Data Warehouse Approach to Historical Analysis of Vehicle Trajectory Data on a Road Network

by

Priyanka Pareek

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Personal devices such as smartphones or fitness bands enable users to record their movement information using GPS sensors, along with associated attribute data such as heart rate, speed, and elevation. Data collected using such devices include both, time-stamped recordings of object movement (called trajectories), and associated timeseries recorded for related attributes. Analysis of attribute information within the context of the path taken can be used to highlight important trends and similarities in movement of different objects. However, data volumes can make historical analysis of these data streams challenging without a proper pre-processing and data management strategy.

In this thesis, we design and implement a data warehouse structure to enable exploratory analysis of movement information for objects traveling in a constrained environment (e.g. a road network). The proposed data warehouse design can be used to analyze differences in vehicle attributes for trips taken on common road segments, or contrast trip-level information for categories of vehicles. Further, we design a spatial partitioning technique that takes advantage of the road network topology and use it to approximate average attribute values for spatial queries. We demonstrate the benefits of our data warehouse structure and analyze the results of our approximation method using a dataset of haul trucks operating in a surface mining environment.

Acknowledgements

I would like to thank my supervisors Dr. Joerg Sander and Dr. Ian Parsons for their constant encouragement and support during the course of my research.

My sincerest gratitude to Dr. Ronald Kube for his invaluable motivation, support and guidance. I would also like to thank my leader Dr. Jim Kresta for his encouragement and support, particularly during the crucial time of thesis writing.

I thank my colleagues and my friends for their constant encouragement and motivation. Your support has been invaluable.

Last but not the least, I thank my family for their sacrifices, unconditional support, encouragement and love. I dedicate this thesis to them.

Table of Contents

Chapter 1. Introduction 1
1.1 Truck and Shovel Operation 1
1.2 Existing Approaches for Analysis of Haul Road Condition 3
1.2.1 Identification of Co-Located Anomalies 3
1.2.2 Real-time Monitoring of Stress Events 4
1.3 Thesis Objectives and Contributions 5
1.4 Methodology and Results 6
1.5 Thesis Organization 7
Chapter 2. Background and Key Concepts 8
2.1 Introduction 8
2.2 Definition of Terms 8
2.2.1 Trajectory 8
2.2.2 Road Network 9
2.3 Map Matching: Determining vehicle position on road network 9
2.4 Data Warehousing 11
2.4.1 Types of summary measures 13
Chapter 3. Related Work 15
3.1 Previous Studies on Driving Efficiency 15
3.2 Trajectory Data Warehousing 17
3.3 Trajectory Data Pre-Processing for Route Determination 20
3.3.1 Algorithms for Offline Map Matching 22
3.4 Spatial Partitioning for Statistical Information Management 23
3.4.1 Graph Partitioning 25
3.5 Summary 26
Chapter 4. Methodology 28
4.1 Equipment Monitoring in a Surface Mining Operation 28
4.2 Data Collection 29
4.3 Generating Haul Truck Trajectories 30
4.3.1 Derivation of Position Recordings 30
4.3.2 Identification of Individual Trajectories 32
4.4 Road Network Digitization 32
4.4.1 Trajectory Decomposition for Extract-Transform-Load (ETL) 34
4.5 Road Network Data Warehouse Design 36
4.6 Road Network Hierarchy for Approximating Descriptive Statistics 38
4.6.1 Developing a Road Network Hierarchy 38
4.6.2 Creating a Graph Partitioned Hierarchical Structure (*GP-Tree*) 41
4.6.3 Spatial Searches using *GP-Tree* 41
4.6.4 Using GP-Tree to Approximate Summary Statistics 43
4.7 Summary 47
Chapter 5. Results and Discussion 48
5.1 Advantages of the Data Warehousing Approach 48
5.2 Analyzing Road Network Data Warehouse Design 49
5.2.1 Simplification of the Data Warehouse Model 49
5.2.2 Demonstration of Queries using the Data Warehouse 49

5.3 Performance Evaluation of *GP-Tree* 52
5.4 Evaluation of Approximation Results 55
5.5 Error Bounds for approximations Using *GP-Tree* 59
5.6 Summary 60
Chapter 6. Conclusion 62
Bibliography 63

List of Abbreviations

GHG	Green House Gases
GPS	Global Positioning System
MBR	Minimum Bounding Rectangle

List of Figures

Figure 1-1 Haul roads at an oil sands operation mine site. Roads are developed using different types of material based on the desired service length. Thus, rate of deterioration of road condition varies based on the road type. Image courtesy of Syncrude Canada Ltd. 2

Figure 1-2 Truck being loaded by a shovel in pit. Uneven and unstable ground surface conditions are visible in this image. Image courtesy of Syncrude Canada Ltd. 4

Figure 2-1 p1 , p2 , p3 are a sequence of GPS recordings captured from a vehicle when traveling near an intersection. Matching to nearest road segment can result in errors in selecting the right road segment, as can be with the selection of match candidate for point p2. 10

Figure 2-2 Example of star schema to store sales for a store (example modified from Han et al., 2006). Tables in blue are the dimension tables and the fact table, Sales, is shown in black. The schema represents a star burst, giving the schema its name. 11

Figure 2-3 Example snowflake schema for the electronics store shown in the figure above (Han, Kamber, & Pei, 2006). Unlike the non-normalized location dimension table shown in the star schema, the location table has been normalized to separate the city information into a separate table (City). 13

Figure 3-1 Conceptual model of a trajectory data warehouse to store aggregate movement data for objects (Leonardi et al., 2014; Nardini et al., 2018)..... 20

Figure 3-2 An example trajectory decomposition. Object position recordings (black) between t1 and t5 are first mapped to a point on the road network (red) represented by edges {e1, ..., e6}. Object travel route is then determined using interpolation. First the path travelled between two recordings is determined. Next, the time taken to travel to the end of the road segment (yellow) is estimated by assuming constant speed of travel between two consecutive recordings. 21

Figure 3-3 Hierarchical partitioning of 2-dimensional space to create rectangular cells of different resolutions. The first level consists of the entire region represented by a single cell. At each level, each dimension is bisected to create 4 cells per top-level cell. Each cell of the (i-1) th level corresponds to one cell of the ith level. 25

Figure 3-4 Example of 2-way graph partitioning that bisects the graph into two non-overlapping set of vertices (blue and red vertices). 26

Figure 4-1 A truck traveling on a road segment. The largest haul truck can have a length of 15 m and is installed with GPS receivers having an accuracy of 7.5 m. Depending on the location of the receiver (front or back of the truck), the center of the truck would be represented by a GPS measurement value within a 30 m distance. 34

Figure 4-2 The image on left is the original satellite image of the mine while the one on the right shows the digitized road network overlaid. Each road in the network was partitioned into smaller segments of 60m to create the base granule for the trajectory warehouse. 34

Figure 4-3 Example trajectory with large gaps in recordings due to zones with poor satellite reception. The first gap is 53 seconds, second gap is 203 seconds, third gap is 78 seconds, fourth gap is 256 seconds (left to right) with the linear distance travelled between the recordings ranging from 579m to 2.5km. Corresponding to the gap periods, 235 sensor readings for the fuel sensor were recorded..... 35

Figure 4-4 Star-schema based implementation of the data warehouse used for analysis of haul roads. A central fact table, RoadTravelFact, stores information on vehicle performance on various haul road segments. Associated parameters used for querying and filtering are stored in the dimension tables (RoadNetworkDim, EquipmentDim, OperatorDim, DateDim, TimeDim and OriginDestDim). 36

Figure 4-5 The image shows the conceptual representation of GP-Tree and its usage as a spatial lookup structure. At each level, the Minimum Bounding Rectangle (MBR) for the subgraph is tested for overlap with the query window. The leaf level nodes (shown in red) store the MBR for subgraphs and keys to individual road segment records within the subgraph. Geometries for all road segments at the leaf level must be individually tested for overlap with the query window..... 42

Figure 4-6 Table structure used to store histograms, minimum and maximum values for attributes. These histograms are used to calculate summary statistics for spatial queries. 45

Figure 5-1 Query to find the number of trips between the selected origin-destination pair along with the average truck load. Note that an accurate count of vehicles that travelled between an origin-destination pair can be calculated by storing trip identifier in each record of the fact table. 50

Figure 5-2 Query to understand variability in engine fuel rate consumed by vehicles traveling the same road section during different shifts. 51

Figure 5-3 Comparison of different in road attribute values for a selection of road segments with different classification (Intersection, Straight Road, Traffic Circle). The chart on the left compares the distribution of speed values for different road segment types. As can be seen, the speed values for road at an intersection or in a traffic circle are distributed around a lower value and has a narrower spread when compared with a segment of a straight road. The chart on the right compares the distribution of engine fuel rates for different road segment types. Here too variability in fuel rates is noticed where the fuel rates increase for roads where the trucks must slow down (e.g., intersection, traffic circle). 51

Figure 5-4 Query execution time (in millisecond) trended against increasing size of the spatial query window (% of total area), when calculating maximum attribute value for all segments in the query window. Results are plotted for 4 different methods: 1. GP-Tree as a spatial index (blue) 2. GP-Tree to approximate the maximum attribute value (orange) 3. Maximum value computation with no spatial index (green) 4. Default server spatial index (red). 54

Figure 5-5 Query execution time (in millisecond) trended against increasing size of the spatial query window (% of total area), when calculating minimum attribute value for all segments in the query window. Results are plotted for 4 different methods: 1. GP-Tree as a spatial index (blue) 2. GP-Tree to approximate the minimum attribute value (orange) 3. Minimum value computation with no spatial index (green) 4. Default server spatial index (red). 54

Figure 5-6 Query execution time (in millisecond) trended against increasing size of the spatial query window (% of total area), when calculating average attribute value for all segments in the query window. Results are plotted for 4 different methods: 1. GP-Tree as a spatial index (blue) 2. GP-Tree to approximate the average attribute value (orange) 3. Average value computation with no spatial index (green) 4. Default server spatial index (red). 55

Figure 5-7 Linear regression plot for Engine Fuel Rate (Actual vs Approximation). The graphs are plotted in increasing value of threshold used to filter regions within the query window of GP-Tree. The chart on the left considers values for all regions satisfying the query constraints when computing average for values within the search window. Observed R2 values in increasing order of threshold for overlap are 0.809, 0.714 and 0.647. 56

Figure 5-8 Linear regression plot for Speed (Actual vs Approximation). The graphs are plotted in increasing value of threshold used to filter regions within the query window of GP-Tree. The chart on the left considers values for all regions satisfying the query constraints when computing average for values within the search window. Observed R2 values in increasing order of threshold for overlap are 0.826, 0.484 and 0.422. 56

Figure 5-9 Linear regression plot for Engine RPM (Actual vs Approximation). The graphs are plotted in increasing value of threshold used to filter regions within the query window of GP-Tree. The chart on the left considers values for all regions satisfying the query constraints when computing average for values within the search window. Observed R2 values in increasing order of threshold for overlap are 0.690, 0.580 and 0.460. 57

Figure 5-10 Mean Absolute Error value as a function of increasing query window area. 58

Figure 5-11 Mean Absolute Error value as a function of increasing query window area. 58

Figure 5-12 Mean Absolute Error value as a function of increasing query window area. 58

Chapter 1. Introduction

Surface mining is a complex operation where transportation of the ore from source to destination consists of a large portion of the costs. For example, in the oil sands industry approximately 30% of the mining operation includes the cost of ore transport and delivery. For ore bodies where the ore is closer to the surface, truck and shovel fleets are used to mine and haul the ore to extraction facilities. This equipment is expensive to purchase, operate and maintain. Any incremental savings due to reduction in costs associated with the surface mining operation can lead to significant dollar value savings for these multibillion-dollar operations.

Equipment used in this operation is extremely large and expensive to maintain. Haul trucks used in the oil sands industry are some of the largest in the world with haul capacity ranging from 280 tons to 400 tons of material reducing the number of trips required to supply mined out ore for extraction. Unlike hard-rock mining operations (gold, coal), oil sands mining equipment operates in extremely harsh climatic conditions due to the location of the mines in Northern Alberta, Canada. Roads can become wet and soggy in spring months resulting in unstable ground conditions where haul trucks can get stuck. Routing within the mine can change rapidly, with addition and removal of roads to meet the surface mine plan. Hence most roads are temporary or semi-permanent and can become unstable over changing, extreme weather conditions.

Several sensors are available on the truck that enable operators to understand issues with truck performance and predict equipment failure conditions. Previously developed tools enable monitoring of overall haul cycle performance for trucks such as total material hauled, haul cycle time average, percentage of time of a haul cycle that was productive, or percentage of truck cycle time that was unproductive (waiting at shovels or dumps). However, a system that enables exploratory spatial analysis of truck attribute information when traveling on haul roads does not exist.

1.1 Truck and Shovel Operation

Truck and shovel operation for open-pit mining is most commonly used to excavate ore from a mine pit in the oil sands. In Canada, the mining equipment is subjected to some of the harshest operating conditions in the world. Weather, abrasive silica sand, highly viscous bitumen, and unstable ground conditions are some of the factors leading to extensive wear and tear of the equipment (“Evolution of Mining Equipment in the Oil Sands,” n.d.). Further, material used for construction of the roads varies

based on the expected lifetime of the roads. This variation in construction material has an impact on the road condition over time resulting in significant impact on equipment maintenance requirement, fuel consumption, Green House Gas (GHG) emissions, drive safety, and tire life ("Haul Road Maintenance: Improving Tire & Mining Truck Life," n.d.). Thus, maintenance of the haul roads and understanding bottlenecks in ore transportation due to degradation of road conditions are of critical importance.

Figure 1-1 Haul roads at an oil sands operation mine site. Roads are developed using different types of material based on the desired service length. Thus, rate of deterioration of road condition varies based on the road type. Image courtesy of Syncrude Canada Ltd.

A haul truck carries material being mined out by a shovel in the overburden or pit location. Generally operated by a single operator during a work shift (12-hour period), where the haul truck travels to a location where it needs to collect material (pit). The material collected (ore or waste) is then trucked to the bitumen processing plant to be crushed and processed, or to a dump site the material is used for building activities such as road construction. A central truck dispatch system dictates the location that the truck needs to travel to once a haul cycle is complete. Information on each haul cycle (distance traveled, material type and operator) are recorded in the dispatch system. Additionally, onboard systems collect information from on-truck sensors and transmit the information back to a central data management

system. The transmitted data includes GPS latitude, longitude and heading values; values from on-board sensors; and the corresponding time at which the measurements were recorded. Each value is stored as a separate time-stamped data stream.

1.2 Existing Approaches for Analysis of Haul Road Condition

Haul trucks in the oil sands mining operation are equipped with sensors that monitor truck vitals, such as, speed, engine fuel rate, engine RPM, and truck positioning using GPS. This data is recorded using on-board systems and transmitted back wirelessly to a central database. These individual timeseries of sensor recordings are archived at the frequency of 1 Hz for each truck. For a single truck sensor, that amounts to 86,400 recordings or over 4 million records per day for 50 trucks operating in a mine. Hence, although the number of truck travels per day is small, the volume per day of all sensor recordings is very high. Due to this large volume, the data needs to be pre-processed to help analyze variations in-road or fleet performance variation over time using readings from multiple sensors and/or trucks.

Existing approaches have focused on filtering records to identify events of abnormal behavior. Multiple sensors are combined and processed to identify anomalous events. The following sections explain the details of these tools.

1.2.1 Identification of Co-Located Anomalies

One of the earlier approaches taken to analyzing haul road conditions was the development of a decision support tool for forensic analysis of the maintenance equipment (Quach & Parsons, 2005). This was done by batch processing of equipment sensor time series and filtering specific events identified by the domain experts. These events could be related to driver behavior, road condition or equipment failure. Examples events include equipment rack events (lateral twisting of a truck frame due to uneven load on the diagonal tires) or steering pump problems. These events could be looked at in isolation for a single truck to determine equipment problems, or where relevant, could be verified for a collection of trucks to determine problems in the environment. For example, several co-located equipment rack events might indicate problems with haul road condition. Such condition-based monitoring of the equipment results in reduced maintenance costs since maintenance activities can be delayed until necessary. Further, potential equipment failures can be predicted early by taking steps to proactively mitigate these potential issues.

Figure 1-2 Truck being loaded by a shovel in pit. Uneven and unstable ground surface conditions are visible in this image. Image courtesy of Syncrude Canada Ltd.

1.2.2 Real-time Monitoring of Stress Events

Another tool developed to monitor truck telemetry data is one by Caterpillar. Caterpillar's Road Analysis Control (RAC) tool ("Cat | Road Analysis Control | Caterpillar," n.d.) integrates with the on-board sensors to provide real-time feedback on haul truck road conditions by monitoring events that could have detrimental impact on the truck operation. It actively monitors events such as the truck rack and pitch events to alert the operator of potential sections of the haul roads that should be avoided since they require maintenance. These events prompt the operator to modify their driving behavior to avoid the event, or if the event is related to the road condition, alerts operators to avoid the associated section of the road. Further, if multiple events are registered along the same section of the road by multiple trucks, it can help schedule maintenance equipment to repair issues with the section of the road.

The above-mentioned approaches are centered on an understanding of previously known equipment and driver behavior issues. Events are defined by domain experts and are encoded into detection algorithms.

Further, the spatial coordinates provide supplementary information that can be used for finding spatial clusters of similar events. However, ad-hoc analysis of the dataset has not been carried out on historical data. Given the volume of telemetry data, a data warehouse that stores aggregate information on the dataset to allow comparison across shifts/seasons/fleets would be valuable.

1.3 Thesis Objectives and Contributions

This work is motivated by an existing dataset collected by various sensors on haul trucks in a surface mining operation. Collected information includes truck location observations, and recordings of other on-board diagnostic values (e.g. speed, fuel usage, gear changes). Enabling ad-hoc exploration and analysis of the truck movement data, without pre-processing into a data warehouse is non-trivial since:

- Raw location observations first need to be pre-processed to determine vehicle path
- Ad-hoc queries on the dataset are challenging due to the volume of multi-dimensional data collected

Hence, extensive pre-processing is required to run spatio-temporal queries against the database. An approach which allows for interaction with movement data filtered on subsets of associated attribute values, thus needs to be developed.

The need to analyze movement data that contains associated attribute information is not limited to surface mining operations alone. Personal devices such as smartphones or fitness bands enable users to record their movement information using GPS sensors, along with associated attribute data such as heart rate, speed, and elevation. Components of such data include both, time series data collected for various attributes, and spatial information that varies over time. Visualization of the time series information within the context of spatial path could highlight important trends and similarities in objects. For example, monitoring path taken with speed, elevation and heart rate could help runners improve their racing strategies. In the field of spatiotemporal data mining, such problems have been explored in trajectory data mining, where a trajectory is defined as the ordered sequence of movement locations of an object (Atluri, Karpatne, & Kumar, 2017).

Previous studies have discussed data warehousing models for trajectories. However, the proposed models are not evaluated using attribute-rich data. Further, performance issues when querying aggregate attribute information for vehicles traveling in a constrained spatial environment have not been discussed.

In this thesis, a data warehouse is developed that enables ad-hoc queries against the multidimensional spatiotemporal data collected for haul trucks traveling on roads. Further, a generalized spatial partitioning approach for road networks is proposed to improve the performance of exploratory spatial queries for computing summary statistics against the data warehouse.

Some of the queries that can be answered by the proposed warehouse structure include:

- Retrieve summary statistics for fuel efficiency of all haul trucks driving the road sections.
- Find sections of haul roads where > *x* trucks traveled during a specific day.
- Analyze variations in distribution of haul truck attribute values when traveling empty vs full, or traveling during night shift vs day shift? Filter on different teams, vehicle types.
- Find road sections and/or time durations during which there is higher strain on the engine (for example, due to higher RPM, lower transmission and lower speeds).
- Identify road segment travel where fuel efficiency is poor.

