Spatio-Temporal Data Warehousing for Exploratory Analysis of Scientific Data

Zhao Jing

Abstract

In scientific fields, the amount of data has been increasing enormously owing to the development of information technologies, high performance computers, and high capacity storage. As one of the common data types of big data, *spatio-temporal data* has been widely applied in various domains such as mobile applications and scientific research. For instance, disaster simulations addressing a particular spatial area and calculate temporal variations in the field after disasters are conducted for the purpose of predictions, decision making, etc. Consequently, managing and analysis of large-scale scientific spaito-temporal data has been in great demand.

In this thesis, we study the *data warehousing* techniques that enabling massive data storage and data exploration for sophisticated analytic processing of large amount of data. Researches on data warehousing involve data modeling, basic operators like data join and aggregation, advanced operators support for exploratory analysis, etc. We focus on the above-mentioned aspects and mainly conduct the following three studies.

First, we propose a simulation data warehouse-approach for interactive analysis of massive simulation data. The objectives of this work include integrating different simulation data sets, as well as enabling exploratory analysis of multiple accumulated simulation data with high-speed response by data preprocessing. We develop a prototype system architecture consists of data storage based on a *multi-dimensional data cube*, as well as an analysis interface to enable basic operators such as drill-down and roll-up for interactive analysis. In addition, we show the usability of the proposed prototype system by a case example using disaster simulation data.

Second, we study advanced operators for exploratory analysis by data summarization techniques, i.e., *histograms*. One of the challenges of exploration-based analysis of spatio-temporal data is the semantic meanings of different granularities such as one day, one hour and one minute on the time dimension, which leads to the exhausting exploration by basic operators. Therefore, more advanced operators such as effective summarization and visualization that navigates users to find interesting knowledge are required.

We study the problem of constructing histograms (i.e., the *spatial V-optimal histogram*) that summarize the data distribution of a specific spatial area during a time interval. We propose exact and approximate algorithms, as well as a heuristic algorithm for efficient construction of hierarchical histograms. As scientific data is usually represented as 2-D

(or 3-D) array structure, *array*-based representation is more appropriate for the representation. The proposed methods are implemented on the state-of-the-art array DBMS, SciDB, which supports efficient scientific computing and analysis of spatio-temporal array data. In addition, we conduct extensive experiments on massive simulation data, real taxi data as well as synthetic data with different distributions, to verify the effectiveness and the efficiency of the proposed methods.

Finally, we study the *difference* analysis of spaito-temporal data on a data warehouse, to detect temporal variations as well as differences between observation datasets with different parameters or conditions. We propose a general framework for constructing histograms on SciDB, as well as both optimal and heuristic histogram construction algorithms based on the structure of quadtree. We conduct experiments on massive simulation data to evaluate the performance of the algorithms, with a case study of the proposed difference operator.

In general, we provide data warehousing techniques including system prototypes, data modeling and advance operators to manage and analyze large-scale spatio-temporal scientific data. In order to enable efficient data visualization and effective data analysis, our proposed methods involve data preprocessing, approximate and heuristic data summarization algorithms based on different structured histograms. Last but not the least, we have conducted extensive experimental evaluations on the performance of our proposed solutions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Background

As big data attracts attention in a variety of fields, research on data exploration for analyzing large-scale scientific data has gained popularity. As one of the common data types of big data, *spatio-temporal data* has been widely applied in various domains such as mobile applications and scientific research [1]. For instance, in scientific fields, simulations are conducted for the purpose of predictions, decision making, etc. As one of the typical simulations, disaster simulations like human evacuation simulation are conducted for effective humanitarian relief and disaster management [2]. Such kind of disaster simulations generates large-scale spatio-temporal data, which contains spatial and temporal information of evacuees on the target area during a period of time.

Analysis of spatio-temporal data can achieve various objectives, such as discovery of interesting mobility patterns of users and shelter location suggestion for disaster management [2, 3]. *User-driven analysis* is designed for traditional database systems, which assume that users are aware of exactly what they are looking for and have good knowledge of the contents of the database. However, as big data has made great success in a variety of fields, it leads to another type of analysis, *data-driven analysis*, in which it is not necessary for users to have prior knowledge about the target dataset or be aware of the type of queries they want to pose. Users often explore massive data by navigation or

visualization tools to find interesting patterns. In the following, we consider three of the typical types of queries conducted by analyzers on the disaster simulation data:

- **Aggregate query:** "Calculate the number of people in each cell of a rough mesh of 1km ×1km by aggregating evacuee data in the specified area after the occurrence of the earthquake."
- **Summary-based query:** "Return the overall spatial distribution of evacuees during the first hour after the earthquake occurs."
- **Difference-based query:** "Return the major change in the distribution of evacuees between the first hour and the second hour after the earthquake occurs."

The aggregate query is a common used operations for exploration of scientific data stored in data warehouses [4, 5]. The aggregation functionality of data warehouses is designed to satisfy the computation of numeric values stored with high response speed. More advanced queries such as the summary-based query and difference based query are used to understand the overall distribution and variation trend of the target data, to navigate users to find interesting knowledge [6–9].

As a result, research on *data exploration* for sophisticated analytic processing of large amount of data has gained popularity [10].

We indicate three main challenges of exploration-based analysis of scientific spatio-temporal data as follows.

1. Scientific data like simulation data are generated by many simulations with different conditions and parameters, which makes the data source diverse and data volume large. Consequently, integrated and interactive analysis of such multi-source data is in demand.
2. Spatio-temporal data has characteristics on spatial and temporal scales like different granularites of time dimension, i.e., one day, one hour and one minute. Instead of exhausting exploration by drill down and roll up operators on the data warehouse,

more advanced operators such as effective summarization and visualization that navigates users to find interesting knowledge are required.

3. As advanced operators such as summarizing the target data and detecting remarkable changes from a large amount of data are time consuming, development of efficient methods cooperating with modern database system technology is also important.

In order to manage and analyze massive spatio-temporal data, we leverage the *data warehousing* techniques for massive data storage and data exploration of scientific data. A data warehouse is a system to realize efficient interactive analysis by data preprocessing and query processing by operators like user-defined functions. Unlike a conventional database that is updated by adding or deleting data, a data warehouse requires data reorganization in an appropriate form for the analysis of accumulated data, and the data in the data warehouse must be configured in such a manner that interactive analysis can be realized from various perspectives. This requirement was pointed out in 1993 by E. F. Codd, a proponent of relational data models. This type of analysis, called *OLAP* (On-Line Analytical Processing) [11], requires a multi-dimensional-structured database. Data warehouse (DWH) [12–14] was proposed at approximately the same time. It shares the motivation and purpose with OLAP. However, DWH is more system-oriented.

1.2 Related Work

Researches on DWH involve data storage (or data modeling) [15–19], data exploration, which contains basic operators like data aggregation and data join [4, 5], advanced operators [6–9] support for sophisticated analysis, etc. [16, 17] were the first ones to propose a framework for spatial data warehouses based on the star-schema [20], in which the cube dimensions can be both spatial and non-spatial, while the measures are regions in space, in addition to numerical data. Other data models such as a chunk-based computation technique for the efficient organization of large multidimensional arrays is developed in [15]. Further, efforts to expand a data model [18] to incorporate the semantics of a geographical space or to introduce a more flexible space division method [19] have been made.

For efficient operations on spatial data warehouses, a spatial database technology was developed and has been realized as an implementation technique of spatial index [4, 5].

In [6–9], advanced operators were proposed for relational data exploration. The aim of the operators proposed in [6–8] was to determine and summarize exceptions without excessive manual exploration, while [9] proposed a novel drill-down operator to discover the top-k "interesting" groups of tuples based on a given score function.

In this thesis, we focus on the data exploration on massive spatio-temporal data using DWH techniques. In order to overcome the above-mentioned challenges of exploration-based analysis of spatio-temporal data, we mainly conduct three studies, as described in the following section.

1.3 Research Objectives and Contributions

In the first study, we aim to integrate different simulation data sets and enable exploratory analysis of multiple accumulated simulation data. As different data sources have different attributes, an integrated data model holding those data is requred. Moreover, since the amount of generated simulation data is increasing, technical development for the effective use of big simulation data is strongly required. In particular, support for exploratory analysis, which is characteristic to science, and support for instantaneous interactive responses to analysis requests for big data are desired.

In the second study, we consider advanced operators for exploratory analysis of spatio-temporal data by data summarization techniques. As the challenges described in the previous section, effective summarization and visualization that navigates users to find interesting knowledge are required. For instance, a succinct summary of the data distribution of a specific spatial area during a time interval is considered to be useful. We summarize the target data based on the notion of histograms [21–23, 48] in the database research area. Different from the main focus of those work, which is to improve the accuracy of estimations, our work focuses on both usability in visual analytics and accuracy of histograms. However, building histograms is often time consuming, especially for high

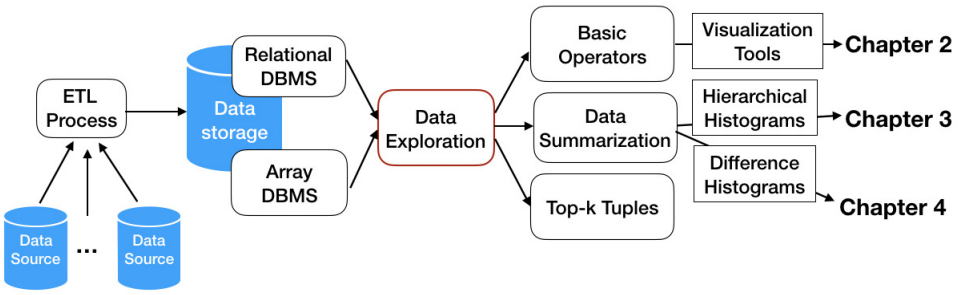


FIGURE 1.1: Research Diagram

accuracy demand even for a 2-dimensional case. Therefore, it is necessary to develop efficient algorithms by effectively using the latest database system technology.

In the last study, we study the *difference* analysis of spaito-temporal data on the data warehouse, to detect temporal variations as well as differences between observation datasets with different parameters or conditions. Various types of difference operators are possible, depending on the properties or application purposes of the target data. In this work, we define a difference histogram based on the notion of histograms as well. Due to the challenges in detecting any remarkable changes within a large amount of data, efficient histogram construction algorithms as well as seamless cooperation with modern data management system are also in demand.

Our contributions are summarized as follows corresponding to Figure 1.1:

- In Chapter 2, we develop a general prototype system architecture for exploratory and integrated analysis of different simulation data sources. The data modeling module in the system architecture is based on a relational data model, i.e., *multi-dimensional data cube* for integrated analysis of multiple data sources. The basic operators such as drill-down and roll-up for exploratory analysis are enabled by visualization techniques.

- In Chapter 3 and Chapter 4, we define two advanced operators based on histograms, one of which is a general hierarchical histogram, defined as the *spatial V-optimal histogram* that summarizes the data distribution of a specific spatial area during a

time interval, and the other one is a difference histogram based on the quadtree structure to detect spatial and temporal variations of the target data.

- For the spatial V-optimal histogram, we propose exact and approximate algorithms, as well as a heuristic algorithm for efficient construction of hierarchical histograms. The proposed methods are implemented on the state-of-the-art array DBMS, SciDB, which supports efficient scientific computing and analysis of spatio-temporal array data.

- For the difference histogram, we propose a general framework for constructing histograms in a bottom-up style on SciDB. Both optimal and heuristic histogram construction algorithms are proposed based on the structure of quadtree.

- We conduct extensive experiments on massive spatio-temporal data to verify the performance of the proposed methods.

- We also show the usability of the proposed prototype system and histogram construction approaches by case studies using disaster simulation data.

In general, we study the data warehousing techniques that enabling exploratory analysis of massive scientific data. Our research considers a relational data model for integrated analysis, as well as an array-based data model for efficienty computing such as data join and data aggregation. Moreover, basic operators and advanced operators are realized on the system to support interactive analysis of spatio-temporal data.

1.4 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we introduce the proposed simulation data warehouse for integrated analysis of disaster information. Then, as an advanced operators on data warehouses, hierarchical histograms for exloratory analysis of spatio-temporal data are studied in Chapter 3. In Chapter 4, a difference operator on the array DBMS is studied. Finally, Chapter 5 concludes the thesis.

Chapter 2

Simulation Data Warehouse for Integrated Analysis of Disaster Information

2.1 Introduction

In scientific fields, the amount of data has been increasing enormously owing to the development of information technologies, high performance computers, and high capacity storage. The science of managing large data is frequently called *data science* and has attracted attention as a new direction of science. In this study, we focus on simulations conducted in different fields. A massive amount of data is generated in these simulations and *data-intensive computing*, which aims to use these data, now receives attention as the "fourth paradigm" of science [24]. Many simulations are conducted to analyze earthquakes, tsunamis, and other disasters [25, 26]. Because the simulations are performed many times using different conditions and parameters, the data volume becomes large. The simulation results are frequently compared or integrated with other simulation data for advanced analysis, which could also be utilized in future studies. The idea of managing the simulation data corresponds to the above-mentioned fourth paradigm.

The majority of disaster simulations have a simulation target of a particular spatial area and calculate temporal variations in the area after the disaster. For example, a tsunami simulation calculates the flood depth at each site by time period. An evacuation simulation calculates the location where evacuees can be found by time period. The captured simulation data links the temporal information such as the time of occurrence of a disaster to the spatial information such as the coordinate of an evacuee's location. In this work, we focus on such kind of simulations, *spatio-temporal simulations*, which addressing spatial and temporal information.

Existing research on disaster simulations focuses on the estimations of simulation models, such as tsunami models, evacuees' mobility models, and fire spread models [25–27]. Such estimated simulation models are used for disaster management, human relief, urban computing, etc. For instance, analyzing the evacuees' mobility patterns enables risk analysis and effective human relief during disasters like earthquakes and fires [2]. However, to the best of our knowledge, there is no research in the literature exploits database techniques to manage such large-scale simulation data and support for the analysis process.

In the present study, we focus on the technical development of a data warehouse where a large amount of diverse simulation data is managed and analyzed. The amount of data produced from today's simulations is increasing and hence, technical development for the effective use of big simulation data is strongly required. In particular, support for exploratory analysis, which is characteristic to science, and support for instantaneous interactive responses to analysis requests for big data are desired.

Let us consider *damage status analysis* as an example of disaster simulation data analysis. For the analysis, it is assumed that an analyzer specifies the scale and location of an earthquake and integrates multiple sets of earthquake simulation data. Suppose that we have a request for the integrated analysis of earthquake intensity in the center of Tokyo, damage from a tsunami, and evacuees' status in the event of an earthquake at the seismic center x offshore Chiba Prefecture, with magnitude y that occurs at noon on a weekday in April. To respond to this request, it is necessary to not only identify and extract the simulation data from the database that matches those conditions, but also show the correlations of different data sources.

Analysis, however, is usually made in an *exploratory* manner. Analysis is not made initially on the details of the simulation data directly. Data are observed from multiple viewpoints to identify the important data and then, for a more detailed analysis, the data are further investigated. For instance, the query "Calculate the number of people in each cell of a rough mesh of 1km × 1km by aggregating evacuee data in the specified area after the occurrence of the earthquake" can be instantaneously executed, if the functionality of grasping the people distribution in each cell is available by techniques like visualization, which is useful for analysis.

With the above-mentioned background, the present study proposes a simulation data warehouse-approach for the interactive analysis of large simulation data and describes a method of realizing the data warehouse. An objective of this study is to integrate different simulation data sets and enable exploratory analysis of multiple accumulated simulation data with high-speed response by data preprocessing. Further, the developed prototype system architecture and a case example of its use are explained.

The rest of this chapter is organized as follows. In Section 2.2, the data warehouse technology that is used as a base for the present study is explained. Section 2.3 is devoted to the architecture of a prototype system and a realization method for the system. In Section 2.4, the development approach for the simulation data warehouse for the target simulation data is provided. Case examples of queries and visualization are presented in Section 2.5. In Section 2.6, related studies are introduced. Finally, the conclusion and discussions of future work identified in the development of the prototype are explained in Section 2.7

2.2 Data Warehouse Technology

Unlike a conventional database that is updated by adding or deleting data, a data warehouse requires data reorganization in an appropriate form for the analysis of accumulated data. Moreover, the data stored in the data warehouse must be configured in such a manner that interactive analysis can be realized from various perspectives.

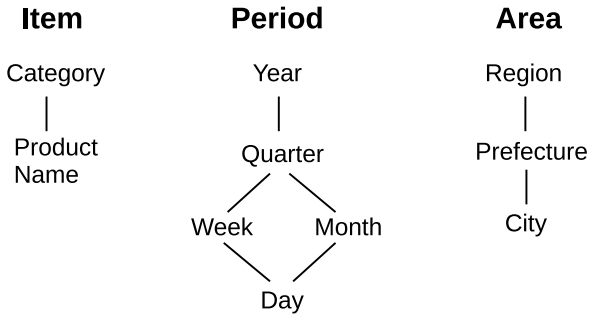


FIGURE 2.1: Example of Concept Hierarchy

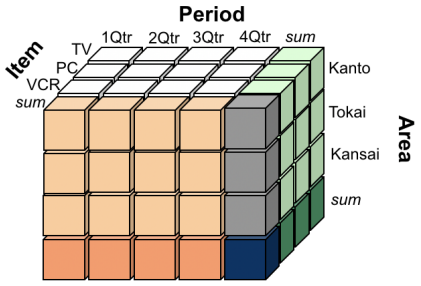


FIGURE 2.2: Example of Data Cube

One of the basic concepts often used for data warehouse is *concept hierarchy*. A single concept hierarchy is assigned to each domain. For example, the concept hierarchies presented in Figure 2.1 are used in a data warehouse for collecting and analyzing the sales of a company that sells electric appliances throughout the country. For "Item", there is a hierarchy of category and product names. For "Period", there is a hierarchy of year, quarter, week, month, and day. For "Area", there is a hierarchy of region, prefecture, and city. For example, an analysis command such as "Analyze the data of total sales amount of LCD televisions in the Prefectures in 2014, and aggregate the total sales amount by various combinations of prefecture and month." can be made based on the concept hierarchy. A variety of analyses can be executed by changing the levels and combinations of the concept hierarchies in a flexible manner or using various calculation functions.

Significant research and numerous data warehouse and OLAP developments have been undertaken. In particular, the *data cube* created by Gray et al. had an important impact [28]. A data cube is a data model used for analysis. It is obtained by arranging

the requirements of a multi-dimensional data analysis and target data conceptually expressed as a multi-dimensional cube. Figure 2.2 presents sales information in the form of a three-dimensional data cube where each cell contains a statistical quantity such as amount of sales. Using calculation tools to process the data cube, one can make a variety of analyses.

In today's commercial relational database management systems (RDBMS), the basic functions of a data warehouse exist, making this appropriate to be used as the basis for the development of a simulation data warehouse system. *MDX* (MultiDimensional eXression) [14, 29] is frequently used as the interface for RDBMSs. It is implemented in Microsoft SQL Server [30] and has become the facto standard of commercial systems. The format of MDX is similar to that of SQL and is written as follows.

```
SELECT
Member selection ON COLUMNS,
Member selection ON ROWS
FROM <cube name>
[WHERE conditions specification]
```

This instructs a multi-dimensional cube to create a two-dimensional tables. `ON COLUMNS` specifies the content along the vertical axis and `ON ROWS` is the content along the horizontal axis. `Member selection` specifies a dimension and level from the concept hierarchy. In the `WHERE` field, conditions for the data cube are set. In the present study, MDX is implemented in the system.

2.3 Configuration of Prototype System

2.3.1 System Architecture

The architecture of the constructed prototype system is illustrated in Figure 2.3. All of the simulation data sets are stored in the data warehouse. For the management of the data warehouse, a commercial RDBMS, Microsoft SQL Server [30], is employed.

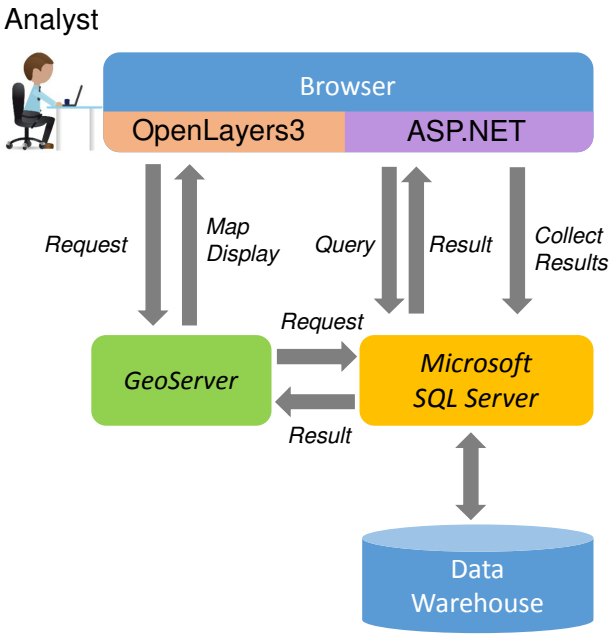


FIGURE 2.3: Prototype System Architecture

Because the system includes the basic functionalities of the data warehouse that contains a multi-dimensional data cube, interactive analysis can be performed efficiently using these built-in tools.

