AN ADAPTIVE FRAMEWORK FOR REAL-TIME SPATIOTEMPORAL BIG DATA ANALYTICS

by

Md Monir Hossain Sharker

B.Sc.(Honors), Electronics and Computer Science, Jahangirnagar University, 1999

M.Sc., Computational Mathematics, Duquesne University, USA, 2010

Submitted to the Graduate Faculty of the

School of Computing and Information in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2017

UNIVERSITY OF PITTSBURGH

SCHOOL OF COMPUTING AND INFORMATION

This dissertation was presented

by

Md Monir Hossain Sharker

It was defended on

August 14, 2017

and approved by

Dr. Rami G. Melhem, Professor, Department of Computer Science

Dr. Marek J. Druzdzel, Professor, Department of Informatics and Networked Systems

Dr. Paul W. Munro, Associate Professor, Department of Informatics and Networked Systems

Dissertation Advisor: Dr. Hassan A. Karimi, Professor, Department of Informatics and Networked Systems

AN ADAPTIVE FRAMEWORK FOR REAL-TIME SPATIOTEMPORAL BIG DATA ANALYTICS

Md Monir Hossain Sharker, PhD

University of Pittsburgh, 2017

Due to advancements in and widespread usage of technologies such as smartphones, satellites, smart sensors, and social networks, collection of spatiotemporal data is growing rapidly. Such massive spatiotemporal data require appropriate techniques and technologies for their efficient analysis and processing. Analyzing massive spatiotemporal data efficiently and effectively is challenging since the data changes dynamically over space and time whereas, often, decisions followed by the analysis need to be made under real-time constraints. Compared to non-spatial data, spatiotemporal data, among other unique characteristics, are multidimensional (x, y, attributes, time) in nature, complex in structures and behaviors, and provides details at different resolutions and scales. These characteristics together make analyzing and processing massive spatiotemporal data in real time a challenging task. Resorting to high-performance computing (HPC) is a common approach for handling this computing challenge but to determine optimal solutions through data and computation analysis, appropriate analytics and computing solutions are needed.

In this dissertation, we proposed a framework which is basically a platform providing spatiotemporal data-intensive analytics for data- and compute-intensive applications that require computation under real-time constraints on given computing resources. The framework is a layered structure consisting of four interrelated components (layers); three on analytics and one on adaptive computing. A graph-based approach is developed as the foundation of the *analytics* components which are: *efficient analytics* – providing acceptable solutions based on current data in the absence of historical data; *predictive analytics* – providing near-optimal solutions by learning from the patterns of historical data and predicting based on the learning; *meta-analytics* – providing optimal solutions by analyzing pattern of past data patterns; and *adaptive computing* that ensures appropriate analytics are applied and computation is completed in real time on available computing resources.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

Since the paradigm shift toward data-centric computing as a way to handle Big Data problem through data mining and machine learning, I have been interested to work toward a long-term goal to have an expert system as a solution. To that end, as a short-term goal, developing a framework for Big Data analytics is a great milestone to achieve. Since analytics of spatiotemporal data are more complex compared to non-spatial data analytics, I started with the former one so that we can generalize it later. The inspiration I got the most to proceed with this research is from my advisor, Professor Hassan A. Karimi, who is also is very interested in this research and has decades of experience in the domain of spatiotemporal data.

I would like to express my sincere gratitude and gratefulness to Dr. Karimi for his patience to guide me with this work, his invaluable advices, encouragement, help, and support during both my bad times and good throughout the course of my research. I am also grateful to the members of my dissertation committee for their advises and comments on my dissertation to improve it. I like to thank my fellow researchers and colleagues who helped me by discussing, challenging, and providing constructive feedback. Most importantly, this work would not be possible without the love, sacrifices, patience, and cooperation from my wonderful family. I dedicate this dissertation to my family.

1.0 INTRODUCTION

We are currently living in the era of 'Big Data'. According to IBM, around 2.5 quintillion ($=2.5x10^{18}$) bytes of data is created every day and the data size is growing exponentially(IBM, 2017a) and most of the data is spatiotemporal in nature. Smart sensors, smartphones, social networks, and satellites are some of the major sources of spatiotemporal data. While capturing, storing, organizing, searching, retrieving, visualizing are some of the major challenges for spatiotemporal big data (Agrawal *et al.*, 2011; Bifet, 2013; Lasinio *et al.*, 2013) developing appropriate analytics for spatiotemporal data-intensive problems is also a major challenge. However, considering the challenges, big data should not be viewed as a problem, rather it has to be embraced as a big resource of opportunities. Advantage of having 'big data' can be achieved, given the computing constraints (e.g., storage, time, computing resources) of applications requirements, by using appropriate analytics in a framework that can adapt with scalability of the data in real time (Furht & Escalante, 2011). As a new approach in handling spatiotemporal Big Data, we propose a framework that recommends computing solutions to real-time applications with spatiotemporal data-intensive tasks given computing resources. In this dissertation, we developed the algorithms as the main components of the framework.

A framework is a platform which basically provides a foundation for applications to use its functionalities so that the application users do not need to reinvent the wheel every time they

need the function(s). Framework is also defined as "*a hierarchical directory that encapsulates shared resources in a single package and serve the same purpose as static and dynamic shared libraries, that is, they provide a library of routines that can be called by an application to perform a specific task*"(Apple, 2014).

Applications handling spatiotemporal big data that require to perform computational tasks in real time using available computing resources will be able to use the proposed framework. The terms used in this dissertation for the framework are briefly explained below.

"Adaptive": The framework is adaptive in that for available computing resources, it is able to scale up or down, if needed, with the data ensuring the most optimum solution.

"Real-Time": Real-time definition here is dependent on the acceptable response time by respective applications. For some applications, response is required "instantly" whereas, for others response is acceptable if available within a given time constraint. In this dissertation, the response is considered "real-time", if it is within a given time constraint acceptable by the application.

"Spatiotemporal Big Data Analytics": Spatiotemporal is a reference to the data that has both spatial and temporal dimensions, along with other (non-spatial) attributes. The data is "Big" if the required analysis and computation on the data cannot be completed in real time using available computing resources. The "data analytics" is the method of analyzing the "spatiotemporal big data" to discover patterns and useful information in any extend of data availability. In this dissertation work, the spatiotemporal big data analytics is the process of examining massive spatiotemporal data to reveal hidden spatiotemporal patterns, unknown correlations, and associated useful information for prioritizing the data so that top priority data could be used for computation that needs to be completed in a given time.

1.1 PROBLEM DESCRIPTION AND CHALLENGES

Analyzing and processing massive spatiotemporal data for applications requiring real-time response are possible if appropriate analytics and required computing resources are available. In reality, explicit modeling of spatial dependencies and associated processing increase computational complexity for spatiotemporal data analytics (Cuzzocrea, 2014; Cuzzocrea *et al.*, 2011). Moreover, if the problem is data- and/or compute-intensive and required to be performed in real time on available (limited) computing resources, the complexities are exacerbated especially if the data changes dynamically over time and locations. The explosive growth in spatiotemporal data, the emergence of social media, advances in location-sensing technologies, and applications requiring real-time response emphasize the need for developing new and computationally efficient analytics tailored to process massive spatiotemporal data in real time.

Most spatial and/or temporal Big Data problems and applications are currently being processed through high-performance computing (HPC); mainly parallel and distributed computing platforms such as clouds, grids, and supercomputers (Di & Wang, 2013). But appropriate HPC resources may not always be available in general and even if available, not many domain scientists and engineers have the required knowledge and expertise in using them effectively. Also, analyzing the data, before applying computation, to make sure that the computation is performed on appropriate data set is important to ensure optimum solution. But, available HPC platforms are mainly designed and developed to provide support with more computing power for faster computation as a solution for data-/compute-intensive problem and those platforms do not provide data analytics support to obtain optimum solution (Engle *et al.*, 2012). These gaps call for development of new approaches that do not depend only on HPC resources but also provide

analytics that are capable of performing analysis on the data and computation in real time using available resources. The analytics are expected to perform two types of analysis: data domain analysis and optimal computation analysis. The data domain could be vector and/or raster having both spatial and temporal dimensions (Sun *et al.*, 2012). To that end, the research question that we aim to address in this dissertation is: What are the algorithms as the main components of a framework that provides computing solutions for spatiotemporal data-intensive tasks under real-time constraint on given computing resources?

The core of the framework is based on two major parts: *analytics* and *adaptive computing*. The analytics part has three components: *efficient analytics*, *predictive analytics*, and *meta-analytics*. The efficient analytics provides acceptable solutions by using recent/current data when sufficient past data is not available. It basically provides an instant solution based on what data is available at a given instance of time over locations. The predictive analytics provides near-optimal solutions by using available historical data, which is usually the case with most spatiotemporal applications, in a learning model to predict next possible patterns of data for decision making in advance. The meta-analytics aims to provide optimal solutions by learning the patterns in data obtained from both efficient analytics and predictive analytics over time to reveal pattern of patterns (if any) which in turn provides feedback to both the efficient analytics and predictive analytics layers for robust computation and prediction respectively. All the analytics components here deal with vector data. The foundation of the analytics is built through a graph-based approach (explained in Chapter 3) where the graph structure captures the relationship among data points and helps preparing the optimal computation schemes with top priority data points.

In the adaptive computing part, the framework provides a methodology to adapt to the characteristics of the data (scale, spatial and/or temporal variabilities) using available computing

resources. The adaptation is performed by taking the higher priority data into account as suitable for available computing resources. The data priorities are determined through each analytics layer. The graph approach also helps the adaptation by providing an optimal computation scheme through which higher priority data points (determined by relationships among data points) are considered for computation as long as the available computing resources can perform computation within given time constraint. By the term "optimal", we mean that given the computing resources and time constraint, our method optimizes its computing load (data/object in this case) and estimating load to provide results on maximum possible load the computing resource can handle. An algorithm triggers the appropriate analytics to adapt the data for available computing resources to provide optimal solution in real time.

The framework is unique in that it offers a workflow that can provide the best possible (acceptable, near-optimal, and optimal) solutions in real time irrespective of data size and available computing resources, mobile devices (e.g., on smartphones) to HPC (e.g., clusters, grid). Since the change pattern of spatiotemporal data is uncertain, framing the change patterns in a workflow is complex. Development of such a framework involves handling some key challenges as follows:

- Providing robust solutions considering the dynamic change of spatiotemporal data and the availability of various computing resources
- Ensuring solutions in real time with least compromise in solution accuracy
- Developing appropriate workflow.

1.2 MOTIVATION

The motivation behind this research stems from the fact that with the advent of ubiquitous location sensing tools and techniques, massive spatiotemporal data is being generated and captured in such a fast rate that existing computing resources cannot efficiently handle them (Karimi, 2014; Zyl, 2014). Massive spatiotemporal data has many applications including location-based services, public health, homeland security, transportation, and global climate change. A survey by "Millennium Project" identified fifteen global challenges facing humanity in 21st century (Millennium-Project, 2017). Out of these fifteen global challenges, seven of them [i.e., 1, 2, 3, 8, 10, 12, and 13 in (Millennium-Project, 2017)] are predominantly associated with spatiotemporal data. According to McKinsey report, use of personal spatial data in real time will save consumers $600 annually by 2020 (Manyika *et al.*, 2011). The size of National Aeronautics and Space Administration (NASA) climate change data only is estimated to be 350 PB by 2030 (Skytland, 2012), Google[1] processes 100+ PB data per day, Facebook[2] handles 30+ PB data per day, trillions of sensors are communicating and populating Internet of Things (IoT) with real-time data (IBM, 2017b) and a major portion of the above mentioned data is spatiotemporal data.

The aforementioned facts highlight some of the examples of how big the spatiotemporal big data problems are. To address the problem an approach needs two main components: appropriate analytics and computing resources. But lack of appropriate analytics, tools, and technologies limit the advantages out of this big spatiotemporal data domain. Currently, advanced analytics, such as Presto (Traverso, 2013), handle spatiotemporal data with same approach as for

[1] www.google.com
[2] www.facebook.com

non-spatial data by ignoring the complex semantics of spatial and temporal dimensions but provides no computation support. For the computing need, advanced computing technologies e.g., SPARK (Spark, 2014; Zaharia *et al.*, 2010), STORM (Storm, 2014) are being used but provides no data analytics support.

Current approaches handle both spatiotemporal data and non-spatial data in same way which ignores the complex semantics of spatial and temporal dimensions and incorrectly handle the uncertainty of spatiotemporal data. To address these shortcomings, a framework for handling the spatiotemporal data by providing both analytics and computing support is a potential approach for solution. Also, lack of efficient non-HPC resources requires a new approach appropriate to handle the big spatiotemporal data on any non-HPC platforms within given time constraint. An important aspect of spatiotemporal data is *uncertainty* (Banerjee & Fuentes, 2012), which stems from the dynamic changes on the data over time and locations, and the new approach should address this aspect as well.

1.3 THE FRAMEWORK

The goal of the proposed framework is to provide a platform that can analyze, scale (down/up), predict, and adapt in response to various computing demands and/or resources to perform necessary computations for any spatiotemporal data-intensive application in real time. There are four components in the architecture of the framework as shown in Figure 1.1: *efficient analytics* for providing acceptable solutions based on current (instant) dataset in the absence of sufficient historical data to learn any pattern; *predictive analytics* for providing near-optimal solutions by

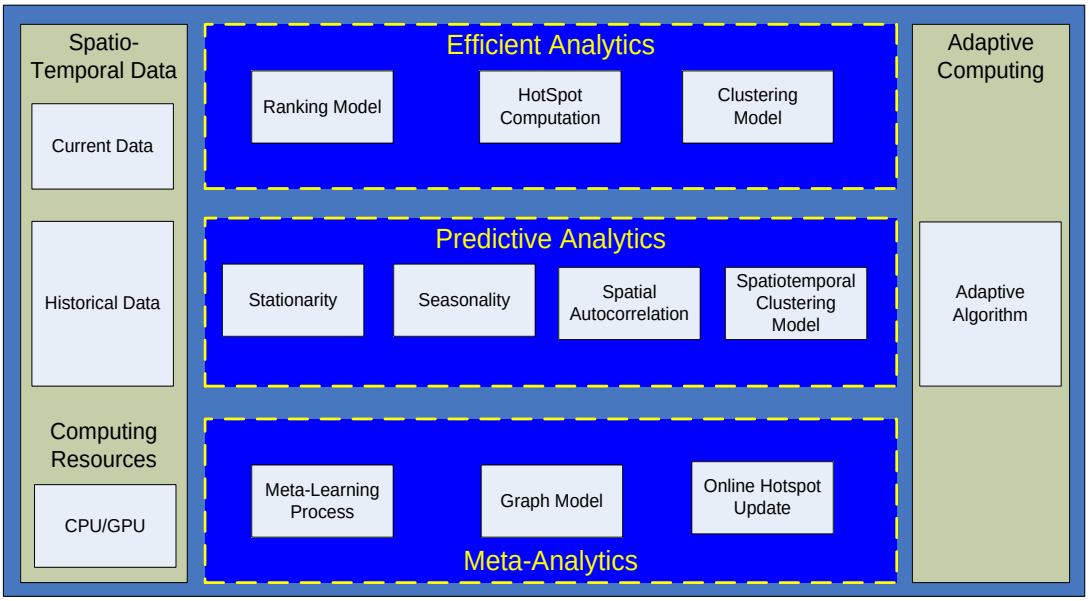


Figure 1.1. Architecture of the spatiotemporal data analytics framework. Input side comprises of spatiotemporal data and given computing resources. Major components are efficient analytics, predictive analytics, and adaptive computing

learning from prior data/pattern and predict future values; *meta-analytics* for providing optimal solutions by detecting/analyzing pattern of patterns for further improving the robustness of efficient and predictive analytics; and an *adaptive computing* component that helps to complete the required computation task(s) on available computing resources in real time.

Each of the module in *efficient analytics* (Ranking model, Hotspot Computation, and Clustering model), *predictive analytics* (Stationarity, Seasonality, Spatial Autocorrelation, and Spatiotemporal Clustering), *and meta-analytics* (Meta-Learning, Graph Model, and Hotspot Updating) is explained in Chapter 4, Chapter 5, and Chapter 6, respectively. The adaptive computing algorithm determines the appropriate analytics by considering application data and computational requirements. A graph-based approach is developed as the foundation that helps the

analytics and the adapting computing algorithm by performing a set of pre-processing tasks to prioritize and optimize the dataset that should be used for computation.

Along with the above mentioned four components, the framework has some other associated and/or inherent components and methods that complement the analytics. The components and methods of the framework are briefly explained bellow.

1.3.1 Input and methods

1.3.1.1 Input

The framework requires four input parameters: spatiotemporal data, computing resources, time constraint, and the required real-time computing task. The computing task is the main computation function to be performed on the input data.

Spatiotemporal data: Spatiotemporal data is any time series data collected over different geographic locations. The data could be produced/collected in regular time interval or irregular time stamps, stationary (fixed rate of change) or non-stationary, and raster or vector. The framework and its components are designed for spatiotemporal vector data. The locations could also be at regular distance interval or randomly located and could be fixed locations or locations which vary over time.

Computing resources: CPU/GPU on mobile devices, on desktops, on centralized/distributed servers, clouds, and supercomputers are some example computing resources. Processor's specification (e.g., MIPS, MFLOPS), number of processors/cores, and latency (if any) are necessary for computing the required computation time. The assumption is that

computing resource specification is available in terms of computation time (T_s) per data points/objects (for given data and computing task) and the total available resource time (T_r).

Time constraint: Time constraint is the upper bound of time (T_0) within which the computation must be completed. Time constraint depends on application requirements; in some cases, it could be fraction of a second whereas, in other cases, it could be hours.

Computing task (f_c): The computing task is the main computation to be performed in real time (within the time constraint T_0) on the dataset. The nature of each computing task and its complexities depend on application characteristics. The main challenge is to complete the computing task within time constraint T_0, especially for data-intensive applications, using available (limited) computing resources.

1.3.1.2 Methods: functions

Ranking Method (f_r): A time window-based variance method is developed to compute the variance in each window over the time range as data is available. The data points/objects are ranked according to the variance of the time windows; higher the variance, higher the rank. The time window-based variance method is used to identify hotspots, the high-ranking data points, in the input dataset. The assumption is that the dataset is very large and the computation must be performed in real time. The objective is to rank the input data based on the order of variability of data over time and location (Cressie & Johannesson, 2008). In this method, the variance of the spatiotemporal data in each time window (there will be a group of data points in each window) is computed and then the groups are rank ordered according to the window variance. Based on the rank order, top ranking data points/objects are considered as the priority candidate input data which is termed as spatiotemporal "hotspots" (Chainey *et al.*, 2008).

Prediction Method (f_p): Prediction method is designed to perform two types of prediction: 1) *Data value prediction,* and 2) *Cluster prediction*. In the data value prediction method, the data pattern, if exists, is learned by using time series spectral analysis (Banerjee *et al.*, 2008; Box *et al.*, 2015) that is performed to analyze the data spectra to learn about the data distribution over time. Using non-parametric regression techniques (Salcedo *et al.*, 1999), another form of linear filtering, the time series data is decomposed into temporal groups to evaluate the trend of the time series. Based on the learning from the past data, an appropriate prediction method is developed using temporal (seasonal) Auto Regressive Integrated Moving-Average (ARIMA) model with both temporal and non-temporal factors. The temporal (seasonal) term here does not necessarily refer to seasons of a year but could be any time interval as appropriate for a given spatiotemporal dataset. The temporal parameters are determined based on applications' requirements, frequency of data, and availability of data over time. The general form of the model is ARIMA(p,d,q)x(P,D,Q)$_s$, where (p,d,q) is the order of the non-temporal part of the ARIMA model with p autoregressive lags, q moving average lags, d order of difference, and (P,D,Q)s is the order of the temporal part with *s* as the number of observations in each season. An appropriate prediction model is derived by determining the parameter estimates as suitable for a given spatiotemporal dataset.

In the cluster prediction method, the temporal patterns of spatial clusters are used to train the aforementioned prediction model so that the clusters could be estimated along with the relationships (*correlations*) among the clusters.

Clustering Method (f_{cl}): A spatial clustering method is used to cluster the spatiotemporal data points/objects to make sure that the entire spatial domain is represented in the "hotspot". Then, within each cluster, the "hotspots" are identified using the ranking method (f_r). A widely used

unsupervised clustering technique, K-means (MacQueen, 1967), is used for the spatial clustering. Random initialization method is used (as shown in the following steps) to ensure unbiased choices.

1. Randomly select K data points, from the data set, as initial centroids of K clusters.
2. Assign each data point to the closest cluster centroid.
3. Compute new cluster centroid for each cluster.
4. Repeat Steps 2 and 3 until the centroids no longer change.

Optimum Target Data Selection Method (f_{op}): When the required computing need is beyond the computing capacity of available resources, this method chooses an optimal number of top priority data points/objects from the ranked candidate list. The candidates are determined from the combination of ranking and/or clustering (explained in Section 3.1.2). This method considers that different computers may have same (homogenous) or different (heterogeneous) computing capability (number of data points that a node can handle per unit time). Based on the combined computing capability of each available computer, the method determines the top candidate data points/objects out of the ranked list that can be used to complete the computation in real time.

1.4 RESEARCH CONTRIBUTIONS

The main contribution of this dissertation work is a framework for real-time spatiotemporal data-intensive processing that can adapt to various computing resources by considering optimal workloads. The outcome of the framework is a workflow that accomplishes the adaptation by identifying spatiotemporal hotspots (priority data points) and by determining the priority hotspots

that should be computed on given computing resources to ensure real-time response. The core of the framework is a graph-based approach followed by three analytics (covering all data availability situations) where each of the analytics has a unique methodology to analyze the data and prepare for the adaptation algorithm. In summary, the contributions of the dissertation are:

- A graph-based approach to construct the foundation of the analytics
- Development of a methodology for spatiotemporal hotspot identification
- Development of efficient analytics
- Development of predictive analytics
- Development of meta-analytics
- An adaptation method to ensure scalability and real-time response
- Development of appropriate workflows

1.5 POTENTIAL APPLICATIONS

Potential applications (Eshel, 2011) of this research include:

- Transportation: traffic monitoring, control, tracking vehicle movement, traffic planning, vehicle navigation, fuel efficient routes
- Mobility: routing and navigation, context-aware routing, GPS trajectory analysis and tracking
- Location-based services: receiving alerts, such as notification flash flood, weather alert, traffic warning

- Sensor network: water/air pollution, chemical hazard monitoring, traffic monitoring, air/water flow
- Meteorology: weather prediction, development and moving storms/tornados updates, tracking of high pressure zones, precipitation areas, changes in freezing level
- Health: detection of epidemic using health data, seasonal health issues like allergy, flu, development of cancer, recommending health-optimal routing

1.6 ORGANIZATION OF THE DISSERTATION

The dissertation is organized as follows. Chapter 2 describes the background and related works of the framework. Chapter 3 provides methods for adapting computation by preprocessing and optimum data selection, using graph approach, for given computing resources. Chapter 4 describes the details of Efficient Analytics, Chapter 5 explains the core approaches and methods in Predictive Analytics, Chapter 6 presents methods in Meta-Analytics, and Chapter 7 presents experiments, results, and illustrates the validation and provides detail discussions. Chapter 8 provides the Workflow, Summary and Conclusions, and the Future research directions of the dissertation.

2.0 BACKGROUND AND RELATED WORKS

This chapter provides the background of spatiotemporal big data, its characteristics, complexities, and grouping of available data types. A detail literature review and comparison of available other related frameworks are provided in the related works section.

2.1. SPATIOTEMPORAL BIG DATA: CHARACTERISTICS AND COMPLEXITIES

Compared to non-spatial data or time-series data, spatiotemporal data has two important attributes: time and location, in addition to other non-spatial attributes. That is, spatiotemporal data, among other unique characteristics, are multidimensional (x, y, attribute, time) in nature, hence complex in structures and behaviors, and provides details at different levels of resolutions and scales. The behavior of any variable in spatial domain differs from its behavior in time domain since the time domain has its own chronological sequences while the location does not have such ordering. The time interval (periodicity) in temporal data is the feature that is not available in spatial data which is why the scales of measurement are also different. Also, temporal data can be analyzed taking the measurement only in one side of the axis which requires extrapolation for prediction instead of interpolation. Accordingly, data model, data structure, storage structure, processing methodologies, analysis metrics and statistics of spatiotemporal data pose additional complexities. Also, the grouping approach of different data types are different (Shekhar *et al.*, 2015) based on the attributes

(non-spatial, spatial, temporal, and spatiotemporal) that the data is grouped on. Different such grouping approaches are shown in Table 2.1.

Table 2.1. Grouping approach for different data types – non-spatial, spatial, and spatiotemporal data

Data Types	Grouping approach	Summarization/inference approach
Non-spatial data	Data points, rows, records	Aggregate statistics: mean, median, mode, sum, count etc.
Spatial Data	Proximity; Euclidean distance	Centroids, medoids, inter-cluster and intra-cluster similarity
Spatiotemporal data	Space-time(ST) cluster	Both spatial and temporal statistics.

There are three models of spatial data: the object model (point, line, polygon), the field model (regular, irregular), and the spatial network model (graph) (Shekhar & Chawla, 2003; Worboys & Duckham, 2004). Spatiotemporal data has additionally the temporal dimension in the spatial data. To incorporate the temporal attribute (dimension) in the spatial domain, spatiotemporal data are distinguished into three models: temporal snapshot model, temporal change model, and event or process model (Li *et al.*, 2008; Yuan, 1996). Temporal snapshot model includes timestamp on same spatial layers at different times. For example, temporal snapshots for point data (object model) are trajectories or a spatial time series. In temporal change model, spatiotemporal data is represented by a spatial layer at its initiation time followed by incremental changes over time. Example of temporal change model would be representing motion, such as Brownian motion (Gelfand *et al.*, 2010), along with speed acceleration on location, rotation and deformation on lines and polygons. In event and process models, temporal information is represented as events or processes where events do not change over time but the processes change (e.g., fast, slow) over time (Campelo & Bennett, 2013).

From clustering perspectives, different approaches are followed for clustering thematic (non-spatial/temporal attributes), spatial, and temporal domains. In thematic domain, variants of k-means clustering is widely used, whereas for spatial clustering properties such as spatial density, proximity are used for clustering. For example, DBSCAN and BIRCH (Ester *et al.*, 1996) are two popular such spatial clustering techniques. For group time series data, temporal and thematic attributes are used together for clustering to determine the changes over time. However, considering all the spatial, temporal, and thematic attributes seamlessly and simultaneously in the same clustering process is very complex (Shekhar *et al.*, 2015). This is because capturing the dynamic changes in data is the most difficult challenge in spatiotemporal clustering which is why traditional clustering algorithms cannot be directly applied to data having spatiotemporal phenomena. Spatiotemporal scan statistics is one such method that is used for clustering to capture the dynamicity of spatiotemporal data (Shekhar *et al.*, 2015).

2.2. RELATED WORKS

Data-intensive computing is common in many problems, in particular those in the spatiotemporal domain. Current computing techniques and tools fall short of handling efficiently the situations when the data size scales up. As of now, there is no framework or platform directly related or similar to the proposed framework. However, in the analytics side, many approaches including subsampling, likelihood and covariance function approximations, process convolutions, Markov random field representations, spectral methods, stochastic partial differential equations (SPDE's), projections in a subspace of reduced dimension and non-parametric methods are proposed to

handle large databases. Despite the large number and diversity of analytic approaches, none is able to handle the real-time computation challenge of data-intensive problems (Sun *et al.*, 2012). Of these approaches, finding a suitable subsample of the entire dataset may be considered as a first attempt to handle the data-intensive computation problems. Though sampling may not always be a perfect representation of the whole population, an appropriate subset of the whole dataset may be used for inference statistically for the entire dataset(Banerjee *et al.*, 2008), and the precision of the result depends on appropriateness of the sampling approach. For example, in a moving window approach (Haas, 1995), the local covariance functions, fitted within the windows, result in incompatible covariance for larger data.

In relation to the predictive analytics approach, Autoregressive Integrated Moving Average (ARIMA) based models have been widely applied as a popular linear model for time series forecasting in many applications. ARIMA is a very effective analytic tool for modeling the empirical dependencies between successive times and failures with satisfactory performance (Ho & Xie, 1998; Walls & Bendell, 1987). A case study on seasonal ARIMA for forecasting air pollution index shows the suitability of using ARIMA for the forecasting (M. H. Lee *et al.*, 2012), though the method did not address the spatial aspect of the temporal data and the real-time computation constraints. In studies on time series data, the general assumption is that there is a linear correlation structure among the time series values. But in complex real-world problems, this linear relationship is not always common due to the fact that such relationship is usually nonlinear. This nonlinear relationship needs to be learnt accordingly since the approximation of linear models to explain complex real-world problems is not always satisfactory (Shi *et al.*, 1999).

Artificial neural networks (ANNs) can function in simple pattern recognition and can be applied to a wide range of application areas including forecasting applications (Shi et al., 1999).

One of the major advantages of ANNs is their flexibility of modeling nonlinear situations. Besides, the model is constructed adaptively based on the features learned from the pattern in the data. This data-driven approach is suitable for many datasets where no theoretical foundation is available. Many empirical studies, including several large-scale forecasting, have deployed and used the ANNs in time series forecasting.

Both ARIMA and ANNs are nonparametric techniques and are very similar in approach for time series data analytics. ANNs have been compared with traditional time series techniques in several studies especially in data-intensive problems. But using ANNs technique for data-intensive problems is challenging to address in many applications, in particular those in the spatiotemporal domain. While new techniques and tools are coping with different challenges (e.g., storage, management) of data-intensive problems, appropriate data analytics is of particular importance (Cuzzocrea *et al.*, 2011; Manyika *et al.*, 2011), especially in real-time forecasting.

A hybrid forecasting model (SARIMABP) combining the seasonal time series ARIMA (SARIMA) and the neural network back propagation (BP) model have been considered to predict seasonal time series data (Tseng *et al.*, 2002; Wang *et al.*, 2013). Their results demonstrated that the hybrid model produced better forecasts than both the ARIMA model and the neural network when applied separately. All of the metrics such as accuracies, MSE (Mean Square Error), MAE (Mean Absolute Error) and MAPE (Mean Absolute Percentage Error), were better for the SARIMABP model than those in models applying them separately. These empirical results illustrated the hybrid model's advantage over the models in isolation.

Autoregressive integrated moving average and ANNs possess differentiated characteristics where the former fits better for linear prediction, whereas the latter for nonlinear prediction. In fact, because of the complexity associated with the historical data and uncertainty, data are usually

composed of both linear and nonlinear components. Hence, a combined model based on ARIMA and ANNs provides a better option to improve the forecasting accuracy. However, combining the ARIMA and ANNs models suffers from errors caused through the integration, which likely propagates in subsequent forecasting.

For addressing the computation challenge, many approaches are in place though their suitability for very large datasets is unclear. A low-rank representation, to obtain a dimension reduction of the covariance matrix, which maps the original process into a new lower dimensional process decreases computational load but is limited to stationary process (Cressie & Johannesson, 2008). Similarly, dimensionality reduction is also obtained by knot model (Banerjee *et al.*, 2008) but the approximation properties in the model are highly sensitive to the number and location of the knots. Another approach to handle the data-intensive computation is optimizing the computing resources. For example, a MapReduce-based cloud service model, *Cura*, provides global resource optimization for data analytics on cloud (Palanisamy *et al.*, 2013) but it focuses only on improving MapReduce performance which does not guarantee to complete the computation in real time.

A decentralized framework approach was proposed to handle hotspots in applications, like distributed stream processing, in real time (Repantis & Kalogeraki, 2008). But decentralized framework is not well suited to spatiotemporal data or dynamic time series data since it does not consider the spatial and temporal variation. Hotspot mapping is another technique which is used for predicting spatial patterns by means of identifying and learning from past spatial hotspots (Chainey *et al.*, 2008) based on time invariant attributes. But current hotspot mapping techniques do not address the spatiotemporal phenomena and the real-time computation challenge.

In terms of related frameworks, different open source frameworks are available for handling different types of Big Data problems. Different types of frameworks are required to run

different types of analytics, processing, and computations. For example, MapReduce-based frameworks like Hadoop (Hadoop, 2017) is available for batch data processing and analytics. Hadoop is an open source framework that uses simple programing models to provide distributed processing on Big Data over clusters of computers. But Hadoop is incapable of real-time data processing and stream processing. A MapReduce extension of apache Hadoop that supports spatial functions like range queries, kNN queries, and spatial join is proposed as SpatialHadoop (Eldawy & Mokbel, 2015). But SpatialHadoop it is not an analytics framework for spatiotemporal data. The advantage of using SpatialHadoop is that spatial function that conventional GIS software cannot execute because of the large data size, SpatialHadoop can executes those spatial functions in a Hadoop environment. It is particularly focused on spatial attributes and features but not suitable to handle the temporal dimension along with the spatial features.

