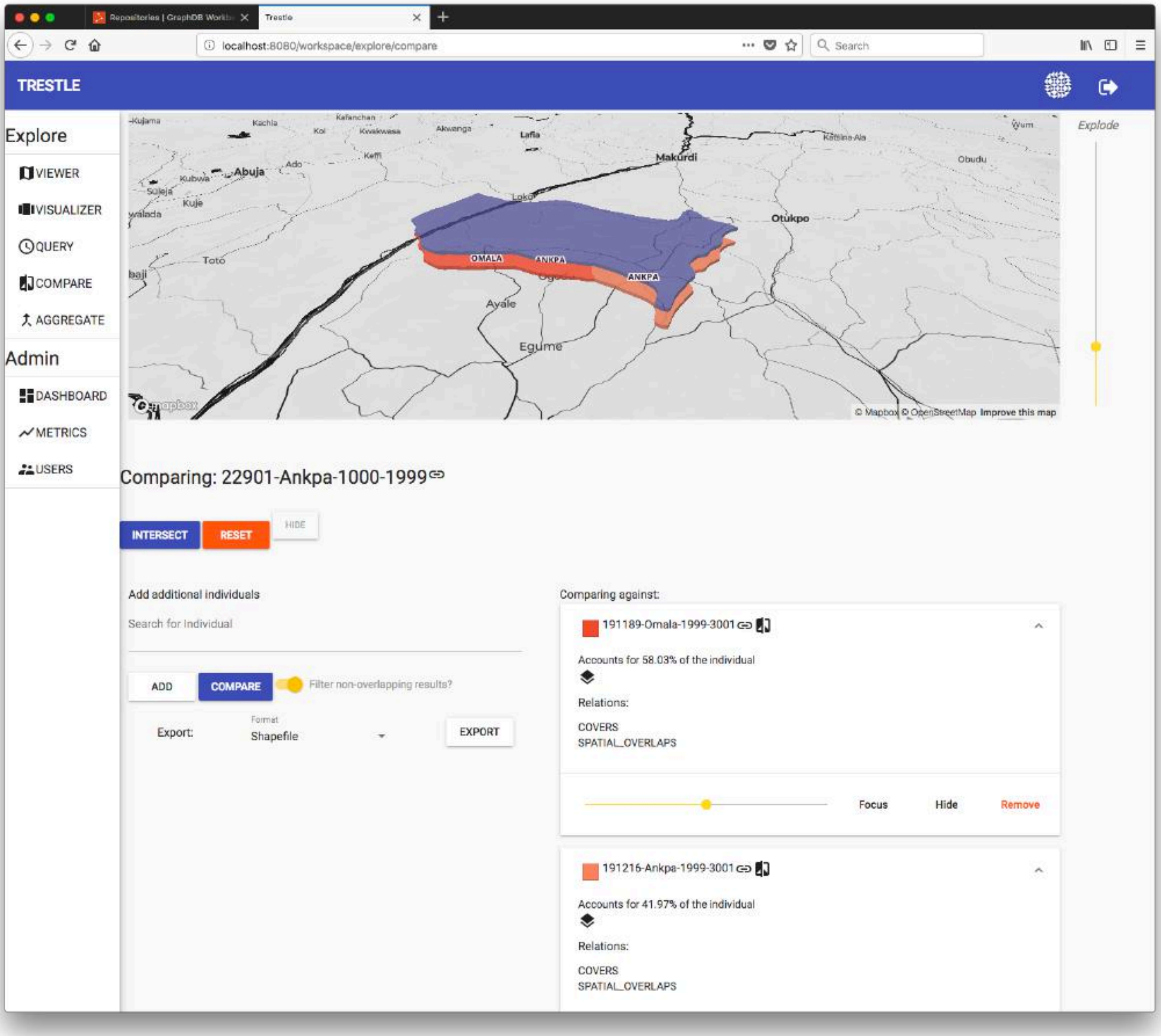
Appendix B, but the major findings are summarized here. Given the results in Table 19, two things are immediately apparent. First, the algorithm was able to correctly identify whether or not a spatial union exists in 98% of the test cases. Given the random sampling, the algorithm only had a single false negative (incorrectly ignoring a spatial union that should exist) and zero false positives (incorrectly identifying a spatial union where one should not be). Second, there is no difference in algorithm performance between the various cutoff values in the sample analyzed. This seems to indicate that the majority of potential spatial unions were nearly perfect matches and the algorithm was able to clearly determine whether or not a split/merge event exists.

Given these findings, the next question that arises is how much spatial change, from the input dataset, the algorithm was able to associate with split/merge events. In total, 46.82% of regions (839 out of 1792) underwent some type of spatial change during the study period; of these, the algorithm was able to match 609 regions with a split/merge event, which accounts for 72.59% of the changed regions in the study.

	90% cutoff	**95% cutoff**	**97.5% cutoff**	**99% cutoff**
True Positive	69	69	69	69
True Negative	30	30	30	30
False Positive	0	0	0	0
False Negative	1	1	1	1
False Positive Rate	0	0	0	0
True Positive Rate	.9857	.9857	.9857	.9857

Table 19: Algorithm evaluation results.

This table presents the results of the algorithm evaluation for the four cutoff values used in this experiment. *True negative* indicates that the algorithm correctly identified the region as not a part of a spatial split/merge. *False negative* indicates that he algorithm incorrectly identified the region as not a part of a spatial split/merge. *False positive* refers to the algorithm incorrectly identifying the region as a part of a spatial split/merge. *True positive* describes the algorithm correctly identifying the region as a part of a spatial split/merge. This table also shows both that the algorithm has a high degree of accuracy and that there is no difference between the algorithm performance for the given cutoff values.



Comparing: 22901-Ankpa-1000-1999

Add additional individuals

Search for Individual

Filter non-overlapping results?

Export: Format Shapefile

Comparing against:

🟥 191189-Omala-1999-3001

Accounts for 58.03% of the individual

Relations:

COVERS
SPATIAL_OVERLAPS

🟧 191216-Ankpa-1999-3001

Accounts for 41.97% of the individual

Relations:

COVERS
SPATIAL_OVERLAPS

Figure 34: Comparison tool for detecting split/merge events through 3D map visualizations.

This comparison tool, from the prototype web application, will be discussed in more detail in

Appendix B.

Through the evaluation process we were able to continually refine the algorithm to increase its match percentage. One example is shown in Figure 35 and Figure 36. This was discovered when evaluating the results from Congo. Here, *Ouesso* county is resized to a fraction of its original area and *Mokeko* is organized around it. This was originally not caught by the algorithm, which performed the spatial computations on the boundaries of the objects which meant that when evaluating *Mokeko* for potential matches, it was not matching against *Ouesso* due to the hole in the interior of the boundary which perfectly encompasses *Ouesso*. To account for this, we added a cleanup process which removed holes from the spatial boundaries, when performing the initial spatial intersection, which increased the number of candidate counties which the algorithm could then evaluate. The actual spatial union computation was not changed and used the true boundaries of the Trestle_Object.

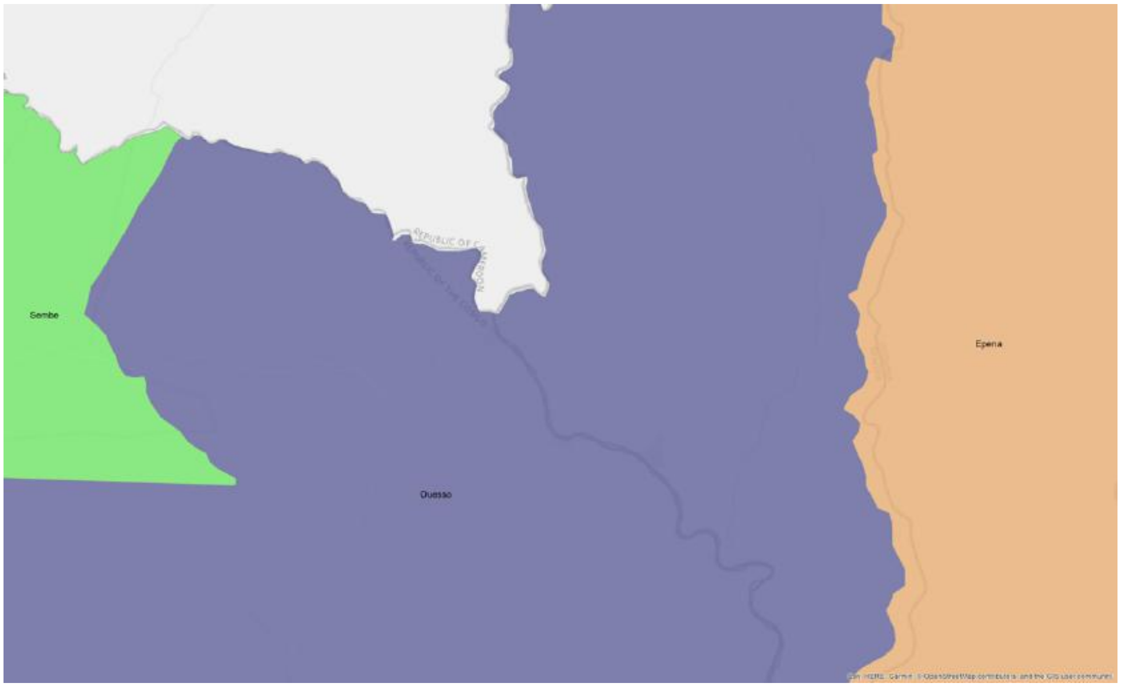


Figure 35: Ouesso, Congo in 1990.

This map shows *Ouesso* county (purple as a large region touching *Sembe* (green) and *Epena* (orange).

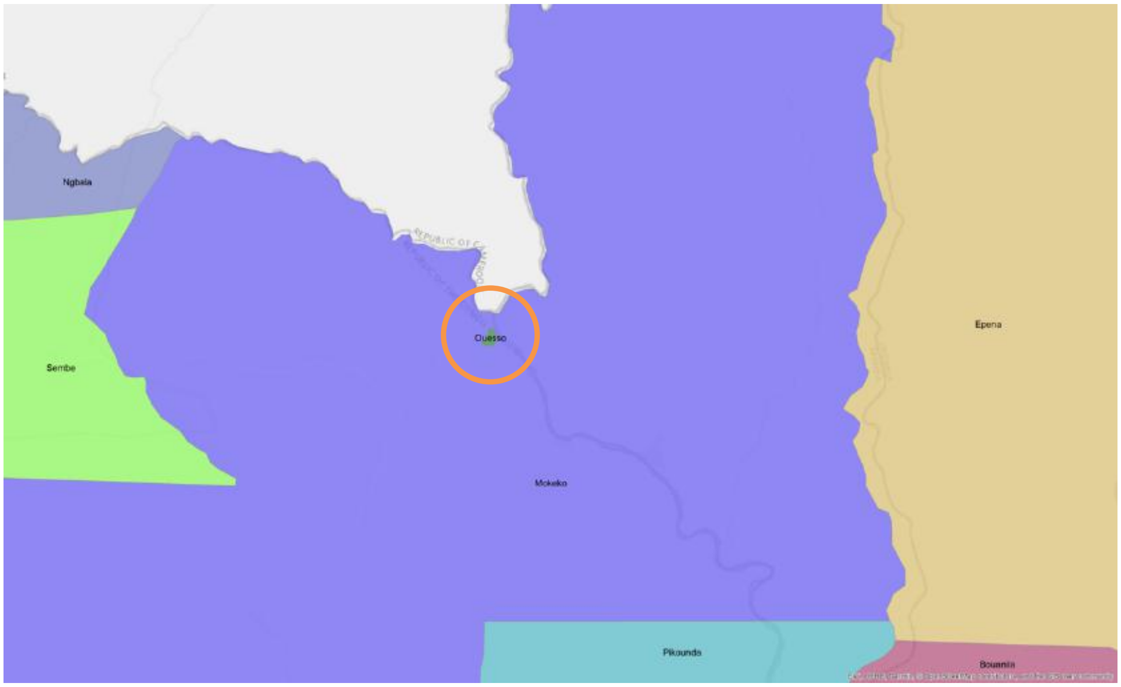


Figure 36:Ouesso, Congo in 2014.

This map shows *Ouesso* county (green and circled) resized to a fraction of its original area and perfectly encompassed by the newly created *Mokeko* county (in light purple).

5.5 Algorithm Limitations and Alternative Analysis Approaches

5.5.1 *Algorithm Limitations*

While the results of this split/merge algorithm are promising, there are some limitations. The first, is that some spatial changes cannot be reconciled by simple split/merge logic. Thus, leaving some percentage of spatial change un-accounted for by Trestle and back into the hands of the domain researchers to resolve, which is the current *status quo* for this type of process. Some of this un-accounted for change could be addressed by adding an additional level of reasoning to the algorithm. Whereas currently, the algorithm only attempts to match a single object in the initial temporal state against a set of objects in the second temporal state (e.g. matching Cidade de Maputo with earlier regions), it is possible to expand the algorithm to look for spatial unions between multiple objects in both temporal states. For example, in the original Congo counties example (Figure 26 and Figure 27), it could be expressed that the initial two counties are a spatial union of the later three counties. This would dramatically increase the computational complexity of the algorithm but may improve its ability to account for additional combinations of spatial change.

The second limitation is that spatial similarity may not always be the most appropriate metric for associating administrative regions. There may be situations in which a researcher wishes to associate areas based on a different attribute such as population count, instead of purely geographic congruence; which will be briefly touched on in the following section. The final limitation is that utilizing the ability to associate regions over time for use in research, relies upon an underlying assumption that non-spatial invariants hold true between the two spatial states. This means, that when comparing an indicator between an initial region and the regions it splits into, simply because the two temporal states cover the same spatial area, does not necessarily imply that other factors have not changed drastically between the before and after state of the spatial change. The researcher is still required to ensure that their model assumptions hold true for all spatial and temporal states of the study.

5.5.2 *Alternative Analysis Approaches*

While there are a number of different ways to analyze the algorithm results, we selected our approach due to its simplicity of implementation and well-documented support in the literature [166]. That being said, there are multiple ways in which the evaluation could be improved and expanded on. The simplest approach is to expand the number *cutoff values* used in the analysis (e.g. setting the value to 60%, 80%, 99%, etc.), in an attempt to increase or decrease the number of false positives or negatives found by the algorithm. Likewise, a future analysis could expand the number of randomly sampled regions in order to increase the probability of encountering a potential false positive or negative. Beyond these simple means of extending and improving the initial analysis, this sub-section will also briefly describe two alternative analysis methods that could be of use.

The first alternative approach is to rather than perform a random sampling of candidate regions, instead sample from a set of potential *outlier* regions; meaning those regions which have some unique spatial property that falls outside the standard deviation of property values for that dataset. Since this algorithm is focused on computing spatial overlaps between regions, these outliers could be regions that are extremely large or extremely small in terms of spatial area (shown in Figure 30) which would help determine the algorithm's sensitivity to the area of the regions being matched. In addition, regions could be selected that have either a high degree of spatial complexity (e.g. the number geographic points in the region's boundary) or a high ratio of complexity to spatial area. This would help validate the algorithm's ability to handle very complex geographies in an accurate manner. The final method for determining potential outliers is to compute the spatial complexity for a given region as well as the complexity for each adjacent object. The result of this computation would be a *spatial adjacency graph* in which each region is a *node* on the graph and links to all other adjacent regions (nodes). The size of the nodes corresponds to the complexity metric utilized with more complex nodes having a larger value. From there, the nodes with the highest total complexity (e.g. the sum of the complexity of a given region along with the complexity of all adjacent regions) could be selected to use as the inputs for the analysis. This graph-based approach is a natural fit for Trestle and the types of algorithms that can natively interact with the underlying graph layout in the triple-store. More details on these types of complex graph algorithms will be given in Section 6.3.1.

The second alternative approach is slightly different from the ones discussed so far in that rather than strictly analyzing the algorithm's results at a given cutoff value, it instead looks to see at which cutoff value the algorithm no longer returns accurate results. The could be done by picking a high initial cutoff value (e.g. starting with 99%), executing the algorithm and then both counting the number of split/merge events found by the algorithm as well as computing the true positive and true negative rates for the initial random sample. From there, the algorithm would be repeatedly run using lower and lower cutoff values, based on a given step interval (e.g. 95%, 90%, 85%, 80%, etc.); at each interval, the number of split/merge events is counted and plotted on a graph. Once the results have been gathered and plotted, for each interval in which the total number of split/merge events increases (signally a potentially higher rate of false positives) above a given threshold the initial analysis (described in Section 5.4) is performed and the true positive and true negative rates are reported. Once the true positive rate drops below a given point (e.g. 80%) then that is point at which the algorithm no longer returns accurate and only the higher cutoff values should be used for determining split/merge events. This approach can be redone for each new dataset (or geographic area) being studied and the results used to set an appropriate cutoff value for returning data to the users.

5.6 Future work: alternative algorithm designs

While the algorithm described in this chapter has been shown to be effective for reconciling changes in administrative boundaries, there are alternative algorithm designs that could be effectively implemented on top of Trestle. Two of these approaches will be briefly detailed in this section but there are numerous other approaches that could be considered as well. Both focus on the idea that spatial area may not be the most descriptive method for associating regions.

The first alternative approach is to utilize methods common in the generation of *cartograms*. A cartogram is a mapping technique in which the initial spatial layout is rescaled (or distorted) to present not merely geographic information, but the relative proportions of some value within the geographic space. One use of cartograms is to redraw US states in the same relative position as they appear on map, but resized to their proportion of the total population or tax revenue, as is shown in Figure 37 (though any other indicator could be used to scale the map as well). While there is some analytical critique as to the accuracy and interpretability of cartograms [169], they could have some utility in addressing the region matching problem. For example, many public health indicators are calculated based on normalized population numbers. This means that for some research problems matching a region based on total spatial area is of less value than matching a region based on total population. For example, a potential split/merge event may not spatially match because a small amount of additional area is included from another region; however, if this small extra area contains only a small number of people (or none at all) it may be sufficient (and perhaps more useful) to identify a split/merge based on the fact that all of the population from the initial region is accounted for in the candidate regions. This approach is further motivated by the fact that there is often a strong inverse/correlation between population size and region area. Thus a small, heavily populated region could have an outsized effect on the statistical outcomes and contributions of a given region.

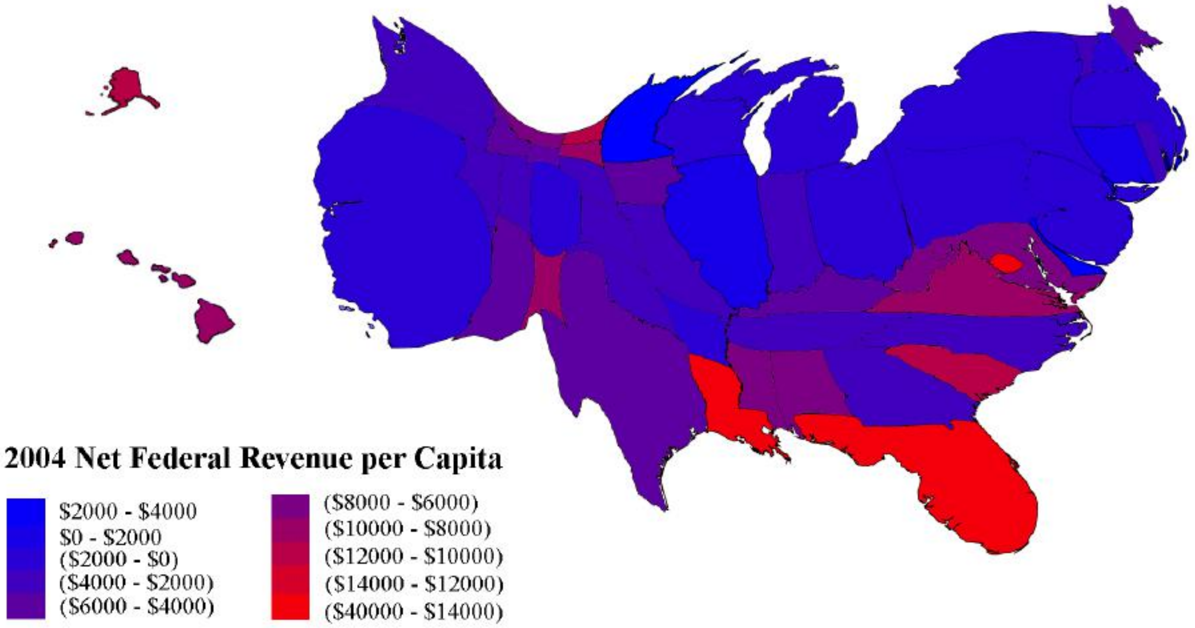


Figure 37: A cartogram of the US showing the size of each State (in 2004) on the basis of Federal Tax contribution [170].

A second alternative approach, proposed by Martin in 2003, termed *Automated Zone Matching (AZM)* [164], was originally designed as a way to match spatial regions which were generated from different datasets (e.g. census tracts vs zip codes) and though they account for different spatial area, could potentially be used in the same research project. The key component of AZM is that it computes a *stress* factor based in differences in various data properties, such as a proportion of the population count. This approach straddles the line between directly linking changed boundaries based on a given similarity and creating new regions for analysis (as will be described further in the following chapter). This proves to be a natural extension of the algorithm described in this chapter in that it follows the same type of search procedure (e.g. identifying candidate geographies based on spatial overlap) but adds an additional step in determining similarity between potential aggregations based on statistically significant differences in the various data properties. The downside of this approach is that not only is it more computationally expensive, but it requires manual intervention on the part of the researcher in order to determine which properties should be considered in the stress metric and how to appropriately weight the various measures. Likewise, use of this approach makes it difficult to determine if the *optimal* zoning match has been reached, as there are numerous methods for scaling and weighting the data property values. The benefit of this approach is that it allows for matching regions which may not be exactly identical in terms of their spatial area, but similar enough in terms of population measures as to support more direct statistical comparisons between the two temporal states.

5.7 Evaluation of Trestle’s Object Relationship Support

As the first phase of the two-phase evaluation outlined at the beginning of this chapter, one goal of the algorithm implementation is to determine whether or not the Trestle data model and management application has the ability to effectively represent different types of relationships between the various Trestle_Objects. After executing the algorithm over the GAUL dataset, the resulting dataset includes four distinct types of relationships:

1. *Spatial relationships:* Each Trestle_Object is spatially related to all other Trestle_Objects that contain some spatial interaction between the two (e.g. touches, overlaps, etc). This means that the newly generated dataset can directly answer spatial queries involving binary spatial relationships (as described in [111]) without requiring those relationships to be computed at query time. In addition, these spatial relationships include the logical extensions described in Table 12.
2. *Temporal relationships:* In the same manner as the spatial relationships described above, this dataset also contains the full set of temporal relationships as described in [113] and Table 12. It should be noted that the temporal relationship expressed in this dataset are only computed for Trestle_Objects that are either spatially related to each other, or logically inferred by the reasoner. This means that *Cidade de Maputo* is specified to be *after* the counties it is spatially equivalent to (see Figure 20 and Figure 21), but has no temporal relationships expressed with counties in Nigeria. This is not a limitation of the Trestle application, but merely a performance optimization in order to avoid dramatically increasing computational complexity. In addition, it was deemed sufficient to provide temporal relationship at the same level of detail as the corresponding spatial relationships.
3. *Spatial unions:* For each spatial union determined by the algorithm, Trestle maintains a split/merge relationship between each of the objects
4. *Trestle collections:* In addition to the direct relationships (spatial, temporal, split/merge), the resulting dataset also includes collections of related objects that share some type of association (spatial or temporal), which allows researchers to quickly gather spatially related objects that may not have a corresponding spatial equivalency (such as the counties described in Figure 24 and Figure 25).

Through these four relationships, it is clearly demonstrated that Trestle has the ability to effectively describe a large number of object relationships in a way that provides direct utility to existing public health research.

5.8 CONCLUSIONS

To summarize, this chapter outlined a common challenge in longitudinal public health research and detailed a novel approach for utilizing Trestle to design and implement an algorithm for spatially and temporally integrating a commonly used spatial dataset. We then validated the idea that a split/merge paradigm can be a useful method for accounting for spatial change over time; especially in a spatially exhaustive dataset such as GAUL. Finally, through the design and implementation of the split/merge algorithm we evaluated the ability of the Trestle system to effectively represent complex relationships between ST-Objects. The following chapter will conclude the evaluation phase by focusing on Trestle's ability to manage and query the internal state of ST-Objects.

Chapter 6. EVALUATION 2: REGIONALIZATION

In the previous chapter, we outlined the two-phase evaluation process for the Trestle data model and management application. Chapter 5 evaluated Trestle's ability to answer research questions which rely upon complex relationships between ST-Objects. What remains left do be done, is to determine whether or not Trestle can provide a similar level of support for answering research questions which require valid sets of information at specific time points, for a given ST-Object. It is this component that will be the focus of this evaluation chapter.