In this thesis, we use a data warehouse approach for historical analysis of trajectory data on a road network. The data warehouse can be used by haul truck fleet managers to conduct exploratory analysis of truck movement history when driving on mine haul roads. Further, a key requirement of the data warehouse is fast query response times for spatial aggregation queries. The goal of our thesis is thus to provide faster query response times for aggregate spatial queries against the proposed data warehouse structure.

1.4 Methodology and Results

Spatiotemporal data mining considers objects with both, spatial and temporal information. Objects with their spatiotemporal information can also contain of recordings for associated attributes. Such scenarios are present in various application domains, such as, neuroimaging, climate science and transportation (Atluri et al., 2017). This domain of research can be categorized further based on the type of spatiotemporal data being collected. One category includes spatial measurements collected at a stationary location over time. Examples of this include information from induction loop sensors installed in roads or data collected over time at weather stations. Another category includes measurements that vary in space and time such as, path taken by a vehicle for a trip or animal migration tracks which changes in space and over time. Trajectory data warehousing studies the aggregation and management of movement data. In the past few years, due to the explosion of movement data being collected, research

in this area has grown. Data manipulation strategies from previous studies on trajectory data pre-processing, trajectory data warehousing, and graph theory are applied to the dataset used in this thesis.

Information on haul cycles is first collected to identify trajectories in the dataset. Haul cycle metadata includes information on trip duration, material being hauled, and locations travelled to and from. Pre-processing of the dataset is carried out to identify the truck route with information on duration of travel on each road segment. This information is used to associate truck sensor readings with sections of the mine roads. A data warehouse is then developed to store haul truck travel path and the associated attribute information for road sections along the path. Further, a generic spatial hierarchy for a road network is developed to improve performance of aggregate attribute queries for spatial regions within the haul road network.

1.5 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2: provides a background of some of the key concepts in trajectory data warehousing.
- Chapter 3: outlines previous approaches to trajectory data management and summarization; steps for pre-processing trajectory data; and the approach used to develop a generic spatial hierarchy for the road network.
- Chapter 4: describes our methodology. We discuss the data collection and manipulation process which is used to identify travel route for individual haul trajectories. This is followed by a discussion of the data warehouse schema developed for summarization and exploration of road-level, or trip-level attribute information. We then discuss the design of a generic spatial partitioning technique for road networks and use it to compute approximate statistic values for spatial queries.
- Chapter 5 demonstrates the data warehouse functionality by reviewing results of some typical queries using the warehouse structure. We the demonstrate the use of our spatial partition in computation of aggregate statistics for spatial queries.
- This thesis concludes with Chapter 6 where we summarize our contributions and provide directions for future work.

Chapter 2. Background and Key Concepts

The previous chapter discussed the need for analyzing vehicle movement data in the context of surface mining operation. This chapter will introduce key concepts needed to understand the problem domain. Sections 2.2.1 and 2.2.2 provide a formal definition of the terms, trajectory and road network. Section 2.3 explains the map matching process and its usage. Section 2.4 provides a discussion of data warehousing and its goals.

2.1 Introduction

Spatiotemporal data consists of data with both, spatial and temporal attributes. The field of trajectory data mining is focused on the study of spatiotemporal data collected from object movement recordings. Classes of problems commonly explored are trajectory pre-processing (W. Lee & Krumm, 2011), clustering (N. Ferreira, Klosowski, Scheidegger, & Silva, 2013), classification (Lee, Jae-gil;Han, jiawei; Li, Xiaolei; Cheng, Lee, Han, Li, & Cheng, 2011), outlier detection (J.-G. Lee, Han, & Li, 2008), and data warehousing (Jenhani & Akaichi, 2014). These studies are motivated by applications such as object route-determination, understanding animal migration patterns, identifying vessels on similar paths, identifying travel modes of people using position information logged from cellphones or identifying outlying or suspicious behaviour using positional logs. Other studies have been motivated by volume of movement data collected by cell phones, fitness bands, and vehicle fleet tracking systems where the goal is to find efficient methods for trajectory indexing and warehousing for historical movement analysis.

The rich background literature available for dealing with trajectory datasets is used to implement the data warehouse presented in this thesis. In the following section, frequently used terms in the area of trajectory data mining are explained before surveying related work in the area.

2.2 Definition of Terms

2.2.1 Trajectory

A trajectory is the path followed by an object in two-dimensional space. It is the sequence of location recordings of an object taken over consecutive timesteps. A trajectory for an object with identifier $\mathbf{id}$ can be represented as the list of tuples, $T = \left((\mathrm{id}, x_1, y_1, t_1), (\mathrm{id}, x_2, y_2, t_2), \dots, (\mathrm{id}, x_n, y_{n,} t_n)\right)$ where $(x_i, y_i) \in \mathbf{R^2}$ is the geographic coordinate recording in 2D space and t_i is the time step at which the recording was

captured with $t_1 < t_2 < t_3 \ldots < t_n$. This movement information could be recorded for object movement in an unconstrained environment, such as, shipping vessels in the sea or movement in a constrained environment, such as vehicles on a road network.

2.2.2 Road Network

A road network defines a constrained environment along which vehicles are likely to travel. It is typically represented using a graph to define the topology of the geographical region. The graph representation, $G = (V, E)$ consists of a series of edges $E = (e_1, e_2, \ldots, e_n)$ that are connected at intersections represented by vertices $(v_1, v_2, \ldots, v_m)$. At an intersection, the vehicle could travel from one road (i.e., edge) to another connected road.

Each edge is geometrically represented by a polyline which can contain two or more points, where each non-terminating point represents road inflections at which the vehicle would continue to travel on the same road, but its direction of travel could change. Additional metadata could be stored such as, speed limits, travel direction restrictions (one-way, two-way), type of road (highway, residential), name of the road for edges, or average stop time for vertices. Using a graph representation of the road network can help application of algorithms such as shortest-path between two vertices in the graph.

2.3 Map Matching: Determining vehicle position on road network

Vehicle travel in urban areas is typically constrained to the road network. Applications such as route determination and travel time estimation require matching users' position information to the road that they are traveling on. Map matching is the process of matching a series of timestamped vehicle GPS (latitude/longitude) recordings to the underlying road network to determine the most likely path taken by the vehicle. The process can be done online when routing an object to its destination in real time, or offline to understand the path taken by the object after a trip is complete. As discussed above, road networks are generally represented as a graph of inter-connected nodes and edges. Additionally, map matching maps the vehicle's position information to the road segments which are represented as edges in the graph network. Hence, the result of map matching represents a sequence of GPS recordings $p_1, p_2, \ldots, p_n$ to a sequence of road segments $p_{r1}, p_{r2}, \ldots, p_{rn}$ where p_{ri} is the location of the vehicle on the road segment r_i that the point is matched to.

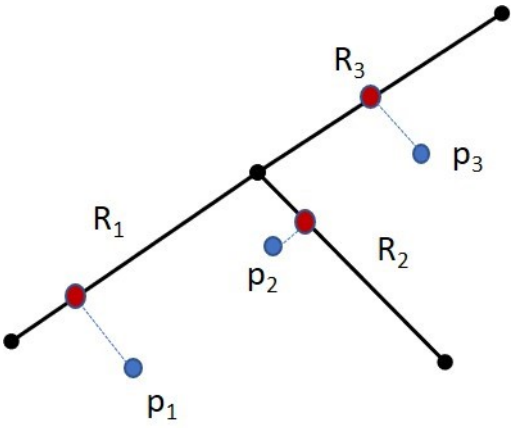

Figure 2-1 p_1 , p_2 , p_3 are a sequence of GPS recordings captured from a vehicle when traveling near an intersection. Matching to nearest road segment can result in errors in selecting the right road segment, as can be with the selection of match candidate for point p_2.

Map matching can be used to compresses the size of a trajectory dataset by representing a sequence of high frequency GPS recordings with a point position on the road taken by the vehicle. These position mappings can be used to determine the vehicle path taken and the individual points can then be discarded, significantly reducing the storage requirements. Further, similarity search is also simplified with a path representation of trajectories since only sequence of road segments, and not individual observation recordings, need to be compared. Hence, map matching is a commonly used pre-processing step for trajectory data with an underlying road network.

Several issues can complicate the map matching effort. Challenges include noisy point recordings which do not intersect with the geometrical representation of the road segment, large gaps in vehicle position records, or incomplete or incorrect digitization of the road network. Figure 2-1 is an example of sequence of noisy GPS recordings —p_1 , p_2 , p_3— and the underlying road network that the vehicle is represented by segments R_1 , R_2 , R_3. Determination of which road segment to map a point to is non-trivial since a naïve distance-based approach will erroneously map point p_2 to the road segment R_2. However, this error can be corrected by using an approach that considers the geometry of path taken. Sparse GPS recordings also pose challenges since linear interpolation between the location recordings will not match the geometry of the road network. Sophisticated algorithms exist that leverage the topology of the road network, direction of travel and vehicle speed to determine the likely path. Hence, location and feasibility of the path both need to be considered for correct determination of object path.

2.4 Data Warehousing

Data warehousing is the process of creating an independent repository for analytical purposes by coalescing data from various sources. It involves integrating heterogenous datasets collected from multiple sources, cleaning transactional records and consolidating the data for storage in a single schema. This process is repeated periodically to refresh the historical dataset. A data warehouse is primarily used to facilitate strategic decision-making and is an essential first step towards exploratory data analysis. The summarized historical data structure is kept separate from the OLTP (Online Transactional Processing) system to avoid impacting performance of the operational system. Further, information stored in data warehouses is generally pre-processed, summarized and re-structured to allow for efficient execution of complex queries, delivering high system query performance.

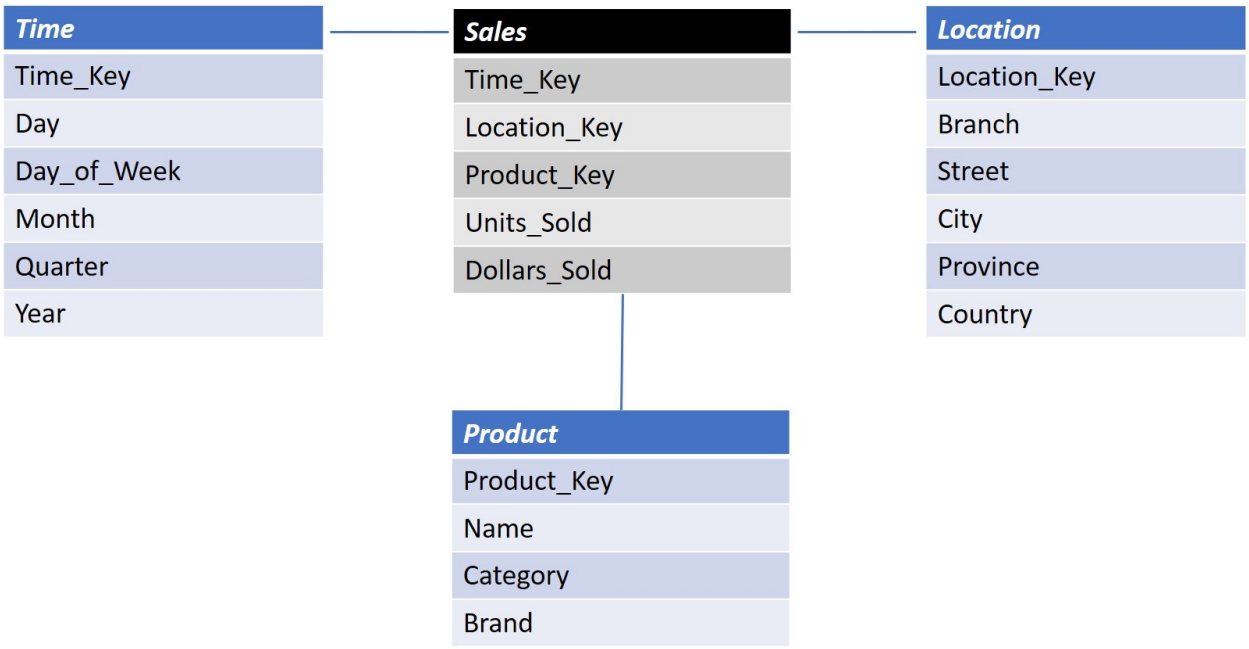

Figure 2-2 Example of star schema to store sales for a store (example modified from Han et al., 2006). Tables in blue are the dimension tables and the fact table, *Sales*, is shown in black. The schema represents a star burst, giving the schema its name.

Dimensional modeling of data warehouses is the preferred approach to data warehouse design (Kimball & Ross, 2002) and consists of two types of tables – *facts* and *dimensions*. The *fact* table stores aggregate measurements, called *measures,* for various attributes considered important based on the problem domain. *Measures* are the pieces of summarized information that can be pre-computed to form the fact table. The *fact* table also references *dimensions* tables which provide the contextual information for each

fact table record. The *dimension* values referenced by each record of the *fact* table provides the metadata for the records in the *fact* table. Additionally, dimension tables could contain de-normalized and hierarchical "look-up" information for values stored in the *fact* table. For each value in the fact table they represent the details for the record. For example, one of the measures in the sales *fact* table for a chain of retail stores could represent the total sales of the store. The contextual information could be provided by data in dimension tables referenced by the record: store location (location dimension), transaction date (time dimension), and type of objects sold (product dimension).

Questions that can be asked of the warehouse are based on summary measures in the dataset. One of the key decisions when designing data warehouses consists of selection of the base granularity (*granule*) being stored in the *fact* table. For example, do queries require comparison of hourly sales trends or would information be required only for daily sales. In the first case, the *granule* would require sales records aggregated hourly, whereas in the second case values aggregated daily would suffice.

Multiple dimensions associated with the summarized data allow exploration of the data using various complex queries and combinations of attributes (Han, Kamber, & Pei, 2006). Hierarchies may exist for each of the attributes, enabling aggregation of data at multiple levels of abstraction. The most common implementation for the multidimensional model are **star schemas** or a **snowflake schemas** (Han et al., 2006).

Figure 2-2 is an example **star schema** implementation created for analytical processing of sales data at an electronics store. The central *sales* table is the *fact* table which stores aggregate data for daily sales and references the supporting, *dimensions* tables (*Time*, *Product* and *Location*) describing attributes of the aggregated data. Attributes provide metadata regarding the measures in the *fact* table which can be used to filter the result subset, for example, only aggregate sales in a specific month. In the example the *dimension* table, *Time*, also specifies the hierarchy of values in the dimension: *day*→ *month*→ *quarter* → *year*. The schema representation resembles a star burst, giving the schema its name.

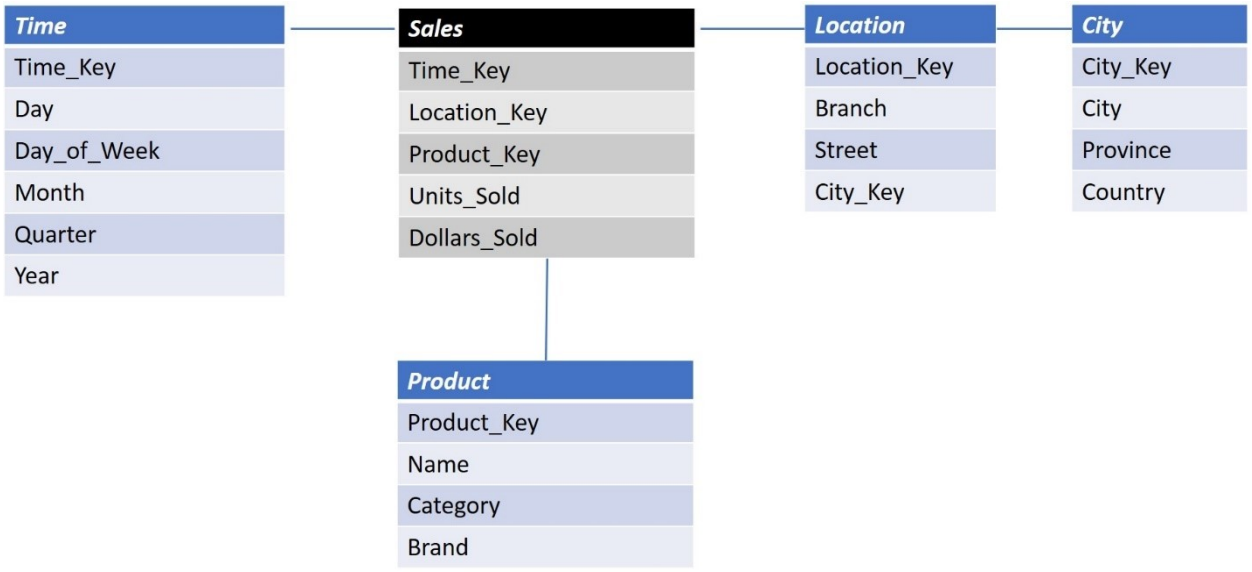

Figure 2-3 Example snowflake schema for the electronics store shown in the figure above (Han, Kamber, & Pei, 2006). Unlike the non-normalized location dimension table shown in the star schema, the location table has been normalized to separate the city information into a separate table (*City*).

A slight variation of the **star schema** is the **snowflake schema** where dimensions are normalized. An example of a store's sales data warehouse structure is shown as a **snowflake schema** in Figure 2-3. Unlike the **star schema**, in the **snowflake schema** the *Location* dimension table is normalized to improve storage due to reduction in duplication, however, the gain in storage is minimal when compared with the reduction in performance and increased complexity of query joins.

Online Analytical Processing (OLAP) tools can help analyze multi-dimensional warehouses from different angles using functions such as summarization, aggregation or consolidation (Han et al., 2006). Various OLAP tools exist that provide dynamic views of summarized data stored in warehouses. To enable high performance OLAP servers that can aggregate data using various groupings of dimensions, the data warehouse that the OLAP server uses must be highly efficient and optimized for cube computation. Hence, developing an optimal data warehouse and understanding how the data will be aggregated at various levels is essential for enhanced OLAP performance.

2.4.1 Types of summary measures

In data warehousing, *measures* can be categorized into three categories based on how they are computed. If the computation of a measure can be distributed to its parts, then the measure is distributive. In the case of distributive measures, measure for a higher-level cell at level *n* can be computed using the value

of the cells at level *n-1*. For example, using the hierarchy identified in Figure 2-2, if sales for each store is aggregated and stored, the sales for the city could be computed as the sum of the sales of all the stores in the city.

The second type of measure is *algebraic*. An algebraic measure can be computed using one or more *distributive* measures. For example, computing the average sales in a city will require using the two *measures*: sum and count of sales for all stores in the city.

On the other hand, *holistic* measures cannot be aggregated by using summaries at lower levels and require revisiting individual records for computation. For example, to compute the median at level n, the list of all records used to compute the median of cells at level $n-1$ need to be scanned. These measures are called *holistic* measures. Another example is the count of trajectories in a road network. During any given time, a moving object could travel between several road segments. If the spatial resolution of the *granule* in the trajectory warehouse is an individual road segment, the object trajectory would be accounted for in multiple road segments. However, if the aggregate number of trajectories is required for a subset of the road segments in the network, a simple sum of the records per trajectory will overestimate the total number of trajectories. The count of distinct trajectories for multiple connected road segments is thus not a sum of the count of trajectories on the individual road segment. Clever approximations for *holistic measures* can provide significant improvements in query time executions for holistic measures by eliminating the need to store references to the original dataset for each cell of the data warehouse structure.

Chapter 3. Related Work

The previous chapter introduced foundational concepts in trajectory data mining. This chapter outlines prior work related to the problem of vehicle performance monitoring on a road network. First, we discuss motivational studies that have analyzed vehicle data with multiple attribute information to understand vehicle or operator performance. Then conceptual designs of data warehouses used for trajectory data management are reviewed. This is followed by a discussion on data manipulation strategy adopted for pre-processing of raw location records, and determination of trajectory route determination. The chapter ends with a discussion of the graph-partitioning problem; an approach that has been used for creating a generic spatial hierarchy for the road network.

3.1 Previous Studies on Driving Efficiency

Eco-driving is driving in a manner that reduces fuel consumption. Following eco-driving advice leads to an increase in mileage and reduces carbon dioxide emissions. Studies to understand eco-driving parameters are typically carried out in a simulated environment where driving parameters (speed, RPM, acceleration) are analyzed to understand impact on vehicle fuel efficiency. This helps with consistency when conducting performance tests. However, the results use simulation data and do not consider travel on roads sections, where performance efficiency could vary based on the environment.