GeoServer [31] is an open source server software for sharing and editing geographic information. Access to the spatial database is made upon request and geographic information stored in the database is extracted as a diagram in vector or raster model. The software also has various functions to address geographic or spatial data.

Analyzers use a web browser as the analysis interface. To display a map on the browser, JavaScript for managing geographic data on the browser and OpenLayers [32], a library of CSS (Cascading Style Sheets), are used. For the management of information other than maps on the browser, the ASP.NET framework of Microsoft is included. This makes queries to the database in the backend and collects the results from the database. It also saves the data results to the database for reference by GeoServer.

2.3.2 Data Warehouse Functions of Microsoft SQL Server

In this section, the data warehouse functions of Microsoft SQL Server used to realize the prototype system are introduced. The data warehouse functions of SQL Server are called multi-dimensional modeling functions and are included in SQL Data Server Tools [33]. These basic functions are also provided by other commercial RDBMSs and hence, the implementation approach proposed in the present study can be used for other RDBMSs.

For the development of a multi-dimensional data cube, it is necessary to specify a target database and then develop a *fact table* that presents the collected information and *dimension tables* that present the dimensions for analysis. The fact table is uniquely determined once the target data are specified. Therefore, a major task is to develop the dimension tables considering concept hierarchies. Then, the cube wizard is used to develop a data cube and attributes of each dimension are added or deleted to develop the desired structure of the data cube. Deployment is required to permit an actual analysis using the data cube.

One of the analysis methods with a data cube is to use the OLAP function, which is a standard function of SQL. However, although this function is standardized as SQL/OLAP [34–36], there are only a small number of RDBMSs that implement the extended functions. Therefore, to implement OLAP in the prototype system, MDX [14, 29] is provided as the query language for SQL Server.

Because queries of this prototype system contain "narrowing-down" processing of data on a map, spatial indexing, a function of the SQL Server, is applied to the data as required. Consequently, a significant improvement of the response speed, in particular in the analysis of narrow areas, can be expected.

2.4 Development of Simulation Data Warehouse

Knowing the demand and background introduced in Section 2.1, in the present study we utilize the data warehouse technology for the integrated and interactive analysis of

simulation data. In particular, with an emphasis on spatio-temporal data such as disaster information, system techniques are developed by addressing the characteristic data and distinctive analysis properties in this field. We therefore use a new term, *simulation data warehouse*, for a data warehouse specialized for simulation data. Because spatial and temporal information are both involved in the spatio-temporal simulation, it is important to provide supports corresponding to them, such as constructing spatial index that aims to optimize query processing. Further, because it holds more exploratory aspect in scientific domain comparing to the business field, it is necessary to analyze data interactively using a trial and error process.

2.4.1 Objective Data Sets and System Requirements

For the development of the prototype system, we obtained two sets of simulation data, earthquake and tsunami. One is the flood depth data of a tsunami in Kochi City after the occurrence of an earthquake, provided by Prof. Koshimura's group at Tohoku University. The other set of data is the evacuation simulation data of the people flow in the same place, Kochi, and obtained from Prof. Sekimoto's group of the University of Tokyo. The sample of the people flow that we use in the present study was obtained from the simulations of approximately 40 thousand people conducted based on the personal trip data after the occurrence of the earthquake.

Because there was a difference in the target area and simulation length between the two sets of data, preprocessing and adjustment of the data sets were performed. The flood depth was simulated every 30 seconds on a $3,504 \times 2,364$ grid and three-hour data of the flood depth were recorded starting from the occurrence of the earthquake. The people flow data were those collected for six hours under the conditions where the earthquake occurred at 9 AM and the peak of the evacuation was 60 minutes after the occurrence of the earthquake. The data were recorded every ten seconds. To address the flood depth data areas, the finest area grid of the data cube was set to $4,096 \times 4,096$. (Hence, some grid cells are empty.) Each grid corresponds to a size of $10m \times 10m$ in real space. The coordinate values of the people flow data was easily assigned to each grid cell. The starting time was set to 9 AM as the people flow started at that time and the ending time

was set to three hours after that because the flood depth data were for three hours. The minimum time division was set to ten seconds. Hence, the same flood depth value was repeated three times.

In order to meet the requests for earthquake and tsunami analysis, the following requirements for the data set were applied.

1. Integrated analysis must be permitted for multiple sets of data. In the present study, two simulation data sets (flood depth and evacuee flow) were used. This is an essential requirement because multiple sets of data must be integrated to analyze and forecast damages.

2. Interactive analysis must be permitted using user interfaces including a visualization functionality. In particular, parameters and restriction conditions of the simulation data must be adjustable. For the visualization, a function of zooming in/out on a map is also necessary.

3. To realize interactive analysis, response time must be in an appropriate range, even for large data. In the present implementation of the system, the target response time must be one second or less.

In this study, Requirement 1 is satisfied by developing a data cube on the assumption of an integrated analysis scenario. The Requirement 2 is addressed using software (GeoServer) to display the geographic and spatial data. To meet Requirement 3, a prior arrangement of the data was undertaken for constructing the data cube. This ensures a compact statistical data set and a short response time for the expected basic processing.

2.4.2 Development of Data Cube

Although many possibilities can be considered for the integrated analysis of the data, the present prototype uses three dimensions, i.e., time, area, and flood depth of the tsunami, to assume a scenario of analyzing the number of evacuees. Figure 2.4 illustrates the data cube created based on these considerations. Each cell of the data cube can be accessed

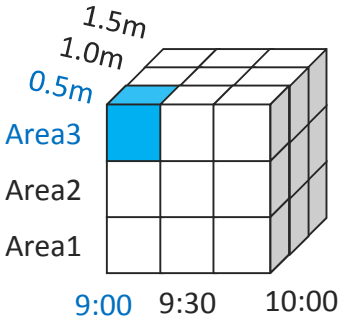


FIGURE 2.4: Data Cube for the Prototype System

using a key of the three dimensions, time, area, and flood depth of the tsunami, and each cell contains the number of evacuees as a fact. It should be noted that not all three dimensions are necessary for the analysis. For example, one can perform an analysis using only the time and area dimensions.

In order to extract strategic knowledge from a data cube, it is necessary to view its data at several levels of detail. In our case, an analyst may want to view the distribution of evacuees on the map at a finer granularity or a coarse granularity. Hierarchies of dimensions enable it by defining a sequence of mappings relating different dimension levels.

In the following segment, we explain how the concept hierarchies of each dimension of the data cube were set. Details of the developed database schema are presented in Appendix A.1. First, for the time dimension, the maximum time period was set to one hour and the minimum time period to ten seconds. Between them, a hierarchy of time periods of 30 minutes, ten minutes, five minutes, one minute, and ten seconds was set. Figure 2.5 is a representation of the hierarchy. At each level of the concept hierarchy, sequence numbers starting from zero are assigned.

For the area dimension, the space was divided into grid cells to which sequence numbers are assigned. In the concept hierarchy, a single cell of the entire dimension is the highest class and the cell is divided to 2×2 to obtain the second level. The cells of each level are further divided until $4,096 \times 4,096 = 2^{12} \times 2^{12}$ cells are obtained. This process creates 13 levels, which are indicated in Figure 2.6. The cell number in each level is determined

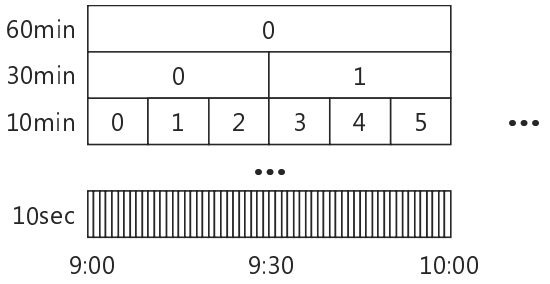


FIGURE 2.5: Concept Hierarchy for Time Dimension

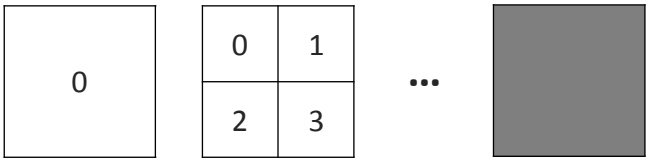


FIGURE 2.6: Concept hierarchy for Area Dimension

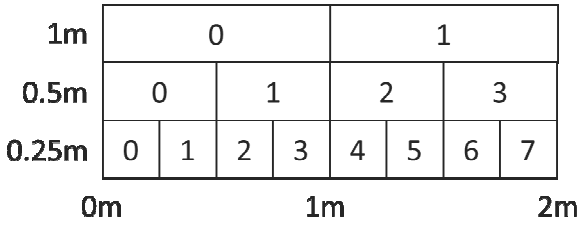


FIGURE 2.7: Concept Hierarchy for Flood Depth Dimension

by the z-order method [37], which assigns relatively similar numbers to the cells located close to each other in a two-dimensional plane for easy calculation of the numbers.

For the flood depth dimension, a concept hierarchy of three levels was created as displayed in Figure 2.7. In this study, the levels are defined by the divisions 1 m, 0.5 m, and 0.25 m, respectively.

Based on the above setting of the dimensions, a query in the MDX language setting the level of each dimension can be issued. For example, the query could be "Obtain the total number of evacuees in each cell with a 10 minute division for the time dimension, 16×16 divisions for the area dimension, and 0.5 m interval for the flood depth." One can use a part of the dimensions (e.g., not use the flood depth), set a range to the dimensions (e.g.,

calculate only from 9 AM to 10 AM), or change the aggregate functions (e.g., average, maximum value, or other).

An RDBMS with the data warehouse functionality has a function of partial prior data arrangement according to possible patterns of queries in order to accelerate response processing when queried [12, 14]. Further, high-speed query processing can be realized by combining values that are partially calculated in advance.

Regarding data warehouse size, the data cube's main body (fact table) displayed in Figure 2.2 is dominant in size in the data warehouse. Because the time division is ten seconds in the finest division level, the number of divisions is $3 \times 60 \times 60 \div 10 = 1,080$ for three hours. The spatial grid size is $4,096 \times 4,096$ and the flood depth dimension is divided to eight divisions. Therefore, if the value of a single cell is presented in 4 bytes, we have $4 \times 1,080 \times 4,096 \times 4,096 \times 8$ (bytes) = 540 (GB). However, the actual measurement data size in the entire data warehouse was 8.2 GB. This is because not all cells have values and data compression is performed by SQL Server. In the present implementation, fine divisions were used. However if ten-second divisions or $10m \times 10m$ divisions are not necessary, the data cube can be made more compact.

2.5 Case Study of Query and Visualization

The user interface (or web browser) used for the analysis in the prototype system can easily set parameters and the narrowing-down condition (e.g., start time, end time). An image of the user interface is presented in Figure 2.8. It is a two-layer image with people flow data in a heat map format, which is the result of the query, presented on an administrative division map in Kochi Prefecture. An approximate number of evacuees in each cell can be found using two slide bars to interactively change the time and flood depth. The area level can be changed using the zoom-in/out function. Every time the designation changes, a new query in MDX language is issued to the backend SQL Server and new total values are calculated and presented. Analyzers can perform analyses using these functions and gradually narrowing-down the areas from a wide map to a narrow map.

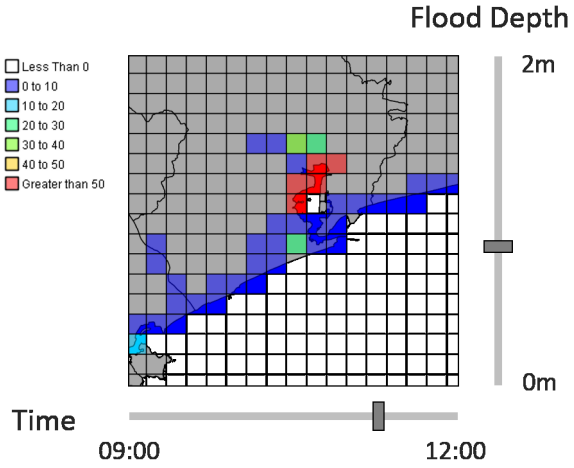


FIGURE 2.8: Image of the User Interface

Figure 2.9 is an example of the zoom-in function. In this example, the maximum number of evacuees in each area in the specified time period from the start time to the end time is presented as a heat map. In this case, we assume that the analyzer reviewed the upper map first and was interested in the central portion. The analyzer uses the "+" icon on the upper left to zoom in and a result is presented immediately. The area is made finer by one level and the adjustment is automatically made by the system. In the prototype system, the response time from when the query was issued to when the result was displayed was approximately 0.7 seconds.

There is also another function realized implemented in the prototype. This creates a motion picture by visualizing a temporal change in the number of evacuees in the target area. Presentation with a motion picture is effective for understanding a change in a situation.

The system requirements described in Section 2.4.1 are examined based on the experimental results. "Integrated analysis must be permitted for multiple sets of data" (Requirement 1) was satisfied by the development of the data cube. It was also verified that the system satisfied "Interactive analysis must be permitted using user interfaces including a visualization function" (Requirement 2), which was a relatively simple requirement for the analysis. For "To realize interactive analysis, response time must be in an appropriate range, even for large data" (Requirement 3), a response time less than

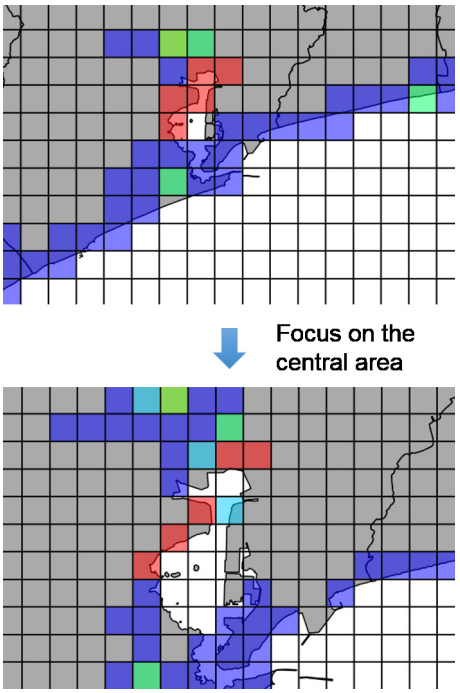


FIGURE 2.9: Zoom-In Feature

the preset condition, one second, was achieved. Therefore, it must be considered that the proposed prototype system satisfied all the requirements. Because the developed system was a prototype with incomplete functionality, it was relatively easy to meet these requirements. When functions are added and expanded in the future, the requirements must be reviewed and a verification of the fulfillment of the requirements elaborated.

2.6 Related Work

Studies on data warehousing and OLAP have been primarily in the business field. However, data warehousing has also been developed for spatio-temporal data closely related to the present study. Explanations and surveys can be found in chapter 11 of [14] or in [38]. The method of developing a concept hierarchy using the inclusion relation between spatial areas, used in the present study, was also employed in conventional spatial data warehouses. A spatial database technology was developed and has been realized as an

implementation technique for the efficient utilization of spatial index [4, 5]. Further, efforts to expand a data model [18] to incorporate the semantics of a geographical space or to introduce a more flexible space division method [19] have been made.

The majority of the studies on spatial data warehouses use geographical data and aim to analyze statistical information (e.g., population distribution). In addition to the spatial information, time information may also be utilized. From this perspective, the data warehousing of movement tracking data such as in [39] is technically closely related to the present study as it integrates and arranges continuous spatio-temporal information. One of the characteristics of simulations is that they use not only spatial information but also time information. Further, it is sometimes difficult to determine whether a certain kind of information, such as the flood depth in this paper, should be treated as a dimension or fact. Because various experimental parameters and conditions are set in simulations, how to analyze the simulation data in an integrated manner when the simulation experiments conducted with different parameters could be a problem. Previous conventional studies did not encounter this issue.

The present study aims to support exploratory analysis. For the exploration of data cubes with data warehouse and OLAP, studies have been made primarily in the business field [6, 8, 40]. An example of the studies was to identify an exception from a data cube to support a user's data exploration. In [6], indices of exceptions are calculated in advance to guide analyzers for data exploration. In a different approach [40], interestingness is calculated based on the context of the user. In [8], advanced operators were proposed for data exploration. The aim of the operators was to determine an exception, as in previous studies, without excessive manual exploration.

Unlike these studies that attempted to identify "exceptions" defined by considering general business situations, our study focused on earthquake and tsunami simulation data on the assumption that users would be interested in indices, such as intensity of damage, which were dependent on the target domain. In the development, we addressed the available content of the simulation data and focused on a combination of the flood depth and the number of evacuees, in particular on their visualization. In the future, we would

like to introduce more advanced indices, in addition to the above, to assist the user's data exploration for earthquake and tsunami damage analyses.

In the present system development, the implementation is realized by utilizing the advanced data warehouse function of Microsoft SQL Server. However, it is expected that dynamic elements could increase in the future as stated in the previous section. In the data cube, computational acceleration is realized by preprocessing and the prior arrangement of the data. Another option is to use the hybrid architecture of an array DBMS and processing system that has been developed in recent years. Array DBMS systems such as SciDB [41, 42] can efficiently manage extensive array data and are suitable for storage and query of data provided on a grid, such as flood depth data.

In this study, the simulation data were visualized. There are many studies on visualization systems. Although a commercial system is available, processing that closely cooperating with database and data warehouse needs to be studied more. For example, [43] used the MapReduce technology to develop an efficient visualization system of spatio-temporal satellite data.

2.7 Conclusion and Future Work

In this chapter, we demonstrated the concept of simulation data warehouse, architecture of a prototype system, and case example of an interactive analysis of disaster simulation data. Through the development of the proposed prototype, issues to be resolved in the future were identified. The followings are some of the problems.

- **Selection of dimensions and facts:** In the business field, it is not difficult to choose dimensions and facts. For example, the dimensions could be "period" and "area" and the fact could be "amount of sales". However, in the analysis of disaster data, flood depth could be used as dimension as in this paper or as fact to issue the query "Visualize flood depth data under a specified condition." Because the combination of dimensions and facts is not fixed, a flexible setting of the dimensions and facts must be possible in a simulation data warehouse.

- **System architecture:** For the simulation data warehouse, we used a commercial RDBMS and utilized the data warehouse functions and spatial database functions that it provided. This policy is effective for the analysis presented in this paper and realizes instantaneous responses. However, it is not sufficient for the above-mentioned combination of flexible dimensions and facts. Google BigQuery [44] or others that use parallel processing on many machines to realize a real-time response without preprocessing could present a computer resource or cost problem. A database management system (DBMS) specialized for the use of large array data, such as SciDB [42], could be considered.

- **Flexible visualization functionality:** As discussed previously, a method for visualizing an answer to a query in a form suitable for analysis is an important factor of realizing interactive analysis. In the prototype system, a simple interface was developed for each type of analysis. However, if the number of analysis types or query types is considerable, the cost would be excessive to develop interfaces for each one. Therefore, the development of a flexible visualization technique is required. As an alternative, an approach of embedding DBMS as a component of the visualization system [45] could be considered.

- **Advanced analysis functionality:** To consider a more advanced analysis function, let us use the query example of "Identify all seismic centers having a flood depth more than 1 m at site x with the parameters other than the seismic center being fixed to specified values." It is possible to answer this query because the data necessary required for the answer exist in the data warehouse. However, the problem is how to express the query and how to process it efficiently. A query language that can describe such a query in a simple manner could be necessary.

- **Cooperation with other systems:** Some analyses could require cooperation with other systems. An example is cooperation with visualization processing. For some queries, advanced visualization is required for presenting the analysis process or the analysis result. In these cases, coordination with a dedicated visualization system would be a practical solution. Further, if advanced statistical processing were required for a part of the target data, coordination with a statistical processing such

as in R would be useful. In addition, to incorporate domain-dependent analysis processing into a system, a function of incorporating into the system and executing a user-defined function provided by a user are also required.

Chapter 3

Hierarchical Histograms for Exploratory Analysis of Spatio-Temporal Data

3.1 Introduction

User-driven analysis is designed for traditional database systems, which assume that users are aware of exactly what they are looking for and have good knowledge of the contents of the database. However, as big data has made great success in a variety of fields, it leads to another type of analysis, *data-driven analysis*, in which it is not necessary for users to have prior knowledge about the target dataset or be aware of the type of queries they want to pose. Users often explore massive data by navigation or visualization tools to find interesting patterns. As a result, research on *data exploration* for sophisticated analytic processing of large amount of data has gained popularity [10].