While there are multiple potential methods for evaluating Trestle's performance in this area, this chapter will follow the approach described in Chapter 5 (solving a public health research challenge, in order to demonstrate system function) and attempt to address an additional challenge in public health research, namely, determining the appropriate spatial and temporal scope for a given research project. On significant difference is that while Chapter 5 developed its own algorithm to solve the given challenge, this chapter will focus on integrating existing approaches and building them on top of the Trestle system. To help contextualize this issue, consider two existing research challenges:

1. US census data is a common source of information used by public health researchers due to its high level of detail and availability at multiple levels of spatial resolution ranging from the state level down to granular census blocks. While researchers often wish to use the most granular data possible, there are privacy restrictions that prohibit publication of findings if there is a potential for identifying individuals in the study area.
2. When attempting to determine the appropriate length of time to perform a longitudinal study, researchers are often forced to consider whether or not any of the spatial regions being studied have been changed during the study period. Another way of looking at this problem is trying to determine the longest period of time for which a certain amount of the spatial area remains consistent.

One solution to both of these issues is a technique called *regionalization*, which is the process of creating new aggregations of existing spatial objects that are specifically designed to address a given research question. These spatial aggregations can be designed to solve either of the two challenges mentioned above. This process is well defined in the geospatial literature, but requires access to both the underlying spatial objects, and their associated values, as well as the spatial relationships between them. Gathering this required information is a manual process and one that is repeated for each unique invocation of the regionalization algorithm. This *artisanal* approach to region design becomes difficult to sustain as the both the size and complexity of the input data increases. We propose that Trestle provides unique capabilities that simplify the design and implementation of these types of algorithms. Primarily through its ability to quickly retrieve the temporally correct data properties for the initial spatial objects, as well as its support for higher-level aggregations which can be used to store and retrieve the results of the regionalization process. We also propose that these types of algorithms provide an excellent platform for fulfilling the second part of the two-phase evaluation proposed in Chapter 5.

Section 6.1 gives a brief introduction into the background of the *regionalization* problem. Section 6.2 describes three major approaches (AZP, SKATER, and REDCAP) which have been developed to address this issue. Section 6.3 describes how the REDCAP algorithm might be integrated with

Trestle via both a direct translation, as well as by modifying the algorithm to better take advantage of the unique features of Trestle. Section 6.4 outlines some future work for both extending the Trestle application to better support these types of research algorithms, as well as for more fully integrating the temporal dimension into regionalization.

6.1 INTRODUCTION TO REGIONALIZATION

Within public health research, it is extremely common to perform studies using existing spatial definitions delivered by a national or regional body. Some of these datasets have been described previously in this dissertation, including GAUL, and the US TIGER dataset, often combined with ACS data properties. These datasets provide the spatial foundation for a large proportion of existing research efforts.

One primary benefit of this approach is that this data is easily accessible, well understood, and often directly comparable with previous studies and research projects. In addition, as has been described in Section 1.2, public health has an inherent political and administrative context and is often required to provide evidence for policies and interventions within a specific political and administrative frame. Phrased another way, <u>if policy occurs in counties, then the data should be reported as such</u>.

But a major downside of this approach is that these *base spatial units* (BSUs) were originally designed to optimize for situations other than spatial research. For example, ZIP code areas are optimized for delivering letters and thus gave rise to ZIP Code Tabulation Areas (ZCTAs) that modified the boundaries to better serve census data gathering [171]. Likewise, the design of other types of BSUs, such as county borders or census blocks are often based on convenience and not necessarily on the most optimum spatial layout [172].

The layout of these spatial regions (also referred to as the *zoning system*) is of critical importance to the researcher both through its enablement of their ability to observe a given phenomenon, as well as its effects on the underlying accuracy of the observations [173]. Thus, it is often desirable to have the ability to redistribute the spatial layout of a given study area, in order to create analysis units which more closely match the goals and requirements of a given study [174]. As the granularity of available spatial data increases, so does our ability to design custom geographies, optimized to the research task at hand, without sacrificing spatial quality.

This process of custom geography creation is known as *regionalization* and has been progressively developed and refined for most of the history of computerized geography [175], [176]. While there are numerous methods for constructing new geographic regions, as well as multiple variations on these methods, three primary approaches will be the focus of this chapter: AZP, SKATER, and REDCAP. Each of these approaches progressively builds upon each other (both conceptually as well directly referencing the other implementations) and aims to solve existing limitations as well as propose novel approaches to region design. The next section will describe each of the three algorithms, in detail.

6.2 OVERVIEW OF EXISTING METHODS

6.2.1 *AZP*

First described by Openshaw in 1977 [177] and refined further over the next several years [172], [173], the *Automated Zoning Procedure* (AZP) is one of the earliest regionalization approaches in geography. It is been successfully used to generate new layouts of the UK Census [172], [178], as well as reconcile datasets collected in different zoning layouts (e.g. counties vs census tracts) [164], [174].

Algorithm Design

In brief, the goal of the algorithm is to find the optimum redistribution of a set of BSUs (e.g. census block groups) into a set of larger *regions* (e.g. regions with a spatial resolution roughly equivalent to U.S. zip code areas). In order to appropriately generate the regions, AZP requires an *objective function* that can compute the *quality* of a given region. This function is entirely left up to the user, but some guidance for selecting an appropriate function is given in [173]. In addition, in their 1995 paper, Openshaw and Rao describe the optimization function they used to create new zones from the 1991 UK census which contain roughly the same numbers of older adults.

Equation 1: AZP optimization function

$$F(Z) = \sum_{j}^{m} abs\left(\sum_{i}^{n} \delta_{ij} P_i - T_j\right)$$

Equation 1 determines the quality of a potential optimization solution (e.g. a proposed distribution of BSUs into larger regions). Here, δ_{ij} is set to 1 if BSU i is a member of Region j, otherwise it is set to 0. P_i is the number of older adults in BSU i and T_j is the target population of older adults in region j, which is specified by the user before executing the algorithm. This optimization function could be modified to support a broad range of data properties and statistical functions; such as setting a min/max population count or optimizing for the smallest amount of variance between a given set of data properties (as described in more detail in Section 6.2.2).

Once the objective function has been determined, the algorithm proceeds through the following steps:

1. Randomly partition the BSUs into a set of *M* spatially contiguous output regions.
2. Calculate the value of the objective function for all the initial regions.
3. Randomly select a region *K* from the list of *M* regions.
4. Find all BSUs that border region K.
5. Randomly select a BSU from the list in Step 4, recompute the objective function with the BSU assigned to region K, instead of its original region. If the optimization function improves, move the BSU from its original region into *K* and update the optimization function value for both *K* and the region which originally contained the BSU. Repeat Step 4 for the newly updated region *K*. If the optimization function decreases or does not change, leave the BSU in its original region and repeat Step 5 with another BSU from the list generated in Step 4.

6. When all adjacent BSUs have been evaluated, return to Step 3 and repeat steps 4-6.
7. Repeat steps until no other optimizing moves can be made.

The end result is a set of regions that optimally fit the user specified objective function; however, the output of the function is not deterministic given the initial randomized starting point of the algorithm. The algorithm is also susceptible to converging on a local maximum, due to the fact that it only evaluates whether or not moving a single BSU would increase the value of the objective function. Openshaw and Rao did propose improving the algorithm through the use of techniques such as *simulated annealing* [179] and *tabu searches* [180], but at the cost of tremendously increased computational complexity [172].

6.2.2 *SKATER*

The *Spatial 'K'luster Analysis by Tree Edge Removal (SKATER)* algorithm, was originally proposed in 2006 as an improvement on the AZP method, which not only aimed to reduce computational complexity, but also to avoid situations in which AZP was susceptible to converging on a local maximum [181].

Algorithm Design

SKATER distinguishes itself from AZP in that rather than taking a purely geographic approach, it instead brings in ideas and concepts from the field of graph theory. Whereas AZP begins by randomly assigning all input regions into larger aggregations, and then progressively refining the aggregations, SKATER redefines the problem in terms of traversing a graph of spatially related objects. Given this different approach, SKATER introduces a number of new terms, common in graph theory, but less often used in geography. A *tree* is a graph of *nodes* (elements in the graph, such as census blocks) in which no two nodes are connected by more than one edge (a relationship between two nodes). A *spanning tree* is a tree containing all input regions where all the nodes are connected by unique paths. A *minimum spanning tree (MST)* is a spanning tree in which all the edges represent the least *cost* path between any two nodes in the tree. The *cost* of a given edge is computed based on the dissimilarity between the data properties of two regions. The design of the cost function is left up to the end user but requires two specific components. First, which data properties will be considered in the cost. An individual region may have numerous properties associated with it, but only a subset may be required in order to determine similarity (or dissimilarity) with other regions, thus the user must specify which properties to include in the given *attribute vector*. The second component is how to perform the actual comparison between the attribute vectors of the various regions. The paper authors recommend a standard *square of the Euclidian distance* metric for most types of data with comparable scales; however, researchers may wish to utilize their own metrics or apply a weighting function to the different attributes.

Once the cost function has been determined, the actual evaluation of the regions is performed in two distinct phases. 1) Build an MST of all connected regions. 2) partition the MST into distinct sub-trees that minimize the given cost function.

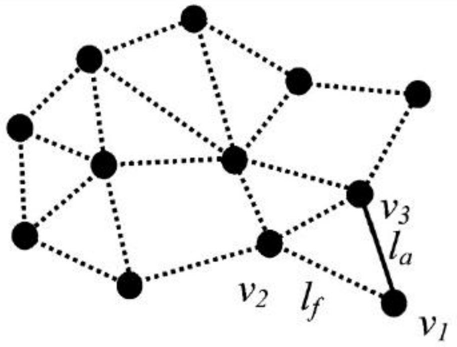
Phase 1: Generate MST

The algorithm first begins with the spatially contiguous graph of the input regions where each region represents a *node* on the graph and each *node* contains a number of *edges* linking to their spatially contiguous neighbors. It then randomly selects a region to begin with.

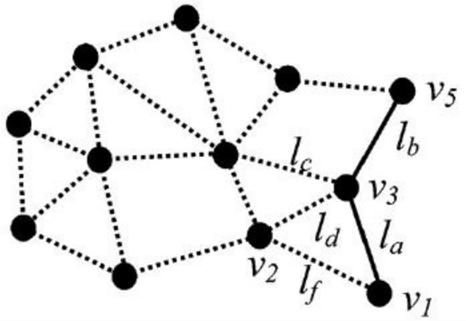
Starting with an MST that contains only the initial region (node) and no spatially contiguous links (edges); the algorithm proceeds through the following steps for each region in the spatial graph:

1. Find all the nodes that are spatially contiguous with the given node.
2. For each node, compute the cost of each edge which links to a node that is not a member of the MST.
3. For the lowest cost edge, add the node and the edge to the MST.
4. Proceed until no new nodes can be added to the MST (e.g. that there are no more vertices that can be added because they all point to members that have already been added to the MST).

This process is illustrated in Figure 38, where T_1 represents the MST, beginning with only the initial node. For each iteration, the new node is evaluated, and the lowest cost node/edge pair is added to the MST.

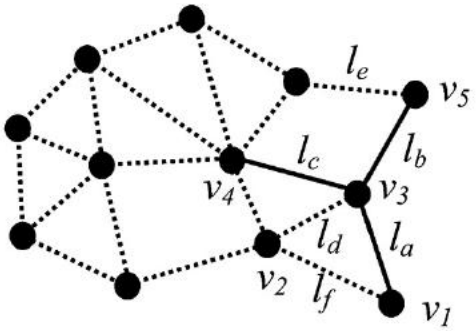


First iteration:
Set $T_1 = (V_1, L_1)$, where $V_1 = \{v_1\}$ and $L_1 = \phi$.
Find the edge of lowest cost ($l_a \langle l_f$).
Step 3: $T_2 \Rightarrow V_2 = \{v_1, v_3\}$ e $L_2 = \{l_a\}$.
Step 4: Repeat Step 2.

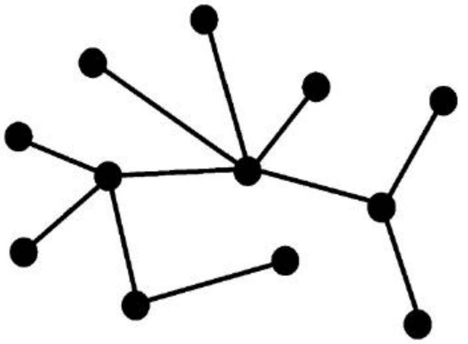


Second iteration:
Find the edge of lowest cost ($l_b \langle l_c \langle l_d \langle l_f$).

Set $T_3 \Rightarrow V_3 = \{v_1, v_3, v_5\}$ and $L3 = \{l_a, l_b\}$.



Third iteration:
Find the edge of lowest cost ($l_c \langle l_d \langle l_e \langle l_f$).
Set $T_4 \Rightarrow V_4 = \{v_1, v_3, v_4, v_5\}$ and $L3 = \{l_a, l_b, l_c\}$.



Final Iteration:
$V_n = V$.

Figure 38: Construction of the MST for *phase 1* of SKATER (Reproduced from [181]).

This image illustrates the process by which SKATER constructs the MST from the initial spatial adjacency graph. It first computes the *cost* of each edge and then starting with the lowest cost edge it progressively expands the initial tree by only select the lowest cost nodes, which connect a node that is not already associated with the tree. *T* represents the MST which consists of a set of nodes *(V)* and edges *(L)*, starting with the initial node *V_1*.

Phase 2: Partition MST

Once the MST has been created, it needs to be partitioned into separate *trees* of spatially contiguous clusters that maximize the quality of the given clusters. This quality measure is calculated by a *sum of squared deviation (SSD)* metric which optimizes for dispersing the values of a given attribute vector across the newly created clusters. As an example, if the attribute vector used to compare regions includes population count and average income. The partitioning algorithm would attempt to create clusters that have the smallest amount of income and population variance across the clusters. Computing the SSD is achieved by the following equations:

Equation 2: SKATER optimization function

$$f_1(S_l^T) = SSD_T - (SSD_{Ta} + SSD_{Tb})$$

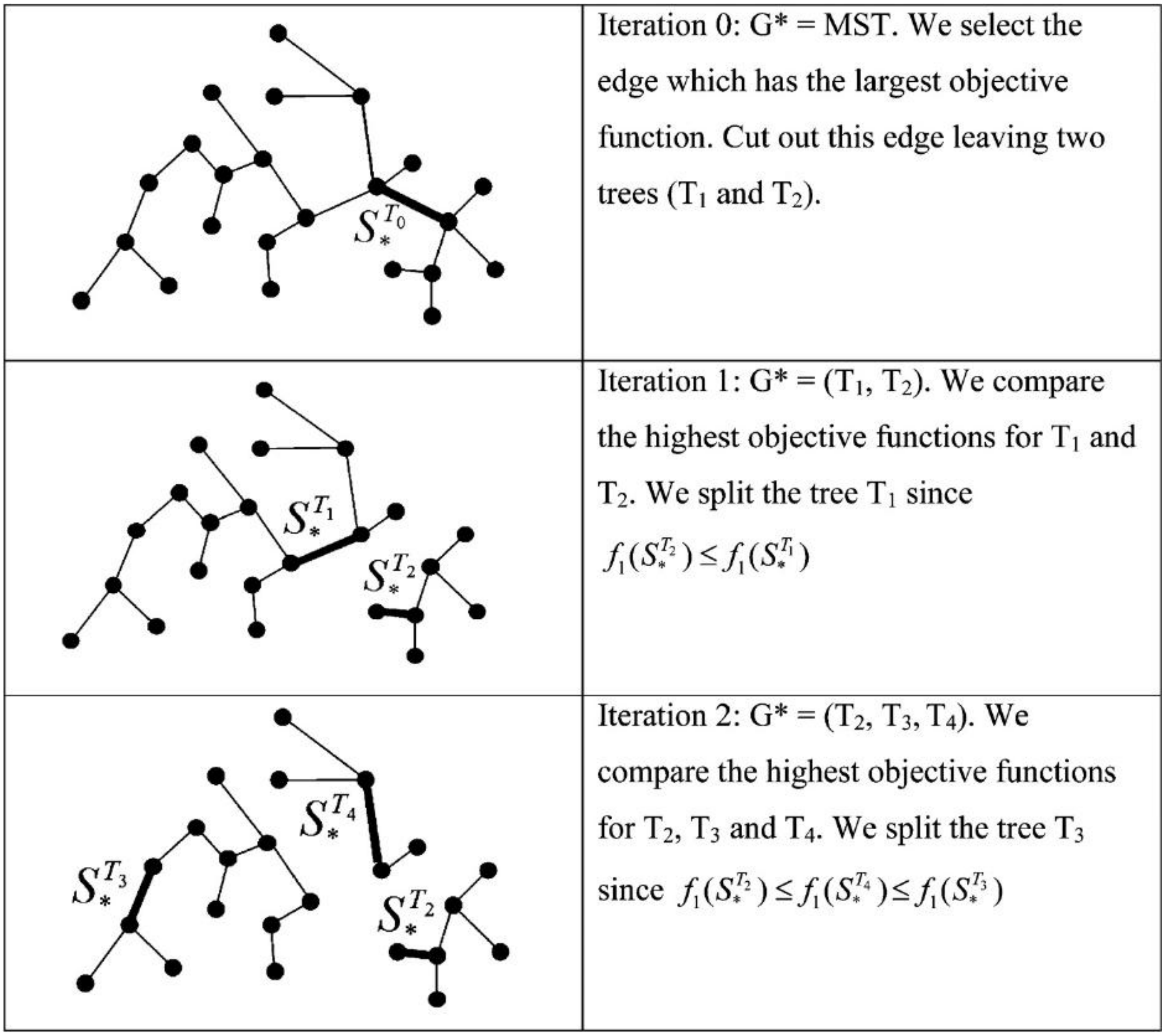
Equation 3: SSD calculation

$$SSD_k = \sum_{j=1}^{m}\sum_{i=1}^{n_k}(x_{ij} - \overline{x_j})^2$$

Once the algorithm has built the initial MST, it then proceeds through the following algorithm:

1. Identify the edge in the tree which has the highest objective function as determined by Equation 2. This is calculated for each edge by assuming that if the given edge were removed from the tree, the result would be two separate trees that would each have their SSD calculated by Equation 3.
2. For the edge with the highest result from Equation 2, remove it, which partitions the MST into two subtrees.
3. For all of the resulting trees, select the one with the highest result from Equation 3 and repeat the previous steps until the desired *stop state* is reached.

The *stop state* could be achieved by either setting a desired number of output regions (e.g. create 3 aggregated objects from the given input regions) or by creating a different objective function that stops partitioning the trees further, such as by setting a minimum or maximum population count for the given regions. This phase is illustrated in Figure 39, which begins with the MST from the previous phase and progressively partitions the trees until the desired output state is reached.



Iteration 0: G* = MST. We select the edge which has the largest objective function. Cut out this edge leaving two trees (T_1 and T_2).

Iteration 1: G* = (T_1, T_2). We compare the highest objective functions for T_1 and T_2. We split the tree T_1 since $f_1(S_*^{T_2}) \leq f_1(S_*^{T_1})$

Iteration 2: G* = (T_2, T_3, T_4). We compare the highest objective functions for T_2, T_3 and T_4. We split the tree T_3 since $f_1(S_*^{T_2}) \leq f_1(S_*^{T_4}) \leq f_1(S_*^{T_3})$

Figure 39: Partitioning the MST for *phase 2* of SKATER (Reproduced from [181]).

One the MST has been produced from *phase 1* of SKATER (shown in Figure 38), partitioning the MST is shown in this figure. This process recursively divides the initial tree into a number of sub-trees until either the desired number of trees (which correspond to regions) has been produced, or some other *stop state* is reached.

This partitioning process is *NP-hard* as it leads to a combinatorial explosion of potential values as the input size grows. To compensate for this, the authors propose a number of optimization heuristics, which significantly reduce the solution search space. The details of these optimizations are beyond the scope of this dissertation, but can be found in [SKATER, 2002]. The key piece of information is that these optimizations reduce the computational load without effecting the algorithm accuracy, or significantly altering the flow of execution.

Compared to AZP, SKATER has shown to be both quicker, as well as to generate clusters of roughly the same level of quality (see [181] for more details on the comparison). It also has the added benefit of being increasingly parallelizable. Each subtree, created in Phase 2, is independent of any other tree (as are the children of those trees) and thus can be processed in parallel. This means, that the parallelizability of the algorithm increases, as it approaches the optimum solution.

One limitation of this approach is that, for some cost functions, it is possible for two edges to have the same cost value, and thus arrive at a non-unique MST. Though the authors specify that given the types of data commonly dealt with (socio-economic data from census records) this is unlikely to happen. If this does occur, it is possible to select a different starting point and arrive at different (and unique solution) or expand the attribute vector to account for more input data properties. But this is something that may be a surprise to the user, especially if they are working with a small attribute vector in which the values of the data properties are likely to be the same (e.g. birth rates or net migration estimates for sparsely populated counties).

6.2.3 *REDCAP*

Rather than defining a single algorithm, as is the case for AZP and SKATER, REDCAP is actually a suite of different clustering techniques that aim to overcome the limitations with the SKATER approach. The first is that when SKATER constructs the spatial contiguity graph, is assumes a static state of the world. Meaning, that it cannot account for objects that may become adjacent to each other if they later belong to two adjacent clusters. The second, is that SKATER cannot guarantee that the data property values of the BSUs within a given region are similar to each other. Instead, it calculates the similarity of a given region to other regions in the proposed solution, which means that some BSUs within a given region may have significantly different values from other region members, even if their combined values are similar to other proposed regions. This is referred to as a *chaining* problem and is a well-known limitation of minimum spanning trees [182].

Algorithm Design

In order to address these limitations, the REDCAP family of algorithms was developed [183]. This family is a collection of three clustering methods and two different contiguity constraints, resulting in six different methods for regionalizing a given input dataset. Each of these combinations provides a different way of computing the similarity of two proposed regions. By comparing a subset of the members of each proposed region through a user provided *dissimilarity* function that can be used to determine the determine how similar two BSUs are to each other, in the same manner as the cost function used by SKATER.

Before computing the dissimilarity, REDCAP has to determine which BSUs in each region should be used in the calculation. The simplest approach is known as *first-order linkage* which means that only regions that are spatially adjacent to each other are used in the dissimilarity calculation. The more complex approach is known as *full-order linkage* which calculates the dissimilarity between a given BSU and all other BSUs, not including BSUs in the same region.

These two approaches are illustrated in Figure 40; for the *first-order* linkage, the dissimilarity function would be used to calculate the differences between spatially adjacent BSUs *only*. Meaning, comparing B -> H and E -> {H, F}. For the *full-order* linking, all edges are used, meaning A -> {H, G, F}, C -> {H, G, F}, D -> {H, G, F}, E -> {H, G, F}, B -> {H, G, F}.

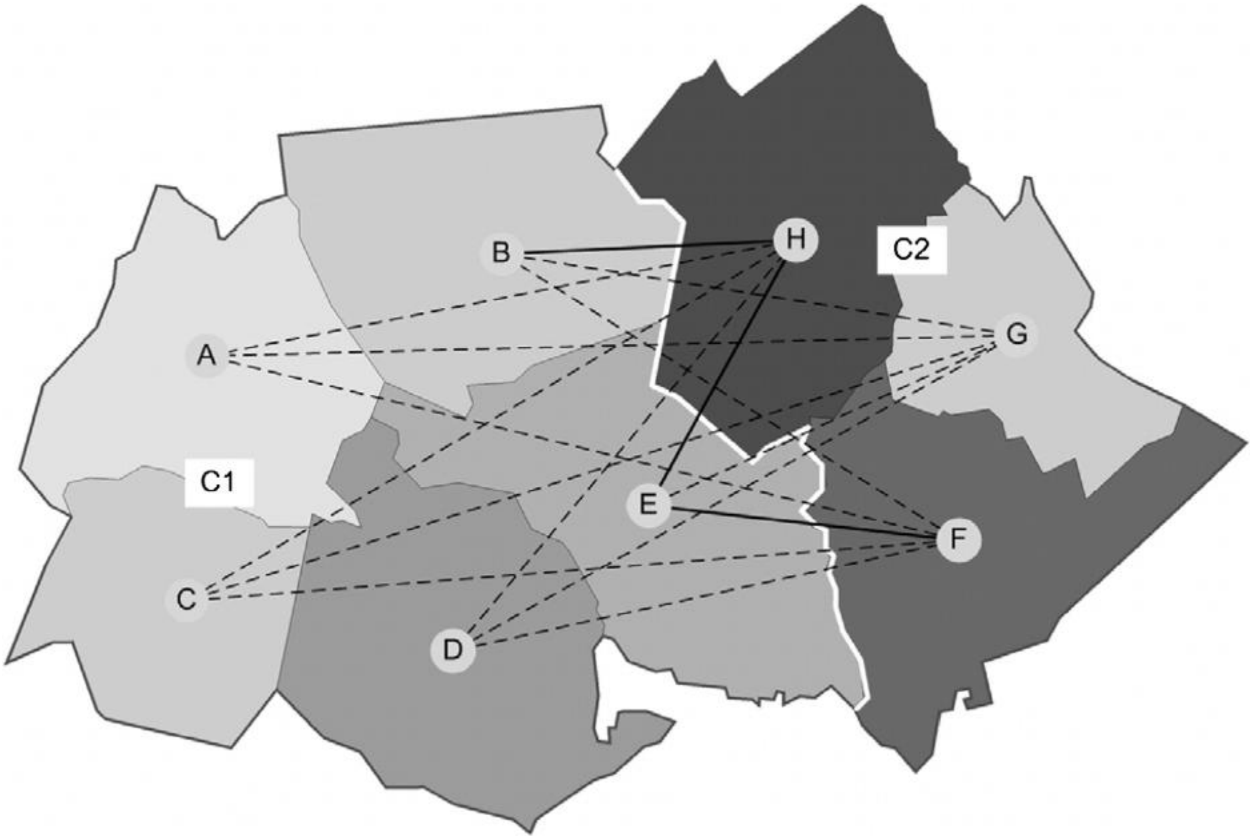


Figure 40: REDCAP linkages between two clusters (Reproduced from [183]).