Table 2.2. Comparison among features of available frameworks for Big Data processing

Features	**Hadoop**	**Spatial Hadoop**	**STORM**	**Apache Drill**	**SPARK**
Developer	Open	Open	Open	Open	Open
Source/License	Open	Open	Open	Open	Open
Processing Loads	Batch	Batch	Real time	Interactive and Ad-hoc	Real time
Latency	High	High	Low	Low	Low
Complexity	Simple	Moderate	Simple	Complex	Simple
Spatiotemporal Analysis	**No**	**No**	**No**	**No**	**No**

For stream processing and real-time computation, Storm (Storm, 2014) is one of the very widely used frameworks. Storm has primitives providing efficient module for parallel real-time computation. Apache Drill (Apache-Drill, 2016; Hausenblas & Nadeau, 2013) is another open source framework, capable to manage Big Data even in petabyte scale over thousands of machines,

used for interactive ad-hoc query and analysis with low latency. For a large-scale data processing, Spark (Spark, 2014) is a general open-source cluster computing framework engine that can run computing module 100 times faster than Hadoop MapReduce in memory and 10 times faster in disk. A comparison among the available open-source frameworks, as mentioned above, is provided in Table 2.2. Though all of these frameworks deal with Big Data processing, none of those frameworks handles spatiotemporal Big Data taking both the spatial and temporal dimension into consideration.

3.0 ADAPTIVE COMPUTING

Adapting with available computing resources in terms of selecting effective set of data load to be computed is a key feature of the proposed framework. The selection process of the data to be fed into computation takes four key factors into consideration: available computing resources, time constraint, computing task, and priority of the data. In this chapter, a graph approach is presented along with a process model followed by a method to determine the nodes in the graph that could be computed and the nodes that should be estimated by adapting with available computing resources. The graph approach is predominantly important for the predictive analytics and meta-analytics (Chapter 5 and Chapter 6). The adaptation for efficient analytics is performed through ranking and associated algorithm for selection of ranked data based on different cases of data availability as explained in Chapter 4.

Algorithm 3.1. Preprocessing in the framework; takes spatiotemporal data and generates spatial clusters

Preprocess()
Input: Spatiotemporal Data(SD)
Output: A complete graph(G)

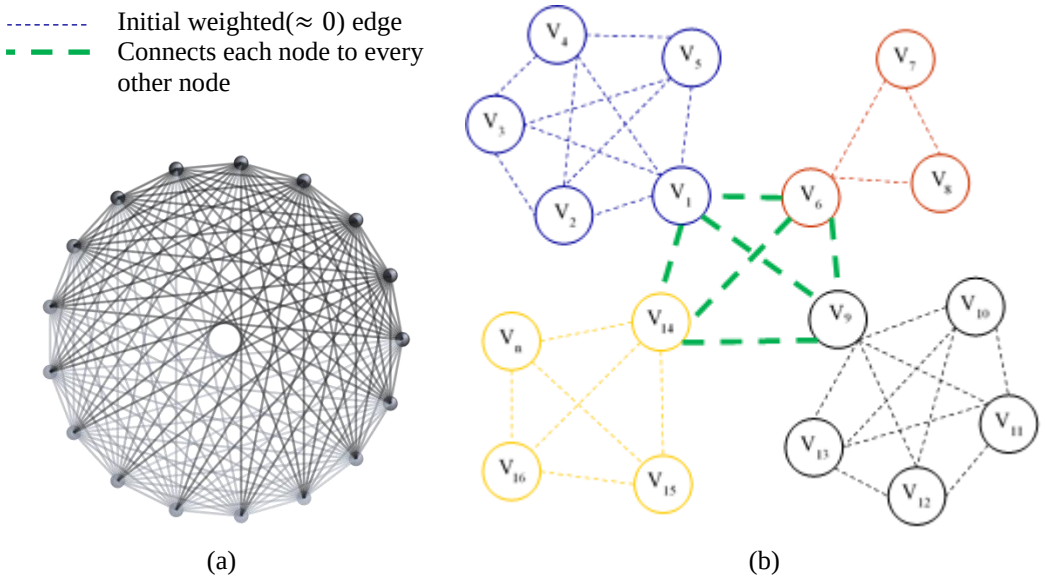
1. begin
2. d ← SD
3. k ← sqrt(len(SD)/2) // number of spatial clusters
4. T ← (IBM, 2017a tn) // timestamps
5. for each t in T
6. $d_r \leftarrow f_r(d)$ // rank order the data in order of significance using the ranking model (f_r)
7. $C \leftarrow f_{cl}(d_r)$ // C = {C_1, C_2, ……., C_k}, compute k spatial clusters
8. $if\ range(T) \geq T_h$ // T_h ← time span sufficient to provide pattern in data
9. $V \leftarrow \{C ; v_i \leftarrow C_i; i = 1, 2, \ldots \ldots \ldots, k\}$ // clusters as node
10. $E \leftarrow$ {edge e_{ij} between $v_i, v_j \in V; w(e_{ij}) \approx 0$ initially} // initial weighted edges
11. $G_t \leftarrow G(V, E)$ // create complete weighted graph
12. end if
13. end
14. end

3.1 PREPROCESSING

The preprocessing tasks prepare the data and create a graph structure so that subsequent analytics can be performed in real time. Preprocessing is performed on sufficient historical data as available and the outcome is used both in predictive analytics and meta-analytics. For each timestamp, dataset is first rank ordered by using the ranking method (f_r). Then by using the clustering method (f_{cl}), a set of k spatial clusters are created (see Algorithm 3.1).

The tasks performed in the preprocessing step are shown in Algorithm 3.1. The input for preprocessing is the spatiotemporal dataset (SD). At first SD is rank ordered using moving window based variance method (f_r) to determine hotspots. For each timestamp in the temporal range of SD, the data points are clustered on spatial domain using the clustering method explained above. Since the data in each cluster is already rank ordered, the hotspots within the clusters are also identified. A complete weighted graph (G = [V, E]) is then created (step 9 through step 11 in Algorithm 3.1) where each cluster is considered as a node of G and all the edges are initiated with a very small (≈ 0) weight since this weight would be modified in the later based on the inter-cluster correlation. The time complexity of the preprocessing algorithm is $O(n^{dk+2}\log n)$ where mostly the clustering task, $O(n^{dk+1}\log n)$, is performed for each timestamp(n). An instance of such a graph is shown in Figure 3.1. The grouping of the nodes initially carries no significance since all the edge weights are the same (≈ 0).

A complete weighted graph (see Figure 3.1) is created using the spatial clusters where each cluster is considered as node and the edges are initiated with a very small weight (≈ 0). Similarly, a graph structure for each timestamp is created and the process provides a series of graphs over time. This series of graphs are the input to the predictive analytics and meta-analytics.



(a) (b)

Figure 3.1. Initial complete graph at a timestamp. (a). The complete graph for n(=17 here); (b) A simplified visualization for the complete graph; the green dotted edges represent connection of each node with every other nodes among the groups of node and the thin dotted edges show connections (weight ≈ **0**) among the nodes

3.2 GRAPH APPROACH

In Graph approach, spatiotemporal data is first clustered using the f_{cl} method as in the preprocessing steps providing an initial complete weighted graph at each timestamp. With available computing resources, if the required computing task(s) is/are not possible to complete for all the clusters within required time constraint, graph approach offers an optimum way to perform the task. In that case, computation is conducted for as many clusters as possible and estimation is used for the rest of the clusters based on their relations with the computed clusters. The idea of spatial clustering is to ensure that the entire geospatial area in the data is represented. Because, if the clustering was performed based on non-spatial attribute in the data set, then only the high ranking data would be selected for computation and this may ignore data from certain

geolocations (with lower ranking) for any chance to be selected for computation. The rank of each data points/object, as computed by the ranking method, is carried over the clusters so that the significance of each data point/object could be used to represent the summary statistics for data in each cluster and determine inter-cluster correlation. Once the spatial clusters are computed, a complete weighted graph is constructed by denoting each cluster as a vertex and assigning very small (≈ 0) initial edge weight for each edge which means that the weight is not updated by using correlation coefficient initially. The weight of the edge between any two nodes is updated using the inter-cluster correlations between the nodes.

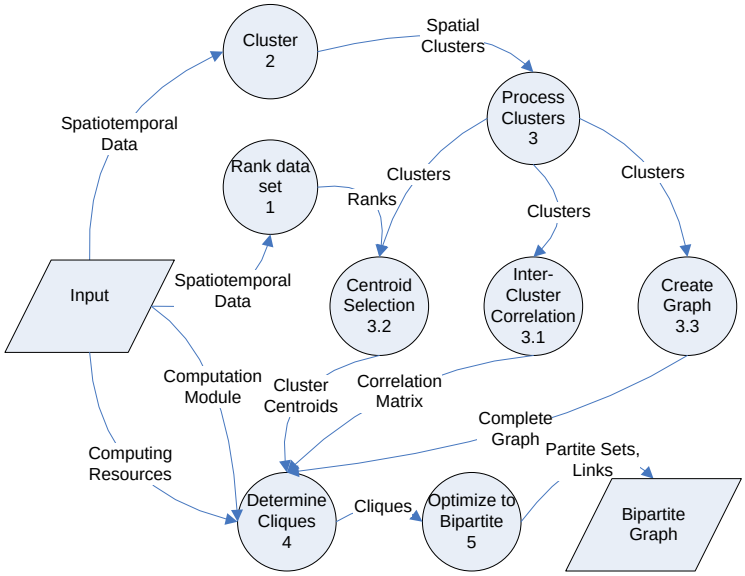


Figure 3.2. Process diagram for correlated candidate selection. The rhombuses are input and output, the circles are processes involved (numbered to indicate order), and the labeled directed lines are data/parameter transfer

Based on the edge weights, the graph is further processed to group highly correlated nodes together as cliques in which the edge weights are above a threshold weight. The total number of nodes that can be computed, computing nodes (CN), are chosen from each clique proportional to

the clique sizes. Once the computing nodes within each clique are determined based on the top ranks, the remaining nodes are chosen as estimating node (EN). Choosing the edge with the highest weight from each estimating node to one of the computing nodes, a bipartite graph within each clique is extracted (see the process diagram as in Figure 3.2). The process diagram demonstrates sequential processes (as numbered 1, 2, 3(3.1, 3.2, 3.3), 4, and 5 in Figure 3.2) to provide the data prepared for computation and prediction. This initial graph process is applied to the data obtained at each timestamp and provides the basis of the analytics.

3.2.1 Determining Computing Nodes (CN) and Estimating Nodes (EN)

The ranked ordered dataset is further analyzed based on correlations among data over geospatial locations. The idea is that if a group of data in one area is highly correlated with a group of data from the same source at a different area, then the computed outcome of the same computing task(f_c) applied on both groups should also be correlated. In such cases, the outcome of computing one of the groups can be used to estimate the outcome of the correlated group. To determine the computing nodes (CN) and estimating nodes (EN) in the graph, for given computing resources let,

$$t_c \leftarrow computing\ time\ per\ cluster\ for\ computing\ task\ f_c$$

$t_e \leftarrow estimating\ time\ per\ cluster\ \ prediction\ method\ f_p$; where, $t_c \gg t_e$

If m be the number of nodes that can be computed out of total k nodes in graph G, the optimization equation to complete the computation and estimation in time T_0 is:

$$mt_c + (k-m)t_e \le T_0 \quad \ldots\ldots\ldots (3.1)$$

Simplifying for m,

$$m \leq \frac{T_0 - kt_e}{t_c - t_e} : T_0 \gg kt_e \; as \; optimized \; by \; k \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.2)$$

Number of clusters that should be estimated is $(k - m)$

3.2.2 Adaptation with Data, Tasks, and Computing Resources

Adaptive computing (AC) component adapts the ranked spatiotemporal data with available computing resources for performing computing tasks of applications and to complete within given time constraint. As illustrated in Algorithm 3.2, AC first determines if sufficient historical data is available to examine pattern. If not, number of data points (*Ct*) that could be computed within time constraint (T_0) on available computing resources is determined and the top ranked *Ct* data points for computation in efficient analytics are selected. If sufficient historical data is available; spatiotemporal clusters are generated, the algorithm optimizes the number of clusters (m) that could be computed and returns the top ranked m clusters for predictive analytics.

This algorithm considers that different computers may have the same (homogenous) or different (heterogeneous) (Gutierrez-Garcia & Sim, 2012) computing capability (number of data points that a computer can handle per unit time). Based on the combined computing capability of individual computer (or computing unit in distributed computing system), the algorithm determines the top priority data points or clusters in the given input it can take into account to complete the computation within given time constraint.

Algorithm 3.2. Finding optimum candidate data points or clusters.

OptimizeData_f_{op}()

Input: Candidate Record Set (RS), Spatial Clusters(C), Number of computing unit available(N), Computing Time(t_c) per data point, Estimating Time(t_e) per data point, Time Constraint(T_0)

Output: Top Candidate Data points(n) to be computed

1. start
2. $if\ range(T(RS)) < T_h$ $//T_h \leftarrow$ time span sufficient to provide pattern in data
3. $i \leftarrow 0; SumRS \leftarrow 0$
4. $while(i < N)$
5. $SumRS \leftarrow SumRS + T_0/t_c[i]$ $//t_c[i] \leftarrow$ computing time per data point in i^{th} node
6. end
7. $Ct \leftarrow Count(SumRS)$// size of data points to compute
8. $TopRS \leftarrow Top(RS, Ct)$//top ranked data points to compute
9. $return\ TopRS, Ct$
10. $else$
11. $k \leftarrow len(C)$ //number of candidate clusters
12. $m \leftarrow \frac{T_0 - kt_e}{t_c - t_e}$ // number of clusters could be computed
13. $TopClust \leftarrow Top(C, m)$ //top ranked clusters to compute
14. $return\ TopClust, m$
15. end if
16. end

The algorithm starts determining the temporal range of the data (step 2) for pattern (if any) in the data to identify. For lacking of historical data availability, Step 2 through step 9 determines the amount of ranked data possible to compute in the given time. For available historical data, step 11 through step 14, rank ordered clusters (and counts of CN and counts of EN) are determined and returned to the calling location (Predictive analytics and Meta-analytics). The time complexity of the algorithm is O(n).

3.2.3 Outcome of Graph Approach

In this approach, the edge weights of the constructed graph are updated by using the inter-cluster correlation coefficients between corresponding nodes. Since computing task (f_c) may not be possible to be completed for all of the nodes on available computing resources within given time constraint, it is necessary to determine the number of nodes that can be computed so that the rest of the nodes can be estimated based on the correlations with the computing nodes. However, it should be ensured that size of CN is at least equal to the number of cliques in graph G by controlling the edge weight threshold while generating the cliques. Number of total possible CN (and EN) can be determined using the method in Algorithm 3.2.

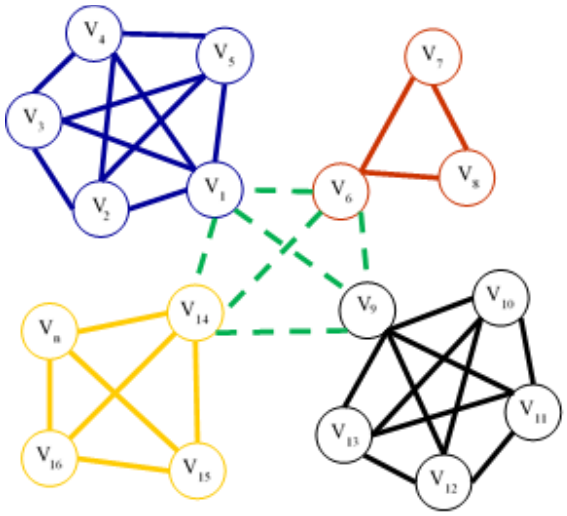


Figure 3.3. Tight cliques in graph G. The thick color line represents strong correlation while the dotted line represents weak or no correlation in the entire graph.

Once the updated weighted graph is available and number of CN (m) and number of EN (k-m) are determined, the process further analyzes the graph to determine *tight cliques* in the graph

G as shown in Figure 3.3. A *tight clique* is a clique where the correlation between any of its two nodes is above a threshold value (>0.50). To determine CN and EN in any clique c, first the number of CN in the clique is computed as:

$$m_c = \frac{|c|}{|G|} \times m: |c| \text{ is the clique size, } |G| \text{ is the graph size} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3.3)$$

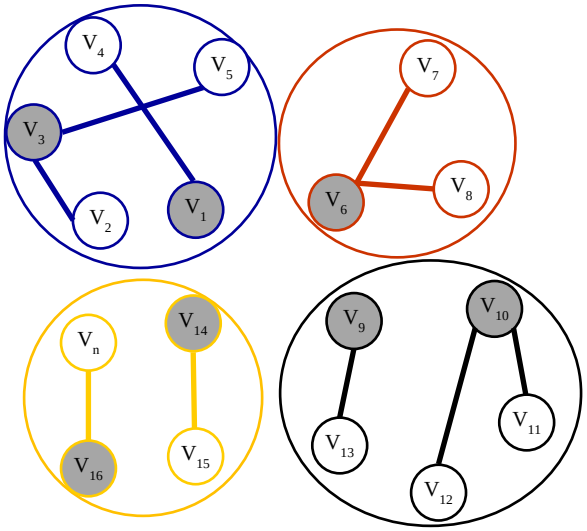


Figure 3.4. Bipartite graph in each clique with the strongest correlations between estimating nodes (transparent) and computing nodes (gray shaded)

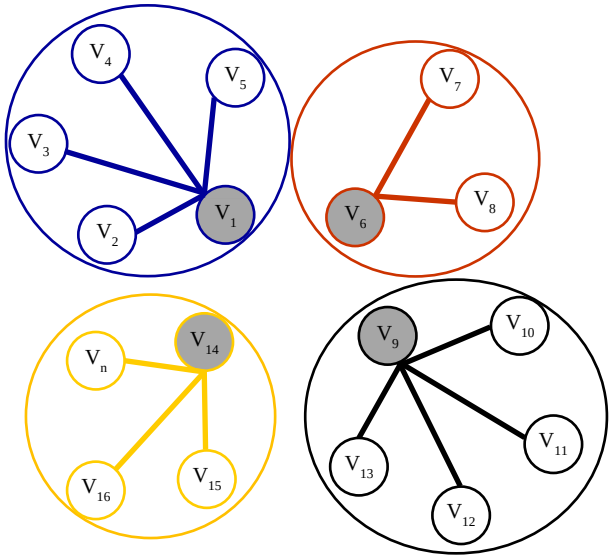


Figure 3.5. Bipartite (star) graph in each clique with the strongest correlations between computing nodes (shaded) and estimating nodes (transparent)

Top ranked m_c nodes in clique c are chosen as CN_c and the remaining $|c| - m_c$ nodes are considered as EN_c. Considering CN_c as one partite and EN_cas the other partite in clique c, leave the edges $\{e_c\}$ only between $\{CN_c\}$ and $\{EN_c\}$ that have the highest weight and exclude the rest of the edges. This results a bipartite graph G_c in clique c. Applying the same method in each clique a set of bipartite graphs are obtained as shown in Figure 3.4.

In case when size of CN is equal to the number of cliques in a graph, the node with the highest rank, called *junction node* (e.g., V_1, V_6, V_9, and V_{14} in Figure 3.5), in each clique is the

only computing node and all the remaining nodes in the clique will be the estimating nodes forming star structures (still set of bipartite graphs) as shown in Figure 3.5. The advantage of generating these bipartite graphs is that each node in EN will be estimated directly using its strongest correlated node in CN.

3.2.4 Graph Approach - Proof of Concept

As a proof of concept for the graph approach, an example walkthrough is illustrated as bellow for a graph of size 17 nodes (see Figure 3.6).

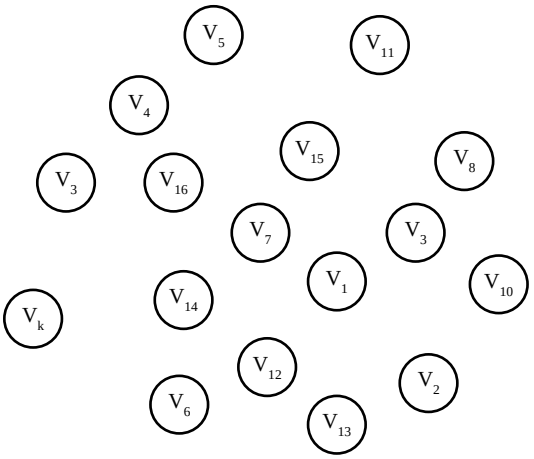


Figure 3.6 Initial spatial clusters for sample graph

Walk-Through:

- With k initial clusters (nodes), form a complete weighted graph G (k=17, edge weight very small ≈ 0) (see Figure 3.7)

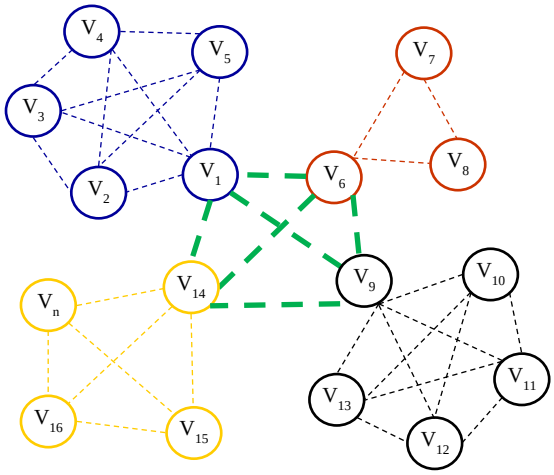
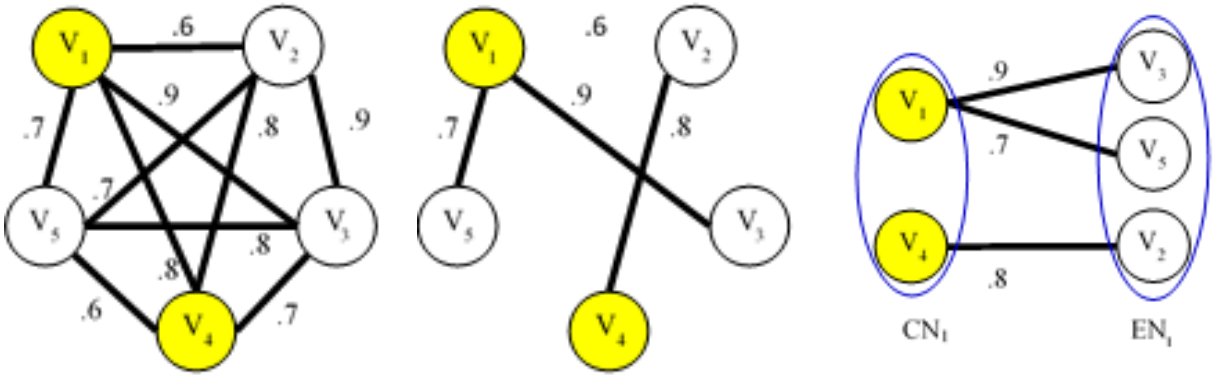


Figure 3.7. Initial spatial graph using the spatial clusters

- After udating edge wieghts using the correlation matrix $r_{ij}: i, j = 1\ to\ k$, cliques in graph G are computed based on edge weight greater than a threshold



(a) (b) (c)

Figure 3.8. Clique 1 of Figure 3.7. (a) Clique with all the weights (b) ENs(transparent) connected through largest weighted edge with CNs (c) Resulting bipartite graph in Clique 1

- **In each clique,** determine the number of nodes that can be computed by proportionately distributing the total number of computing nodes(n). For example, if the total number of nodes that could be computed is 7, then number of nodes to compute in clique 1 (as shown in Figure 3.8(a)) can be determine as

$$n_1=\text{int}\ (5/17)x7=2$$

where, 17 is the graph size, 5 is the size of the clique of interest, and 7 is the total number of nodes that could be computed.

- Identify the top ranked n1 (=2) nodes (e.g. V1 and V4 in clique 1, Figure 3.8(b)) as the computing nodes(CN1) and the remaining nodes as estimating nodes(EN1).
- Reorganize the cliques as a bipartite graph. G1=(CN1, EN1,E1) where, E1 is the set of edges with maximum weights connecting each node in EN1 to any node in CN1 as shown in Figure 3.8(c)

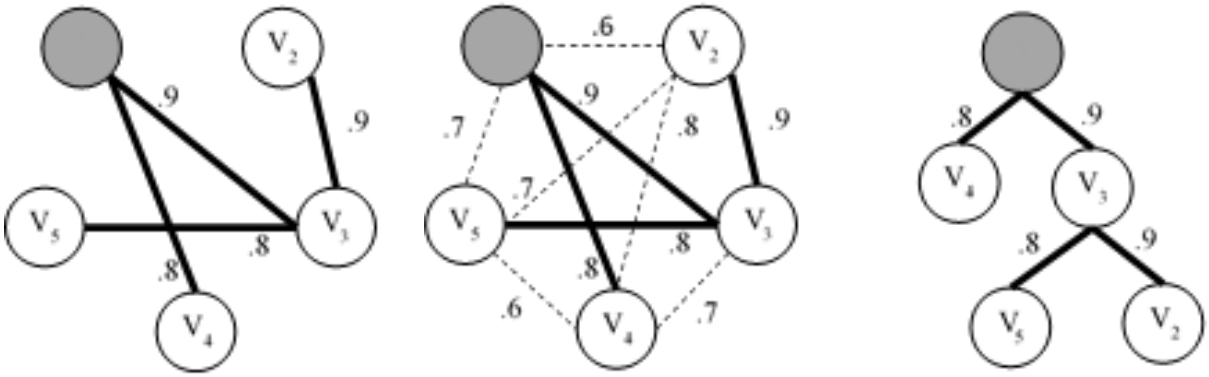


Figure 3.9. MST from clique 1. Shaded node is the root cluster that is a computing node

- Process continues untill all cliques(eventually the whole graph) are processed and finally are accumulated into a set of bipartite graphs (BP) out of the input graph G. We also tried Minimum Spanning Tree (MST) (see Figure 3.9) approach which reduces the number of computing nodes but compromises accuracy since some nodes may be estimated based on other estimated node; not directly from computed node.

- Apply computation task (f_c) to each node in the computing node partite [here only shown for CN1; for instance in V1 and V4 here].

- Estime each node in EN1 by using prediction function (f_p) . f_p works for each node in EN1 by regressing on associated computed node in CN1 and using the edge weight. For example, V3 is estimated by regressing on V1 and using the edge weight 0.9. Similarly, V5 is estimated by regressing on V1 and using the edge weight 0.7 and V2 is estimated by regressing on V4 and using the edge weight 0.8.

4.0 EFFICIENT ANALYTICS

The main contributions of this chapter are efficient analytics components, as part of the framework, to compute the spatiotemporal "hotspots" and provide solution within time and computing resource constraints when sufficient historical data is either not available or the data shows none or random pattern over locations. The components of Efficient Analytics (EA) are organized in this chapter as follows. Section 4.1 introduces the necessity of EA and highlights the available methods, tools, and techniques for handling data-intensive problems and their limitations. Section 4.2 describes the requirement analysis to analyze all possible cases of data, constraint, and available resources to determine the computing requirements. Section 4.3 describes the T-Window approach where a temporal sliding window-based variance method is used to rank order the data. Section 4.4 presents S-Window approach where a spatial window-based clustering method is used so that the entire spatial domain in the input data is represented in the hotspot computation. Section 4.5 introduces an algorithm designed to perform the EA functionalities as part of the framework.

4.1 WHY EA?

In EA, the analysis is performed on the spatiotemporal data available instantly since sufficient and useful historical data may not be available always for all applications. The term "efficient" stems from the fact that the EA provides a solution even if sufficient data points are not available to apply any data mining and/or machine learning techniques to find if there exist any particular pattern of

spatiotemporal variations in the data. Even if enough historical data is available, useful pattern of variation may not be found in the data by investigating the descriptive statistics of the data that provide indication of any possible pattern. The pattern of the variation is useful because looking at the pattern, it is easy to predict more accurate results compared to the results obtained based only on few data points. The main idea in EA is to determine spatiotemporal "hotspots" in the available dataset so that target computing task can be applied on the top priority data points to ensure acceptable solution within a given time constraint using available computing resources.

The hotspots are high priority candidate data, over different locations, where the variations in those locations are high. In EA, the hotspots are computed based on ranking of the data points where the ranking process involves two approaches. In the first approach, temporal window (T-Window) based variations on spatiotemporal data over locations are used for ranking. In the second approach, when only very few spatiotemporal data points are available, and where the first approach does not apply, a spatial window (S-Window) based clustering method is used for the ranking. In both approaches, a stratified random sampling is used to scale the data size to meet the time and computing resource constraints. However, to handle all possible cases, before applying any of the two approaches, data is analyzed to determine if sufficient resources are available to compute the entire dataset within the required time constraint.

Today, efficient computing for data-intensive applications, in particular those in the spatiotemporal domain, is a major problem. Available computing resources, tools, and techniques have limited capability of processing efficiently when the data size keeps scaling up. On top of that lack of effective analytics and/or unavailability of appropriate analytics pose challenge on processing the task in the required time. Especially, current hotspot techniques do not address the temporal dimension in spatial domain and real time computing constraints.

4.2 REQUIREMENT ANALYSIS

Before applying any further analytics or computing tasks, it is important to determine what volume of input spatiotemporal data could be handled for required computing tasks using available computing resources within given time constraint. When the required computing task is beyond the computing capacity of available resources, a method is needed to choose an optimal number of top priority data points from the entire dataset. Accordingly, the candidate data points are determined from the combination of ranking and/or clustering (explained in Section 4.3 and Section 4.4 respectively) methods. But in case the available computing resources are sufficient to compute the entire dataset within time constraint, there is no need for any ranking and/or clustering to apply. Three cases are possible: *All-Data*, *Optimal-Data*, and *Partial-Optimal-Data* as explained below.

4.2.1 Case 1: All-Data

If there are sufficient computing resources (number of nodes in this case) available to compute on the entire data within time constraint, which is the best case, there would be no need to filter any data out of the computation. A method is developed to determine the data volume that is possible to compute on a given computation scenario as shown in Algorithm 4.1. Here available resource time (Tr) per node, number of available nodes (N), and estimated computation time (Ts) per data point are provided. The assumption is that all the available nodes are homogeneous (similar CPU speed and memory size). The algorithm returns the whole dataset (D) if sufficient computing resources are available to complete the required computation in time. Otherwise, if sufficient

computing resources are not available, the algorithm returns ∞ which means that some other method needs to be used to process the data before applying the computing task. The algorithm has a linear time complexity O(n).

Algorithm 4.1 Determining the optimal computable data size for given computing resource time

Optimal Data: $f_{op}()$

Input: Spatiotemporal Data(D), Number of processors/computing nodes(N), Resource time(Tr), Estimated computation time per data point(Ts), time constraint(To)
Output: Candidate Dataset(CD) for computation, S

1. begin
2. $m \leftarrow \frac{T_r}{T_s}$ // data size limit that could be handled in each node
3. $S \leftarrow \mathrm{m} * \mathrm{N}$ // total size of data possible to handle
4. if ($S \leq size(D)\ and\ T_r \leq T_0$)
5. $CD \leftarrow \mathrm{D}$
6. else
7. $CD \leftarrow \infty$
8. end

4.2.2 Case 2: Optimal-Data

When computing resources are limited, optimum number of top ranking data from the ranked datasets are fed for the computation. The process takes available resource time (Tr) per node, number of available nodes (N), ranked dataset (RD), and estimated computation time (Ts) per data point as input. Based on the combined computing resources of the nodes, the process in this case determines the top ranking RD it can take into account to complete the computation in the required time constraint. Accordingly, the top ranked data points are fed into the computation.

4.2.3 Case 3: Partial-Optimal-Data

This is a special situation of the Optimal-Data case as explained above. Sometime, one may want just to get an idea of the outcome of a computing task on a fraction of the entire dataset quickly. If only a fraction (e.g. 50%) of the top ranking optimal-data are expected to be used for computing to examine the outcome only in the fraction, then this case does apply. The process in this case takes percentage of the ranked data (P), number of total nodes (N), available resource time (Tr) per node, ranked dataset (RD), and estimated computation time (Ts) per record as input. The outcome of this case demonstrates the result based only on the top P percent (expected fraction of the entire dataset) of ranked data. Both Case 2 and Case 3 require temporal sliding window (T-Window) method and/or spatial window (S-Window) method to determine the optimal data.

4.3 TEMPORAL SLIDING WINDOW METHOD

Temporal sliding window (T-Window) method captures the variability of data over time and locations. For a given window size (Δt), the T-Window method computes the variance of data in each window over the time range (T) as data is available. The data points/objects are ranked according to the variance of the time windows; higher the variance, higher the rank. The T-Window method is shown in Figure 4.1. The method is used to identify hotspots, the high-ranking data points, in the input dataset. The assumption is that the dataset is very large and the computation must be performed in a given time constraint. The objective is to rank the input data based on the order of variability of data over time and location (Cressie & Johannesson, 2008). In

this method, the variance of the spatiotemporal data in each time window of size (Δt) is computed and then the groups are rank ordered according to the window variance ($\boldsymbol{\sigma}$). Based on the rank order, top ranking data points/objects are considered as the priority candidate input data which is termed as spatiotemporal "hotspots"(Chainey *et al.*, 2008).