As one of the common data types of big data, *spatio-temporal data* has been widely applied in various domains such as mobile applications and scientific research [1]. For instance, in scientific fields, simulations are conducted for the purpose of predictions, decision making, etc. As one of the typical simulations, disaster simulations like human

evacuation simulation are conducted for effective humanitarian relief and disaster management [2]. Such kind of disaster simulations generates large scale spatio-temporal data, which contains spatial and temporal information of evacuees on the target area during a period of time. Analysis of disaster simulation data can achieve various objectives, such as discovery of interesting patterns and shelter location suggestion [3]. Consequently, researches on enabling exploratory analysis of spatio-temporal data has been increasing in demand. Moreover, since a large amount of simulation data is generated by simulations with different conditions and parameters (e.g., time when earthquake occurs), *data warehousing* techniques that enabling massive data storage and exploration are also important [46].

In this study, we focus on the exploration-based analysis of spatio-temporal array data, which is based on the *array* representation for large-scale scientific data computation. While the traditional relational model cannot handle such array data, we exploit the functionality of array database management systems (array DBMSs) [47], to support scientific computing and exploratory analysis. A motivating example is as follows. Consider an earthquake analyst intends to explore the evacuation simulation data after an earthquake. A query like "Return the spatial distribution of evacuees during the first hour after the earthquake occurs." is used to understand the movement of evacuees during the given period of time.

There are two main challenges of exploration-based analysis of spatio-temporal array data: 1) Spatio-temporal data has characteristics on spatial and temporal scales because of the semantic meanings of different granularities, such as one day, one hour and one minute on the time dimension, as well as country, city and district on the spatial dimension. However, instead of exhausting exploration by drill-down and roll-up operators, more advanced operators such as effective summarization and visualization that navigates users to find interesting knowledge are required. 2) Since the target spatial area of simulation data is usually quite large, it is difficult for users to determine which part of the area should be focused on. In this case, a succinct summary of the overall region is considered to be useful.

In order to summarize the spatio-temporal array data to show the overall data distribution, we use the notion of *histogram* [22], which is widely used in database systems to summarize relations for query selectivity estimation and approximate query answering [21, 23, 48, 49]. Such histograms provide a concise summary of the target data, and the accuracy depends on the chosen partition rules. Other techniques for data summarization such as Minimum Description Length (MDL) principle [50, 51] for summarizing query results of OLAP applications, as well as *wavelets* [52, 53], which is important for image compression and has also been used extensively for approximate query answering in the database area, are not the focus of this work.

Existing work on histograms generally focus on the accuracy of histograms or approximation of query results (such as selectively estimation), while our work considers the quality of histograms as well as the efficiency of histogram construction, which is important for visualization and data exploration. In this study, we define a *spatial V-optimal histogram* to represent the array data on the two-dimensional spatial area, as an extension of one-dimensional *V-optimal histogram* [23]. However, building histograms is often very time consuming, especially for spatial V-optimal histogram, which is proved to be NP-hard [54]. Unlike the error-bounded V-optimal histogram construction problem studied in [54], we focus on the space-bounded histogram construction, and propose exact and approximate algorithms based on dynamic programming for *hierarchical* histogram construction. We prove that the exact and approximate algorithms have approximation bounds with respect to the arbitrary partitioning method. In order to enable interactive analysis of large-scale array data, we also propose a heuristic approach based on *l-grid partitioning*, which is a specific hierarchical partitioning, to speed up the histogram construction further. In addition, we implement our proposed histogram construction methods on SciDB [41, 42], an open source array-oriented DBMS for the efficient storage and manipulation of large array data. We perform experimental evaluation on evacuation simulation data as well as real taxi trajectory data to show the effectiveness and the efficiency of the proposed methods.

Our contributions can be summarized as follows:

- We study the problem of constructing histograms that summarize the data distribution of a specific spatial area during a time interval (Section 3.2).
- We propose both exact and approximate algorithms to construct the *spatial V-optimal histogram*. In addition, we also propose a heuristic algorithm based on l-grid partitioning for fast histogram construction (Section 3.3).
- We implement the proposed histogram construction algorithms on SciDB, the state-of-the-art array DBMS (Section 3.4).
- We conduct extensive experiments on massive simulation data, real taxi data as well as synthetic data with different distributions, to verify the effectiveness and the efficiency of the proposed methods (Section 3.5).

The rest of the chapter is organized as follows: Section 3.2 introduces preliminaries and the problem definitions in this work. Section 3.3 presents our method for constructing hierarchical spatial histograms for array data analytics. Section 3.4 discusses the implementation of our method on an array DBMS, SciDB. Experimental results and analyses are reported in Section 3.5. Section 3.6 introduces the related work on histograms and data visualization. Section 3.7 concludes the chapter.

3.2 Preliminaries

In this section, first we introduce the basic definition of spatio-temporal arrays. Then, we formally define the histogram construction problem that we are going to investigate in this work. In addition, we discuss the complexity of the problem, as well as existing partitioning approaches in the literature.

3.2.1 Problem Definitions

Definition 3.1 (Spatio-temporal Array)**.** A *spatio-temporal array* $\mathcal{A}(Dim, Attr)$ is a three-dimensional array with numeric attribute values. Dim consists of dimensions x, y and t,

while *Attr* contains a list of attributes $(a_1, \cdots, a_{|Attr|})$. Dimensions x and y correspond to a two-dimensional spatial grid structure, and dimension t is represented by a series of time stamps with equal time interval τ. Each element $e_{x,y,t}$ in the spatio-temporal array contains a tuple of numeric attribute values corresponding to the position (x, y, t), that belongs to the domains of dimensions, i.e., $x \in Dim_x, y \in Dim_y$ and $t \in Dim_t$.

Note that the time interval τ can be one minute, one hour, etc., depending on the dataset and the purpose of analysis.

In order to catch an insight of spatio-temporal arrays, we use the notion of *histograms* from the studies on selectivity estimation and query optimization in the database area [21, 23, 48, 49]. In this work, we formulate the problem of constructing histograms for array data exploration based on the *space-bounded V-optimal histogram* [23], which is defined as follows. For a given number of buckets, a V-optimal histogram is the one with the number of buckets bounded by the specified threshold, but having the least variance, where variance is the sum of squared differences between the actual and approximate frequencies. While [23] proposed algorithms with a quality guarantee for *unidimensional V-optimal* histograms, the problem becomes totally different even in the two-dimensional case.

In our work, we focus on the two-dimensional space-bounded V-optimal histogram because of its bound on the number of buckets, which is able to enhance the usability of visualization. More precisely, the number of buckets in the result histogram affects the quality of the visualization, since visualizing the objective data by a histogram with too many buckets will confuse an analyst and increase the inefficiency of exploratory analysis [55]. Related work on visualization such as MuVE [55] defines the usability using the number of views, as one of the objectives for view recommendation. Also, [56] studies the techniques of aggregation, sampling and filtering to reduce the size of the result for interactive visualization. One of the intentions of [56] is that, visualization with too many objects to draw on the screen is too dense to be useful to the user. Therefore, a space bound (i.e., the number of buckets) is considered to be important for the usability of visualization.

Definition 3.2 (Spatial V-optimal Histogram)**.** Given a spatial-temporal array $\mathcal{A}$, an attribute a, an integer B defining the limit number of buckets, and an error metric $E()$, the *spatial V-optimal histogram* H of $\mathcal{A}$ consists of a set of buckets $\{b_1, b_2, \cdots, b_B\}$ with the minimum error. The histogram is generated by partitioning the whole spatial region into B non-overlapping buckets. Each bucket b_l ($1 \leq l \leq B$) has a corresponding rectangle area $b_l.area$ and an aggregated value $b_l.val$. The value of $b_l.val$ is calculated by averaging the attribute values of elements in $\mathcal{A}$, the area of which is covered by $b_l.area$.

Next, we define the error function. It is based on the notion of *sum of squared error* (SSE), which is a common error metric for measuring difference between two data distributions.

Definition 3.3 (Error Function)**.** The *error function* E of a spatial V-optimal histogram H is defined as

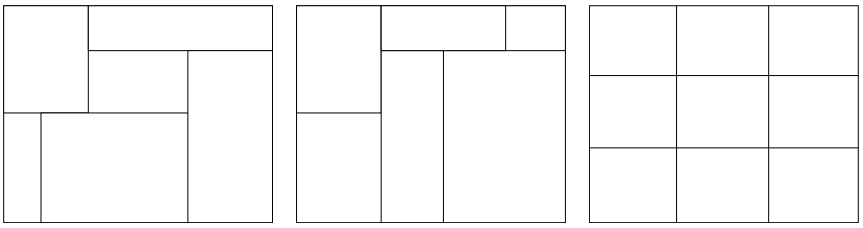
$$E_B(H) = \sum_{l=1}^{B} \sum_{e \in \mathcal{A} \wedge e.(x,y) \in b_l.area} (e.a - b_l.val)^2. \tag{3.1}$$

The defined error function has two properties, *monotonicity* and *superadditiveness*. An error function is *monotonic* if $E(r) \leq E(R)$ for two buckets r and R, where $r \in R$. We say an error metric is *superadditive* if $E(r) + E(R) \leq E(r \cup R)$, where r and R are two disjoint buckets.

3.2.2 Discussion

[54] proved the problem of minimizing the heft (i.e., error metric value) of the partitioning with p tiles on two dimensional array is NP-hard for some measures, where the metric of sum of squared error (SSE) is one of those metrics. The problem of constructing a spatial V-optimal histogram defined in this work is intrinsically the same as the proved NP-hard problem, and the proof is based on a reduction from the *Planar 3-SAT problem* (shown to be NP-complete in [57]).

In general, the histogram construction problem changes with different partitioning strategies. For a two-dimensional array, there are many types of partitioning. Here, we show the three commonly used ones in Figure 3.1 [54]. *Arbitrary partitioning* is the one with



(a) Arbitrary (b) Hierarchical (c) $p \times p$

FIGURE 3.1: Types of Partitioning

no restrictions on the sub-rectangles shown in Figure 3.1(a). It is obvious that the computation cost of arbitrary partitioning is the most expensive one, which is proved to be NP-hard as described above. In this study, we consider the *hierarchical partitioning* shown in Figure 3.1(b). A general hierarchical partitioning can be represented by a binary tree in which each node represents a subarray, and the root represents the whole region of the computed array. The $p \times p$ *partitioning* conducts p times of partitions on each dimension, as shown in Figure 3.1(c). It is a special case of the hierarchical one if the sibling nodes of the hierarchical tree are the same along one dimension.

We propose both exact and approximate algorithms for hierarchical partitioning, and prove that compared to the arbitrary partitioning method, the errors introduced by the exact or the approximate algorithm are bounded. In order to enable interactive analysis of large-scale array data, we also propose heuristic approaches based on the l-grid partitioning, which is a special case of hierarchical partitioning.

3.3 Hierarchical Histograms

In this section, first we propose an exact histogram construction algorithm based on dynamic programming for general hierarchical partitioning. Since the computational cost of dynamic solution is expensive, we propose an approximate solution with an approximation bound. Both of the two algorithms are proved to have approximation bounds with respect to the arbitrary partitioning, in which no restrictions on the arrangement of buckets. In addition, we also propose a heuristic algorithm for the efficiency.

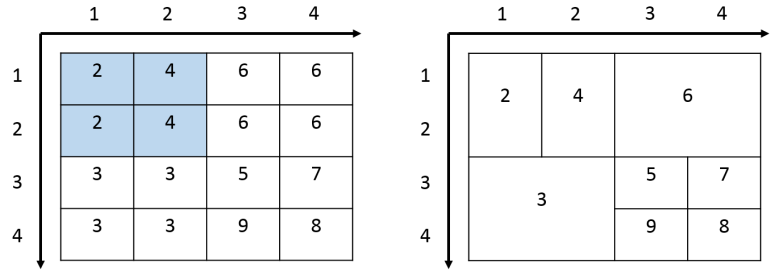
3.3.1 Exact Algorithm

Recall that the problem is to find a histogram which has at most B buckets with minimum error. We define $E^*_B(R)$ as the minimum error of region $R([i \cdots j], [k \cdots l])$ with at most B buckets. The computation strategy of the exact algorithm is based on the following equation.

$$\begin{aligned} E^*_B(R) = \min_{i \le x < j, k \le y < l, 1 \le b < B} \{ & E^*_b(i \cdots x, k \cdots l) + E^*_{B-b}(x+1 \cdots j, k \cdots l), \\ & E^*_b(i \cdots j, k \cdots y) + E^*_{B-b}(i \cdots j, y+1 \cdots l) \} \end{aligned} \tag{3.2}$$

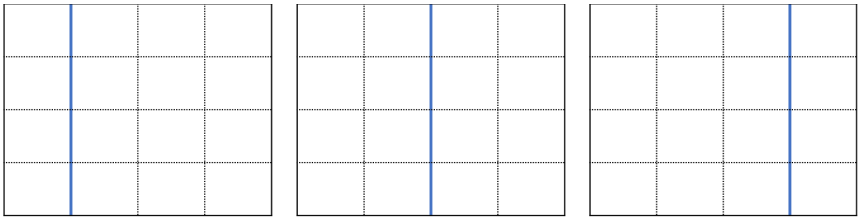
That is, the minimum error of region R is the minimum error among the possible partitions along x (horizontal) and y (vertical) dimension with possible number of buckets. As shown in Eq. (3.2), b represents the possible number of buckets assigned to each subregion by a partition. For each partition, there are $B-1$ possible values of b, i.e., $1, 2, ..., B-1$. Furthermore, there are $j-i$ possible partitions along x dimension, and $l-k$ possible partitions for y dimension. For instance, given a 4×4 array $\mathcal{A}$ shown in Figure 3.2(a), the whole region of $\mathcal{A}$ is represented as $R([1 \cdots 4], [1 \cdots 4])$. There are 3 possible partitions along y dimension (lines in blue), each of which splits the whole region into two subregions, as shown in Figure 3.3. Note that, for each subregion, there are also several possible partitions to split it into further smaller subregions. The same holds for x dimension.

The exact algorithm is based on dynamic programming using Eq. (3.2) as described above. Consider the whole region of the input array is represented as R, in order to compute the optimal result of R, we incrementally compute the possible subregions of R by increasing the side length of each dimension. E.g., for array $\mathcal{A}$ with region $R([1,4],[1,4])$, we incrementally compute the subregions with side lengths of $1, 2, 3$ and 4, for each dimension. As a result, there are 10^2 possible subregions of the whole region for two dimensions. Figure 3.2(b) shows an optimal histogram of A when the number of buckets $B = 8$, the error of which is 0.



(a) An example 2-D array $\mathcal{A}$ (b) The optimal histogram ($B = 8$)

FIGURE 3.2: An Example Array with the Optimal Histogram



(a) (b) (c)

FIGURE 3.3: Possible Partitions Along y-Dimension

3.3.1.1 Complexity Analysis and the Approximation Bound

The total computation time is computed by $O(n^4(nB^2 + t_E))$, where $O(n^4)$ represents the possible number of subregions, and $O(nB^2 + t_E)$ represents the cost of computing candidate histograms for each subregions. t_E is an upper bound on the time taken to calculate the error metric value of any given bucket. For each subregion, we compute $O(B)$ optimal result histograms with bucket number $b \in [1, B]$. Moreover, for the result histogram with one bucket, one calculation of the error of the target region is necessary, which is t_E, while computing the result histograms with bucket number b ($1 < b \leq B$), $O(nB)$ computation cost is needed based on Eq. (3.2). Since the error of a given bucket can be computed in $O(1)$ by efficient methods such as holding the statistic information of subregions for further computation, t_E can be reduced to $O(1)$. Therefore, the total computation time is $O(N^{2.5}B^2)$, where N is the size of the computed array (i.e., $N = n^2$).

In order to compute the optimal histogram of a target region with a given bucket number B, the optimal results (i.e., a list of optimal solutions with the possible number of buckets) of its subregions are temporally held. Since there are totally $O(N^2)$ possible subregions for the whole region R, the space complexity of the algorithm is $O(N^2B)$.

In what follows, we argue that the proposed exact algorithm for hierarchical histogram construction can approximately solve the problem of arbitrary partitioning with the approximation bound.

Lemma 3.4. *Consider any arbitrary partitioning of an array $\mathcal{A}$ with superadditive error metric at most δ and B buckets. There exists a hierarchical histogram of $\mathcal{A}$ with error metric at most δ and at most $4B$ buckets.*

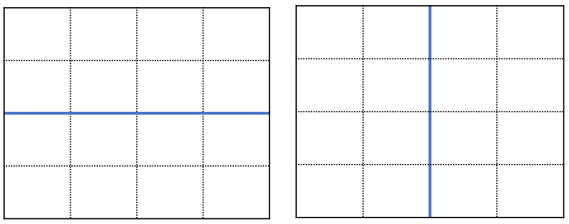
The proof of Lemma 3.4 is shown in Appendix B.1 Based on the above observation with the construction algorithm of hierarchical histograms, we can obtain the following result.

Theorem 3.5. *Let δ_0 be the minimum error of the arbitrary partitioning problem with at most B_0 buckets under a superadditive error metric. Then, in $O(N^{2.5}B^2)$ time, we can compute a histogram consists of at most B buckets with error $\delta \leq \delta_0$ and $B \leq 4B_0$.*

3.3.2 Approximate Algorithm

Since the proposed exact algorithm for hierarchical partitioning is quite computationally expensive, next we propose an approximation algorithm to improve the running time of dynamic programming algorithm, by limiting the permissible side intervals of buckets. Moreover, the approximation algorithm can control the approximation bound in a structured manner as described below.

Given a set of positive integers $L_0, L_1, L_2, ..., L_k$, which is a geometric progression of values, such that $L_0 = n$ and L_{i+1} divides L_i, for $0 < i < k$. For the ease of explanation, we consider the case of common ratio equals to 2 in the rest of the chapter. E.g., given $L_0 = 16, L_k = 4$, the set of values is $\{16, 8, 4\}$. For each L_i, there is a set of permissible intervals S_i that the length of each interval in S_i is a multiple of L_i, and less than L_{i-1}. For instance, if $L_0 = 2L_1$, the permissible intervals of L_1 for each dimension comprise the following intervals with side lengths L_1: $[1, L_1], [L_1 + 1, 2L_1]$. For L_k, since we aim to compute the optimal result for the intervals with length L_k, there's no constraints on the interval length. Therefore, the permissible intervals of S_k contain intervals in S_i for $i = k$ as defined above, as well as all their subintervals.



(a) partition on x dimension (b) partition on y dimension

FIGURE 3.4: Possible Partitions for Side Length of $L_0 = 4$

In the same example of a two dimensional array $\mathcal{A}$ in Figure 3.2(a), each dimension corresponds to the target spatial region of interval $[1, 4]$. Each element of the array contains a numeric attribute value at the corresponding position. Given $L_0 = 4$ and $L_k = L_1 = 2$, the permissible intervals of each dimension (x or y) for L_0, represented as S_0 is $[1, 4]$. For L_1, which is also L_k, the permissible intervals S_1 consists of $[1, 2]$ and $[3, 4]$ with side length 2, as well as all the subintervals of them, i.e., $[1, 1], [2, 2], [3, 3], [4, 4]$.

Then, we define a *k-hierarchical histogram* based on the following *partition rules* for each dimension.

1. For intervals with side lengths in S_i $(0 \leq i < k)$, the permissible partitions split the interval into subintervals in S_{i+1}, the lengths of which are multiples of L_{i+1}.

2. For intervals with side length of L_k, all the subintervals are permissible by partitions with no constraint.

For instance, Figure 3.4 shows the possible partitions (lines in blue) for intervals with length $L_0 = 4$. Based on the partition rule (1), there's only one possible partition along x and y dimension, respectively, since the subintervals should equal to $L_1 = 2$. For $L_k = 2$ (rule (2)), consider a subregion $r([1, 2], [1, 2])$, the shaded area shown in Figure 3.2(a), the possible partitions along x and y dimension are also two (one for each dimension).

Based on the partition rules described above, we propose an approximation algorithm to efficiently construct histograms, which is again based on dynamic programming. The basic idea is to incrementally compute the candidate results of possible subregions by increasing side lengths. The pseudo-code is illustrated in Algorithm 1.