This figure shows the two different linking methods supported by REDCAP. For *first*-order linking (shown via the solid lines), only immediately adjunct regions are linked together. Meaning, comparing B -> H and E -> {H, F}. For the *full-order* linking (shown via the dashed lines), all edges are linked together, meaning A -> {H, G, F}, C -> {H, G, F}, D -> {H, G, F}, E -> {H, G, F}, B -> {H, G, F}.

For either of the two linkage approaches described above, REDCAP defines 3 clustering methods that determine how the edge lengths are computed. The first clustering method is *single linkage clustering (SLK)* which computes the dissimilarity of two regions based on the closest pair of data points from each region. This is illustrated in Equation 4 and is conceptually identical to the SKATER algorithm. *L* and *M* represent two regions being compared, while d_u and d_v represent the data point between compared between the two regions.

Equation 4: SLK algorithm

$$d_{SLK}(L, M) = min_{u \in L, v \in M}(d_{uv})$$

The second clustering approach, known as *average linkage clustering (ALK)* (shown in Equation 5), defines dissimilarity as the average distance between region data points.

Equation 5: ALK algorithm

$$d_{ALK}(L, M) = \frac{1}{|L||M|} \sum_{u \in L} \sum_{v \in M} (\, d_{uv})$$

The final clustering method (shown in Equation 6) is known as *complete linkage clustering (CLK)* which is similar to the ALK approach but uses the maximum distance between the data points of two BSUs to compute the dissimilarity.

Equation 6: CLK algorithm

$$d_{CLK}(L,M) = max_{u \in L, v \in M}(d_{uv})$$

While there are differences between the various clustering methods, they are conceptually similar enough that it is only necessary to describe one of the approaches, in order to understand how the entire REDCAP family functions. For the remainder of this section, we will describe the *Full-Order-ALK* approach, as it has been shown to produce the highest quality regions (along with *Full-Order-CLK)* and provides the most complete implementation illustration.

Like SKATER, this algorithm features two distinct phases, building the initial spatially contiguous tree, and partitioning the tree to achieve the desired number of output regions.

Phase 1: Build the spatially contiguous tree

The algorithm begins with the spatially contiguous graph, similar to the process described by SKATER. In this graph, each BSU begins as a member of its own region and is represented as the *nodes* of the graph. The *edges* are the links between spatially adjacent objects and it is these edges that are of interest to the algorithm. Each edge is assigned a specific *length* which corresponds to the closeness of attribute vectors between the two BSUs as determined by the chosen clustering method.

Beginning with this graph, it first computes the length of each edge (meaning how similar two spatially adjacent BSUs are to each other, in terms of the data properties being used in the comparison) for the entirety of the graph and sorts the edges into a list in ascending order (*E)*[43]. List *(T)* maintains the final output list of all the necessary edges.

It then proceeds through the following algorithm, until all the BSUs have been assigned to the tree starting with the shortest (most similar) edge between two objects.

1. Determine if the two objects linked by the current edge meet the following criteria:
    a. Both are members of two different clusters (*m* and *l)*.
    b. The two clusters are spatially adjacent.
    c. The length of the edge is greater than or equal to the current average edge length between clusters *m* and *l*.
2. If any of the above conditions are not met, repeat Step 1 with the next shortest edge in list *E*.
3. Otherwise, find the shortest edge in *E* which connects *m* and *l*.
4. Add the edge to *T* and merge cluster *m* into *l*.
5. For each current cluster *c* (except the two clusters in this step) update the average distance between *c* and the new cluster *ml*, using the following algorithm:

[43] As a performance optimization, *E* is actually implemented as a binary search and sorting tree but is described here as a list for simplicity.

$$avgDist(c, ml) = \frac{[avgDist(c, l) * numEdges(c, l) + avgDist(c, m) * numEdges(c, m)]}{[numEdges(c, l) + numEdges(c, m)]}$$

6. Remove any edges that link cluster *c* with *m* or *l* from list *E*.
7. Add a new edge that links cluster *c* with *ml* into *E* with the updated average distance computed in Step 5.
8. Repeat Step 1 with the next shortest edge in list *E*.

The output of Phase 1 is a spatially contiguous tree of clustered BSUs, with each member linked to the adjacent object most similar to itself (based on the average distance of the attribute vector). Once generated, the next step is to partition the tree into optimal clusters.

Phase 2: Partition the spatially contiguous tree

The partitioning phase proceeds in largely the same manner as the SKATER algorithm. Starting with the initial tree, it recursively partitions the tree into a given number of output regions which exhibit the most homogenous distribution of a given set of data properties between the various regions. This is determined by Equation 7, in which *H(R)* is determined by the *SSD* given in Equation 3

Equation 7: Calculating the change in heterogeneity between two given sub-trees

$$h_g^* = \max\left(H(R) - H(R_a) - H(R_b)\right)$$

Equation 8: Overall heterogeneity of a proposed regionalization solution

$$H_k = \sum_{j=1}^{k} H(R_j)$$

Beginning with the single cluster produced in Phase 1, the partitioning algorithm proceeds through the following steps to determine how to partition a given tree.

1. For the given tree *T*, compute the heterogeneity of all its members using Equation 3.
2. Create a map *M* of edges in tree *T* with an associated change in heterogeneity.
3. For each edge *E* in *M* do the following:
    a. Temporarily remove *E* from *T* thus creating two sub-trees *Ta* and *Tb*.
    b. Compute the heterogeneity for *Ta* and *Tb*.
    c. Determine the amount of decrease in heterogeneity between *T* and *Ta* and *Tb* using Equation 7. Associate this value with *E* in *M*.
4. Determine the optimum split of *T* into *Ta* and *Tb* by removing edge *E* which has the largest decrease in heterogeneity, as selected from *M*.
5. Continue until the desired number of regions has been reached.

The output of REDCAP is a set of optimal regions, which have the highest degree of homogeneity of the attribute vector, both between the regions, as well as between the members of a given region. When compared to SKATER, REDCAP provides improved performance and spatial accuracy, due to the ability to recompute the spatial contiguity graph as the trees change over time. It should be noted that a modification to REDCAP was published in 2011, which provided additional improvements to account for potential unevenness of data distribution for smaller spatial areas [184]. The original REDCAP algorithm is described here due to its implementation simplicity over the improved method, as well as the fact that the updated algorithm only differs in the metric used to compute the edge weights in Phase 1.

6.3 Integrating REDCAP with Trestle

Given that we have now described three major approaches to addressing the regionalization challenge, it is time to evaluate how these algorithms might make use of the Trestle data model and management application.

In general, there are two major strategies for integration. The first, is to use Trestle simply to retrieve the necessary input data, such as the input BSUs for AZP, or the spatial adjacency graph for SKATER and REDCAP. The second, is to modify the existing algorithms to take advantage of the additional features in Trestle to simplify the algorithm design and implementation. The first part of this section will detail how Trestle can be used to support gathering the initial dataset, while the second part will describe strategies for improving existing algorithms through extended use of Trestle.

6.3.1 *Gathering the initial data*

Each of the algorithms are designed to be executed on an initial dataset, which can be gathered via a single retrieval from the underlying datastore. For AZP, this involves gathering the BSU boundaries and randomly generating initial regions to then partition. For SKATER and REDCAP, both start with a spatial adjacency graph, which is then processed by the respective algorithms.

It is this initial dataset that Trestle can help optimize, in two major ways. The first is in gathering the correct information for the temporal period being aggregated. Each algorithm assumes a static map state which requires valid data for a specific point in time. For Trestle, this can be achieved through the use of Trestle_Objects, which allows data from multiple time points to be combined into unified spatial objects (such as US counties which may have data for multiple years). This means, Trestle supports performing regionalization at multiple time points, without requiring manually changing input datasets. The second optimization is through easily generating the spatial adjacency graph. Since Trestle provides the ability to denote relationships between Trestle_Objects (such as the fact that two counties are spatially adjacent) it can perform an initial analysis to build the spatial object graph and on subsequent queries return the graph, without requiring expensive spatial computations at query time. In addition, Trestle already maintains its data in a native graph layout, which means it optimally supports graph-based algorithms without any special modifications and in a performant manner. These types of approaches are an excellent fit for Trestle given that way that the underlying triple-store is optimized to traverse relationships between various nodes, without requiring extensive translation between table and graph data layouts. Existing modeling approaches are not optimized for these types of queries, which can difficult to perform in traditional relational databases, especially as data size increases. Trestle does not suffer from this limitation.

To illustrate how these algorithms might make use of Trestle, we will show an example of performing the regionalization of a subset of US counties in the state of Washington, using data from August, 2013[44]. This initial starting dataset is shown in Figure 41. As a first step, we combined county data from the US Tiger Dataset into unified Trestle_Objects by joining the spatial boundaries with their corresponding ACS data properties for the years 2011 through 2016. This was accomplished through a custom merge process which combined the data on a year-by-year basis, with each year of data merged together by the Trestle management application. The source code for this process is shown in Appendix C. The output of the data loading process is a set of unified Trestle_Objects which contain the various data properties of a single county, for each year of data. This is illustrated in Figure 42 and the corresponding data definition, used by the Trestle application, is given in Code 3. More details on the data loading and merging process are given in Section 4.2.3.

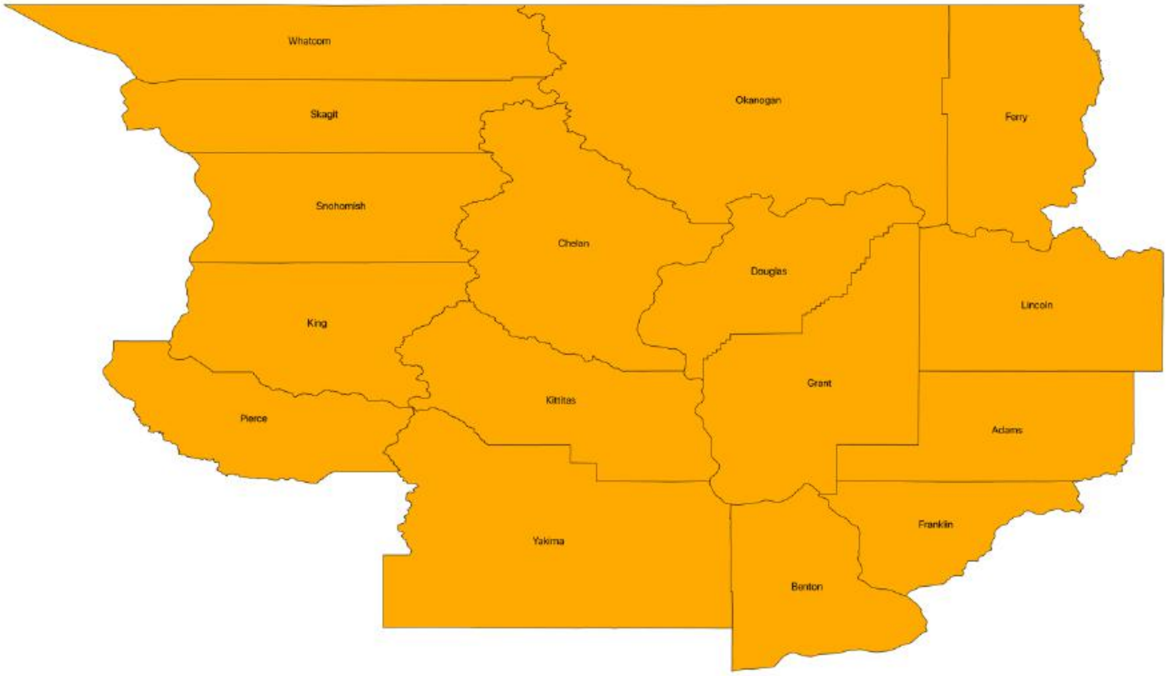


Figure 41: Selection of Washington counties in 2013 used for REDCAP integration example.

This map shows the initial counties used for the REDCAP/Trestle integration example. These counties are all found in Washington State, using the US TIGER dataset.

[44] Census estimates are updated at the mid-point of each year, meaning that a given population estimate is valid from July 1st of a given to year, to June 30th of the following year.

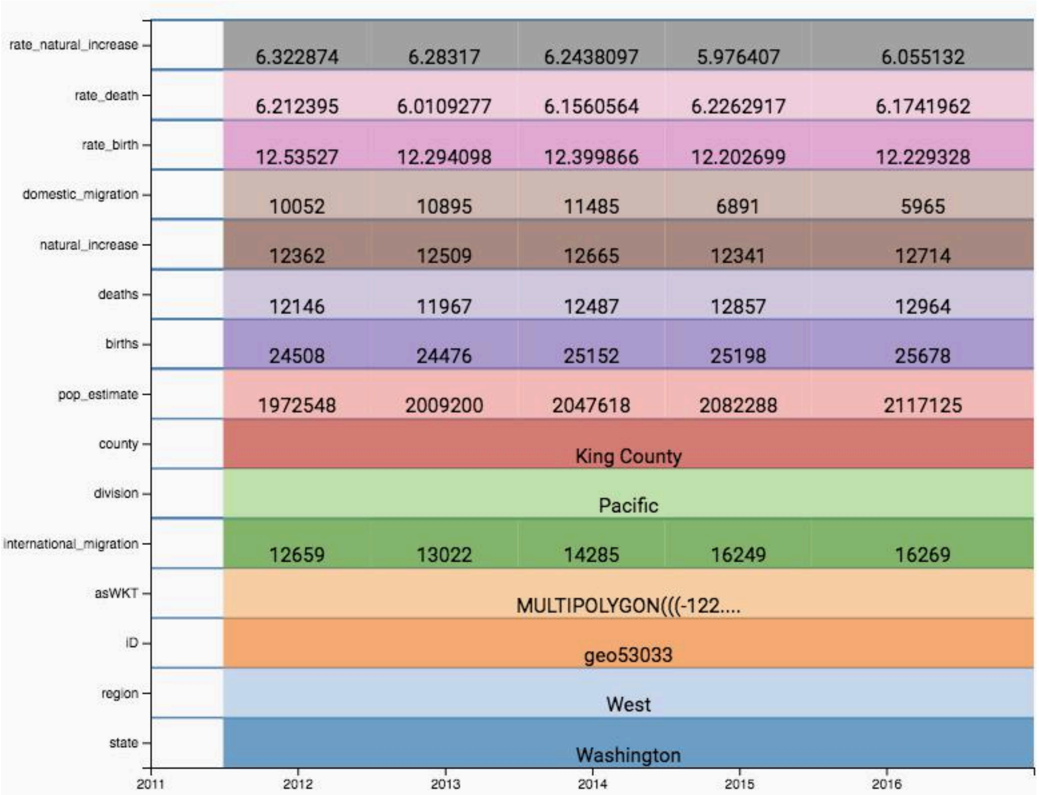


Figure 42: Data properties for King County, Washington, as they appear over time.

This figure shows the various data properties and values associated with *King County, Washington* from 2011 to 2016. The Y-axis shows the properties found in the ACS dataset, while the values of the properties are laid out on the X-axis, for the time interval for which they are valid.

```
@DatasetClass(name = "TigerCountyCensus")
public class TigerCountyObject {

    private final String geom;
    private final String geoid;
    private final String region;
    private final String division;
    private final String state;
    private final String county;
    private final int pop_estimate;
    private final int births;
    private final int deaths;
    private final int natural_increase;
    private final int international_migration;
    private final int domestic_migration;
    private final float rate_birth;
    private final float rate_death;
    private final float rate_natural_increase;
    private final LocalDate record_start_date;

    public TigerCountyObject(String geoid, String geom, String region, String
division,
                             String state, String county, int pop_estimate, int
births,
                             int deaths, int natural_increase, int
international_migration,
                             int domestic_migration, float rate_birth, float
rate_death,
                             float rate_natural_increase, LocalDate start_date)
{
        this.geoid = geoid;
        this.geom = geom;
        this.region = region;
        this.division = division;
        this.state = state;
        this.county = county;
        this.pop_estimate = pop_estimate;
        this.births = births;
        this.deaths = deaths;
        this.natural_increase = natural_increase;
        this.international_migration = international_migration;
        this.domestic_migration = domestic_migration;
        this.rate_birth = rate_birth;
        this.rate_death = rate_death;
        this.rate_natural_increase = rate_natural_increase;
        this.record_start_date = start_date;
    }

    @Spatial(projection = 4269)
    public String getGeom() {
        return geom;
    }

    @IndividualIdentifier
    public String getGeoid() { return geoid; }

    public String getRegion() {
        return region;
    }
    public String getDivision() {
        return division;
    }

    public String getState() {
        return state;
    }

    public String getCounty() {
        return county;
    }

    public int getPop_estimate() {
        return pop_estimate;
    }

    public int getBirths() {
        return births;
    }

    public int getDeaths() {
        return deaths;
    }

    public int getNatural_increase() {
        return natural_increase;
    }

    public int getInternational_migration() {
        return international_migration;
    }

    public int getDomestic_migration() {
        return domestic_migration;
    }

    public float getRate_birth() {
        return rate_birth;
    }

    public float getRate_death() {
        return rate_death;
    }

    public float getRate_natural_increase() {
        return rate_natural_increase;
    }

    @StartTemporal(name = "start_date")
    public LocalDate getRecord_start_date() {
        return record_start_date;
    }
}
```

Code 3: Tiger County dataset definition.

This code sample displays the *data definition* required by Trestle in order to the TIGER and ACS datasets. It follows the pattern shown in Code 2, but customized for this use case.

Once the data has been loaded, the next step is to build the spatial adjacency graph which will be the input for the appropriate algorithm. This can be accomplished via a single method in the Trestle API, which is shown in Code 4 along with the annotated parameters. Parameter 1 defines the type of object being returned from the management application. This is standard for all Trestle API calls and tells the management application how to map the data properties from the underlying graph layout, into a usable object format. One benefit of this approach is that all data returned by Trestle is in a standard Java object format which is immediately usable by other applications and algorithms[45], as shown in Code 3. The second parameter specifies which object to begin which when building the adjacency graph. The next two parameters define the shape of the graph itself; parameter 3 defines which metric to use when computing the weight of the various edges (also called the *cost* in SKATER and *length* in REDCAP), this function is defined by the user and may contain any number of data properties or stateful computations. The key point is that the entire Trestle_Object for the given county is available to be used in the computation, not just a single data property. This is shown in the *CountyCompute*e class in Code 4. Parameter 4, the *CountyFilter* class, specifies a user defined function for determining which objects should be included in the spatial graph. In this example, we restrict the output to include only the subset of counties shown in Figure 41. Parameters 5 and 6 specify the temporal restriction for when to execute the adjacency calculation. This is part of Trestle's *bi-temporal* support and specifies the valid time and database time to use for graph construction. In this example, we are running the computation for data valid on August 1st, 2013, and the most recent version of the facts known to the database (as denoted by the *null* value for parameter 6). What is important to note about this specific API method, is that it is possible to recompute the adjacency graph for any specific point in time, simply by changing the value of Parameter 5.

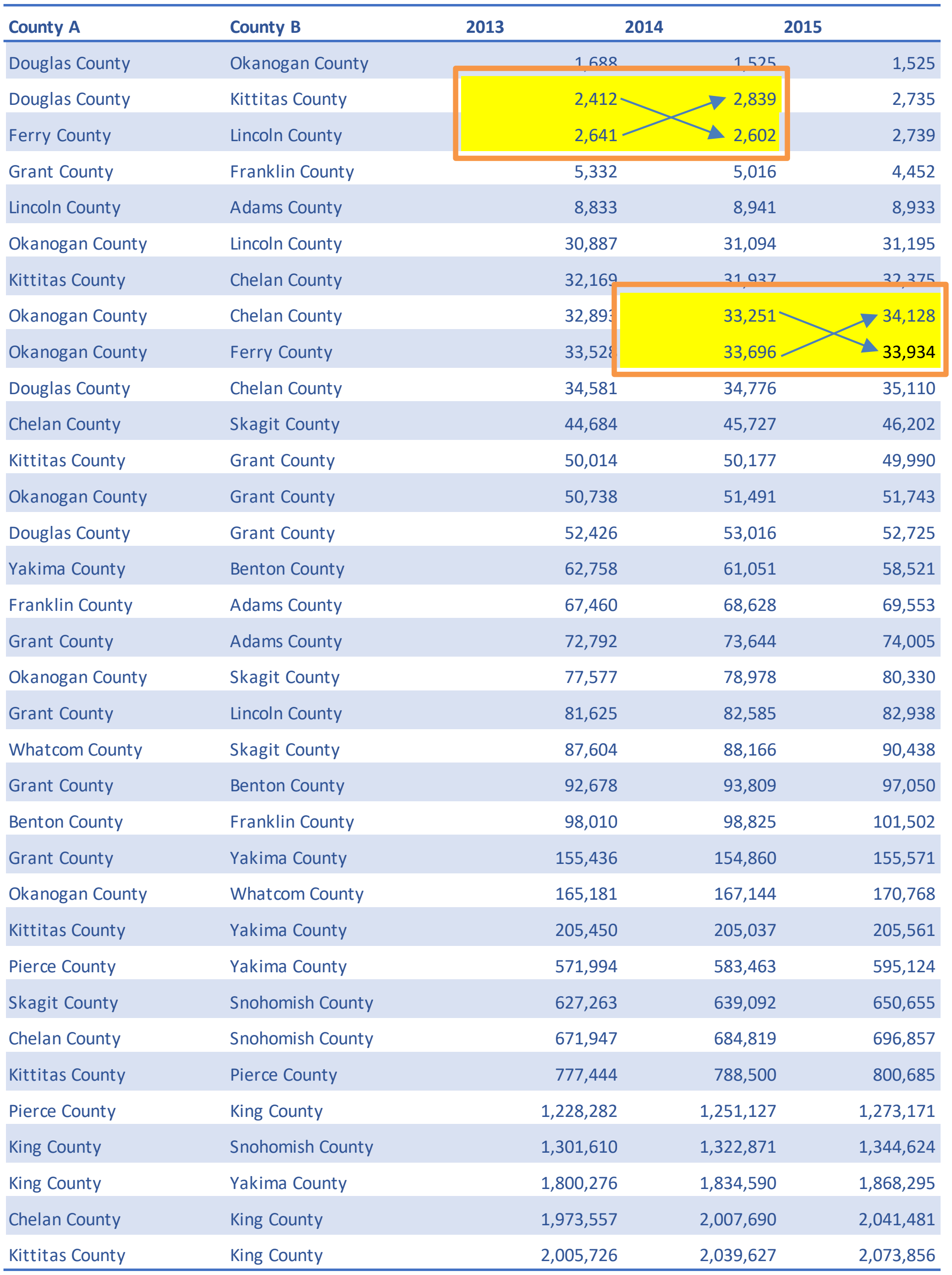
[45] It is also possible to extend Trestle to support object models in other programming languages. Currently, Trestle also supports returning data in the standard JSON format for use in web applications.

```
counties = new HashMap<>();
counties.put("Douglas County", 53017);
counties.put("Chelan County", 53007);
counties.put("Okanogan County", 53047);
counties.put("Kittitas County", 53037);
counties.put("Grant County", 53025);
counties.put("Whatcom County", 53073);
counties.put("Skagit County", 53057);
counties.put("Snohomish County", 53061);
counties.put("King County", 53033);
counties.put("Pierce County", 53053);
counties.put("Yakima County", 53077);
counties.put("Benton County", 53005);
counties.put("Franklin County", 53021);
counties.put("Adams County", 53001);
counties.put("Lincoln County", 53043);
counties.put("Ferry County", 53019);


        final AggregationEngine.AdjacencyGraph<TigerCountyObject, Integer> county_graph =
                reasoner.buildSpatialGraph(
//                        1. Type of the data to return from Trestle
                        TigerCountyObject.class,
//                        2. Initial starting point for Graph generation
                        counties.get("Douglas County").toString(),
//                        3. User defined class to compute the edge lengths.
                        new CountyCompute(),
//                        4. User defined class to determine which counties to
include in the graph generation
                        new CountyFilter(counties),
//                        5. Valid time to perform graph generation
                        VALID_AT,
//                        6. Database to perform graph generation
                        null);

    public static class CountyCompute implements Computable<TigerCountyObject,
TigerCountyObject, Integer> {

        @Override
        public Integer compute(TigerCountyObject nodeA, TigerCountyObject nodeB) {
//           5. Calculate the edge length based on the absolute difference in population
count between the two counties.
            return FastMath.abs(nodeA.getPop_estimate() -  nodeB.getPop_estimate());
        }
    }

    public static class CountyFilter implements Filterable<TigerCountyObject> {

        private final Map<String, Integer> counties;

        CountyFilter(Map<String, Integer> counties) {
//           5. Only include a specific subset of counties in the spatial graph
            this.counties = counties;
        }

        @Override
        public boolean filter(TigerCountyObject nodeA) {
            return this.counties.containsKey(nodeA.getCounty());
        }
    }
```

Code 4: Java code to generate spatial adjacency graph for regionalization using Trestle APIs.

This example shows the required code to generate the spatial adjacency graph using the Trestle APIs. More detail in given in the paragraph on page 144; but the required parameters are described in the code example itself.