However, recent eco-driving studies have studied datasets where information is collected when driving on roads in order to understand variations in operator driving differences on fuel consumption (J. C. Ferreira, de Almeida, & da Silva, 2015; Jakobsen, Mouritsen, & Torp, 2013). In their study, Jakobsen et al. work with a rich dataset of vehicle parameters collected using a vehicle communication interface, CAN bus. The data is collected at a high frequency of 1 Hz for four similar vehicles driving city roads for 9 months. GPS data is used to map-match position observations and associate the corresponding vehicle measurements. This information is used to analyze differences in operator fuel efficiency when driving on the same road sections. However, only a small number of vehicles are analyzed as a part of the study and the authors identify the need to extend study to more vehicles. A data management strategy to store and retrieve information on vehicle travel over road segments would be required to study a larger group of vehicles. Further work is required to extend the study to include data from multiple vehicles and analyze variations in performance over a longer duration of time.

Table 3-1 Vehicle parameters collected in study by Ferreira et al. (2015). Over 20 parameters were collected and aggregated individual bus travel trip records.

Aggregate Trip level Parameters	Other External Information (collected hourly)
Inertial Movement (%)	Weather
Acceleration (#Events)	Visibility (km)
Breaking (#Events)	Humidity
Excessive Acceleration (#Events)	Cloud Coverage
Excessive Breaking (#Events)	Wind Direction
Clutch (#Events)	Weather events (rain, storm, heavy rain)
RPMS Idle Rotation (% of Trip Time)	
RPM Green (% of Trip Time)	
RPM Yellow (% of Trip Time)	
RPM Red (% of Trip Time)	
Average Fuel Consumption	
Distance	
Average Speed	

Another study by Ferreira et al. (2015) analyzed eco-driving behavior on a larger dataset and included more vehicle drive parameters. The collected dataset included vehicle monitoring information for a period of over 2 years of travel for 745 public transportation buses. Unlike the approach by Jakobsen et al., the collected data did not consist of raw readings of vehicle sensors. Instead pre-configured events were recorded such as period when car was in acceleration, time ignition was turned on/off, breaking, or clutch use. These readings were sampled at a rate of 2 Hz, aggregated for each vehicle trip, and then archived in a data warehouse for analysis and determination of factors that impact fuel consumption (liters/100 km). Driver performance was then categorized into groups based on trip fuel consumption, and training was provided to those driving with high vehicle fuel usage. A list of the trip parameters is shown in Table 3-1. Over 20 numerical parameters were recorded for each vehicle trip. However, analysis was performed

using only trip-level information. The approach did not consider the underlying road network due to which impact of road conditions could not be evaluated.

Similar to the studies described above, the haul truck dataset analyzed in this thesis is rich in attribute information archived at a frequency of 1 Hz. Large volume of raw data and the need to run specialized queries against the vehicle telemetry dataset highlight the need for pre-processing and warehousing the data for exploratory analysis.

The field of trajectory data mining is the study of object movement in space and over time. Literature in trajectory data warehousing is thus reviewed to understand data warehousing strategies, as discussed in Section 3.2. Further, the need for analysis of road-level information requires determination of vehicle travel route from vehicle position records. This data manipulation step is described in Section 3.3. In Section 3.4 we then discuss a graph-partitioning approach used to determine a hierarchical spatial partition for the road network. This spatial partition is used to enable fast computation of summary statistics for region queries by using approximations as a trade-off for query performance improvement.

3.2 Trajectory Data Warehousing

Initial trajectory data management approaches focused on limitations of database management systems (DBMS) in modeling trajectory or moving object data types. Several enhancements to the database engine were developed to incorporate trajectory representation as a native type of DBMSs by extending the data definition and query language to include moving object data types (Güting, Behr, & Düntgen, 2010; Pelekis, Frentzos, Giatrakos, & Theodoridis, 2015). However, due to the large volume associated with moving object data, research focus shifted to efficient warehousing of trajectory data for identification of trends. Proposed models focused on data warehouse design with the intent of preserving trajectory semantics or trajectory geometry.

Moreno and Arango proposed a conceptual model for warehousing trajectory information using multidimensional modeling (Moreno & Arango, 2010). The model design intended to preserve trajectory semantics by enriching the raw location records with information on object behavior or activity. This was done by partitioning each trajectory into distinct temporal sets called *episodes*. Each *episode* recorded information on a specific activity that had associated measurements. For example, daily movement of a person could be described using a sequence of *episodes*, such as, *drive to work→ park at work→ drive to home* as shown in Table 3-2. Each *activity* has a set of measurements specific to the activity, as listed in

the **Measures** column. Hence, each trajectory was decomposed into a set of tuples -- <*location, timestamp, activity, [list of measure(s)*]>. Due to the variable list of *measures* associated with each *episode,* the authors proposed a design with multiple *fact* tables, one for warehousing information on each type of *episode*. However, the proposed design was inefficient and not scalable since addition of *episodes* would result in a proliferation of *fact* tables. Further, re-creation of the sequence of *episodes* from a trajectory would require assimilation of records from multiple *fact* tables.

Table 3-2 Example semantic representation of a person's movement trajectory described as a sequence of *episodes* as defined by Moreno et al. (Moreno & Arango, 2010). The table lists the sequence of timestamped location recordings, each associated with user's activity such as driving or parked at work.

Time Stamp	Location	Activity	Measures
$\mathbf{t_1}$	$<x_1, y_1>$	Drive	<*Speed*, *Acceleration*, *Fuel Consumed*>
$\mathbf{t_2}$	$<x_2, y_2>$	Drive	<*Speed*, *Acceleration*, *Fuel Consumed*>
$\mathbf{t_3}$	$<x_3, y_3>$	Parked (at Work)	<*Stop Duration*>
$\mathbf{t_4}$	$<x_4, y_4>$	Drive	<*Speed*, *Acceleration*, *Fuel Consumed*>
$\mathbf{t_5}$	$<x_5, y_5>$	Drive	<*Speed*, *Acceleration*, *Fuel Consumed*>
$\mathbf{t_6}$	$<x_6, y_6>$	Parked (at Home)	<*Speed*, *Acceleration*, *Fuel Consumed*>

Comparable to the above-described semantic model for trajectory, authors Campora et al. propose a semantic model based on periods of *stops* and *moves* (Campora, Fernandes De Macedo, & Spinsanti, 2011). A trajectory data warehouse design is proposed that uses such a representation where a trajectory is segmented into movement primitives (*stops* or *moves*). Associated spatial and temporal information is stored in separate *dimension* tables, and a *fact* table stores the required *measures* (average speed, duration). A third dimension is designed independent of the data warehouse structure and lists *events* processed using records in the *fact* table. For example, records in the *fact* table that are in spatial proximity during lunch hours could indicate popular eateries and could be used to suggest restaurants to tourists. Building the *events* dimension after a trajectory is segmented and loaded in the data warehouse structure allows for flexibility in using the pre-processed information for different applications. For example, the

same structure could be used by a traffic supervision system to identify spots of congestions during sports events and could suggest alternative traffic routing for future events.

Semantic based approaches provide an understanding of movement information. In large, anonymized vehicle movement datasets information on vehicle activity is typically not available and must be determined using data mining. However, an inherent semantic representation for trajectories that travel in constrained environments is the route taken by the object. Road networks contain metadata which can be correlated with trajectory travel path to provide information on drive characteristics. Road network metadata can include route taken to reach a point of interest, expected duration of travel based on speed limits, and duration of stops at traffic lights. Hence, representing raw observations as path taken on a road network can provide implicit knowledge about the trajectory.

A series of publications (Leonardi et al., 2014; Nardini et al., 2018) outlined the conceptual framework for data warehousing of moving objects in order to analyze aggregate movement patterns for objects traveling in a spatial region (such as, vehicle traffic in the city, animal migration trajectories). The model developed could accommodate object movement in free space or in a constrained environment. The data warehouse was designed using a traditional star-schema approach and then used to store summaries for large trajectory volumes within a spatial region. The warehouse design was motivated by the need to store spatiotemporal aggregate information for a group of objects due to the high volume of data or where the information identifying the object needed to be obfuscated due to privacy concerns.

Figure 3-1 is the conceptual design proposed by the authors for warehousing trajectory information (Leonardi et al., 2014; Nardini et al., 2018). Each *granule* of the fact table stores aggregate information for a spatial region and temporal interval. The hierarchy for spatial and temporal intervals is specified in the corresponding *Spatial Hierarchy* and *Temporal Hierarchy dimension* tables, with each *dimension* potentially describing multiple hierarchies on top of the base *dimension* value. Each spatiotemporal *granule* (represented by a grid cell if the spatial hierarchy is grid-based) stores aggregate information for multiple trajectories with corresponding *measures* in the *Intra-Granule Fact* table. These include total number of trajectories visiting the spatial region during the time interval, combined distance traveled by all trajectories, combined travel time for all trajectories, and count of trajectories starting or ending within each *granule*. An additional dimension, *Object_Hierarchy*, stores information on object profiles. This warehouse design was then used to review aggregate trends in the spatio-temporal domain.

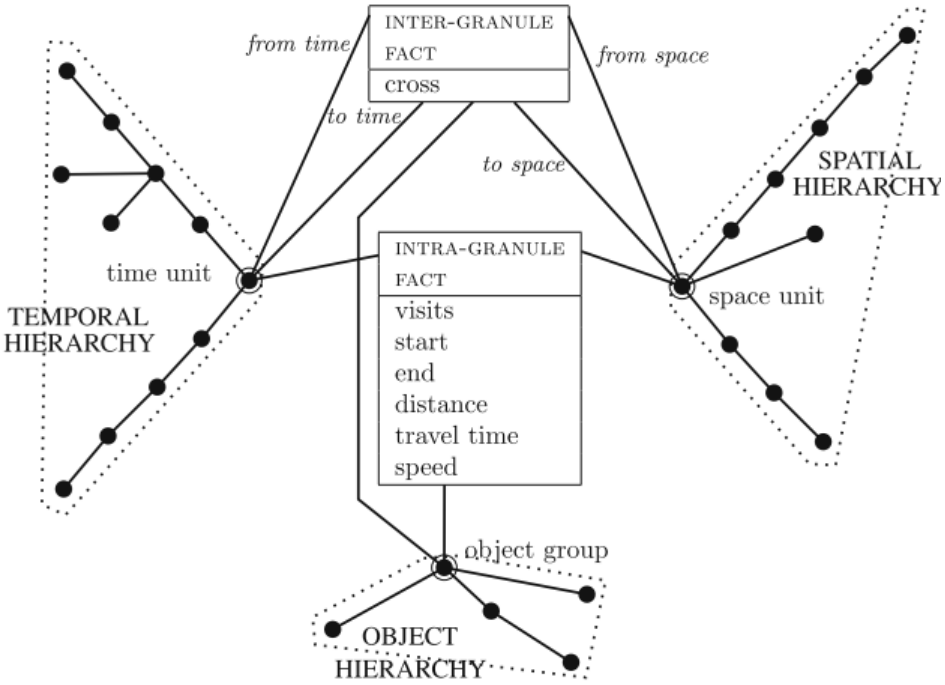

Figure 3-1 Conceptual model of a trajectory data warehouse to store aggregate movement data for objects (Leonardi et al., 2014; Nardini et al., 2018).

The model defined had an additional layer of complexity due to the need to store aggregate object information for OLAP operations, such as, summation of object counts over a spatial region. Due to the nature of spatiotemporal applications, objects could move between spatial boundaries during a time interval. The same moving object could thus be accounted for multiple times when aggregating object count over a large spatial region. The aggregated count of objects in a region or during a time interval could thus be grossly overestimated. To account for transitions between spatial regions, an *Inter-Granule Fact* table was maintained that accounted for object transitions over spatial boundaries (moving from one spatial region to another) or temporal boundaries (staying at the same location over a longer duration). This information was used when approximating the presence of objects over spatial or temporal regions.

3.3 Trajectory Data Pre-Processing for Route Determination

A road network edge (road segment) is the spatial *granule* considered for the data warehouse in this work. Each trajectory must first be partitioned into a set of sub trajectories that correspond to the sequence of edges in the road network. This process is referred to as *trajectory decomposition* (Leonardi et al., 2014).

Time travelled on each road can then be used to aggregate the sensor recordings measured during the period.

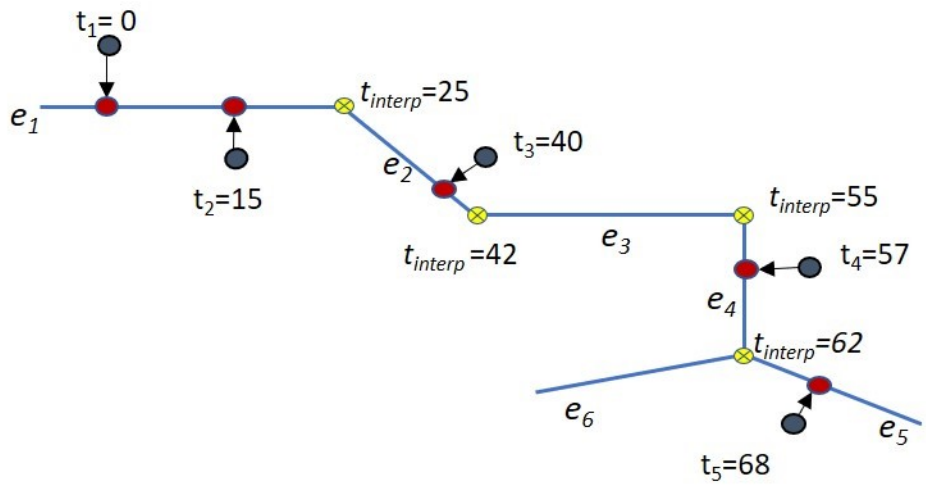

Figure 3-2 An example trajectory decomposition. Object position recordings (black) between t1 and t5 are first mapped to a point on the road network (red) represented by edges {e1, ..., e6}. Object travel route is then determined using interpolation. First the path travelled between two recordings is determined. Next, the time taken to travel to the end of the road segment (yellow) is estimated by assuming constant speed of travel between two consecutive recordings.

Figure 3-2 is an example of *trajectory decomposition* for a trajectory represented by a sequence of position recordings (black circles) taken between timestamps t_1 *to* t_5. First, map matching is used to map the observations to the underlying road network (red circles). These mapped GPS observations can then be used to determine the route between two points on the network. The start and end time for travel along each road segment in the route is then estimated assuming constant speed between two consecutive GPS recordings (yellow circles). Hence, discrete location recordings can be converted into sequence of travels on road segments where each trajectory is represented as a sequence of sub-trajectories where each sub-trajectory is represented as <*Spatial Granule Id, Travel Start Time, Duration*>. The observation recordings are thus converted to a sequence of travels along the spatial granule used in the data warehouse. Based on these recordings, the associated sensor recordings for each trajectory can then be aggregated for each of these road segments.

Determining an associated point on the road network is a non-trivial problem and has been heavily researched. Several studies exist discussing algorithms for matching vehicle positions to the underlying road network. An overview of approaches for solving the offline map algorithms is discussed below.

3.3.1 Algorithms for Offline Map Matching

Offline map-matching algorithms are applied for post-processing location recordings for historical vehicle trips. Since the location points for an entire trip are available, a lookahead strategy can be applied to determine the path taken by the trajectory increasing the accuracy of the map matching algorithm. Algorithms in map matching can be classified as geometric, topological or probabilistic based on the approach taken. GPS measurement noise, gaps in point recordings, and errors or gaps in digitization of the underlying network can lead to ambiguity in vehicle path determination. Strength of map-matching algorithm varies based on their ability to resolve the path with existing gaps in observations.

Geometric map matching is the most basic approach and considers only the geometry of the network for map matching (White, Bernstein, & Kornhauser, 2000). Examples of techniques used in geometric matching, that is, point-to-point, point-to-curve, curve-to-curve geometric matching. In point-to-point matching, an observation is matched to the closest point in the network. This method is extremely sensitive to the method used for digitization of the network roads where each road segment is a polyline with several end points. Point-to-curve matching maps an observation to the nearest linear segment in the network by computing distance to the curve as the length of the shortest line connecting the two. Nearest neighbour is matching to the most likely candidate road is not resilient to noise and can result in inefficient looping in driving path, or unexpected U-turns. Curve-to-curve matching matches the shape of the curve formed by consecutive observations with the curves formed by their candidate matches. The candidate matches for a curve most representative of the point curve, are selected as the right match. Once again, this method is sensitive to GPS error, since a curved formed by GPS points with error is going to be erroneous with likelihood of mismatches.

Topological map matching algorithms (Quddus, Ochieng, Zhao, & Noland, 2003; Zhao et al., 2017) utilize both, geometry and connectivity of segments in network to determine the most likely path of the vehicle. Algorithms that account for the network topology have reduced error rates since they account for the plausibility of a sequence of points following the candidate path based on the topology of the network. In this approach, the most likely candidate match for each observation is determined using a metric that calculates similarity between road network geometry and actual observations. The metric used to identify a candidate includes a weighted sum of (1) difference between road bearing and the vehicle's direction of travel, (2) distance of GPS point from candidate road, and (3) connectivity of the candidate road from

the previous road. However, a single mismatched observation can have an impact on the overall path determined by the algorithm since multiple paths are not simultaneously evaluated.

The third category of map matching algorithm uses probabilistic models to identify trajectory routes from GPS recordings. Newson and Krumm propose a probabilistic map matching approach (Newson & Krumm, 2009) using a Hidden Markov Model (HMM). This approach is applied to map-matching by using the noisy GPS recordings as the sequence of *observations*. Using probabilistic modeling for the potential sequence of states, the most probable sequence of points matched to the road network is then selected as the output for the HMM-based map-matching model.

3.4 Spatial Partitioning for Statistical Information Management

Several index structures have been proposed to improve the performance of trajectory information retrieval when using spatial and/or temporal filters, or when using trajectory trip parameters (origin, destination, trip start). A common index structure for spatial data is the R-tree (Guttman, 1984). R-tree indexes use a depth-balanced tree for multi-dimensional information (e.g. spatial coordinates). Levels in the tree are created by recursively grouping spatially co-located objects, and representing the grouped objects using Minimum Bounding Rectangles (MBRs). MBRs are organized hierarchically to create a depth-balanced index structure where all leaf-level nodes are at the same level. When using the R-Tree framework to index the 2-dimensional trajectory path represented as a polyline, a single MBR would be created for the path traveled by the trajectory. However, the MBR can be a poor representation for individual trajectories since trajectories for long travel could result in large volume MBRs overlapping for multiple trajectories. The selectivity of the spatial index could be poor for region-based queries since the bounding region for several trajectories could overlap, resulting in poor selectivity. To account for this issue, variations of R-tree based indexes have been described in literature to improve efficiency for indexing trajectories.

Other research has explored partitioning of larger trajectories and grouping similar sub-trajectories (Botea, Mallett, Nascimento, & Sander, 2007; Cudre-Mauroux, Wu, & Madden, 2010; Rasetic, 2005). For example, *TrajStore* is an adaptive indexing structure where an optimal trajectory splitting strategy is first used to create trajectory partitions. Trajectories similar in shape are then clustered together. A lossy compression technique is employed to store a representative sub-trajectory for each trajectory cluster.

Queries on the indexed trajectory data first use the spatial indexing structure to retrieve spatial regions followed by a scan of a time-based index structure to further filter the data on the temporal predicate.

Efforts have also been made to improve retrieval of trajectory location recordings based on trip-level metadata, such as, origin, destination, or travel time. *TripCube* is a cube-based indexing structure for efficient retrieval of massive trajectory position logs (Xu, Zhang, Claramunt, & Li, 2018). Xu et al. define a two-level trip-oriented vehicle indexing structure to retrieve the location recordings along the path followed by a vehicle. The authors argue that typically trajectory datasets are queried on origin, destination or trip start time filters. Thus, they propose a cube-based index structure where the trip metadata <*origin, destination and departure time*> is used to index vehicle trips. The index records reference a set of vehicle metadata records, storing information on the start location and the sequence of position records to be retrieved from vehicle GPS logs. Such an indexing structure is designed for faster retrieval of raw position observations for massive trajectory datasets.