Algorithm 1: AprDP($\mathcal{A}, B, S$)

Input: $\mathcal{A}$: input array, B: maximal number of buckets, $S = \{L_k, L_{k-1}, \cdots, L_0\}$: permissible side lengths

Output: Result histogram $H_B(\mathcal{A})$

```
1  C ← ∅, C' ← ∅;                                          // Initialize the candidates
2  foreach xl ∈ S do
3      C ← C', C' ← ∅;      // Update C with the results of the previous loop,
         clear C'
4      foreach yl ∈ S do
5          Divide the whole region of A evenly into subregions R with size (xl, yl);
6          foreach r ∈ R do
7              foreach b ∈ [1, B] do
8                  if xl = y_l = L_k then
9                      Compute H_b(r) by the exact algorithm;
10                 else if xl = L_k ∨ yl = L_k then
11                     Compute H_b(r) by Eq. (3.2) based on partition rules (1) and (2);
12                 else
13                     Compute H_b(r) by Eq. (3.2) based on partition rule (1);
14                 C' ← C' ∪ H_b(r);                        // Update candidates
15 return H_B(A);
```

We incrementally compute the result histograms of subregions with side lengths in S_i for each dimension, where $i \in \{k, k-1, \cdots, 0\}$ (Line 2-14). For each subregion r, we compute the result histogram $H_b(r)$ with bucket number $b \in [1, B]$ based on Eq. (3.2) (Line 6-14). As different partition rules are conducted according to different side lengths of target regions, we consider three cases as follows. First, we call the exact algorithm for the subregions with both of the two side lengths equal to L_k (Line 8-9). The second case is that, only one of the side lengths equals to L_k (Line 10-11). In this case, we partition on the dimension with side length larger than L_k by partition rule (1), while for the dimension with side length of L_k, we use partition rule (2). The final case is that both of the side lengths are larger than L_k, in which case we partition the target region based on partition rule (1). After we compute the result histograms of the whole region of $\mathcal{A}$, the result $H_B(\mathcal{A})$ is returned.

3.3.2.1 Complexity Analysis and Approximation Bounds

There are $O(\frac{n^2}{L_k^2})$ possible subregions, for each of which, we compute $O(B)$ result histograms with the possible bucket number $b \in [1, B]$. The computation cost of each $H_b(r)$ is $O(L_k^5 B)$, calculated by the cost of exact algorithm as mentioned in the last section, which is larger than the cost of other cases (i.e., Line 10-13 in Algorithm 1). As a result, the computation cost for all of the generated subregions is $O(n^2 L_k^3 B^2)$. We assume that L_k is bounded by n^ϵ, where ϵ is a small positive constant factor, then the total computation cost is represented as $O(N^{1+1.5\epsilon} B^2)$.

For space complexity, as shown in Algorithm 1, the candidates of subregions with region size (xl, yl), for each xl and all $yl \in S$ are temporally held in C'. There are the $O(\frac{n^2}{L_k^2})$ possible subregions, for each of which, there are $O(B)$ possible candidate results with different bucket numbers, as well as $O(L_k)$ candidate results of subregions with one side length equals to 1, are needed for further computation. Therefore, the total space complexity is $O(\frac{n^2}{L_k} B)$, represented as $O(N^{1-\frac{\epsilon}{2}} B)$.

In the following part, we also argue that the proposed approximation algorithm can approximately solve the hierarchical partitioning problem as well as arbitrary partitioning problem with the approximation bounds.

Lemma 3.6. *Consider an optimal hierarchical histogram of an array $\mathcal{A}$ with superadditive error metric at most δ and B buckets. There exists a k-hierarchical histogram of $\mathcal{A}$ with error metric at most δ and at most $(2k+1)^2 B$ buckets.*

The proof of the above lemma is shown in Appendix B.2. Based on the analysis described above, we can conclude as follows, the proof of which is explained in Appendix B.3.

Theorem 3.7. *For any superadditive error metric, in $O(N^{1+1.5\epsilon} B^2)$ time, we can compute a histogram consists of at most B buckets with error $\delta \leq \delta_0$ and $B \leq O((1-\epsilon)^2 (\log n)^2 B_0)$, where δ_0 is the minimum error of any arbitrary histogram with at most B_0 buckets.*

By setting ϵ appropriately, the algorithm can be executed in near linear running time and has a quality guarantee as well. The smaller ϵ indicates less computation cost but more unnecessary buckets, which means the worse result quality, vice versa.

3.3.3 Heuristic Algorithm

As described above, the computational cost of the approximate approach is still high, when the array size N and the number of buckets B is large. Considering the usability of visualization, which demands for immediate response of queries, we also propose a heuristic algorithm based on a more specific hierarchical partitioning, the *l-grid* hierarchical partitioning, which is introduced as follows.

The overall spatial region corresponds to the root of the l-grid tree structure at depth 0, every partition evenly divides the interval of target region into l sub-intervals for each dimension. For instance, the spatial V-optimal histogram is considered as a 2-dimensional case, each partition of the target region generates $l \times l$ sub-regions with the same dimension length. Figure 3.5 shows an example histogram construction by l-grid partitioning when $l = 2$.

The heuristic algorithm follows a a *top-down greedy* strategy to conduct the l-grid partitioning, until the number of buckets is not less than B. We define the greedy criteria as *Max-Red*, which chooses a partition with the maximum error reduction at each time. As the example shown in Figure 3.5 when $l = 2$, the first partition conducts with only one choice, which is to partition the whole region into four subregions. Then, the second partition conducts by comparing the four possible partitions of current buckets. Since the error reduction of the right-down bucket is the largest one, we further split this bucket into four subregions to construct the result histogram when $B = 7$.

The number of additional buckets Δb generated by each partition, is computed as $l^2 - 1$, if no empty cell exists, e.g., 3 additional buckets are generated when $l = 2$. Then, the total partition time is computed by $\frac{B-1}{l^2-1}$, since the maximum number of buckets is B. As the whole region is considered as 1 bucket, the first partition divide the whole region into l^2 subregions, further partition generates $l^2 - 1$ additional buckets as mentioned above. Therefore, the list of bucket numbers generated by each partition is shown as $(l^2 - 1) + 1, 2(l^2 - 1) + 1, \cdots, B$. For each partition, the greedy strategy is conducted by comparing all the buckets that can be split. As a result, the total computation cost is $O(B^2 l^{-2} t_E)$, where t_E is an upper bound cost of computing the error metric value of a given bucket.

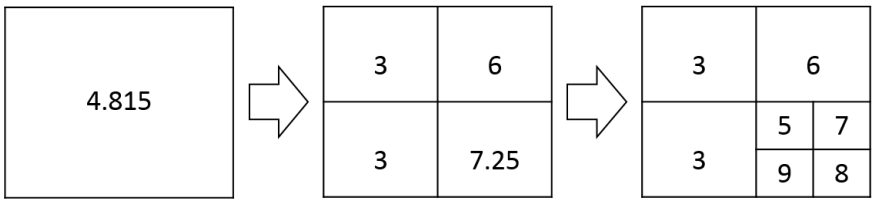


FIGURE 3.5: An Example l-Grid Partitioning ($l = 2$)

3.4 Implementation on Array DBMS

Since scientific data typically have spatio-temporal structure, indexed arrays are often used to represent scientific data, while the traditional relational model cannot handle such array data [58]. Some array database management systems are developed to support scientific computing and online analytical processing (OLAP), such as RasDaMan [59], ArrayDB [47], SciDB [41, 42, 60], etc. In this work, we implement the proposed histogram construction methods on SciDB, one of the state-of-the-art array DBMSs.

3.4.1 Overview of SciDB and Our Configuration

SciDB is a parallel database system based on the shared nothing architecture, and designed specifically for scientific data management and analysis. In SciDB, array data is organized on disks as *chunks*, which also work as the I/O units. Chunking ensures that logically co-located cell values are clustered into the same physical block, and enables efficient query processing. Chunks are distributed among all instances of a SciDB cluster. SciDB uses a variety of functions for chunk mapping, and the default is the hash-based distribution. During query execution, each instance node executes query locally with the partial array data, and transfers intermediate results across nodes when global computing is necessary.

SciDB supports a variety of operators for array data, such as data extraction (e.g., *slice*, *between*) and aggregation (e.g., *aggregate*, *regrid*). For aggregate operators, a lot of aggregate functions (e.g., sum, min, and max) are supported. In addition to built-in operators and aggregate functions, users are also allowed to implement user-defined operators (UDOs) and user-defined aggregates (UDAs) using C++ language based on the plugin

mechanism. Once a plugin operator is loaded into the SciDB library system, it can perform in the similar way with the built-in operators of SciDB for data processing. SciDB also support a python interface, SciDB-Py, that leverages data-parallel computing of scientific data on Python, while SciDB acts as a back-end database server. The details of implementation will be explained in the following section.

Our experiments are performed on a SciDB cluster, which consists of a coordinator server (CPU: Intel Xeon E5-2637 v4 @ 3.50GHz × 2, RAM: 128GB, OS: Ubuntu 14.04 LTS 64bit) and three worker servers (CPU: Intel Xeon E5620 @ 2.40GHz × 2, RAM: 32GB, OS: Ubuntu 14.04 LTS 64bit), each with three instances. There are totally 10 SciDB instances that work in parallel. The used version of SciDB is 16.9.

3.4.2 Implementation Details

The proposed methods, i.e., exact algorithm, approximate algorithm and greedy algorithm, are implemented on SciDB as UDAs, represented as $DPMerge$, $AprMerge$ and *LGreedy*, respectively. The UDAs are executed by employing the *aggregate* operator that conducts aggregation on the input array. An example of the aggregate query is shown as follows:

```
aggregate(array_name,UDA(atr_name))
```

where *array_name* is the name of input array, and *atr_name* is the name of the aggregated attribute.

Using the UDAs described above, users are able to conduct the proposed histogram construction methods on any target array by setting it as the input array. Moreover, parallel computation is available by the *regrid* operator, which conducts aggregation on the subregions with specified block sizes at the same time. An example of the *regrid* operator is as follows:

```
regrid(array_name,blk_x,blk_y,UDA(atr_name))
```

where *blk_x* and *blk_y* represent the specified block sizes for each dimension. In addition, using the SciDB-Py interface, visualization and interactive analysis of histograms are also available.

3.5 Experiments

3.5.1 Objective Datasets

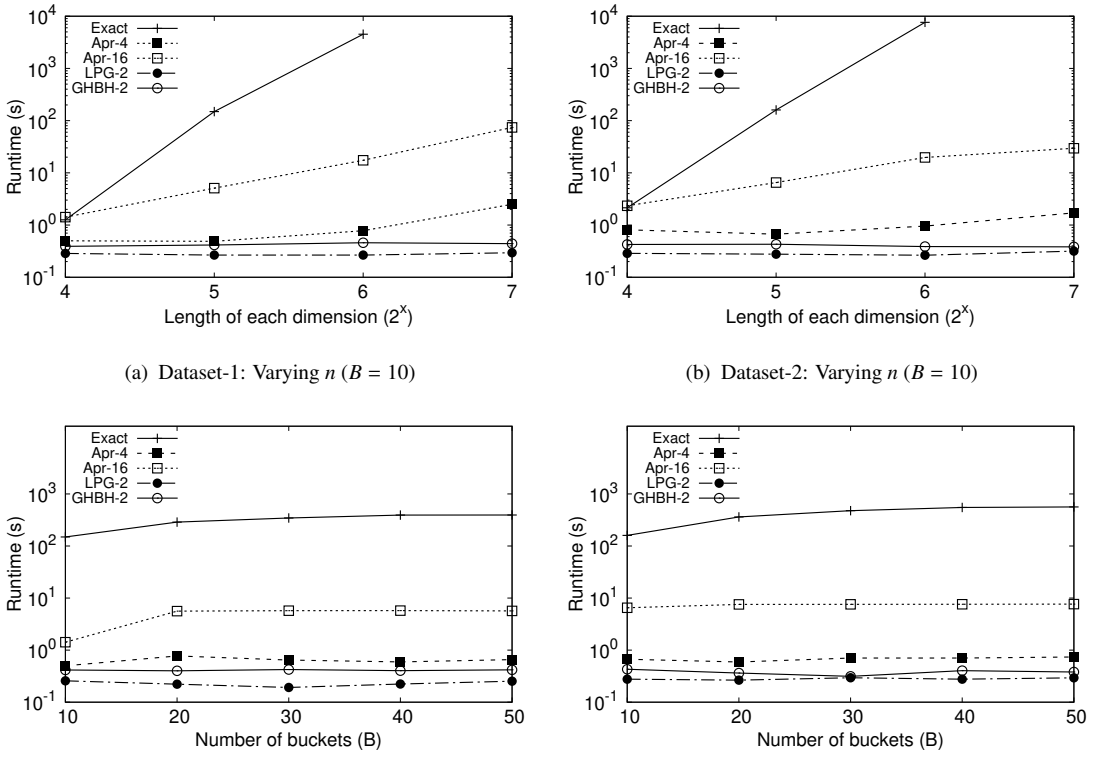
The objective datasets consist of an evacuation simulation data in the event of large-scale earthquakes in Kantou area of Japan, a real dataset of taxis' trajectory data in Beijing [61, 62], and synthetic datasets with three different distributions. The simulation data, represented as Dataset-1, contains abount 194 millions records of evacuees' mobility data during 24 hours after an earthquake occurs. The real trajectory data, represented as Dataset-2, contains about 15 million data points of 10,357 taxis during the period of Feb. 2 to Feb. 8, 2008.

The records of Dataset-1 and Dataset-2 are both in the format of (id, time, x, y): id denotes user ID; time, a time stamp; x, y, the location of a user. We preprocess the two datasets by dividing the spatial regions with a maximum grain size of $4,096 \times 4,096$, while the number of objects in each cell is aggregated by every minute and ten minutes, respectively. The aggregated data resulting from the preprocessing are loaded to SciDB and are subjected to the query processing. The schemas of the generated spatio-temporal array stored in SciDB are represented as follows.

```
Dataset-1<Num:int64>
[X=1:4096;Y=1:4096;T=1:1440]

Dataset-2<Num:int64>
[X=1:4096;Y=1:4096;T=1:900]
```

Both of the two above arrays consist of 3 dimensions as X, Y, T, and the attribute *Num* is the sum of objects of each grid cell.



(a) Dataset-1: Varying n ($B = 10$)

(b) Dataset-2: Varying n ($B = 10$)

(c) Dataset-1: Varying B ($n = 32$)

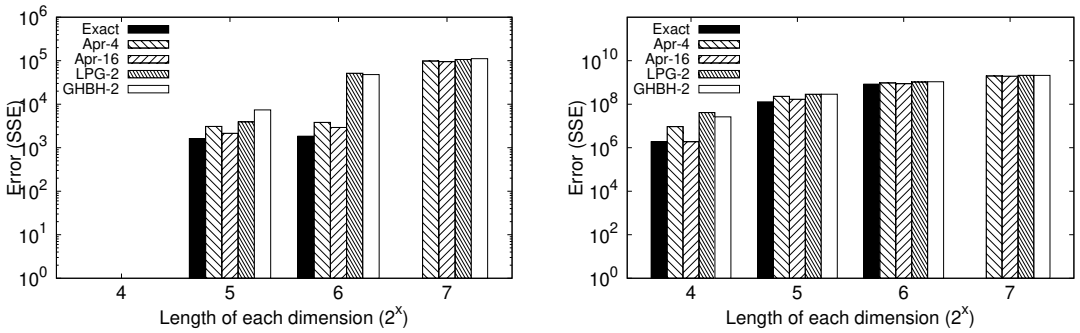
(d) Dataset-2: Varying B ($n = 32$)

FIGURE 3.6: Efficiency (Execution Time)

In addition, we generate synthetic data that contains an array with the uniform distribution of interval $(0, 1)$, as well as others with the Gaussian distribution. We generate two Gaussian distribution array data with the same mean (0) and different variances (0.25 and 1), which is represented as Gaussian-1 and Gaussian-2, respectively.

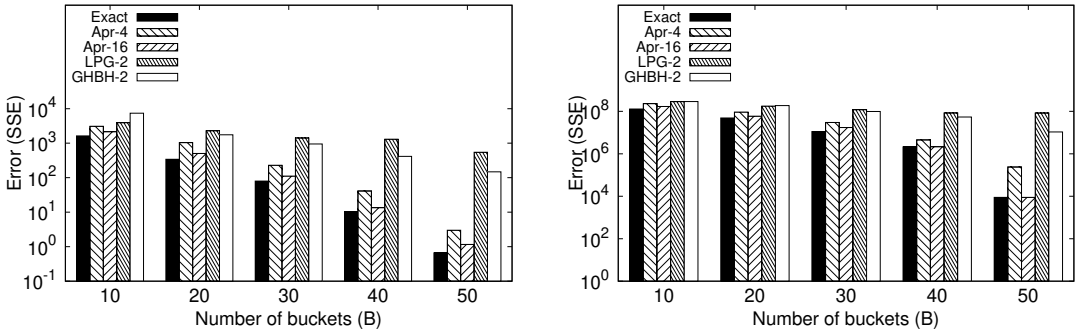
3.5.2 Experimental Setup

We perform extensive experiments to evaluate the quality and efficiency of the proposed methods on a SciDB cluster, the details of which is introduced in Section 3.4. First, we compare the proposed approximate and heuristic algorithms with exact algorithm, as well as *GHBH* (grid hierarchical binary histogram), which is proposed in [63]. Since the computation cost of the exact algorithm is extremely expensive, we generate array data with lengths from 16 to 128 for each dimension, and set the number of buckets varying from 10 to 50. Then, we conduct experiments on large datasets, each dimension length of which varies from 64 to 1024, to evaluate the scalability of the proposed algorithms



(a) Dataset-1: Varying n ($B = 10$)

(b) Dataset-2: Varying n ($B = 10$)



(c) Dataset-1: Varying B ($n = 32$)

(d) Dataset-2: Varying B ($n = 32$)

FIGURE 3.7: Quality (Error)

and *GHBH* on efficiency and quality. In addition, we conduct experiments on synthetic data to measure the robustness of the proposed algorithms under different distributions.

The compared methods are represented in short as in Table 3.1. To evaluate the result quality of *Apr*, we set the parameter L_k as 4 and 16 to compare the errors of result histograms with ground truth, which is computed by the exact algorithm. For the l-grid heuristic algorithm, we set l as 2, which is a general grid-based partitioning, i.e., quadtree-based partitioning. Also, we show the results of *GHBH* when parameter k is set as 2, which performs best. We use a hyphen to concatenate algorithm names and parameter settings, e.g., *Apr-4* means the approximate algorithm with an L_k of 4.

The quality of the algorithms is measured by the errors of result histograms in terms of the input array. We use the sum of squared error (SSE), which is defined in Eq. 3.1 to measure the quality. The efficiency is evaluated by the execution time of the algorithms. We evaluate the efficiency and quality of the proposed methods by varying the dimension length n of the computed array and the number of buckets B.

TABLE 3.1: Symbols of Compared Methods

Proposed methods	Symbol
Exact algorithm	Exact
Approximate algorithm	Apr
Heuristic algorithm based on l-grid partitioning	LPG
Grid hierarchical binary histogram	GHBH

3.5.3 Comparing with Exact Algorithm

3.5.3.1 Evaluation of Efficiency

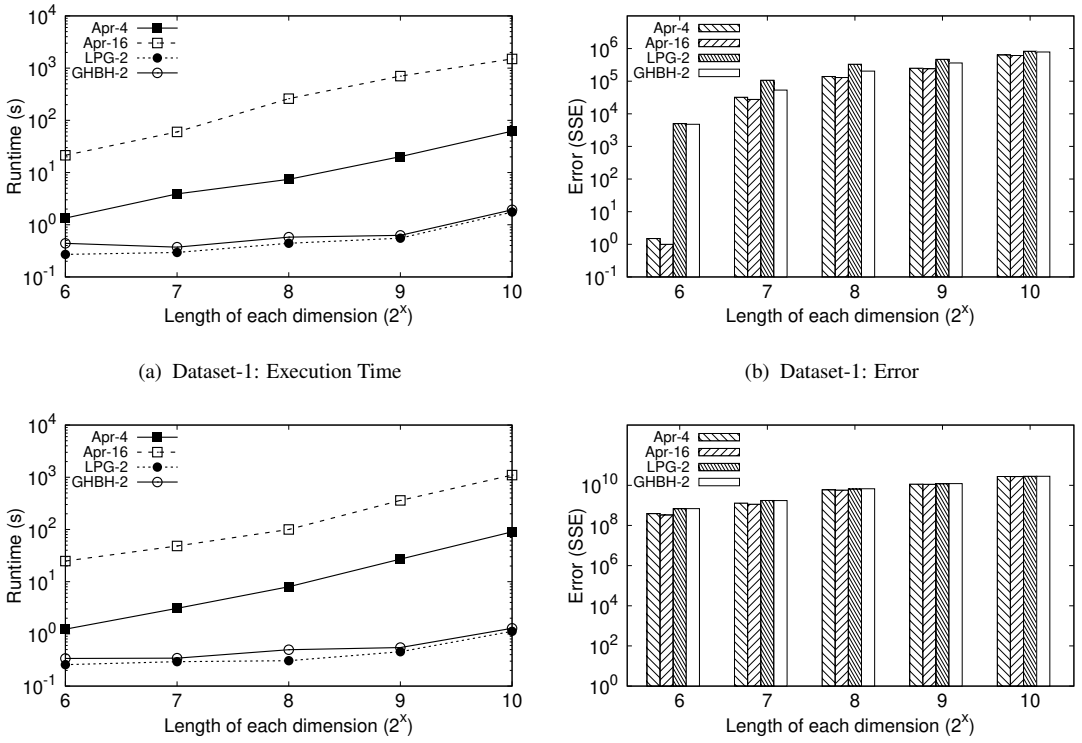
Figure 3.6 shows the execution time of the compared methods by varying the dimension length n (i.e., array size of $n \times n$), as well as the number of buckets B. The result shows that *Exact* is the most time consuming one, and it becomes inexecutable when n is larger than 128, as shown in Figure 3.6(a) and Figure 3.6(b). The reason of its unavailability is that, the generated candidate results by the dynamic programming algorithm exceeds the main memory. As shown in Table 3.2, the space complexity of *Exact* is $O(N^2B)$, which is the most expensive one than other methods.