The output of this API method is given in Table 20, which corresponds to the edges between all the nodes (counties) in the spatial graph, sorted by their difference in population count (the output of the *CountyCompute* class) with duplicate edges removed. The table also shows the same output for the years 2014 and 2015. Even in this small dataset, it is possible to observe differences in the graph output, not only in terms of the actual change in population count between the counties, but also in the order of which the counties are sorted (differences in sort order are highlighted in yellow). Specifically, between 2013 and 2014, the edge between *Douglas County* and *Kittitas County* changes position with *Ferry County* and *Lincoln County*. Likewise, for 2014, the edge between *Okanogan County* and *Chelan County*, changes places with *Okanogan County* and *Ferry County*. While trivial, these changes have the potential to dramatically effect the regionalization output and help illustrate the importance of having the ability to explicitly control the temporal space in which algorithm executes. Once the initial graph has been generated, it can be passed to the appropriate regionalization algorithm for further use.



County A	County B	2013	2014	2015
Douglas County	Okanogan County	1,688	1,525	1,525
Douglas County	Kittitas County	2,412	2,839	2,735
Ferry County	Lincoln County	2,641	2,602	2,739
Grant County	Franklin County	5,332	5,016	4,452
Lincoln County	Adams County	8,833	8,941	8,933
Okanogan County	Lincoln County	30,887	31,094	31,195
Kittitas County	Chelan County	32,169	31,937	32,375
Okanogan County	Chelan County	32,893	33,251	34,128
Okanogan County	Ferry County	33,528	33,696	33,934
Douglas County	Chelan County	34,581	34,776	35,110
Chelan County	Skagit County	44,684	45,727	46,202
Kittitas County	Grant County	50,014	50,177	49,990
Okanogan County	Grant County	50,738	51,491	51,743
Douglas County	Grant County	52,426	53,016	52,725
Yakima County	Benton County	62,758	61,051	58,521
Franklin County	Adams County	67,460	68,628	69,553
Grant County	Adams County	72,792	73,644	74,005
Okanogan County	Skagit County	77,577	78,978	80,330
Grant County	Lincoln County	81,625	82,585	82,938
Whatcom County	Skagit County	87,604	88,166	90,438
Grant County	Benton County	92,678	93,809	97,050
Benton County	Franklin County	98,010	98,825	101,502
Grant County	Yakima County	155,436	154,860	155,571
Okanogan County	Whatcom County	165,181	167,144	170,768
Kittitas County	Yakima County	205,450	205,037	205,561
Pierce County	Yakima County	571,994	583,463	595,124
Skagit County	Snohomish County	627,263	639,092	650,655
Chelan County	Snohomish County	671,947	684,819	696,857
Kittitas County	Pierce County	777,444	788,500	800,685
Pierce County	King County	1,228,282	1,251,127	1,273,171
King County	Snohomish County	1,301,610	1,322,871	1,344,624
King County	Yakima County	1,800,276	1,834,590	1,868,295
Chelan County	King County	1,973,557	2,007,690	2,041,481
Kittitas County	King County	2,005,726	2,039,627	2,073,856

Table 20: Population difference between counties for the years 2013 through 2015

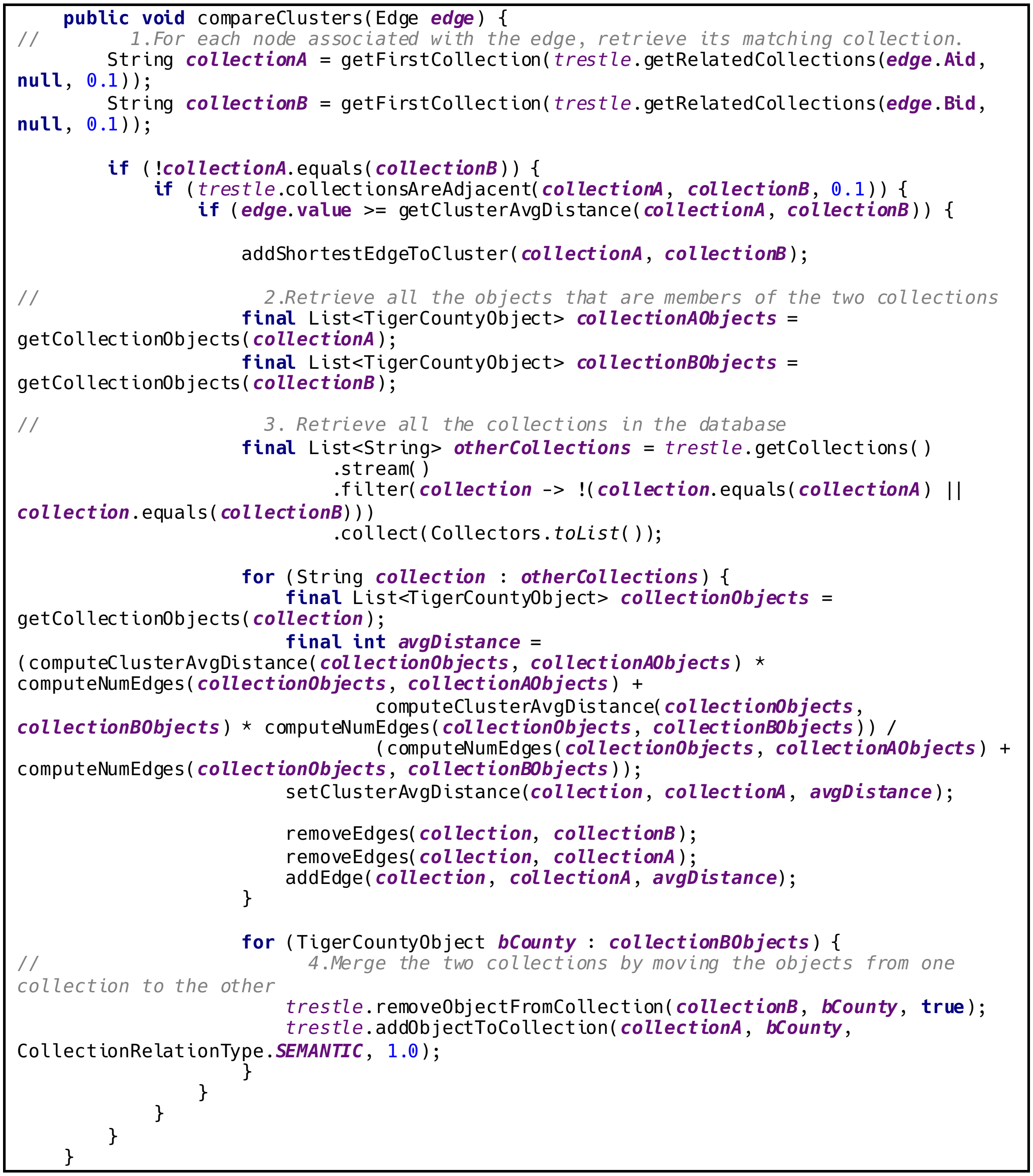
This table gives the output of the spatial graph computation (shown in Code 4). The ordering of the counties is shown based on the graph generation using data 2013. The graph generation was then performed for years 2014 and 2015, which clearly shows two instances (highlighted and boxes) where the ordering of the output would change based on the year of data used. It should be noted that this change is visible even in a small dataset, such as the one used in this example.

6.3.2 *Refactoring existing algorithms*

Beyond the initial support for retrieving the initial dataset, Trestle also provides some additional functionality which can be of use when developing stateful spatial algorithms. Meaning, those types of algorithms which need to progressively modify, and recall, the spatial state of the world during execution.

Both SKATER and REDCAP are designed to maintain their own internal state during execution. They do so by maintaining lists and indexes which track which BSUs are associated with which clusters, at specific points in time. While this works effectively, and means the algorithms only need to make a single request to the database, it also means that implementers and algorithm developers carry an extra burden of bookkeeping and state management in order to design and implement more complex spatial algorithms.

To help reduce this burden, Trestle provides some additional features that could be leveraged, the most useful of which, is the *Trestle_Collections* functionality. Described in Section 4.2.2.3, Trestle_Collections are associations of Trestle_Objects which have some expressed relationship (spatial, temporal, and semantic). In this example, collections can be used in Step 2 of REDCAP to simplify the tracking of adjacent objects and building up temporary clusters of objects, which can be used to generate the final regions. An example of modifying REDCAP to utilize the collections feature is given in Code 5, which represents a refactoring of Step 2 of the REDCAP algorithm. This example method would be called for each edge in the spatial graph. What is important to note is that the refactored design makes continual use of Trestle to build, update, and remove objects from collections as the algorithm runs. The benefit of this is that REDCAP itself is no longer required to track which objects are in which temporary clusters and is relieved of the burden of maintaining and updating the spatial adjacency graph. It should be noted that REDCAP still maintains its own internal data structure, which tracks the edges that link the various nodes.



```
    public void compareClusters(Edge edge) {
//        1.For each node associated with the edge, retrieve its matching collection.
        String collectionA = getFirstCollection(trestle.getRelatedCollections(edge.Aid,
null, 0.1));
        String collectionB = getFirstCollection(trestle.getRelatedCollections(edge.Bid,
null, 0.1));

        if (!collectionA.equals(collectionB)) {
            if (trestle.collectionsAreAdjacent(collectionA, collectionB, 0.1)) {
                if (edge.value >= getClusterAvgDistance(collectionA, collectionB)) {

                    addShortestEdgeToCluster(collectionA, collectionB);

//                    2.Retrieve all the objects that are members of the two collections
                    final List<TigerCountyObject> collectionAObjects =
getCollectionObjects(collectionA);
                    final List<TigerCountyObject> collectionBObjects =
getCollectionObjects(collectionB);

//                    3. Retrieve all the collections in the database
                    final List<String> otherCollections = trestle.getCollections()
                            .stream()
                            .filter(collection -> !(collection.equals(collectionA) ||
collection.equals(collectionB)))
                            .collect(Collectors.toList());

                    for (String collection : otherCollections) {
                        final List<TigerCountyObject> collectionObjects =
getCollectionObjects(collection);
                        final int avgDistance =
(computeClusterAvgDistance(collectionObjects, collectionAObjects) *
computeNumEdges(collectionObjects, collectionAObjects) +
                                computeClusterAvgDistance(collectionObjects,
collectionBObjects) * computeNumEdges(collectionObjects, collectionBObjects)) /
                                (computeNumEdges(collectionObjects, collectionAObjects) +
computeNumEdges(collectionObjects, collectionBObjects));
                        setClusterAvgDistance(collection, collectionA, avgDistance);

                        removeEdges(collection, collectionB);
                        removeEdges(collection, collectionA);
                        addEdge(collection, collectionA, avgDistance);
                    }

                    for (TigerCountyObject bCounty : collectionBObjects) {
//                        4.Merge the two collections by moving the objects from one
collection to the other
                        trestle.removeObjectFromCollection(collectionB, bCounty, true);
                        trestle.addObjectToCollection(collectionA, bCounty,
CollectionRelationType.SEMANTIC, 1.0);
                    }
                }
            }
        }
    }
```

Code 5: REDCAP phase 2, translated to use *Trestle_Collections.*

This code example shows how *phase 2* of the REDCAP algorithm (given on page 138) might be rewritten to take advantage of the Collections feature in Trestle. The major components of this code example are given as comments within the code itself (denoted by the // at the start of the line).

6.4 FUTURE WORK

While the previous sections described the process for integrating Trestle with existing regionalization algorithms, and the two primary ways of doing so, there are still a number of ways by which integration could be further improved. Not only to rely more heavily on the capabilities of Trestle, and thus reduce the complexity of the algorithm designs, but also to add the ability to perform even more complex analyses which are either extremely difficult to perform with existing GIS applications, or simply impossible. Both of these improvements will be the focus of this section.

6.4.1 *Improving Collections*

As demonstrated in Section 6.3.2, Trestle's collections feature can greatly reduce the complexity of spatio-temporal algorithms by provided a consistent way of managing groupings of related ST-Objects, even in a temporary fashion. In Chapter 5, Trestle_Collections were used to group spatially overlapping boundaries, before they were evaluated by the spatial union algorithm. In this chapter, we have shown that both the SKATER and REDCAP algorithms can make use of collections either during the tree building, or partitioning phase. Unfortunately, REDCAP is unable to fully take advantage of the collections feature due to some current limitations. As shown in Figure 43, Trestle_Collections are currently designed to link a given Trestle_Object to a Trestle_Collection through a Trestle_Relation object. This allows each object to be associated a collection along with some additional context. Unfortunately, this reduces its utility when associations need to be tracked *between* Trestle_Objects, such as how REDCAP needs to maintain the various edge lengths between nodes, in order to build the initial tree. In other words, REDCAP needs the ability to create temporary *sub-graphs* of related objects, which it can then traverse to determine the correct order of merging the various nodes. This is not currently possible in Trestle and is part of the reason why the example refactoring shown in Code 5 still requires some amount of internal state to be managed by the algorithm itself. A potential solution to this problem is shown in Figure 44 in which the Trestle_Relation object is modified to link two Trestle_Objects through some relationship (temporal, spatial or semantic) along with the required metadata (e.g. amount of spatial overlap, length of edge, etc.). This would allow Trestle_Collections to truly function as sub-graphs of related objects in which each Trestle_Object is a node on the graph and each *Trestle_Relation* serves as the edge between two nodes; greatly improving the flexibility and utility of the collections feature and further reducing the implementation complexity of their spatio-temporal algorithms. These improvements will be integrated in future versions of the Trestle application.

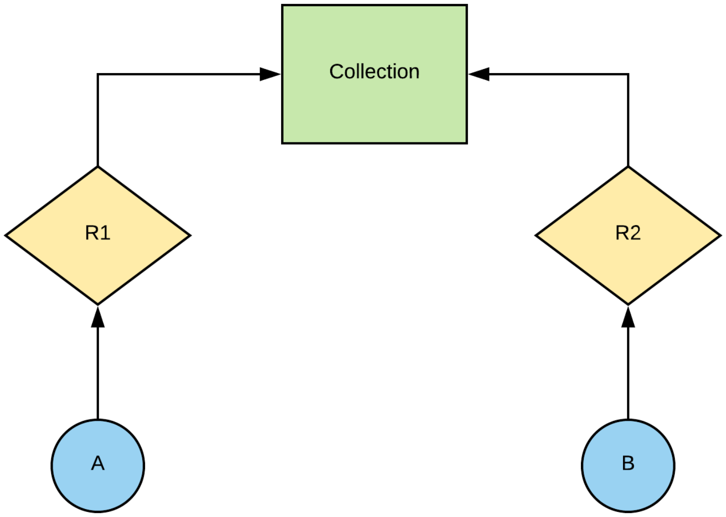


Figure 43: Current design of Trestle_Collections.

This figure shows the current design of the Collection feature, which only supports associating Trestle_Objects (blue circles) with a collection (green rectangle) through a specific *member_of* relationship (yellow diamonds). While useful, this limits Trestle's ability to support more complex associations that need to create collections from directly related objects.

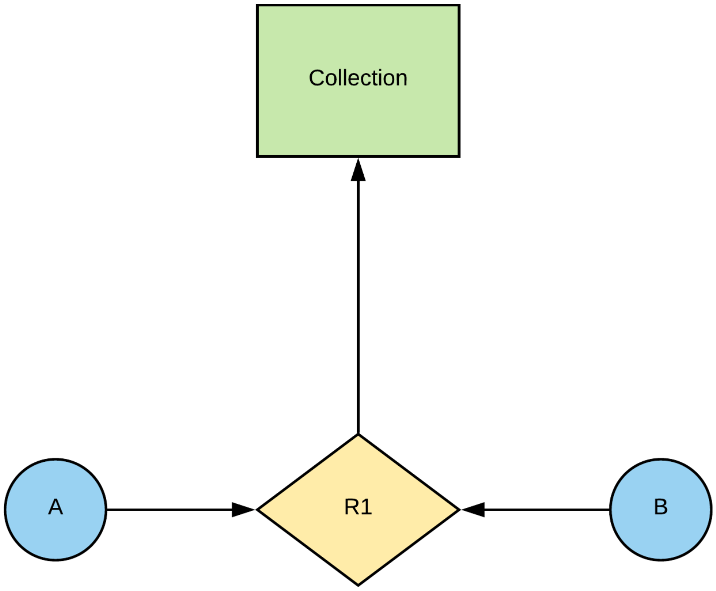


Figure 44: Future design of Trestle_Collections.

This figure shows the propsed future design of the Collection feature, which supports associating Trestle_Objects (blue circles) with a collection (green triangle) through a specific spatial, temporal, or semantic relationship (yellow diamond) between the two objects. This significantly improves Trestle's ability to support associations of related objects which *through* their relationships, need to be associated to a collection.

6.4.2 *Improve access to the SPARQL query language*

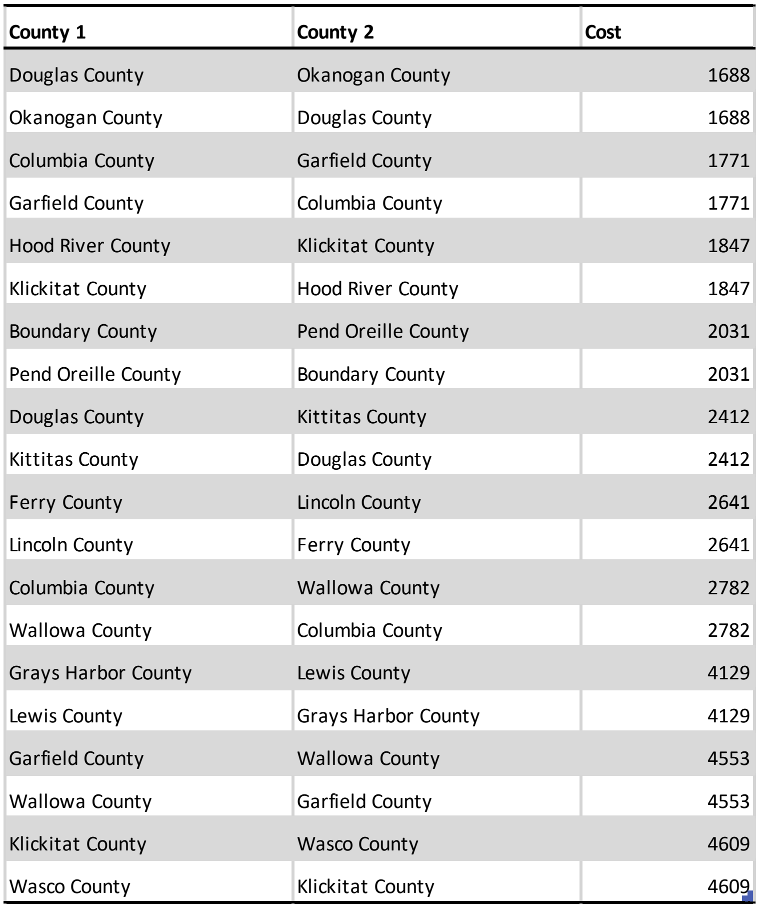
In the preceding section, we showed how Trestle can support the regionalization algorithms, through existing API methods, without requiring special code or data transformations. While this is tremendously useful, there are still situations in which users may wish to gain direct access to the underlying object graph and perform more complex queries which are difficult to encode in API specifications. Fortunately, Trestle supports a graph-oriented query language, the *SPARQL Protocol and RDF Query Language* (SPARQL), which can help address this need. Designed to support the types of queries common in semantic web applications, this language is oriented around graph traversal and querying in the same way that the *Structured Query Language* (SQL) is designed to query relational database tables. One drawback of SPARQL is that it features an entirely different query structure and interaction paradigm from traditional data query languages. This can present a significant burden to the end user in order to effectively utilize the graph data layout. In order to address this, Trestle provides a robust set of API features which largely removes the requirement of the user to make use of the graph features; however, internally, Trestle makes heavy use of SPARQL in order to implement some of its more complex features. That said it only provides limited support for exposing the raw query capabilities to the end user, due in part to the complexity in translating between the graph layout and the object layout found in programming languages such as Java. Indeed, a significant proportion of the code in the management application is devoted entirely to translating between these two data layouts. That being said, there are still situations in which directly leveraging the query facilities would significantly improve the design and implementation of spatio-temporal algorithms on top of Trestle.

As an example, Code 6 shows a simple implementation of the cost computation from Section 6.3.1 but done entirely in SPARQL and thus performed in the database itself, rather than the management application, as is the case with the API method. The downside of this approach is the added complexity foisted on the algorithm implementer, as they are required to manage the data translation from the underlying graph layout. The output of the SPARQL query in Code 6 is given in Table 21, which shows that a significant amount of work is required to retrieve usable ST-Objects from the raw graph results, such as removing duplicate edges. Future work in Trestle can provide additional functionality for improving the ease of use of the underlying query capabilities which may significantly benefit future researchers wishing to take full advantage of the underlying object graph.

```
BASE <http://nickrobison.com/dissertation/trestle.owl#>
PREFIX : <http://trestle.nickrobison.com/demonstration/>
PREFIX trestle: <http://nickrobison.com/dissertation/trestle.owl#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ogc: <http://www.opengis.net/ont/geosparql#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX orageo: <http://xmlns.oracle.com/rdf/geo/>
PREFIX ogcf: <http://www.opengis.net/def/function/geosparql/>
PREFIX geosparql: <http://www.opengis.net/ont/geosparql#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX xml: <http://www.w3.org/XML/1998/namespace>
PREFIX spatial: <http://www.jena.apache.org/spatial#>
SELECT distinct (?c as ?county1) (?mc as ?county2) (ABS(?o-?mo) as ?cost) WHERE {
   ?m rdf:type trestle:Trestle_Object .
   ?m trestle:has_fact ?f .
   ?f trestle:valid_from ?vf .
   ?f trestle:valid_to ?vt .
   ?f :pop_estimate ?o .
   ?m trestle:has_fact ?fc .
   ?fc :county ?c .
   ?m trestle:spatial_meets ?mt.
   ?mt trestle:has_fact ?mfc .
   ?mft :county ?mc .
   ?mt trestle:has_fact ?mf .
   ?mf trestle:valid_from ?vf .
   ?mf trestle:valid_to ?vt .
   ?mf :pop_estimate ?mo .
   FILTER(?vf>="2013-07-01T00:00:00Z"^^xsd:dateTime && ?vt<="2014-07-
01T00:00:00Z"^^xsd:dateTime) .
} ORDER BY (?cost) LIMIT 20
```

Code 6: SPARQL query for computing the spatial adjacency graph for *phase 1* of REDCAP.

This code sample illustrates the spatial adjacency graph computation (originally shown in Code 4) using the raw SPARQL query functionality, rather than the Java APIs. This illustrates the uniqueness of the SPARQL query language and the difficulty in translating between the graph and table paradigms. The output of this query is given in Table 21. More details on SPARQL can be found in [155].



County 1	County 2	Cost
Douglas County	Okanogan County	1688
Okanogan County	Douglas County	1688
Columbia County	Garfield County	1771
Garfield County	Columbia County	1771
Hood River County	Klickitat County	1847
Klickitat County	Hood River County	1847
Boundary County	Pend Oreille County	2031
Pend Oreille County	Boundary County	2031
Douglas County	Kittitas County	2412
Kittitas County	Douglas County	2412
Ferry County	Lincoln County	2641
Lincoln County	Ferry County	2641
Columbia County	Wallowa County	2782
Wallowa County	Columbia County	2782
Grays Harbor County	Lewis County	4129
Lewis County	Grays Harbor County	4129
Garfield County	Wallowa County	4553
Wallowa County	Garfield County	4553
Klickitat County	Wasco County	4609
Wasco County	Klickitat County	4609

Table 21: Results of SPARQL query from Code 6, using data from 2014.

6.4.3 *Extending regionalization to support indicator changes over time*

All three of the algorithms described in this chapter focus on re-organizing spatial objects based on their data properties at a static point in time. However, there exists much interest in regionalizing a dataset based on the change in indicators over time. For example, partitioning data based on the amount of change in population count over the past five years. While these types of analyses may be difficult to achieve in traditional GIS applications, for all the reasons described in the dissertation thus far, Trestle provides some unique facilities for evaluating how temporal queries might be included in existing regionalization algorithms.

Trestle already provides the ability to retrieve facts (the value of a given data property at a specific point in time) for a Trestle_Object at multiple time points. With that available data, it would be trivial to modify the cost function from REDCAP to consider data from multiple time points. However, introducing a temporal component requires that several invariants hold true for the data being utilized.

- *No unobserved events*: The analytical assumption is that two indicators at different time points reasonably describe the time period between those two events. Meaning, that if a region has a population count for 1990 and 1991, then it is assumed that at no temporal instant between those points did an event occur which dramatically changed the indicator value (e.g. the population count did not go to zero on June 9th, 1990). The same requirements must also hold true for the spatial area of a given region.

- *Spatially exclusivity*: Each spatial region describes an area that is not accounted for by any other region at the same time point.

Provided these assumptions are met, it is possible to use *Trestle* to compute optimization functions that are responsive to temporal change in values and potentially open up new avenues for generating regions which enable even richer and more complex spatio-temporal research than can currently be achieved with temporally static algorithms.