Papadias et al. propose the use of non-overlapping spatial partitions to store aggregate information for trajectories traversing a partition (Papadias, Yufei Tao, Kanis, & Jun Zhang, 2002). The use of the partitioning structure has been demonstrated in exploring trends in movement data, such as, variation in traffic congestion over time for a spatial region. For example, a grid can be used to partition the spatial region using uniform-sized grid cells. Trajectories are split at boundaries of grid cells to identify movement within the cell during a time interval. This aggregate information can then be used to analyze overall movement trends in space and time. Although structures such as a grid structure can be used to discretize and manage statistics for spatial regions, a single cell could overlap multiple roads from a road network. Thus, information on movement semantics such as path taken by the object are lost (Leonardi et al., 2014).

Wang et al. describe a spatial summarization technique to enable filtering of large spatial datasets using a hierarchical grid based structure (Wang, Yang, & Muntz, 1997). The hierarchical grid is created by recursively partitioning each dimension in the spatial region to create rectangular cells. This creates a hierarchy of rectangular cells, varying in resolution at each level, with a single cell enclosing the entire spatial region at the root level, as shown in Figure 3-3. Thus, the cell at level *i* corresponds to the union of the cells at level *i+1* with the root level (level 1) covering the entire spatial region. Attribute statistics are then pre-computed for all objects in grid cells at the leaf level. This statistical information includes count of objects in the cell, average attribute values, attribute standard deviations and information on attribute data distribution. Parent cell statistics are then computed using statistical information of its child grid cells.

For queries on object attributes, the queries are answered by recursively exploring child cells that satisfy the query constraints using the summary statistics, significantly reducing the search space.

The computational complexity of a query answered using the grid structure varies based on the query. If the intent is to retrieve regions that satisfy query constraints, a scan of the grid structure is sufficient to identify regions meeting the criteria. The computational complexity is hence, reduced from O(n) to O(K), where K is the number of base cells. If individual objects that satisfy the query constraint need to be retrieved then in the worst-case scenario, all objects would need to be retrieved in addition to the overhead of scanning each grid cell.

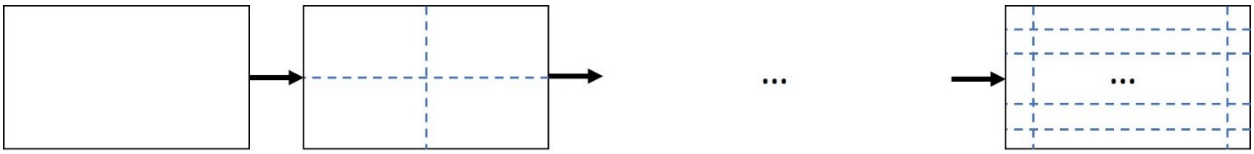

Figure 3-3 Hierarchical partitioning of 2-dimensional space to create rectangular cells of different resolutions. The first level consists of the entire region represented by a single cell. At each level, each dimension is bisected to create 4 cells per top-level cell. Each cell of the (i-1) th level corresponds to one cell of the ith level.

The dataset in this thesis is also attribute rich with a need to retrieve road segments based on spatial filters. The grid-based hierarchy is well-setup for problems exploring spatial data points in an unconstrained spatial region. In the case of the trajectory dataset, the underlying road network constrains the movement of vehicles. Since the problem focuses on haul road analyses, we explore a spatial partitioning approach that utilizes the road network topology. In this thesis, we thus test an alternative hierarchical partitioning approach that could benefit from the road network representation and partition the spatial search space using this approach.

3.4.1 Graph Partitioning

As discussed in the previous section, a road network topology is generally represented as a graph where the vertices are the junction points between a set of edges. Partitioning of graphs is a fundamental problem of computer science widely used to decompose a graph-representation of a problem into separate units that can be processed concurrently. The technique is widely used in complex networks, such as large social networks for load balancing with minimized communication between the partitions.

A graph-partitioning algorithm can then be used to partition the nodes of a graph into *k* distinct partitions. Graph partitioning has applications in various areas such as VLSI designs, task scheduling and scientific computing. The approach can be extended to transportation road network. Graph partitioning algorithms have following objective functions:

1. Minimize number of connecting links (edges) between the various partitions
2. Partition into nearly balanced groups with nearly equal number of vertices.
3. Generate partitions that cover all vertices and be mutually exclusive.

Consider a graph defined as $G = (V, E)$ where V is the set of vertices, and E is the set of edges. Graph partitioning can be used to create k partitions $\{V_1, V_2, \ldots, V_k\}$ such that each partition contains roughly the same number of vertices. Further, the vertices are disjoint $V_i \cap V_j = \emptyset$ *and* $V_1 \cup V_{2\ldots} \cup V_{k=} V$ and the number of edges that are between vertices in different parts of the graph (also called the cut-set) is minimized. Figure 3-4 is an example of 2-way partitioning for a graph, where the original graph is bisected to generate two subgraphs.

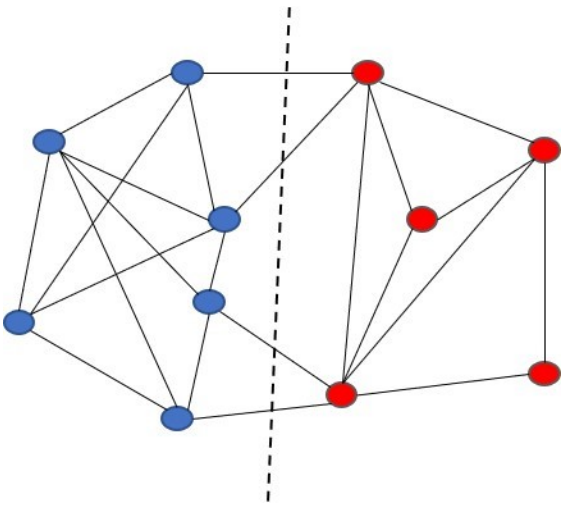

Figure 3-4 Example of 2-way graph partitioning that bisects the graph into two non-overlapping set of vertices (blue and red vertices).

3.5 Summary

In summary, our work proposes a data warehouse design to store summary for sections of roads in data warehouse. The summary statistics on shorter segments allows vehicle performance trending over time

and several classes of region-based queries on vehicle parameters. Previous eco-driving studies were conducted on smaller vehicle datasets or analyzed summaries for entire vehicle trips. These models are insufficient for a proper evaluation and understanding of large vehicle datasets. In this study, we evaluate a larger vehicle dataset with multiple attributes, and propose a data warehousing approach that can be used for exploratory analysis of vehicle trips across road sections within a road network. Further, we develop an effective hierarchical road partitioning strategy to pre-compute summary statistics and provide fast approximations for region-based queries. The next chapter explains the methodology in detail.

Chapter 4. Methodology

The previous chapter discussed prior work done for trajectory data warehousing. This chapter discusses the methodology and outcome of various pre-processing steps prior to populating the data warehouse. First, the process for monitoring and recording information for equipment at a surface mining operation is described. This is followed by the data collected for analysis and identification of trajectories. Next, we discuss the approach taken to identify haul truck travel route using map-matching process. This is followed by the schema of the data warehouse used to store historical vehicle travel information. The chapter concludes with the design of spatial hierarchy for the road network, called *GP-Tree* using graph-based partitioning. Further, we describe a method to compute approximate descriptive statistics for attributes when querying using spatial filters.

4.1 Equipment Monitoring in a Surface Mining Operation

Surface mining at Syncrude is a 24/7 operation where shovels excavate ore or overburden material. This material is then transported via haul trucks to the location of the dump site. Ore is the bitumen rich material that is resized into smaller lumps at large double-rolled crushers and then transported to the plant site for further processing. Overburden is the rock and soil above the ore body and needs to be removed to allow access to the ore. It is then used for construction activities such as building dams, roads and dykes; or it can be isolated in an enclosed area for future placement.

Using pre-surveyed geological information for the mine and daily visual surveys, the geology of the material at the excavation site is determined. Material at each dig location in the mine is then directed by dispatchers according to a daily mine plan. A central dispatching system is operated by the dispatcher to direct trucks and shovels to meet mining targets. Further, the dispatch system also logs information on each haul truck cycle, where a haul cycle is the metadata related to movement of a haul truck from its current location to the site of a shovel, and then to the site where the material is dumped.

Based on the model of the haul truck, individual sensors mounted on a truck record its position information and other variables, such as, speed, RPM and fuel level.

4.2 Data Collection

Values for individual haul truck sensors are archived in a system called Process Information (PI). Each sensor is recorded as a separate data stream of timestamped values. The PI system was queried to retrieve the vehicle sensor for haul trucks active during the period from September 1, 2015 to December 31, 2015. This interval corresponded to a period 2 months prior and 2 months after the capture of a satellite image, as explained in Section 4.4. During this period a total of 186,625 haul cycles were completed by 78 different trucks. The total distance covered by all haul trucks during this period was 1.8 million kilometers, with an average of 10 km distance traveled per haul cycle.

Table 4-1 lists the haul truck sensor values recorded in PI. The list of sensors varied based on the truck manufacturer and model. As shown in the table, GPS recordings were archived separately as four different sensor data streams: Latitude, Longitude, Heading and GPS speed.

Next, metadata for each haul truck cycle captured in the truck dispatch system was collected for the same 4-month period in 2015. Each haul cycle record contains information on the sequence of events in a haul truck cycle. An empty truck starts a new haul cycle at a dump location. The start of a haul cycle is recorded in the dispatch as the beginning of a new haul cycle record which is continually updated during the haul truck cycle. A single record contains several pieces of information. The time at which the truck leaves the dump is recorded as the **cycle start time**. The haul truck then travels to the location of its load recorded as the **source id**. When the truck parks itself near the shovel, the haul cycle record is updated with the **load start time**. Once the material is loaded on the truck, the **weight** of the load and the **type of the material** gets updated in the haul cycle record. This is followed by an update to the **dump end time** and

destination id once the material is dumped by the truck at the dump site. A new cycle record then gets generated to track truck movement for next the haul cycle.

A typical haul cycle record thus consists of the information described in the tuple:

<*Haul Cycle Identifier, Truck Identifier, Shovel Identifier, Origin Identifier, Destination Identifier, Tonnage Hauled, Cycle Start Time, Cycle End Time, Dump Start Time, Dump End Time, Material Type*>

Table 4-1 The list of sensor readings taken from haul trucks. This list can vary based on the make and model. Location information was recorded separately as 4 different data streams (Latitude, Longitude, Heading, Speed) and had to be merged to determine position history.

Sensor Name	Usage	Units
Latitude	Dimension	degrees°
Longitude	Dimension	degrees°
Heading	Dimension	degrees from North
GPS Speed	Numerical Attribute	kmph
Ground Speed	Numerical Attribute (Continuous)	kmph
Engine Speed	Numerical Attribute (Continuous)	RPM
Engine Fuel Rate	Numerical Attribute (Continuous)	gal/hr
Engine Coolant Temperature	Numerical Attribute (Continuous)	°F

4.3 Generating Haul Truck Trajectories

As described in Chapter 2, a trajectory is a sequence of timestamped position recordings for a vehicle. Using the sensors recorded for each haul cycle, position information was reconstructed by time-aligning latitude, longitude and heading sensor values. These timestamped recordings were then split into individual trajectories. This process is described in the sections below.

4.3.1 Derivation of Position Recordings

Time-stamped latitude, longitude and heading data streams were time aligned to get GPS recordings for each truck. The algorithm used to create these position recordings is outlined in Algorithm 4-1. Longitude and latitude sensor data sets were first ordered by their timestamps and merged using a full outer join.

Next, we iterated through the joined set of records to determine rows with either longitude or latitude values missing. If either the latitude or longitude value was missing, it was backfilled using the value recorded at the previous timestamp, if the gap between consecutive recordings was lower than a threshold, $\delta_{seconds}$. This approach is valid since the archival system (PI) is configured to record values only when there is a change in the value of the sensor. Next, the list of heading records was looked up for any sensor recordings at the timestamp.

Finally, all rows with either latitude or longitude records missing were discarded and the list of timestamped <latitude, longitude, heading> tuples was then returned as the GPS log for the equipment.

```
Input:
latitudes <- ordered list of timestamped latitude records for a truck
longitudes <- ordered list of timestamped longitude records for a truck
heading <- ordered list of timestamped heading records for a truck

Output:
      List of timestamped location recordings.

Algorithm:
joined <- fullOuterJoin(latitude, longitude)

FOR i = 2 to joined.Count:
      row = joined[i]
      IF(row.latitude = null or row.longitude = null) THEN
            previousRow = joined [i-1]
            IF(row.Timestamp – previousRow.Timestamp < δseconds) THEN
                  CASE WHEN (row.latitude = null)then
                        row.latitude = previousRow.latitude
                  ELSE
                        row.longitude = previousRow.longitude
            END
      ENDIF
      headingRow <- heading[heading.Timestamp = row.Timestamp]
      row.heading = headingRow.value
ENDFOR

joined.remove(x => x.latitude = null or x.longitude = null)

RETURN joined
```

Algorithm 4-1 This algorithm describes the process of creating position logs for each equipment using the timestamped values from latitude, longitude and heading sensors.

4.3.2 Identification of Individual Trajectories

Vehicle load can have an impact on vehicle performance. For example, fuel consumption is higher when hauling material as opposed to traveling without load. Each truck haul cycle was split to create two trajectories using information in haulcycle records table:

1. Period when truck was traveling empty. (period between the cycle start and load start time)
2. Period when truck was traveling with material (period between the load end time and the dump end time).

These two travel periods were used to split the position recordings for the equipment and define individual trajectories. Each trajectory record was then stored as <*Trajectory Identifier, Truck Identifier, Start Time, End Time, Source Identifier, Destination Identifier*>. The process resulted in a total of 373,250 trajectories.

4.4 Road Network Digitization

Surface mine road network changes frequently to accommodate changes in pits. Temporary haul roads could be added or removed to create a route to access a new pit location or to provide alternative access routes. Information on the boundaries for each haul road are recorded in a database system, however, the method of digitization only records the road boundaries for partial sections of the road. These boundaries are not connected and is stored as disconnected segments. Hence, using boundary data for determination of the road network topology would be non-trivial. Further, information on commissioning and decommissioning of haul roads is not accurate providing limited confidence in set of boundaries generated. Hence, the surveyed information was not used for this thesis and a satellite image of the mine was instead digitized to outline the roads in the mine.

As described in Section 2.2.2 a road network can be defined as a graph, where the edges represent the connectivity of roads. Since a digitized road network structure was not available, a satellite image of the mine area was manually digitized to define the haul roads. This digitized representation of the haul roads was needed to highlight topology of the underlying network. Recommendations on creating a road network layer using satellite imagery were followed (“Digitizing Map Data — QGIS Tutorials and Tips,” n.d.). Each visible haul road was digitized using a polyline at the center of the haul road visible in the

satellite image. Roads were terminated at intersections and connected to the other roads. Inflections in the polyline indicated a change in the driving direction.

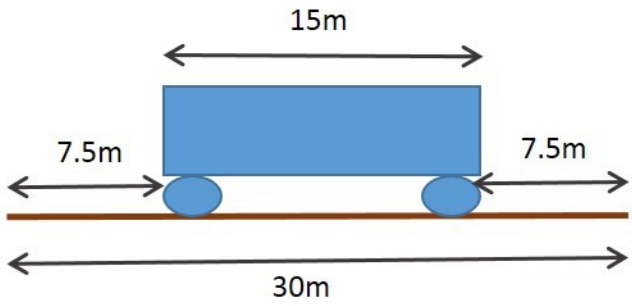

Figure 4-1 A truck traveling on a road segment. The largest haul truck can have a length of 15 m and is installed with GPS receivers having an accuracy of 7.5 m. Depending on the location of the receiver (front or back of the truck), the center of the truck would be represented by a GPS measurement value within a 30 m distance.

The maximum length of a haul truck is 15 m and a GPS on the truck has a horizontal measurement accuracy of 7.5 m. We wanted to account for a truck on the road when the center of the truck was on the road section. Depending on the position of the installed GPS receiver (front or back of the truck), the center of the truck could be represented by a measurement captured anywhere within a 30 m diameter (see Figure 4-1). Hence, the size of the base granule needed to be larger than 30m. A road segment with a length of 60 m would have a 45 m section of the road (75% of its entire length) where a GPS measurement would indicate that the truck's midpoint was on the road. Hence, after the digitization process, each road was partitioned into approximately similar sized segments of 60 m length to create the base granule for the spatial hierarchy, as shown in Figure 4-2. The digitized and partitioned road network resulted in 7234 road segments for a combined road length of 434 km.

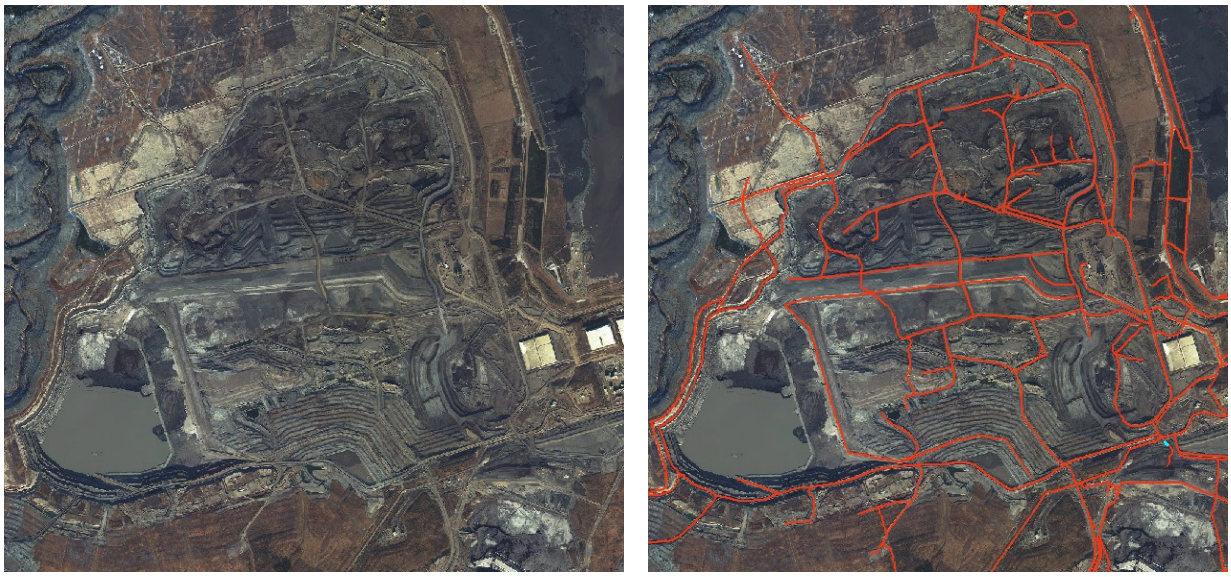

Figure 4-2 The image on left is the original satellite image of the mine while the one on the right shows the digitized road network overlaid. Each road in the network was partitioned into smaller segments of 60m to create the base granule for the trajectory warehouse.

4.4.1 Trajectory Decomposition for Extract-Transform-Load (ETL)

The trajectory decomposition described in Section 3.3 was used to determine the sequence of sub trajectories identifying the path taken by a vehicle on the road network. First, a probabilistic map-match approach was used to map individual points to the base network. Implementation of a Hidden Markov Model-based map matching algorithm by Newson et al., Sandwych HMM (Li, 2018) was modified to determine matches for vehicle location recordings (Newson & Krumm, 2009). Further, a web-based tool was developed to verify the trajectory route and its map-matched path using ASP.NET MVC 5, Leaflet JavaScript library and Microsoft SQL Server 2016. GDAL2Tiles utility was used to create tiles for the base satellite image used in the web tool and GPS coordinates for the recorded GPS points were converted into metric coordinates to match the projection system of the base map.