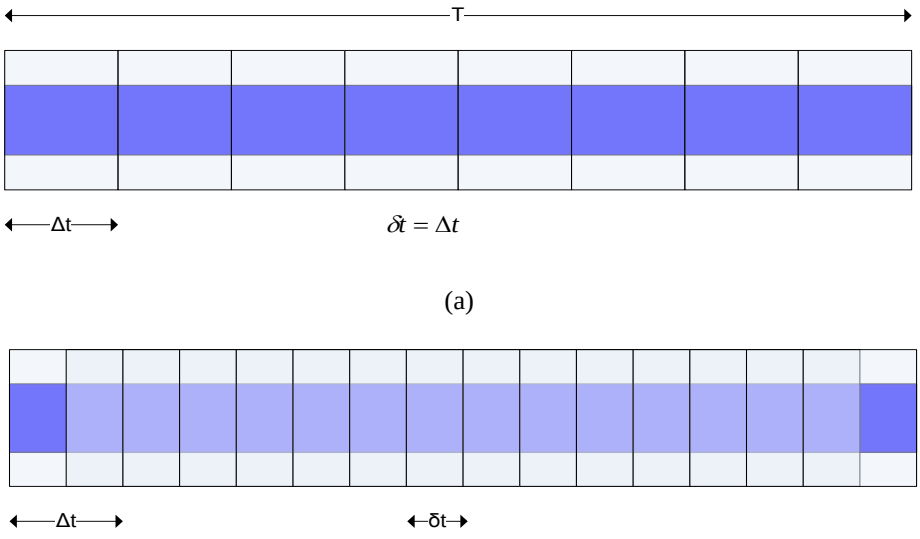


Figure 4.1. Sliding window method (time window). (a) non-overlapping window ($\delta t = \Delta t$), (b) overlapping window ($0 < \delta t < \Delta t$)

The T-window method starts with spatiotemporal dataset D over time T. The method is explained below.

(D_t^L) is spatiotemporal dataset at time t at a location L and is expressed as:

$D_t^L = \{D_1, D_2, D_3 \dots \dots \dots D_k\}, where, D_k is\ a\ datapoint/object\ in\ (D_t^L) at\ time\ t$

Time window size = Δt

Window sliding step = δt

Number of windows to compute is found as:

$$n = \left\lceil \frac{T - \Delta t}{\delta t} \right\rceil + 1 \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \quad (4.1)$$

Variance of the i^{th} window:

$$\boldsymbol{\sigma}_i = \left\{ var\left(D_{\Delta t_i}\right); i = 1{:}\, n \right\} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \quad (4.2)$$

The data points ordered based on the variance over windows are found as:

$$d = \frac{sort}{dsc}\left(D_{\Delta t}, \sigma\right) \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \quad (4.3)$$

The rank of data points in window $D_{\Delta t_i}$,

$$r\left(D_{\Delta t_i}\right) = index\left(D_{\Delta t_i}, d\right) \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \quad (4.4)$$

The method ranks the data points in any spatiotemporal dataset according to data variability to identify “hotspots”, the data with top variations over time and locations. Based on the computed rank of the data points, hotspots are scaled in accordance with the available computing resources and the required time constraint.

4.4 SPATIAL WINDOW METHOD

Spatial window (S-Window) method addresses the case when the available spatiotemporal data may not have sufficient temporal range to capture any temporal variations over locations. The data may be just current and/or some recent instances where the T-window method is not suitable since it requires window variance to compute. The S-window method applies a spatial clustering on such data to cluster them based on locations first. For n time instances (t_1, t_2, ……., t_n) of data, spatial clusters (one at each timestamp) are computed for the same geographic area for each instance.

Method of the spatial clustering, which is a K-means based spatial clustering (MacQueen, 1967), is shown in Algorithm 4.2.

Algorithm 4.2. K-Means based spatial clustering to generate the spatial clusters over the area of input spatiotemporal data

Clust: $f_{cl}()$

Input: A, the spatiotemporal data/object points to be clustered
k, estimated number of clusters
δ, threshold of convergence
Output: C, a set of K clusters

1. $i = 0$
2. for $n = 1\ to\ k$
3. $c_{n,0} = random(a \in A)$
4. do
5. $i = i + 1$
6. $A_{n,i} = \{a \in A \mid \forall(n' \neq n), \|a - c_{n,i-1}\| \leq \|a - c_{n',i-1}\|\}$
7. for $n = 1\ to\ k$
$c_{n,i} = \frac{1}{|A_{n,i}|}\sum_{a \in A_{n,i}} a$
9. while $\sum_{n=1}^{k}\|c_{n,i} - c_{n,i-1}\|^2 > \delta$
10. $C = \{c_n,\ \ n = 1 \ldots . k\}$
11. return C

The input for the algorithm is the spatiotemporal dataset (A), estimated number of clusters (k), and proximity threshold of convergence (δ). The algorithm produces the set (C) of k clusters. Here the value of k depends on the range of the spatial domain (geographic extent); the larger the area, the higher k value. However, large number of clusters require more computation than smaller number of clusters. Usually, k is chosen as, $k = \sqrt{n/2}$ where, n is the data size. The proximity threshold of convergence (δ) determines at what point of proximity the iteration of clustering

should end and settle the clusters. The time complexity of the spatial clustering algorithm is $O(n^{dk+1}\log n)$.

In the clustering method, the target is to minimize the error through an *objective function,* which is sum of squared error (SSE) and is expressed as:

$$SSE = \sum_{j=1}^{K} \sum_{x_i \in C_j} \left\| x_i - c_j \right\|^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.5)$$

where, $\left\| x_i - c_j \right\|^2$ is the distance measure (or similarity measure) between a data point x_i and the cluster center c_j. The new cluster centerc_j, for jth cluster C_j having $n_j$data points, is computed as the mean of all data points in the cluster:

$$c_j = \frac{1}{n_j} \sum_{x_i \in C_j} x_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.6)$$

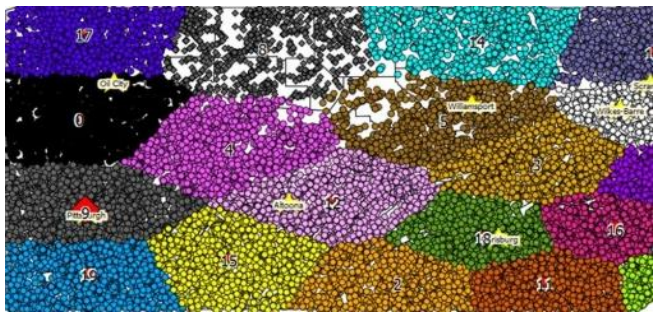


Figure 4.2. Example spatial clusters at one timestamp. Each color represents a different spatial cluster

Spatial clustering method is used here to group the closer data points together since closer data points are more related to each other than distant data points; according to Toblers' first law of geography (Tobler, 1970); "everything is related to everything else, but near things are more related than distant things". A sample spatial cluster is shown in Figure 4.2 where the data points are POI locations over a geographic area.

4.5 EA ALGORITHM

In the absence of sufficient historical spatiotemporal data, extracting pattern or even extracting temporal variation of the data over locations does not provide any useful information. Even if sufficient amount of historical data is available but there is no regular pattern found (e.g., no or random pattern) in the preprocessing phase (explained in Chapter 3), EA should handle the data. So, data availability wise there are two cases:

1. Past spatiotemporal data is available but no regular pattern found
2. Only some instances of data available where no pattern and temporal variation can be extracted

Algorithm 4.3 Efficient Analytics – takes current spatiotemporal data and provides spatiotemporal hotspots (top candidates to apply the computation tasks) and computation result

EfficientAnalytics()

Input: Current/Recent Spatiotemporal Data(SD)
Output: Spatiotemporal Hotspots and Computation result

1. start
2. num $\leftarrow f_{op}(SD)$ //data size computable in time T_o for task f_c
3. $ds \leftarrow NULL$
4. if num $\geq$ len(SD)
5. $ds \leftarrow SD$
6. else
7. $ssd \leftarrow$ sample(SD) //random sample
8. $k \leftarrow$ sqrt(len(sd)/2)
9. $C \leftarrow f_{cl}(sd)$ // C = {C_1, C_2, ……., C_k} // create k clusters
10. for each $c_i \in C; i = 1\ to\ k$
11. $d_r \leftarrow f_r(ssd(c_i))$ // rank order sampled data in order of significance in each cluster
12. $ds \leftarrow$ columbind(d_r) //a data frame keeping each ranked cluster data as a column
13. end
14. end if
15. $result \leftarrow f_c(ds)$
16. end

EA algorithm, as presented in Algorithm 4.3, is designed to handle both cases. The algorithm first determines (step 2) how many data points could be computed within time constraint (T_o) for the computing task (f_c) on available computing resources using algorithm 4.1 (f_{op}). Two scenarios are possible: i) it can compute all data points and ii) it can compute part of the data points. In the first scenario, the computing task (f_c) is applied to the entire dataset and the computed results are returned (step 4-5, and step 15). But, in data-intensive applications, this situation is rare since the required computing resources may not always be available.

In the second scenario, which is common in most data-intensive applications, both of the data availability cases are handled. The first case (historical data available but no or random pattern exist) the T-Window method (explained in section 4.3) is applied to obtain the ranked ordered dataset. The fraction of the data, that the computing resource is capable to compute on, is chosen from the ranked dataset. In case the amount of data in the same rank exceeds the capacity, stratified sampling (Murthy, 1967) is applied in each window (data group) to meet the size constraint. The second case (historical data available and regular patterns exists) is handled by S-Window method where a spatial clustering is used to group the data (step 7-15). The time complexity of EA algorithm is O ($n + n^{dk+1} log n$).

The sampling can be used either before or after the clustering where pros and cons are there in either of the cases. If the sampling is done before clustering, the clustering process is completed faster compared to the case if sampling is done after clustering. Because, sampling reduces the size of the clustering input. But the cons of sampling before clustering is that some of the geographical area may not get selected resulting no representation of that area in further analysis since this sampling is based on non-spatial attribute. Advantage of sampling after clustering is that most of the area is covered but limitation is that clustering process takes longer time to complete.

5.0 PREDICTIVE ANALYTICS

When sufficient historical data is available, the predictive analytics component learns the patterns, if exist, in the data. Two types of prediction are performed in predictive analytics: *data value prediction* using statistical methods for time series prediction and *cluster prediction* by analyzing the changes in the temporal graph patterns of clusters over locations.

5.1 DATA VALUE PREDICTION

The *data value prediction* is tailored through time series models. In general, time series can be used for modeling prediction and forecasting for planning and decision making. Time series data analysis usually involves decomposition of data into a trend, such as temporal, cyclical, or irregular component. The current values of any time series data are influenced by the values occurred in the past. For example, the temperature in current hour has a relationship with the temperature in the hour before in a given location; autocorrelation effect. This effect is addressed in time series data analysis by *lagging,* a backshift operator, which operates on an element of a time series to produce the previous element. Regression without lags fails to account for the relationships through time and overestimates.

Each element in a time series is a random draw from a population with zero mean and constant variance which is termed as *white noise* assumption (Hamilton, 1994). To analyze and forecast equally spaced univariate time series data, an Auto Regressive Integrated Moving-

Average (ARIMA) or autoregressive moving-average (ARMA) based model is formulated and used. The model predicts a value, by means of a response time series graph, as a linear combination of its own historical values, historical errors (also called shocks or innovations), and current/historical values of other time series graphs. Autoregressive (AR) and moving average (MA) models correct for violations of the white noise assumption.

5.1.1 Prediction Models

In *AR models* the value of a variable in one time period is related to its values in previous time periods. For lag p, AR(p) is an autoregressive model and stated at time t as:

$$y_t = \mu + \sum_{i=1}^{p} \alpha_i y_{t-i} + \epsilon_t \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.1)$$

where μ is a constant parameter, α_p is the coefficient for the lagged variable in time *t-p*, ϵ_t is white noise. For instance, for lag 1, AR(1) is expressed as:

$$y_t = \mu + \alpha_1 y_{t-1} + \epsilon_t \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.2)$$

where y_t is the value at time t and determined by its previous value (lag=1) y_{t-1}.

The *Moving Average (MA) models* depict the possibility of a relationship between a variable at a given time and the residuals from previous time periods. For a lag q, MA(q) is a moving average model and it is expressed for time t as:

$$y_t = \mu + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.3)$$

where $\beta_q$is the coefficient for the lagged error term in time *t-q*. For example, for lag 1, MA(1) model is expressed as:

$$y_t = \mu + \beta_1 \epsilon_{t-1} + \epsilon_t \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots . (5.4)$$

ARMA models combine both *p* autoregressive terms and *q* moving average terms, also called ARMA(p,q).

$$y_t = \mu + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{j=1}^{q} \beta_j \epsilon_{t-j} + \epsilon_t \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots .. (5.5)$$

ARMA-based models are suitable for stationary time series data whereas ARIMA-based models are suitable for both stationary and non-stationary time series data. While an ARMA-based model can immediately be used to the time series at hand as is, ARIMA-based models are usually applied after the differencing of the time series data. Differencing is performed by taking the differences between consecutive values instead of the values themselves. Lags of the differenced series that appear in the prediction equation are called *auto-regressive* terms while the lags of the forecast errors are called *moving average* terms. The time series data in which differencing is done to make the data stationary (explained below) is called *integrated* version of a stationary series. Also, temporality plays important roles in ARIMA-based models. The three terms, *stationarity*, *differencing*, and *temporality*, are defined as bellow:

Stationarity: A stationary process has a mean and variance that do not change over time and the process does not have any trends/patterns. Stationarity is a requirement of modeling an ARMA(p, q) process. For lag 1, an AR(1) process can be expressed as:

$$y_t = \delta y_{t-1} + \epsilon_t \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.6)$$

This is stationary if $|\delta| < 1$ and ϵ_t is white noise.

Differencing: When a variable y_t is not stationary, a common solution is to use differenced variable and the d[th] order difference is found as:

$$\Delta_t = y_t - y_{t-d} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.7)$$

The variable y_t is integrated of order one, denoted as $I(1)$, if taking a first difference produces a stationary process. That is, for the first order difference, the differenced variable is expressed as:

$$\Delta_t = y_t - y_{t-1}$$

Hence, an ARMA model with p autoregressive lags, q moving average lags, and a difference in the order of d, is denoted as ARIMA (p, d, q).

Temporality(Seasonality): Temporality is a type of autocorrelation pattern where patterns likely to repeat every *season* like daily, monthly, quarterly, yearly, or any time interval. Temporality must be taken into consideration before a time series model can be fitted. ARIMA models, for time series with temporality, often use differencing operators and autoregressive and moving average parameters at lags that are multiples of the length of the temporal cycle (Dickey & Fuller, 1979). Thus, the general notation for the order of a temporal ARIMA model (M. H. Lee *et al.*, 2012) with both temporal and non-temporal factors is ARIMA(p, d, q)x$(P,D,Q)_s$, where (p,d,q) is the order of the non-temporal part of the ARIMA model; (P,D,Q)s is the order of the temporal part and s is the number of observations in each season.

Since stationarity in the time series data is a requirement in ARIMA predictive models, stationarity is ensured by successively differencing the data until it is no longer present. The sample autocorrelation function (ACF) and partial autocorrelation function (PACF) are two means for testing the stationarity. After a few lags, ACF of a stationary series decay exponentially whereas PACF cut off completely. For an autoregressive (AR) process, the sample ACF decays gradually

while the sample PACF cuts off after a few lags. The reverse phenomena are observed for moving average (MA) process. For MA process, the sample ACF cuts off after a few lags while the sample PACF decays gradually. In case where both the ACF and the PACF decay gradually, an ARMA model is recommended. ACF and PACF are defined bellow.

ACF: The proportion of the autocovariance of a dependent variable y_t and y_{t-p} (for p lags) to the variance of y_t at time index t:

$$ACF(p) = \rho_p = \frac{cov(y_t, y_{t-p})}{var(y_t, y_{t-p})} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.8)$$

ACF(p) gives the cross correlation between y_t and y_{t-p}. For example, for an AR(1) model with lag 1, the ACF is $ACF(1) = \rho$. This function tails off which means that the lengths of the line segments in the ACF plot gradually decay, and this pattern continues for increasing lags. This behavior indicates non-stationarity of the time series. Conversely, for stationary series ACF plot quickly declines to zero.

PACF: It is a simple correlation between y_t and y_{t-p}minus the part explained by the intervening lags and is expressed as follows:

$$\rho_p^* = corr(y_t - E^*(y_t|y_{t-1}, \dots, y_{t-p+1}), y_{t-p}) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.9)$$

where $E^*(y_t|y_{t-1}, \dots, y_{t-p+1})$ is the minimum mean-squared error predictor of y_t by $y_{t-1}, \dots, y_{t-p+1}$. PACF identifies the extent of the lag in an AR model. For an AR(1) model, the PACF is a spike for the first lag and then cuts off. A summary of ACF and PACF properties for AR(p), MA(q), and ARMA(p,q) models are given in Table 5.1.

Table 5.1 ACF and PACF properties for ARMA-based models.

	AR(*p*)	MA(*q*)	ARMA(p,q)
ACF	Tails off	Cuts off after lag *q*	Tails off
PACF	Cuts off after lag *p*	Tails off	Tails off

Spatial Autocorrelation: A variable is said to exhibit spatial autocorrelation if the presence of the variable in a locality influences the presence of the variable in the neighboring localities to be more or less likely. Spatial autocorrelation analysis tests whether the observed value of the variable at one locality is independent of the values of the variable at neighboring localities. It is basically the correlation of a variable with itself over locations. Spatial autocorrelation can determine the level of interdependence of a variable with itself in its neighboring region, and the nature and strength of that interdependence. The measure is obtained by *Moran's I* and/or *Geary's coefficient C* (see APPENDIX A). Spatial autocorrelation could be classified as either **positive or negative**. Spatial autocorrelation would be positive when all similar values appear near each other while a negative spatial autocorrelation occurs when dissimilar values appear near each other.

5.1.2 Model Selection for Data Value Prediction

An algorithm for optimum spatiotemporal prediction model selection is depicted in Algorithm 5.1 to determine an appropriate model for data value prediction. The basic idea is to identify and predict hotspot values by analyzing time series data varying over locations (Maciejewski *et al.*, 2011; Repantis & Kalogeraki, 2008). The algorithm takes *data*, autoregressive lags(*p*), moving average lags(*q*), and order of differences(*d*) for non-temporal analytics while taking the same

parameters, i.e., P, Q, and D, for temporal analytics for s number of observations in each temporal span within a time window T. The algorithm computes a set of factors that could be used to evaluate and determine appropriate model for spatiotemporal hotspot prediction.

Algorithm 5.1. Model selection for data value prediction – selects the most appropriate prediction model by evaluating their goodness of fit using their different performance metrics.

DVPredModel()

Input: data, Y, p, q, d, P, Q, D, s, T, model ← {AR(p).R, MA(q).R,
ARMA(p,q,d).R, ARIMA(p,q,d).R, SARIMA(p,q,d).(P,Q,D)s.R, …}
Output: AIC, BIC, R^2, -2LogLH, MAPE, MAE, Mu, Sigma, Model

1. begin
2. DS_{HS} ← Optimize(f_r (data, Y)) //Ranked hotspot dataset
3. Y← DS_{HS}.Y
4. d.Y←Y_t-Y_{t-1}, t=1….T //difference variable
   //Stationarity test
5. Stationarity ← min(Lag_Cut_Off(ACF(Y))) < min(Lag_ Cut_Off(ACF(d.Y))) ? 1, 0
   //Response variable selection
6. R ← (Stationarity)? Y, d.Y
7. for i = 1 to Length(model M)
8. $\mathbf{V}_i$ ← Execute(M_i(R, data))
9. end
10. W ← ($\mathbf{V}_1$, $\mathbf{V}_2$, $\mathbf{V}_3$, ……., $\mathbf{V}_n$) //Weight Matrix
11. MW[]← 0 //Model weight
12. for j = 1 to n
13. for k=1 to |Vn|
14. MWj ← MWj+Wjk
15. end
16. end
17. MI ← Index(min(MW_i, i=1, 2, ….., n)) // Optimum Model Index
18. Model ← model(MI) // Optimum Model
19. return Model
20. end

The algorithm (Algorithm 5.1) starts with a set of hotspot records (DS_{HS}) obtained by applying the ranking function (f_r) to *data* and by optimizing the hotspots [step 2] manageable on given computing resources in real time. The stationarity of the data is tested by comparing the cut off lags using ACF both on original outcome variable Y and its differenced variable d.Y [step 3-

step6]. Once the variable of analysis (R = Y or d.Y) is determined, models are applied on R[step 7-step 9]. A weight matrix (*W*) is created [step 10] using the outcome vectors (containing the output evaluating factors) of the models. Analyzing the weight matrix (*W*), the most optimum prediction model (one with the minimum weight) is determined for the given data [step 11-step 19]. The complexity of this algorithm is O(1) since the choice of models are fixed and time needed to parameter estimate is also fixed. The algorithm returns the appropriate prediction model for any given time series data set.

5.2 CLUSTER PREDICTION

Based on the patterns in spatial clusters in the temporal graphs, a prediction model (f_p) is developed to provide prediction on clusters. For n timestamps (t_1, t_2, ……., t_n), n sets spatial clusters (one at each timestamp) are computed for the same geographic area. Accordingly, n sets of temporal graphs of spatial clusters are generated using the clustering method as explained in Chapter 4. For fixed k and dimension (d) the complexity of the algorithm is $O(n^{dk+1} log\ n)$ for n data points/objects.

In the cluster prediction method, a time series of n sets of spatial clusters and its corresponding graphs are further analyzed for inter-cluster correlation matrix. Based on the correlations among the clusters each of the graphs is further grouped into nodes set where all the nodes are strongly correlated. Now, using the ranking method (f_r) and the graph approach (as explained in Chapter 3) a set of bipartite graphs are generated from each graph at a timestamp where one partite is the set of computing nodes and the other partite is the set of estimating nodes

whose values are predicted based on the values in corresponding correlated computing node. Now taking all the computing nodes in one partite and all the estimating nodes in the other partite, a single bipartite graph is produced at each timestamp.

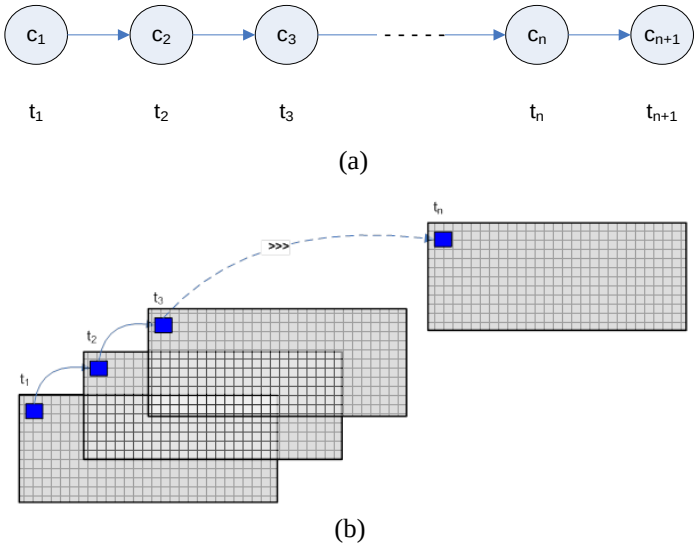


Figure 5.1. Spatial clusters over n timestamps. (a) The cluster sequences, (b) spatial clusters over same area of interest at different timestamps

In the temporal range of the spatiotemporal data if there are n-timestamps, n-bipartite graphs (also termed as n temporal graphs) are created. In this set of bipartite graphs, a time series of clusters are obtained and this cluster time series in the temporal graphs over the entire time span are used to predict cluster in the next time point. For example, learning from the pattern of clusters time series, generated at t_1 through t_n, in a node, the cluster c_{n+1} at t_{n+1} (see in Figure 5.1) can be predicted. Following the same procedure, all the clusters at time t_{n+1} are predicted and constructing the graph with the predicted clusters in turns provides the predicted graph at time t_{n+1}.

5.2.1 Prediction Method: Using Inter-Cluster Correlation

In this section a detail derivation is provided to establish the justification of using inter-cluster correlation coefficient (r) for estimation by demonstrating how r is directly related to the parameter (β_1) estimate in the prediction model.

Assumption: **y**(outcome) is continuous and normally distributed.

For n timestamps (t_1, t_2, …….., t_n) at a cluster location, let the regression equation be represented as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ \beta_0 + \beta_1 x_3 \\ \vdots \\ \beta_0 + \beta_1 x_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$=> \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$=> \boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon} \quad \ldots\ldots\ldots \quad (5.10)$$

where,

$$\boldsymbol{Y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}, \boldsymbol{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

To estimate $\boldsymbol{\beta}$, from equation (5.10) the residual

$$\boldsymbol{\epsilon} = \boldsymbol{Y} - \boldsymbol{X\beta} \quad \ldots\ldots\ldots \quad (5.11)$$

The objective is to minimize the sum of squared (SS) residual (least square estimation).

SS residual,

$$\sum_{i=1}^{n} \boldsymbol{\epsilon}_i{}^2 = [\epsilon_1 \quad \epsilon_2 \quad \epsilon_3 \dots \epsilon_n] \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_n \end{bmatrix} = \epsilon^T \epsilon \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.12)$$

Thus equation (5.11) yields,

$$\epsilon^T \epsilon = (\boldsymbol{Y} - \boldsymbol{X\beta})^T (\boldsymbol{Y} - \boldsymbol{X\beta}) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.13)$$

Taking derivative with respect to β and equating to 0, we have,

$$\frac{\partial}{\partial \boldsymbol{\beta}} (\boldsymbol{Y} - \boldsymbol{X\beta})^T (\boldsymbol{Y} - \boldsymbol{X\beta}) = 0$$

$$\Rightarrow -2\boldsymbol{X^T}(\boldsymbol{Y} - \boldsymbol{X\beta}) = 0$$
$$\Rightarrow \boldsymbol{X^T X\beta} = \boldsymbol{X^T Y}$$

$$\Rightarrow \boldsymbol{\beta} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T Y} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (5.14)$$

Now,

$$\boldsymbol{X^T X} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} = \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i{}^2 \end{bmatrix}$$

$$(\boldsymbol{X^T X})^{-1} = \frac{\mathbf{1}}{\boldsymbol{n} \sum_{i=1}^{n} x_i{}^2 - (\sum_{i=1}^{n} x_i)^2} \begin{bmatrix} \sum_{i=1}^{n} x_i{}^2 & -\sum_{i=1}^{n} x_i \\ -\sum_{i=1}^{n} x_i & n \end{bmatrix}$$

and

$$\boldsymbol{X^T Y} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}$$

Then from equation (5.14), $\boldsymbol{\beta}$ is found as,

$$\boldsymbol{\beta} = \frac{\mathbf{1}}{\boldsymbol{n}\sum_{i=1}^{n} x_i{}^2 - (\sum_{i=1}^{n} x_i)^2}\begin{bmatrix} \sum_{i=1}^{n} x_i{}^2 & -\sum_{i=1}^{n} x_i \\ -\sum_{i=1}^{n} x_i & n \end{bmatrix}\begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}$$

$$= \frac{\mathbf{1}}{n\sum_{i=1}^{n} x_i{}^2 - (\sum_{i=1}^{n} x_i)^2}\begin{bmatrix} \sum_{i=1}^{n} x_i{}^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} x_i y_i \\ n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i \end{bmatrix}$$

$$= \frac{\mathbf{1}}{\sum_{i=1}^{n} x_i{}^2 - n\bar{x}^2}\begin{bmatrix} \bar{y}\sum_{i=1}^{n} x_i{}^2 - \bar{x}\sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \end{bmatrix}$$

$$= \frac{\mathbf{1}}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\begin{bmatrix} \bar{y}\sum_{i=1}^{n} x_i{}^2 - \bar{x}\sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \end{bmatrix}, \qquad \sum_{i=1}^{n} x_i{}^2 - n\bar{x}^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2$$

$$= \frac{\mathbf{1}}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\begin{bmatrix} \bar{y}\sum_{i=1}^{n} x_i{}^2 - \bar{y}(n\bar{x}^2) + \bar{x}(n\bar{x}\bar{y}) - \bar{x}\sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \end{bmatrix}$$

$$= \frac{\mathbf{1}}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\begin{bmatrix} \bar{y}\left(\sum_{i=1}^{n} x_i{}^2 - n\bar{x}^2\right) - \bar{x}(\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}) \\ \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} \end{bmatrix}$$

$$= \begin{bmatrix} \bar{y} - \dfrac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\bar{x} \\ \dfrac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \end{bmatrix} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \qquad \sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y} = \sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

So, $\beta_0 = \bar{y} - \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}\bar{x} = \bar{y} - \beta_1\bar{x}$ …… (5.15)

and

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5.16)$$

Equation (5.16) can be further represented as:

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \times \frac{\frac{1}{n}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}{\frac{1}{n}\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5.16)$$

But,

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = cor(X, Y) = r$$

and

$$\frac{1}{n}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2} = standard\ deviation(Y) = s_y$$

$$\frac{1}{n}\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} = standard\ deviation(X) = s_x$$

Thus, equation (5.16) yields,

$$\beta_1 = r \times \frac{s_y}{s_x} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5.17)$$

Equation (5.17) demonstrates that the regression coefficient β_1 is directly dependent on correlation coefficient (r). So, estimation by regression based on correlation coefficient is justified. In prediction through regression method, the regression coefficient β_1 is directly related to the coefficient r. Depending on data and applications, the regression coefficient β_1 will be updated accordingly. The relationship between regression coefficient β_1 and correlation coefficient r depicts that prediction based on inter-cluster correlation is valid and if there exists strong inter-cluster correlation then the values of estimating nodes can be predicted through any prediction method as appropriate for the data and application.

5.3 PREDICTIVE ANALYTICS ALGORITHM

Predictive analytics component is used when historical spatiotemporal data is available so that estimation can be done based on the pattern in the data. To complete the computation or estimation process in real time (within time constraint T_0) a set of preprocessing tasks are performed (as shown in Algorithm 3.1) to prepare the data to be analyzed by a predictive analytics and determine the top candidate data set. Preprocessing tasks are needed both for predictive analytics and meta-analytics.

Predictive analytics acts with the spatiotemporal clusters(C) and the graph (G) as produced by preprocessing, data value prediction, and the cluster prediction method. The predictive analytics is depicted in Algorithm 5.2. The correlations among clusters are computed and the graph edge-weights are updated with the correlation coefficient (r) between each pair of node (v_i, v_j) in G (step 4-step 9.

By using adaptive computing component (as explained in Chapter 3), the number of clusters (nodes) that could be computed by computing task (f_c) and the number of clusters that could be estimated by regressing on the strongly correlated computing nodes within the given time constraint are determined (step 11- step 21). In graph G, all the cliques are identified where the edge weights are greater than (or equal to) a threshold weight (e_h) such that the total number of cliques is less than (or equal to) the number of nodes that could be computed. This is to ensure that, in each clique, at least one node is computed (e.g., v_1, v_2, v_3, v_4,…..v_n in Figure 5.2) so that all the other nodes within the clique can be estimated based on the computed node in that clique.