6.5 Conclusion

In conclusion, this chapter focused on the final component of the two-part evaluation outlined in Chapter 5. Through a focus on the needs of common algorithms designed to solve challenges within the field of regionalization, we successfully demonstrated the ability of Trestle to support complex spatio-temporal research through two major facilities. First, the ability of Trestle to create unified ST-Objects and retrieve temporally correct data for creating new regionalizations. Second, through Trestle's flexible object graph which enables creating new groupings of related objects, which reduces the necessity of algorithms to manage their own internal and temporary state. Finally, we outlined some future work within the Trestle project that can both improve existing algorithms, as well as enable new types of research that are currently limited by existing systems and processes.

Chapter 7. CONCLUSION

In this concluding chapter of the dissertation, we briefly summarize the major scientific contributions of the Trestle project, as well as some future directions for both novel research and engineering improvements.

7.1 Summary and Contributions

This section provides a summary of each chapter (excluding Chapter 1, which served as the introduction) as well as the overall scientific contributions. Each of these chapters served to help fulfill the research aims of this dissertation, which are:

- *Aim 1: Design and implement a unified method for automatically building and managing spatial objects from complex spatio-temporal data.*
- *Aim 2: Develop a graph-based approach for integrating spatial objects from historical records using a global dataset.*
- *Aim 3: Demonstrate using the query interface for solving a common challenge of generating custom research geographies for public health research.*

Chapter 2 provided an introduction into some specific challenges related to spatio-temporal research within the field of global health. Specifically, issues related to the spatial and temporal context of the geographic area being studied. Finally, we introduced a set of desiderata for any proposed spatio-temporal data model or GIS application.

Chapter 3 outlined four existing spatio-temporal data models and provided a technical evaluation framework for determining both the strengths of the existing methods, as well as potential areas for improvement. In addition, each data model was evaluated against the desiderata outlined in Chapter 2.

Chapter 4 described the graph-based spatio-temporal data model at the core of the Trestle system. This chapter detailed the spatio-temporal ontology and semantic reasoner approach that was chosen in order to address some of the limitations of existing data models. In addition, we evaluated the proposed data model against the criteria described in Chapter 3. Finally, we described a prototype data management application which enables usage of the data modeling approach. This chapter fulfills *Aim 1*.

Chapter 5 began the evaluation component of the dissertation by focusing on the first of two evaluation processes. The first process was to validate the ability of the data model and management application to effectively represent relationships between spatial objects. This was be accomplished through the design and implementation of a prototype algorithm to automatically identify spatial changes in a global administrative unit database through a *split/merge* event paradigm. A challenging issue in public health research. This chapter fulfills *Aim 2*.

Chapter 6 concluded the evaluation of the dissertation by testing the data model's ability to effectively represent the internal state of time-varying spatial objects. This was achieved by describing several common algorithms for generating custom research geographies, which require valid data at the point in time that the geographies are generated. This chapter then described how these types of algorithms can be effectively implemented within the data model, as well as detailed algorithm improvements that are enabled by access to the types of data contained in the Trestle data model. This chapter fulfills *Aim 3*.

In summary, this project has the following primary scientific contributions:

- A spatio-temporal data model designed to leverage graph layout and storage systems to represent complex temporal changes in spatial objects within a unified data model, that is optimized for complex public health research
- A data management and query application which abstracts the complexities of the underlying temporal logic to present a unified method of data interaction for the end user.
- A graph-based algorithm for analyzing spatial and temporal interactions between administrative districts and linking districts through time using a split/merge event paradigm.
- A temporally integrated dataset of a subset of county level districts in sub-Saharan Africa, useful for public health research.
- Demonstrated using Trestle to improve existing algorithms designed to aid researchers in developing custom geographies for specific research outcomes.

7.2 FUTURE WORK

While the data model and management application provide a solid foundation for spatio-temporal data management, there are still several avenues of future work that remain unexplored.

- *Derived and computed properties:*
  Currently, Trestle only supports facts which are directly expressed as members of a given dataset, but there are multiple scenarios in which a researcher may have need of a property which is not found in the dataset but can be computed from several other available properties. For example, a population density metric which relies on know the correct population count and spatial area covered at the point in time the metric is needed. Similar to the *computed column* feature found in relational databases, Trestle should have the ability to compute derived properties based on related fact values at a given temporal point. This would dramatically improve the ability of Trestle to model complex relationships and data properties but given the complexity of computing and normalizing various data properties, this requires careful research and design.

- *Raster data support:*
  At the present time, Trestle only supports working with *vector* (lines, points, polygons) datasets. Future work should attempt to provide a unified interaction model for working with both *vector* and *raster* (image) data. While there has been some work on integrating these two data formats [185], [186], it remains an open research problem. One of the major issues that prevents effective integration, is that raster images represent a given data property (such as elevation) as a continually varying *field* in which each image *pixel* corresponds to a value at that given point. This is different than how vector data presents a single value for a given spatial entity (e.g. only a single population count of a county is given). This means that raster data can represent how the population count of a county

varies throughout the region, while vector data can only present that single data property. This means the interaction paradigm varies between the two formats, both from the user perspective and the perspective of the data management application. For example, returning the population count of a county, in vector format, simply requires returning a single value, whereas the same operation for raster data requires summing up the value of each *pixel* in the image. In order to represent both formats, Trestle would need the ability to determine which format each spatial entity (or data indicator) is represented in, and which summary calculations need to be performed in order to return the correct value. This can be improved through the use of the *computed properties* feature outlined in the preceding bullet point.

An additional challenge in integrating raster data with Trestle is that most triple stores do not have native support for working with this type of data, either natively as geospatial data or more generically in binary formats[46]. This requires Trestle to implement additional features for manually storing, querying, and updating raster data, which may present a significant engineering challenge. Especially since the end goal is to provide truly integrated query support for both raster and vector data.

- *Multiple language and tool support:*
  As previously mentioned, public health researchers make use of a number of different programming languages (such as R, Python and Javascript) as well as existing spatial analysis tools such as QGIS, ArcMap and GRASS. Future work in Trestle will aim to further integrate with these tools and allow researchers to store and manage their research data in Trestle, while still performing their analysis within their existing tools and workflows. The current support for standard APIs and web protocols is an excellent starting point, but there is still much work to be done in order to further reduce the burden of implementation with existing tools and programming languages.

[46] A notable exception is the *Oracle* database, which provides robust support for most types of spatial data, alongside their optimized triple-store implementation.

Chapter 8. REFERENCES

[1] S. Pressfield, *The War of Art: Break Through the Blocks and Win Your Inner Creative Battle*. New York, N.Y.: Steven Pressfield, 2012.

[2] Raphael, "Saint George and the Dragon." National Gallery of Art, Washington, National Archives, Washington, DC, 1506.

[3] J. B. . Harley, "The Map and the Development of the History of Cartography," in *The History of Cartography*, J. B. . Harley and D. Woodward, Eds. Chicago, IL: The University of Chicago Press, 1987, pp. 1–42.

[4] J. Herring, M. J. Egenhofer, and A. U. Frank, "Using category theory to model GIS applications," in *Proceedings of the 4th International Symposium on Spatial Data Handling*, pp. 820–829.

[5] G. Valentine, "Living with difference: reflections on geographies of encounter," *Prog. Hum. Geogr.*, vol. 32, no. 3, pp. 323–337, Jun. 2008.

[6] T. Poiker, "Preface," *Cartogr. Geogr. Inf. Syst.*, vol. 22, no. 1, pp. 3–4, Jan. 1995.

[7] R. Tomlinson, "The Canada geographic information system," *Hist. Geogr. Inf. Syst. Perspect. from pioneers*, vol. 2132, 1998.

[8] M. F. Goodchild, "Reimagining the history of GIS," *Ann. GIS*, vol. 24, no. 1, pp. 1–8, Jan. 2018.

[9] S.-L. Shaw, "Guest editorial introduction: time geography – its past, present and future," *J. Transp. Geogr.*, vol. 23, pp. 1–4, Jul. 2012.

[10] H. J. Miller, "A Measurement Theory for Time Geography," *Geogr. Anal.*, vol. 37, no. 1, pp. 17–45, Jan. 2005.

[11] D. B. Richardson, "Real-Time Space–Time Integration in GIScience and Geography," *Ann. Assoc. Am. Geogr.*, vol. 103, no. 5, pp. 1062–1071, Sep. 2013.

[12] H. J. Miller, "Geographic information science I: Geographic information observatories and opportunistic GIScience," *Prog. Hum. Geogr.*, vol. 41, no. 4, pp. 489–500, May 2017.

[13] C. Goodison, A. Guillaume thomas, and S. Palmer, "The Florida Geographic Data Library: Lessons Learned and Workflows for Geospatial Data Management," *J. Map Geogr. Libr.*, vol. 12, no. 1, pp. 73–99, Jan. 2016.

[14] M. F. Worboys, "A Unified Model for Spatial and Temporal Information," *Comput. J.*, vol. 37, no. 1, pp. 26–34, Jan. 1994.

[15] M. Bloch, *Strange Defeat: a statement of evidence written in 1940*. London: Oxford University Press, 1949.

[16] T. Hagerstrand, "The Two Vistas," *Geogr. Ann. Ser. B Hum. Geogr.*, vol. 86, no. 4, pp. 315–323, Dec. 2004.

[17] M. Dijst, "Time Geographic Analysis," in *International Encyclopedia of Human Geography*, vol. 11, R. Kitchin and N. Thrift, Eds. Amsterdam: Elsevier, 2009, pp. 266–278.

[18] B. Lenntorp, "Time geography - at the end of its beginning," *GeoJournal*, vol. 48, no. 3, pp. 155–158, Jul. 1999.

[19] T. Hägerstrand and others, "Time-geography: focus on the corporeality of man, society, and environment," *Sci. Prax. Complex.*, pp. 193–216, 1985.

[20] 197 U.S. 11, *Jacobson v. Massachusetts*. 1905.

[21] L. O. Gostin, *Public Health Law: Power, Duty and Restraint*, 2nd ed. Berkely, CA: University of California Press, 2008.

[22] J. S. Buechner, H. Constantine, and A. Gjelsvik, "John Snow and the Broad Street Pump: 150 Years of Epidemiology," *Med. Heal. Rhode Island.*, vol. 87, no. 10, p. 314, 2004.

[23] S. Hajna, D. L. Buckeridge, and J. A. Hanley, "Substantiating the impact of John Snow's contributions using data deleted during the 1936 reprinting of his original essay On the Mode of Communication of Cholera," *Int. J. Epidemiol.*, vol. 44, no. 6, pp. 1794–1799, Dec. 2015.

[24] G. D. Smith, "Commentary: Behind the Broad Street pump: aetiology, epidemiology and prevention of cholera in mid-19th century Britain," *Int. J. Epidemiol.*, vol. 31, no. 5, pp. 920–932, Oct. 2002.

[25] C. Duncan, K. Jones, and G. Moon, "Context, composition and heterogeneity: Using multilevel models in health research," *Soc. Sci. Med.*, vol. 46, no. 1, pp. 97–117, Jan. 1998.

[26] A. Case, A. Fertig, and C. Paxson, "The lasting impact of childhood health and circumstance," *J. Health Econ.*, vol. 24, no. 2, pp. 365–389, Mar. 2005.

[27] D. B. Richardson, N. D. Volkow, M.-P. Kwan, R. M. Kaplan, M. F. Goodchild, and R. T. Croyle, "Spatial Turn in Health Research," *Science (80-.).*, vol. 339, no. 6126, pp. 1390–1392, Mar. 2013.

[28] R. C. Brownson, J. F. Chriqui, and K. A. Stamatakis, "Understanding Evidence-Based Public Health Policy," *Am. J. Public Health*, vol. 99, no. 9, pp. 1576–1583, Sep. 2009.

[29] G. A. Elmes, "GIS in Public Healthcare Planning: The United States Perspective," in *GIS in Public Health Practice*, R. Maheswaran and M. Craglia, Eds. Boca Raton, FL: CRC Press, 2004, pp. 205–26.

[30] M. H. Boyle and J. D. Willms, "Place Effects for Areas Defined by Administrative Boundaries," *Am. J. Epidemiol.*, vol. 149, no. 6, pp. 577–585, Mar. 1999.

[31] S. I. Hay and R. W. Snow, "The Malaria Atlas Project: Developing Global Maps of Malaria Risk," *PLoS Med*, vol. 3, no. 12, p. e473, Dec. 2006.

[32] M.-P. Kwan, "The Uncertain Geographic Context Problem," *Ann. Assoc. Am. Geogr.*, vol. 102, no. 5, pp. 958–968, Sep. 2012.

[33] C. Leal and B. Chaix, "The influence of geographic life environments on cardiometabolic risk factors: a systematic review, a methodological assessment and a research agenda," *Obes. Rev.*, vol. 12, no. 3, pp. 217–230, Mar. 2011.

[34] P. Bernard, R. Charafeddine, K. L. Frohlich, M. Daniel, Y. Kestens, and L. Potvin, "Health inequalities and place: A theoretical conception of neighbourhood," *Soc. Sci. Med.*, vol. 65, no. 9, pp. 1839–1852, Nov. 2007.

[35] C. S. Aneshensel and C. A. Sucoff, "The Neighborhood Context of Adolescent Mental Health," *J. Health Soc. Behav.*, vol. 37, no. 4, p. 293, Dec. 1996.

[36] R. L. LaGrange, K. F. Ferraro, and M. Supancic, "Perceived Risk and Fear of Crime: Role of Social and Physical Incivilities," *J. Res. Crime Delinq.*, vol. 29, no. 3, pp. 311–334, Aug. 1992.

[37] B. S. Rosenthal and W. C. Wilson, "The association of ecological variables and psychological distress with exposure to community violence among adolescents.," *Adolescence*, vol. 38, no. 151, pp. 459–479, 2003.

[38] S. E. Wiehe, S. C. Hoch, G. C. Liu, A. E. Carroll, J. S. Wilson, and J. D. Fortenberry, "Adolescent Travel Patterns: Pilot Data Indicating Distance from Home Varies by Time of Day and Day of Week," *J. Adolesc. Heal.*, vol. 42, no. 4, pp. 418–420, Apr. 2008.

[39] L. A. Basta, T. S. Richmond, and D. J. Wiebe, "Neighborhoods, daily activities, and measuring health risks experienced in urban environments," *Soc. Sci. Med.*, vol. 71, no. 11, pp. 1943–1950, Dec. 2010.

[40] K. Elgethun, R. A. Fenske, M. G. Yost, and G. J. Palcisko, “Time-location analysis for exposure assessment studies of children using a novel global positioning system instrument.,” *Environ. Health Perspect.*, vol. 111, no. 1, pp. 115–122, 2003.

[41] M. Kiskowski, “A Three-Scale Network Model for the Early Growth Dynamics of 2014 West Africa Ebola Epidemic,” *PLoS Curr.*, pp. 1–15, 2014.

[42] R. Pastor-Satorras and A. Vespignani, “Epidemic Spreading in Scale-Free Networks,” *Phys. Rev. Lett.*, vol. 86, no. 14, pp. 3200–3203, Apr. 2001.

[43] R. Pastor-Satorras and A. Vespignani, “Epidemic dynamics in finite size scale-free networks,” *Phys. Rev. E*, vol. 65, no. 3, p. 035108, Mar. 2002.

[44] A. Ellaway, S. Macintyre, and A. Kearns, “Perceptions of Place and Health in Socially Contrasting Neighbourhoods,” *Urban Stud.*, vol. 38, no. 12, pp. 2299–2316, Nov. 2001.

[45] M.-P. Kwan and G. Ding, “Geo-Narrative: Extending Geographic Information Systems for Narrative Analysis in Qualitative and Mixed-Method Research∗,” *Prof. Geogr.*, vol. 60, no. 4, pp. 443–465, Sep. 2008.

[46] A. M. Noor, J. J. Mutheu, A. J. Tatem, S. I. Hay, and R. W. Snow, “Insecticide-treated net coverage in Africa: mapping progress in 2000–07,” *Lancet*, vol. 373, no. 9657, pp. 58–67, Jan. 2009.

[47] A. M. van Eijk *et al.*, “Coverage of malaria protection in pregnant women in sub-Saharan Africa: a synthesis and analysis of national survey data,” *Lancet Infect. Dis.*, vol. 11, no. 3, pp. 190–207, Mar. 2011.

[48] A. Wollum, R. Burstein, N. Fullman, L. Dwyer-Lindgren, and E. Gakidou, “Benchmarking health system performance across states in Nigeria: a systematic analysis of levels and trends in key maternal and child health interventions and outcomes, 2000–2013,” *BMC Med.*, vol. 13, no. 1, p. 208, Dec. 2015.

[49] C. E. Armstrong *et al.*, “Subnational variation for care at birth in Tanzania: is this explained by place, people, money or drugs?,” *BMC Public Health*, vol. 16, no. S2, p. 795, Sep. 2016.

[50] C. A. Gotway and L. J. Young, “Combining Incompatible Spatial Data,” *J. Am. Stat. Assoc.*, vol. 97, no. 458, pp. 632–648, Jun. 2002.

[51] R. A. Olea, *Geostatistical glossary and multilingual dictionary*, no. 3. New York, New York, USA: Oxford University Press, 1991.

[52] A. S. Fotheringham and D. W. S. Wong, “The Modifiable Areal Unit Problem in Multivariate Statistical Analysis,” *Environ. Plan. A*, vol. 23, no. 7, pp. 1025–1044, Jul. 1991.

[53] S. Openshaw, “Concepts and techniques in modern geography number 38: the modifiable areal unit problem,” *Norwick Geo Books*, 1984.

[54] S. Openshaw Taylor, P J, “A million or so correlation coefficients: three experiments on the modifiable areal unit problem,” in *Statistical Applications in the Spatial Sciences*, London: Pion, 1979, pp. 127–144.

[55] S. J. Dark and D. Bram, “The modifiable areal unit problem (MAUP) in physical geography,” *Prog. Phys. Geogr. Earth Environ.*, vol. 31, no. 5, pp. 471–479, Feb. 2007.

[56] A. Hagopian *et al.*, “Mortality in Iraq Associated with the 2003–2011 War and Occupation: Findings from a National Cluster Sample Survey by the University Collaborative Iraq Mortality Study,” *PLoS Med.*, vol. 10, no. 10, p. e1001533, Oct. 2013.

[57] P. W. Setel *et al.*, “A scandal of invisibility: making everyone count by counting everyone.,” *Lancet*, vol. 370, no. 9598, pp. 1569–77, Nov. 2007.

[58] K. A. Lindblade *et al.*, “Sustainability of Reductions in Malaria Transmission and Infant Mortality in Western Kenya With Use of Insecticide-Treated Bednets,” *JAMA*, vol. 291,

no. 21, p. 2571, Jun. 2004.

[59] R. W. Snow, M. Craig, U. Deichmann, and K. Marsh, “Estimating mortality, morbidity and disability due to malaria among Africa’s non-pregnant population,” *Bull. World Health Organ.*, vol. 77, no. 8, pp. 624–640, 1999.

[60] R. P. Ndugwa *et al.*, “Comparison of all-cause and malaria-specific mortality from two West African countries with different malaria transmission patterns,” *Malar. J.*, vol. 7, no. 1, p. 15, Jan. 2008.

[61] K. Zinszer *et al.*, “Integrated Disease Surveillance to Reduce Data Fragmentation – An Application to Malaria Control,” *Online J. Public Health Inform.*, vol. 7, no. 1, p. e181, Feb. 2015.

[62] United Nations Food and Agriculture Office, “Global Administrative Units Layer.” 2015.

[63] Center for International Earth Science Information Network, “Gridded Population of the World (GPWv3),” 2004. [Online]. Available: http://sedac.ciesin.columbia.edu/data/collection/gpw-v3.

[64] S. S. Lim *et al.*, “A comparative risk assessment of burden of disease and injury attributable to 67 risk factors and risk factor clusters in 21 regions, 1990-2010: a systematic analysis for the Global Burden of Disease Study 2010.,” *Lancet*, vol. 380, no. 9859, pp. 2224–60, Dec. 2013.

[65] J. E. Dobson, E. A. Bright, P. R. Coleman, R. C. Durfee, and B. A. Worley, “LandScan: A global population database for estimating populations at risk,” *Photogramm. Eng. Remote Sensing*, vol. 66, no. 7, pp. 849–857, 2000.

[66] A. J. Tatem, A. M. Noor, C. von Hagen, A. Di Gregorio, and S. I. Hay, “High Resolution Population Maps for Low Income Nations: Combining Land Cover and Census in East Africa,” *PLoS One*, vol. 2, no. 12, p. e1298, Dec. 2007.

[67] S. I. Hay *et al.*, “A world malaria map: Plasmodium falciparum endemicity in 2007,” *PLoS Med*, vol. 6, 2009.

[68] S. I. Hay, C. A. Guerra, A. J. Tatem, P. M. Atkinson, and R. W. Snow, “Urbanization, malaria transmission and disease burden in Africa,” *Nat Rev Microbiol*, vol. 3, 2005.

[69] U.S. Geological Survey, “U.S. Geological Survey (USGS) Digital Elevation Models (DEM). [http://edc2.usgs.gov/geodata/index.php].” .

[70] U.S. Geological Survey, “U.S. Geological Survey (USGS) Famine Early Warning Systems Network (FEWS NET) African Data Dissemination Service (ADDS).” U.S. Geological Survey.

[71] M. Hutchinson, H. Nix, and J. McMahon, “Africa - A Topographic and Climate Database (CD-ROM).” 1996.

[72] A. Gemperli *et al.*, “Mapping malaria transmission in West and Central Africa,” *Trop. Med. Int. Heal.*, vol. 11, no. 7, pp. 1032–1046, Jul. 2006.

[73] F. C. Tanser, B. Sharp, and D. le Sueur, “Potential effect of climate change on malaria transmission in Africa,” *Lancet*, vol. 362, no. 9398, pp. 1792–1798, 2003.

[74] L. Gosoniu, P. Vounatsou, N. Sogoba, and T. Smith, “Bayesian modelling of geostatistical malaria risk data.,” *Geospat. Health*, vol. 1, no. 1, pp. 127–139, 2006.

[75] L. Gosoniu, P. Vounatsou, N. Sogoba, N. Maire, and T. Smith, “Mapping malaria risk in West Africa using a Bayesian nonparametric non-stationary model,” *Comput. Stat. Data Anal.*, vol. 53, no. 9, pp. 3358–3371, Jul. 2009.

[76] P. Agbu and M. James, “NOAA/NASA Pathfinder AVHRR Land Data Set User’s Manual,” Greenbelt, 1994.

[77] N. Riedel *et al.*, “Geographical patterns and predictors of malaria risk in Zambia: Bayesian geostatistical modelling of the 2006 Zambia national malaria indicator survey

(ZMIS)," *Malar. J.*, vol. 9, no. 1, p. 37, 2010.

[78] J. A. Omumbo, S. I. Hay, R. W. Snow, A. J. Tatem, and D. J. Rogers, "Modelling malaria risk in East Africa at high-spatial resolution," *Trop. Med. Int. Heal.*, vol. 10, no. 6, pp. 557–566, 2005.

[79] F. Giardina *et al.*, "Estimating the Burden of Malaria in Senegal: Bayesian Zero-Inflated Binomial Geostatistical Modeling of the MIS 2008 Data," *PLoS One*, vol. 7, no. 3, p. e32625, Mar. 2012.

[80] L. Gosoniu, A. Msengwa, C. Lengeler, and P. Vounatsou, "Spatially Explicit Burden Estimates of Malaria in Tanzania: Bayesian Geostatistical Modeling of the Malaria Indicator Survey Data," *PLoS One*, vol. 7, no. 5, p. e23966, May 2012.

[81] G. Raso *et al.*, "Mapping malaria risk among children in Côte d'Ivoire using Bayesian geo-statistical models," *Malar. J.*, vol. 11, no. 1, p. 160, 2012.

[82] "U.S. Geological Survey (USGS) Land Processes Distributed Active Archive Center (LP DAAC) MODIS land products. [https://lpdaac.usgs.gov/lpdaac/products/modis_products_table]." .

[83] World Resources Institute, "African Data Sampler (CD-ROM)." World Resources Institute, Washington, DC, 1995.

[84] U. Deichman, "African Population Database." National Center for Geographic Information Analysis, University of California, Santa Barbara, CA, 1996.

[85] "Oak Ridge National Laboratory (Oak Ridge, TN) LandScan^{TM} Global Population Database. [http://www.ornl.gov/landscan/]." .

[86] T. O. Alimi *et al.*, "A multi-criteria decision analysis approach to assessing malaria risk in northern South America," *BMC Public Health*, vol. 16, no. 1, pp. 1–10, 2016.

[87] "World Health Organization (WHO) The HealthMapper Database. [http://www.who.int/health_mapping/tools/healthmapper/en/index.html]." .

[88] United Nations Environmente Programme, "Global Resource Information Database (GRID)." [Online]. Available: http://www.grid.unep.ch/index.php?lang=en.

[89] F. L. Snijders, "Rainfall monitoring based on Meteosat data— a comparison of techniques applied to the Western Sahel," *Int. J. Remote Sens.*, vol. 12, no. 6, pp. 1331–1347, Jun. 1991.

[90] Global Land Cover Network (FAO), "Africover," 2013. [Online]. Available: http://www.glcn.org/activities/africover_en.jsp.

[91] Walter Reed Army Institute of Research, "Walter Reed Biosystematics Unit," 2016. [Online]. Available: www.wrbu.org.

[92] GBIF Secretariat, "Global Biodiversity Information Facility," 2016. .