Figure 4-3 Example trajectory with large gaps in recordings due to zones with poor satellite reception. The first gap is 53 seconds, second gap is 203 seconds, third gap is 78 seconds, fourth gap is 256 seconds (left to right) with the linear distance travelled between the recordings ranging from 579m to 2.5km. Corresponding to the gap periods, 235 sensor readings for the fuel sensor were recorded.

The dataset collected suffered from an approximate base map representation and intermittent gaps greater than 30 seconds between consecutive GPS recordings. As can be seen in Figure 4-3, within 30 seconds, a vehicle can travel 250m – 600m at a speed ranging from 30-60kmph. The gaps were suspected to be the result of dead-zones or errors in data archival. Post completion of the map-matching steps, of the 63 million location recordings, 40 million were matched to the road network. This represented approximately 63% of matched points. This was followed by the remainder of the trajectory decomposition steps to determine vehicle route. This resulted in 15 million road travel records. This dataset was loaded into the trajectory data warehouse described below.

4.5 Road Network Data Warehouse Design

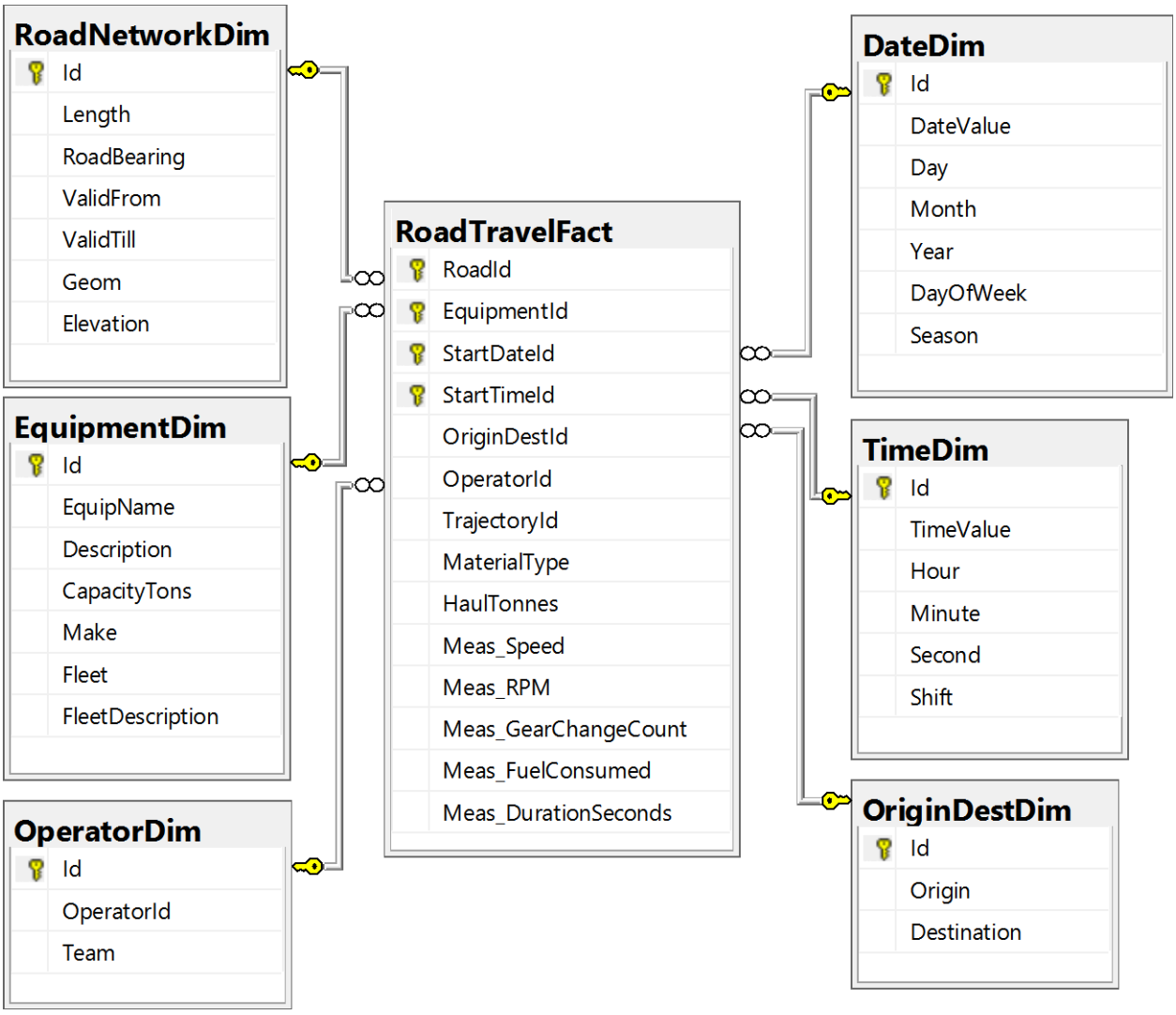

Figure 4-4 Star-schema based implementation of the data warehouse used for analysis of haul roads. A central *fact* table, *RoadTravelFact*, stores information on vehicle performance on various haul road segments. Associated parameters used for querying and filtering are stored in the *dimension* tables (*RoadNetworkDim*, *EquipmentDim*, *OperatorDim, DateDim, TimeDim* and *OriginDestDim*).

Figure 4- 4-4 shows the schema of the trajectory data warehouse realized in Microsoft SQL Server 2016. A key requirement for data warehouse design is the selection of the base *granule*. Movement of a single vehicle on a road segment was selected as the base level *granule* stored in the *fact* table since the intent of the data warehouse was to provide analysis for individual haul road segments. The central *fact* table, *RoadTravelFact*, was populated with information on vehicle travel for each road segment along with

corresponding *Measures* such as, vehicle material load (tonnage), Engine RPM, fuel used and number of gear changes.

Adopting a *star schema* design for the data warehouse, *RoadNetwork, Equipment*, and *Operator, Date, Time and Origin-Destination* were selected as the dimensions used to slice-and-dice the data.

- *RoadNetwork* dimension contained the list of haul road segments along with the associated metadata.
- *Equipment* dimension contained the list of all haul trucks with metadata on make and model of the haul trucks, material haul capacity and fleet information.
- *Date* and *Time* dimension provided the ability to aggregate records on road trip start.

Since records in the *RoadTravelFact* table contained information on travel information for each road segment, a decision had to be made regarding attributes for the entire vehicle trip. Key dimension tables were created to store trip-level information recorded for each haul truck cycle. The *OriginDestDim* table recorded the trip origin and destination locations. Each origin and destination pair was created using named location identifiers in the original transactional database. Having a single combined dimension resulted in fewer join queries when retrieving data from the data warehouse.

The *OperatorDim* table was used for information identifying truck operator for each vehicle trip. This too was incorporated and stored for each road travel record in the data warehouse.

Further, information identifying the trajectory was incorporated as a degenerate dimension. Each *fact* table entry thus contained the original trip record that the entry belonged to. As described earlier in Section 4.3, a trajectory record consisted of metadata identifying each individual trip for a truck when it is either traveling with load or traveling empty. Including this identifier in the *fact* table helped filter on distinct number of trips when aggregating travel information across roads or time steps. This allowed extension of the schema for trip-level analysis, in addition to road network performance.

Finally, a composite key was selected as the clustered primary key for the *RoadTravelFact* table and consisted of a subset of reference to dimension columns <*RoadId, EquipmentId, StartDateId, StartTimeId*>. The dimensions were setup with surrogate keys, where the key had no intrinsic meaning to permit changes to the descriptive dimension values. This also allowed graceful handling of missing dimension records in the *fact* table. For example, to satisfy the foreign key constraint on *Operator*

dimension, records with missing operator information referenced the surrogate key for operator dimension record with a default value of -1 for records with missing operator information. Further, having a surrogate key also mitigated concerns around records being missed when values are grouped by a NULLABLE dimension value.

Measures for each attribute for a road segment was calculated as the average of all the sensor recordings that intersected the period of travel on the road segment.

4.6 Road Network Hierarchy for Approximating Descriptive Statistics

Data warehouses are typically used for summarization or aggregation of data across one or more dimensions. Aggregation of attribute values for data points within a spatial region is commonly carried out to visualize spatial variations. These aggregations can be computed using all data points within the spatial region, however, interactive exploration of this summary information can be expensive and slow when the volume of data points within the spatial region is large. To improve responsiveness of the data warehouse when computing descriptive statistics (minimum, maximum, mean) for spatial regions, we propose a method for fast computation of approximate attribute values. With that objective we develop a generic spatial partitioning for the road network and use that to approximate the descriptive statistics for each attribute.

We describe our approach in the following sections. First, we discuss the development of a hierarchical structure for the road network by recursively partitioning the road network graph. Next, we describe the use of our road network hierarchy to select hierarchy partitions with a spatial query window. Finally, we conclude with the method used to approximate attribute statistics using our road network spatial hierarchy.

4.6.1 Developing a Road Network Hierarchy

Partitioning for transportation networks of large geographical regions is typically achieved using separation at geopolitical boundaries, such as, Continent → Country → Province → City→Municipal regions. The highest level could contain statistics, such as, average travel time, for national highways, followed by provincial highways connecting major hubs in different provinces at the next level, followed by a level storing the city's arterial roads. The leaf level would contain all the interior roads for each community. To prune the search space for spatial queries, multiple levels of the network hierarchy are

then iteratively traversed to terminate at the required depth. At each stage, regions of the graph could be removed from the search space by eliminating sections of the graph that do not apply to the query. For example, a search between the city-centers of two major cities in different provinces would be managed using multiple queries, as follows:

1. Query the path between the two cities by doing a search of the country level and the provincial level highway network.
2. Query the city sub-network to connect the major city artery to the highway system.
3. Query the interior sub-network to compute the path to the city center from the arterial road.

If the conceptual partition and hierarchy of the spatial region is not available (as is the case for our haul road network) a generic spatial hierarchy can be created using grid-based partitioning. A simplified grid-based approach is used where each region is iteratively split into smaller areas. However, the grid-based approach does not leverage the road network shape, topology or segment distribution in space when creating the spatial partition.

Our trajectory data warehouse was designed to enable spatial analytics on sections of haul roads. To accommodate this need, road segments were selected as the spatial *granule*. Due to the volume of information stored in the data warehouse *fact* table, however, computation of attribute summaries within a spatial region had poor performance.

Summary tables are used in data warehouses to compute sub-totals for data warehouse dimensions and improve the performance of aggregate queries. For example, a data warehouse that is used to track all store sales report on daily sales values could contain a large volume of sales records per day. A summary statistics table for the *Time* dimension could then be used to store daily sales summaries and improve performance of reports generated for daily or monthly sales.

A similar approach could be used to enhance query performance for aggregate spatial queries. However, an appropriate spatial hierarchy which could be used for aggregation of road-level records when pre-computing summary information for spatial queries was missing. Hence, a spatial hierarchy was developed that accounted for the structure of the road network and balanced the distribution of road segments across partitions of the hierarchy.

4.6.1.1 Partitioning the Set of Graph Edges

As described in the previous chapter, graph partitioning algorithms have multiple objective functions one of which is to partition the graph into distinct groups containing nearly equal number of vertices. For this a multi-level recursive partitioning algorithm (Karypis & Kumar, 1998) was selected. However, in this application, there was the need to distribute network edges (i.e., road segments) into distinct and contiguous components. The approach taken to partition the edges is described below.

METIS is a state-of-the-art software tool for graph-partitioning implementing a multi-level recursive partitioning algorithm. The graph partitioning algorithm works in three steps: 1) coarsen the graph 2) partition the coarsened representation 3) un-coarsen the graph to get the final partitions. This algorithm works on reducing the size of the graph by recursively "coarsening" the graph where edges and vertices are collapsed to create a smaller graph. Partitioning is then carried out for the smaller, contracted representation of the graph. The partition is then projected back through the coarsening sequence to get the partitioned graph.

Typical road networks are not uniform and consist of dense regions or communities separated by natural separators such as bridges or mountains. Partitioning approaches tailored to road networks, such as PUNCH (Delling, Goldberg, Razenshteyn & Werneck, 2011), pre-process a graph so that the partitioning accounts for "natural" cuts in the graph. For example, PUNCH contracts dense regions of the graph prior to partitioning. This preserves communities by selecting roads that interconnect the dense sections as the edge cuts. However, our graph structure for haul roads contained long haul roads with few vertices having degrees greater than 2 (at intersections), which is very different from typical road networks. The need for pre-processing used for typical road networks was thus, not required. Hence, we used METIS for graph partitioning since it is a more general and the most efficient approach out of all available ones (Sui, Nguyen, Burtscher & Pigali, 2010).

We used the *METIS* solver recursively to partition the graph and generate a hierarchy of subgraphs. Recursive bi-sectioning was applied such that at each level i a total of 2^{i-1} disjointed subgraphs with a mutually exclusive set of road segments was generated. First, the entire graph was selected as level 1 or root of the hierarchy. Graph partitioning was applied to bisect the root level graph with nearly equal number of vertices at level 2. Edges spanning the two disjointed partitions (referred to as the *cut-set*) were distributed between the bisections as follows:

1. The smaller of the two subgraphs was first assigned a random selection of edges from the *cut-set* to balance the number of edges in the two subgraphs.
2. Any remaining edges were then distributed equally between the two subgraphs.

Partitioning of a subgraph was terminated once the subgraph consisted of a single edge, or the partitioning algorithm failed to find any further contiguous partitions in the subgraph. Thus, a hierarchy of subgraphs was generated, where each parent subgraph at level *i* consisted of the union of two subgraphs at level *i+1*. Further, for each subgraph at level *i* the child subgraphs generated at level *i+1* were nearly balanced in the number of graph vertices.

4.6.2 Creating a Graph Partitioned Hierarchical Structure (*GP-Tree*)

The hierarchy of graph partitions resulted in a binary tree structure where each tree node consisted of a subgraph with mutually exclusive set of edges. The tree height was selected to be the depth of path to the level where all subgraphs had a minimum acceptable edge density. This resulted in a depth-balanced tree, henceforth referred to as *GP-Tree*.

To enable use of *GP-Tree* for filtering on spatial queries, each node of the tree also stored a geometric representation of the subgraph. In this case, we selected the Minimum-Bounding Rectangle (MBR) to represent each subgraph. Hence, each internal node of *GP-Tree* consisted of the tuple <*MBR, ChildNodePointer1, ChildNodePointer2*>. Further, the leaf-level entries in *GP-Tree* stored a reference to the set of road segments that were a part of the leaf-level subgraph <*MBR, Road Segment Pointer*>.

4.6.3 Spatial Searches using *GP-Tree*

With the partitioned hierarchical structure of the road network using *GP-Tree*, we describe its use as a spatial index. For a region-based query, the tree was traversed in a top-down manner to retrieve the set of road segments in the query region.

At each node of the tree, the MBR for the subgraph is used to evaluate whether the subgraph is relevant to the search region (see Figure 4-5). First, the root-level MBR is evaluated for overlap with the query region. If the MBR for the root-level node intersects with the search region, the child level nodes are traversed further and evaluated as follows:

1. If the MBR for the node does not intersect the query region, the child nodes are not explored and further test against the child nodes are not carried out.
2. If the MBR for the node are contained within the query region, no future spatial comparisons are performed for the child nodes. The branch of the tree is traversed up to the leaf-level and all the road segment pointers at the leaf-level are returned as the part of the query result.
3. If the MBR for the node overlaps with the spatial query region but is not contained entirely within the search region, the child elements are not explored further.
4. If the node is a leaf-level node and the node MBR intersects the query region, road segments referenced by the leaf-level node are retrieved. This first primary filtering step can result in false positives. Hence, the retrieved road segments are individually compared in a secondary filtering step to determine object overlap with the search region. Pointers to all the road segments that intersect with the query region are then returned as the result set.

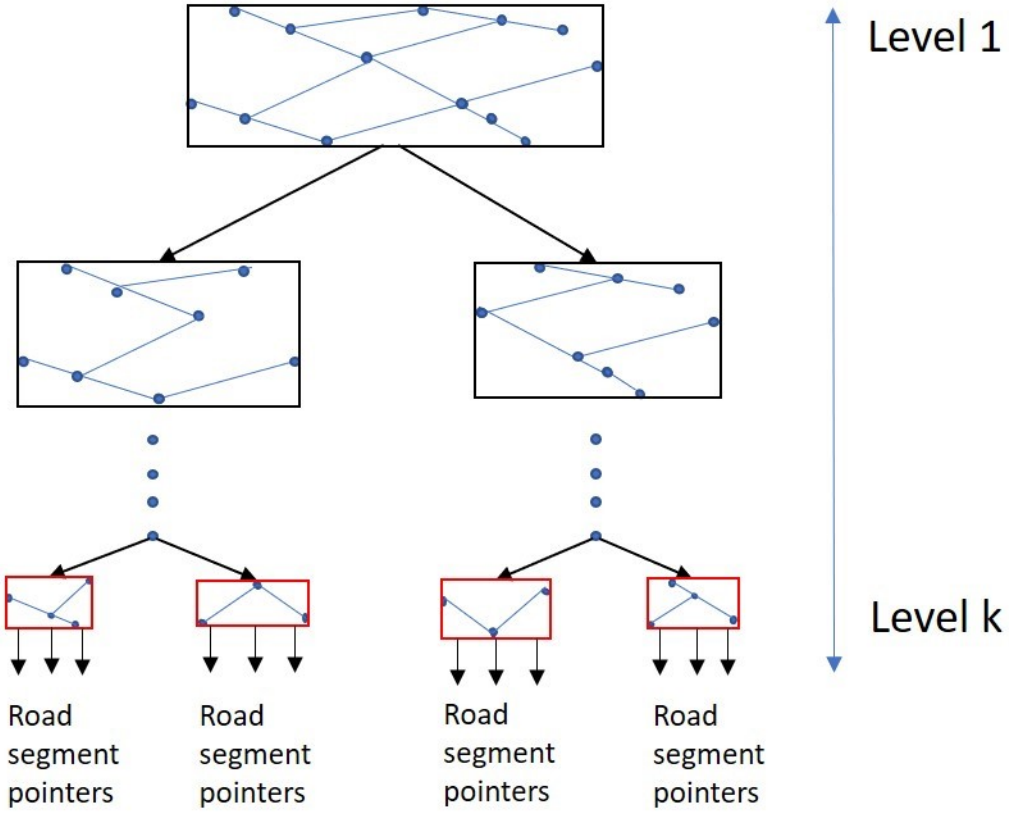

Figure 4-5 The image shows the conceptual representation of *GP-Tree* and its usage as a spatial lookup structure. At each level, the Minimum Bounding Rectangle (MBR) for the subgraph is tested for overlap with the query window. The leaf level nodes (shown in red) store the MBR for subgraphs and keys to individual road segment

records within the subgraph. Geometries for all road segments at the leaf level must be individually tested for overlap with the query window.

4.6.4 Using GP-Tree to Approximate Summary Statistics

In our method, histograms were used to create non-parametric estimates of distribution for each numeric attribute. Non-parametric methods relieve the need for parameters and make no assumptions about the underlying distribution. For each node of *GP-Tree* we generated a frequency histogram to describe the distribution of attributes collected for all travels on roads in the node subgraph. The frequency histogram for an attribute was then used to compute approximations for basic summary statistics for attribute values within a query region. The process to generate the histograms and use those to approximate summary statistics is described below.

We first created a frequency histogram for each *measure* (attribute) and sub graph within *GP-Tree*, and then used the histograms to approximate values for aggregate queries (minimum, maximum, average) within a spatial query window. The process for generating the histograms is described below.

```
Input:
binCount <- Number of bins in histogram for each attribute
attribute <- Attribute for which histogram must be created
roadTravels <- List of entries in the RoadTravelFact table

Output:
      List of road segments with a histogram for each segment

Steps:
attr_max = max(attribute)
attr_min = min(attribute)
bin_interval = [attr_max - attr_min]/ binCount


FOR i = 1 to binCount:
      bin_start = attr_min +(i-1)* bin_interval
      bin_end = bin_start + bin_interval
      bins[i] = [bin_start, bin_end)
END FOR

FOREACH entry in roadTravels:
      index = ⌊entry.attr / bin_interval⌋
      freq_bins[index] += 1
END FOREACH

RETURN bins
```

Algorithm 4-2 Steps used to create frequency histograms for each road segment attribute. Separate histograms with uniform-width bins were generated for each attribute value. The width of each bin was determined using the number of bins provided as input for the algorithm.