Algorithm 5.2: Predictive analytics algorithm – takes historical spatiotemporal data and provides candidate data points in clusters as nodes of bipartite graph where one partite contains all computing nodes and the other partite contains all estimating nodes

PredAnalytics()

Input: Spatiotemporal Data (D)

Output: Spatiotemporal Prediction and Computation result

1. start
2. (C,G) ← Call *Preprocess(D)*
3. $n \leftarrow f_{op}$(D) //number of clusters that can be computed in G
4. $CNodes \leftarrow$ NULL; $ENodes \leftarrow$ NULL; $E2CLinks \leftarrow$ NULL
5. for each $c_i \in C; i = 1\ to\ |C|$
6. for each $c_j \in C; j = 1\ to\ |C|$
7. $r_{ij} \leftarrow corr(c_i, c_i)$ // matrix of correlation coefficient among clusters
8. $w(e_{ij}) \leftarrow r_{ij}$ // update edge weight between each pair of node (v_i, v_j) in G(V,E)
9. end
10. end
11. $cq \leftarrow clique(G);\ \forall w(e) > e_h$ // cliques in G with weight ≥ e_h (threshold) and $|cq| \leq n$
12. for each q in cq
13. $nC_q \leftarrow$ round $\left(\frac{|q|}{|G|} n\right)$ //number of nodes to be computed within clique
14. $nE_q \leftarrow |q| - nC_q$ //number of nodes to be estimated within clique
15. $CN_q \leftarrow \text{Top}(\text{rank}(\text{v: v} \in \text{V}_q), nC_q)$ // top ranked nC_q computing nodes in q; a partite set
16. $EN_q \leftarrow \text{V}_q - CN_q$ // estimating nodes in q; another partite set
17. $e_q \leftarrow \{\text{e} \in \text{E}_q : e_{max}(u \in EN_q, v \in CN_q)\}$ // set of maximum weighted edges connecting estimating nodes(EN) to computing nodes(CN)
18. $CNodes \leftarrow CNodes \cup CN_q$ //Accumulating CN
19. $ENodes \leftarrow ENodes \cup EN_q$ //Accumulating EN
20. $E2CLinks \leftarrow E2CLinks \cup e_q$// Accumulating max weighted edges from EN to CN
21. *end*
22. $BP(G) \leftarrow G(CNodes, ENodes, E2CLinks)$ //bipartite graph out of G

//Computation in real time

23. for each cnode in BP(CNodes)
24. $resComp(cnode) \leftarrow f_c(cnode)$ // apply computing task on each CN
25. *end*
26. for each enode in BP(ENodes)
27. $resEst(enode) \leftarrow f_{reg}(cnode, e(enode))$ // apply estimating task on each EN by regressing on correlated computing node
28. *end*
29. *end*

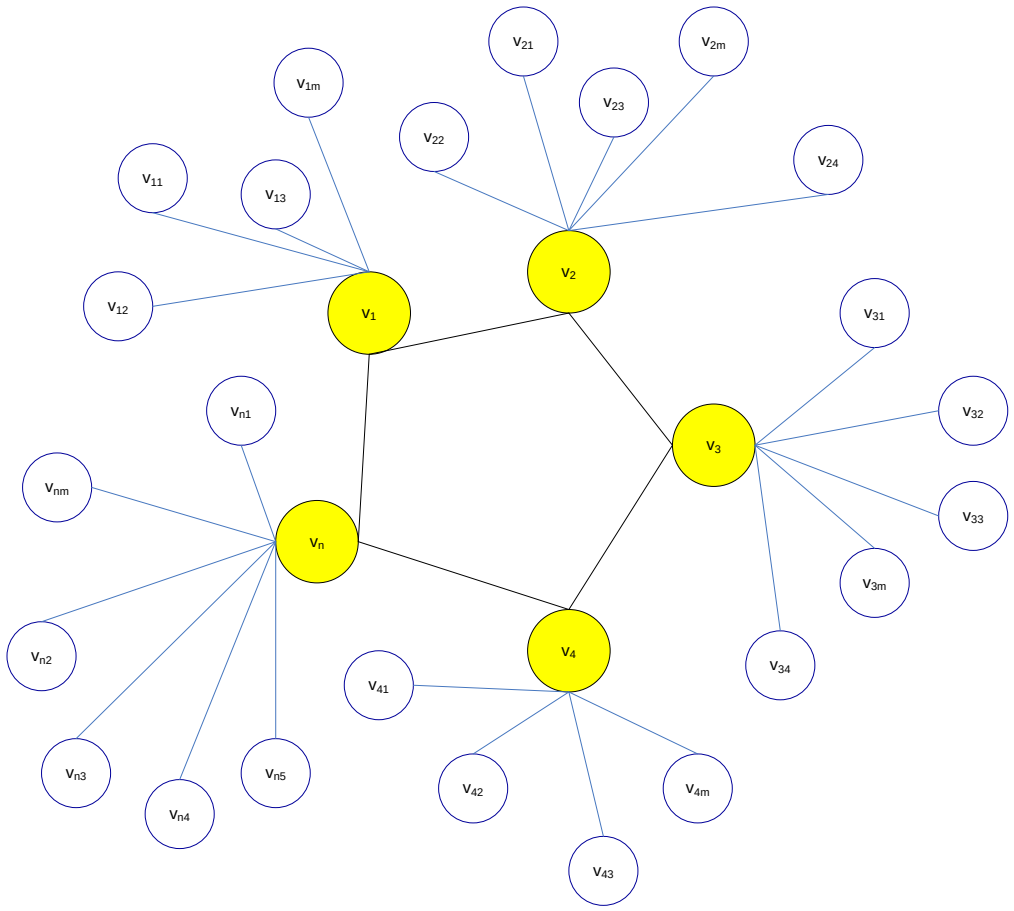


Figure 5.2. Regrouping of a spatial graph into a group of nodes where at least one computing node exists. In this case, v_1, v_2, v_3, v_4, v_n are computing nodes and all others are estimating nodes

If the number of computing nodes is greater than the number of cliques, then more nodes within cliques can be computed and that will improve accuracy. In this case, the number of computing nodes (nC_q) in each clique (q) is determined by the proportion of the clique size to the size of the graph. Since the dataset in each node under each clique is ranked, top ranked nC_q nodes are chosen as computing node (CN_q) and the remaining nodes in the clique are estimating nodes (EN_q). For each node in EN_q, maximum weighted link to a node in CN_q is determined and thus a bipartite graph (CN_q is one partite, EN_q the other partite, and set of maximum weighted links between CN_q and EN_q) out of each clique is generated. The process continues for all the cliques

and results will be a set of bipartite graphs which is further accumulated to one bipartite graph at each timestamp. Finally, all CNs are computed applying f_c (step 23- step 25) and all ENs are estimated by regressing on corresponding computing nodes in CN (step 26 – step 28) using the parameter estimate as determined by the respective correlation coefficients. The time complexity of the predictive analytics algorithm is O (n^2).

6.0 META-ANALYTICS

Meta-analytics is basically the analytics about efficient and predictive analytics. For efficient analytics, meta-analytics is performed over the previously analyzed and saved results in different scenarios. For example, ranking variations over locations, performance of available computing resources in different cases are further analyzed in meta-analytics. However, meta-analytics for efficient analytics may not provide temporal patterns since efficient analytics are only used when sufficient historical data are not available.

The meta-analytics of predictive analytics outcome is for the purpose of analyzing the pattern of changes in graph patterns over time and making inference about the pattern change (Cheng *et al.*, 2014). The aggregated graph for each predictive analytics is saved which becomes the input of the meta-analytics. The outcome of meta-analytics component will be a set of inferences that will help for better prediction by incorporating the inferences into the updated prediction model. For example, identifying any abnormal pattern (and/or abnormal subgraph) in the patterns of temporal graphs could be an effect of significant change at the corresponding location at that instance of time. The focus of the meta-analytics on predictive analytics outcome can go in many directions including tight clique analysis, mining frequent subgraphs, isomorphic graph analysis, etc. In this chapter, we focused on analysis of tight cliques and mining frequent subgraphs. Meta-analytics also suggests a schema for both efficient analytics and predictive analytics, so that further analysis can be done on the data generated over time as in the schema.

6.1 CLIQUE ANALYSIS

A clique is defined here as tight clique if the weight of any edges between any two nodes is above a threshold value (e.g. >0.50). In the graph approach (as explained in Chapter 3), since the edge weight is determined by the correlation between the clusters connected through the edge, any group of spatial clusters correlated to each other by a coefficient above the threshold is treated as tight clusters (e.g., tight clique).

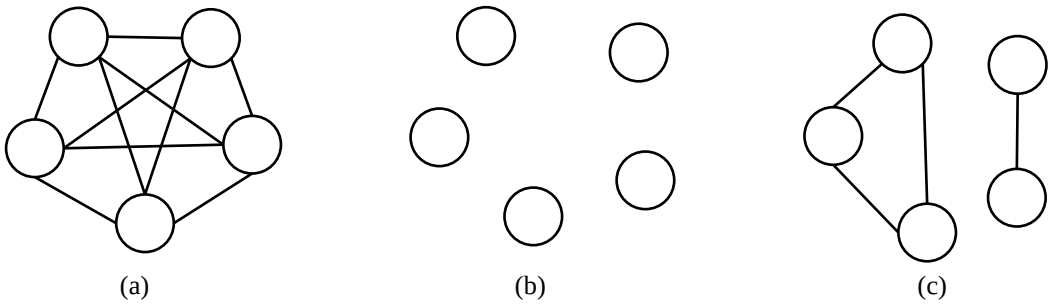


Figure 6.1. Possible clique cases in a graph. (a) Best case ($\boldsymbol{K_n}$), (b) Worst case ($\boldsymbol{K_1}$), (c) Average case ($\boldsymbol{K_m; 1 < m < n}$)

Three cases are possible in the tight clique formation: $K_n, K_1, and\ K_{m,}\ where\ 1 < m < n$ (see Figure 6.1). In the first case (K_n), all the nodes in the entire graph are strongly correlated and the graph itself is the only one clique. This is the best-case scenario since computing is needed only for the root node (junction node) of the clique, which is only one node in the computing nodes (CN) in the bipartite graph, whose result can be used to estimate all the other nodes in the entire graph. However, in this scenario, it is possible to compute (instead of estimate) more nodes for more accuracy if available computing resources support the computation.

In K_1 , the correlation among any two nodes is weak (<0.50) and will leave a single node in each group. This is the worst-case scenario since all the nodes are candidate for computation. In this case, the top ranking clusters (nodes), as many as possible for available computing resources to process them, are fed for computation leaving the low ranking nodes. The accuracy of the outcome may be compromised a bit in this case but that is the best possible option for limited computing resources.

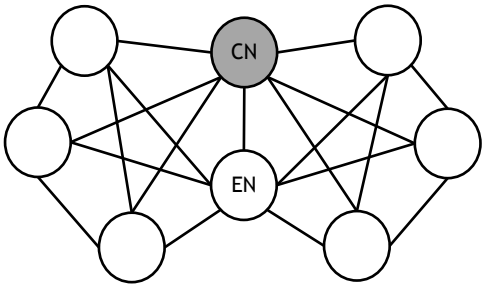


Figure 6.2. Example overlapping cliques. CN is the computing node and EN is the estimating nodes which are overlapped and common node in both cliques (left and right).

The average case scenario is K_m , where $|K_1| < |K_m| < |K_n|$. In $K_m$case, cliques could be both *non-overlapping* and *overlapping*. Non-overlapping cliques are those where no two cliques have a common node and overlapping cliques are those having common nodes. If the common node in the overlapping cliques (as shown in Figure 6.2) is a computing node(CN) then the overlapping node is used as computing node by all sharing cliques. But, if the common node in the overlapping cliques is an estimating node (EN), then its link with the largest weight that connects it with a computing node is used for estimation.

Possible cliques of similar patterns over the temporal graphs at different locations may infer that the changes of values at corresponding locations are similar. These prior inferences of

similarity or dissimilarity based on patterns in cliques may further contribute to faster prediction. For example, prediction for similar cliques may even be completed without computing all cliques but just one. That will even further reduce the computation time.

6.2 FREQUENT SUBGRAPH MINING

A frequent subgraph is a graph structure that occurs frequently across in a set of graphs. Mining frequent subgraphs in temporal graphs generated on data at each timestamp helps learn the graph changing patterns over time. A *frequent subgraph*, if exists in the graph pattern, may suggest similar inter-relationships among the nodes over time (Eberle & Holder, 2013). That means, if a subgraph is frequent in the temporal graphs, the correlation among the nodes within that subgraph is consistent which in turn helps robustness in subgraph prediction. This feedback could further optimize computation by reducing number of subgraphs that need to be processed. Mining frequent subgraph (H. Lee *et al.*, 2015) could be accomplished as described below.

A graph g(v,e) is a subgraph of another graph G(V,E), i.e., g $\subseteq G$ if there exists a subgraph isomorphism. A subgraph isomorphism is an injective function:

$$f: V(g) \rightarrow V(G) \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (6.1)$$

such that

$$\forall v \in V(g)$$

$$\forall (u,v) \in E(g), (f(u), f(v)) \in E(G)$$

To find a frequent subgraph (g) in a set (S) of n temporal graphs, we need to compute and compare (with a minimum threshold) the support of g in S where,

$$S = \{G_1, G_2, G_3, \dots\dots\dots., G_n\} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (6.2)$$

The supporting graph set (S_g) of g is:

$$S_g = \{G_i | g \in G_i, G_i \in S\} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (6.3)$$

Then the support of g is computed as:

$$s(g) = \frac{|S_g|}{|S|} \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (6.4)$$

So, g is a frequent subgraph of G if

$$s(g) \geq s_0 \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots \dots (6.5)$$

where, s_0 is a minimum threshold support.

Even if the above two meta-analytics approaches (clique and frequent subgraph) are in place, there may still be some clusters (nodes) neither part of any clique nor frequent subgraph. An analysis of patterns on the collection of these nodes also is another form of meta-analytics. Once a pattern revealed, the pattern may suggest appropriate handling method for these nodes.

6.3 SCHEMA FOR EA AND PA

Over the course of efficient analytics (EA) and predictive analytics (PA) the outcome of EA and PA can give an insight about the models, methods, and approaches used for EA and PA. Revealing the pattern and providing feedback to EA and PA after analyzing their outcomes would help to improve the performance and accuracy of EA and PA. Especially, spatiotemporal hotspots, that both EA and PA works on, are updated based on the feedback from meta-analytics. To that end, both EA and PA outcome could be formalized through a schema so that meta-analytics can look at the data in the schema for its efficient analysis and provide effective feedback to EA and PA

along with recommendation for relevant applications. A schema is a formal structure that helps managing data as supported by relational database management system (RDBMS). So, the schema recommended here brings basically the summary of spatiotemporal Big Data into a structure supported by RDBMS which makes the analysis easier and faster. The schema is recommended, as explained below, for EA and PA based on the features of the outcome for each.

6.3.1 Relation for EA

The relation for EA is given below,

$$\boldsymbol{EA}(E\_ID, Date, Temporal\_Range, Spatial\_Span, Data\_Size, Comp\_Unit, T\_Const, Clust\_Num$$
$$Task\_T\_U, Rank\_Time, Clust\_Time, Comput\_Time)$$

where,

E_ID – Unique ID for each computing task

$Date$ – Date when the task performed

$Temporal\_Range$ – Range of the temporal dimension of spatiotemporal data

$Spatial\_Span$ – The geospatial region that contains the data

$Data\_Size$ – Total number of data points/objects

$Comp\_Unit$ – Number of unit computing resources

T_Const – Time constraint

$Clust\_Num$ – Number of spatial clusters

$Task\_T\_U$ – Time required to complete the computation task for a unit data on a unit computing resource

$Rank\_Time$ – Time required to complete the ranking of the entire data

$Clust\_Time$ – Time required to perform the spatial clustering on the entire data

$Comput\_Time$ – Total time required to complete the computation task on the given computing resources

6.3.2 Relation for PA

The relation for PA is given below,

$$\boldsymbol{PA}(P\_ID, Date, Temporal\_Range, TimeStamps, Spatial\_Span, Data\_Size, Clust\_Num,$$
$$Comp\_Unit, T\_Const, CN, EN, Task\_T\_U\_CN, Task\_T\_U\_EN, Rank\_Time, Clust\_Time,$$
$$CN\_Time, EN\_Time, Pred\_Time, C\_Time, E\_Time)$$

where,

P_ID – Unique ID for each computing task

$Date$ – Date when the task performed

$Temporal\_Range$ – Range of the temporal dimension of spatiotemporal data

$TimeStamps$ – Number of timestamps in the $Temporal\_Range$. This provides the number of spatial graph in each timestamp.

$Spatial\_Span$ – The geospatial region that contains the data

$Data\_Size$ – Average (on $TimeStamps$) number of data points/objects

$Clust\_Num$ – Average (on $TimeStamps$) number of clusters generated over the $Spatial\_Span$

$Comp\_Unit$ – Number of unit computing resources

T_Const – Time constraint

CN– Average number of computing nodes

EN– Average number of estimating nodes

$Task\_T\_U\_CN$ – Average time required to complete the computation task for a computing node on a unit computing resource

$Task\_T\_U\_EN$ – Average time required to complete estimation for an estimating node on a unit computing resource

$Rank\_Time$ – Average time required to complete the ranking of the entire data

$Clust\_Time$ – Average time required to perform the spatial clustering on the entire data

$Pred\_Time$– Total time required to complete the data value prediction on the given computing resources

C_Time – Total time required to complete the computation task for computing nodes on the given computing resources

E_Time – Total time required to complete the estimation task for estimating nodes on the given computing resources

7.0 EVALUATION: RESULTS AND DISCUSSION

The method used for evaluation of the framework is through empirical results for representative spatiotemporal Big Data applications and proof of concept for the graph approach. Evaluation of algorithms proposed in the framework is performed by testing *completeness, correctness,* and *termination* for each algorithm on invariant condition(s) in *initialization, maintenance*, and *termination* phase. For empirical approach, there are both preliminary results on sample data of a spatiotemporal Big Data application and subsequent detail results for two different such applications. In both cases, we used computing platforms both in local environment which comprises computers of heterogeneous specifications and parallel and distributed cluster computing environment with uniform specifications of each computer in the cluster. To test completeness, all cases are covered to show that all possible input are processed by the algorithms. For correctness testing, it is shown that for each input, the algorithms produce expected output. And for testing termination, it is shown that for all cases the algorithms exit after producing expected output. In this chapter, the evaluation is presented through experimental results on data obtained from two spatiotemporal Big Data applications followed by a discussion of the results.

7.1 DATA AND EXPERIMENTS

We evaluate the developed methodology, analytics, and algorithms by using representative applications which involve analytics of massive amount of spatiotemporal data and real-time

response. For experimentation we used two major data sources along with other supporting datasets, as described below, which produce massive spatiotemporal data and two applications that require real-time response: 1) Air quality data from US EPA for computing least air pollution exposure routes (APE application) and 2) Road traffic data from NYSDOT for predicting traffic volume in NY streets (Traffic Application).

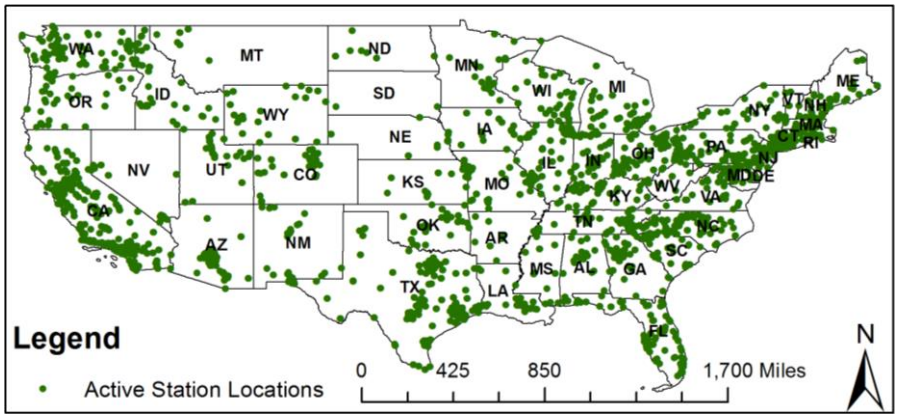
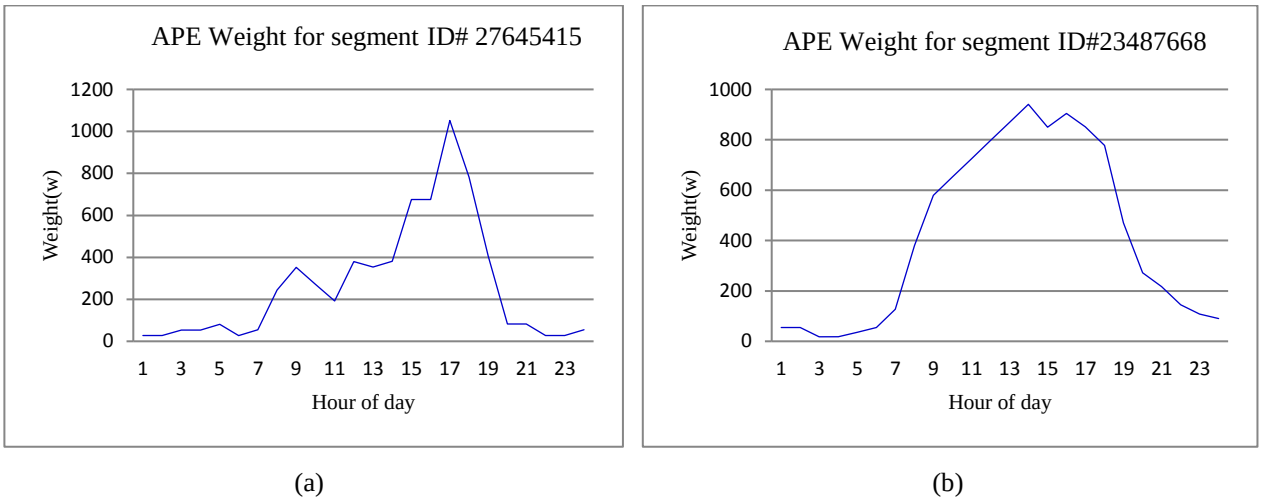


Figure 7.1 Distribution of monitoring stations in the USA; density is high in highly populated areas

7.1.1 Air Quality Data and Application

The source of the air quality data is the US EPA and the application based on this dataset is to compute routes with least air pollution exposure which requires real time air quality data. The US EPA has air pollution monitoring stations all over the U.S.A. which are not located uniformly (see Figure 7.1). The EPA provides real-time air quality data and forecasts Air Quality Index (AQI) for the geographic regions across the United States, Canada, and parts of Mexico. Air Quality Subsystem (AQS), administered by AIRNow gateway in coordination with EPA, obtains air pollution and meteorological measurements from national, state, and local agency monitoring

stations. AIRNow receives real-time *ozone(O_3)* and particulate matter (*$PM_{2.5}$)* data from over 4,000 monitoring stations and provides forecasts for more than 300 cities (EPA, 2011a). All data provided by AIRNow gateway are the contribution of more than 120 local, state, tribal, provincial, and federal government agencies. Official regulatory air quality data could be obtained from EPA's AQS (EPA, 2011b). Each monitoring station records the observed level of pollution in real time and updates hourly the EPA data repository managed by AQS. AIRNow gateway provides data, which is available in flat file format, to public through an FTP site and API.



(a) (b)

Figure 7.2. Weights at same set of timestamps at different locations (a) For road segment ID# 27645415, (b) For segment ID# 23487668

Two types of data files are used from EPA: *monitoring station locations* and *hourly data*. The monitoring station data file contains station information including location (latitude, longitude), time offset, and pollutants it monitors. The observed pollution is recorded in a data file hourly. The hourly data files contain the value of pollution observed at each monitoring station for each pollutant for which the station is responsible. Thus, for one year there are 8,760 such hourly

data files. For this experiment, the AQIs were computed based on Ozone (O_3) and $P_{2.5}$ exposure. Sample air pollution exposure weight (APE) of the data is shown in Figure 7.2 where at same timestamp the measures of air quality are different at different locations.

The AQI at a monitoring station location is calculated by using a standard equation based on six ranges of pollutant concentrations and corresponding index table (EPA 2009), as:

$$I_p = \frac{I_{Hi} - I_{Lo}}{BP_{Hi} - BP_{Lo}}\left(C_p - BP_{Lo}\right) + I_{Lo} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7.1)$$

where I_p is the index for pollutant p, C_p is concentration of pollutant p, BP_{Hi} is upper bound (breakpoint) of the range that C_p is in, BP_{Lo} is lower bound (breakpoint) of the range that C_p is in, I_{Hi} is the AQI value corresponding to BP_{Hi}, and I_{Lo} is the AQI value corresponding to BP_{Lo}. Basically the pollution index (I_p) is computed for all pollutants (p) using the above equation and the breakpoint (range) table at the monitoring stations. The maximum value of I_p over all pollutants at a monitoring station is considered as the AQI for the station at that given time. Thus for a station the AQI is determined as:

$$\text{AQI} = \max\left(\text{I}_\text{p}\ \forall \text{p}\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7.2)$$

Another data source that needed is the road network database which is used to compute actual routes. The road network database for the experiments was derived from NAVTEQ (CIA, 2012; NAVTEQ, 2010 in shapefile format for Pennsylvania (PA) that contains total 606, 995 road segments and, within PA, Pittsburgh road network which consists of 40,606 road segments was also used for local data (Pittsburgh, PA) ranking and hotspot analysis. The reason for using of NAVTEQ street network in the experiments is that it provides attributes, such as speed category, road class, and segment length that are needed for APE weight calculation and routing. Compared to the road network used in the experiment, road network in reality is much larger which pose the

challenge of real-time data-intensive computing. For example, the entire road network of the USA is 4,378,156 miles long and contains around 5,5802,096 road segments [TopTen, 2014 #686). Based on the time it takes to compute APE for the road segments in Pittsburgh, the estimated time to compute and update the APE weight for one instance of data file for the entire US road network is about 196 hours for a computer (e.g. 2.5 GHz and 4GB memory). This computation time is not acceptable for this application since a new data file is available every hour with the latest air pollution updates. The challenge is to bring the computing time from over 196 hours down to minutes.

Using the above data and applying the APE weight computing method (Sharker & Karimi, 2014), a weight is to be computed for each road segment of the entire road network and based on the APE weight computed routes provide least air pollution exposure. So, the data file containing the hourly air quality data at different locations and the road network data are used as input data whereas, the weight computing method and computing least air pollution exposure route are the computing tasks for the experiment.

7.1.2 NY Road Traffic Data

NYS traffic data are collected in real time through sensor network connecting eleven regions: Capital District (Region 1), Mohawk Valley (Region 2), Central New York (Region 3), Genesee Valley (Region 4), Western New York (Region 5), Central Southern Tier (Region 6), North Country (Region 7), Hudson Valley (Region 8), Southern Tier (Region 9), Long Island (Region 10), and New York City (Region 11) (NYSDOT, 2010). For each region, continuous traffic volume data, along with other related data, are observed, recorded, and made available for public.

All available traffic data from NYSDOT is verified and passed NYSDOT quality control standards maintained by the Highway Data Services Bureau (HDSB). The data collected by the NYSDOT is used to assess transportation needs and the road network system performance, to develop highway planning, for route planning, and in design of projects related to traffic.

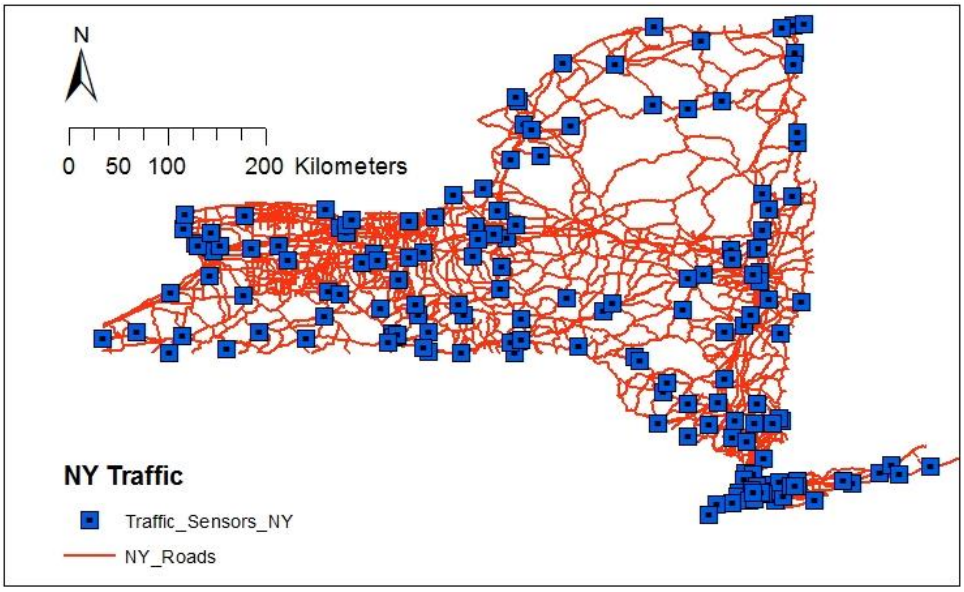
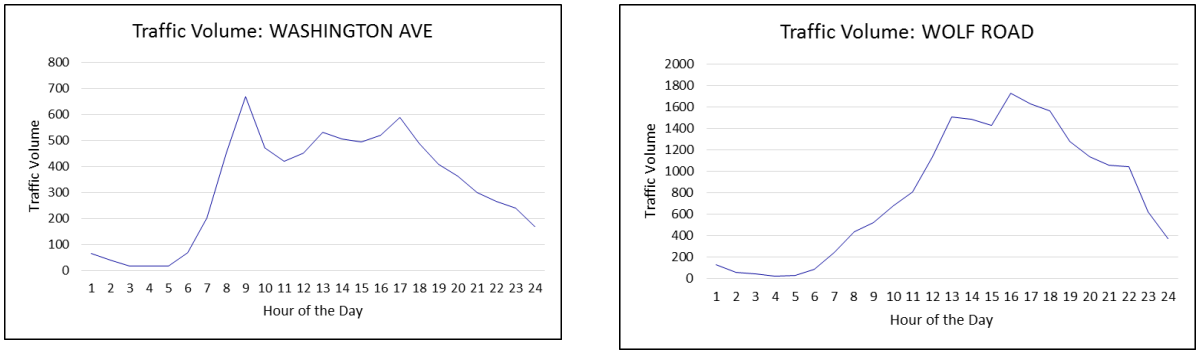


Figure 7.3 Active traffic monitoring stations in NY overlaid on NY Road network

NYSDOT uses two traffic monitoring systems: 1) Statewide Traffic Monitoring System and 2) portable traffic counter program. In Statewide Traffic Monitoring System, 177 permanent continuous count stations (see Figure 7.3) collect traffic volume, speed, vehicle classification and weigh-in-motion data around the clock. Portable traffic counter program takes inventory counts on the Federal and State highway systems, along with county and town roads. Approximately 12,000 counts of 2-7 days duration are taken annually in approximately 8,200 traffic control

sections. The entire NY road network has 1,111,162 is road segments. A snapshot of the data at the same set of timestamps at two different roads in Albany, NY is shown in Figure 7.4.



(a) (b)

Figure 7.4 Traffic volume at same set of timestamps at different locations in Albany, NY (a) Washington Ave (b) Wolf Road

The application in this case is to predict the traffic volume for any road learning from the historical volume data available from NYSDOT for the same road. The computation time for the prediction is important since many activities in any city with heavy traffic, like NY, is planned based on the possible traffic volume. As the data here is updated hourly, computation for the prediction is also expected within hour.

7.1.3 Experiments Setting

For both datasets and applications, experiments were designed for adaptive computing module both in local computers (desktops and laptops) of heterogeneous specifications (processing speed, memory size) and remotely on parallel and distributed cluster computing environment with uniform specifications of each cluster computer. The same computing environments were used for

all subsequent experiments on analytics and predictions as applicable. The air quality data (for APE application) for the Pennsylvania road network database and the traffic volume data for NY road network were taken as input spatiotemporal data.

To learn the nature of the spatiotemporal data, before applying the methods and algorithms as proposed in this framework, a time series spectral analysis (Box *et al.*, 2015) was performed to analyze the data spectra to see the data distribution over time (see Figure 7.5). Using non-parametric regression techniques (another form of linear filtering) (Salcedo *et al.*, 1999), the time series data was decomposed to evaluate the trend of the time series (see Figure 7.6).

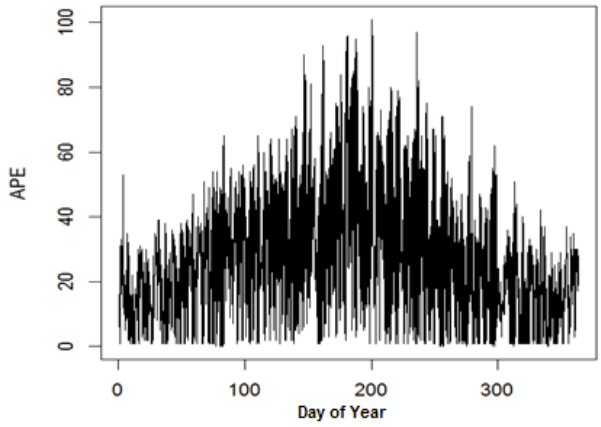


Figure 7.5. The spectral plot of time series data decomposed into periodic components

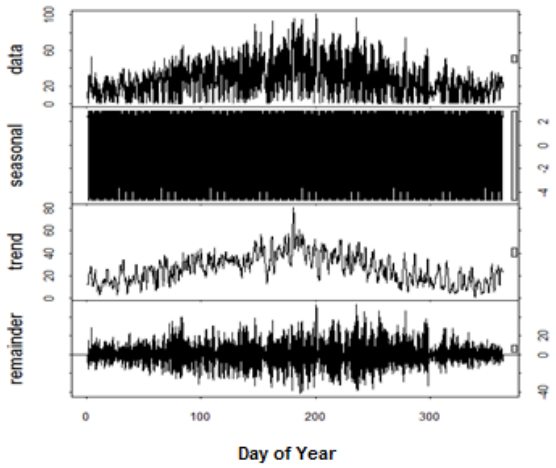


Figure 7.6. Seasonal decomposition of time series data (trend, seasonal, and remainder)

In the seasonal decomposition, the *trend* reflects the long term progression of the series, *seasonal* shows the seasonality which is a periodic change, and the *remainder* is the residuals from the seasonal plus trend fit. The trend is evident (see Figure 7.6) that the APE values change

significantly in the time window of one month. Potential differences in APE are shown using box plots for each time window (see Figure 7.7). These differences, demonstrated through boxplots, validate the ranking approach that is implemented on this time series data for hotspot identification.