[93] L. Bengtsson, X. Lu, A. Thorson, R. Garfield, and J. von Schreeb, "Improved response to disasters and outbreaks by tracking population movements with mobile phone network data: a post-earthquake geospatial study in Haiti.," *PLoS Med.*, vol. 8, no. 8, p. e1001083, Aug. 2011.

[94] "Haiti Earthquake - Population Movements out of Port-au-Prince (as of 17 Feb 2010)." UNOCHA, 20-Feb-2018.

[95] ACAPS, "Call Detail Records: The use of mobile phone data to track and predict population displacement in disaster," 2013.

[96] P. Meier and R. Munro, "The Unprecedented Role of SMS in Disaster Response : Learning from Haiti," *SAIS Rev. Int. Aff.*, vol. 30, no. 2, pp. 91–103, 2010.

[97] X. Lu, L. Bengtsson, and P. Holme, "Predictability of population displacement after the 2010 Haiti earthquake.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 109, no. 29, pp. 11576–81, Jul. 2012.

[98] D. Coyle and P. Meier, "New Technologies in Emergencies and Conflicts: The Role of Information and Social Networks," Washington, DC and London, UK, 2009.

[99] M. Lichman and P. Smyth, "Modeling human location data with mixtures of kernel densities," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014, pp. 35–44.

[100] D. A. Roberts *et al.*, "Benchmarking health system performance across regions in Uganda: a systematic analysis of levels and trends in key maternal and child health interventions, 1990–2011," *BMC Med.*, vol. 13, no. 1, p. 285, Dec. 2015.

[101] K. P. Schwarz and N. El-Sheimy, "Mobile mapping systems–state of the art and future trends," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 35, no. Part B, p. 10, 2004.

[102] M. F. Goodchild, "Prospects for a Space–Time GIS," *Ann. Assoc. Am. Geogr.*, vol. 103, no. 5, pp. 1072–1077, Sep. 2013.

[103] P. Wegner, "Dimensions of object-based language design," in *Conference proceedings on Object-oriented programming systems, languages and applications - OOPSLA '87*, 1987, pp. 168–182.

[104] M. Yuan, "Modelling Semantical, Spatial and Temporal Information in a GIS," in *Geographic Information Research: Bridging the Atlantic*, M. Craglia and H. Couclelis, Eds. London: Taylor & Francis, 1996, pp. 334–347.

[105] S. N. Khoshafian and G. P. Copeland, "Object identity," *ACM SIGPLAN Not.*, vol. 21, no. 11, pp. 406–416, Nov. 1986.

[106] C. J. Date, *An Introduction to Database Systems*, 4th ed. Reading, MA: Addison-Wesley, 1983.

[107] T. Abraham and J. F. Roddick, "Survey of Spatio-Temporal Databases," *Geoinformatica*, vol. 3, no. 1, pp. 61–99, 1999.

[108] T. Ott and F. Swiaczny, *Time-Integrative Geographic Information Systems*. Heidelberg, Germany: Springer Berlin / Heidelberg, 2001.

[109] M. F. Goodchild and Y. Shiren, "A hierarchical spatial data structure for global geographic information systems," *CVGIP Graph. Model. Image Process.*, vol. 54, no. 1, pp. 31–44, Jan. 1992.

[110] M. Yuan, "Use of a Three-Domain Representation to Enhance GIS Support for Complex Spatiotemporal Queries," *Trans. GIS*, vol. 3, no. 2, pp. 137–159, Mar. 1999.

[111] M. J. Egenhofer, "Reasoning about binary topological relations," in *Advances in spatial databases*, O. Günther and H.-J. Schek, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991, pp. 141–160.

[112] E. Clementini and P. Di Felice, "A comparison of methods for representing topological relationships," *Inf. Sci. - Appl.*, vol. 3, no. 3, pp. 149–178, May 1995.

[113] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983.

[114] P. Agouris, S. Gyftakis, and A. Stefanidis, "Using a Fuzzy Supervisor for Object Extraction within an Integrated Geospatial Environment," in *International Archives of Photogrammetry and Remote Sensing*, 1998, pp. 191–195.

[115] G. Fu, C. B. Jones, and A. I. Abdelmoty, "Ontology-Based Spatial Query Expansion in Information Retrieval," in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, R. Meersman and Z. Tari, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1466–1482.

[116] T. J. Davis and C. P. Keller, "Modelling and visualizing multiple spatial uncertainties," *Comput. Geosci.*, vol. 23, no. 4, pp. 397–408, May 1997.

[117] Snodgrass and llsoo Ahn, “Temporal Databases,” *Computer (Long. Beach. Calif).*, vol. 19, no. 9, pp. 35–42, Sep. 1986.

[118] G. Ozsoyoglu and R. T. Snodgrass, “Temporal and real-time databases: a survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 4. pp. 513–532, 1995.

[119] J. J. Berman, “Introduction,” in *Principles of Big Data*, J. J. B. T.-P. of B. D. Berman, Ed. Boston: Elsevier, 2013, pp. xix–xxvi.

[120] G. Langran, “Issues of implementing a spatiotemporal system,” *Int. J. Geogr. Inf. Syst.*, vol. 7, no. 4, pp. 305–314, Jul. 1993.

[121] B. Magaš, *Croatia Through History: The Making of a European State*. London: Saqi Books, 2007.

[122] G. Langran and N. R. Chrisman, “A Framework For Temporal Geographic Information,” *Cartogr. Int. J. Geogr. Inf. Geovisualization*, vol. 25, no. 3, pp. 1–14, Oct. 1988.

[123] M. Cserép and R. Giachetta, “Operation-based revision control for geospatial data sets,” *Geomatics Work. n° 12 – “FOSS4G Eur. Como 2015,”* no. July 2015, pp. 139–152, 2015.

[124] M. Zaharia *et al.*, “Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing,” in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, 2012, p. 2.

[125] M. F. Worboys, “A model for spatio-temporal information,” in *Proceedings of the 5th International Symposium on Spatial Data Handling*, 1992, pp. 602–611.

[126] A. Guttman, “R-trees,” in *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD ’84*, 1984, p. 47.

[127] R. K. V Kothuri, S. Ravada, and D. Abugov, “Quadtree and R-tree indexes in oracle spatial,” in *Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD ’02*, 2002, p. 546.

[128] K. I. Lin, H. V Jagadish, and C. Faloutsos, “The TV-tree: An Index Structure for High-dimensional Data,” *VLDB J.*, vol. 3, no. 4, pp. 517–542, Oct. 1994.

[129] J. K. Lawder and P. J. H. King, “Querying multi-dimensional data indexed using the Hilbert space-filling curve,” *ACM SIGMOD Rec.*, vol. 30, no. 1, pp. 19–24, Mar. 2001.

[130] W. Wu, Y. Zheng, H. Qu, W. Chen, E. Groller, and L. M. Ni, “BoundarySeer: Visual analysis of 2D boundary changes,” in *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2014, pp. 143–152.

[131] S. J. Rey and M. V Janikas, “STARS: Space-Time Analysis of Regional Systems,” in *Handbook of Applied Spatial Analysis*, M. M. Fischer and A. Getis, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 91–112.

[132] G. M. Jacquez, “Space-Time Intelligence System Software for the Analysis of Complex Systems,” in *Handbook of Applied Spatial Analysis*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 113–124.

[133] S. Gebbert and E. Pebesma, “TGRASS: A temporal GIS for field based environmental modeling,” *Environ. Model. Softw.*, vol. 53, 2014.

[134] E. Pultar, T. Cova, M. Yuan, and M. Goodchild, “EDGIS: a dynamic GIS based on space time points,” *Int. J. Geogr. Inf. Sci.*, vol. 24, no. 3, pp. 329–346, Mar. 2010.

[135] R. Bose and F. Reitsma, “Advancing Geospatial Data Curation,” Edinburgh, UK, GEO-013, 2005.

[136] A. Sotnykova, C. Vangenot, N. Cullot, N. Bennacer, and M.-A. Aufaure, “Semantic Mappings in Description Logics for Spatio-temporal Database Schema Integration,” in *Journal on Data Semantics III*, S. Spaccapietra and E. Zimányi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 143–167.

[137] R. J. Miller *et al.*, “The Clio project: managing heterogeneity,” *SIgMOD Rec.*, vol. 30, no.

1, pp. 78–83, 2001.

[138] N. Guarino and P. Giaretta, "Ontologies and knowledge bases: towards a terminological clarification," in *Towards Very Large Knowledge Bases, Knowledge Building & Knowledge Sharing 1995*, N. J. I. Mars, Ed. Amsterdam: IOP Press, 1995, pp. 25–32.

[139] N. Guarino, "Understanding, building and using ontologies," *Int. J. Hum. Comput. Stud.*, vol. 46, no. 2–3, pp. 293–310, Feb. 1997.

[140] B. Smith and D. M. Mark, "Do Mountains Exist? Towards an Ontology of Landforms," *Environ. Plan. B Plan. Des.*, vol. 30, no. 3, pp. 411–427, Jun. 2003.

[141] P. Garbacz, R. Trypuz, B. Szady, P. Kulicki, P. Gradzki, and M. Lechniak, "Towards a formal ontology for history of church administration," in *Formal Ontology in Information Systems*, A. Galton and R. Mizoguchi, Eds. IOP Press, 2010, pp. 345–358.

[142] Gene Ontology Consortium, "Creating the gene ontology resource: design and implementation.," *Genome Res.*, vol. 11, no. 8, pp. 1425–1433, 2001.

[143] N. Collier *et al.*, "A multilingual ontology for infectious disease surveillance: rationale, design and challenges," *Lang. Resour. Eval.*, vol. 40, no. 3–4, pp. 405–413, Oct. 2007.

[144] K. D. Mandl *et al.*, "Implementing syndromic surveillance: a practical guide informed by the early experience.," *J. Am. Med. Inform. Assoc.*, vol. 11, no. 2, pp. 141–50, Mar. 2004.

[145] O. Seneviratne, L. Kagal, and T. Berners-Lee, "Policy-Aware Content Reuse on the Web," in *The Semantic Web - ISWC 2009 SE - 35*, 2009, vol. 5823, pp. 553–568.

[146] E. Ferrari and B. M. Thuraisingham, *Web and information security*. Hershey, PA: IRM Press, 2006.

[147] A. U. Frank, "Ontology for Spatio-Temporal Databases," in *Spatio-Temporal Databases: The CHOROCRONOS Approach*, M. Koubarakis, T. Sellis, and A. U. Frank, Eds. Berlin/Heidelberg: Springer, 2003, pp. 9–77.

[148] C. Parent, S. Spaccapietra, and E. Zimányi, "Spatio-temporal conceptual models," in *Proceedings of the seventh ACM international symposium on Advances in geographic information systems - GIS '99*, 1999, pp. 26–33.

[149] I. Budak Arpinar, A. Sheth, C. Ramakrishnan, E. Lynn Usery, M. Azami, and M.-P. Kwan, "Geospatial Ontology Development and Semantic Analytics," *Trans. GIS*, vol. 10, no. 4, pp. 551–575, Jul. 2006.

[150] P. Grenon and B. Smith, "SNAP and SPAN: Towards Dynamic Spatial Ontology," *Spat. Cogn. Comput.*, vol. 4, no. 1, pp. 69–104, Mar. 2004.

[151] T. Nyerges, M. Roderick, S. Prager, D. Bennett, and N. Lam, "Foundations of sustainability information representation theory: spatial–temporal dynamics of sustainable systems," *Int. J. Geogr. Inf. Sci.*, vol. 28, no. 5, pp. 1165–1185, May 2014.

[152] B. Harbelot, H. Arenas, and C. Cruz, "LC3: A spatio-temporal and semantic model for knowledge discovery from geospatial datasets," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 35, pp. 3–24, Dec. 2015.

[153] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler, "OWL 2: The next step for OWL," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 6, no. 4, pp. 309–322, 2008.

[154] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen, "From SHIQ and RDF to OWL: the making of a Web Ontology Language," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 1, no. 1, pp. 7–26, 2003.

[155] D. Allemang and J. Hendler, *Semantic Web for the Working Ontologist*, 2nd ed. Waltham, MA: Elsevier, 2011.

[156] M. A. Musen, "The protégé project," *AI Matters*, vol. 1, no. 4, pp. 4–12, Jun. 2015.

[157] K. Hornsby and M. J. Egenhofer, "Identity-based change: a foundation for spatio-temporal

knowledge representation," *Int. J. Geogr. Inf. Sci.*, vol. 14, no. 3, pp. 207–224, Apr. 2000.

[158] C. C. Aggarwal and H. Wang, "Graph Data Management and Mining: A Survey of Algorithms and Applications," in *Managing and Mining Graph Data*, C. C. Aggarwal and H. Wang, Eds. Boston, MA: Springer US, 2010, pp. 13–68.

[159] M. Perry and J. Herring, "OGC GeoSPARQL - A Geographic Query Language for RDF Data," 2012.

[160] S. Sakr, M. Wylot, R. Mutharaju, D. Le Phuoc, and I. Fundulaki, "Distributed Reasoning of RDF Data," in *Linked Data*, S. Sakr, M. Wylot, R. Mutharaju, D. Le Phuoc, and I. Fundulaki, Eds. Cham: Springer International Publishing, 2018, pp. 109–126.

[161] R. Gu, S. Wang, F. Wang, C. Yuan, and Y. Huang, "Cichlid: Efficient Large Scale RDFS/OWL Reasoning with Spark," in *2015 IEEE International Parallel and Distributed Processing Symposium*, 2015, pp. 700–709.

[162] D. Comer, "Ubiquitous B-Tree," *ACM Comput. Surv.*, vol. 11, no. 2, pp. 121–137, Jun. 1979.

[163] B. Stantic, R. Topor, J. Terry, and A. Sattar, "Advanced indexing technique for temporal data," *Comput. Sci. Inf. Syst.*, vol. 7, no. 4, pp. 679–703, 2010.

[164] D. Martin, "Extending the automated zoning procedure to reconcile incompatible zoning systems," *Int. J. Geogr. Inf. Sci.*, vol. 17, no. 2, pp. 181–196, Mar. 2003.

[165] N. Cressie and C. K. Wikle, *Statistics for Spatio-Temporal Data*, 1st ed. Wiley, 2011.

[166] A. A. Taha and A. Hanbury, "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool," *BMC Med. Imaging*, vol. 15, no. 1, p. 29, Dec. 2015.

[167] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, Mar. 1991.

[168] E. Green, "Development as State Making: District Creation and Decentralisation in Uganda," London, 2, 2008.

[169] J. A. Dougenik, N. R. Chrisman, and D. R. Niemeyer, "An Algorithm To Construct Continuous Area Cartograms," *Prof. Geogr.*, vol. 37, no. 1, pp. 75–81, Aug. 2018.

[170] Wikipedia Contributors, "NetRevenue2004Cartogram," 2012. [Online]. Available: https://commons.wikimedia.org/wiki/File:NetRevenue2004Cartogram.png. [Accessed: 29-Sep-2018].

[171] P. Hurvitz, "What is the difference between ZIP code 'boundaries' and ZCTA areas?," 2008. [Online]. Available: http://gis.washington.edu/phurvitz/zip_or_zcta/. [Accessed: 01-Aug-2018].

[172] S. Openshaw and L. Rao, "Algorithms for Reengineering 1991 Census Geography," *Environ. Plan. A*, vol. 27, no. 3, pp. 425–446, Mar. 1995.

[173] S. Openshaw, "An optimal zoning approach to the study of spatially aggregated data," in *Spatial representation and spatial interaction*, I. Masser and P. J. B. Brown, Eds. Boston, MA: Springer US, 1978, pp. 95–113.

[174] D. Martin, "Optimizing census geography: the separation of collection and output geographies," *Int. J. Geogr. Inf. Sci.*, vol. 12, no. 7, pp. 673–685, Nov. 1998.

[175] R. Sammons, "A simplistic approach to the redistricting problem," in *Spatial representation and spatial interaction*, I. Masser and P. J. B. Brown, Eds. Boston, MA: Springer US, 1978, pp. 71–94.

[176] J. Byfuglien and A. Nordgård, "Region-Building — a Comparison of Methods," *Nor. Geogr. Tidsskr. - Nor. J. Geogr.*, vol. 27, no. 2, pp. 127–151, Jan. 1973.

[177] S. Openshaw, "Optimal Zoning Systems for Spatial Interaction Models," *Environ. Plan. A Econ. Sp.*, vol. 9, no. 2, pp. 169–184, Feb. 1977.

[178] S. Alvanides, S. Openshaw, and P. Rees, "Designing your own geographies," 2002.

[179] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983.

[180] A. Hertz and D. de Werra, "Using tabu search techniques for graph coloring," *Computing*, vol. 39, no. 4, pp. 345–351, Dec. 1987.

[181] R. M. AssunÇão, M. C. Neves, G. Câmara, and C. Da Costa Freitas, "Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees," *Int. J. Geogr. Inf. Sci.*, vol. 20, no. 7, pp. 797–811, Aug. 2006.

[182] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep. 1999.

[183] D. Guo, "Regionalization with dynamically constrained agglomerative clustering and partitioning (REDCAP)," *Int. J. Geogr. Inf. Sci.*, vol. 22, no. 7, pp. 801–823, Jul. 2008.

[184] D. Guo and H. Wang, "Automatic Region Building for Spatial Analysis," *Trans. GIS*, vol. 15, no. s1, pp. 29–45, Jul. 2011.

[185] T. J. Cova and M. F. Goodchild, "Extending geographical representation to include fields of spatial objects," *Int. J. Geogr. Inf. Sci.*, vol. 16, no. 6, pp. 509–532, Sep. 2002.

[186] M. F. Goodchild, M. Yuan, and T. J. Cova, "Towards a general theory of geographic representation in GIS," *Int. J. Geogr. Inf. Sci.*, vol. 21, no. 3, pp. 239–260, Mar. 2007.

[187] E. L. Koua and M. Kraak, "A usability framework for the design and evaluation of an exploratory geovisualization environment," in *Proceedings. Eighth International Conference on Information Visualisation, 2004. IV 2004.*, 2004, pp. 153–158.

[188] E. L. Koua, A. Maceachren, and M. -J. Kraak, "Evaluating the usability of visualization methods in an exploratory geovisualization environment," *Int. J. Geogr. Inf. Sci.*, vol. 20, no. 4, pp. 425–448, Apr. 2006.

[189] A. Kjellin, L. W. Pettersson, S. Seipel, and M. Lind, "Evaluating 2D and 3D visualizations of spatiotemporal information," *ACM Trans. Appl. Percept.*, vol. 7, no. 3, pp. 1–23, Jun. 2010.

[190] S. Fuhrmann and W. Pike, "User-centered Design of Collaborative Geovisualization Tools," in *Exploring Geovisualization*, A. M. MacEachren and M.-J. B. T.-E. G. Kraak, Eds. Oxford: Elsevier, 2005, pp. 591–609.

[191] D. J. Solove and P. M. Schwartz, *Information Privacy Law*, 3rd ed. New York, N.Y.: Aspen Publishers, 2009.

[192] Environmental Systems Reserach Institute. Inc, "ESRI Shapefile Technical Description," Redlands, CA, 1997.

Chapter 9. APPENDICES

9.1 APPENDIX A

This first appendix goes into details regarding the prototype web application and visualization tools developed for this dissertation. While mentioned briefly in Section 5.4, this section will give more details into the design and implementation of the various tools, which are built on top of the Trestle system and utilize the core APIs to achieve their desired function.

It should be mentioned that these are prototype tools and visualizations which have not been subjected to any type of evaluation or design review. There some *rough edges* which still need to be addressed and features remaining to add, but the overall theory should be apparent and throughout this dissertation these tools have proven to be tremendously useful. Future work may include some more focused evaluation and design sessions, based on the accepted methodologies for evaluating geospatial visualization tools [187]–[190].

The web application comprises three major visualizations: the *dataset viewer*, *entity visualizer*, *spatial comparison tool*. Each of which will be described in the remaining sections of this appendix.

9.1.1 *Dataset viewer*

The first visualization built into the web application was the *dataset viewer*, which was designed as a first proof of concept of Trestle's ability to load and query spatio-temporal data. Figure 45 shows the viewer with data loaded from the GAUL dataset for the year 2013. The box on the right-hand side of the screen shows the available datasets in Trestle, which can be selected and added to the map (in this example it contains both GAUL as well as the dataset used in Chapter 6). The map itself shows the spatial layout of the Trestle_Objects for that period of time, which is modified by the yellow slider in the middle of the screen. Figure 46 shows the same spatial area but updated to show the state of the Trestle_Objects for the year 2007. These two images clearly show the different regions that comprised *Maputo* between the years 2007 and 2013.

The final feature of this tool is the ability to export data, currently shown on the map, into common, non-temporal spatial formats. Currently the tool supports *ESRI Shapefiles* and *GeoJSON* files[47]; however, support for additional formats (such as *TopoJSON*[48] or *KML*[49]) can be added in the future. This not only improves interoperability with existing research systems but also illustrates Trestle's ability to serve a central data repository, supporting existing research projects and workflows.

[47] *GeoJSON* is a spatial extension to the *JSON* data format, commonly used for exchanging data between web services.

[48] https://github.com/topojson/topojson

[49] https://developers.google.com/kml/

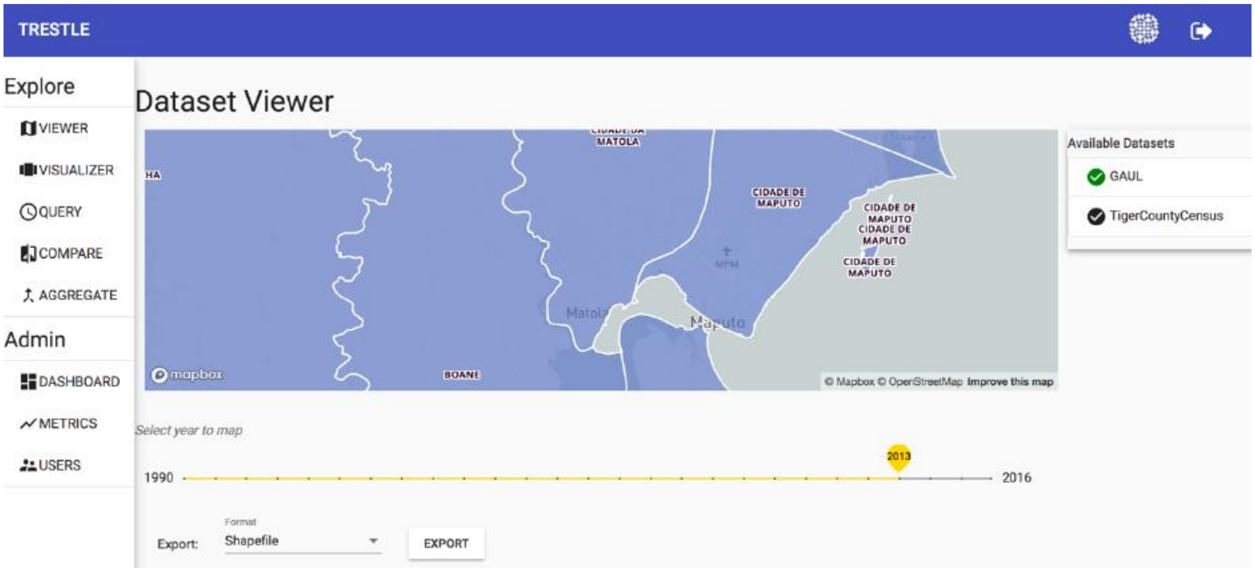


Figure 45: Dataset viewer showing GAUL data for *Maputo* in 2013.

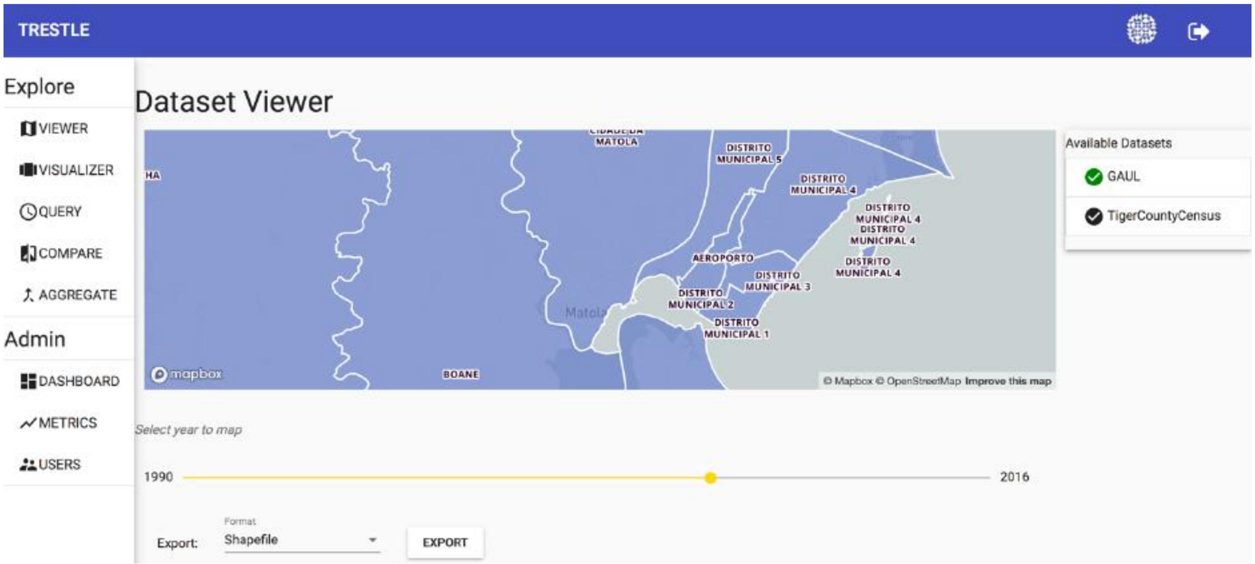


Figure 46: Dataset viewer showing the same area, but for the year 2007.