The number of histogram bins for each attribute was selected using Sturges' rule (Sturges, 1926). It is an idealized frequency histogram with k bins, where binomial coefficient is used to determine the frequency of the *i*th bin. Hence, using binomial expansion, the number of samples that can be represented using Sturges' rule is as follows:

$$N = \sum_{i=0}^{k-1} \binom{k-1}{i} = (1+1)^{k-1} = 2^{k-1}$$

where N is the number of sample points. Using the above formula, the number of bins is calculated as:

$$k = \ Number\ of\ Bins = 1 + log_2 N = \ 1 + 3.3\ log_{10} N$$

The bin count is thus proportional to the log of entries in the *fact* table. Other non-parametric histogram-based methods, such as, Scott's rule (Scott, 2009) and Freedman and Diaconis's (Freedman & Diaconis, 1981) account for the spread of the data when computing the bin size. Scott's rule is proportional to the estimated standard deviation, whereas Freedman and Diaconis's rule accounts for the interquartile range of the data. However, their computation requires a scan and ordering of all the records in the data warehouse *fact* table.

Once the bin count was selected, a frequency histogram for each partition-attribute pair was generated. To speed-up generation of frequency histograms, road segment histograms for each attribute were first computed, steps for which are described in Algorithm 4-2. The road-level histograms were discarded at a later stage since in a production data warehouse environment, this step would only need to be carried out once on initial load. After the first load of the summary table, updates would only need to be performed incrementally during subsequent data load operations. Since we stored histograms for each level of the binary tree without significantly impacting the size of the approximation structure, we selected the Sturges' rule for bin-size calculation.

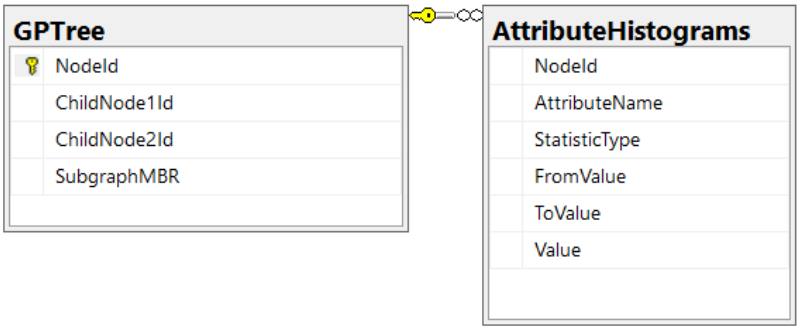

Figure 4-6 Table structure used to store histograms, minimum and maximum values for attributes. These histograms are used to calculate summary statistics for spatial queries.

Using the road-level histogram for an attribute, histogram for leaf-level nodes were generated by aggregating bin counts for roads belonging to the node subgraph. A bottom-up approach was then used to generate attribute histograms for the rest of the nodes of *GP-Tree.* Since the partitioned subgraphs at each level consisted of a mutually exclusive set of roads, the frequency distribution for partitions at all levels in the hierarchy could be computed using a bottom-up approach. Hence, the frequency histogram for each parent graph was then computed as the aggregation of histograms from its child subgraphs. In

addition to the frequency histograms, minimum and maximum values were also computed for each attribute-node pair.

The table structure used to store the histograms, minimum and maximum values for attributes is shown in Figure 4-6. The *GP-Tree* index structure is stored in table *GP-Tree*, with each record in *GP-Tree* containing reference to the child nodes, along with the MBR for the subgraph. The **AttributeHistograms** table is used to store the bins and the associated frequency counts, along with attribute minimum and maximum value for each node in *GP-tree*. Example data stored in **AttributeHistograms** table is shown in Table 4-2 . The table describes data for part of the frequency histogram for Engine Fuel Rate computed for the root node subgraph, that is the entire spatial region. The table also includes the minimum and maximum values for each node.

Table 4-2 Example summary pre-computed for Engine Fuel Rate attribute and the root node of *GP-Tree*. Each row contains a record for a numerical attribute and the bin [From, To) with the count of data recordings where the attribute falls within the bin. The table is also used to store the statistics for minimum and maximum values for the attribute.

Node Id	Attribute Name	Type	From	To	Value
1	Engine Fuel Rate	Bin	521.4867	546.3194	218017
1	Engine Fuel Rate	Bin	546.3194	571.1521	340769
1	Engine Fuel Rate	Bin	571.1521	595.9848	668950
1	Engine Fuel Rate	Bin	595.9848	620.8175	601743
1	Engine Fuel Rate	Bin	620.8175	645.6502	284641
1	Engine Fuel Rate	Maximum	-1.0000	-1.0000	645.65
1	Engine Fuel Rate	Minimum	-1.0000	-1.0000	0.0000

The pre-computed summary statistics for each partition were used to approximate min, max and average values for road segment attributes using the *GP-Tree* index. For a spatial region query, *GP-Tree* was traversed in a top-down fashion. As described in the previous section, *GP-Tree* was traversed to determine nodes with subgraphs that overlapped the spatial query window. Further, we introduced a user-specified threshold, τ, that was used to exclude graph nodes where the bounding region overlap with the query

region was less than τ. The final set of subgraphs that were *relevant* to the query were then selected to compute the summary approximations. Unlike the use of *GP-Tree* as a spatial index, records associated with individual road segments were not retrieved.

Once *relevant* nodes that satisfied a region-based query were selected using *GP-Tree*, the pre-computed summary statistic for the attribute values were used to approximate the values for minimum, maximum or average value for the query region. The minimum/maximum statistic from all *relevant* partitions was selected as the result of minimum/maximum query. For average value computation, the bin frequencies for the relevant partition attribute histograms were aggregated to create a frequency histogram for the query region. The average attribute value for road segments in the query region was then approximated using the aggregate frequency histogram as follows:

$$Average = \frac{\sum_{k=1}^{bincount} \left(From\,Value[Bin_k] + \frac{Interval[Bin_k]}{2} \right) * \; Frequency[Bin_k]}{\sum_{k=1}^{bincount} Frequency[Bin_k]}$$

Hence, for each attribute bin, the mid-point of the bin was scaled by the frequency of the objects in the bin. These values were summed up and divided by the total count of values in the aggregate histogram to approximate the average summary statistic.

4.7 Summary

In this chapter we outlined the data pre-processing steps and the design of our data warehouse structure. Significant data manipulation was performed to process the trajectories to be loaded into the proposed data warehouse structure. However, our spatial data warehouse still lacked an implicit or explicit hierarchy. To overcome this limitation, we developed a generic spatial hierarchy for our road network, called *GP-Tree,* using recursive graph partitioning. Finally, we leveraged the *GP-Tree* structure to improve performance of spatial aggregate queries against our data warehouse using a summary table to that pre-computes information required for our spatial queries.

Chapter 5. Results and Discussion

In the following sections we demonstrate the results of implementation of our data warehouse model using surface mining data for haul trucks. We first discuss the merits of using a data warehousing approach for the surface mining dataset. Next, we discuss differences between the data warehouse model implemented in this thesis, and a conceptual design reviewed earlier. We then describe a selection of queries for exploratory data analysis that are now possible using the data warehouse implementation.

The data warehouse was designed to improve historical analysis of haul road performance and was expected to grow in volume over time. However, the growth of data volume can introduce performance issues when running spatial summary queries interactively. To overcome issues with performance, we demonstrate the use of our novel spatial indexing structure, *GP-Tree*, for computation of fast approximations for aggregate spatial queries.

5.1 Advantages of the Data Warehousing Approach

Information recorded to enable exploratory analysis of vehicle haul trips was stored in disparate sources. The data warehousing approach assimilated information from various sources to enable ad-hoc analysis of haul truck trajectories or road segment performance information. Further, raw sensor recordings for haul trucks were not geo-referenced and contained periods of large gaps in position information. As a part of our initial data manipulation step, we resolved these issues to provide a complete representation of vehicle movement on the haul road network. Further, as an outcome of the map-matching and route determination process, over 60 million sensor recordings for vehicle trip information were reduced to 15 million points using the map-matching approach. Hence, the data manipulation step reduced the data volume by a factor of 4.

Further, we carefully selected our spatial granule such that we could record high level of spatial detail when storing aggregate information on roads in the road network. Hence, we partitioned the original roads in our road network into smaller sections, storing travel information for each section in our partition (road segment), to improve the spatial resolution of information recorded. Warehousing a higher level of spatial detail enabled tracking of variability in driving performance associated with road conditions.

Additionally, each record of the data warehouse contained information on multiple attributes recorded by haul truck sensors. Hence a large multidimensional structure was generated which could be used to

slice-and-dice on various dimensions of the data warehouse. Further, the multivariate information recorded for equipment travel on each haul road enabled historical analysis of road network performance.

5.2 Analyzing Road Network Data Warehouse Design

We first provide a summary of the differences between the conceptual data warehouse model discussed in Section 3.2 (Leonardi et al., 2014; Nardini et al., 2018), and the implementation of the data warehouse in this thesis. Then, we demonstrate queries possible using our Trajectory Data Warehouse model.

5.2.1 Simplification of the Data Warehouse Model

The data warehouse design by Leonardi et al.(2014) aggregated information on several trajectories in each spatio-temporal *granule* of the *fact* table. Hence, an approximation approach was required to avoid overestimating object count within a query window spanning multiple spatio-temporal boundaries. The authors proposed an additional *fact* table (*Inter-Granule Fact*) to approximate object transitions in space or time. Approximating the number of objects in a query window thus required aggregation of records from both, the *InterGranule* and the *IntraGranule fact* tables. On the other hand, our data warehouse model was used for exploratory analysis of road network and contained information identifying the trajectory that traveled a road for each record of the *fact* table. The identifier could hence be used to distinctly count the number of trajectories within the spatiotemporal query window, eliminating the need for an *algebraic* approximation. Additionally, this identifier could also be used to provide trip-level information for a single trajectory.

In the design by Leonardi et al., the pre-processing step required individual trajectories to be split between spatial *and* temporal boundaries of the base-level *granule*. For example, a trajectory first had to be partitioned into sub-trajectories when it crossed either a spatial or temporal boundary. In the proposed design, however, the need to partition the trajectory on temporal dimension was eliminated by using the *Time* dimension reference only for the start time of travel on a road segment. To determine travel end time on a road segment, the duration of travel was included as a *measure*. This reduced the complexity of the pre-processing step while still allowing queries on travel start/end time ranges.

5.2.2 Demonstration of Queries using the Data Warehouse

In this section, we highlight some of the queries possible with the proposed data warehouse. Although the data warehouse stores information on travel on individual road segments, information on entire

vehicle trips were still possible. Some examples of queries on trip-level metadata included number of trips between selected origin-destination filter, total overburden material hauled during the day shift or number of loads delivered to a location. Figure 5-1 is an example query where the number of trips between an object-destination pair is retrieved along with information on the average tonnage hauled during each trip. Note that the query has reduced complexity compared to the reviewed structure since the number of distinct trajectories can be queried from the *fact* table without aggregating measures from a secondary *fact* table containing information on object transitions between temporal or spatial granules.

```
SELECT  NumberOfTrips = COUNT(DISTINCT TrajectoryId)
       , AvgTonnage = AVG(HaulTonnes)
FROM RoadTravelFact f JOIN OriginDestDim od ON
       f.OriginDestId = od.Id
       JOIN DateDim d ON
              f.StartDateId = d.Id
WHERE od.Destination = @Destination
       AND od.Origin = @Origin
       AND HaulTonnes > 0
       AND d.Month = @Month
```

Figure 5-1 Query to find the number of trips between the selected origin-destination pair along with the average truck load. Note that an accurate count of vehicles that travelled between an origin-destination pair can be calculated by storing trip identifier in each record of the *fact* table.

In addition to trip-level metadata for different trajectories, the data warehouse could be queried to understand and variation in attribute measurements for trucks traveling on a selected road over time. The example query in Figure 5-2 averages engine fuel rate per shift, per road segment using vehicle travel information recorded for road segments. The results can be trended over time to variability in average engine fuel rate across different shifts for a selected road. Extreme variations during a shift can indicate a change that needs to be investigated.

Further, our dimension table, *RoadNetworkDim*, stores information on geometry of the roads in the network. Metadata identifying the type of road (e.g. intersection, straight road, or traffic circle) can also be included for each type of road. Roads can then be analyzed by selecting a subset of roads within a spatial region or selecting those on a specific attribute type. An example comparison of vehicle measures for a subset of roads of different types is shown in Figure 5-3. The selected road segments include one that is at an intersection, one along a straight travel section with no stops, and a road segment that is a part of a traffic circle. These segments are compared for their speed and engine fuel rate averages. As observed, the average speed profile for the road segment along a straight section consists of higher

speeds and varies from the road segment section at the traffic circle or that near an intersection (where the speed values lie in a smaller range and are centered around a lower speed value).

```
SELECT  f.RoadId
       , d.StartDate
       , t.Shift
       , AVG(Meas_EngFuelRate)
FROM RoadTravelFact f JOIN OriginDestDim od ON
              f.OriginDestId = od.Id
       JOIN DateDim d ON
              f.StartDateId = d.Id
       JOIN TimeDim t ON
              f.StartTimeId = t.Id
GROUP BY f.RoadId, d.StartDate, t.Shift
```

Figure 5-2 Query to understand variability in engine fuel rate consumed by vehicles traveling the same road section during different shifts.

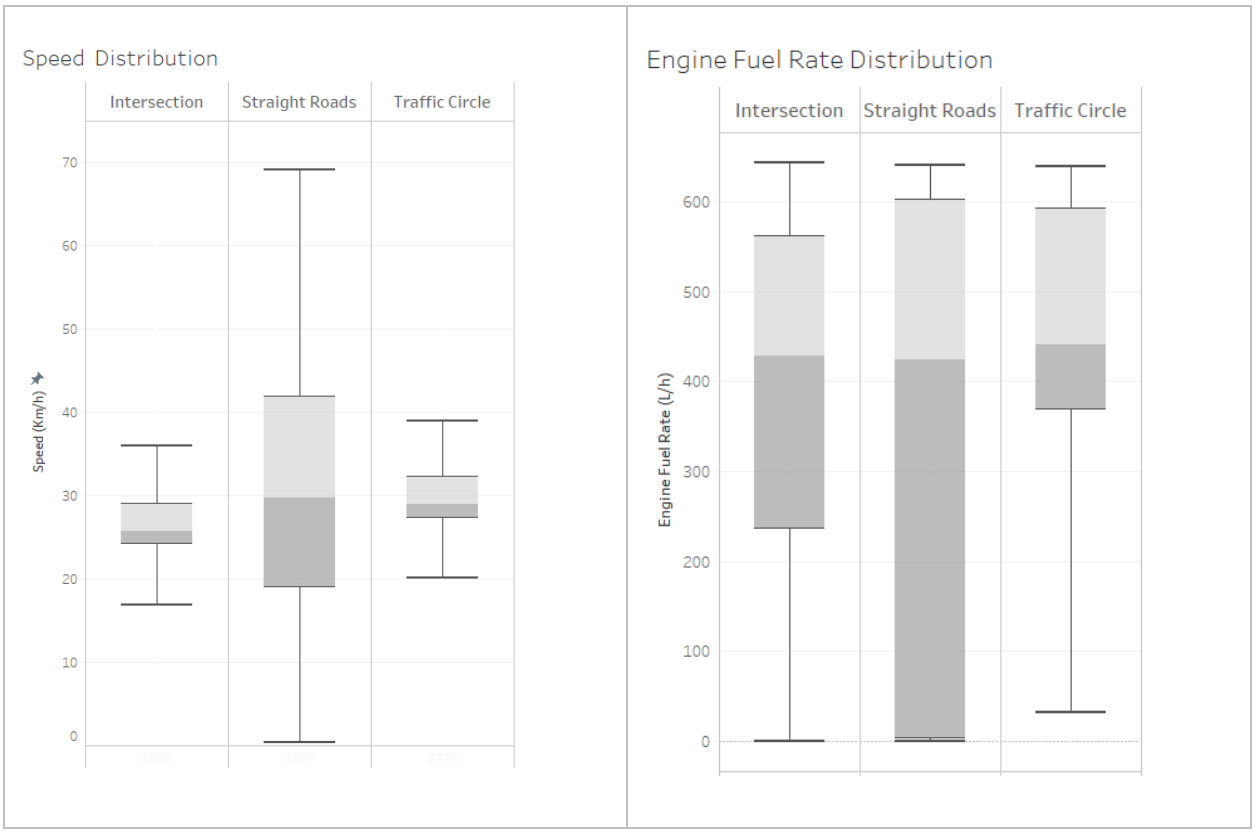

Figure 5-3 Comparison of different in road attribute values for a selection of road segments with different classification (Intersection, Straight Road, Traffic Circle). The chart on the left compares the distribution of speed values for different road segment types. As can be seen, the speed values for road at an intersection or in a traffic circle are distributed around a lower value and has a narrower spread when compared with a segment of a straight road. The chart on the right compares the distribution of engine fuel rates for different road segment types. Here too variability in fuel rates is noticed where the fuel rates increase for roads where the trucks must slow down (e.g., intersection, traffic circle).

5.3 Performance Evaluation of *GP-Tree*

A primary application of the developed data warehouse structure was to enable fast response time for spatial queries when computing descriptive statistics for the different attributes. For this we first developed a generic spatial hierarchy for our road network structure and pre-computed a summary table such that descriptive statistics on spatial query region could be answered using approximations, trading accuracy for performance.

To test our spatial partitioning approach, we first analyzed runtime performance of aggregate queries against *GP-Tree*. This approach was tested against methods that performed a direct lookup of records in the fact table. The list of all methods that we compared are outlined below:

1. *GP-Tree* structure as a spatial index: In this case, we used *GP-Tree* to scan the road network hierarchy and retrieve the set of road segments that fell within the query window. Summary statistics were then computed using a direct lookup of relevant road segment records in the base *fact* table.
2. Default Microsoft SQL Server spatial index: The default server spatial index was created on the road network geometry column to enable spatial search of roads within the query window. Summaries were calculated for road segments in the query window.
3. No Spatial Index: In this case, no spatial index on road network geometry was created and the query optimizer could select indexes on the attribute columns to retrieve relevant records.
4. *GP-Tree* to approximate summary statistics using the method described in Section 4.6.4

Queries of varying window sizes were used to retrieve records from the *RoadNetwork fact* table and then report on basic summary statistics (Minimum, Maximum and Average) for each attribute (*measures*). We first selected a set of 12 points randomly placed in the mine region and used those as centers for the spatial query windows. Next, we created multiple windows of increasing area around each center point by varying the length of the square bounding the query center. The resulting spatial windows ranged from 0.001% to 30% of the total mine area. Overall, for each query center we generated 15 different query windows for a total of 180 spatial queries.

All tests were run on a machine with 12GB RAM, 250 GB SSD drive and a 2.7 GHz dual core processor using Microsoft SQL Server 2012 database. The *fact* table records were clustered on <*RoadId, StartDateId, StartTimeId*> columns, and non-clustered indexes were created for all the attributes in the database. The

height of *GP-Tree* was selected such that the leaf-level subgraphs consisted of at least tens of road segments in each subgraph (approximately 0.6 km road length for the current application). For the mine road network, this resulted in a depth-balanced binary tree with a height of 10 levels.

We then ran our test on the 180 spatial query regions, calculating time taken to compute minimum, maximum and average attribute value for each attribute in the spatial query region. The query execution time for all the attributes was averaged for each query window to determine average response time. Average response time for each calculated statistic (Minimum, Maximum or Average) was trended against the spatial query windows of increasing region size. To avoid differences in query execution times due to caching of data pages, before each query execution all clean buffer pages were flushed to disk to test all queries with a cold buffer cache.