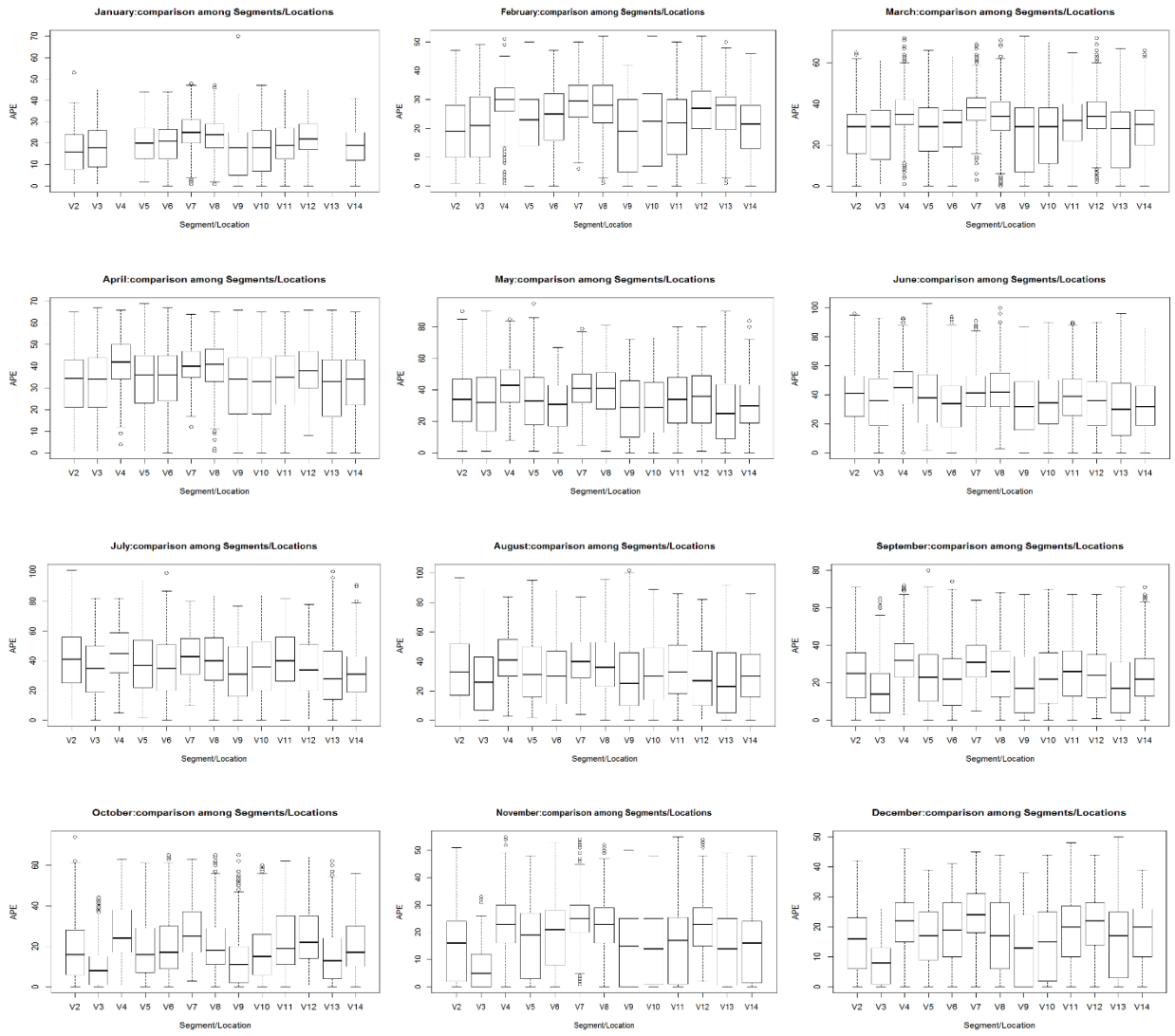


Figure 7.7 Box plot for each time window (January to December) demonstrating the differences in APE

A one-month time window was used to compute the variance in each window. The time window could be yearly, quarterly, and even hourly. The record set was ranked according to the variance of the time windows. Once the ranking was completed, the rank ordered dataset was fed as input to the adaptation algorithm for selecting appropriate amount of rank ordered data set from the top. For demonstration, the adaptation algorithm takes the entire Pittsburgh road network dataset (ranked) into account for computing to determine the amount of computing time it takes to complete the hotspots computations. The performance of a computer, with a given CPU speed and memory size, is measured in terms of computing time. The algorithm considers the situation in which the available computers are limited, each with same or different specifications. Given the limited computing resources, the adaptation algorithm determines the size of the input data points that the available computing resources can process in real time. Accordingly, the optimal number of records are chosen from the ranked record set starting from the top for computation.

To compare the effectiveness of using ranking and clustering together in our method, we conducted an experiment by computing accuracy on different data sizes for each of the three approaches: random- data is selected randomly from entire data set, *ranking only* – only top ranking data set is selected, and *ranking and clustering* - spatial clusters are generated over the entire data area and then ranked the data in each spatial cluster. In our method (*ranking and clustering*) a representative set of data is selected from each cluster which in turn represents the entire area. The results are demonstrated in Section 7.2.1. To demonstrate the time performance and its variation over different sizes of input data on different computing platforms, experiments were conducted on heterogeneous computers with different specifications. Pittsburgh road network dataset was taken as a test dataset in the experiment. For the time performance experiment on the road network of Pittsburgh, PA, six heterogeneous computers (windows, Mac) were used.

The experiments were conducted in three phases each on different size of data taken from the computed hotspots: (a) n = 10,151 (b) n = 20,303 and (c) n = 40,606.

For the predictive analytics component, using non-parametric regression techniques (another form of linear filtering), the time series data was decomposed to evaluate the trend of the time series. For data value prediction, the Box-Jenkins methodology (Box *et al.*, 2015) for ARIMA model selection was followed. The methodology works in three steps: *identification, estimation,* and *diagnostics*.

Identification step

A series of actions were performed in the identification step as follows.

- Examine the time plot of the data series
  - ✓ Visualize the trend (if any) and change in variance
  - ✓ Identify outliers, missing values, and breaks in the data
  - ✓ Transform the data if needed using logarithms, differencing, and/or de-trending
- Examine the ACF and PACF
  - ✓ Compare the sample ACF and PACF of different ARMA models and based on the comparison results estimate suitable models and its appropriate p, d, and q parameters
  - ✓ Examine with several models on EPA data with/without differencing

Estimation step

In the estimation step, we estimate ARMA models and examine the models for various coefficients. The goal is to come up with a model that has significant coefficients estimated and satisfactory goodness of fit (Ho & Xie, 1998).

Diagnostic step

Once a model is developed, it still needs to go through a number of diagnostic processes to make sure that the obtained model is a good fit. For a model that is a good-fit for the test data, the residuals from the model should go through a white noise test as follows:

- ✓ Normality test by looking at the histogram of the residuals and also by using a quantile-quantile (Q-Q) plot
- ✓ Examine independence by using ACF and PACF of the residuals, which is expected to be a white noise
- ✓ Ljung-Box-Pierce statistic test for the magnitude of the autocorrelations of the correlations as a group
- ✓ Investigate Akaike Information Criteria (AIC) and Schwarz's Bayesian Criterion (SBC) for goodness of model fit

For the data analytics and implementation of the algorithms in the proposed framework, we used RStudio (RStudio, 2011) which is an open source software based on R (an open-source language for statistical computing and analysis) and JMP (JMP, 1989), a SAS based statistical analysis software (academic version). For geostatistical analysis and mapping, we used ArcGIS (ArcGIS, 2014) a geographic information system (academic version). As the distributed computing platform, we used a cluster computing environment where 20 computers (computing unit on cluster computing environment) each with 8GB memory allocation were used for the computation for both datasets and applications.

For experimentation, we varied the number of nodes on the cluster environment to verify if computing can be performed within the time constraint. The total times that is required for computing on the entire dataset for both applications are computed for each of the computing

resource arrangement. One hour time constraint were used since for both applications the computation is required to be completed in one hour. For the given time constraint, for each application, the amount of data from hotspot that could be computed within the time constraint and the corresponding accuracies are reported.

7.2 RESULTS AND VALIDATIONS

The results are evaluated and validated both logically and empirically. For each algorithm, we did the validation logically (by testing invariant condition in all three steps, initial, maintenance, and termination) for algorithm correctness and completeness followed by analyzing implementation results on real data for two independent spatiotemporal data sets.

7.2.1 Comparison of Data Selection Methods

In terms of metrics to compare the methods, random selection and rank only selection methods are not directly comparable with our method of selecting data by ranking and clustering. The proposed methodology (ranking and clustering) identifies a representative data set for the entire data optimized for available computing resources and time constraint to apply it's computation task(s) and guarantees to provide solution within the time constraint whereas, applying the computing task directly to the random raw data (or only top ranking data) has no guarantee to complete in time constraint and that does not represent changes on other data in the

area that were not selected for computation. In terms of accuracy, we conducted an experiment by computing accuracy on different data sizes for each of the three methods to compare.

Table 7.1 Comparison of Accuracy among the three approaches

Approach	Data Size [Data Points]	Accuracy [APE]	Accuracy [Traffic]
Random	25000	56.35	57.57
	50000	60.21	62.33
	75000	62.46	64.67
	100000	63.21	65.11
	150000	63.1	66.03
Ranking Only	25000	57.53	58.91
	50000	61.76	63.44
	75000	63.33	65.23
	100000	63.91	66.1
	150000	64.01	66.93
Ranking and clustering	25000	72.33	73.47
	50000	76.15	77.83
	75000	78.23	79.13
	100000	79.17	80.15
	150000	79.93	80.73

The comparison results are shown in Table 7.1. It is evident from the comparison that the accuracy of our approach (ranking and clustering) is better than the other two as shown in Figure 7.8. The accuracy of random selection and rank only selection are comparable while the accuracy of our method outperforms both. For instance, while the range of accuracy for both random and ranking only selection methods are 56% to 67%, the accuracy range for ranking and clustering method is 72% to 81%.

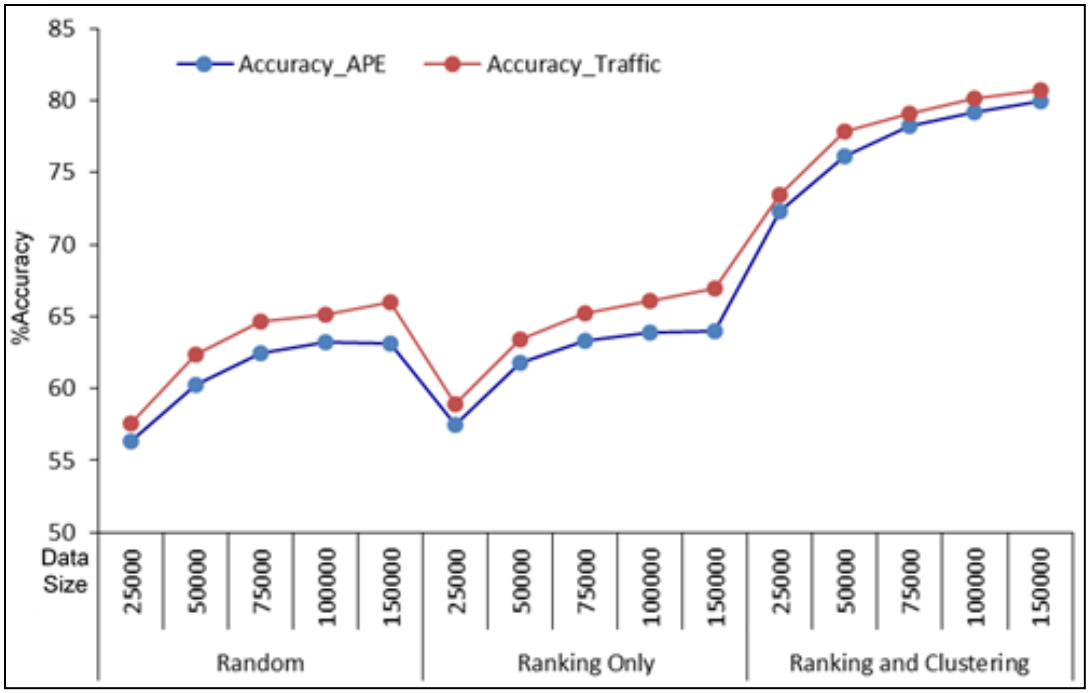


Figure 7.8. Comparative accuracies of random selection, rank only selection, and ranking and clustering methods. The accuracy range for both random and ranking only selection methods are 56% to 67% whereas, the accuracy range for ranking and clustering method is 72% to 81%

7.2.2 Preprocessing and Adaptive Computing

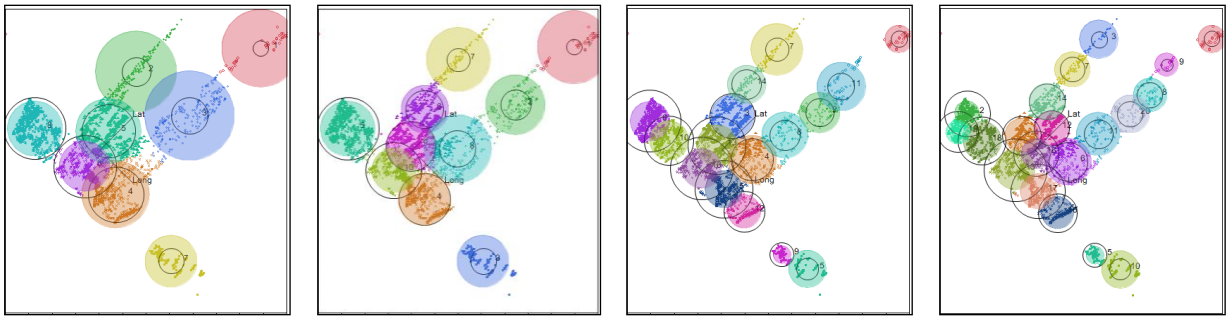
In the preprocessing phase, the data is first ranked using the ranking method using RStudio. The ranking result (the selected subset) based on the window variance (with window size one month) is shown in Table 7.2 for Pittsburgh PA. The first column (Seg/Loc) shows the segment ID (which is used to locate the segment in the road network map) and the values in subsequent columns are the computed ranks for the segments over each month. Monthly ranking is performed since we used one month time window. The higher the rank value, the lower the priority (rank); the top rank starts from one (1). The ranking method is important since through ranking, the top priority data points are selected to be computed and updated. Ranking method helps selecting the computing

nodes within each cluster so that top priority data points are used in computation ensuring better estimation for the lower priority data points. This is just a subset as example of ranking results.

Table 7.2. A subset of the computed rank over the months (time window) for road segments

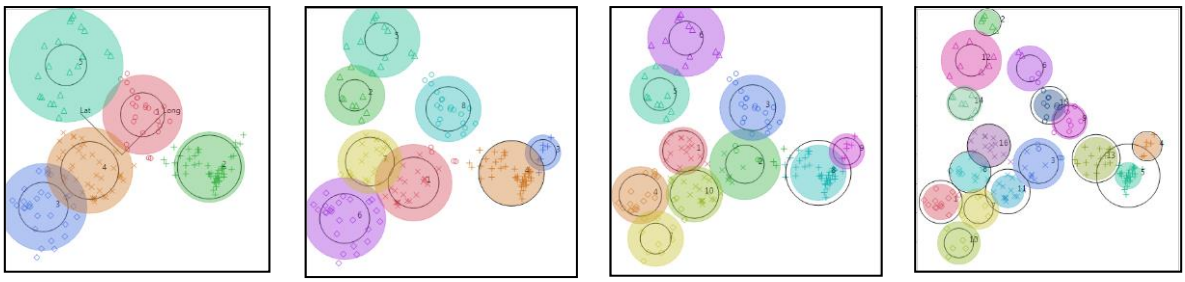
Seg/Loc	Rank											
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
420030008	6	6	7	6	7	5	3	6	7	2	7	7
420030010	3	3	4	2	1	3	4	8	11	13	13	13
420030067	12	11	12	12	12	12	12	12	12	9	9	11
420031005	5	7	6	7	5	2	2	3	5	5	8	9
420050001	7	8	8	8	10	6	7	2	3	4	6	6
420070002	11	13	13	13	13	13	13	13	13	12	12	12
420070005	8	10	10	11	11	11	10	11	8	8	11	2
420070014	2	2	1	1	3	4	6	4	2	11	4	1
421250005	1	1	2	3	4	7	5	5	4	3	3	3
421250200	4	4	5	5	8	9	8	7	6	1	1	5
421255001	10	12	11	10	6	8	9	9	9	7	10	10
421290006	13	5	3	4	2	1	1	1	1	6	2	4
421290008	9	9	9	9	9	10	11	10	10	10	5	8

In ranking method, data points in highly polluted area could be ranked low and vice versa since rank is computed based on variation in changes, not level of pollution; higher the change higher the rank. The rank is, in fact, computed following a relative ranking procedure, which means that the level of variation may be different for the same rank value for different datasets. This is because, no matter how much variation a dataset has, the adaptation algorithm should be able to scale up/down any dataset based on available computing resources.



(a) (b) (c) (d)

Figure 7.9. Different clustering over PA APE dataset. (a) 8 clusters; (b) 10 clusters; (c) 16 clusters; (d) 20 clusters



(a) (b) (c) (d)

Figure 7.10. Different clustering over NY Traffic Data (a) 5 clusters; (b) 8 clusters; (c) 10 clusters; (d) 16 clusters

Validation of Preprocessing Algorithm (Algorithm 3.1)**:**

Invariant condition: G is a complete graph

1. *Initialization*: Initially, G is a complete graph containing null vertex and edge
2. *Maintenance:*
   a. For any time stamp t_m, G is a complete graph
   b. For time stamp t_{m+1} (m+1 < n), G is also a complete graph because for each time stamp the data set is ranked, clustered, clusters are assigned as vertices, small (≈ 0) weights assigned to all edges among the vertices, and a complete graph is created

3. *Termination:* After $t = t_m$, the algorithm terminates producing a set of complete graphs for all time stamps.

In preprocessing algorithm, the spatiotemporal data is clustered using K-means clustering through JMP. Different number of clusters are generated for each data set as shown in Figure 7.9 for PA APE dataset and in Figure 7.10 for NY Traffic data. Different Ks (number of clusters) are tried such as for PA APE dataset 8, 10, 16, and 20 clusters and for NY Traffic data 5, 8, 10, 16 clusters are generated for each timestamp. The size of each colored circle indicates the cluster size. Once the clusters are generated, a complete spatial weighted graph is created considering each cluster as a node of the graph and assigning a very small (≈ 0) weight for each edge of the complete graph (see Figure 3.1). Such graph is represented by JMP using the matrix representation of weighted graph (see Table 7.5 for 7 clusters and Table 7.6 for 8 clusters). The weight matrix of the graph is later updated by inter-cluster correlation (discussed later in this chapter, see Figure 7.18) coefficients as the new weights.

Validation of Adapting Computing Algorithm (Algorithm 3.2)**:**

Invariant condition: TopRS and TopClust provide top candidate data

1. *Initialization*: Initially, both TopRS and TopClust contain no data, so the condition holds
2. *Maintenance:*
    a. For unavailability of historical data, TopRS contains top ranked Ct amount of data points from RS (step 2-8); the invariant condition holds.
    b. For available historical data, m top ranked clusters are assigned to TopClust and the condition still holds.

3. *Termination:* The algorithm terminates after returning TopRS or TopClust which still contains the top ranked data for computation.

The adaptive computing algorithm selects optimum number of data points from the ranked data set so that available computing resources can handle the data within time constraint. Accordingly, for Pittsburgh road network, this algorithm selected top ranked data points (TopRS) for all three cases: 1) All-Data, n=40606, 2) Optimal-data, n=20303, and 3) Partial-Optimal-data, n=10151 as explained in Section 4.2.

7.2.3 Analytics Algorithms

The EA algorithm works on the rank ordered dataset and once the graph is created. Since both of the data set we have here contains historical data, which is especially suitable for predictive analytics, JMP randomly chose sample data as a subset of the entire temporal data and used this subset (with no historical data) for the EA. JMP generated clusters and represented as a matrix of a complete weighted graph as discussed above. If the size of the subset is beyond the capacity of available computing resource to handle, JMP again uses stratified random sampling to obtain suitable amount of data. Finally, RStudio is used to apply the computation.

Validation of Efficient Analytics Algorithm (Algorithm 4.3)**:**

Invariant condition: *ds* is spatiotemporal hotspot (top priority data) and *result* provides computing results

1. *Initialization:* Initially, ds = null and computing task on ds provides null result, so the condition holds
2. *Maintenance:*

a. If the size of entire data is within the capacity to compute, ds contains the entire data and result provides the outcome of the computing task on ds; the condition holds.

b. If the size of entire data is beyond the capacity to compute, SD is randomly sampled, clustered, and in each cluster the sampled data is rank ordered and accumulated as a data frame into *ds* which again contains the spatiotemporal hotspots and the computing task on ds provides results on ds (steps 7-14); the condition still holds.

3. *Termination:* The algorithm terminates after producing *ds* and applying computing task (f_c) on ds.

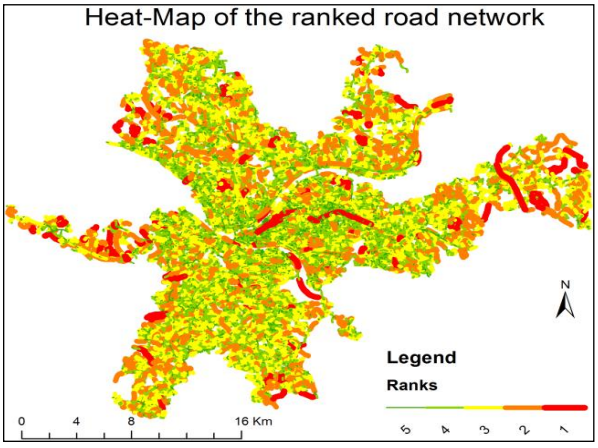


Figure 7.11. Heat map of Pittsburgh road network. The red segments are high priority segments ("hotspots") while the green segments are low priority segments

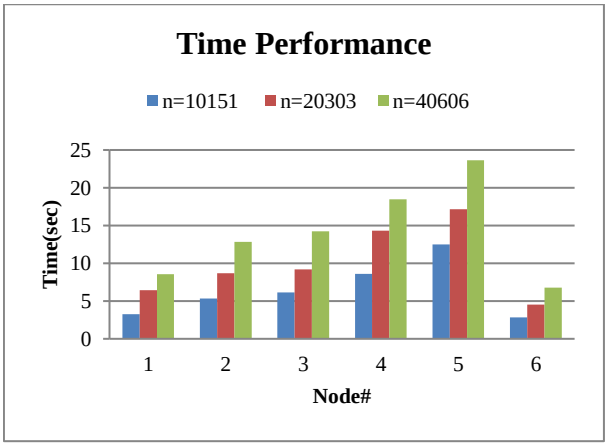


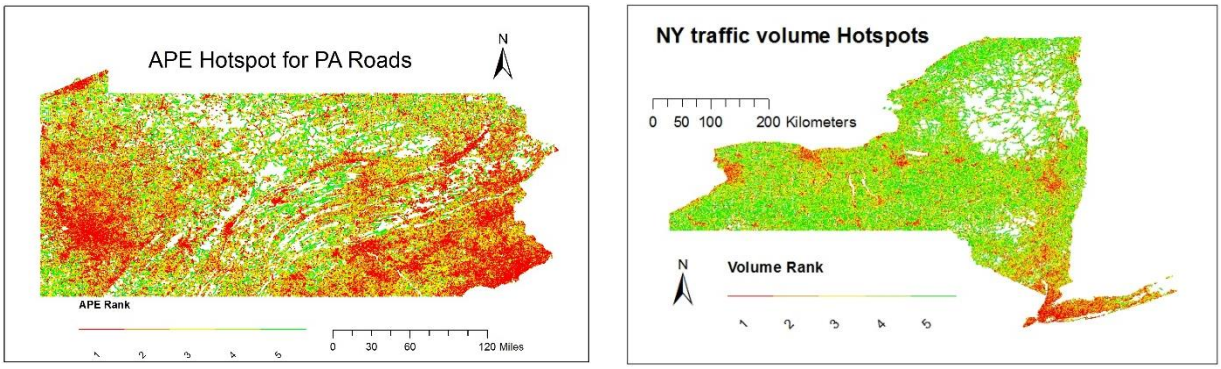
Figure 7.12. Comparison of weight computation times for different input data sizes on different computers

The ranking results (ds) and the computed hotspots in real dataset is better visualized through a heat-map as shown in Figure 7.11 where hotspots in Pittsburgh road network is demonstrated by a normalized and colored ranks from 1 (red) to 5 (green). Table 7.3 demonstrates

the computation time for one hourly data for the Pittsburgh road network on different computers and for different input data sizes. The time comparison is shown in Figure 7.12. Given the size of the experimental dataset used in this experiment which is a subset of the entire USA road network, the computation time was less than an hour. The resulting hotspot maps for both full datasets (PA APE and NY Traffic) are shown in Figure 7.13 and the computation performances are compared later.

Table 7.3. Computation time of APE weights for a sample set of Pittsburgh road segments

Memory (GB)	CPU Speed (Ghz)	OS (Bit)	Time(Sec) n=10151	Time(Sec) n=20303	Time(Sec) n=40606
6	2.53	64	3.25	6.44	8.55
4	2.2	32	5.35	8.67	12.83
2	2.2	32	6.12	9.18	14.24
1	1.33	32	8.6	14.33	18.45
0.5	1.33	32	12.5	17.15	23.65
8	2.8	64(Mac)	2.85	4.55	6.76



(a) (b)

Figure 7.13. Computed hotspot maps. (a) PA APE, (b) NY Traffic

Validation of model selection algorithm for data value prediction (Algorithm 5.1)**:**

Invariant condition: Optimum model for spatiotemporal data value prediction

1. *Initialization:* Initially, Model = null and the condition holds
2. *Maintenance:*
    a. The input data is tested for stationarity; both for real data and it's difference variable(steps 2-6)
    b. For the stationary response variable, all the models are executed and the statistics (AIC, BIC, R2, -2LogLH, MAPE, MAE, Mu, Sigma) for each model are used as a vector (V) forming a weigh matrix for the models (steps 7-10).
    c. Computing the overall weight for each model and comparing the weights, the model with minimum (smaller the better) weight is returned (steps 11-19). This holds the condition of providing optimum model selection.
3. *Termination:* The algorithm terminates after exhausting and executing all the models. After termination, Model (variable) still has the optimum model satisfying the in variant condition.

In the predictive analytics, the model selection algorithm, for data value prediction, selects appropriate prediction model for a given data set is and the algorithm is implemented through JMP. A summary statistics for both the original variable (Y) and its difference variable (d.Y) is given in APPENDIX B. For the stationarity test, both ACF and PACF are computed (see APPENDIX B). Using the ARIMA based time series analysis tools in JMP, five models are examined and the prediction is performed by the best model identified. The comparison of the non-seasonal models, AR, MA, ARMA, and ARIMA, are shown in Figure 7.14 for PA APE dataset. The red line is the prediction line while the blue points are the observed values. The red line after the blue vertical

line represents predicted values. The predicted values show the trend of values by each of the non-seasonal models.

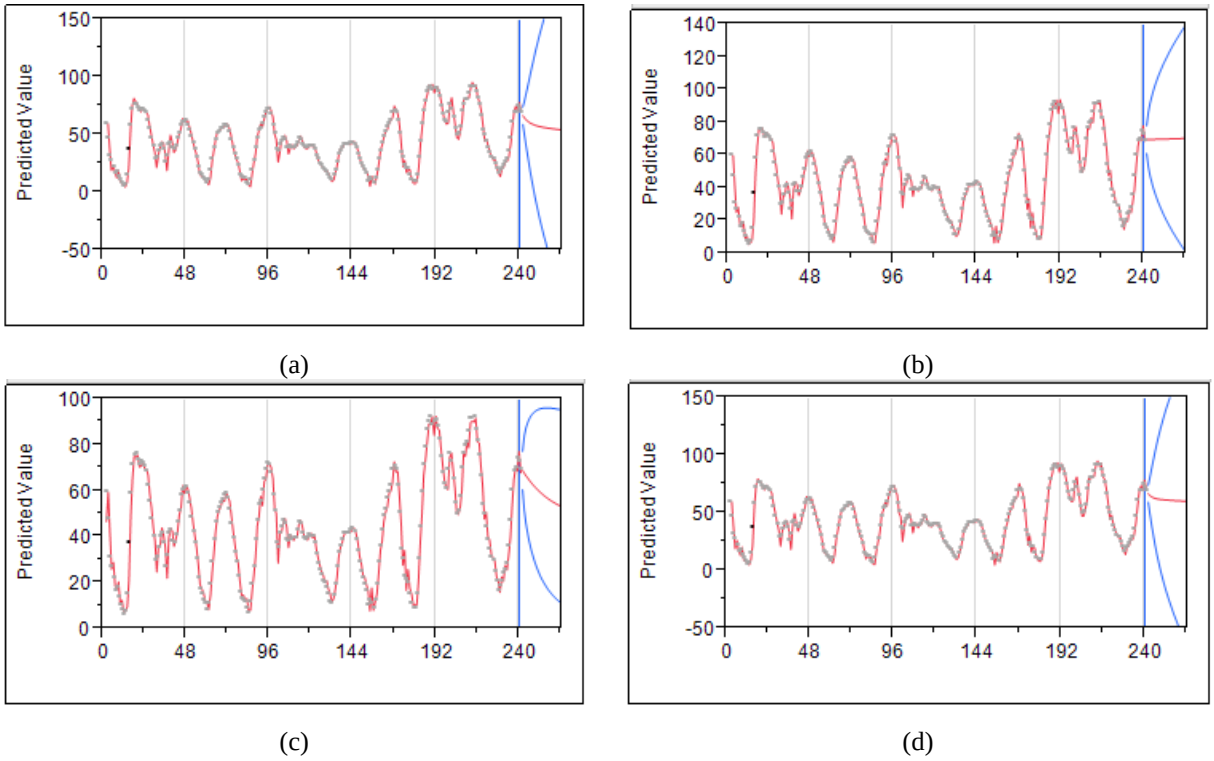


Figure 7.14. Different prediction models for PA APE dataset. (a) AR(1,1), (b) MA(1,1), (c) ARMA(1,1), and (d) ARIMA(1,1,1)

To test the goodness of fit (Haas, 1995) for all these non-temporal prediction models, we examined the following test statistics:

- ✓ AIC/SBC – Akaike's Information Criterion (AIC), Schwarz's Bayesian Criterion (SBC). SBC is also known as Bayesian Information Criterion (BIC); smaller values of these criteria indicate better fit of the model
- ✓ R^2– a percentage of variance that the model can explain; the larger the better

- ✓ –2LogLikelihood – maximum likelihood function evaluated at the best-fit parameter estimates; smaller values indicate better fits
- ✓ MAPE – Mean Absolute Percentage Error; smaller the better
- ✓ MAE – Mean Absolute Error; smaller the better

Among the non-seasonal models, ARIMA (1,1,1) has the best fit given the pattern of the time series data. The claim is further confirmed by the test statistics—where AIC is 1315.64, SBC is 1326.07, R^2 is 0.97, -2LogLH 1309.64, MAPE is 9.09, and MAE is 2.83—for ARIMA(1,1,1). The AIC, SBC, R^2, -2LogLH, MAPE, and MAE for ARIMA (1,1,1) are the smallest (smaller the better) among all the non-temporal predictive models while the R^2 is sharing the top position along with AR(1) demonstrating the superiority of ARIMA(1,1,1) over others. The comparison of the test statistics are shown in Table 7.4.

Table 7.4. The comparative statistics of the models in terms of performance metrics

Model	DF	Variance	AIC	SBC	R^2	-2LogLH	MAPE	MAE
Seasonal ARIMA(1, 1, 1)(1, 1, 1)12	222	13.32	1269.03	1286.16	0.96	1259.03	8.85	3.07
ARIMA(1, 1, 1)	236	14.16	1315.64	1326.07	0.97	1309.64	9.09	2.83
ARI(1, 1)	237	14.78	1324.83	1331.78	0.97	1320.83	9.11	2.91
IMA(1, 1)	237	18.59	1379.39	1386.35	0.96	1375.39	10.42	3.32
ARMA(1, 1)	237	18.16	1383.93	1394.37	0.96	1377.93	10.76	3.36

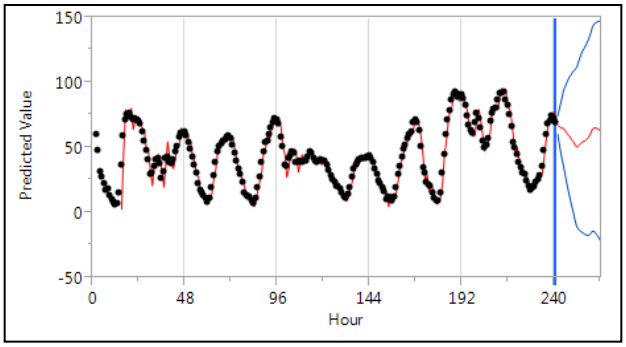


Figure 7.15. Prediction plot for SARIMA (1,1,1)(1,1,1)12. The black dotted line is the actual one while the red line is the predicted one

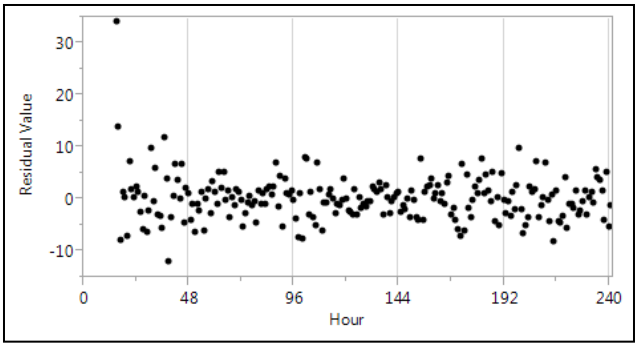


Figure 7.16. Residual plot for SARIMA (1,1,1)(1,1,1)12. Random residual plot ensures that the model is stable

The contribution of temporal parameters is demonstrated by the SARIMA (1,1,1)(1,1,1)12 predictive model and the result of the prediction is shown in Figure 7.15. The prediction line is significantly different in SARIMA compared to the non-temporal models. The test statistics (as shown in Table 7.4) in SARIMA outperforms (the shaded row) other models in terms of AIC, SBC, -2LogLH, and MAPE while the R^2 and MAE are also close to the ARIMA model. To further confirm the suitability of SARIMA, the residual plot is obtained, as shown in Figure 7.16, which demonstrates that the residuals look random. The randomness of residuals means that the model satisfies required assumptions (e.g., zero mean, constant variance) (Box & Pierce, 1970) and the model is stable.