9.1.2 *Entity Visualizer*

The next visualization is a diagram which shows Trestle_Objects in the graph layout utilized by Trestle. Referred to as the *entity visualizer,* it was originally created to diagnose errors in the Trestle management application as new features were being developed. As previously mentioned, translating between the table and graph layouts is a challenging issue and existing tools are not well designed for showing graph relationships for a given entity, especially in the native Trestle graph layout. Thus, we developed our own set of tools for easily viewing both the graph layout of a Trestle_Object as well as its associated facts, relationships and values. It should be emphasized that this visualization was originally developed as a debugging tool is not intended to be directly used by end users. That being said, there are cases in which this tool could be of value to developers and system administrators.

Figure 47 shows the initial view of the entity visualizer. The map on the left shows the current spatial layout of the given entity (Trestle_Object). This feature allows mapping of any entity even if it does not currently exist in the real-world. For example, visualizing the *Aeroporto* region would show the most recent spatial boundary, even though that entity was merged into Cidade de Maputo and no longer exists. The diagram on the right shows the graph layout of entity, along with any and all associated facts. The dark-blue circle represents the *object header* of the entity, along with its associated *exists temporal* (shown with the light-blue circle)[50]. The facts are shown in the orange circles and have the added ability to visualize changing entity states over time. For facts that are not currently valid, either because there is a new version of that fact or the *valid temporal* is in the past (from the perspective of when the visualization is generated), these nodes are shown in light-orange. This allows the user to disambiguate which facts are valid at a given point in time, which is useful as the number of facts for a given entity grows over time. Figure 48 shows the same visualization as the previous image, but instead of showing *entity/fact* associations, it shows *entity/object relationships*; these two views (facts vs relationships) can be dynamically switched between or shown simultaneously. This figure also illustrates the prototype nature of the visualization in that with all the relationships visible, the graph is crowded to the point of being difficult to interpret. This will be addressed in a future software release.

[50] The exists temporal is currently labeled as *valid temporal* in the graph legend. This is an artifact of the prototype implementation and will be fixed in a future version.

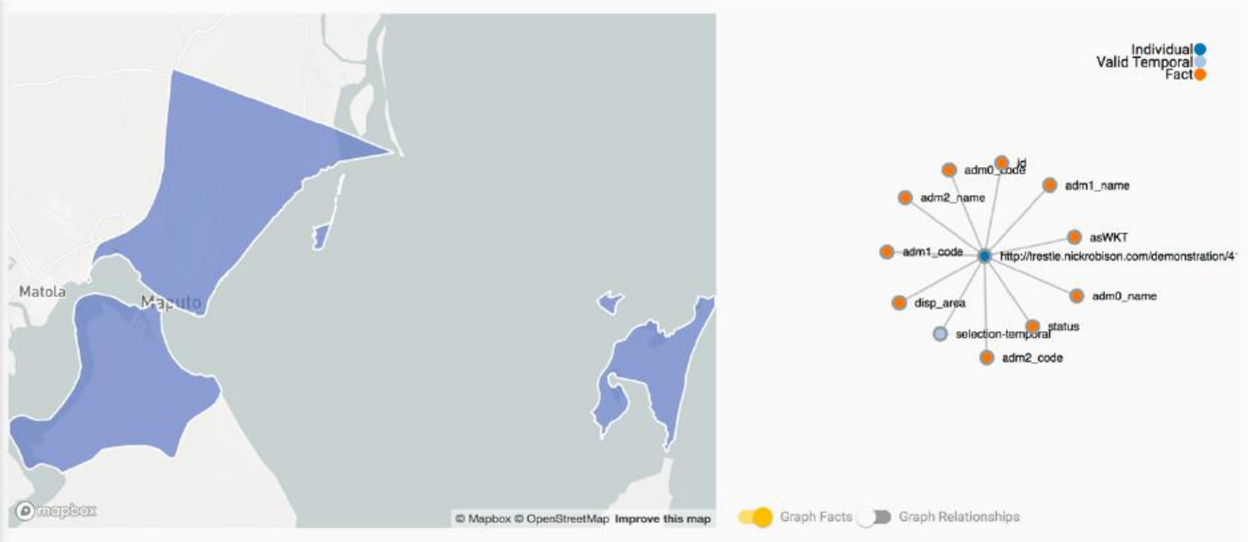


Figure 47: Entity visualizer showing Cidade de Maputo, with its associated facts.

This image shows the *entity visualization* for *Cidade de Maputo*. The map on the left shows the most recent spatial layout of the entity (regardless of whether or not the entity currently exists in the real-world). The graph diagram on the right shows the *object header* (dark blue circle), the existence interval of the entity (light-blue circle), and the various facts associated with it (orange circles).

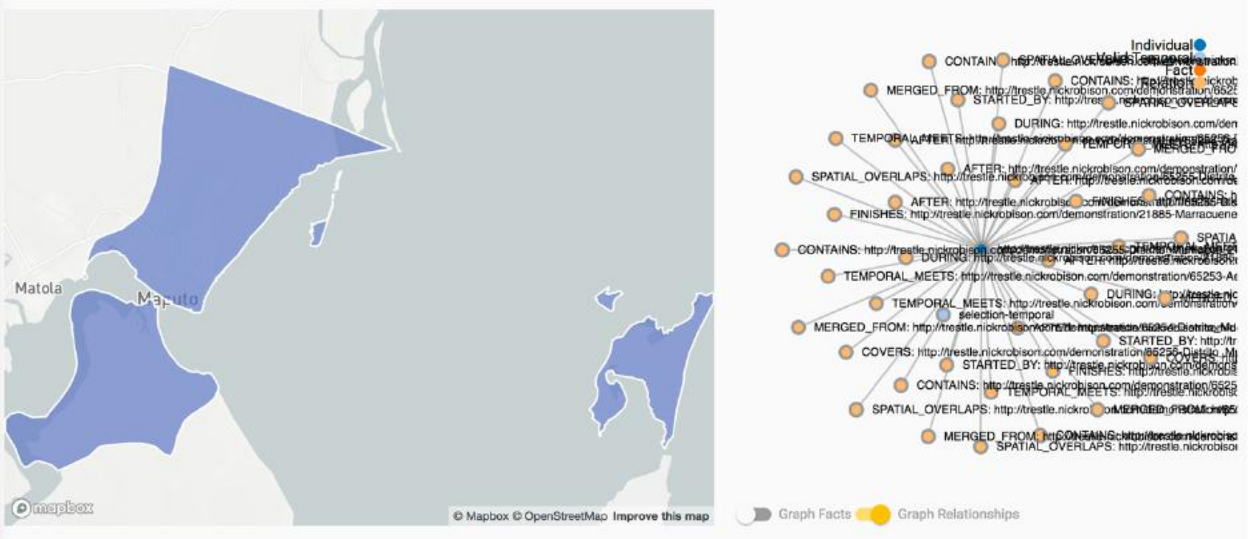

Individual ●
Valid Temporal ●
Fact ●
CONTAINS
MERGED_FROM: http://trestle.nickrobison.com/demonstration/6525
CONTAINS
STARTED_BY: http://tres
DURING: http://trestle.nickrobison.com/den
TEMPORAL_MEETS
AFTER
SPATIAL_OVERLAPS: http://trestle.nick
AFTER: http://trestle.nickrobison.com/demonstration/
AFTER: http://trestle.nickrobis
FINISHES: http://trestle.nickrobison.com/demonstration/21885-Marracuene
CONTAINS: http://trestle.nickrobison
DURING: http://trestle
TEMPORAL_MEETS: http://trestle.nickrobison.com/demonstration/65253-A
DURING: http://trestle.nic
TEMPORAL_MEETS: http://trestle.nickrobison.com/demonstration
selection-temporal
MERGED_FROM: http://trestle.nickrobison
STARTED_BY: http://tr
COVERS: http://trestle.nickrobison.com/demonstration/65256-Distrito
STARTED_BY: http://trestle.nickrobison.com/demonstra
FINISHES: http://trestle.nickrobis
CONTAINS: http://trestle.nickrobison.com/demonstration/6525
TEMPORAL_MEETS: http://trestle.nickrobis
SPATIAL_OVERLAPS: http://trestle.nickro
MERGED_FROM
MERGED_FROM
CONTAINS
SPATIAL_OVERLAPS: http://trestle.nickrobiso
SPATIA
Matola
Maputo
mapbox
© Mapbox © OpenStreetMap Improve this map
Graph Facts
Graph Relationships

Figure 48: Entity visualizer showing Cidade de Maputo, with its object relationships.

This figure again shows *Cidade de Maputo* but instead of visualizing the associated *facts* it shows the associated *object relationships* with other entities in the triple-store.

Figure 49 and Figure 50 show two different views of the data given in the previously described figures. Figure 49 shows the facts associated with the entity drawn as a time series. The X-axis represents the existence interval of the given entity and the Y-axis shows the various data properties associated with it. The individual facts are drawn as rectangles extending for the time period which they are valid. This approach is distinct from the one described above in that it allows for visualizing changes in facts and values over time. A more complex example, showing multiple fact value changes was previously given in Figure 7.

Figure 50 lists the object relationships for the given entity, in a table layout. Each relationship specifies the relationship type, such as overlaps (spatial), before (temporal), contains (spatial), etc. In addition, each relationship lists the Trestle_Object the entity is related to, along with a link (shown in blue) to show that entity in the visualizer. Figure 51 shows a final method for viewing fact values. Instead of the graph or time series layout, this gives a tabular view of the individual facts along with their associated values, datatypes, and temporal intervals (or points). This gives the user the ability to view fact values which may be too complex to be effectively shown in the time series view (such as long names or WKT spatial values), as well as the concrete datatype used to represent the value in the Trestle management application, which aids in developing tooling and support for existing applications and workflows.

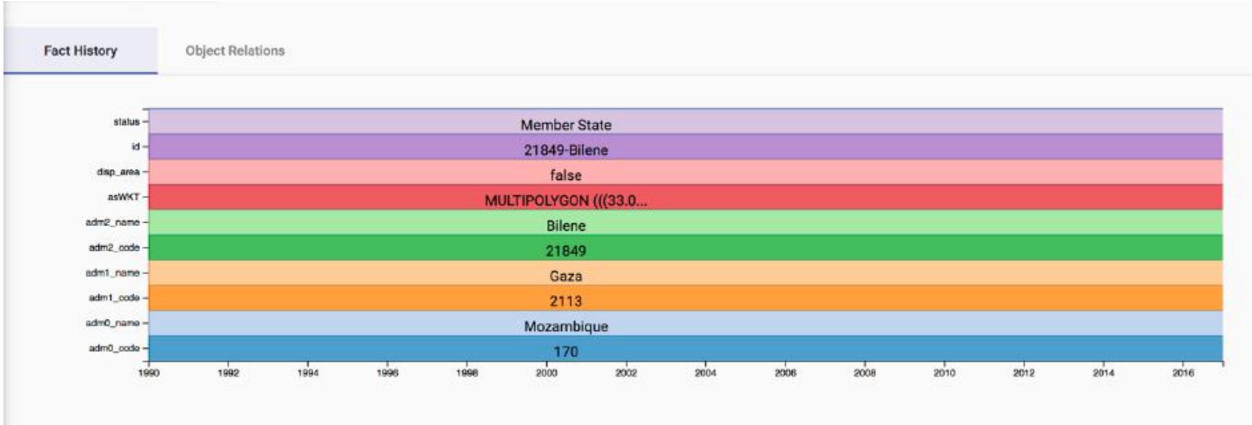
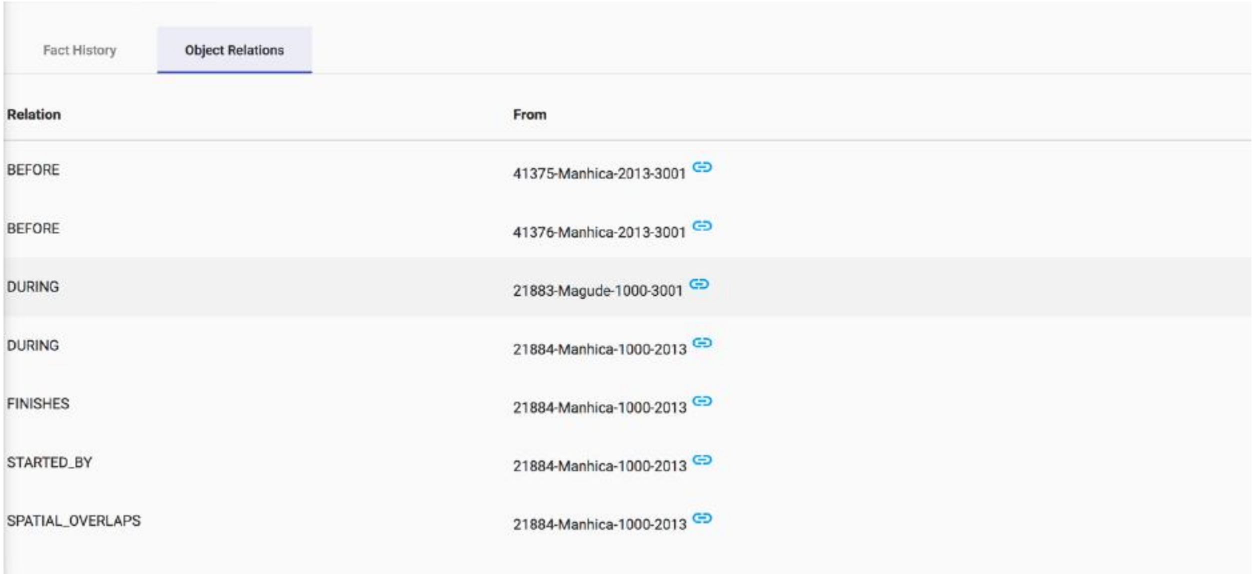


Figure 49: Visualizing the fact history of Cidade de Maputo.

This figure shows an alternative method of visualizing an entity's associated facts. Instead of a graph layout, facts are shown as rectangle, which extends along the X-axis for as long as that fact is valid. In this example, no fact values have changed and thus each property only has a single *fact* for the entire lifetime of the object. A more complex example with multiple fact value changes is shown in Figure 7.

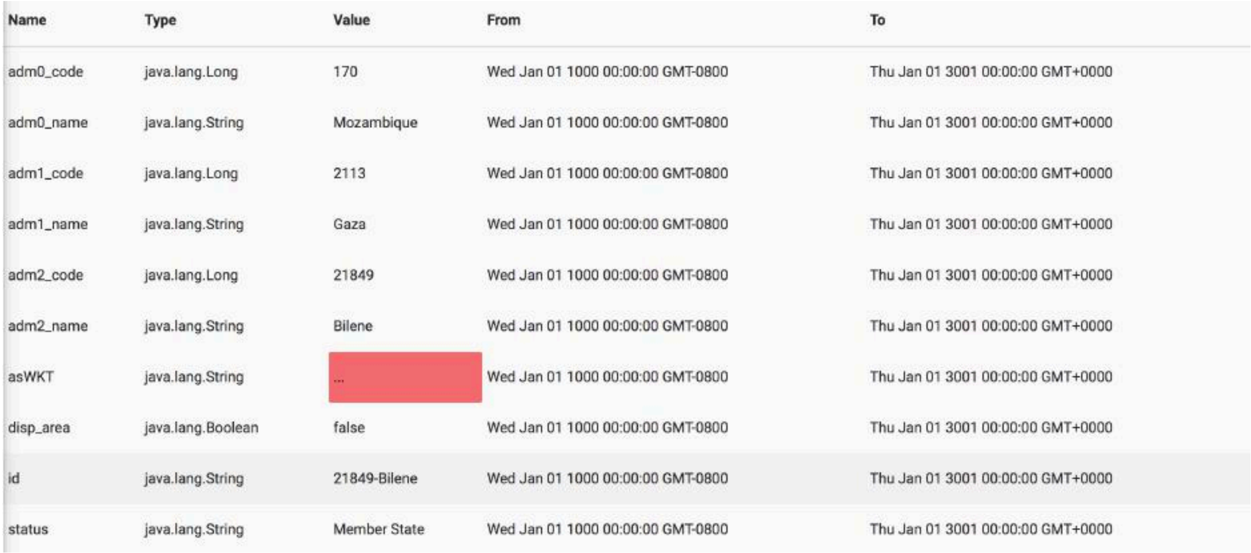


Fact History | Object Relations

Relation	From
BEFORE	41375-Manhica-2013-3001 🔗
BEFORE	41376-Manhica-2013-3001 🔗
DURING	21883-Magude-1000-3001 🔗
DURING	21884-Manhica-1000-2013 🔗
FINISHES	21884-Manhica-1000-2013 🔗
STARTED_BY	21884-Manhica-1000-2013 🔗
SPATIAL_OVERLAPS	21884-Manhica-1000-2013 🔗

Figure 50: Table layout of entity object relationships.

This figure gives a different view of the object relationships shown, in graph form, in Figure 48. Here, the relationships are presented in a table format with links (shown in blue) to the entity related to. Clicking the link switches the entity visualization to the linked Trestle_Object.



Name	Type	Value	From	To
adm0_code	java.lang.Long	170	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
adm0_name	java.lang.String	Mozambique	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
adm1_code	java.lang.Long	2113	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
adm1_name	java.lang.String	Gaza	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
adm2_code	java.lang.Long	21849	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
adm2_name	java.lang.String	Bilene	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
asWKT	java.lang.String	...	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
disp_area	java.lang.Boolean	false	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
id	java.lang.String	21849-Bilene	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000
status	java.lang.String	Member State	Wed Jan 01 1000 00:00:00 GMT-0800	Thu Jan 01 3001 00:00:00 GMT+0000

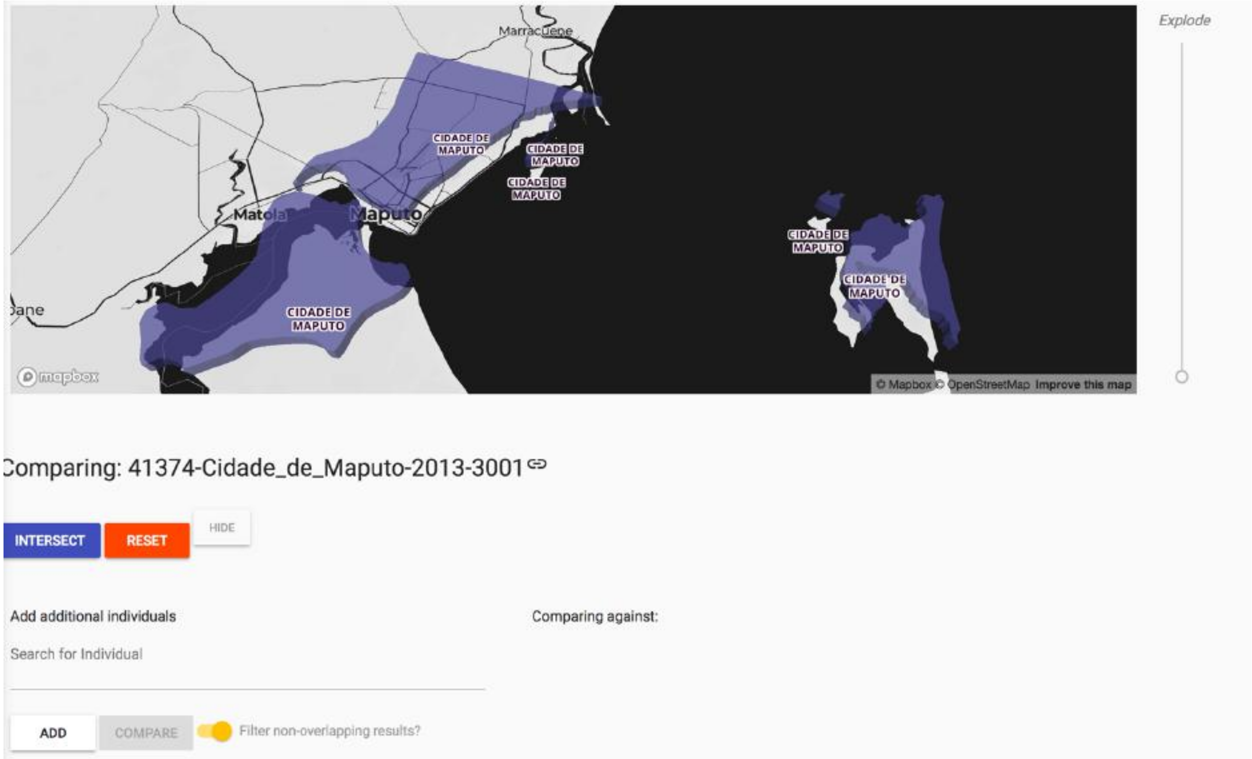
Figure 51: Technical view of facts showing values, datatypes and temporals intervals.

This figure gives a different view of the fact relationships shown, in graph form, in Figure 47. Here, the facts are presented in a table layout along with the concrete Java datatype, the value of the fact and the temporal interval (or point) for which the fact is valid.

9.1.3 *Spatial comparison tool*

The final visualization tool developed for Trestle is the *spatial comparison tool*. This was created to assist in the analysis of the split/merge algorithm results from Chapter 5. When attempting to determine whether or not a split/merge event occurred, a method was needed to view the temporal and spatial interactions between a given Trestle_Object and any other objects in the area. As there were no readily available mapping tools that would serve this purpose, a custom tool was developed. The following images will illustrate the operation of tool and how it is used in the Chapter 5 algorithm analysis.

Figure 52 shows the initial view of the comparison tool with *Cidade de Maputo* loaded as the entity to compare against. The screen gives an *Intersect* option (blue button) which when selected will intersect the object's boundary with any other Trestle_Objects in the same dataset and load all the objects onto the map. The results of the intersection are shown in Figure 53. The objects are drawn in the same manner as a traditional geographic map layout but instead of mapping all the objects flat against the map's surface, the objects are instead mapped along the Z-axis (height axis). Each object is drawn based on its *existence interval*. The *offset* of the object (its initial height above the map surface) is based on the start temporal of the exists interval. The *thickness* of the object is based on the end temporal of its existence interval. This allows the user to quickly determine the temporal dynamics of the objects' being studied, as well as how long each object has existed, in comparison to other objects on the map.



Comparing: 41374-Cidade_de_Maputo-2013-3001

Add additional individuals

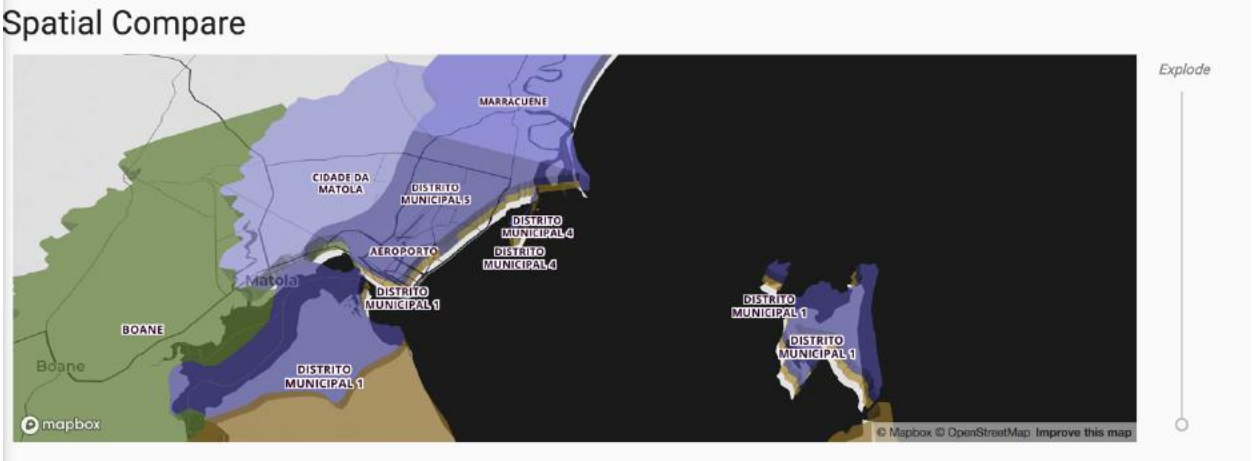
Search for Individual

Comparing against:

Filter non-overlapping results?

Figure 52: Initial view of the spatial comparison tool.

This shows the initial view of the spatial comparison tool, with *Cidade de Maputo* loaded as the entity to compare against. Clicking the *Intersect* button (shown in blue) will spatially intersect it with all other Trestle_Objects in that dataset.