The result also demonstrates that the approximate and heuristic algorithms are more efficient than *Exact*. For the approximate algorithm, *Apr-16* is much slower than *Apr-4*, since the large parameter L_k indicates larger ϵ of Theorem 3.7 in Section 3.3.2, which affects the time complexity of the approximate algorithm. Also, *Apr* runs slowly than the heuristic algorithms *GHBH-2* and *LPG-2*, but the distinction is minor when data size is small and the parameter L_k is small. As shown in Figure 3.6, the execution time of *Apr-4* is close to *GHBH-2* when data size is small ($n < 128$), and it is not affected by the increase of the bucket number B. Moreover, we can improve the efficiency of *Apr* by decreasing the parameter L_k at the expense of quality.

On the other hand, the proposed heuristic algorithm (*LPG-2*) is faster than *GHBH-2*, and it is the most efficient one, even when $l = 2$. As the time complexity of *LPG* shown in Table 3.2, the larger the l is, the more efficient of *LPG*. Moreover, the increasing of bucket number B does not affect the efficiency of the approximate and heuristic algorithms, since B is much smaller than N.

TABLE 3.2: Computation Cost of Proposed Methods

Proposed methods	Time complexity	Space complexity
Exact	$O(N^{2.5}B^2)$	$O(N^2B)$
Apr	$O(N^{1+1.5\epsilon}B^2)$	$O(N^{1-\frac{\epsilon}{2}}B)$
LPG	$O(B^2l^{-2}t_E)$	$O(B)$



(a) Dataset-1: Execution Time

(b) Dataset-1: Error

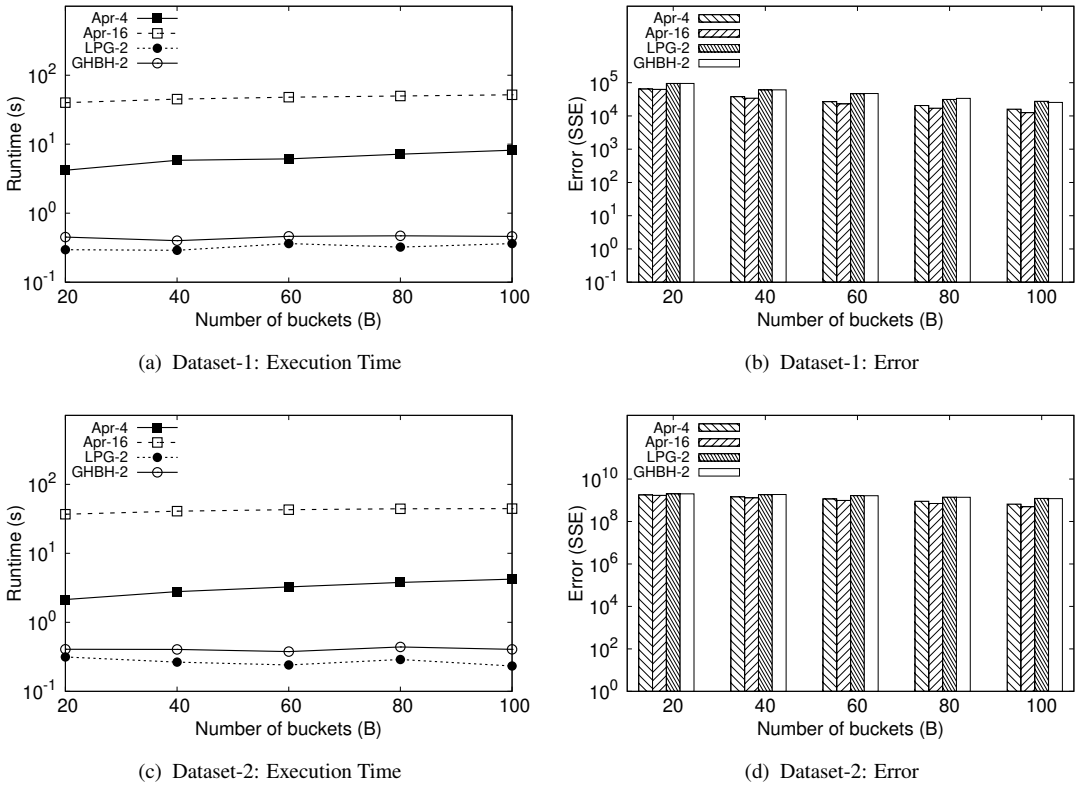
(c) Dataset-2: Execution Time

(d) Dataset-2: Error

FIGURE 3.8: Varying n ($B = 50$)

3.5.3.2 Evaluation of Quality

We also compare the corresponding errors of result histograms (i.e., SSE) as shown in Figure 3.7. The errors increase as n increases or B decreases, which is intuitive. For Dataset-1 as illustrated in Figure 3.7(a), the errors of the compared methods are all 0 when n is 16, due to the small size of array. For the approximate algorithm, the result demonstrates that increasing the parameter L_k makes the result error decrease. It verifies Theorem 3.7 in Section 3.3.2, since large L_k indicates large ϵ, which means the high time complexity as well as high approximation of results. Moreover, both of *Apr-4* and *Apr-16* perform better than *LPG-2* and *GHBH-2*, and the larger the parameter L_k, the closer the results of *Apr* to the ground truth.



(a) Dataset-1: Execution Time

(b) Dataset-1: Error

(c) Dataset-2: Execution Time

(d) Dataset-2: Error

FIGURE 3.9: Varying B ($n = 128$)

Figure 3.7(a) and Figure 3.7(b) show the influence of the array size n on the result errors. When data size is small (e.g., $n = 32$ in Dataset-1 and $n = 16$ in Dataset-2), the errors of *GHBH-2* and *LPG-2* differ slightly, which depends on the target data. As data size becomes larger, the errors of *GHBH-2* and *LPG-2* are almost the same, and their difference with *Apr* also becomes smaller. This indicates that, the larger the array size, the influence of the partitioning strategy becomes weaker on the result errors.

Figure 3.7(c) and Figure 3.7(d) demonstrate that, the distinction between the approximate algorithm and heuristic algorithms becomes larger when B increases. The reason is that, for the heuristic algorithms *LPG* and *GHBH*, larger B indicates more times of partitions, which causes more imprecise on the results. In addition, when B is small ($B \leq 30$), *LPG-2* and *GHBH-2* perform similar with each other, while when B becomes larger, *GHBH-2* performs better than *LPG-2*, due to the flexible structure of *GHBH* with respect to *LPG*.

3.5.4 Comparing Proposed Methods with GHBH

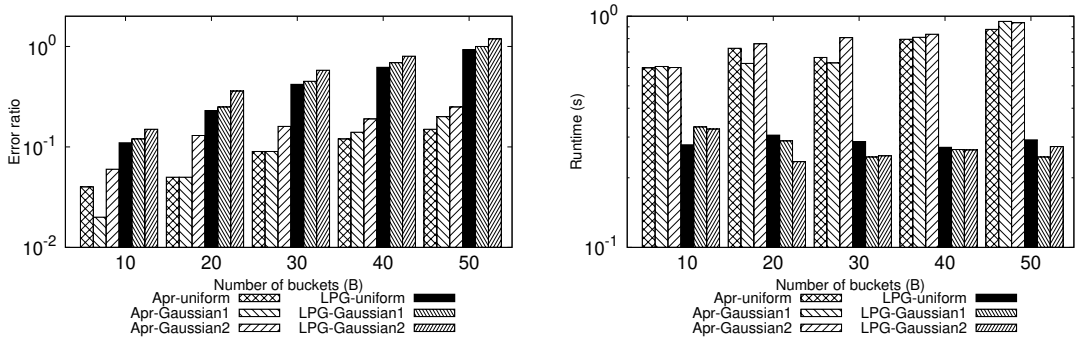
In what follows, we evaluate the efficiency and quality of the proposed approximate and heuristic algorithms as well as *GHBH*, using large size of data, where the length of each dimension varies from 64 to 1024, and the number of buckets changes from 10 to 100.

3.5.4.1 Varying the Array Size ($n \times n$)

Figure 3.8 demonstrates the results of execution time and errors of the compared algorithms by varying n, which is the length of each dimension. In Figure 3.8(a) and Figure 3.8(c), the execution time of the approximate algorithm with different L_k increases with the increase of n. The smaller the L_k is, the faster the approximate algorithm. The reason is also the same with the one explained in the previous section, which is determined by the time complexity $O(N^{1+1.5\epsilon}B^2)$, where small L_k indicates small ϵ. The result also verifies that the efficiency of the heuristic algorithm *LPG* is the best one, and *GHBH-2* is slightly slower than *LPG-2*. On the other hand, Figure 3.8(b) and Figure 3.8(d) show that, the quality of results generated by *LPG-2* and *GHBH-2* is worse than the approximate algorithm, and the distinction between them becomes smaller when array size becomes larger. One of the reasons of the reduced distinction is that the larger the array size is, the more imprecise of the approximate algorithm is, according to Theorem 3.7 The other reason is that, the larger the array size, the influence of the partitioning strategy becomes weaker on the result quality.

3.5.4.2 Varying the Number of Buckets (B)

Figure 3.9(a) and Figure 3.9(c) show the execution time of the compared methods are not sensitive to the number of buckets B. This can be explained their time complexity results in Table 3.2. Since the value of B is a small constant value (here, we consider it as $[10, 100]$), which is much smaller than array size N, the execution time is not sensitive to B. The errors of the results are shown in Figure 3.9(b) and Figure 3.9(d). The approximation of the approximate algorithm is also verified by the fact that *Apr-16* performs better than *Apr-4*. However, the distinction between them is not obvious since the array



(a) Quality (Error Ratio)

(b) Efficiency (Execution Time)

FIGURE 3.10: Data Distributions, Varying B ($n = 16$)

size is large. Meanwhile, the accuracy of the approximate algorithm is better than *LPG-2* and *GHBH-2*, and the larger the B is, the bigger the gap between them. This can be also explained by the imprecise of results caused by the increase of partitions as mentioned before. The result of Dataset-2 is less obvious than Dataset-1, because its larger data density, which leads to the imprecise of result histograms.

3.5.5 Performance on Different Data Distributions

Further, we perform experiments to measure the approximate algorithm (*Apr-4*) and heuristic algorithm (*LPG-2*) on different data distributions. The quality of the methods is measured by the error ratio with the ground truth E_0, i.e., $(E - E_0)/E_0$, where E is the SSE value of the result histogram, and E_0 is computed by *Exact*. Figure 3.10(a) demonstrates that the quality of the two algorithms with Uniform data and Gaussian-1 (Gaussian data with smaller variance) is better than Gaussian-2 (the one with larger variance). This is because the error (SSE) of Gaussian-2 is intrinsically large, which leads to the result with more deviation. Meanwhile, the quality of algorithms on the uniform data and Gaussian-1 is similar. Furthermore, the execution time of them with the three distributions are similar with each other, as shown in Figure 3.10(b).

3.5.6 Discussion

The experimental results verify the efficiency and quality of the proposed approximate and heuristic algorithms. Comparing to the exact algorithm, the execution time of the

approximate and heuristic algorithms is faster, and not as space consuming as the exact algorithm. Meanwhile, the quality and efficiency of the approximate algorithm, which is proved in Theorem 3.7, is also verified by the fact that the larger the parameter L_k is, the more accuracy of the result and more time consuming. We also observed that, when data size is small (e.g., $n < 128$), the efficiency of *Apr-4* is close to the existing heuristic method *GHBH-2*, with significant quality improvement. Moreover, the increase of the number of buckets B makes the distinction of quality more obvious.

For the heuristic algorithm (*LPG*), it runs faster than *GHBH* even when l is 2, with similar result quality. Moreover, *LPG* has good efficiency even if we increase the array size and the number of buckets. Although the accuracy of the results are worse than the approximate algorithm, the larger the array size, the distinction is less obvious. In addition, the efficiency of the approximate and heuristic algorithms is not affected by different data distributions, while the accuracy on data with the uniform distribution and Gaussian distribution with a small variance is better than Gaussian distribution with a large variance.

3.6 Related Work

A *histogram* is one of the popular approaches in summarizing large datasets and often used in database systems [22]. One of the applications of histograms is query cost estimation based on cardinality estimation of a query result [21, 23, 48, 49, 64–68]. Researches in the literature focus on the histogram construction methods and the estimation of attribute values or frequencies, which is not the focus of this paper. For the construction of multi-dimensional histograms, as the cost of constructing optimal histogram is prohibitively large [54], existing techniques use heuristics to partition the data space into buckets, while they do not provide any guarantees on the quality of histograms.

MHIST [49] is based on a greedy strategy *MaxDiff(v,a)*, defined in [64], to partition the buckets in the most need of partitioning. Each partition divides the two consecutive values with the largest "area gap" [64] along one dimension. *MHIST* iteratively conducts partitioning until it reaches a given number of buckets. [63] proposes a *Grid Hierarchical Binary Histogram* (*GHBH*) based on the binary partition schema, which is the most

relevant one to our work. In GHBH, each partition of a block is constrained to be laid onto a grid, which divides the block into a number of equal-size sub-blocks. This number is a parameter of the partition, given by k. The paper also proposes a greedy algorithm to construct a histogram with a given storage bound. Several greedy criteria are used, and the experimental results show that combination of the *Max-Var* and *Max-Red* performs best than other state-of-the-art techniques, such as *MHIST* and *GENHIST* [66, 69].

Histograms considering *spatial and temporal* aspects are also studied for approximate query answering [70–72]. [70] proposed a framework for building and continuously maintaining spatio-temporal histograms. The histogram has an $N \times N$ grid structure, which is progressively refined by query feedbacks, to support continuous queries on dynamic streaming data. [71] studied the approximate query answering for future location-based query, using predictions based on spatio-temporal histograms. [72] evaluated several spatial partitioning techniques by their effects on the performance of spatial queries. However, their focused problems are different from ours and their methods are not adaptive to solve our problem.

In this paper, we consider a general hierarchical histogram based on binary partition, as well as a more specific one (l-grid-based) for summarization. For the general hierarchical structured histogram, we propose a k-hierarchical histogram, which has different constraints on the buckets with GHBH mentioned above. Moreover, we proved and verified the approximation bound of k-hierarchical histogram with respect to the optimal histogram as mentioned in Section 3.3.2. The comparison experiments of our proposed methods and GHBH are shown in Section 3.5.

Another feature of our approaches is the use of an array DBMS, such as RasDaMan [59], ArrayDB [47], SciDB [41, 42, 60] and SAGA [73]. Since array DBMSs are now becoming popular in scientific computing and online analytical processing (OLAP), their effective use is an important issue in the database research area. We have used SciDB for implementing the proposed three algorithms. Using the feature of user-defined operators, we could easily implement our operators.

In addition, as data analytics has recently attracted increasing attention, data visualization is found effective in supporting interactive analyses and many studies on the subject

are now underway. From the database perspective, technologies that can instantly visualize large-scale data or select data to be visualized are important. Since histograms can present the overview of data distribution in a summarized way, visualization is also another popular usage of histograms [55, 74–76]. For example, MuVE [55] visualizes data by bar graphs as a result of their consideration on the viewpoints that will concentrate data into specified conditions remarkably different from the whole data. SeeDB system for the visualization of databases, though intended for category attributes, is also closely related to this study [74–76].

3.7 Conclusions and Future Work

In this paper, aiming to realize the advanced analysis functionality of large-scale spatio-temporal array data, we defined a spatial V-Optimal histogram and propose histogram construction algorithms for hierarchical partitioning. We proposed exact and approximate algorithms with approximation bounds, as well as a heuristic approach to further speed up the histogram construction for data exploration. To improve the efficiency of the proposed methods on massive data, we implemented them on SciDB, the state-of-the-art array DBMS.

In order to verify the effectiveness and efficiency of our methods, we conducted experiments on the massive evacuation simulation data and real taxi data, as well as synthetic data with different distributions. Experimental results demonstrate that the proposed approximate and heuristic algorithms perform well in efficiency than the exact algorithm, and also verify the approximation quality of the approximate algorithm. The approximate algorithm performs best on quality with similar efficiency comparing to other heuristic algorithms when data size is small. As data size increases, the proposed heuristic algorithm performs best on efficiency with similar result quality with the approximate algorithm.

In this work, we treat the spatio-temporal data as a series of spatial data, and introduced aggregation and summarization approaches by constructing histograms on spatial data. However, methods for appropriately setting the time interval adapting to the requests of analysis is on demand. In future work, we consider to construct a three-dimensional

spatio-temporal histogram that deals with both spatial and temporal dimensions, which has not been studied in the literature as far as we know. Moreover, we also intend to solve the problem of updating histograms for continuous queries of data exploration. To enable interactive analysis of large-scale spatio-temporal array data, developing techniques on array DBMS that improving the efficiency of query processing is also in demand.

Chapter 4

Difference Analysis of Spatio-Temporal Data on Array DBMS

4.1 Introduction

As big data attracts attention in a variety of fields, research on data analytics for sophisticated analytic processing of a large amount of data in a database has gained popularity [10, 77]. As for spatio-temporal databases, there are growing demands for analyzing large-scale spatio-temporal data in various domains such as mobility data, moving trajectory data, and scientific data [43, 78]. In scientific fields, a lot of simulations are conducted for the purpose of predictions, decision making, etc. For instance, disaster simulations like human evacuation simulations and tsunami simulations are conducted for effective humanitarian relief and disaster management [2]. Analysis of disaster simulation data can achieve various objectives, such as interesting patterns discovery, and decision support like suggesting an appropriate location as a shelter of disasters [3]. Moreover, since large amounts of simulation data are generated by those simulations with different conditions and parameters, *data warehouse* techniques that enabling massive data storage and exploration are in demand. Our research group has been engaged in researches on spatio-temporal data warehouses where large-scale spatio-temporal simulation data are

specially stored to enable interactive analyses, which are referred to as simulation data warehouses [79].

This study specifically examines *differences* as one of the basic analytic requirements for *spatio-temporal data warehouses*, in which detection of temporal changes, as well as differences between observation datasets with different parameters or conditions should be required. Consider that an earthquake analyst intends to explore the evacuation simulation data after an earthquake. In this case, a query like "Return the major change in the distribution of evacuees between the first hour and the second hour after the earthquake occurs." is considered to be useful to understand the variation trend in the movement of evacuees.

Various types of difference operators are possible, depending on the properties or application purposes of the target data. However, it is not clear which types of difference operators are appropriate for the above-mentioned data analysis. In this paper, we define a general-purpose operator based on the notion of histograms [21–23, 48], which are widely studied for the purpose of query optimization and selectivity estimation in database systems. Different from the main focus of those work, which is to improve the accuracy of estimations, our work aims to enable exploratory and interactive analysis of big simulation data. The difference operator studied in this paper focuses on both usability in visual analytics and accuracy of histograms.

An intuitive example of the difference operator is shown in Figure 4.1. The left and middle ones represent the aggregate results of evacuees in time intervals T_1 and T_2, respectively. The intensity of cells correspond to the aggregate values (the number of moving users in the cell during the specified period of time). One can choose to use the aggregate values at the two time intervals T_1, T_2 as they are or to use the normalized aggregate values as divided by the total number, depending on object data and their applications. In this paper, it is assumed for the sake of simplicity that users have selected the former method. The figure on the right side of Figure 4.1 approximately represents the differences between the aggregate results of T_1 and T_2. In the heat map representation, the more the color of the cell is "hot" (red is the extreme one), the larger the increase of

aggregate values from T_1 to T_2. Meanwhile, the "cold" (blue in the extreme case) regions correspond to large decrease from T_1 to T_2. In order to represent a rough trend of differences, any adjacent cells with a similar trend of differences are merged into one cell. Given quadtree-like spatial divisions to make the sides of a cell equal in size to the power of 2 in length. Presentation of such output results enables users to easily grasp any differences between two different time segments.