Graphs in Figure 5-4, Figure 5-5 and Figure 5-6 trend the average query execution time against an increasing query window size (calculated as a percentage of total area). The different trend lines compare average query execution time for each of 4 methods in our test. The experiment was repeated for each road level attribute and execution time results were averaged for all the attributes. As can be observed from the 3 graphs, although the time taken to compute averages was higher, overall trends in average query execution time for each method were similar for all three summary statistics (Maximum, Minimum or Average). An increase in window size reduces the selectivity of the spatial index. Hence, an increase in query response time was observed when growing spatial query window area. The worst query response time was observed when not using a spatial index since the query optimizer selected the appropriate index on the attribute column to retrieve records from the *fact* table. Default server spatial indexing performed better than our indexing approach due to the multi-cell representation for each spatial object. The multi-cell representation eliminated the dead space when representing spatial objects using Minimum Bounding Rectangles, as in the case of *GP-Tree*, resulting in faster response times when identifying the relevant road segments in the query window.

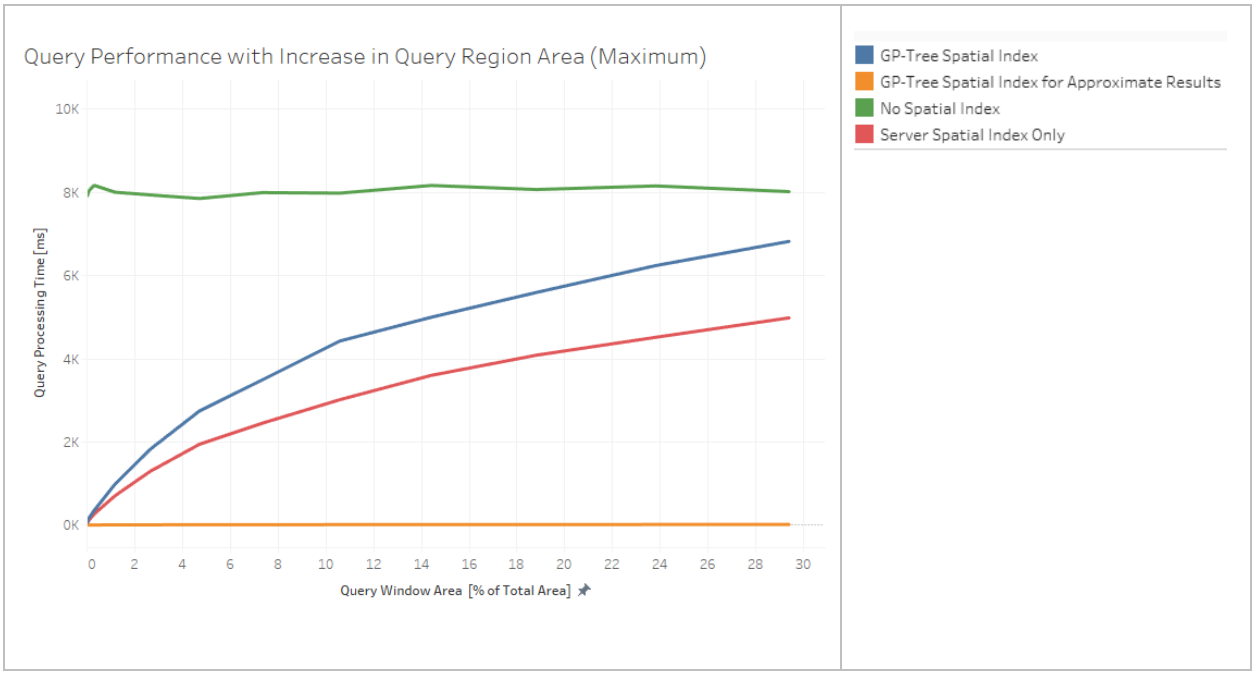

Figure 5-4 Query execution time (in millisecond) trended against increasing size of the spatial query window (% of total area), when calculating maximum attribute value for all segments in the query window. Results are plotted for 4 different methods: 1. *GP-Tree* as a spatial index (blue) 2. *GP-Tree* to approximate the maximum attribute value (orange) 3. Maximum value computation with no spatial index (green) 4. Default server spatial index (red).

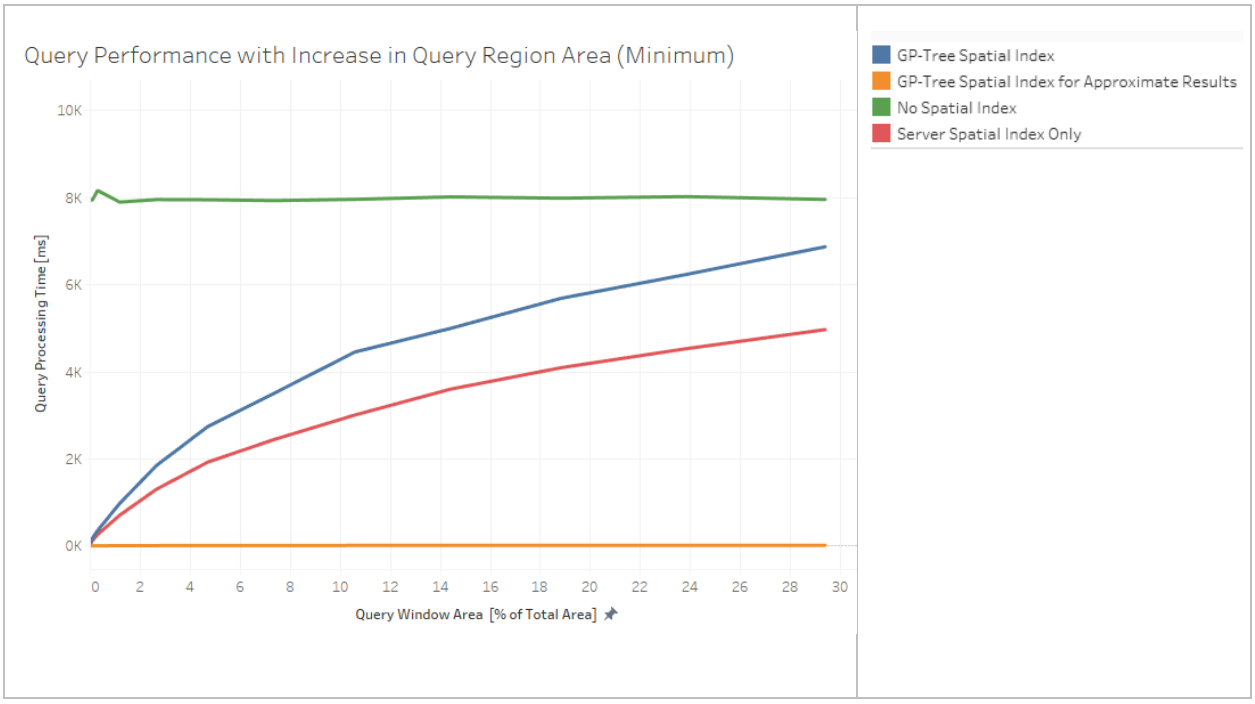

Figure 5-5 Query execution time (in millisecond) trended against increasing size of the spatial query window (% of total area), when calculating minimum attribute value for all segments in the query window. Results are plotted for 4 different methods: 1. *GP-Tree* as a spatial index (blue) 2. *GP-Tree* to approximate the minimum attribute value (orange) 3. Minimum value computation with no spatial index (green) 4. Default server spatial index (red).

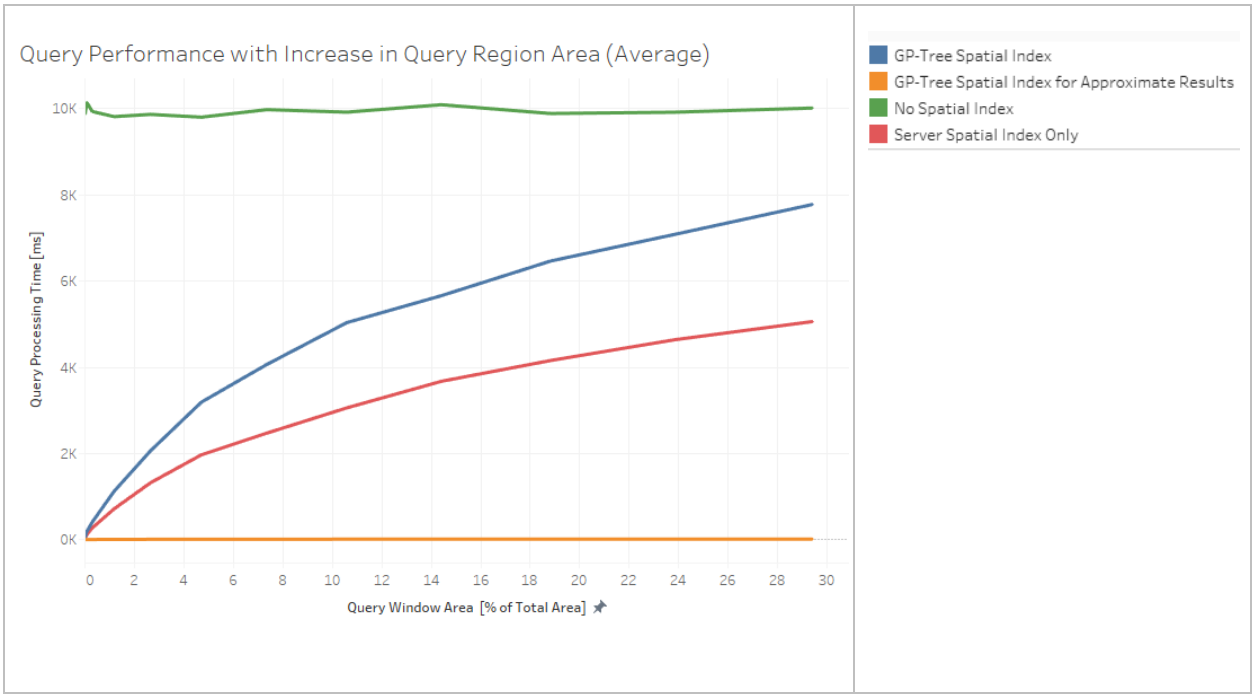

Figure 5-6 Query execution time (in millisecond) trended against increasing size of the spatial query window (% of total area), when calculating average attribute value for all segments in the query window. Results are plotted for 4 different methods: 1. *GP-Tree* as a spatial index (blue) 2. *GP-Tree* to approximate the average attribute value (orange) 3. Average value computation with no spatial index (green) 4. Default server spatial index (red).

Further, the query execution time when using *GP-Tree* for approximate statistic significantly outperformed the other methods. The approximation approach did not retrieve any records from the *fact* table, and only accessed the summary statistics calculated for each node of *GP-Tree*. For a tree structure with height h, the number of records in the tree could be computed as $2^h - 1$. Hence for all spatial queries, *GP-Tree* index reduced the search space from O(N) to O(2^h), where N is the number of entries in the *fact* table and h is the number of levels of *GP-Tree*.

The average value computation required largest number of records to be scanned from the *GP-Tree* based summary table. Since the size of our histogram was proportional to $logN$, the worst-case complexity for summary value calculation was proportional to $O(2^h logN) < N$. Using this approximation strategy thus, dramatically improved the performance of the spatial queries when computing summary statistics.

5.4 Evaluation of Approximation Results

We assessed the quality of our approximation computation method by reviewing the correlation between the actual and predicted values, and the mean absolute error for each attribute. For each attribute we

repeated our tests for increasing threshold value for region overlap, τ: 0%, 5%, 10%. A plot of actual versus approximations was plotted and the goodness-of-fit measure (R^2 value) calculated using linear regression.

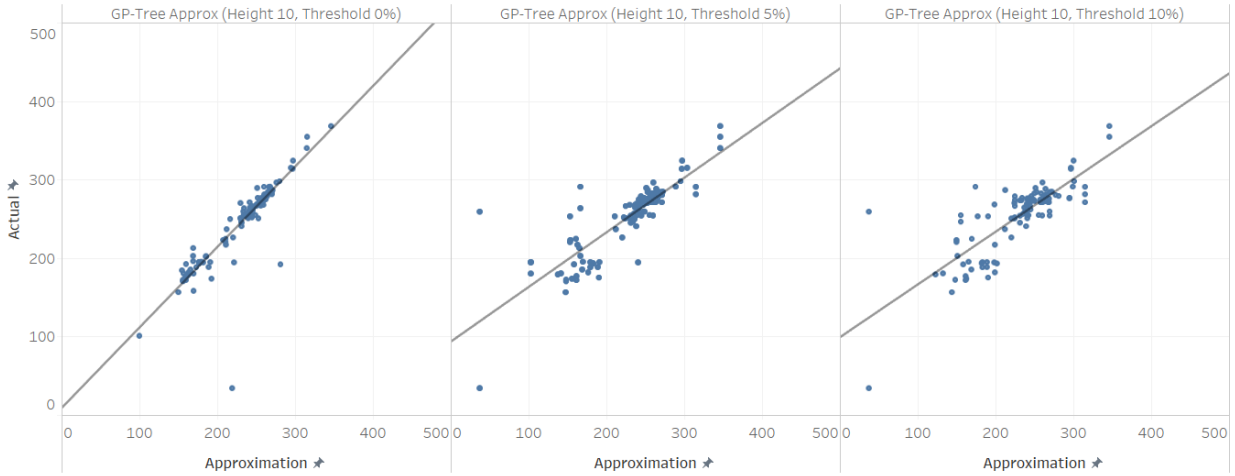

Figure 5-7 Linear regression plot for Engine Fuel Rate (Actual vs Approximation). The graphs are plotted in increasing value of threshold used to filter regions within the query window of *GP-Tree*. The chart on the left considers values for all regions satisfying the query constraints when computing average for values within the search window. Observed R^2 values in increasing order of threshold for overlap are 0.809, 0.714 and 0.647.

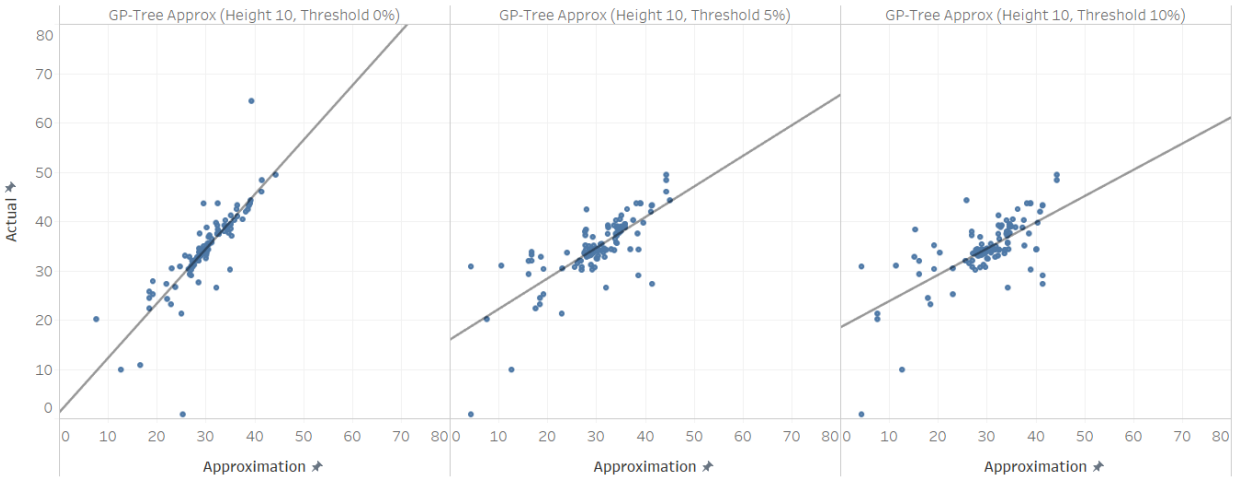

Figure 5-8 Linear regression plot for Speed (Actual vs Approximation). The graphs are plotted in increasing value of threshold used to filter regions within the query window of *GP-Tree*. The chart on the left considers values for all regions satisfying the query constraints when computing average for values within the search window. Observed R^2 values in increasing order of threshold for overlap are 0.826, 0.484 and 0.422.

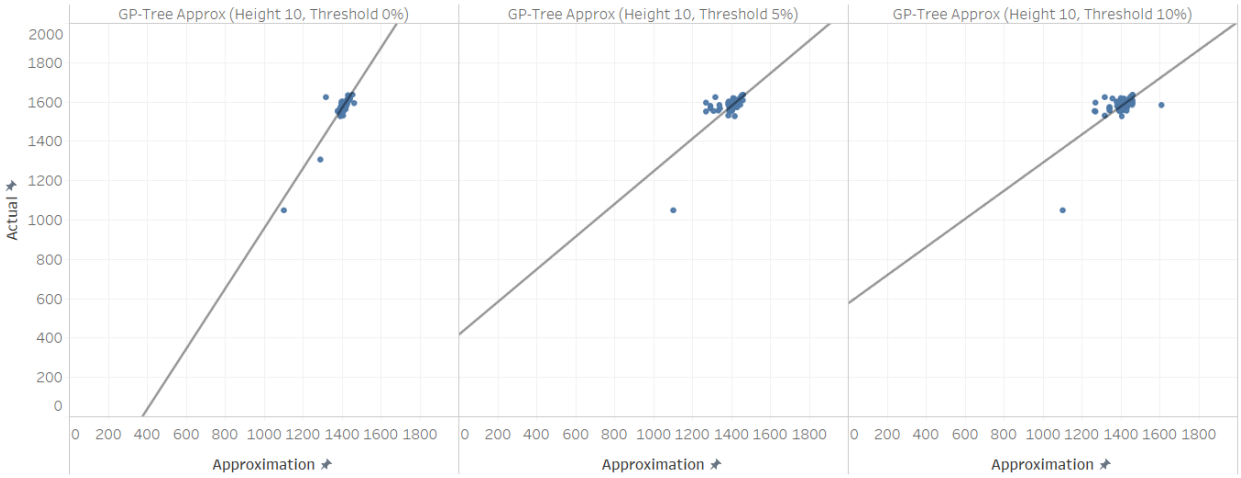

Figure 5-9 Linear regression plot for Engine RPM (Actual vs Approximation). The graphs are plotted in increasing value of threshold used to filter regions within the query window of *GP-Tree*. The chart on the left considers values for all regions satisfying the query constraints when computing average for values within the search window. Observed R^2 values in increasing order of threshold for overlap are 0.690, 0.580 and 0.460.

We observed moderate to high correlation between the actual and approximate values. The linear regression plots for Engine Fuel Rate can be observed in Figure 5-7, for Engine Speed can be observed in Figure 5-8 and for Engine RPM can be observed in Figure 5-9. Highest correlation was observed when all regions within the query window were considered (0% threshold). The observed R^2 with the 0% threshold was thus 0.809 for Engine Fuel Rate, 0.826 for speed and 0.690 for Engine RPM.

Additionally, for each attribute we observed the trend for average error in value of approximation, as a function of query window size. This was done by calculating the Mean Absolute Error (MAE) for the results of each attribute and window-size pair as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|x_{approx} - x_{actual}\right|$$

Where x_{approx} is the approximate value and $x_{actual}$is the actual query result value. A single MAE was computed for all results of queries with the same spatial area. The trend for MAE as a function of query window size is shown for the Engine Fuel Rate attribute in Figure 5-10, Speed in Figure 5-11 and Engine RPM in Figure 5-12. The MAE for all three attributes are higher for queries with smaller window sizes (% of total area < 5) and the results for larger query window sizes stabilize with growth in window size.