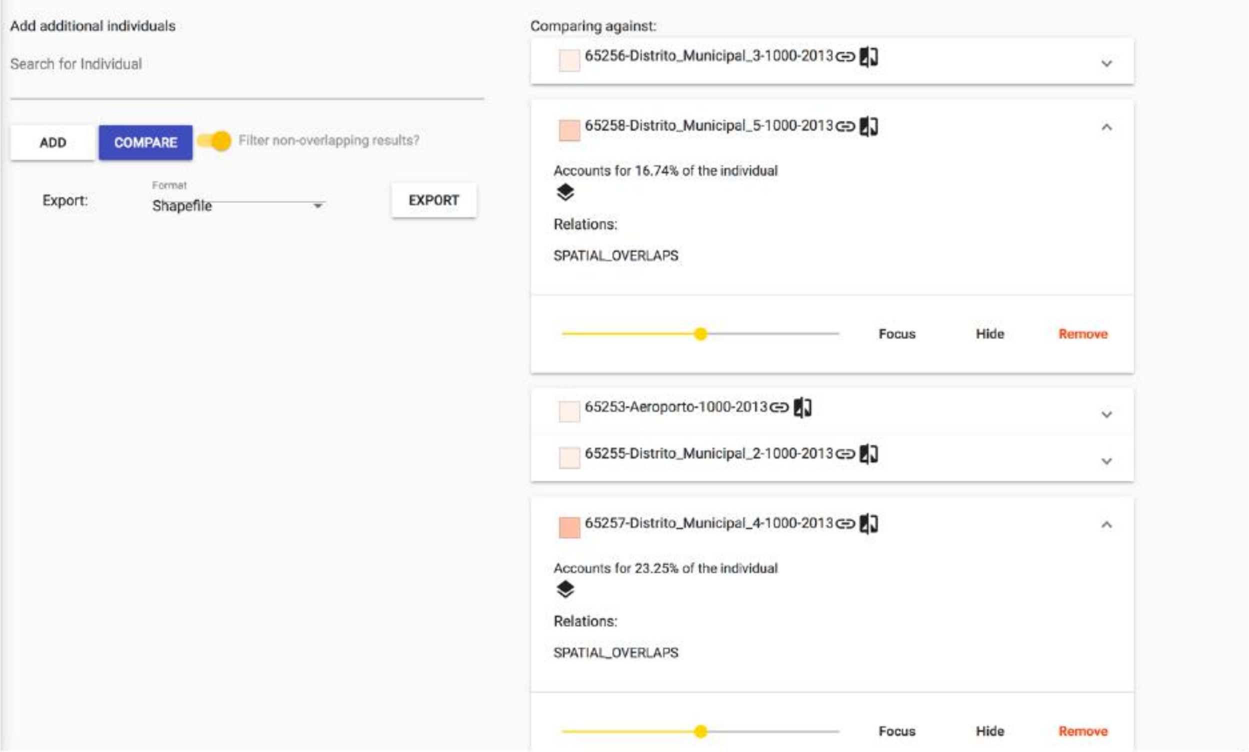


Spatial Compare

Figure 53: Cidade de Maputo with all other spatially interacting Trestle_Objects.

The results of the spatial intersection for *Cidade de Maputo* (drawn in purple) is shown along with all other Trestle_Objects which spatially overlap with its boundary. Maputo is drawn on top of the other objects because it comes temporally *after* the other objects and is thus higher on the Z-axis.

Once the initial intersection is performed, the actual *spatial union calculation* (described in Section 5.3.3) is executed and the results presented on the screen. By default, any non-overlapping Trestle_Objects are removed from the comparison and only overlapping results are shown. Each Trestle_Object is then shaded on the red color scale based on the amount of spatial area contributed to the initial object (Maputo). Figure 54 shows the *table* layout of the comparison results, with each overlapping object listed, along with some additional features to either focus on the object (hide all other objects from view) or focus on the actual spatial overlap by adding it as a new layer on the map. In addition, each result has the ability to link back to the *entity visualization* tool or restart the comparison with that Trestle_Object. Figure 55 shows the results of the calculation, as displayed to the user on the map.



Filter non-overlapping results?

Export: Format: Shapefile

Comparing against:

65256-Distrito_Municipal_3-1000-2013

65258-Distrito_Municipal_5-1000-2013

Accounts for 16.74% of the individual

Relations:

SPATIAL_OVERLAPS

65253-Aeroporto-1000-2013

65255-Distrito_Municipal_2-1000-2013

65257-Distrito_Municipal_4-1000-2013

Accounts for 23.25% of the individual

Relations:

SPATIAL_OVERLAPS

Figure 54: Comparison report of Cidade de Maputo and overlapping Trestle_Objects.

This figure shows the output of the spatial comparison tool for *Cidade de Maputo*. The results of the tool are shown on the right side of the screen listed under each Trestle_Object which overlaps with Maputo. Non-overlapping objects are removed from the results and the boundary of each object is shaded (in red) based on the amount of area contributed to Maputo.

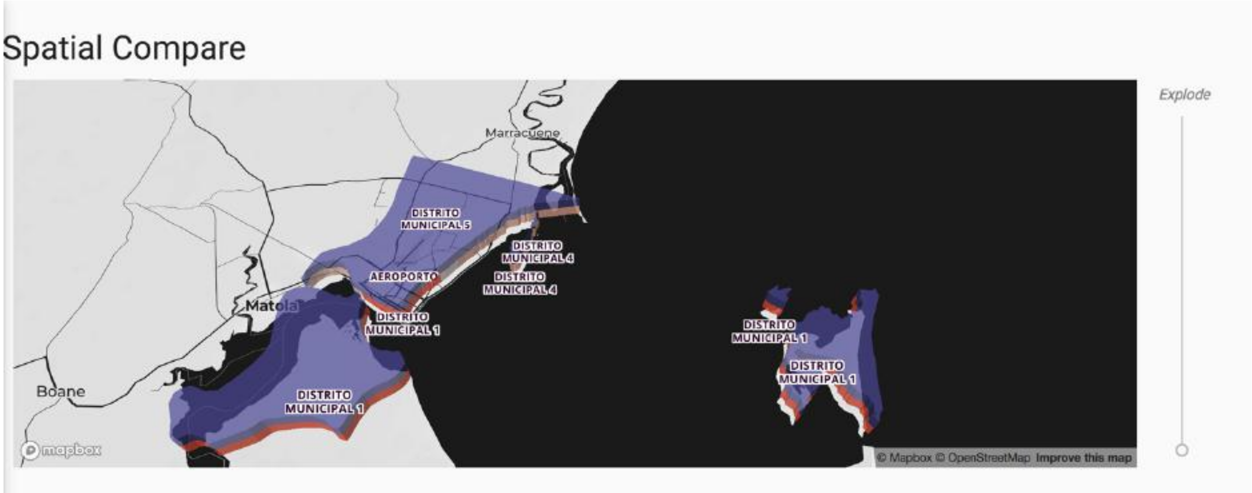


Figure 55: Cidade de Maputo with only spatially overlapping Trestle_Objects.

The output of the spatial comparison tool, with each overlapping Trestle_Object shaded in red, based on how much spatial area is contributed to Maputo.

Once the results have been computed and added to the map, the user now has some additional tools for exploring exactly how the objects spatially and temporally interact, in order to determine whether or not a split/merge event exists. As previously mentioned, the 3D nature of the map view means that the user can not only zoom in and out and rotate the top-down view (as is possible in any 2D map visualization), but they can also adjust the *pitch* (the view angle between the user and the surface of the map) and move the individual Trestle_Objects up and down along the Z-axis to get a better view of the interactions. This is shown in Figure 56, the *explode* slider on the right side of the screen allows moving the comparison object (Maputo) higher or lower on the Z-axis, while each additional object has a *height* slider (shown in Figure 54 as a yellow slider within each object report) that allows for the same functionality for each object. This tool helps confirm that there is indeed a split/merge event between *Cidade de Maputo* and the 6 regions which previously covered the exact same spatial area.

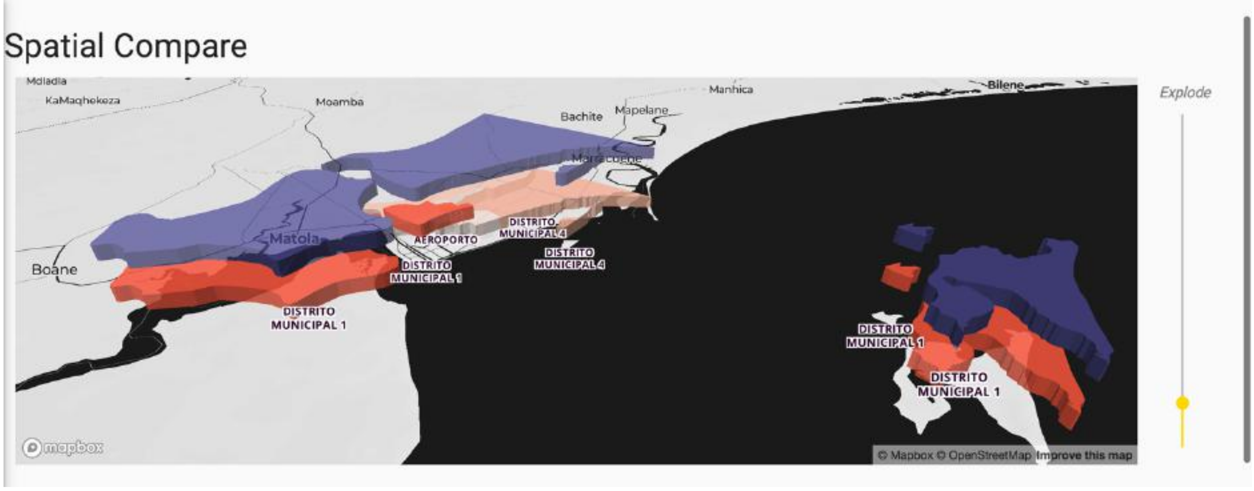


Figure 56: 3D view of Cidade de Maputo and spatially overlapping Trestle_Objects.

This shows *Cidade de Maputo* and all spatially overlapping objects but rotated across 3 axes to give a clearer picture of how *Cidade de Maputo* interacts with the overlapping objects. In addition, both *Cidade de Maputo* and *Distrito Municipal 1* are *elevated* along the Z-axis in order to give more visual separation between the various objects.

9.2 APPENDIX B

This appendix contains the full results for the split/merge algorithm analysis. Column values indicate whether or not a split/merge event was present.

Region	Gold standard	90% cutoff	95% cutoff	97% cutoff	99% cutoff
190498-Mouyondzi-2011-3001	TRUE	TRUE	TRUE	TRUE	TRUE
65256-Distrito Municipal 3-1000-2013	TRUE	TRUE	TRUE	TRUE	TRUE
191189-Omala-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191067-Shira-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191365-Emuoha-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
22913-Yagba West-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
191085-Gulani-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
190487-Ignie-2011-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191013-Anka-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
23027-Kajola-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191253-Abeokuta South-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191108-Kachia-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
23017-Ibadannorth-east-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
191415-Kagarko-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22675-Ningi-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191060-Birnin Kudu-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
190962-Sokoto North-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191361-Isiala Ngwa South-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191300-Amuwo Odofin-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191128-Pankshin-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22997-Egbedore-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
191390-Tai-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
190987-Monguno-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191233-Ibadan North-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191014-Mafa-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
41374-Cidade de Maputo-2013-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191127-Mokwa-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
190461-Mayoko-2011-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22962-Ijebu-Ode-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
190994-Birnin-Magaji/Kiyaw-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191345-Njaba-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191298-Ojo-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191185-Oyo West-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191396-Ikot Abasi-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE

191059-Albasu-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
22855-Rano-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
22788-Isu-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
23020-Ibadansouth-west-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
190503-Loumo-2011-3001	TRUE	TRUE	TRUE	TRUE	TRUE
14458-Lekana-1000-2011	FALSE	FALSE	FALSE	FALSE	FALSE
22952-Shiroro-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
22710-Monguno-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
190986-Charanchi-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191007-Auyo-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191247-Oluyole-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191275-Isi-Uzo-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22744-Warri North-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
23037-Akwanga-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
23022-Ido-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
14466-Ouesso-1000-2011	TRUE	TRUE	TRUE	TRUE	TRUE
191157-Pategi-1999-3001	TRUE	FALSE	FALSE	FALSE	FALSE
22627-Ikot Ekpene-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
22672-Katagum-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
23016-Ibadannorth-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
190980-Guri-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
23118-Fika-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191288-Ikeja-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
190509-Epena-2011-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191218-Oturkpo-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
190970-Dutsi-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191421-Isoko North-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191255-Etsako West-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191086-Yauri-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191319-Njikoka-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
23104-Bali-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
190482-Ongogni-2011-3001	TRUE	TRUE	TRUE	TRUE	TRUE
23035-Oyo-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
22897-Yauri-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191051-Katagum-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
190995-Kankia-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
23066-Ikwerre-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
22885-Argungu-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191019-Gummi-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22636-Oron-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE

190990-Kaura Namoda-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191044-Wudil-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191205-Ajaokuta-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22738-Ndokwawe-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191104-Alkaleri-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191156-Kwali-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191055-Garum Mallam-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22856-Rimin Gado-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191124-Chanchaga-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191402-Ibeno-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191018-Ringim-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
190468-Makabana-2011-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191111-Song-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
190979-Yankwashi-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22682-Apa-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191097-Hawul-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
191168-Ilorin West-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
22756-Ovia South-West-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
22971-Akure-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
191228-Ogori/ Magongo-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
12888-Bozoum-1000-2003	TRUE	TRUE	TRUE	TRUE	TRUE
22695-Oturkpo-1000-1999	TRUE	TRUE	TRUE	TRUE	TRUE
22709-Mobbar-1000-1999	FALSE	FALSE	FALSE	FALSE	FALSE
191343-Ughelli North-1999-3001	FALSE	FALSE	FALSE	FALSE	FALSE
191093-Bayo-1999-3001	TRUE	TRUE	TRUE	TRUE	TRUE
14459-Boko-1000-2011	TRUE	TRUE	TRUE	TRUE	TRUE
True Positive:		69	69	69	69
True Negative:		30	30	30	30
False Positive		0	0	0	0
False Negative		1	1	1	1
FPR		0	0	0	0
TPR		0.985714286	0.985714286	0.985714286	0.985714286

9.3 APPENDIX C

This appendix contains a selection of Java code for combining multiple years of TIGER data into unified Trestle_Objects. It serves as an example of how Trestle might be integrated with existing spatial data stores.

```java
package com.nickrobison.trestle.tigerintegrator;

import com.google.common.collect.ImmutableMap;
import com.nickrobison.trestle.ontology.exceptions.MissingOntologyEntity;
import com.nickrobison.trestle.reasoner.TrestleBuilder;
import com.nickrobison.trestle.reasoner.TrestleReasoner;
import com.nickrobison.trestle.reasoner.exceptions.TrestleClassException;
import com.nickrobison.trestle.reasoner.annotations.DatasetClass;
import com.nickrobison.trestle.reasoner.annotations.IndividualIdentifier;
import com.nickrobison.trestle.reasoner.annotations.Spatial;
import com.nickrobison.trestle.reasoner.annotations.temporal.StartTemporal;
import org.apache.commons.lang3.builder.EqualsBuilder;
import org.apache.commons.lang3.builder.HashCodeBuilder;
import org.checkerframework.checker.nullness.qual.Nullable;
import com.typesafe.config.Config;
import com.typesafe.config.ConfigFactory;
import org.semanticweb.owlapi.model.IRI;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.File;
import java.sql.*;
import java.time.Duration;
import java.time.Instant;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;

/**
 * Created by detwiler on 2/16/17.
 */
public class TigerLoader {
    private static final Logger logger = LoggerFactory.getLogger(TigerLoader.class);
    private static int firstYear = 2011;
    private static int lastYear = 2015;
    // supporting data structures
    /*
        Regions:
        1 = Northeast
        2 = Midwest
        3 = South
        4 = West
     */
    public static final Map<Integer,String> regionMap = ImmutableMap.of(1,
"Northeast", 2, "Midwest",
            3, "South", 4, "West");

    /*
        Divisions:
```

```
        1 = New England
        2 = Middle Atlantic
        3 = East North Central
        4 = West North Central
        5 = South Atlantic
        6 = East South Central
        7 = West South Central
        8 = Mountain
        9 = Pacific
     */
    public static final Map<Integer,String> divisionMap = ImmutableMap.<Integer,
String>builder()
            .put(1, "New England")
            .put(2, "Middle Atlantic")
            .put(3, "East North Central")
            .put(4, "West North Central")
            .put(5, "South Atlantic")
            .put(6, "East South Central")
            .put(7, "West South Central")
            .put(8, "Mountain")
            .put(9, "Pacific")
            .build();

    private Config config;
    private String connectStr;
    private String username;
    private String password;
    private String reponame;
    private String ontLocation;
    private String ontPrefix;
    private List<TigerCountyObject> tigerObjs;

       public class TigerCountyObject {
           private final String geom;
           private final String geoid;
           private final String region;
           private final String division;
           private final String state;
           private final String county;
           private final int pop_estimate;
           private final int births;
           private final int deaths;
           private final int natural_increase;
           private final int international_migration;
           private final int domestic_migration;
           private final float rate_birth;
           private final float rate_death;
           private final float rate_natural_increase;
           private final LocalDate record_start_date;

           public TigerCountyObject(String geoid, String geom, String region, String
       division,
                                    String state, String county, int pop_estimate, int
       births,
                                    int deaths, int natural_increase, int
       international_migration,
                                    int domestic_migration, float rate_birth, float
       rate_death,
                                    float rate_natural_increase, LocalDate start_date)
       {
               this.geoid = geoid;
               this.geom = geom;
               this.region = region;
               this.division = division;
               this.state = state;
               this.county = county;
               this.pop_estimate = pop_estimate;
               this.births = births;
               this.deaths = deaths;
               this.natural_increase = natural_increase;
               this.international_migration = international_migration;
               this.domestic_migration = domestic_migration;
               this.rate_birth = rate_birth;
               this.rate_death = rate_death;
               this.rate_natural_increase = rate_natural_increase;
               this.record_start_date = start_date;
           }

           @Spatial(projection = 4269)
           public String getGeom() {
               return geom;
           }

           @IndividualIdentifier
           public String getGeoid() { return geoid; }

           public String getRegion() {
               return region;
           }

           public String getDivision() {
               return division;
           }

           public String getState() {
               return state;
           }

           public String getCounty() {
               return county;
           }

           public int getPop_estimate() {
               return pop_estimate;
           }

           public int getBirths() {
               return births;
           }

           public int getDeaths() {
               return deaths;
           }

           public int getNatural_increase() {
               return natural_increase;
           }

           public int getInternational_migration() {
               return international_migration;
           }
```

```
           public int getDomestic_migration() {
               return domestic_migration;
           }

           public float getRate_birth() {
               return rate_birth;
           }

           public float getRate_death() {
               return rate_death;
           }

           public float getRate_natural_increase() {
               return rate_natural_increase;
           }

           @StartTemporal(name = "start_date")
           public LocalDate getRecord_start_date() {
               return record_start_date;
           }
       }


    TigerLoader() throws SQLException
    {

        config = ConfigFactory.parseResources("tiger-loader.conf");
        connectStr = config.getString("trestle.graphdb.connection_string");
        username = config.getString("trestle.graphdb.username");
        password = config.getString("trestle.graphdb.password");
        reponame = config.getString("trestle.graphdb.repo_name");
        ontLocation = config.getString("trestle.ontology.location");
        ontPrefix = config.getString("trestle.ontology.prefix");
        tigerObjs = buildObjects();
    }

    public void loadObjects()
    {
        TrestleReasoner reasoner = new TrestleBuilder()
                .withDBConnection(connectStr, username, password)
                .withName(reponame)
                .withOntology(IRI.create(ontLocation))
                .withPrefix(ontPrefix)
                .withInputClasses(TigerCountyObject.class)
                .withoutCaching()
                .initialize()
                .build();

        for(int count=0; count<tigerObjs.size(); count++)
        {
            if(count%1000==0)
                logger.info("Writing trestle object {}", +count);

            TigerCountyObject tigerObj = tigerObjs.get(count);
            try {
                final Instant start = Instant.now();
                reasoner.writeTrestleObject(tigerObj);
//                reasoner.writeTrestleObject(tigerObj,startTemporal,null);
                final Instant end = Instant.now();
                logger.info("Writing object {} took {} ms", count,
Duration.between(start, end).toMillis());
```

```
        } catch (TrestleClassException e) {
            e.printStackTrace();
            System.exit(-1);
        } catch (MissingOntologyEntity e) {
            e.printStackTrace();
            System.exit(-1);
        }
    }

    reasoner.getMetricsEngine().exportData(new File("./tiger.csv"));


    reasoner.shutdown();
}

private List<TigerCountyObject> buildObjects() throws SQLException
{
    String connectStr = config.getString("data_db.connection_string");
    String queryStr = config.getString("data_db.query");
    if(connectStr==null||queryStr==null)
        return null; // should probably throw an exception here

    List<TigerCountyObject> objects = new ArrayList<>();

    Statement stmt = null;
    try {
        Connection conn = DriverManager.getConnection(connectStr);
        stmt = conn.createStatement();
        for(int year=firstYear; year<=lastYear; year++)
        {
            String shapetable = "shp"+year;
            queryStr = queryStr.replaceAll("<shapetable>",shapetable);

            ResultSet rs = stmt.executeQuery(queryStr);
            while(rs.next()) {

                String geom = rs.getString("geotext");
                if(geom==null)
                    continue; // all entries must have spatial data

                String geoid = rs.getString("geoid");

                // convert region code to region name
                int regionCode = rs.getInt("REGION");
                String region = regionMap.get(regionCode);

                // convert division code to division name
                int divisionCode = rs.getInt("DIVISION");
                String division = divisionMap.get(divisionCode);

                String state = rs.getString("STNAME");

                String county = rs.getString("CTYName");

                // get population data
                int pop_estimate = rs.getInt("POPESTIMATE"+year);
                int births = rs.getInt("BIRTHS"+year);
                int deaths = rs.getInt("DEATHS"+year);
                int natural_increase = rs.getInt("NATURALINC"+year);
                int international_migration = rs.getInt("INTERNATIONALMIG"+year);
                int domestic_migration = rs.getInt("DOMESTICMIG"+year);
```

```
                    float rate_birth = rs.getFloat("RBIRTH"+year);
                    float rate_death = rs.getFloat("RDEATH"+year);
                    float rate_natural_increase = rs.getFloat("RNATURALINC"+year);
                    LocalDate record_start_date = LocalDate.of(year,7,1);
                    //LocalDate record_end_date = LocalDate.of(year+1,7,1);

                    // construct Trestle object
                    TigerCountyObject tcObj = new
TigerCountyObject(geoid,geom,region,division,state,county,

pop_estimate,births,deaths,natural_increase,international_migration,domestic_migration
,

rate_birth,rate_death,rate_natural_increase,record_start_date);
                    objects.add(tcObj);

                }
                rs.close();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if(stmt!=null)
                stmt.close();
        }

        return objects;
    }

    public boolean verifyObjects() throws TrestleClassException, MissingOntologyEntity
{
        TrestleReasoner reasoner = new TrestleBuilder()
                .withDBConnection(connectStr, username, password)
                .withName(reponame)
                .withOntology(IRI.create(ontLocation))
                .withPrefix(ontPrefix)
                .withInputClasses(TigerCountyObject.class)
                .withoutCaching()
                .withoutMetrics()
                .build();

        boolean allEquivalent = true;
        for(int count=0; count<tigerObjs.size(); count++)
        {
            TigerCountyObject tigerObj = tigerObjs.get(count);
            String id = tigerObj.getGeoid();
            LocalDate startDate = tigerObj.getRecord_start_date().plusMonths(1);
            TigerCountyObject outObj =
reasoner.readTrestleObject(TigerCountyObject.class, id, startDate, null);

            if(!tigerObj.equals(outObj))
            {
                logger.error("Error, Trestle input object and output object not
equivalent; in:"+tigerObj.getGeoid()+", out:"+outObj.getGeoid());
                allEquivalent = false;
            }
        }

        reasoner.shutdown();

        return allEquivalent;
    }

    public void computeRelations() throws TrestleClassException, MissingOntologyEntity
{
        TrestleReasoner reasoner = new TrestleBuilder()
                .withDBConnection(connectStr, username, password)
                .withName(reponame)
                .withOntology(IRI.create(ontLocation))
                .withPrefix(ontPrefix)
                .withInputClasses(TigerCountyObject.class)
                .withoutMetrics()
                .build();

//        Make a set of unique IDs
        final Set<String> objectIDs = tigerObjs
                .stream()
                .filter(obj -> obj.getState().equals("Washington"))
                .map(TigerCountyObject::getGeoid)
                .collect(Collectors.toSet());

//        Set the computation time
        final LocalDate validAt = LocalDate.of(2013, 8, 1);

        for (String id : objectIDs) {
            final Instant computeStart = Instant.now();
            logger.info("Computing relationships for {}", id);
            reasoner.calculateSpatialAndTemporalRelationships(TigerCountyObject.class,
id, validAt);
            logger.info("Writing relations for object {} took {} ms", id,
Duration.between(computeStart, Instant.now()).toMillis());
        }
    }

    public static void main(String[] args)
    {
        System.out.println("start time: "+Instant.now());
        try {
            TigerLoader loader = new TigerLoader();
            loader.loadObjects();
            loader.computeRelations();
            loader.verifyObjects();
        } catch (SQLException | MissingOntologyEntity | TrestleClassException e) {
            e.printStackTrace();
        }
        System.out.println("end time: "+Instant.now());
    }
```