The model comparison is further demonstrated in the combined forecasting plot as shown in Figure 7.17 with 95% confidence interval along with both the ACF and PACF for residuals. The forecasting plot for SARIMA (1,1,1)(1,1,1)12, aqua color, demonstrates the most appropriate forecasting pattern(compared to the actual, the yellow, pattern) given the time series data over different locations. The plots for both residual ACF and residual PACF, bottom left and right, respectively, shown in Figure 7.17, demonstrate that the SARIMA based model (aqua line) tails

off before all the other models. This tailing off earlier is another important indicator which confirms the claim that SARIMA based model outperforms others. Thus, the model selection algorithm selected SARIMA(1,1,1)(1,1,1)12 as the optimum model for the given data set.

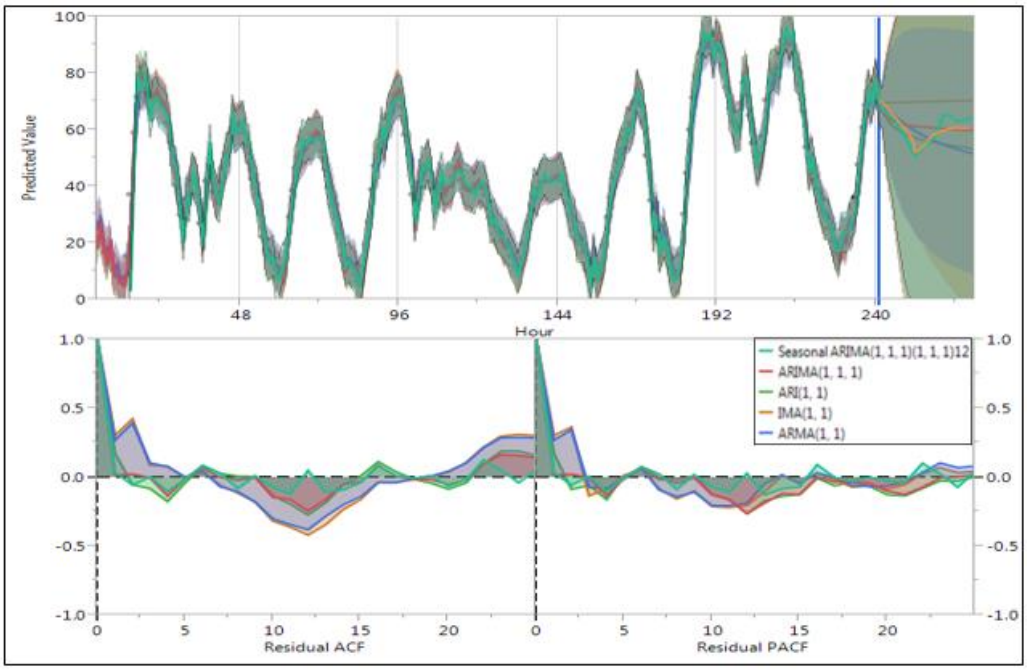


Figure 7.17. Combined prediction plot in 95% confidence interval along with both the ACF and PACF for residuals. Prediction line for SARIMA(1,1,1)(1,1,1)12 (aqua color) outperforms others which is confirmed by the residual ACF/PACF plot that tails off before all others

Validation of Predictive Analytics Algorithm (Algorithm 5.2)**:**

Invariant condition: Edge weight between any EN to its CN is the largest

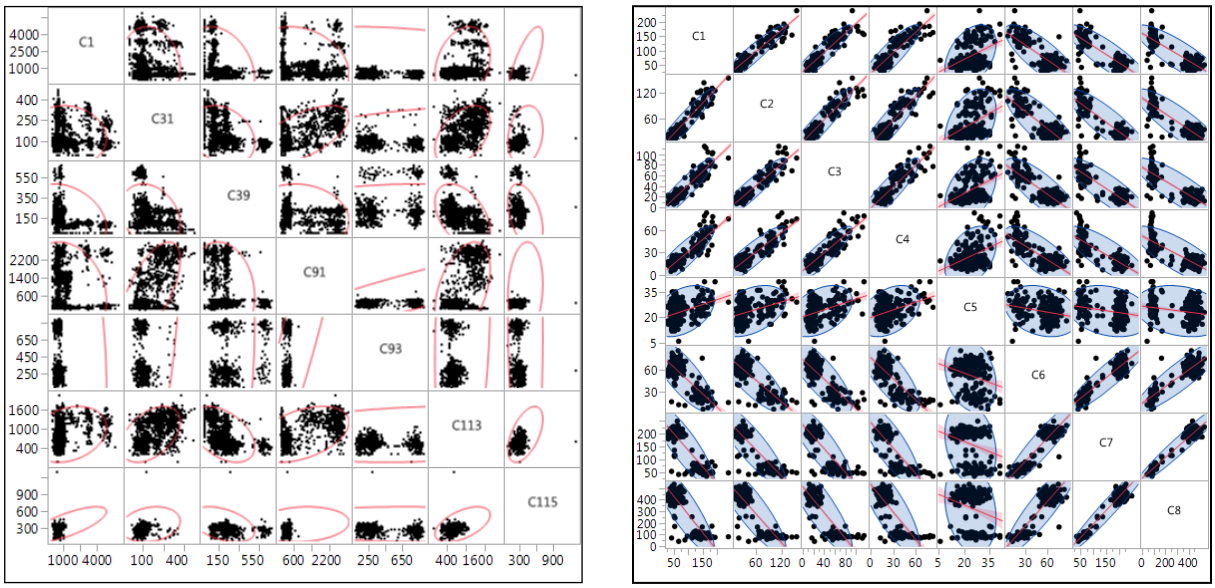
1. *Initialization:* Initially, $CNodes \leftarrow \text{NULL}; ENodes \leftarrow \text{NULL}; E2CLinks \leftarrow \text{NULL};$ the condition holds
2. *Maintenance:*

a. Once the weight matrix is updated by the inter-cluster correlation coefficients (steps 5-10), cliques in each graph are identified and in each clique ENodes and Cnodes are determined. E2Clink between ENodes and Cnodes is the largest because, e_q is taken as maximum weighted link (steps 11-21); the condition holds.

b. In the bipartite graph (step 22), each ENodes in EN partite is connected to it CNodes in CN partite through E2CLinks which is still the largest.

c. Computing on EN using f_c and estimating on EN (step 23-28) do not change the link weight between them, rather uses the weight for estimation. This ensures that the condition holds

3. *Termination:* The algorithm terminates after computing each node in CN partite and estimates each node in EN partite leaving BP unchanged; still holding the condition.



(a) (b)

Figure 7.18. Inter-cluster correlation matrix. (a) For 7 random clusters in APE (b) For 8 random clusters in NY traffic data

For the cluster prediction in predictive analytics, an inter-cluster correlation matrix is computed using JMP on the previously generated clusters. For seven random clusters in APE data, the inter-cluster correlation matrix is shown in Figure 7.18.a and for eight random clusters in NY traffic data, the inter-cluster correlation matrix is shown in Figure 7.18.b. The slopes of the red lines in the intersection of any two clusters represent the nature of the correlation (positive-zero-negative, strong-weak). The correlation coefficients for seven random clusters in APE data are shown in Table 7.5 and the correlation coefficients for 8 random clusters in NY traffic are shown in Table 7.6. Both Table 7.5 and Table 7.6 provide the actual value of correlation coefficients and eventually these values are used to update the weight of the graphs resulting the weight matrix of the weighted graph representation in JMP. The confidence interval for correlation coefficients are provided in APPENDIX B.1.

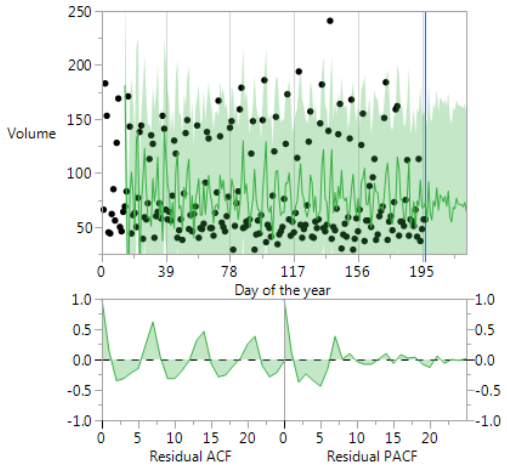
Table 7.5. Inter-cluster correlation coefficients for 7 clusters in APE data

	C1	C31	C39	C91	C93	C113	C115
C1	1.0000	-0.1126	-0.3476	-0.2372	-0.3720	0.2594	0.7360
C31	-0.1126	1.0000	-0.2570	0.6301	0.4808	0.4494	0.1910
C39	-0.3476	-0.2570	1.0000	-0.1409	0.0199	-0.4437	-0.4296
C91	-0.2372	0.6301	-0.1409	1.0000	0.9207	0.4395	0.1248
C93	-0.3720	0.4808	0.0199	0.9207	1.0000	0.1900	-0.0711
C113	0.2594	0.4494	-0.4437	0.4395	0.1900	1.0000	0.5441
C115	0.7360	0.1910	-0.4296	0.1248	-0.0711	0.5441	1.0000

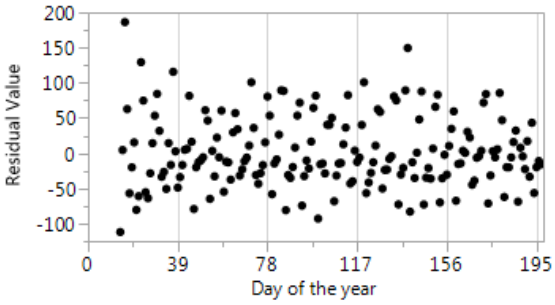
Table 7.6. Inter-cluster correlation coefficients for 8 clusters in NY traffic data

	C1	C2	C3	C4	C5	C6	C7	C8
C1	1.0000	0.9459	0.9282	0.8821	0.4102	-0.7954	-0.8633	-0.8647
C2	0.9459	1.0000	0.9324	0.8984	0.4117	-0.8168	-0.8722	-0.8714
C3	0.9282	0.9324	1.0000	0.8962	0.4182	-0.8076	-0.8597	-0.8521
C4	0.8821	0.8984	0.8962	1.0000	0.3992	-0.7677	-0.8134	-0.8043
C5	0.4102	0.4117	0.4182	0.3992	1.0000	-0.2583	-0.2460	-0.2325
C6	-0.7954	-0.8168	-0.8076	-0.7677	-0.2583	1.0000	0.9226	0.9009
C7	-0.8633	-0.8722	-0.8597	-0.8134	-0.2460	0.9226	1.0000	0.9701
C8	-0.8647	-0.8714	-0.8521	-0.8043	-0.2325	0.9009	0.9701	1.0000

Since seasonal ARIMA, SARIMA(1,1,1)(1,1,1)12, provides the best outcome for the spatiotemporal data set for PA APE dataset, we tried the same model for NY traffic data set as well for prediction within clusters and the result is shown in Figures below. The traffic prediction was performed in all three traffic situation: low traffic (1 am) sown in Figure 7.19, high traffic (9 am) sown in Figure 7.20, and average traffic as shown in Figure 7.21.

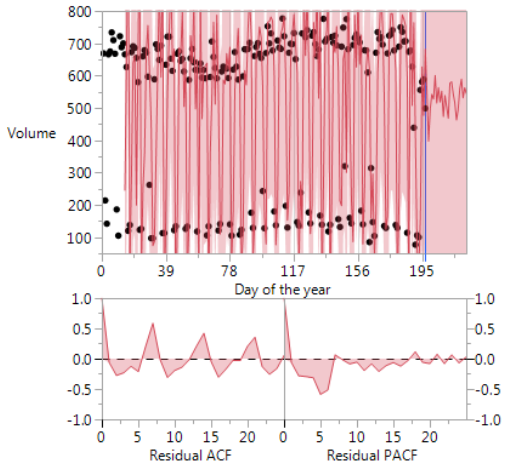


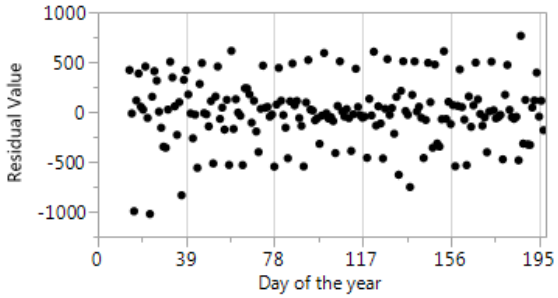
(a)



(b)

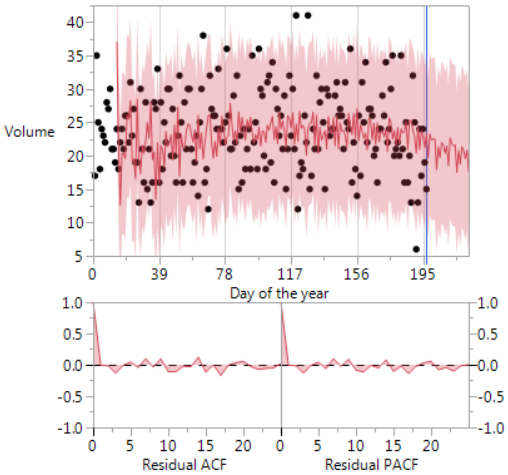
Figure 7.19. NY road traffic prediction at 1 am. (a) Prediction with ACF and PACF, (b) Residual plot

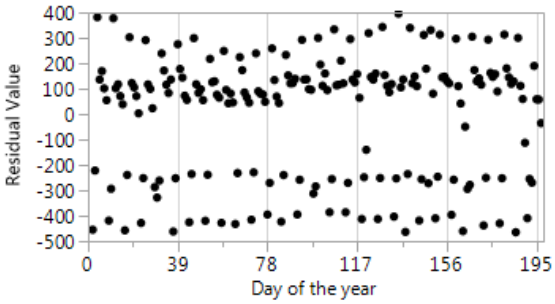




(a) (b)

Figure 7.20. NY road traffic prediction at 9 am. (a) Prediction with ACF and PACF, (b) Residual plot





(a) (b)

Figure 7.21. NY road traffic prediction at average hours. (a) Prediction with ACF and PACF, (b) Residual plot

Table 7.7. Performance metrics for Seasonal ARIMA model for the NY traffic data

Statistics	Hour 1	Hour 5	Hour 9	Hour 13	Hour 17	Hour 21
DF	179	179	179	179	179	179
Sum of Squared Errors	346189.372	6822.86314	13984507.9	108788.411	1067961.29	491448.988
Variance Estimate	1934.01884	38.1165538	78125.7423	607.756488	5966.26417	2745.52507
Standard Deviation	43.977481	6.17386053	279.509825	24.6527177	77.2415961	52.3977583
Akaike's Information Criterion	1964.92995	1238.31297	2639.19563	1750.44186	2165.40567	2012.82015
Schwarz's Bayesian Criterion	1981.00463	1254.38765	2655.27031	1766.51654	2181.48035	2028.89483
RSquare	-0.2666237	-0.1943607	-0.5558424	-0.3431755	-0.6491359	-0.2387488
RSquare Adj	-0.2949282	-0.2210503	-0.5906098	-0.3731906	-0.6859881	-0.2664303
MAPE	57.4141894	26.087528	94.3267712	92.762563	58.1739734	18.0630126
MAE	39.5557594	5.56497792	225.644998	22.1541738	62.3654323	39.3840224
-2LogLikelihood	1954.92995	1228.31297	2629.19563	1740.44186	2155.40567	2002.82015

The same experiment is conducted for each hour of the day for entire year. The statistics (metrics) for the goodness of fit of the model on NY traffic is given in Table 7.7. Also, the residual plot in Figure 7.19 through Figure 7.21 show randomness of the residuals which signals the stability of the model. So, the prediction in values within clusters using seasonal ARIMA shows stable results which is further confirmed by the goodness of fit metrics in Table 7.7.

Different graph patterns and properties (cliques and frequent subgraphs) are examined over clusters as generated above. The cliques were determined by the subset of the correlation coefficient matrix in JMP. The corresponding subgraph of the subset where the inter-cluster correlations are above threshold (>0.5) is treated as clique. The frequent subgraphs were determined by the subsets, by JMP, in the same correlation coefficient matrix that represents the weighted graph. Any subset that occurs in subsequent graphs on different time stamps were considered as the subset of frequent subgraph. The value prediction of the frequent subgraphs were

performed using the same model (seasonal ARIMA). The schema that meta-analytics suggested (as explained in section 6.3) are populated when the computation tasks are finally performed on the spatiotemporal hotspots, both for Efficient Analytics and Predictive analytics.

7.2.4 Justification of Threshold and Computation Results

The observation is that, a larger number of clusters show stronger correlation among clique nodes which help for better prediction. But a larger number of clusters produces more cliques which require more computations. So, there should be a trade-off between threshold of the correlation and the number of cliques which could be obtained empirically. Also, the number of frequent subgraphs are directly proportional to the number of clusters which suggest more clusters to consider.

To determine an appropriate threshold value for correlation coefficient while determining cliques, an experiment were conducted that shows the effects of increasing the threshold value. Experiments were conducted for r = 0.50, r = 0.70, and r = 0.90 on same set of 800, 400, 200, and 100 clusters separately. For each case, changes in both average clique size and number of single nodes are observed.

The results show that, for higher threshold, clique size decreases and number of cliques increase producing more single nodes. For instance, Figure 7.22 shows that the average clique sizes are over 7 for each set of 800, 400, 200, and 100 cluster graph for threshold r = 0.50 whereas, the average clique sizes are between 3 and 4 for the same set of 800, 400, 200, and 100 cluster graph for threshold r = 0.70 and the average clique sizes are between 2 and 3 for the same set of 800, 400, 200, and 100 cluster graph for threshold r = 0.90. Figure 7.23 shows changes in number

of cliques of size more than 2 and number of single nodes for different inter-cluster correlation coefficients (0.5, 0.7, 0.9) as threshold. Evidently, number of cliques (blue part bars in Figure 7.23) decreases and the number of single nodes (red part of the bars) increases for r = 0.90 compared the corresponding changes for r = 0.05, and r = 0.70.

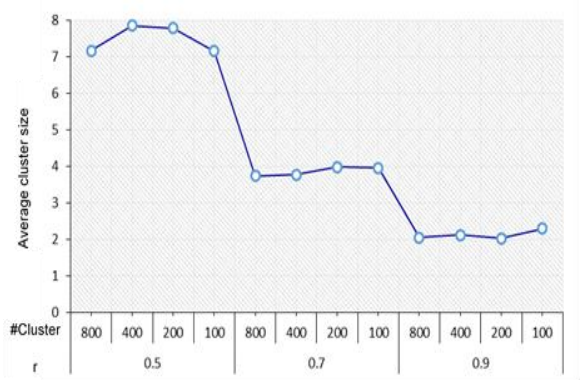


Figure 7.22. Changes in average clique size for different inter-cluster correlation coefficients (0.5, 0.7, 0.9) as threshold

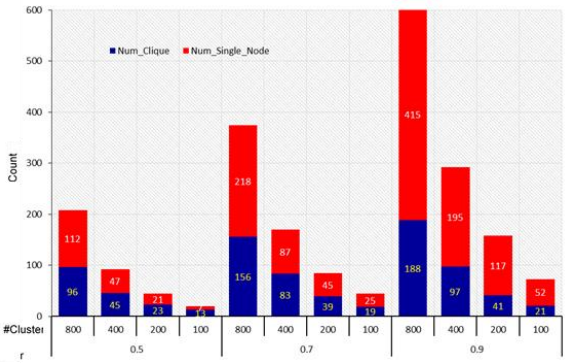


Figure 7.23. Change in number of clique of size >=2 and number of single nodes for different inter-cluster correlation coefficients (0.5, 0.7, 0.9) as threshold.

The effects of such changes are;

a. Number of computing node within cluster decreases leaving more uncorrelated nodes alone

b. More EN becomes candidate of CN

c. Overall Computation time will increase because of increasing the iteration to cover all clusters

d. A high ranking node, which is treated as CN, may not be used for any estimation

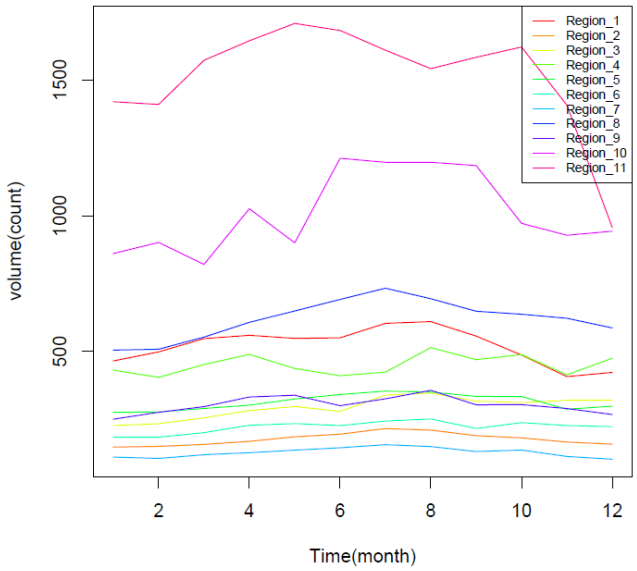


Figure 7.24. Average yearly traffic volume data for NY. Each colored line represents the volume of traffic data in one of the eleven regions over a year

The spatiotemporal data for NY traffic volume is demonstrated in Figure 7.24 where there are 11 regional data sets. The ranked data set for both PA APE and NY Traffic are provided to 20 cluster computers for performing the computation task for each application. Experiments were conducted on different number of clusters computers by starting with 2 and then increasing it to upto 20. In each case, the total required computing time, amount of data points that could be computed within given time constraint (1 hour), and the accuracy are observed along with other attributes as in the schema suggested by meta-analytics. The results for both data sets are compared below.

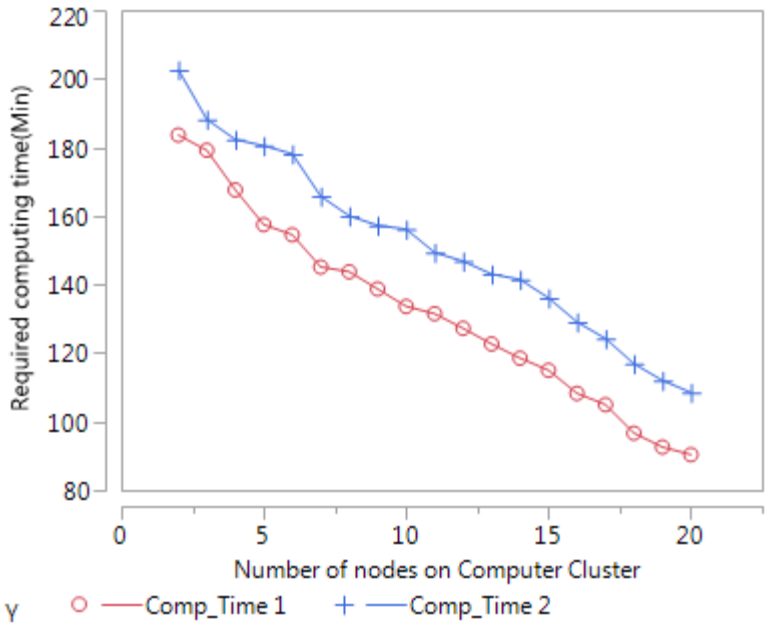
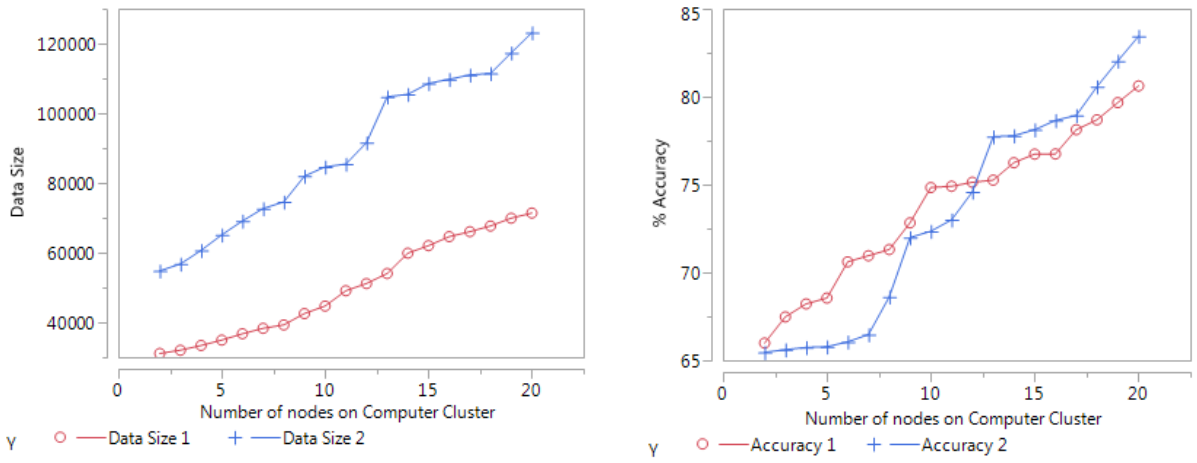


Figure 7.25. Required computing time for PA EPA data and NY traffic data

Figure 7.25 shows the required computation times for the entire data for both data sets. The blue line shows the computing time for PA APE data set while the red line shows the computing time for NY traffic data set. Clearly, for both data sets the required computing time is more than an hour whereas, the time constraint is one hour. Once the time constraint is imposed using the framework, the computation on each computing scenarios (number of cluster computers), the computing were completed within the time constraint (see Figure 7.26 a.). But in each scenario, a selected subset (hotspots) of the data for both data sets were used. Again, the blue line shows the optimal data sizes for PA APE data set and the red line shows the optimal data for NY traffic. The difference between the sizes of selected data for the same number of computers are due to the difference in computing tasks of the respective applications. The accuracy of the computation is

shown in Figure 7.26.b where, it is evident that the higher the number of computer used the larger the size of the data can be processed for computation resulting higher accuracy.



(a) (b)

Figure 7.26. Results of computing for PA EPA data and NY traffic data on distributed cluster environment. (a) Amount of data points that could be computed in one hour, (b) Accuracy of different scenarios

Both Figure 7.25 and Figure 7.26 are snapshot demonstration of one experimental scenario. To validate the robustness of the computation results, we conducted the experiment in iterations and used 10-fold cross validation method to further validate the results. The results of 10-fold cross validation in each case Required computing time for PA EPA data and NY traffic data, Amount of data points that could be computed in one hour, and Accuracy of different scenarios are shown in APPENDIX B (Figure B.7).

7.3 DISCUSSION

The foundation of our methodology is a graph based approach where the graph is constructed by spatial cluster as nodes and the inter-cluster correlation as edge weight. The spatiotemporal data in each cluster is rank ordered using our ranking method. The alternative is computing on the entire data, which may not be possible in cases with limited computing resources. One approach is randomly selecting data (as shown in Figure 7.27) points from entire data which may be missing important/high priority candidate data points to be updated. Another approach is to rank order the data based on attribute(s) of interest to determine the high priority candidate data points which may consistently select highly localized data (see Figure 7.28). We developed one such method (ranking by moving window (time) based variance) and took the top ranked data as suitable for available computing power to perform required computation.

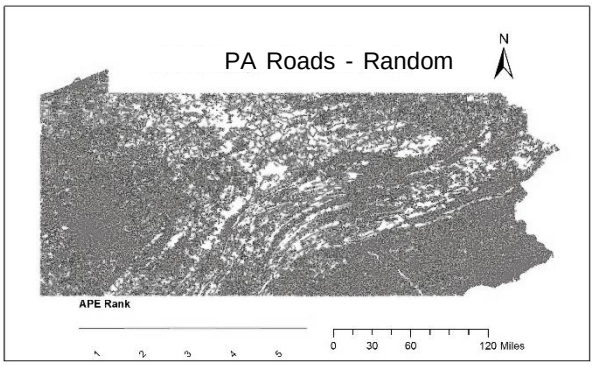


Figure 7.27. Selection without ranking. Selects data randomly regardless its significance of change

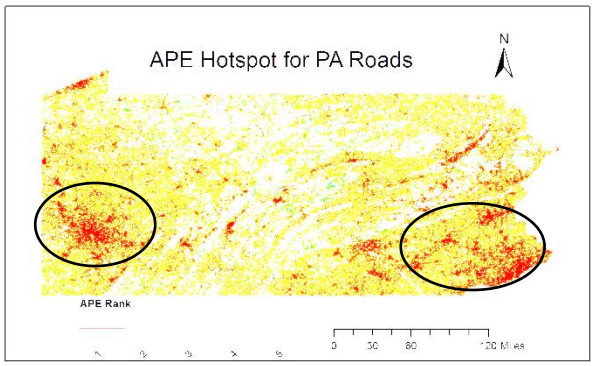


Figure 7.28. Selection of top ranked data. Selects only top ranked localized data

The limitation of this ranking only method is that only the data from certain high prevalent (in terms of change level) geographic area are picked for computation. For example; level of air pollution, as well as the rate of change, is different in different area. By using the ranking method to select the top ranked data points only, some high prevalent area will be represented leaving some low prevalent area not picked at all. Integrating spatial clustering and ranking together, the entire geographic area is represented and taken into computation, which is meant by "better" approach. A summary of the comparison among the three approaches are provided in Table 7.8. Using the ranking and clustering method, a representative data set is selected and provided to appropriate models in subsequent analytics.

Table 7.8. Summary of comparison among the three approaches: Random, Rank Only, and Ranking and Clustering

Random	Rank Only	Ranking and clustering
Unnecessary data point computation	Only high ranking data points would be computed	High priority data points are computed
Missing important/high priority candidate data points to be updated	May leave low ranking data points possibly never updated	Better coverage of area
No estimation, only computation, less overall data size coverage	May ignore even highly polluted area where the rate of change is low	Can apply estimation based on computed node.
Poor accuracy	Only computation, no estimation, less data coverage overall, poor accuracy	This covers larger data size and performs with better accuracy

Given the complex nature of spatiotemporal data (e.g., stationary or non-stationary, regular or random in both spatial and temporal domain), an appropriate model is needed to analyze and predict value for next (future) spatiotemporal instance and real time computation. To determine appropriate prediction model for a given spatiotemporal data set, the performance characteristics (AIC/SBC, R^2, –2LogLikelihood, MAPE, MAE) of each applicable model are analyzed for goodness of fit (APPENDIX A) for various parameters and one with the top performance is

recommended. For instance, in EPA data set, among the non-seasonal models, ARIMA (1,1,1) has the best fit given the pattern of the time series data. The claim is further confirmed by the test statistics—where AIC is 1315.64, SBC is 1326.07, R^2 is 0.97, -2LogLH 1309.64, MAPE is 9.09, and MAE is 2.83—for ARIMA(1,1,1). Also, the contribution of temporal parameters is demonstrated by the SARIMA(1,1,1)(1,1,1)12 predictive model and the result of the prediction demonstrates that the prediction line is significantly a better fit in SARIMA compared to the models with no temporal effect. To further confirm the suitability of SARIMA, the residual plot demonstrates that the residuals look random which means the model is stable. The model comparison is further demonstrated in the combined prediction outcome plot with 95% confidence interval along with both the ACF and PACF for residuals. The prediction line for SARIMA(1,1,1)(1,1,1)12 demonstrates the most appropriate forecasting pattern given the time series data over different locations. The plots for both residual ACF and residual PACF, bottom left and right, respectively, demonstrate that the SARIMA-based model tails off before all the other models (see Figure 7.17). This tailing off earlier is another important indicator to confirm that SARIMA-based model may be recommended for datasets with temporal dimension included.

We have actually two approaches of prediction: data value prediction and cluster prediction as explained in Chapter 5. These ARIMA-based models are all about data value prediction. For the cluster prediction approach, we used prediction for estimating nodes by regressing on the value of strongly correlated computing node using the correlation coefficient. The data value prediction approaches can be applied to any node (that has historical data available) without depending on the correlation of that node to any other nodes. It is common that there will be some nodes that are not correlated to any other node. To estimate for those uncorrelated nodes we need an appropriate

data value prediction method. So, for correlated nodes we use the cluster prediction approach, and for uncorrelated nodes, we use the ARIMA-based approach.