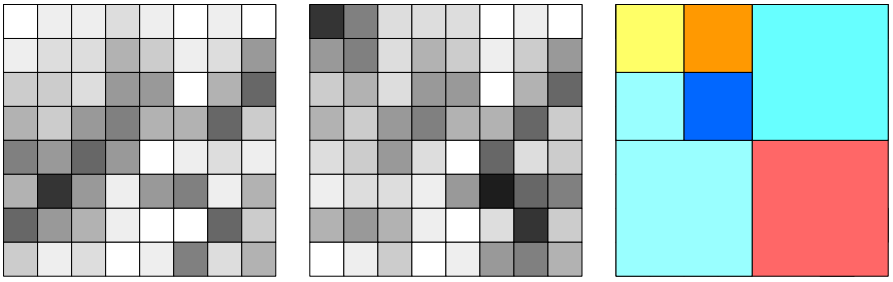


FIGURE 4.1: Image of Difference Operator

On the other hand, due to the challenges in detecting any remarkable changes within a large amount of data, it is necessary to develop efficient algorithms by effectively using the latest database system technology. As scientific data is usually represented as 2-D (or 3-D) array structure, array-based representation is more appropriate for the representation [58]. In this paper, we propose difference histogram construction methods, and implement the algorithms on SciDB [41, 42, 60], which is an open source array-oriented DBMS for the efficient storage and manipulation of large array data.

Our contributions can be summarized as follows:

- We study the problem of detecting differences among massive spatio-temporal data, and define the difference histogram based on the quadtree-based structure. (Section 4.2).
- We propose a general framework for constructing histograms on SciDB, the state-of-the-art array DBMS, as well as both exact and heuristic histogram construction algorithms. (Section 4.3).
- We discuss the overview of SciDB and our configuration on SciDB. (Section 4.4).

- We conduct extensive experiments on massive simulation data to verify the effectiveness and efficiency of the proposed methods, as well as a case study of the difference histogram (Section 4.5).

4.2 Problem Definitions

For the purpose of analyzing spatio-temporal array data, which is defined as Definition 3.1 in Section 3.2, we focus on the variation of data distribution by time and spatial dimensions, which is considered as one of the most important requirements of data analytics. Therefore, we formulate the problem of detecting differences at both spatial and temporal scales as below.

We define some notions before the problem definition.

Definition 4.1 (Aggregated Array)**.** Given a spatio-temporal array $\mathcal{A}$ and a time interval T, an *aggregated array* for T, $\mathcal{A}[T]$, is a two-dimensional array such that each (x, y) element value of $\mathcal{A}[T]$, $\mathcal{A}[T](x, y)$, is obtained by applying the aggregate function to all the (x, y, t) element values of $\mathcal{A}$ ($t \in T$).

The aggregate function depends on the target dataset and application, but the difference operation, applied next, should be meaningful. In our example of tsunami disaster simulation, $\mathcal{A}(x, y, t)$ corresponds to the number of evacuees within the spatial cell $cell(x, y)$ for time t. Therefore, the *sum* aggregate function is used, and $\mathcal{A}[T](x, y)$ is the total number of evacuees within $cell(x, y)$ while time interval T.

Definition 4.2 (Difference Array)**.** Given a spatio-temporal matrix $\mathcal{A}$ and two time intervals T_i and T_j, the *difference array* of aggregated arrays $\mathcal{A}[T_i]$ and $\mathcal{A}[T_j]$, $\mathcal{DA}_{ij}$, is defined by $\mathcal{DA}_{ij}(x, y) = \mathcal{A}[T_j](x, y) - \mathcal{A}[T_i](x, y)$ for each (x, y) element.

In order to catch an insight of difference arrays, we use the notion of *histograms* from the studies on selectivity estimation and query optimization in the database area. As the most of histogram construction problems in the literature of database are proved to be

NP-hard [54], we exploit a *quadtree-based hierarchical structure* to construct a difference histogram. Next, we formulate the notion of a difference histogram. It is based on hierarchical space partitioning applying the idea of quadtree [80].

Definition 4.3 (Difference Histogram)**.** Given a spatio-temporal array $\mathcal{A}$, two time intervals T_i and T_j, an integer B defining the size of a bucket, and an error metric $E()$, the *difference histogram* H of the difference array $\mathcal{DA}_{ij}$ consists of a set of buckets $\{b_1, b_2, \ldots, b_B\}$, generated by partitioning the whole spatial region of $\mathcal{DA}_{ij}$ into B non-overlapping buckets based on the quadtree-based partitioning. Each bucket b_l ($1 \leq l \leq B$) has a corresponding rectangle area $b_l.area$, and an aggregated value $b_l.val$. The set of elements that belong to a bucket b_l is represented as $Elem_l$, which satisfies the following conditions: $\{\forall e \in Elem_l \mid e \in \mathcal{DA}_{ij} \wedge e.area \in b_l.area\}$. The value of $b_l.val$ is calculated by averaging elements in $Elem_l$:

$$b_l.val = \frac{\sum_{e \in Elem_l} e}{|b_l.area|} \tag{4.1}$$

While the error metric is chosen by different purposes, we use *Sum Squared Error* (SSE), as defined in Definition 3.1 in Section 3.2, which is a common error metric for measuring difference between two data distributions.

4.3 Quadtree-based Histogram Construction Methods

In this work, we propose histogram construction methods based on a *quadtree-based hierarchical structure*. The overall spatial region corresponds to the root of the quadtree at depth 0, and each leaf node at the largest depth represents an element in the difference array, which stores the count information (the number of evacuees in our example). In addition, every internal node in the quadtree contains four child nodes. Every partitioning of an internal node generates four child nodes if they are non-empty cells.

Intuitively, there are two ways of constructing histograms, that is, top-down partitioning and bottom-up merging. Considering the case of non-empty cells, given array size N (number of grids of the overall spatial region, i.e., $2^d \times 2^d$) and the number of buckets

B, since every partition(or merge) can generate(or reduce) 3 more(or less) buckets, the number of partition times P(or merge times M) can be known as $\left\lfloor \frac{B-1}{3} \right\rfloor$ (or $\left\lfloor \frac{N-B}{3} \right\rfloor$). Both top-down partitioning and bottom-up merging cause large computation cost if array size N or bucket number B is large. For instance, the computation cost of top-down partitioning includes the minimal number of possible combinations of partitioning, which is computed as $4\times7\times10\times...\times(B-3)$, as well as the cost of computing the optimal solution of every partitioning. The computation cost increases if the array is sparse, since the number of partitioning times increases. In the worst case, there are $B!$ possible partitioning for a given region.

Considering the large computation cost described above, in the following part, we propose a general framework for constructing histograms based on the quadtree structure, taking advantages of an array DBMS as data storage backend. Moreover, we propose an exhaustive algorithm as well as a greedy algorithm which can be efficiently implemented. In addition, we present a hybrid approach of the exhaustive and the greedy algorithms for taking trade-offs of two algorithms.

4.3.1 SciDB Functionality and Histogram Construction Framework

Since scientific data typically have spatio-temporal structure, indexed arrays are often used to represent scientific data, while the traditional relational model cannot handle such array data [58]. Some array database management systems are developed to support scientific computing and online analytical processing (OLAP), such as RasDaMan [59], ArrayDB [47], SciDB [41, 42, 60], etc.

In this work, we exploite one of the state-of-the-art array DBMSs, SciDB, as the storage backend of the target spatio-temporal array data, and also implement the histogram construction approaches as User-Defined Aggregates (UDAs) on SciDB. The UDAs are executed by the built-in operators such as `regrid` operator of SciDB. For example, `regrid(H, 2, 2, myUDA(attr_name))` conducts merging with block size of 2×2 on attr_name of input array H. Built-in operators like `regrid` are executed efficiently on

Algorithm 2: Histogram construction framework on SciDB

Input: A: input array, B: number of buckets, dep_{max}: the maximum depth of the quadtree

Result: H: output array of result histogram

```
1 H_0 ← initialize(A, dep_max) ;                    // Histogram initialization
2 H ← regrid(H_0, 2, 2,leafMerge(leafNode) as interNode) ; // Merge leaf nodes
3 dep_cur ← dep_max − 1 ;                           // Depth of parent nodes
4 while dep_cur − 1 > 0 do
5 |   H ← regrid(H, 2, 2,internalMerge(interNode) as interNode) ;   // Merge
  |    internal nodes
6 |   dep_cur − −;
7 return H;
```

SciDB. Using the bottom-up construction framework described as follows, we can fully utilize the query processing power of SciDB.

The pseudocode of implementing the histogram construction algorithms on SciDB is shown in Algorithm 2, which can be easily extended to implement other construction strategies. The inputs of the difference operator contain a difference array A, the desired number of buckets B, and the maximum depth dep_{max} of the quadtree. The algorithm returns a result histogram, which is stored as an array with only one element, i.e., the solution of the root node.

In the proposed framework, we implement the histogram initialization phase as a UDO and merging phases as UDAs. In details, we implement a UDO *initialize* for the initialization phase to transfer the numeric attribute values to leaf nodes (Line 1). The histogram construction methods conduct merging in a bottom-up style until reaching the root node. The merging strategies are implemented as UDAs, *leafMerge* and *internalMerge* for merging leaf nodes and internal nodes, respectively (Line 4-6). Note that, other UDAs that conduct merging by other strategies can also be implemented in the same way.

4.3.2 Exhaustive Approach

We briefly describe a baseline algorithm for constructing a histogram. It is based on the exhaustive approach and finds the *optimal* histogram. We assume that the dataset is already available in the DBMS with a grid structure. The input of the algorithm is an

array A_d with grid size $2^d \times 2^d$ at level d of the quadtree. Each element (grid cell) of the array A_d contains a list of solutions, each of which is a bucket set with bucket number b of the corresponding area, where $b \in [1, b_{max}]$. b_{max} is the maximum number of buckets of the target area of the element.

For each of the nodes in level $d-1$ of the quadtree, it iteratively generates a list of solutions with bucket number $b \in [1, b_{max}]$. First, it create one bucket by combining the four children of it at level d. Then, it creates four buckets for the four children with the total error of them. The generation step continues by choosing the optimal solution with different bucket arragement, until the number of buckets reaches the budget b_{max}.

Based on the construction framework explained in the previous section in Algorithm 2, the exhaustive approach can be implemented as an *internalMerge*. The total cost of the histogram construction depends on the number of candidates, which is $O(2^d)$, as well as the computation cost of the errors of target buckets. We can reduce the computation cost t_E of the error function, which is called many times in the process.

The upper bounded of t_E is $O(N)$, we aim to reduce it to $O(1)$ by using the aggregation functionalities of SciDB. We temporally store computed aggregated values for each nodes and compute required statistics values for the error function incrementally based on the following equations.

$$n = \sum_i n_i \tag{4.2}$$

$$m = \frac{\sum_i m_i n_i}{n} \tag{4.3}$$

$$E(n) = \sum_i E_i + \sum_i m_i^2 n_i - m^2 n \tag{4.4}$$

where, $m, n, E(n)$ represent the average, number of leaf nodes, and error metric value of node n, respectively, and m_i, n_i and E_i are the corresponding values of node n's child

nodes. Using the above equations, the error metric values of parent nodes can be computed based on their child nodes, rather than all of the contained leaf nodes. Therefore, the cost of t_E is decreased to $O(1)$.

4.3.3 Greedy Approach

As described above, the computational cost of the exhaustive algorithm is quite expensive, even using the aggregation functionalities of SciDB. To improve the response time, we propose an efficient *greedy* approach. The algorithm is also adaptive to the bottom-up framework of the implementation on SciDB.

The basic idea is to iteratively partition the target region of a given node n into k subregions, where $k \in \{1, \ldots, b_{max}\}$. The value of b_{max} is the maximum number of buckets the node is able to contain, which is computed by the smaller value of limit number of buckets (B) and the number of leaf nodes covered by the node ($size(n)$). For each partition of the node, we choose the one with the maximum error reduction *maxRed* from the candidate results of its child nodes. The process finishes when the number of buckets is larger than k. Note that, we also store the candidate results of nodes to compute the candidate results of its parent node based on Equation 4.24.34.4, in order to reduce the computation cost of multiple accesses of covered leaf nodes.

4.3.4 Hybrid Approach

The *hybrid* approach is based on the following observation: In general, the larger the depth of a given node in a histogram, the smaller the error of the node compared to its ancestor nodes. Which means that if we employ the greedy approach for the deep level of the complete quadtree, the quality of the resulting histogram would be less effected. We empirically set a depth threshold δ. We use the exhaustive approach when the depth is less than δ and use the greedy approach otherwise. The depth threshold δ is chosen to take trade-offs of the two proposed algorithms, the larger the δ is, the more accuracy the result histogram is, vice versa. The experimental evaluation is shown in the experiment part (Section 4.5).

4.4 Overview of SciDB and Our Configuration

SciDB is a parallel database system based on the shared nothing architecture, and designed specifically for scientific data management and analysis. In SciDB, array data is organized on disks as *chunks*, which also work as the I/O units. Chunking ensures that logically co-located cell values are clustered into the same physical block, and enables efficient query processing. Chunks are distributed among all instances of a SciDB cluster. SciDB uses a variety of functions for chunk mapping, and the default is the hash-based distribution. During query execution, each instance node executes query locally with the partial array data, and transfers intermediate results across nodes when global computing is necessary.

SciDB supports a variety of operators for array data, such as data extraction (e.g., *slice*, *between*) and aggregation (e.g., *aggregate*, *regrid*). For aggregate operators, a lot of aggregate functions (e.g., sum, min, and max) are supported. In addition to built-in operators and aggregate functions, users are also allowed to implement user-defined operators (UDOs) and user-defined aggregates (UDAs) using C++ language based on the plugin mechanism. Once a plugin operator is loaded into the SciDB library system, it can perform in the similar way with the built-in operators of SciDB for data processing. SciDB also provide an R interface for statistic computing and visualization, which enables the case study of the difference analysis studied in this work.

Our experiments are performed on a SciDB cluster, which consists of a coordinator server (CPU: Intel Xeon E5-2637 v4 @ 3.50GHz $\times$ 2, RAM: 128GB, OS: Ubuntu 14.04 LTS 64bit) and 3 worker servers (CPU: Intel Xeon E5620 @ 2.40GHz $\times$ 2, RAM: 32GB, OS: Ubuntu 14.04 LTS 64bit), each with 2 servers. There are totally 10 SciDB instances that work in parallel. The used version of SciDB is 16.9.

4.5 Experiments

4.5.1 Datasets

The target data sets used in the experiments are evacuation simulation results in the event of large-scale earthquakes in Kochi City and Kantou area in Japan, respectively. Dataset-1 is the sample evacuation simulation data of six hours compiled under the conditions in which an earthquake occurs at 9 a.m., and the evacuations peak is set as sixty minutes after its occurrence. The simulation data is about 40,000 people's evacuations based on person trip data. Dataset-2 contains 194,226,288 records of evacuees' mobility data during 24 hours after an earthquake occurs.

Each simulation data is a collection of records in the format of $(id, time, x, y)$, where id denotes user ID, $time$ is a time stamp, and (x, y) is the location of a user at $time$. Simulation data are static data that remain unchanged after simulations. In this work, in the query processing on data warehouses, it is generally taking advantages of static simulation data, to preprocess the data and make interactive processing more efficient. We preprocess the simulation data by dividing the spatial region with a maximum grid size of $2^{12} \times 2^{12} = 4,096 \times 4,096$ (i.e., $d = 12$), and the number of evacuees in each cell is aggregated every minute. The aggregated datasets resulting after the preprocessing are loaded into SciDB and they are used as the targets of query processing of the difference operator.

The schema of the generated spatio-temporal arrays stored in SciDB are represented as follows.

```
Kouchi_eva<Num:int64>
[X=1:4096:0:4096;Y=1:4096:0:4096;T=1:360:0:256]

Kantou_eva<Num:int64>
X=1:4096:0:4096;Y=1:4096:0:4096;T=1:1440:0:1024]
```

TABLE 4.1: Symbols of Proposed Methods

Proposed methods	Symbol
Exhaustive approach	Exact
Greedy approach	Greedy
Hybrid approach	Hybrid_3, Hybrid_4, Hybrid_5

Both of the two above arrays consist of 3 dimensions as X, Y, T, and the attribute *Num* is the sum of evacuees of each grid cell. Considering the data sparsity of arrays, the non-empty cells of Dataset-1 and Dataset-2 are about 144,169 and 879,333, respectively.

4.5.2 Experimatal Results

We conduct experiments to evaluate the effectiveness and efficiency of the proposed methods. For the evaluation of efficiency, the total execution time for generating difference arrays and constructing histograms, as well as the histogram construction time are considered. For the evaluation of effectiveness, we used the ratio of error values of a histgram computed by the greedy (or hybrid) method with an "optimal" histogram computed by the exhaustive method—the ratio is computed as $(E - E^*)/E^*$, where E is the error value of a greedy (or hybrid) histogram and E^* is the error of an exhaustive histogram..

In what follows, experimental results are presented by varying the array size $2^d \times 2^d$ and the number of buckets B. As shown in Table 4.1, *Exact* represents the exhaustive algorithm, and *Greedy* is the greedy approach. For the hybrid approach, *Hybrid_3*, *Hybrid_4*, *Hybrid_5* correspond to the approaches when threshold δ is set as 3, 4, and 5, respectively.

4.5.2.1 Varying the Array Size ($2^d \times 2^d$)

Using different granularity of time intervals to aggregate Dataset-1 and Dataset-2, difference arrays with different numbers of non-empty cells are generated. Figure 4.2 and Figure 4.3 show that the execution time increases as the array size increases. The execution time of the exhaustive algorithm is larger than the execution time of other two algorithms.

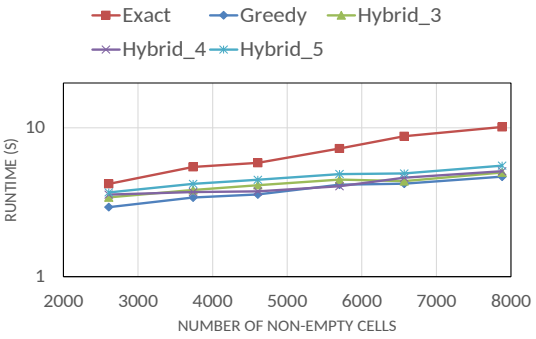


FIGURE 4.2: Runtime: Varying Array Size (Dataset-1, $B = 50$)

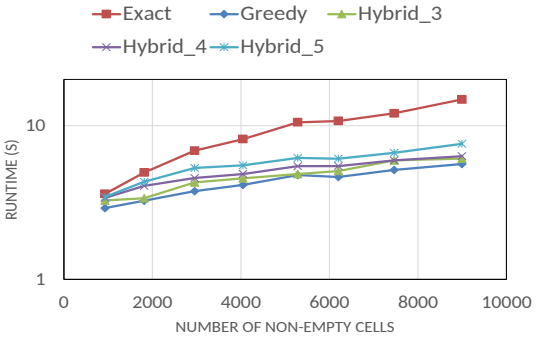


FIGURE 4.3: Runtime: Varying Array Size (Dataset-2, $B = 50$)

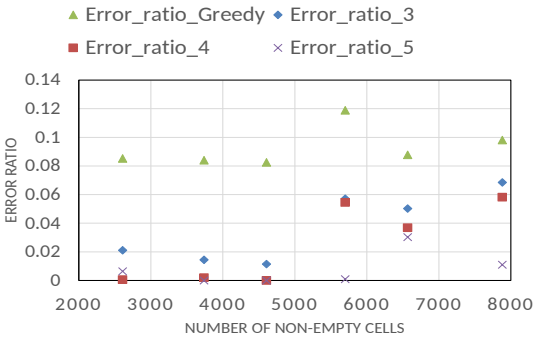


FIGURE 4.4: Error Ratio: Varying Array Size (Dataset-1, $B = 50$)

FIGURE 4.5: Error Ratio: Varying Array Size (Dataset-2, $B = 50$)

In Figure 4.4 and Figure 4.5, the corresponding error ratios are presented. The quality scores of resulting histograms follow the ordering of Hybrid_5 > Hybrid_4 > Hybrid_3 > Greedy (greater is better). This is because that the larger the depth threshold of the hybrid approach, the more accurate the result is. We can conclude that the execution time of the hybrid algorithm is close to that of the greedy algorithm, while the quality of the hybrid algorithm is better than the greedy algorithm when δ is 4 or 5.

4.5.2.2 Varying the Number of Buckets (B)

Next, we conduct experiments on two arrays of size $2^{12} \times 2^{12} = 4,096 \times 4,096$ with different densities. The two arrays called diffArray_sparse and diffArray_dense contain 8,460 and 15,248 non-empty cells, respectively.