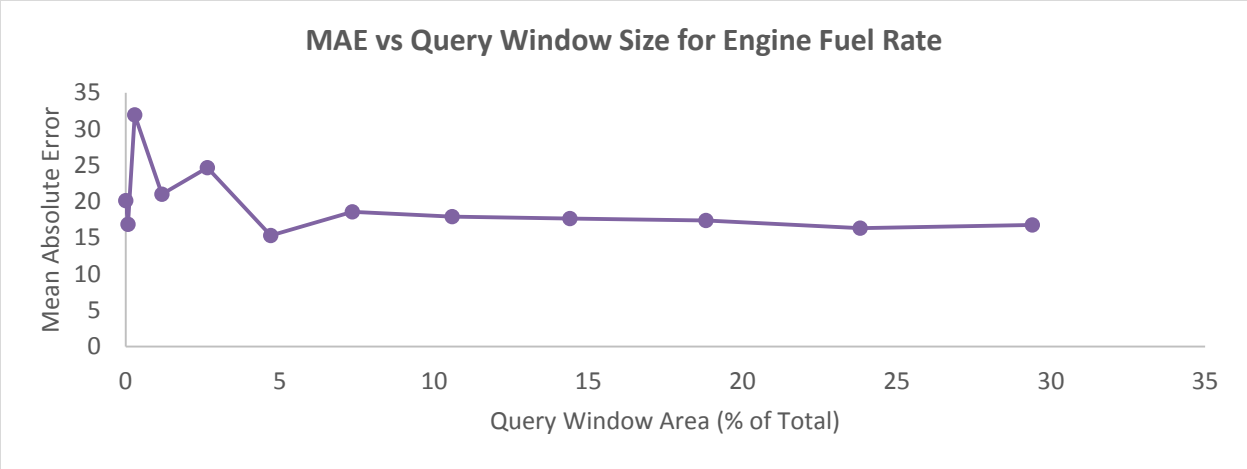

Figure 5-10 Mean Absolute Error value as a function of increasing query window area (Engine Fuel Rate).

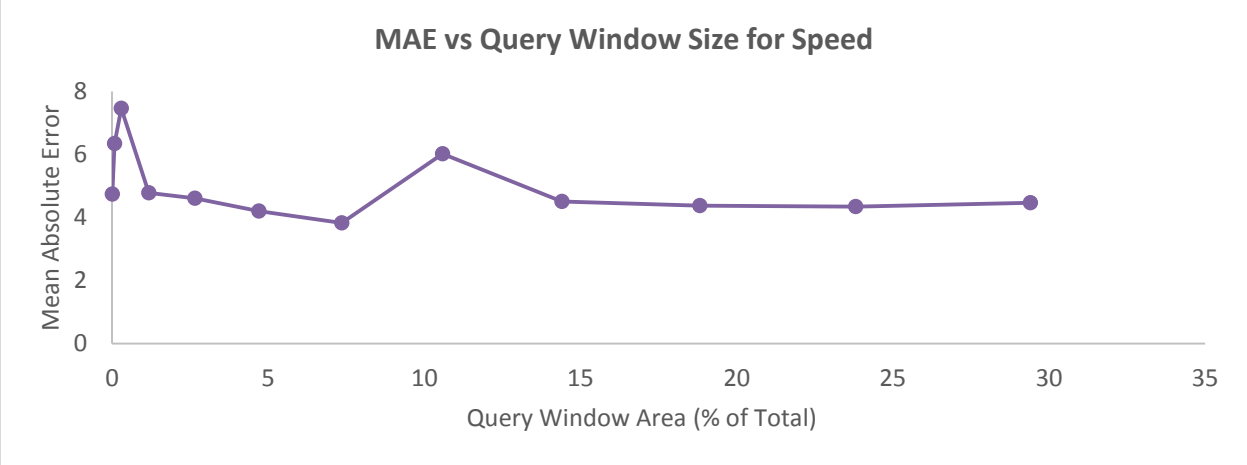

Figure 5-11 Mean Absolute Error value as a function of increasing query window area (Speed).

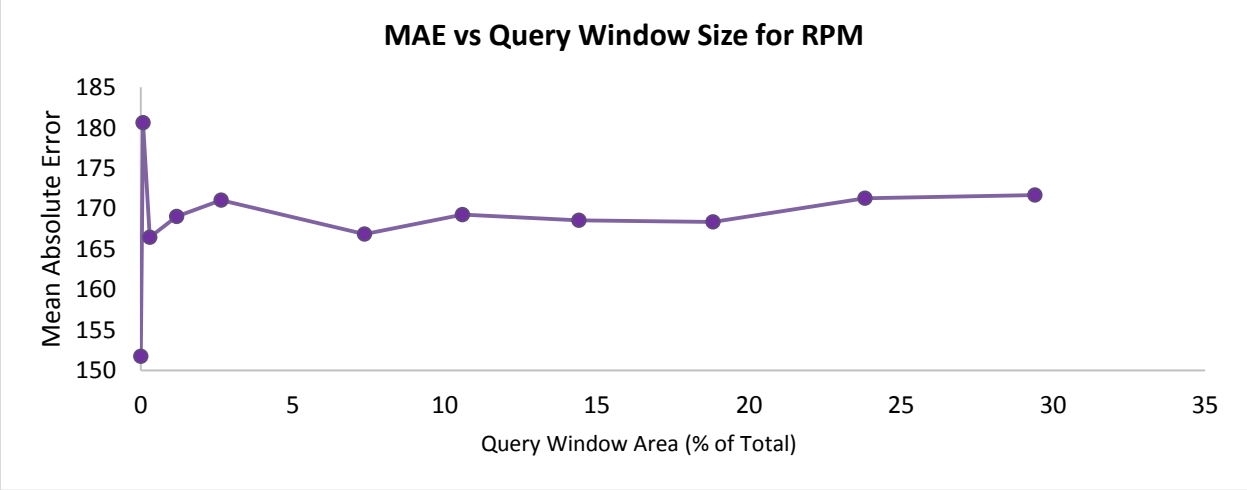

Figure 5-12 Mean Absolute Error value as a function of increasing query window area (Engine RPM).

GP-Tree approximation results for Minimum and Maximum value computation had poor correlation and high error rates. The use of *GP-Tree* for calculating minimum and maximum values is hence not recommended.

5.5 Error Bounds for approximations Using *GP-Tree*

The *GP-Tree* stores the Minimum Bounding Rectangle (MBR) for each node in the tree, where a node is a partition of the road network graph containing multiple road segments. Our search algorithm uses a top-down approach to scan the search space and identify relevant nodes that satisfy the spatial query. Further, we use a frequency histogram to model the distribution of attribute values for road segments represented by a tree node, and estimate the mean using the midpoint of each bin as representative of all values within the bin. To evaluate the quality of our approximation, we determine the error bounds for results when using a *GP-Tree* for approximation of average attribute values for roads within the query window. This is done separately for two cases: 1. Spatial query window completely overlaps MBR for all *relevant* nodes 2. Spatial query window partially overlaps MBR for certain nodes.

In the case when the MBR for all nodes is entirely within the search window, we can provide an upper bound and lower bound for the approximation result calculated by a *GP-Tree*. The frequency histogram for each node of a *GP-Tree* contains a count of values that fall within the limits of the bin, hence all values accounted for within a single bin have values in the range [lower limit, upper limit). In the worst case, all sample points within the bin have the bin limit as their true value. In the case where the bin values have the lower limit as their true value, the true average value is calculated as:

$$Average_{LowerLimit} = \frac{\sum_{k=1}^{bincount} (From\,Value[Bin_k]) * Frequency[Bin_k]}{\sum_{k=1}^{bincount} Frequency[Bin_k]}$$

Where *FromValue* is the lower limit of the bin and *Frequency* is the count of entries within a bin. Thus, the error between the expected value and the actual value is then calculated as

$$Error = Average - Average_{LowerLimit}$$

$$Error = \frac{\sum_{k=1}^{bincount} \left(From\,Value[Bin_k] + \frac{Interval[Bin_k]}{2}\right) * Frequency[Bin_k]}{\sum_{k=1}^{bincount} Frequency[Bin_k]} - \frac{\sum_{k=1}^{bincount} (From\,Value[Bin_k]) * Frequency[Bin_k]}{\sum_{k=1}^{bincount} Frequency[Bin_k]}$$

$$Error = \frac{\sum_{k=1}^{bincount} \left(\frac{Interval[Bin_k]}{2}\right) * Frequency[Bin_k]}{\sum_{k=1}^{bincount} Frequency[Bin_k]}$$

$$Error = \left(\frac{Interval[Bin]}{2}\right)$$

Where $Interval[Bin_k]$ is the width of the k^{th} bin. Since we use equal width bins, the error is limited by a value equal to half the bin interval. The upper limit of the bin is not included in the bin interval hence we can only conclude that the upper bound for error is strictly lower than $\frac{Interval[Bin]}{2}$. Hence, for queries where the query window overlaps regions of *GP-Tree*, we can determine the error bound as:

$$|Error| \leq \left(\frac{Interval[Bin]}{2}\right)$$

For cases where the query window does not completely overlap the MBR of the spatial regions *relevant* to the query, that is, the query window partially intersects the MBRs for certain nodes *relevant* to the query, the error cannot be easily determined.

5.6 Summary

In this chapter we summarized the results of our data warehouse model by outlining advantages of our data warehousing approach. We contrasted our data model with previous approach for trajectory data warehousing and demonstrated trip-level and road-level queries possible using the modifications proposed for our data warehouse design. We then demonstrated the considerable performance gains from using our spatial indexing structure, *GP-Tree*, for approximating summary statistics. Although *GP-Tree* did not perform better than a multi-cell spatial index for spatial queries, it provided considerable performance benefits when the *GP-Tree* hierarchy was used to create a summary table for spatial queries.

We observed strong correlation between actual and approximation calculated using *GP-Tree*-based summary table for computation of average values. Further we provided error bounds for results for average values computed using our approximation approach and observed that the Mean Absolute Error stabilized for larger query window sizes. Finally, we determined an error bound for the values computed using of our *GP-Tree* based approximation method.

Chapter 6. Conclusion

Real-world driving information can be leveraged to assess road conditions or evaluate operator driving behaviour. In this thesis, a large volume of telemetry information for haul trucks operating in a surface mining environment was used to enable exploratory data analysis of haul road network performance. The information collected consisted of several sensor recordings for the haul trucks along with movement information. Previous data manipulation strategies proposed for data reduction of vehicle movement data were leveraged to summarize information for a large volume of object movement data.

First, we proposed a modified trajectory data warehouse design for exploratory data analysis of object movement in constrained environment, such as, road networks using vehicle diagnostic values as attributes. This data warehouse was used to analyze historical performance of road network in a surface mining environment. The data warehouse design enabled trip-level and road-level information for each haul trajectory.

One contribution of this thesis is the creation of a generic spatial hierarchy for our road network structure, since an implicit or explicit hierarchy was unavailable. We leveraged this spatial hierarchy to create a summary table for fast computation of summary statistics for spatial queries. The summarized dataset was used to calculate an approximation for average, minimum and maximum attribute values for objects within a spatial query window. Our approximation method performed considerably faster when compared with the use of spatial indexes to retrieve records from the underlying data warehouse. Further, results of our approximation approach showed high correlation with the actual values when used for average value computation. Finally, we provided a method to compute the error bounds for the approximation result.

Future work would include extending the use of our spatial hierarchy for temporal and spatio-temporal queries.

Bibliography

Atluri, G., Karpatne, A., & Kumar, V. (2017). Spatio-Temporal Data Mining: A Survey of Problems and Methods, *1*(1), 1–37. Retrieved from http://arxiv.org/abs/1711.04710

Botea, V., Mallett, D., Nascimento, M. A., & Sander, J. (2007). PIST: An Efficient and Practical Indexing Technique for Historical Spatio-Temporal Point Data. *GeoInformatica*, *12*(2), 143–168. https://doi.org/10.1007/s10707-007-0030-3

Campora, S., Fernandes De Macedo, J. A., & Spinsanti, L. (2011). St-Toolkit: A Framework for Trajectory Data Warehousing. In *AGILE*. Retrieved from http://www.simonecampora.com/blog/wp-content/uploads/2009/05/sp_110.pdf

Cat | Road Analysis Control | Caterpillar. (n.d.). Retrieved February 4, 2018, from https://www.cat.com/en_IN/support/operations/technology/fleet-management-solutions/road-analysis-control.html

Cudre-Mauroux, P., Wu, E., & Madden, S. (2010). TrajStore: An adaptive storage system for very large trajectory data sets. *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, 109–120. https://doi.org/10.1109/ICDE.2010.5447829

Delling, D., Goldberg, A. V., Razenshteyn, I. & Werneck, R. F. (2011). Graph Partitioning with Natural Cuts. In *Proceedings of the 35th Int. Parallel & Distributed Processing Symposium (IPDPS)*, 1135–1146.

Digitizing Map Data — QGIS Tutorials and Tips. (n.d.). Retrieved February 24, 2017, from http://www.qgistutorials.com/en/docs/digitizing_basics.html

Evolution of Mining Equipment in the Oil Sands. (n.d.). Retrieved February 4, 2018, from http://www.oilsandsmagazine.com/technical/mining/surface-mining/equipment

Ferreira, J. C., de Almeida, J., & da Silva, A. R. (2015). The Impact of Driving Styles on Fuel Consumption: A Data-Warehouse-and-Data-Mining-Based Discovery Process. *IEEE Transactions on Intelligent Transportation Systems*, *16*(5), 2653–2662. https://doi.org/10.1109/TITS.2015.2414663

Ferreira, N., Klosowski, J. T., Scheidegger, C. E., & Silva, C. T. (2013). Vector field k-means: Clustering trajectories by fitting multiple vector fields. *Computer Graphics Forum*, *32*(3 PART2), 201–210. https://doi.org/10.1111/cgf.12107

Freedman, D., & Diaconis, P. (1981). On the histogram as a density estimator:L2theory. *Zeitschrift Für Wahrscheinlichkeitstheorie Und Verwandte Gebiete*, *57*(4), 453–476. https://doi.org/10.1007/BF01025868

Güting, R. H., Behr, T., & Düntgen, C. (2010). SECONDO: A Platform for Moving Objects Database Research and for Publishing and Integrating Research Implementations. *IEEE Data Eng. Bull.*, *33*(2), 56–63. Retrieved from http://dna.fernuni-hagen.de/papers/PaperPlugins.pdf

Guttman, A. (1984). R-trees. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data - SIGMOD '84* (p. 47). New York, New York, USA: ACM Press. https://doi.org/10.1145/602259.602266

Han, J., Kamber, M., & Pei, J. (2006). *Data mining: concepts and techniques* (3rd ed.). Morgan Kaufmann.

Haul Road Maintenance: Improving Tire & Mining Truck Life. (n.d.). Retrieved January 20, 2018, from https://www.cat.com/en_US/by-industry/mining/articles/improving-tire-and-mining-truck-life.html

Jakobsen, K., Mouritsen, S. C. H., & Torp, K. (2013). Evaluating eco-driving advice using GPS/CANBus data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - SIGSPATIAL'13* (pp. 44–53). New York, New York, USA: ACM Press. https://doi.org/10.1145/2525314.2525358

Jenhani, F., & Akaichi, J. (2014). Mobile Data Warehousing: A Survey. *International Journal of Innovation and Scientific Research ISSN*, *10*(2), 2351–8014. Retrieved from http://www.ijisr.issr-journals.org/

Karypis, G., & Kumar, V. (1998). A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, *20*(1), 359–392. https://doi.org/10.1137/S1064827595287997

Kimball, R., & Ross, M. (2002). *The Data Warehouse Toolkit* (Third). Wiley. Retrieved from http://www.essai.rnu.tn/Ebook/Informatique/The Data Warehouse Toolkit, 3rd Edition.pdf

Lee, Jae-gil;Han, jiawei; Li, Xiaolei; Cheng, H., Lee, J.-G., Han, J., Li, X., & Cheng, H. (2011). Mining Discriminative Patterns for Classifying Trajectories on Road Networks. *IEEE Transactions on Knowledge and Data Engineering*, *23*(5), 713–726. https://doi.org/10.1109/TKDE.2010.153

Lee, J.-G., Han, J., & Li, X. (2008). Trajectory Outlier Detection: A Partition-and-Detect Framework. In *2008 IEEE 24th International Conference on Data Engineering* (pp. 140–149). IEEE. https://doi.org/10.1109/ICDE.2008.4497422

Lee, W., & Krumm, J. (2011). Trajectory Preprocessing. In Y. Zheng & X. Zhou (Eds.), *Computing with Spatial Trajectories* (1st ed., pp. 3–33).

Leonardi, L., Orlando, S., Raffaetà, A., Roncato, A., Silvestri, C., Andrienko, G., & Andrienko, N. (2014). A general framework for trajectory data warehousing and visual OLAP. *GeoInformatica*, *18*(2), 273–312. https://doi.org/10.1007/s10707-013-0181-3

Li, W. (2018). Sandwych.MapMatchingKit. Retrieved from https://github.com/oldrev/mapmatchingkit

Moreno, F., & Arango, F. (2010). A Conceptual Trajectory Multidimensional Model: An Application to Public Transportation. *Dyna*, (166), 142–149. Retrieved from http://www.scielo.org.co/pdf/dyna/v78n166/a17v78n166.pdf

Nardini, F. M., Orlando, S., Perego, R., Raffaetà, A., Renso, C., & Silvestri, C. (2018). Analysing Trajectories of Mobile Users: From Data Warehouses to Recommender Systems (pp. 407–421). https://doi.org/10.1007/978-3-319-61893-7_24

Newson, P., & Krumm, J. (2009). Hidden Markov map matching through noise and sparseness. *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS '09*, 336–343. https://doi.org/10.1145/1653771.1653818

Papadias, D., Yufei Tao, Kanis, P., & Jun Zhang. (2002). Indexing spatio-temporal data warehouses. In

Proceedings 18th International Conference on Data Engineering (pp. 166–175). IEEE Comput. Soc. https://doi.org/10.1109/ICDE.2002.994706

Pelekis, N., Frentzos, E., Giatrakos, N., & Theodoridis, Y. (2015). HERMES: A Trajectory DB Engine for Mobility-Centric Applications. *International Journal of Knowledge-Based Organizations*, *5*(2), 19–41. https://doi.org/10.4018/ijkbo.2015040102

Quach, T., & Parsons, I. (2005). Increasing Usability of Information from Maintenance Telemetry Data. *Research Department Progress Report, Syncrude Canada Ltd.*, *34*(4).

Quddus, M. A., Ochieng, W. Y., Zhao, L., & Noland, R. B. (2003). A general map matching algorithm for transport telematics applications. *GPS Solutions*, *7*(3), 157–167. https://doi.org/10.1007/s10291-003-0069-z

Rasetic, S. (2005). *Trajectory Splitting Models for Efficient Spatiotemporal indexing*. University of Alberta.

Scott, D. W. (2009). Sturges' rule. *Wiley Interdisciplinary Reviews: Computational Statistics*, *1*(3), 303–306. https://doi.org/10.1002/wics.35

Sturges, H. A. (1926). The Choice of a Class Interval. *Journal of the American Statistical Association*, *21*(153), 65–66. https://doi.org/10.1080/01621459.1926.10502161

Sui, X., Nguyen, D., Burtscher, M. & Pingali, K. (2010). Parallel graph partitioning on multicore architectures. In *Proceedings of the 23rd international conference on Languages and compilers for parallel computing - LPC' 10* (246–260).

Wang, W., Yang, J., & Muntz, R. (1997). STING : A Statistical Information Grid Approach to Spatial Data Mining. In *Proceedings of 23rd International Conference on Very Large Data Bases* (pp. 1–10).

White, C. E., Bernstein, D., & Kornhauser, A. L. (2000). Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, *8*(1–6), 91–108. https://doi.org/10.1016/S0968-090X(00)00026-7

Xu, T., Zhang, X., Claramunt, C., & Li, X. (2018). TripCube : A Trip-oriented vehicle trajectory data indexing structure. *Computers, Environment and Urban Systems*, *67*, 21–28. https://doi.org/10.1016/j.compenvurbsys.2017.08.005

Zhao, X., Cheng, X., Zhou, J., Xu, Z., Dey, N., Ashour, A. S., & Satapathy, S. C. (2017). Advanced Topological Map Matching Algorithm Based on D–S Theory. *Arabian Journal for Science and Engineering*.