In our current approach we do not exclude any cluster while creating the graph for two reasons: 1) we do not need to create the graph each time a new data is available but just update the clusters and correlation matrix, and 2) any node may be weakly correlated to other node based on currently available data but the same node could be strongly correlated with another node after being updated with new data.

In the graph, we determine computing nodes (CN) as one partite and nodes strongly correlated with any of the computing nodes as estimating nodes (EN) which are in another partite in a bipartite graph. For the set of estimating nodes we use cluster prediction approach. But there will still be some nodes remaining in the graph that are not strongly correlated to any other node. To estimate these remaining set of nodes we use data value prediction approach. The data value approach is performed after the data is clustered. Since the characteristics (e.g., stationarity, randomness) of a given spatiotemporal data set is carried over each cluster, the prediction model appropriate for the dataset is also appropriate for the clusters out of it. So, the same data value approach is applied to each cluster which is not part of the nodes in the bipartite graph.

STARIMA would be another prediction model for data value prediction which particularly advantageous over ARIMA for spatiotemporal data (Duan *et al.*, 2016). However, estimating of parameters and coefficients in STARIMA model suffers overfitting resulting poor predictive performance since the prediction is over influenced even to minor variation in training data (Everitt & Skrondal, 2002). Also, using the same model based on the same correlation structure for prediction at different timestamps compromises accuracy as it is evident from Figure 7.4 that the variations in spatiotemporal data over different time is different.

Spatial autocorrelation plays an important role in our current graph approach. If the spatiotemporal data has spatial autocorrelation, it will violate the “independence” assumption of residuals. Accordingly the Error Sum of Squares (Error SS) is underestimated which influences the value of the test statistics resulting Type I error.

Accumulation of data over time pose challenges both on data processing and computing on time since the computation needs to capture the dynamic changes. Integrating spatial dynamics with this time varying data makes the challenges even harder. On top of that if the application requires real-time response, the necessity of special method and/or technique is inevitable. Where the existing approaches offer exclusive strategies and methods to address the computing challenges, the proposed methods offer inclusive approach. That means, the proposed methods focus on analytics, rather than managing by adding more computing power, which is independent of data size and computing resource limitations. Consequently, no matter how big is the data and how powerful is the computing resources, a solution is guaranteed. The scalability challenge of spatiotemporal Big Data is well addressed in this approach through the adaptation method as shown in the results (see Figure 7.26.a) which illustrates that the method successfully scaled the data size as appropriate for the computing resources. Also, the proposed approach will be adaptable with different computing platforms as it has demonstrated acceptable accuracies (see Figure 7.26.b) on different computing resources for different data.

8.0 CONCLUSION AND FUTURE RESEARCH

The adaptive framework for real-time spatiotemporal Big Data analytics is a layered structure of four major layers (components) each of which has a set of functionalities described in this work with associated algorithms. The goal was to provide a workflow for data-/compute-intensive problem to have a suitable solution through appropriate analytics. The framework provides necessary functionalities, like a library of functions and methods once the prototype framework is developed, of spatiotemporal Big Data analytics and real time computation as deem appropriate for spatiotemporal data intensive applications. This chapter provides the insight of a workflow that will automatically be generated once the framework is implemented, summarizes and concludes the work done in this dissertation along with its limitations, and suggests future research directions.

8.1 THE WORKFLOW

The framework is unique in that it offers a workflow that can provide the best possible (acceptable, near-optimal, and optimal) solutions in real time irrespective of data size and available computing resources. A workflow is a sequence of processes to accomplish a task from its initiation to completion. Since the change pattern of spatiotemporal data has uncertainty over time and location, framing the change patterns in a workflow is complex. However, this workflow could be considered as the foundation prototype and future research may complement it by offering more functionalities deemed needed for different types of spatiotemporal applications.

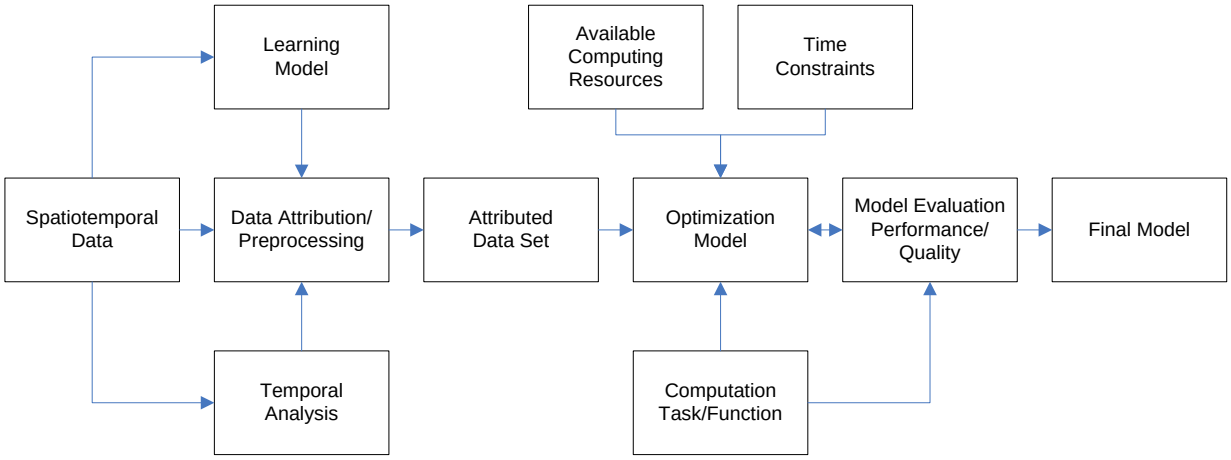


Figure 8.1 Workflow for spatiotemporal Big Data analytics

In this workflow (see Figure 8.1), a spatiotemporal Big Data analytics task is initiated by providing the spatiotemporal data of the application as input. The data is fed through three modules: Learning Model, Attribution/Preprocessing module, and temporal analysis module. If the data has sufficient temporal span, the learning model recognizes any temporal pattern in the data. For time series data spectral analysis and seasonal decomposition model and for spatial domain, inter-cluster correlation matrix in graph approach (as explained in Chapter 3) is used for the learning. In the Attribution/Preprocessing module, the spatiotemporal data goes through cleaning, attribution, and ranking module. The ranking is performed by temporal moving window-based variance method (as explained in Chapter 3). The temporal analysis module determines if the data has sufficient temporal details to be analyzed through predictive analytics; if not, it suggests for efficient analytics.

Once the data is attributed, that is rank ordered, clustered, prepared through graph approach (in case of spatial graphs over temporal span) an adaptation module takes the attributed data, available computing resources, time constraint, and the computation task/function of the

application. Considering and analyzing all the associated components (attributed data, available computing resources, time constraint, and the computation task/function) the adaptation module determines the most optimum data to take into computation and feed the data to computation module. Model outcome is evaluated using test statistics and goodness of fit test and feedback is provided on the model performance which is used for providing the final model for the application. The outcome of the workflow is finally the optimum computation results for current data (for efficient analytics EA), predicted results for historical data (predictive analytics PA) and inferences on the outcome of EA and PA through meta-analytics as illustrated in the framework layout (see Figure 8.2). The meta-analytics analyzes the outcome of EA and PA which are formalized through database schema as explained in Chapter 6.

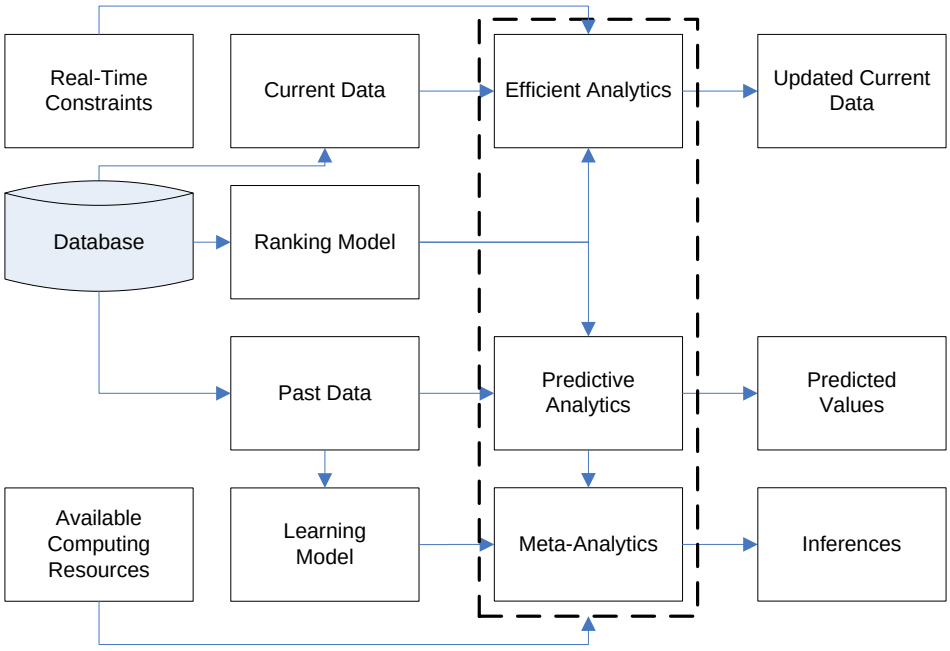


Figure 8.2 Layout of the outcome of spatiotemporal Big Data analytics framework

8.2 SUMMARY AND CONCLUSIONS

Spatiotemporal Big Data analytics is a challenging problem especially when the available computing resources are limited and the analytics and the computing outcome are expected within a time constraint. In this dissertation, we proposed a framework which basically is a platform providing spatiotemporal data-intensive analytics and computing solution for data-/compute-intensive applications that require computation under real-time constraints on given computing resources. The framework is a layered structure consisting of four interrelated components (layers); three on analytics and one on adaptation. A Graph based approach is developed as the foundation of the analytics components which are: efficient analytics – providing acceptable solutions based on current data in the absence of historical data; predictive analytics – providing near-optimal solutions by learning from the pattern of historical data and predicting based on the learning; and meta-analytics – providing optimal solutions by analyzing pattern of past data patterns. The adaptive computing component ensures that appropriate analytics is/are applied and computation is completed in real time on available computing resources.

The main contributions of efficient analytics components, as part of the framework, to compute the spatiotemporal "hotspots" and provide solution within time and computing resource constraints when sufficient historical data is either not available or the data shows none or random pattern over locations. The ranking method using a time window-based variance technique is an unbiased approach since each data point is equally likely to be ranked irrespective of its location. Assigning a uniform rank to each data point within a window variance may not be the precise rank for the data point compared to actual variation in that data point. However, adjusting the window size would address the precision issue.

The adaptation algorithms are experimented on different computing environments which may not be a perfect representation of all possible platforms but the performance summary demonstrates the rationality of the proposed method. There is always a trade-off between available resources and computing time performance. For lack of required computing resources, only the top ranking candidate from the input are considered for computation and update. Reducing computation time by filtering the input space may not provide high quality result overall but it will provide the most essential part computed in real time.

Given the historical time series data, analyzing the pattern of data change over time, ARIMA(1,1,1) model may predict the future values with satisfactory level of accuracy as evident from the experimental results. While ARIMA(1,1,1) outperforms other non-seasonal models (e.g., AR, MA, ARMA), SARMIA(1,1,1)(1,1,1)12 mode, which is basically a Seasonal ARIMA, fits the best for the given time series data. This means that the data has seasonal effect in it. However, this may not guarantee that the same model will be a good fit for every time series data. A different model may be suitable for a different dataset.

Different graph structures and graph properties are examined using matrix representation of graph in JMP to analyze the patterns of spatiotemporal data patterns to improve both efficient and predictive analytics. Different patterns of changing data patterns obtained from meta-analytics may improve the other two analytics. A *frequent subgraph*, if exists, in the graph pattern may suggest similar inter-relationships among the nodes. This feedback could further optimize the computation process by reducing computations for similar subgraphs.

Due to the complex nature of models, data structures, and algorithms for analyzing spatiotemporal data, providing computing solution under real-time constraints is a complex task. Once this complex solution is available, the methodology could be customized to be applicable to

data with other characteristics in other domains as well. This generalization will provide a platform of domain independent analytics to any Big Data application. The scalability of data-/compute-intensive problem, in the generalized framework, would be met by the adaptation module.

The main limitation of the framework would be a trade-off between accuracy of the computed results and available computing resources to accomplish the task in real time. The framework is unique in that the optimum solution is provided by adapting with available resources. For example, the best case scenario, the adaptation method takes the entire candidate input if available computing resources are capable to complete the computation in real time. This situation will ensure the best accuracy but, in Big Data domain, this situation usually is not the case. In the second case, which is very common in real-world situations, the top candidate input as appropriate for the available computing resources is taken into computation which ensures optimum accuracy possible. In the third case, any user required level of accuracy is ensured either by adjusting the input with available computing resources or by recommending minimum resource requirements.

Another limitation is that within time window variance a uniform rank is assigned to each data point, which may not be precise compared to actual variation in data point. But this is an optimum way the framework can process given its resource limitation and time constraint. Also, in predictive analytics, prediction is performed using regression based on inter-cluster correlation but given the type of application and data, regression approach may not be the best choice for all applications. In meta-analytics, clique analysis and frequent subgraph mining approach is used in this framework. Other graph phenomenon such as independent set analysis, isomorphic graph analysis, inter-node acquaintances analysis could also be explored.

Security and fault tolerance is another important aspect that is not addressed in this framework. Authentication or security perspective is not in the scope of this dissertation but worth

exploring. However, there are other tools and services available currently to complement the framework with security, user management, data/application management, fault tolerance and any other related services as necessary.

8.3 FUTURE RESEARCH DIRECTIONS

For any spatiotemporal data, observed patterns change over time and locations. To capture the change and accommodate the effect of this change in the optimization methods, it is challenging task to investigate the pattern of patterns observed over time and locations. Consequently, new machine learning technique may determine what ranking method will be appropriate for which situation. This may recommend new data mining method that will dig into patterns obtained by existing data mining methods and reveal a pattern of patterns. Once the ranking is accomplished based on the observed pattern of patterns the new technique may recommend the best optimization method for the given situation.

To that end, this domain specific (spatiotemporal) framework could be further extended to a generalized domain independent Expert System for Big Data Analytics in Real-time (ESBAR). ESBAR would be a complete platform comprising of Data Centric Computing (DAC) using high performance computing (HPC) resources, data itself, and the generalized frameworks for data analytics and computing solution. In DAC, the computation is taken to the data location rather transferring the data since the data is expected to be massive. Resorting to high-performance computing (HPC) is a common approach for handling computing challenge but to determine optimal solutions through data analysis, appropriate analytics using cutting-edge data mining and

machine learning is a must. Also, domain scientists and engineers across disciplines may not have the requisite knowledge and experience on and access to HPC platforms (e.g., clouds, grids, or supercomputers). Furthermore, many of the existing distributed computing techniques on clouds and grids are under development, lack robustness, not fully validated, not cost-effective, lack real-time response, and not available for widespread use yet. Therefore, ESBAR is aimed to analyze massive volumes of spatiotemporal data in real time and process them on available computing resources.

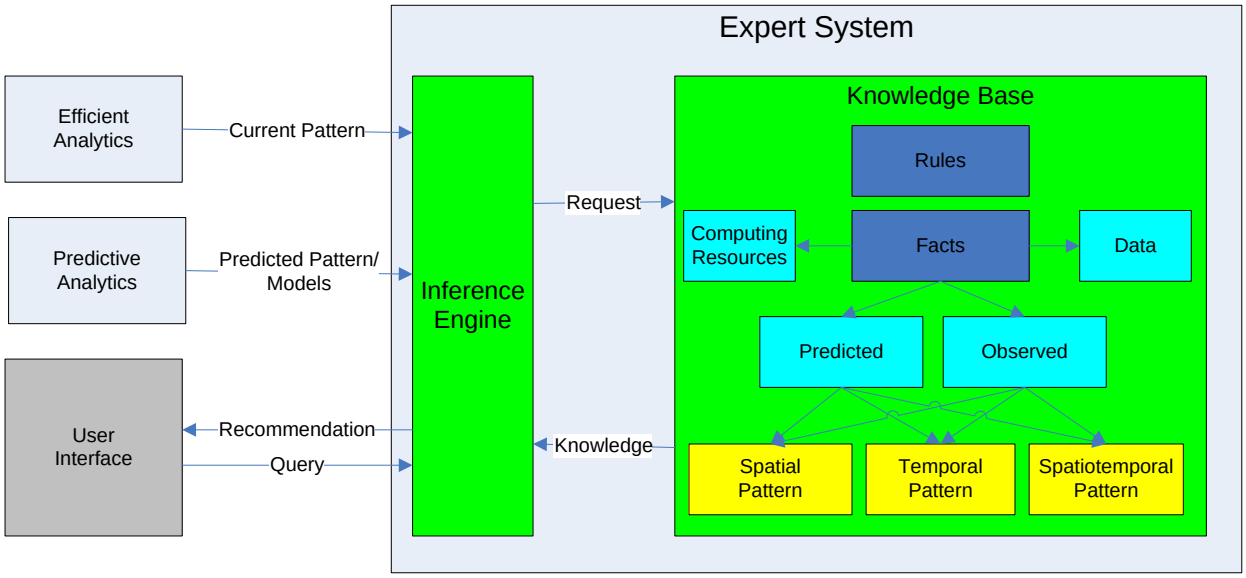


Figure 8.3 Expert System for Big Data Analytics in Real-time (ESBAR) concept diagram

The concept diagram of ESBAR is depicted in Figure 8.3. A rule-based system (see Figure 8.3) will adapt the tasks of the components in the framework on data analytics to complete the tasks within a time constraint on available computing resources. Once the system knows the outcome patterns and models for efficient and predictive analytics and user query, it analyzes the

request with the help of a knowledge base. In the knowledge base, a set of rules (if-then-else rules generated based on the meta-analytics) are exhausted based on thematic, spatial, temporal, spatiotemporal, and/or any other type of patterns (see Table 8.1) and will return recommendation to the user based on the knowledge as provided by the system.

Table 8.1 Rules structure for ESBAR

Domain	Input	Rules
System	Computing Resources	rule <rule id>:
Thematic	Attribute data	[<condition ID>: <condition>,
Temporal	Time Series	.....]
		==>
Spatial	Locations	[<action>, ...]
Spatiotemporal	Spatial and Temporal	

Once the ESBAR is completed, the methodology and its components can be developed as a software tool. The tool can be implemented on any computing platform (e.g., cloud, desktop, mobile) where it can provide analytics with real-time response for applications by realizing the underlying specifications of the platform and by adopting to the requirements of user applications. This dissertation built the foundation of the proposed framework. For a complete software framework that can be used by researchers interested in computing solutions for spatiotemporal data-intensive tasks given computing resources under real-time constraint, the developed components and algorithms need to be coded and packaged with appropriate user interfaces, databases, and APIs.

Depending on application requirements, the tool may be deployed differently. For example, the tool for the APE application could be implemented and deployed by taking one of the following "thick-thin" approaches:

- ✓ **As a service by providers (thin) -** In this case, all components are deployed and executed at the server side and the users can request and receive information on any client machines
- ✓ **As a distributed system** - The server can be made responsible for the PA, the resource optimization, and the MA and the client for the EA component
- ✓ **All on a client machine (thick) -** Depending on the specifications of the client machine, the solutions may be near-optimal or optimal.

APPENDIX A

STATITICAL TESTS AND PARAMETERS

A.1 DICKEY-FULLER TEST

A stationary time series data (y_t) usually has a tendency to return to a constant mean. In other words in a stationary time series data, larger current values (positive trend) are followed by smaller next values (negative trend), and vice versa. This trend is usually followed while predicting the future values in time series data. In this way, the level of the series will be a significant predictor of change in next time point and the next predicted value will have a coefficient of opposite sign. Assume an AR(1) model as given bellow:

$$y_t = \delta y_{t-1} + \epsilon_t$$

$$=> y_t - y_{t-1} = \delta y_{t-1} - y_{t-1} + \epsilon_t$$

$$=> \Delta y_t = (\delta - 1) y_{t-1} + \epsilon_t$$

$$=> \Delta y_t = \rho y_{t-1} + \epsilon_t$$

The null hypothesis is that there is a unit root, $\delta=0$, meaning y_t is non-stationary or a unit root is present if $|\delta|= 1$. Since the test is done over the residual term instead of raw data, it is not possible to use standard t-distribution for critical values and a special Dicky-Fuller table (Dickey and Fuller, 1979) is used for the statistic. Also, we can estimate the above model and test for the

significance of the ρ coefficient for the difference variable. If the null hypothesis is not rejected ($\rho^* = 0$), then Δy_t is non-stationary which recommends the difference variable to be used in the model.

For a larger and more complicated set of time series data an augmented version of the Dickey-Fuller test, called *Augmented Dickey-Fuller test*, is used. The augmented Dickey-Fuller test evaluates also the stationarity of time series data. The null hypothesis here is that $\rho^* = 0$. The model will be non-stationary if $\rho^* = 0$. The more negative it is, the stronger the rejection of the hypothesis. In the above model, a mean μ and additional lags of the dependent variable can be added as:

$$\Delta y_t = \mu + \rho^* y_{t-i} + \sum_{i=1}^{p-1} \theta_i \Delta y_{t-i} + \epsilon_t$$

To capture the time trend in a time series data, an improved version of Dickey-Fuller test, called *Dickey-Fuller test with a time trend*, is applied. The model with a time trend is expressed as:

$$\Delta y_t = \mu + \varphi t + \rho^* y_{t-i} + \sum_{i=1}^{p-1} \theta_i \Delta y_{t-i} + \epsilon_t$$

Here the test hypothesis is, $\varphi = 0$ and $\rho^* = 0$. The model will be non-stationary or will have a unit root present if $\rho^* = 0$.

A.2 TESTING FOR WHITE NOISE

To test for white noise (Hamilton, 1994) Box-Pierce statistic (Box & Pierce, 1970) is used which is expressed as:

$$Q = n\sum_{i=1}^{p}\rho_i^2$$

The Box-Pierce test statistic is a simplified version of the Ljung–Box statistic which tests if any of a group of autocorrelations of a time series data is different from zero. The Ljung-Box statistic is expressed as:

$$Q' = n(n+2)\sum_{i=1}^{p}\frac{\rho_p^2}{n-p}$$

where n is the sample size and ρ_p is the sample autocorrelation at lag p.

The null hypothesis that data are independently distributed which implies that the series is white noise. Here, Q has a χ^2distribution with p degrees of freedom.

A.3 GOODNESS OF FIT TEST

To test how the models fit for the time series data, a goodness of fit test is conducted where two measures for the goodness of fit are examined: *Akaike Information Criterion (AIC)* and Schwarz's Bayesian Criterion (SBC). SBC is also known as *Bayesian Information Criterion (BIC)*. Both AIC and SBC measure the trade-off between model fit and complexity of the model. AIC and SBC are expressed as:

$$AIC = -2\ln(L) + 2k$$

$$SBC = -2\ln(L) + \ln(N)k$$

where L is the likelihood function value evaluated with the parameter estimates, N is the number of data points, and k is the number of estimated parameters. The lower the AIC or SBC value, the better the model fits.

A.4 SPATIAL AUTOCORRELATION

Moran's I:

$$I = \frac{n}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - \bar{x})\,(x_j - \bar{x})}{\sum_{i=1}^{n}(x_j - \bar{x})^2}$$

Where, w_{ij} is the weight between observations I and observation j. I could be positive (positive autocorrelation) as well as negative (negative autocorrelation).

Geary's C:

$$C = \frac{n-1}{2\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\,(x_i - x_j)^2}{\sum_{i=1}^{n}(x_j - \bar{x})^2}$$

C=1 no autocorrelation

1>C>=0 positive autocorrelation

C>1 negative autocorrelation.

APPENDIX B

SUPPLEMENTARY RESULTS

B.1 CONFIDENCE INTERVAL FOR THE CORRELATION COEFFICIENTS

Table B.1.a. CI for Correlation coefficients- APE Data

CI of Correlation				
Variable	by Variable	Correlation	Lower 95%	Upper 95%
C31	C1	-0.1126	-0.1546	-0.0702
C39	C1	-0.3476	-0.3909	-0.3028
C39	C31	-0.2570	-0.3032	-0.2096
C91	C1	-0.2372	-0.2733	-0.2005
C91	C31	0.6301	0.6037	0.6552
C91	C39	-0.1409	-0.1896	-0.0914
C93	C1	-0.3720	-0.4346	-0.3059
C93	C31	0.4808	0.4213	0.5362
C93	C39	0.0199	-0.0548	0.0944
C93	C91	0.9207	0.9085	0.9313
C113	C1	0.2594	0.2102	0.3072
C113	C31	0.4494	0.4069	0.4900
C113	C39	-0.4437	-0.4845	-0.4010
C113	C91	0.4395	0.3966	0.4806
C113	C93	0.1900	0.1170	0.2609
C115	C1	0.7360	0.7001	0.7681
C115	C31	0.1910	0.1188	0.2612
C115	C39	-0.4296	-0.4880	-0.3674
C115	C91	0.1248	0.0514	0.1969
C115	C93	-0.0711	-0.1450	0.0036
C115	C113	0.5441	0.4899	0.5941

Table B.1.b. CI for Correlation coefficients-Traffic Data

CI of Correlation				
Variable	by Variable	Correlation	Lower 95%	Upper 95%
C2	C1	0.9459	0.9289	0.9589
C3	C1	0.9282	0.9059	0.9453
C3	C2	0.9324	0.9114	0.9485
C4	C1	0.8821	0.8467	0.9097
C4	C2	0.8984	0.8676	0.9224
C4	C3	0.8962	0.8647	0.9206
C5	C1	0.4102	0.2869	0.5202
C5	C2	0.4117	0.2885	0.5215
C5	C3	0.4182	0.2957	0.5272
C5	C4	0.3992	0.2747	0.5105
C6	C1	-0.7954	-0.8416	-0.7376
C6	C2	-0.8168	-0.8586	-0.7643
C6	C3	-0.8076	-0.8513	-0.7528
C6	C4	-0.7677	-0.8195	-0.7034
C6	C5	-0.2583	-0.3842	-0.1229
C7	C1	-0.8633	-0.8951	-0.8228
C7	C2	-0.8722	-0.9020	-0.8341
C7	C3	-0.8597	-0.8923	-0.8183
C7	C4	-0.8134	-0.8558	-0.7600
C7	C5	-0.2460	-0.3729	-0.1100
C7	C6	0.9226	0.8987	0.9410
C8	C1	-0.8647	-0.8961	-0.8245
C8	C2	-0.8714	-0.9014	-0.8331
C8	C3	-0.8521	-0.8863	-0.8086
C8	C4	-0.8043	-0.8487	-0.7487
C8	C5	-0.2325	-0.3606	-0.0959
C8	C6	0.9009	0.8707	0.9243
C8	C7	0.9701	0.9605	0.9773

B.2 SUMMARY STATISTICS FOR ORIGINAL AND DIFFERENCED VARIABLE

Table B.2. Summary statistics for original and differenced variable

Variable	Min	1st Q	Median	Mean	3rd Q	Max
Y	6.7	25.72	41.39	43.73	60.04	92.36
d.Y	-16.16	-4.12	-0.68	0.039	3.35	21.91

B.3 ACF AND PACF PLOTS FOR ORIGINAL AND DIFFERENCED VARIABLES

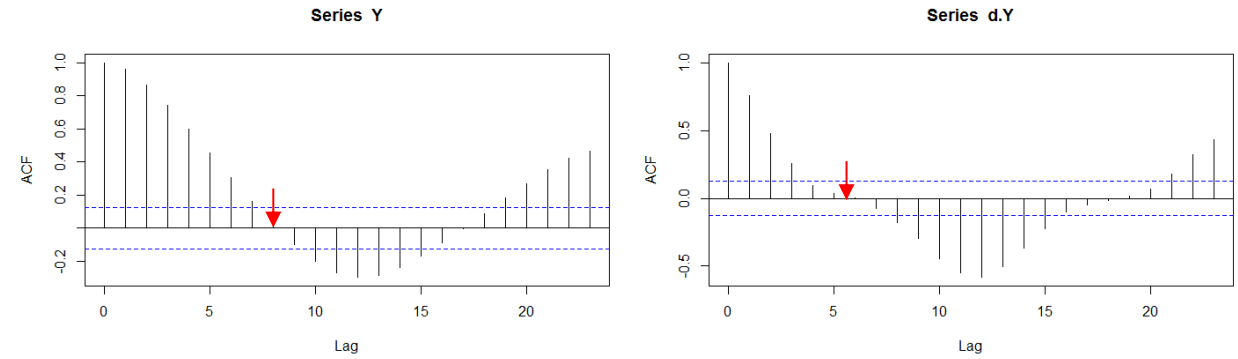


Figure B.3.a. ACF for original variable(Y) and the differenced variable(d.Y)

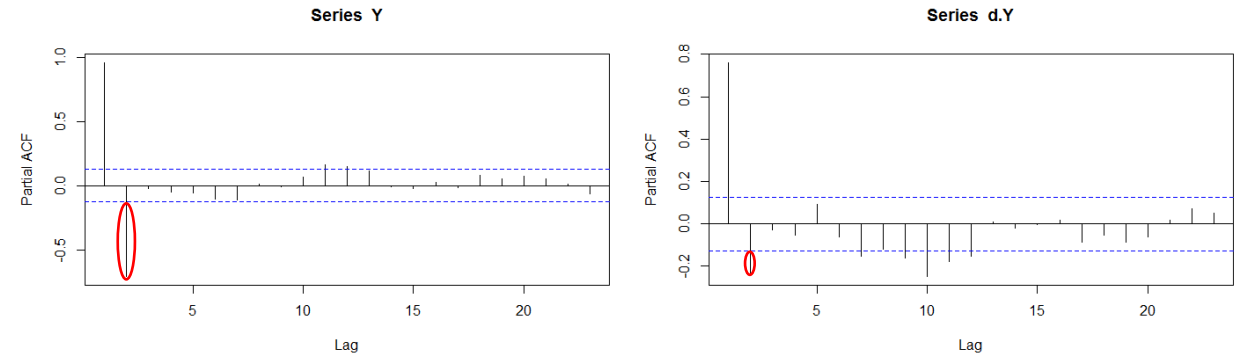
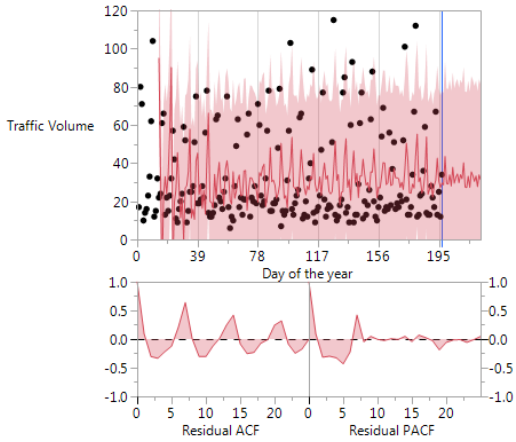
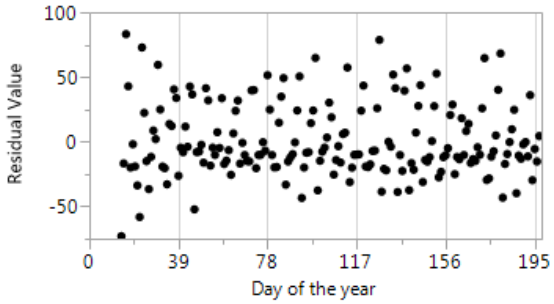


Figure B.3b. PACF for original variable(Y) and the differenced variable(d.Y)

B.4 NY ROAD TRAFFIC PREDICTION AT 1PM



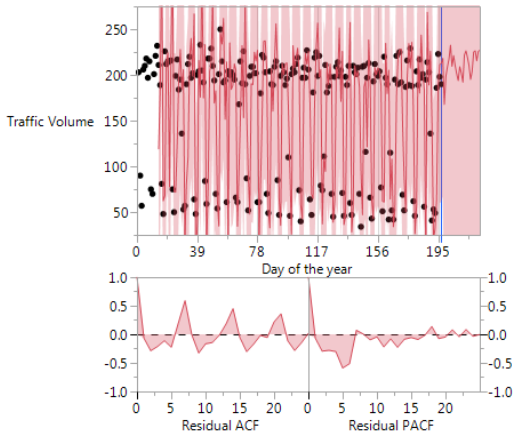
(a)



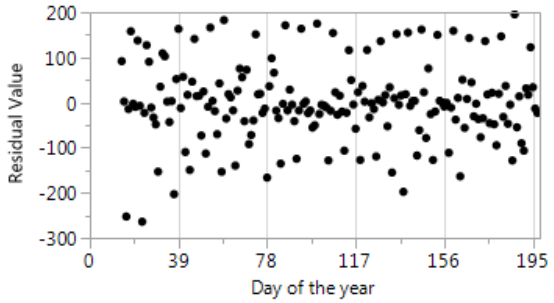
(b)

Figure B.4 NY road traffic prediction at 1 pm.
(a) Prediction with ACF and PACF, (b) Residual plot