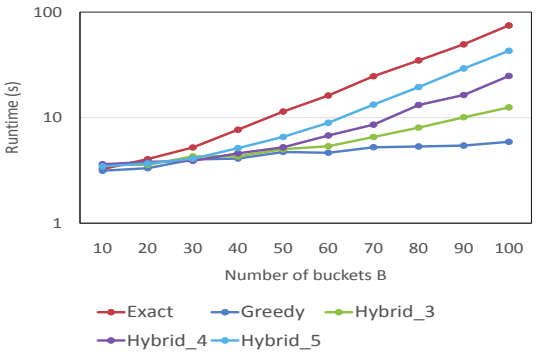


FIGURE 4.6: Runtime: Varying B (diffArray_sparse)

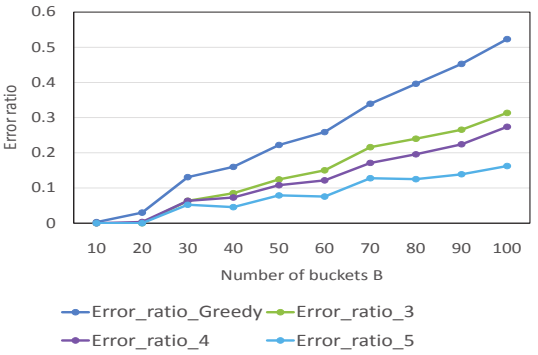


FIGURE 4.7: Error Ratio: Varying B (diffArray_sparse)

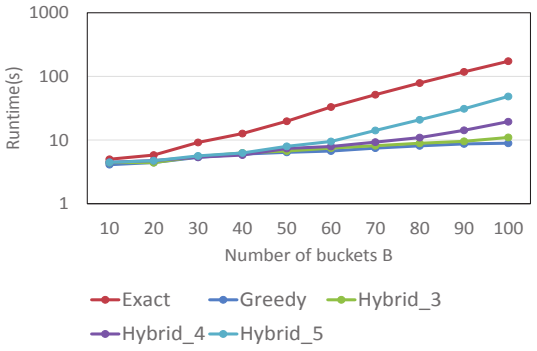


FIGURE 4.8: Runtime: Varying B (diffArray_dense)

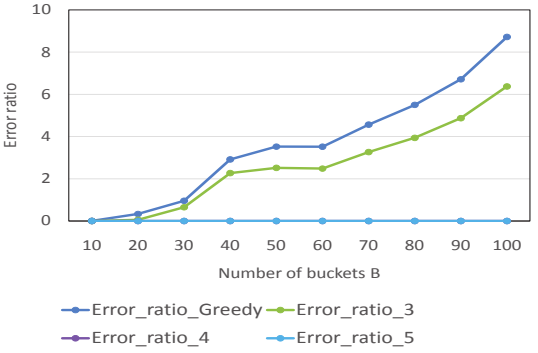


FIGURE 4.9: Error Ratio: Varying B (diffArray_dense)

Figure 4.6 and Figure 4.7 show the execution time and error ratio when varying B for diffArray_sparse. We can see that both of execution time and error ratio increases when B increases. Also, the effectiveness of the greedy algorithm is the best, while the quality of the hybrid algorithm is good even when B is large.

The results for diffArray_dense are shown in Figure 4.8 and Figure 4.9. The execution time follows the similar trend as in Figure 4.6, while Hybrid_4 performs similarly with the greedy algorithm. Meanwhile, the quality of Hybrid_4 and Hybrid_5 are good even the number of buckets increases. Comparing to diffArray_sparse, both the effectiveness and efficiency of Hybrid_4 are good for diffArray_dense, in the case when the data density is higher.

4.5.3 Case Studies

In this section, we demonstrate a case study of the difference histogram proposed in this work. The case study is implemented in R (programming language) and executed via the R interface provided on SciDB.

In the processing of the difference analysis, the `subset` operator and `aggregate` operator that are provided by the R interface of SciDB extract all aggregated information on corresponding cell sets based on the specified conditions for the spatio-temporal domains. In the process of taking direct differences between two array data, first calculate their connections by using the `merge` operator, and set any cell connections with a null value as 0. Next, generate difference arrays by taking differences in attributes, and then apply the above-mentioned histogram construction approaches, and the results are represented in raster graphics by the SciDB's `raster` operator and further processed into heat maps.

Figure 4.10 shows an image of mass evacuation data, which represents the aggregated mass evacuation frequencies (the number of evacuees per area) at the time segments $[9:00, 15:00]$ for each cell after roughly dividing the areas around Kochi City subject to the simulations into 64×64 cells. The central part of the data corresponds to the central part of Kochi City with the coast line running below. The figure shows high evacuation frequencies in the vicinity of the city's central part. However, temporal changes in the mass evacuations cannot be identified from the figure.

Then, we utilize the following query for the case study.

> Seek the difference histograms of the differences in the distribution of evacuees at time segments $T_1 = [9:00, 10:00]$ and $T_2 = [11:00, 12:00]$ in the entire area of the evacuation simulation data.

Figure 4.11 and Figure 4.12 show visualized data of the total numbers of evacuees at the time segments T_1 and T_2 respectively. They represent input data for the internal processing of the difference operator. These figures show that as compared with the distribution of

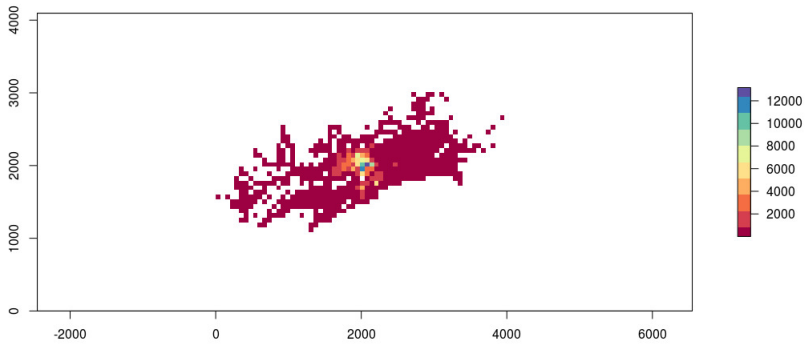


FIGURE 4.10: Evacuation Data Distributions

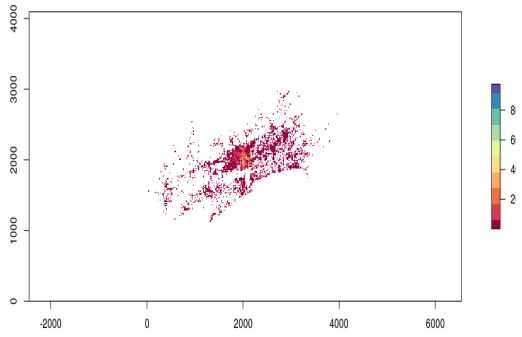


FIGURE 4.11: Evacuation Data at $T_1 = [9:00, 10:00]$

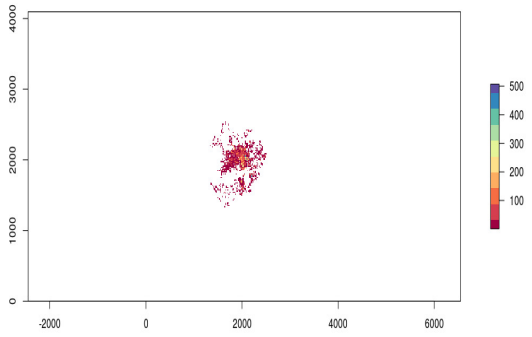


FIGURE 4.12: Evacuation Data at $T_2 = [11:00, 12:00]$

evacuees at the time segment T_1, the total number of evacuees at the time segment T_2 is smaller and is differently distributed. However, a simple comparison of the two figures barely reveals any specific changes between them. Therefore, these two array data are subjected to difference processing to analyze the actual changes. Figure 4.13 displays directly obtained differences in the values of the two time segments. It shows detailed data than but is a little too finely grained to grasp an overall trend of mass evacuations.

Figure 4.14, Figure 4.15 and Figure 4.16 show the results of difference histograms with $B = 20$, $B = 50$ and $B = 100$, respectively. The difference histograms are visualized by heat maps.

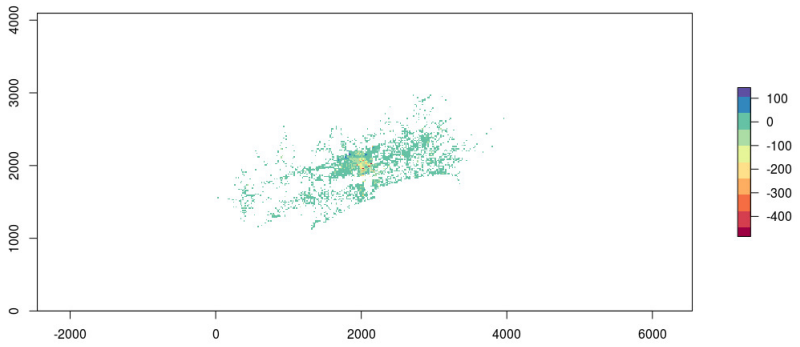


FIGURE 4.13: Direct Differences at Time Segments T_1 and T_2

In Figure 4.13, the parts with significant difference values are first filtered with a minimum bounding box before constructing histograms that approximate their overall trends. Basically, the larger the number of cells B, the more approximated will be the results of difference histograms to actual data distributions. However, for the sake of visualization, cells with similar trends are displayed integrally and the parts with more remarkable differences are highlighted by histograms. For example, the histogram results in Figure 4.14 show the regions where the evacuees have most increased or decreased. If analyzers want to obtain more detailed information, they can just increase the value of B and return the histograms for the system to process.

Since more cells do not always generate better visualization effects, analyzers should carry out interactive analyses by coordinating object regions and settings of B until they can obtain useful information. For instance, as in Figure 4.16 ($B = 100$), the central part is not clearly displayed when B has a large value. In such cases, analyzers can iterate the difference operator limited to the green rectangular region R_1 for the number of cells $B = 100$. Then the difference histograms limited to the region R_1 can be obtained as shown in Figure 4.17, from which more detailed difference distributions can be understood. Thus, the approaches proposed in this study proves to be capable of detecting remarkable differences in any local, large differences by seeking difference histograms of limited regions and visualizing them as an enlarged view.

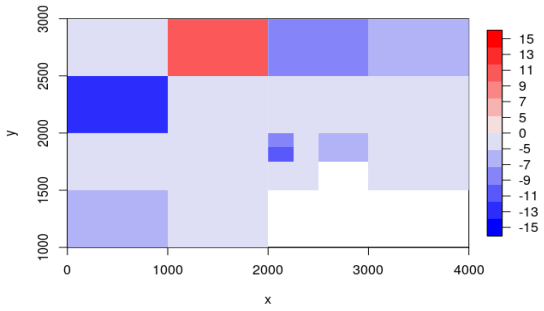


FIGURE 4.14: Results of the Difference Histogram ($B = 20$)

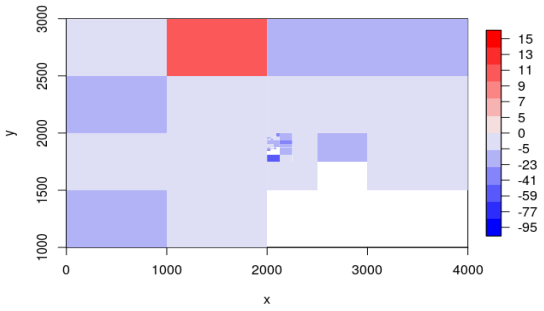


FIGURE 4.15: Results of the Difference Histogram ($B = 50$)

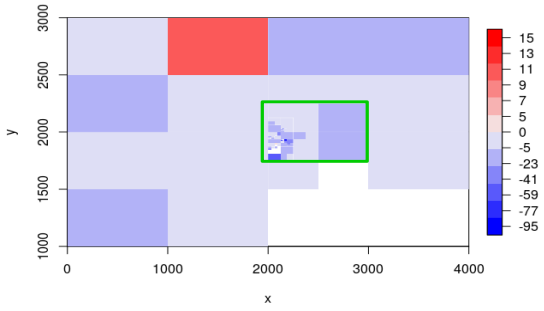


FIGURE 4.16: Results of the Difference Histogram ($B = 100$)

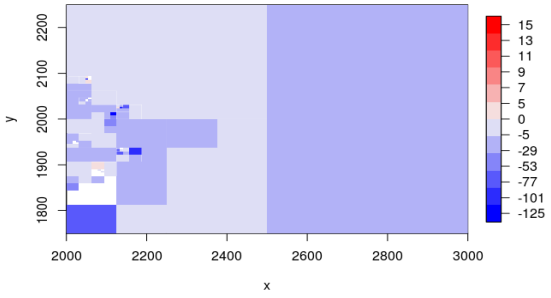


FIGURE 4.17: Difference Histogram Limited to Region R_1 ($B = 100$)

4.6 Related Work

As data analytics has recently attracted increasing attention [77], data visualization is found so effective in supporting interactive users' analyses that many studies on that subject are now under way. From the database point of view, technologies that can instantly visualize large-scale data or select data to be visualized are important. For example, MuVE [55] visualizes data by bar graphs as a result of their consideration on the viewpoints that will concentrate data into specified conditions remarkably different from the whole data. SEEDB system for the visualization of databases, though intended for category attributes, is also closely related to this study [74].

A *histogram* is one of the popular approach in summarizing the large datasets and often used in database systems [22]. One of the application area of histograms is query cost estimation based on cardinarity estimation of a query result. Another popular usage of histograms is visualization since they can present the overview of data distribution in

a summarized way. In our paper, we have used histograms for summarizing the overall structure of the differences between two 2-D arrays. Since construction of accurate multi-dimensional histograms are prohibitively large [21, 23, 48], we consider a ristricted quadtree-like structure for summarizing difference arrays.

Another feature of our approach is the use of an array DBMS, such as RasDaMan [59], ArrayDB [47], SciDB [41, 42, 60] and SAGA [73]. Since array DBMSs are now becoming popular in scientific computing and online analytical processing (OLAP), their effective use is an important issue in the database research area. We have used SciDB for implementing the proposed three algorithms. Using the feature of user-defined operators, we could easily implement our operators.

4.7 Conclusions

In order to realize sophisticated analytical functionalities, we proposed the difference histograms to detect any differences in simulation data with spatio-temporal characteristics. We proposed a general framework for histogram construction utilizing the functionalities of SciDB, as well as both optimal and heuristic approaches to construct difference histograms. We evaluated the proposed algorithms by conducting experiments on massive evacuation simulation data to verify the effectiveness and efficiency of our methods. Experimental results demonstrate the efficiency as wells as the effectiveness of the proposed approaches. We also conducted a case study by visualizing such difference hisograms. We discussed the effectiveness of the histogram constructing approaches by the visualization.

Chapter 5

Conclusions and Future Work

In this Chapter, we conclude this thesis in Section 5.1, and present several future topics in Section 5.2.

5.1 Conclusions

In this thesis, we studied the data warehousing techniques which enable the exploratory analysis of large-scale spatio-temporal data.

In the first work, we studied the integrated analysis of different simulation data sets and exploratory analysis of multiple accumulated simulation data. We developed a prototype system architecture, with data storage based on a *multi-dimensional data cube* for integrated analysis of multiple data sources, as well as an analysis interface to enable basic operators such as drill-down and roll-up for interactive analysis. We demonstrated the usability of the proposed prototype system by a case example using disaster simulation data.

In the second and third work, we studied advanced operators for exploratory analysis of spatio-temporal data by data summarization techniques. We summarized the target data based on the notion of histograms in the database research area. We defined a general

hierarchical histogram, called spatial V-optimal histogram, that summarizes the data distribution of a specific spatial area during a time interval, as well as a difference histogram based on the quadtree structure, which detects spatial and temporal variations of the target data.

However, building histograms is often time consuming, especially for the defined spatial V-optimal histogram. We propose both exact and heuristic algorithms for efficient construction of hierarchical histograms. The proposed methods are implemented on the state-of-the-art array DBMS, SciDB, which supports efficient scientific computing and analysis of spatio-temporal array data. In particular, for the difference histogram, we proposed a general framework based on quadtree for constructing histograms in a bottom-up style on SciDB.

Last but not the least, we conducted extensive experiments on massive spatio-temporal data to verify the performance of the proposed methods, and also show the usability of the proposed prototype system and histogram construction approaches by case studies using disaster simulation data.

5.2 Future Work

In the following, we list some remained topics of this thesis.

- **Cooperation with other systems:** In this thesis, we consider managing spatio-temporal data on a relational DBMS for integrated analysis of relational data, as well as an array DBMS, which enables efficient scientific computing and analysis of spatio-temporal array data. Some analyses could require cooperation with other systems. An example is cooperation with visualization processing. For some queries, advanced visualization is required for presenting the analysis process or analysis result. Further, if advanced statistical processing were required for a part of the target data, coordination with a statistical processing such as R would be necessary.

- **Other types of difference operators:** The difference operator proposed in Chapter 4 assumes that users specify two time segments T_1 and T_2. Such an approach, however, is applicable only when the time segments to be specified by users are known in advance. For more useful and evolutionary approaches, it could be extended as follows: Users can only specify the regions subject to analyses and the time segment width τ as an aggregate unit. The data sets can be aggregated in each time segment width τ and a sequence of aggregate results $T_1, T_2, \ldots, T_m$ can be constructed. Then, any differences between T_{i+1} and T_i $(1 \leq i \leq m-1)$ are determined and the top k pairs are selected in order of difference size from the largest one. In other words, selecting k pairs in order of the value of differences that users need to notice could effectively help users save their analyzing burden.

- **Semantic extension of the difference operator:** Another future issue required to be addressed could be a semantic extension of the difference operator. Although in this thesis we have simply noticed nothing but the difference sizes, we hope to develop the difference operator on information such as whether differences have an increasing tendency or a decreasing one and the speeds at which they increase or decrease. The way to visualize such differences is another important matter of consideration. Therefore, we intend to review how to visualize them. We also plan to develop an installation technology making the most of the functions of the array-oriented DBMS.

Appendix A

Appendix for Chapter 2

A.1 Schema of the Simulation Data Warehouse

Figure A.1 presents the schema description of the simulation data warehouse developed in the present prototype.

`RecordID` is a major key ID uniquely assigned. `EvacuationRecordTable` corresponds to the fact table. `TimeKey`, `PlaceKey`, `DepthKey` present foreign keys corresponding to the time, area, and flood depth dimensions, respectively. `Number` is the number of evacuees in each cell. That is, this table provides the number of evacuees from the viewpoint of time, area, and flood depth.

`TimeTable` is the dimension table. `TimeKey` is a major key ID. `hour`, `min30`, `min10`, `min5`, `min`, and `sec10` contain the sequence number at their respective level of the concept hierarchy.

`PlaceTable` and `DepthTable` are both dimension tables. Their details are omitted here.

```
CREATE TABLE TimeTable(
  TimeKey int NOT NULL PRIMARY KEY,
  hour int,
  min30 int,
  min10 int,
  min5 int,
  min int,
  sec10 int
)

CREATE TABLE PlaceTable(
  PlaceKey int NOT NULL PRIMARY KEY,
  AreaID1 int,
  AreaID2 int,
  AreaID4 int,
  AreaID8 int,
  AreaID16 int,
  AreaID32 int,
  AreaID64 int,
  AreaID128 int,
  AreaID256 int,
  AreaID512 int,
  AreaID1024 int,
  AreaID2048 int,
  AreaID4096 int,
)

CREATE TABLE DepthTable(
  DepthKey int NOT NULL PRIMARY KEY,
  cm100 int,
  cm50 int,
  cm25 int
)

CREATE TABLE EvacuationRecordTable(
  RecordID int NOT NULL PRIMARY KEY,
  TimeKey int FOREIGN KEY REFERENCES TimeTable(TimeKey),
  PlaceKey int FOREIGN KEY
    REFERENCES PlaceTable(PlaceKey),
  DepthKey int FOREIGN KEY
    REFERENCES DepthTable(DepthKey),
  Number int
)
```

FIGURE A.1: Schema of Simulation Data Warehouse

Appendix B

Appendix for Chapter 3

B.1 Proof of Lemma 3.4

It is proved that any rectangular partition can be converted into a hierarchical one by splitting each rectangle to at most four disjoint rectangles [81]. Consider a given histogram which has B buckets with error δ, it can be converted into a hierarchical one with at most $4B$ buckets. Moreover, since the error metric is superadditive, the metric value of this resulting hierarchical histogram is at most δ.