B.5 NY ROAD TRAFFIC PREDICTION AT 5PM



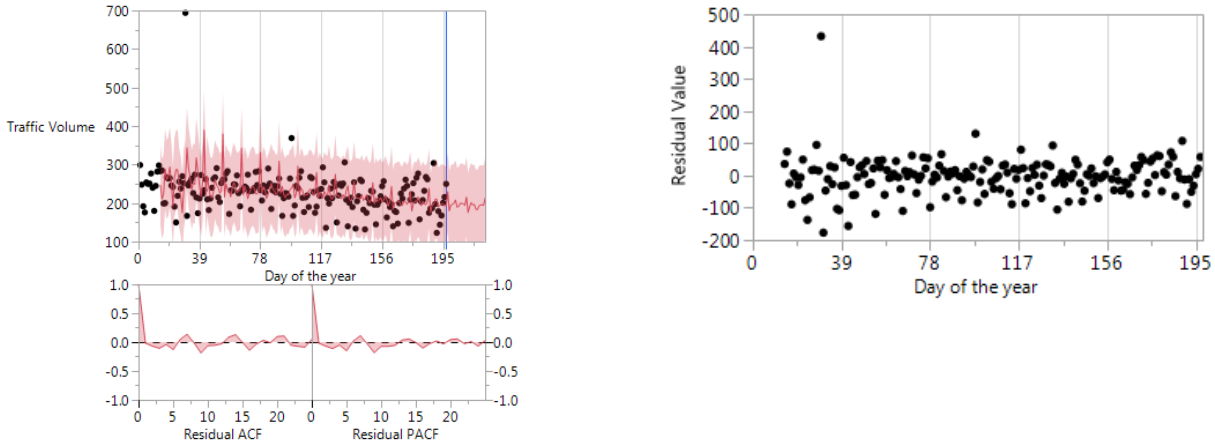
(a)



(b)

Figure B.5 NY road traffic prediction at 5 pm.
(a) Prediction with ACF and PACF, (b) Residual plot

B.6 NY ROAD TRAFFIC PREDICTION AT 9PM



(a) (b)

Figure B.6 NY road traffic prediction at 9 pm.
(a) Prediction with ACF and PACF, (b) Residual plot

B.7 COMPUTATION RESULTS OF 10-FOLD CROSS VALIDATION

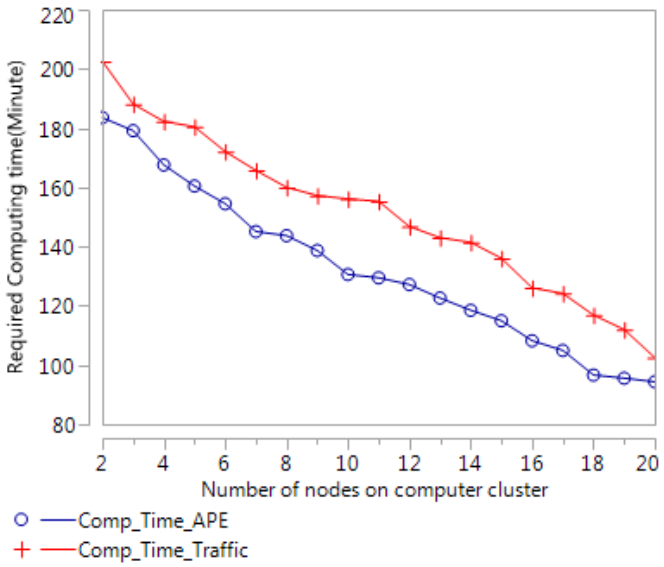
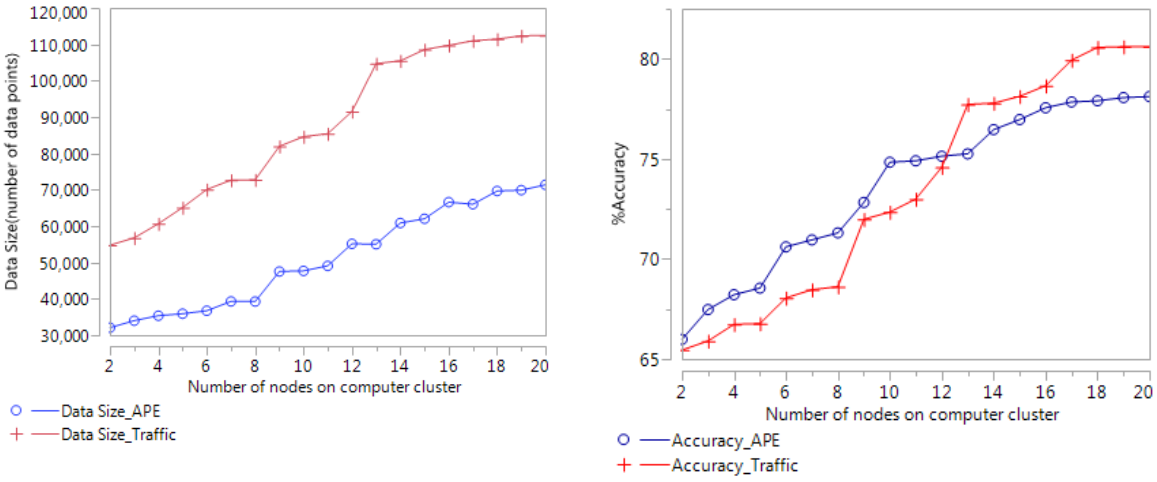


Figure B.7.1. Required computing time for PA EPA data and NY traffic data



(a) (b)

Figure B.7.2 Results of computing for PA EPA data and NY traffic data on distributed cluster environment. (a) Amount of data points that could be computed in one hour, (b) Accuracy of different scenarios

B.8 ANALYSIS OF NY TRAFFIC DATA

```
> data<-read.csv("CC_VOL_All_2015.csv")
> dim(data)

[1] 105370     43

> names(data)

 [1] "RC"          "Station"     "Region"      "DOTID"       "CCID"
 [6] "FC"          "Route"       "Roadname"    "County"      "County.FIPS"
[11] "Begin.Desc"  "End.Desc"    "Station.ID"  "ROAD"        "One.Way"
[16] "YEAR"        "MONTH"       "DAY"         "DOW"         "I1"
[21] "I2"          "I3"          "I4"          "I5"          "I6"
[26] "I7"          "I8"          "I9"          "I10"         "I11"
[31] "I12"         "I13"         "I14"         "I15"         "I16"
[36] "I17"         "I18"         "I19"         "I20"         "I21"
[41] "I22"         "I23"         "I24"

> table(data$RC)

   1    2    3    5    6    7   11   12   13   15   16   17   18   21   22   23
1180  590 3340 3258  730 5222 2584 2104 1532 2666  690 1422  704 2152  720  722
  24   25   26   32   33   34   35   36   41   42   43   44   46   47   51   52
 726  702 2640  720 4714 2304  362 1350 1212  702 3244 2648  706 1450 2186 2430
  53   54   61   62   63   64   66   71   72   73   74   75   81   82   83   84
3642 2528 1224 1906 1314 4116  728 2828 1304 3178 1414 2852 1398  700 3226 2062
  85   86   87   91   92   93   94   95   96   97
 636  718 2066 2420  724 1404 1074 1780 1692  724

> length(names(table(data$RC)))

[1] 58

> table(data$Station)

   1    2    4    5    6    8    9   10   12   13   14   15   16   17   18   19
1970 1416 2188  728 1364  674 1312 1452  574  726  580  522  700  394  706  726
  20   21   22   24   25   27   28   29   32   35   36   37   41   44   45   46
 260 1356  722  988  718  528  722 2066 1426  540  702  694 1346  676  674  728
  48   49   51   54   55   56   58   60   62   65   67   68   69   70   72   73
 678  154  730  720  728  706  720 1074  662  452  408  634  796  726 1020  720
  76   78   79   81   84   89   96   97   98  104  108  110  113  120  121  131
 600  512 2130  724  400  620  712  584  684  284  348  718  724  724  702  586
 137  139  144  145  151  156  158  164  166  174  180  182  193  199  204  221
 730  728  714 1208  730  694  626  712 1404  730  626  726  680  728  532  682
 222  225  227  236  241  247  257  270  273  279  283  291  295  312  313  315
2102  714  420 1452  714  728  450  588  590  690  576  728  582  722  724  580
 316  330  337  353  358  362  378  395  416  429  434  449  453  460  464  467
 684  498  694  708  636  718  718  704  714  362  404  658  694  502  656   48
 469  475  503  517  526  538  541  567  569  578  601  684  821  901  902  903
```

```
 712  552  716  528  726  724  578  416  518  718  730  670  700  730  702  730
 905  910  935  938  947 1001 1012 1021 1284 2580 8015 8100 8169
 686 1304  688  720  286  442  706  712  590  636  592  666  726

> length(names(table(data$Station)))

[1] 141

> table(data$Region)

    1     2     3     4     5     6     7     8     9    10    11
11702  7662  9450  9962 10786  9288 11576 10806  9818  8562  5758

> length(names(table(data$Region)))

[1] 11

> data$day.mean<-apply(data[,20:43],1,mean)
> #names(data)
> data$day.mean[1:20]

 [1] 323.1250 321.0833 284.2500 239.2917 160.5417 177.7500 289.0833 290.7917
 [9] 307.8333 311.8750 315.0000 316.8750 322.5000 310.3750 311.5833 297.7917
[17] 215.7917 231.1250 163.9583 187.0833

> table(data$YEAR)

  2015
105370

> data1<-data[,c(1:6,17,18,44)]
> head(data1)

  RC Station Region  DOTID CCID FC MONTH DAY day.mean
1 11      17      1 100428 1146 14     6   5 323.1250
2 11      17      1 100428 1146 14     6   5 321.0833
3 11      17      1 100428 1146 14     6   6 284.2500
4 11      17      1 100428 1146 14     6   6 239.2917
5 11      17      1 100428 1146 14     6   7 160.5417
6 11      17      1 100428 1146 14     6   7 177.7500

> names(data1)

[1] "RC"       "Station"  "Region"   "DOTID"    "CCID"     "FC"       "MONTH"
[8] "DAY"      "day.mean"

> table(data1$Region,data1$MONTH)

      1    2    3    4    5    6    7    8    9   10   11   12
  1   962  812  868  876 1006 1056 1082 1062  996  956  986 1040
  2   668  604  660  658  648  656  638  650  600  654  606  620
  3   846  730  834  818  748  744  832  780  818  828  732  740
```

```
  4   892  822  902  890  856  734  718  840  794  850  818  846
  5   906  712  846  912  950  852  958  972  862  952  964  900
  6   906  782  856  864  834  820  772  716  756  672  644  666
  7  1006  874  996  984 1004  976  970  906  944 1042  968  906
  8   962  876  942  944  986  870  832  864  800  918  840  972
  9   836  738  772  754  838  800  838  840  850  908  812  832
  10  770  720  746  776  752  748  678  744  724  670  604  630
  11  480  480  532  532  540  490  508  430  468  516  456  326

> data2 <- split( data1 , f = data1$Region )
> length(data2)

[1] 11

> names(data2)

 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11"

> data3<-sapply( data2 , function(x) {my.mean<-with(x, aggregate(list(day.mean), by = list(M
+                 my.mean <- do.call(data.frame, my.mean)
+                colnames(my.mean) <- c('MONTH', 'day.month.mean')
+                x<-merge(x, my.mean, by = 'MONTH') } )
> length(data3)

[1] 110

> names(data2)

 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11"

> my.data<-NULL
> MONTH<-seq(1,12,1)
> my.data<-data.frame(MONTH)
> for(i in 1:11){
+   r1<-as.data.frame(data2[[i]])
+   #r1<-data.frame(data2$i)
+   print(dim(r1))
+   print(head(r1))
+   my.summary <- with(r1, aggregate(list(day.mean), by = list(MONTH),
+                     FUN = function(x) { mon.mean = mean(x, na.rm = TRUE) } ))
+   my.summary <- do.call(data.frame, my.summary)
+   colnames(my.summary) <- c('MONTH', paste0("Region_",i))
+ print(my.summary)
+ my.data <-merge(my.data, my.summary, by = 'MONTH')
+ }

[1] 11702     9
  RC Station Region  DOTID CCID FC MONTH DAY day.mean
1 11      17      1 100428 1146 14     6   5 323.1250
2 11      17      1 100428 1146 14     6   5 321.0833
3 11      17      1 100428 1146 14     6   6 284.2500
```

```
4 11      17      1 100428 1146 14     6   6 239.2917
5 11      17      1 100428 1146 14     6   7 160.5417
6 11      17      1 100428 1146 14     6   7 177.7500
   MONTH Region_1
1      1 465.1333
2      2 498.4775
3      3 547.4593
4      4 559.6025
5      5 548.2617
6      6 550.3550
7      7 603.6312
8      8 609.9705
9      9 556.7030
10    10 486.6755
11    11 407.4471
12    12 423.0532
[1] 7662    9
      RC Station Region  DOTID CCID FC MONTH DAY day.mean
11703 21       4      2 100013 2180  7     1   1 29.58333
11704 21       4      2 100013 2180  7     1   1 31.25000
11705 21       4      2 100013 2180  7     1   2 43.29167
11706 21       4      2 100013 2180  7     1   2 41.29167
11707 21       4      2 100013 2180  7     1   3 34.50000
11708 21       4      2 100013 2180  7     1   3 34.62500
   MONTH Region_2
1      1 148.0352
2      2 150.7730
3      3 157.8534
4      4 168.9752
5      5 186.0749
6      6 195.6299
7      7 216.3179
8      8 210.3846
9      9 190.0850
10    10 182.0704
11    11 166.5215
12    12 158.8335
[1] 9450    9
      RC Station Region  DOTID CCID FC MONTH DAY  day.mean
19365 32      21      3 100032 3280 14     1   1  65.25000
19366 32      21      3 100032 3280 14     1   1  55.45833
19367 32      21      3 100032 3280 14     1   2 103.37500
19368 32      21      3 100032 3280 14     1   2  94.70833
19369 32      21      3 100032 3280 14     1   3  77.50000
19370 32      21      3 100032 3280 14     1   3  76.41667
   MONTH Region_3
1      1 227.2962
2      2 234.1240
3      3 255.6407
```

```
4       4 282.5841
5       5 297.4508
6       6 279.3386
7       7 339.4341
8       8 346.6358
9       9 317.1671
10     10 311.7921
11     11 320.2047
12     12 320.7340
[1] 9962    9
      RC Station Region  DOTID CCID FC MONTH DAY  day.mean
28815 41     131      4 100451 4181  4     1   1  89.79167
28816 41     131      4 100451 4181  4     1   1  97.62500
28817 41     131      4 100451 4181  4     1   2 184.12500
28818 41     131      4 100451 4181  4     1   2 182.12500
28819 41     131      4 100451 4181  4     1   3 139.75000
28820 41     131      4 100451 4181  4     1   3 139.41667
   MONTH Region_4
1      1 431.3374
2      2 404.9386
3      3 451.9754
4      4 489.2436
5      5 437.4305
6      6 410.7984
7      7 424.0850
8      8 514.4271
9      9 469.8263
10    10 488.5978
11    11 414.1062
12    12 475.1450
[1] 10786     9
      RC Station Region  DOTID CCID FC MONTH DAY day.mean
38777 51      44      5 100097 5181  4     1  15 183.7500
38778 51      44      5 100097 5181  4     1  15 184.8333
38779 51      44      5 100097 5181  4     1  16 190.2500
38780 51      44      5 100097 5181  4     1  16 196.8750
38781 51      44      5 100097 5181  4     1  17 156.7917
38782 51      44      5 100097 5181  4     1  17 165.1667
   MONTH Region_5
1      1 276.0986
2      2 277.1709
3      3 290.3334
4      4 302.3483
5      5 325.0883
6      6 341.5882
7      7 354.2120
8      8 350.5880
9      9 334.6206
10    10 334.2247
```

```
11    11 287.5837
12    12 298.5719
[1] 9288    9
      RC Station Region  DOTID CCID FC MONTH DAY day.mean
49563 61       4      6 100382 6180  6     1   1 41.25000
49564 61       4      6 100382 6180  6     1   1 41.87500
49565 61       4      6 100382 6180  6     1   2 87.25000
49566 61       4      6 100382 6180  6     1   2 90.54167
49567 61       4      6 100382 6180  6     1   3 55.50000
49568 61       4      6 100382 6180  6     1   3 55.33333
   MONTH Region_6
1      1 184.6581
2      2 184.5161
3      3 201.1998
4      4 228.3759
5      5 234.5136
6      6 227.1697
7      7 243.9394
8      8 251.3631
9      9 216.5557
10    10 238.1784
11    11 227.4083
12    12 223.4043
[1] 11576     9
      RC Station Region  DOTID CCID FC MONTH DAY day.mean
58851 71     199      7 100032 7183  4     1   1 45.37500
58852 71     199      7 100032 7183  4     1   1 45.75000
58853 71     199      7 100032 7183  4     1   2 74.95833
58854 71     199      7 100032 7183  4     1   2 78.33333
58855 71     199      7 100032 7183  4     1   3 55.58333
58856 71     199      7 100032 7183  4     1   3 61.41667
   MONTH Region_7
1      1 111.2080
2      2 106.4182
3      3 119.9077
4      4 127.3808
5      5 136.7371
6      6 145.6418
7      7 156.8992
8      8 150.0367
9      9 131.6935
10    10 137.1349
11    11 113.5495
12    12 103.4163
[1] 10806     9
      RC Station Region  DOTID CCID FC MONTH DAY  day.mean
70427 81     174      8 100147 8180  4     1   1  60.70833
70428 81     174      8 100147 8180  4     1   1  59.37500
70429 81     174      8 100147 8180  4     1   2  97.16667
```

```
70430 81     174      8 100147 8180  4     1   2 107.83333
70431 81     174      8 100147 8180  4     1   3  76.87500
70432 81     174      8 100147 8180  4     1   3  77.08333
   MONTH Region_8
1      1 505.0845
2      2 508.2793
3      3 553.0340
4      4 607.2473
5      5 649.0853
6      6 691.6132
7      7 732.9164
8      8 694.2273
9      9 648.1332
10    10 637.0896
11    11 622.2727
12    12 586.7258
[1] 9818    9
      RC Station Region  DOTID CCID FC MONTH DAY day.mean
81233 91      60      9 100397 9181 14     6   3 616.0417
81234 91      60      9 100397 9181 14     6   3 610.0833
81235 91      60      9 100397 9181 14     6   4 638.5833
81236 91      60      9 100397 9181 14     6   4 639.2500
81237 91      60      9 100397 9181 14     6   5 649.9583
81238 91      60      9 100397 9181 14     6   5 644.9583
   MONTH Region_9
1      1 250.7148
2      2 276.1849
3      3 296.8531
4      4 332.4602
5      5 339.0148
6      6 300.4695
7      7 325.7781
8      8 356.9125
9      9 303.6848
10    10 304.2459
11    11 289.9867
12    12 267.9767
[1] 8562    9
      RC Station Region  DOTID CCID FC MONTH DAY  day.mean
91051  3       6     10 100228  342 14     1   8 1072.4583
91052  3       6     10 100228  342 14     1   8  861.1250
91053  3       6     10 100228  342 14     1   9 1092.5000
91054  3       6     10 100228  342 14     1   9  891.0000
91055  3       6     10 100228  342 14     1  10  960.9583
91056  3       6     10 100228  342 14     1  10  721.5417
   MONTH Region_10
1      1  860.5626
2      2  901.3981
3      3  820.5099
```

```
4      4 1025.6019
5      5  900.0554
6      6 1211.0526
7      7 1196.4374
8      8 1196.5008
9      9 1184.3610
10    10  971.8061
11    11  927.7396
12    12  942.7658
[1] 5758    9
      RC Station Region  DOTID CCID FC MONTH DAY day.mean
99613  1       9     11 100011  182 14     1   1 301.8750
99614  1       9     11 100011  182 14     1   1 308.0000
99615  1       9     11 100011  182 14     1   2 471.5833
99616  1       9     11 100011  182 14     1   2 500.6667
99617  1       9     11 100011  182 14     1   3 558.4583
99618  1       9     11 100011  182 14     1   3 455.4583
   MONTH Region_11
1      1 1419.4990
2      2 1409.3753
3      3 1571.8570
4      4 1643.7032
5      5 1707.5813
6      6 1681.7113
7      7 1609.0906
8      8 1541.0659
9      9 1583.1083
10    10 1620.7628
11    11 1406.6552
12    12  956.3218

> my.data

   MONTH Region_1 Region_2 Region_3 Region_4 Region_5 Region_6 Region_7
1      1 465.1333 148.0352 227.2962 431.3374 276.0986 184.6581 111.2080
2      2 498.4775 150.7730 234.1240 404.9386 277.1709 184.5161 106.4182
3      3 547.4593 157.8534 255.6407 451.9754 290.3334 201.1998 119.9077
4      4 559.6025 168.9752 282.5841 489.2436 302.3483 228.3759 127.3808
5      5 548.2617 186.0749 297.4508 437.4305 325.0883 234.5136 136.7371
6      6 550.3550 195.6299 279.3386 410.7984 341.5882 227.1697 145.6418
7      7 603.6312 216.3179 339.4341 424.0850 354.2120 243.9394 156.8992
8      8 609.9705 210.3846 346.6358 514.4271 350.5880 251.3631 150.0367
9      9 556.7030 190.0850 317.1671 469.8263 334.6206 216.5557 131.6935
10    10 486.6755 182.0704 311.7921 488.5978 334.2247 238.1784 137.1349
11    11 407.4471 166.5215 320.2047 414.1062 287.5837 227.4083 113.5495
12    12 423.0532 158.8335 320.7340 475.1450 298.5719 223.4043 103.4163
   Region_8 Region_9 Region_10 Region_11
1  505.0845 250.7148  860.5626 1419.4990
2  508.2793 276.1849  901.3981 1409.3753
3  553.0340 296.8531  820.5099 1571.8570
```

```
4  607.2473 332.4602 1025.6019 1643.7032
5  649.0853 339.0148  900.0554 1707.5813
6  691.6132 300.4695 1211.0526 1681.7113
7  732.9164 325.7781 1196.4374 1609.0906
8  694.2273 356.9125 1196.5008 1541.0659
9  648.1332 303.6848 1184.3610 1583.1083
10 637.0896 304.2459  971.8061 1620.7628
11 622.2727 289.9867  927.7396 1406.6552
12 586.7258 267.9767  942.7658  956.3218
```

```
> my.data1<-my.data[,2:12]
> ts.plot(my.data1,gpars= list(col=rainbow(11)))
> legend("topright", colnames(my.data1), col=rainbow(11), lty=1, cex=.6
```

BIBLIOGRAPHY

Agrawal, D., Das, S., & Abbadi, A. E. (2011). Big data and cloud computing: current state and future opportunities. Paper presented at the Proceedings of the 14th International Conference on Extending Database Technology, Uppsala, Sweden.

Apache-Drill. (2016). Schema-free SQL Query Engine for Hadoop. Retrieved November 16, 2016, from https://drill.apache.org/.

Apple. (2014). Framework Programming Guide. Retrieved December 15, 2015, from https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/BPFrameworks/Concepts/WhatAreFrameworks.html.

ArcGIS. (2014). ArcGIS 10.3. Retrieved January 5, 2015, from https://www.esri.com/en-us/home.

Banerjee, S., & Fuentes, M. (2012). Bayesian modeling for large spatial datasets. Wiley Interdisciplinary Reviews: Computational Statistics, 4(1), 59-66. doi:10.1002/wics.187

Banerjee, S., Gelfand, A. E., Finley, A. O., & Sang, H. (2008). Gaussian predictive process models for large spatial data sets. J R Stat Soc Series B Stat Methodol, 70(4), 825-848. doi:10.1111/j.1467-9868.2008.00663.x

Bifet, A. (2013). Mining big data in real time. Informatica, 37(1).

Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: forecasting and control: John Wiley & Sons.

Box, G. E., & Pierce, D. A. (1970). Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. Journal of the American Statistical Association, 65(332), 1509-1526.

Campelo, C. E., & Bennett, B. (2013). Representing and reasoning about changing spatial extensions of geographic features. Paper presented at the International Conference on Spatial Information Theory.

Chainey, S., Tompson, L., & Uhlig, S. (2008). The utility of hotspot mapping for predicting spatial patterns of crime. Security Journal, 21(1-2), 4-28.

Cheng, H., Yan, X., & Han, J. (2014). Mining graph patterns. In Frequent Pattern Mining (pp. 307-338): Springer.

CIA. (2012). The World Fact Book. COUNTRY COMPARISON : ROADWAYS. Retrieved December 15, 2015, from https://www.cia.gov/library/publications/the-world-factbook/rankorder/2085rank.html.

Cressie, N., & Johannesson, G. (2008). Fixed rank kriging for very large spatial data sets. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 70(1), 209-226.

Cuzzocrea, A. (2014). Big data mining or turning data mining into predictive analytics from large-scale 3vs data: The future challenge for knowledge discovery. Paper presented at the International Conference on Model and Data Engineering.

Cuzzocrea, A., Song, I.-Y., & Davis, K. C. (2011). Analytics over large-scale multidimensional data: the big data revolution! Paper presented at the Proceedings of the ACM 14th international workshop on Data Warehousing and OLAP.

Di, S., & Wang, C.-L. (2013). Dynamic optimization of multiattribute resource allocation in self-organizing clouds. IEEE Transactions on parallel and distributed systems, 24(3), 464-478.

Dickey, D. A., & Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. Journal of the American Statistical Association, 74(366a), 427-431.

Duan, P., Mao, G., Zhang, C., & Wang, S. (2016). STARIMA-based traffic prediction with time-varying lags. Paper presented at the Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on.

Eberle, W., & Holder, L. (2013). Incremental Anomaly Detection in Graphs. Paper presented at the Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on.

Eldawy, A., & Mokbel, M. F. (2015). SpatialHadoop: A MapReduce framework for spatial data. Paper presented at the Data Engineering (ICDE), 2015 IEEE 31st International Conference on.

Engle, C., Lupher, A., Xin, R., Zaharia, M., Franklin, M. J., Shenker, S., & Stoica, I. (2012). Shark: fast data analysis using coarse-grained distributed memory. Paper presented at the Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data.

EPA. (2011a). AIRNow Gateway. Retrieved June 6, 2013, from http://airnowgateway.org/.

EPA. (2011b). EPA-AQS. Retrieved June 6, 2013, from http://www.epa.gov/ttn/airs/airsaqs/.

Eshel, G. (2011). Spatiotemporal data analysis: Princeton University Press.

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. Paper presented at the Kdd.

Everitt, B., & Skrondal, A. (2002). The Cambridge dictionary of statistics (Vol. 106): Cambridge University Press Cambridge.

Furht, B., & Escalante, A. (2011). Handbook of data intensive computing: Springer Science & Business Media.

Gelfand, A. E., Diggle, P., Guttorp, P., & Fuentes, M. (2010). Handbook of spatial statistics: CRC press.

Gutierrez-Garcia, J. O., & Sim, K. M. (2012). GA-based cloud resource estimation for agent-based execution of bag-of-tasks applications. Information Systems Frontiers, 14(4), 925-951.

Haas, T. C. (1995). Local prediction of a spatio-temporal process with an application to wet sulfate deposition. Journal of the American Statistical Association, 90(432), 1189-1199.

Hadoop. (2017). What Is Apache Hadoop? Retrieved April 20, 2017, from http://hadoop.apache.org/.

Hamilton, J. D. (1994). Time series analysis (Vol. 2): Princeton university press Princeton.

Hausenblas, M., & Nadeau, J. (2013). Apache drill: interactive ad-hoc analysis at scale. Big Data, 1(2), 100-104.

Ho, S., & Xie, M. (1998). The use of ARIMA models for reliability forecasting and analysis. Computers & industrial engineering, 35(1-2), 213-216.

IBM. (2017a). What is big data? Bringing big data to the enterprise. Retrieved June 10, 2017, from https://www-01.ibm.com/software/in/data/bigdata/.

IBM. (2017b). IBM Big Data & Analytics Hub. Retrieved June 10, 2017, http://www.ibmbigdatahub.com/gallery/quick-facts-and-stats-big-data.

JMP. (1989). Statistical discovery from SAS. Retrieved January 10, 2015, from https://www.jmp.com/en_us/home.html.

Karimi, H. A. (2014). Big Data: techniques and technologies in geoinformatics: CRC Press.

Lasinio, G. J., Mastrantonio, G., & Pollice, A. (2013). Discussing the "big n problem". Statistical Methods & Applications, 22(1), 97-112.

Lee, H., Shao, B., & Kang, U. (2015). Fast graph mining with HBase. Information Sciences, 315, 56-66.

Lee, M. H., Rahman, N. H. A., Latif, M. T., Nor, M. E., & Kamisan, N. A. B. (2012). Seasonal ARIMA for forecasting air pollution index: A case study. American Journal of Applied Sciences, 9(4), 570-578.

Li, Z., Chen, J., & Baltsavias, E. (2008). Advances in photogrammetry, remote sensing and spatial information sciences: 2008 ISPRS congress book (Vol. 7): CRC Press.

Maciejewski, R., Hafen, R., Rudolph, S., Larew, S. G., Mitchell, M. A., Cleveland, W. S., & Ebert, D. S. (2011). Forecasting hotspots—A predictive analytics approach. IEEE Transactions on Visualization and Computer Graphics, 17(4), 440-453.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Paper presented at the Proceedings of the fifth Berkeley symposium on mathematical statistics and probability.

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). Big data: The next frontier for innovation, competition, and productivity. 2011. McKinsey Global Institute.

Millennium-Project. (2017). Global Challenges for Humanity. Retrieved July 5, 2017, from http://www.millennium-project.org/millennium/challenges.html.

Murthy, M. N. (1967). Sampling theory and methods. Sampling theory and methods.

NAVTEQ. (2010). NAVTEQ Map. Retrieved June 15, 2015, from https://www.here.com/en/navteq.

NYSDOT. (2010). New York State Department of Transportation. Retrieved January 18, 2017, from https://www.dot.ny.gov/divisions/engineering/technical-services/highway-data-services/hdsb.

Palanisamy, B., Singh, A., Liu, L., & Langston, B. (2013). Cura: A cost-optimized model for mapreduce in a cloud. Paper presented at the Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on.

Repantis, T., & Kalogeraki, V. (2008). Hot-spot prediction and alleviation in distributed stream processing applications. Paper presented at the Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on.

RStudio. (2011). Open sorce and enterprise-ready professional software for R. Retrieved January 15, 2015,. from https://www.rstudio.com/.

Salcedo, R., Ferraz, M. A., Alves, C., & Martins, F. (1999). Time-series analysis of air pollution data. Atmospheric Environment, 33(15), 2361-2372.

Sharker, M. H., & Karimi, H. A. (2014). Computing least air pollution exposure routes. International Journal of Geographical Information Science, 28(2), 343-362.

Shekhar, S., & Chawla, S. (2003). Spatial databases: a tour (Vol. 2003): prentice hall Upper Saddle River, NJ.

Shekhar, S., Jiang, Z., Ali, R. Y., Eftelioglu, E., Tang, X., Gunturi, V., & Zhou, X. (2015). Spatiotemporal data mining: a computational perspective. ISPRS International Journal of Geo-Information, 4(4), 2306-2338.

Shi, S. M., Da Xu, L., & Liu, B. (1999). Improving the accuracy of nonlinear combined forecasting using neural networks. Expert Systems with Applications, 16(1), 49-54.

Skytland, N. (2012). What is NASA doing with Big Data today? Retrieved December 30, 2016, from https://open.nasa.gov/blog/what-is-nasa-doing-with-big-data-today/.

Spark. (2014). Apache Spark-Lightning-fast cluster computing. Retrieved December 30, 2016, from https://spark.apache.org/.

Storm. (2014). Distributed and fault-tolerant realtime computation. Retrieved March 3, 2015, from http://www.storm-project.net/.

Sun, Y., Li, B., & Genton, M. G. (2012). Geostatistics for large datasets. In Advances and challenges in space-time modelling of natural events (pp. 55-77): Springer.

Tobler, W. R. (1970). A computer movie simulating urban growth in the Detroit region. Economic geography, 46(sup1), 234-240.

Traverso, M. (2013). Presto: Interacting with petabytes of data at Facebook. Retrieved March 15, 2015, from https://www.facebook.com/notes/facebookengineering/presto-interacting-with-petabytes-of-data-atfacebook/10151786197628920.

Tseng, F.-M., Yu, H.-C., & Tzeng, G.-H. (2002). Combining neural network model with seasonal time series ARIMA model. Technological Forecasting and Social Change, 69(1), 71-87.

Walls, L., & Bendell, A. (1987). Time series methods in reliability. Reliability Engineering, 18(4), 239-265.

Wang, L., Zou, H., Su, J., Li, L., & Chaudhry, S. (2013). An ARIMA-ANN hybrid model for time series forecasting. Systems Research and Behavioral Science, 30(3), 244-259.

Worboys, M. F., & Duckham, M. (2004). GIS: a computing perspective: CRC press.

Yuan, M. (1996). Temporal GIS and spatio-temporal modeling. Paper presented at the Proceedings of Third International Conference Workshop on Integrating GIS and Environment Modeling, Santa Fe, NM.

Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. HotCloud, 10(10-10), 95.

Zyl, T. v. (2014). Machine Learning on Geospatial Big Data. In H. A. Karimi (Ed.), Big Data Techniques and Technologies in Geoinformatics. Boca Raton: Taylor & Francis.