B.2 Proof of Lemma 3.6

Consider the optimal histogram of array $\mathcal{A}$ with B buckets is known as H^* with error δ, and the optimal k-hierarchical histogram of $\mathcal{A}$ is represented as H. For any arbitrary bucket b^* in H^*, there are at most $(2k+1)^2$ buckets generated in H with the same error reduction as b^* in H^*. The reason is basically based on the fact of possible partitions of the k-hierarchical histogram are constrained by the given k integers, i.e., $L_1, L_2, \cdots, L_k$. In detail, consider the side interval of b^* along one dimension is $[a, b]$ with length larger than L_1, where n is a multiply of L_1, as shown in Figure B.1. For the endpoint a, there are at most k partitions that are necessary if we use the approximate algorithm based on the binary partitioning, since each partition is conducted only on the positions specified by

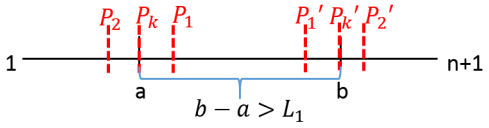


FIGURE B.1: Number of Partitions for $[a, b]$

L_i as mentioned. As shown in Figure B.1, there are at most $P_1, P_2, ..., P_k$ partitions for a, and the same holds for endpoint b. Therefore, there are totally $2k + 1$ intervals generated in H for interval $[a, b]$, so that there are at most $(2k + 1)^2$ buckets generated in H for each bucket in H^*. Moreover, since the error metric is monotonic and superadditive, the error of H is not bigger than $E(H^*)$. Based on the above mentioned observation, there are at most $(2k + 1)^2 B$ buckets in H, with an error of at most δ.

B.3 Proof of Theorem 3.7

As we know, ϵ determines the time complexity of the approximate algorithm, as well as the length of L_k, which determines the possible permissible side lengths by $k = \log n - \log L_k$. Since $L_k = n^\epsilon$, then $k = (1 - \epsilon) \log n$. Based on Lemma 3.6, we can know that, for an optimal hierarchical histogram of array A that has at most B_0 buckets with error δ_0, in $O(N^{1+1.5\epsilon} B^2)$ time, we can find a k-hierarchical histogram that has at most $O((1 - \epsilon)^2 (\log n)^2 B_0)$ buckets with error δ_0. Moreover, based on the above results and Theorem3.5, we can get the approximation bound for arbitrary histograms as $O(4(1 - \epsilon)^2 (\log n)^2 B_0)$, which is represented as $O((1 - \epsilon)^2 (\log n)^2 B_0)$ as well.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Prof. Yoshiharu Ishikawa for the continuous support of my Ph.D. studies. His valuable academic guidance and constant encouragement helped me all the way to the completion of this thesis. Meanwhile, his patience, honesty, diligence, enthusiasm and immense knowledge not only impressed me greatly, but also enlightened my future career.

I would also like to thank the rest of my thesis committee: Prof. Katsuhiko Toyama, Prof. Takami Yasuda and Prof. Yousuke Watanabe for their careful reading, insightful comments, and fruitful discussions.

My sincere thanks also go to Prof. Chuan Xiao and Dr. Kento Sugiura, co-authors of papers I published during my Ph.D. studies, for their constructive advice and thoughtful discussions.

I would like to take this opportunity to express my gratitude to Prof. Lei Chen, Prof. Jianliang Xu, and Prof. Kevin Zheng, for their unceasing encouragement and various forms of support.

I am also very grateful to the other current and past members of our laboratory at Nagoya University. Their kindness and friendship brought my unforgettable experience in this laboratory and enriched my life.

Last but not the least, I would like to thank my parents, for raising me and supporting me spiritually throughout my life. I would like to thank my friends who make my life merry and full of happiness as well.

Bibliography

[1] Ahmed Eldawy and Mohamed F. Mokbel. The era of big spatial data: A survey. *Found. Trends databases*, 6(3-4):163–273, December 2016.

[2] Xuan Song, Quanshi Zhang, Yoshihide Sekimoto, Ryosuke Shibasaki, Nicholas Jing Yuan, and Xing Xie. A simulator of human emergency mobility following disasters: Knowledge transfer from big disaster data. In *AAAI*, pages 730–736, 2015.

[3] Vagelis Hristidis, Shu Ching Chen, Tao Li, Steven Luis, and Yi Deng. Survey of data management and analysis in disaster situations. *J. Syst. Softw.*, 83(10):1701–1714, October 2010.

[4] Dimitris Papadias, Panos Kalnis, Jun Zhang, and Yufei Tao. Efficient OLAP operations in spatial data warehouses. In *Proc. SSTD*, pages 443–459, 2001.

[5] Dimitris Papadias, Yufei Tao, Panos Kalnis, and Jun Zhang. Indexing spatio-temporal data warehouses. In *Proc. ICDE*, pages 166–175, 2002.

[6] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proc. EDBT*, pages 168–182, 1998.

[7] Sunita Sarawagi. Explaining differences in multidimensional aggregates. In *Proc. VLDB*, volume 99, pages 7–10, 1999.

[8] Sunita Sarawagi and Gayatri Sathe. I^3: Intelligent, interactive investigation of OLAP data cubes. In *Proc. ACM SIGMOD*, 2000.

[9] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. Smart drill-down: A new data exploration operator. *Proc. VLDB Endow.*, 8(12):1928–1931, August 2015.

[10] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. Overview of data exploration techniques. In *Proc. ACM SIGMOD*, pages 277–281. ACM, 2015.

[11] E. F. Codd, S. B. Codd, and C. T. Smalley. Providing OLAP to user-analysis: An IT mandate. E.F. Codd and Associates, 1993. `http://www.minet.uni-jena.de/dbis/lehre/ss2005/sem_dwh/lit/Cod93.pdf`.

[12] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 3rd edition, 2011.

[13] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, 3rd edition, 2002.

[14] Alejandro Vaisman and Esteban Zimányi. *Data Warehouse Systems: Design and Implementation*. Springer, 2014.

[15] Sunita Sarawagi and Michael Stonebraker. Efficient organization of large multidimensional arrays. In *Proc. ICDE*, pages 328–336, Washington, DC, USA, 1994. IEEE Computer Society.

[16] Jiawei Han, Nebojsa Stefanovic, and Krzysztof Koperski. Selective materialization: An efficient method for spatial data cube construction. In *Research and Development in Knowledge Discovery and Data Mining*, pages 144–158, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[17] Nebojsa Stefanovic, Jiawei Han, and Krzysztof Koperski. Object-based selective materialization for efficient implementation of spatial data cubes. *IEEE Trans. on Knowl. and Data Eng.*, 12(6):938–958, November 2000.

[18] Leticia Gómez, Bart Kuijpers, and Alejandro Vaisman. A data model and query language for spatio-temporal decision support. *GeoInformatica*, 15(3):455–496, 2011.

[19] Silvia I. Gómez, Leticia A. Gómez, and Alejandro A. Vaisman. A generic data model and query language for spatiotemporal OLAP cube analysis. In *Proc. EDBT*, 2012.

[20] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley Publishing, 3rd edition, 2013.

[21] Swarup Acharya, Viswanath Poosala, and Sridhar Ramaswamy. Selectivity estimation in spatial databases. In *Proc. ACM SIGMOD*, pages 13–24, 1999.

[22] Yannis Ioannidis. The history of histograms (abridged). In *Proc. VLDB*, pages 19–30, 2003.

[23] H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. Optimal histograms with quality guarantees. In *Proc. VLDB*, pages 275–286, 1998.

[24] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[25] Satomi Hayashi and Shunichi Koshimura. The 2011 Tohoku tsunami flow velocity estimation by the aerial video analysis and numerical modeling. *Journal of Disaster Research*, 8(4):561–572, 2013.

[26] Noriaki Hirokawa and Toshihiro Osaragi. Earthquake disaster simulation system: Integration of models for building collapse, road blockage, and fire spread. *Journal of Disaster Research*, 11(2):175–187, 2016.

[27] Toshihiro Osaragi. Modeling a spatiotemporal distribution of stranded people returning home on foot in the aftermath of a large-scale earthquake. *Natural Hazards*, 68(3):1385–1398, Sep 2013.

[28] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatarao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[29] MultiDimensional eXpressions. http://en.wikipedia.org/wiki/MultiDimensional_eXpressions.

[30] Microsoft SQL Server. http://www.microsoft.com/en-us/sqlserver.

[31] GeoServer. http://geoserver.org/.

[32] OpenLayers 3. http://openlayers.org/.

[33] Multidimensional modeling (Adventure Works tutorial). https://msdn.microsoft.com/en-us/library/ms170208(v=sql.120).aspx.

[34] Andrew Eisenberg and Jim Melton. SQL standardization: The next steps. *ACM SIGMOD Record*, 29(1):63–67, March 2000.

[35] Andrew Eisenberg, Krishna Kulkarni, Jim Melton, Jan-Eike Michels, and Fred Zemke. SQL:2003 has been published. *ACM SIGMOD Record*, 33(1):119–126, March 2004.

[36] Jim Melton. *Advanced SQL:1999 – Understanding Object-Relational and Otehr Advanced Features*. Morgan Kaufmann, 2003.

[37] Hanan Samet. Object-based and image-based object representations. *ACM Computing Surveys*, 36(2):159–217, June 2004.

[38] Leticia Gómez, Bart Kuijpers, and Bart Moelans. A survey of spatio-temporal data warehousing. *International Journal of Data Warehousing and Mining*, 5(3):28–55, 2009.

[39] Luca Leonardi, Gerasimos Marketos, Elias Frentzos, Nikos Giatrakos, Salvatore Orlando, Nikos Pelekis, Alessandra Raffaetà, Alessandro Roncato, Claudio Silvestri, and Yannis Theodoridis. T-Warehouse: Visual OLAP analysis on trajectory data. In *Proc. ICDE*, pages 1141–1144, 2010.

[40] Sunita Sawaragi. User-adaptive exploration of multidimensional data. In *Proc. VLDB*, pages 307–316, 2000.

[41] Michael Stonebraker, Paul Brown, Alex Poliakov, and Suchi Raman. The architecture of SciDB. In *Proc. SSDBM*, volume 6809 of *LNCS*, pages 1–16, 2011.

[42] Michael Stonebraker, Paul Brown, Donghui Zhang, and Jacek Becla. SciDB: A database management system for applications with complex analytics. *IEEE Computational Science & Engineering*, 15(3):54–62, 2013.

[43] Ahmed Eldawy, Mohamed F. Mokbel, Saif Al-Harthi, Abdulhadi Alzaidy, Kareem Tarek, and Sohaib Ghani. SHAHED: A MapReduce-based system for querying and visualizing spatio-temporal satellite data. In *Proc. ICDE*, pages 1585–1596, 2015.

[44] Google BigQuery. https://cloud.google.com/bigquery/?hl=en.

[45] Alexander Aiken, Jolly Chen, Michael Stonebraker, and Allison Woodruff. Tioga-2: A direct manipulation database visualization environment. In *Proc. ICDE*, pages 208–217, 1996.

[46] Jing Zhao, Yoshiharu Ishikawa, Yukiko Wakita, and Kento Sugiura. Difference operators in simulation data warehouses. *J. Disaster Res.*, 12(2):347–354, 2017.

[47] Kenneth Salem Arunprasad P. Marathe. Query processing techniques for arrays. *The VLDB Journal*, 11(1):68–91, 2002.

[48] Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. STHoles: A multidimensional workload-aware histogram. In *Proc. ACM SIGMOD*, pages 211–222, May 2001.

[49] Viswanath Poosala and Yannis E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *Proc. VLDB*, pages 486–495, 1997.

[50] Laks V. S. Lakshmanan, Raymond T. Ng, Christine Xing Wang, Xiaodong Zhou, and Theodore J. Johnson. The generalized MDL approach for summarization. In *Proc. VLDB*, pages 766–777. VLDB Endowment, 2002.

[51] Shaofeng Bu, Laks V. S. Lakshmanan, and Raymond T. Ng. MDL summarization with holes. In *Proc. VLDB*, pages 433–444. VLDB Endowment, 2005.

[52] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *Proc. ACM SIGMOD*, pages 448–459. ACM, 1998.

[53] Jeffrey Jestes, Ke Yi, and Feifei Li. Building wavelet histograms on large data in mapreduce. *Proc. VLDB Endow.*, 5(2):109–120, October 2011.

[54] S. Muthukrishnan, Viswanath Poosala, and Torsten Suel. On rectangular partitionings in two dimensions: Algorithms, complexity, and applications. In *ICDT*, pages 236–256, 1999.

[55] Humaira Ehsan, Mohamed A. Sharaf, and Panos K. Chrysanthis. MuVE: Efficient multi-objective view recommendation for visual data exploration. In *ICDE*, pages 731–742, 2016.

[56] Leilani Battle, Michael Stonebraker, and Remco Chang. Dynamic reduction of query result sets for interactive visualizaton. *2013 IEEE International Conference on Big Data*, pages 1–8, 2013.

[57] David Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[58] Florin Rusu and Yu Cheng. A survey on array storage, query languages, and systems. *CoRR*, abs/1302.0103, 2013.

[59] Peter Baumann, Andreas Dehmel, Paula Furtado, Roland Ritsch, and Norbert Widmann. The multidimensional database system RasDaMan. In *Proc. ACM SIGMOD*, pages 575–577, 1998.

[60] Paradigm4: Creators of SciDB a computational DBMS. `http://www.paradigm4.com/`.

[61] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *SIGKDD*, KDD '11, pages 316–324, 2011.

[62] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: Driving directions based on taxi trajectories. In *SIGSPATIAL*, GIS '10, pages 99–108, 2010.

[63] Filippo Furfaro, Giuseppe M. Mazzeo, Domenico Saccà, and Cristina Sirangelo. Compressed hierarchical binary histograms for summarizing multi-dimensional data. *Knowl. Inf. Syst.*, 15(3):335–380, 2008.

[64] Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proc. ACM SIGMOD*, SIGMOD '96, pages 294–305. ACM, 1996.

[65] Ashraf Aboulnaga and Surajit Chaudhuri. Self-tuning histograms: Building histograms without looking at data. In *Proc. ACM SIGMOD*, SIGMOD '99, pages 181–192, New York, NY, USA, 1999. ACM.

[66] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras, and Carlotta Domeniconi. Approximating multi-dimensional aggregate range queries over real attributes. In *Proc. ACM SIGMOD*, SIGMOD '00, pages 463–474, New York, NY, USA, 2000. ACM.

[67] Ning An, Zhen-Yu Yang, and Anand Sivasubramaniam. Selectivity estimation for spatial joins. In *Proceedings of the 17th International Conference on Data Engineering*, pages 368–375, Washington, DC, USA, 2001. IEEE Computer Society.

[68] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. *ACM Trans. Database Syst.*, 27(3):261–298, September 2002.

[69] Dimitrios Gunopulos, George Kollios, J. Tsotras, and Carlotta Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. *The VLDB Journal*, 14(2):137–154, April 2005.

[70] Hicham G. Elmongui, Mohamed F. Mokbel, and Walid G. Aref. Spatio-temporal histograms. In *Proc. 9th Int'l Conf. on Advances in Spatial and Temporal Databases Spatial and Temporal Databases*, SSTD'05, pages 19–36, Berlin, Heidelberg, 2005. Springer-Verlag.

[71] Hyun Kyoo Park, Jin Hyun Son, and Myoung-Ho Kim. Dynamic histograms for future spatiotemporal range predicates. *Inf. Sci.*, 172(1-2):195–214, 2005.

[72] Ahmed Eldawy, Louai Alarabi, and Mohamed F. Mokbel. Spatial partitioning techniques in SpatialHadoop. *Proc. VLDB Endow.*, 8(12):1602–1605, August 2015.

[73] Yi Wang, Arnab Nandi, and Gagan Agrawal. SAGA: Array storage as a DB with support for structural aggregations. In *Proc. SSDBM*, SSDBM '14, pages 9:1–9:12, New York, NY, USA, 2014. ACM.

[74] Aditya Parameswaran, Neoklis Polyzotis, and Hector Garcia-Molina. SeeDB: Visualizing database queries efficiently. *PVLDB*, 7(4):325–328, 2013.

[75] Manasi Vartak, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. SeeDB: Automatically generating query visualizations. *PVLDB*, 7(13):1581–1584, August 2014.

[76] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. SeeDB: Efficient data-driven visualization recommendations to support visual analytics. *PVLDB*, 8(13):2182–2193, September 2015.

[77] Yoshiharu Ishikawa. Research trend and future prospects for large-scale data analytics. *IEICE Trans. on Information and Systems (Japanese Edition)*, J97-D(4):718–728, 2014. (in Japanese).

[78] Gennady Andrienko, Natalia Andrienko, and Stefan Wrobel. Visual analytics tools for analysis of movement data. *SIGKDD Explor. Newsl.*, 9(2):38–46, December 2007.

[79] Jing Zhao, Kento Sugiura, Yuanyuan Wang, and Yoshiharu Ishikawa. Simulation data warehouse for integration and analysis of disaster information. *Journal of Disaster Research*, 11(2):255–264, 2016.

[80] Rphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[81] Fabrizio d'Amore and Paolo Giulio Franciosa. On the optimal binary plane partition for sets of isothetic rectangles. *Inf. Process. Lett.*, 44(5):255–259, 1992.

List of Publications

Journal Papers

- Jing Zhao, Yoshiharu Ishikawa, Lei Chen, Chuan Xiao, and Kento Sugiura. "Building Hierarchical Spatial Histograms for Exploratory Analysis in Array DBMS". IEICE Transactions on Information and Systems, Vol. E102-D, No.4, Apr. 2019. (accepted for publication)

- Jing Zhao, Kento Sugiura, Yuanyuan Wang, Yoshiharu Ishikawa, "Simulation Data Warehouse for Integration and Analysis of Disaster Information", *Journal of Disaster Research*, Vol. 11, No. 2, pp. 255–264, March 2016.

- Jing Zhao, Yoshiharu Ishikawa, Yukiko Wakita, Kento Sugiura, "Difference Operators in Simulation Data Warehouses", *Journal of Disaster Research*, Vol. 12, No. 2, pp. 347–354, March 2017.

- Yusuke Kawai, Jing Zhao, Kento Sugiura, Yoshiharu Ishikawa, Yukiko Wakita, "An Analysis Technique of Evacuation Simulation Using an Array DBMS", *Journal of Disaster Research*, Vol. 13, No. 2, pp. 338–346, March 2018.

International Conference/Workshop Papers

- Renhe Jiang, Jing Zhao, Tingting Dong, Yoshiharu Ishikawa, Chuan Xiao, Yuya Sasaki, "A Density-based Approach for Mining Movement Patterns from Semantic Trajectories", *The IEEE Region 10 Conference (TENCON 2015)*, Macau, China, November 2015.

- Tingting Dong, Yoshiharu Ishikawa, Chuan Xiao, Jing Zhao, "*k*-Expected Nearest Neighbor Search over Gaussian Objects", *The 4th International Conference on Network and Computing Technology (ICNCT 2015)*, Rome, Italy, December 2015.

- Jing Zhao, Yoshiharu Ishikawa, Chuan Xiao, Kento Sugiura, "Histogram Construction for Difference Analysis of Spatio-Temporal Data on Array DBMS", *2018 Australasian Database Conference (ADC 2018)*, pp. 41–52, Lecture Notes in Computer Science, Vol. 10837, Gold Coast, Australia, May 2018.

- Jing Zhao, Yoshiharu Ishikawa, Chuan Xiao, Kento Sugiura, "Simulation Data Summarization Based on Spatial Histograms", *21st International Conference on Network and Computing Technology (ICNCT 2019)*, Paris, France, January 2019. (accepted for publication)

国内学会発表（査読なし）

- 趙セイ，董テイテイ, 石川佳治, ,「参加型センシングにおけるプライバシー保護手法」, 情報処理学会第76回全国大会, 5N-7, 2014年3月.

- 趙セイ, 杉浦健人, 姜仁河, 佐々木勇和, 石川佳治, 「参加型センシングのための空間データベース問合せ処理」, 第13回情報科学技術フォーラム (FIT 2014), 2014年9月.

- 趙菁, 姜仁河, 董テイテイ, 佐々木勇和, 石川佳治, 「参加型センシングのためのタスク割当て手法」, 第7回データ工学と情報マネジメントに関するフォーラム（DEIM 2015）, C6-5, 2015年3月.

- 姜仁河, 趙菁, 董テイテイ, 佐々木勇和, 石川佳治, 「密度に基づく意味的な軌跡パターンの発見」, 第7回データ工学と情報マネジメントに関するフォーラム（DEIM 2015）, E8-3, 2015年3月.

- 趙セイ, 石川佳治, 肖川, 董テイテイ, 佐々木勇和, 「空間クラウドソーシングのための多様性を考慮したタスク割り当て手法」, 情報処理学会研究報告, 2015-DBS-161(8), 2015年8月.

- 趙菁，石川佳治，杉浦健人，王元元，佐々木勇和, 瀧本祥章,「シミュレーションデータウエアハウスにおける災害情報の統合分析」, 第8回データ工学と情報マネジメントに関するフォーラム (DEIM 2016), A2-3, 2016年2月.
- 趙セイ, 石川佳治, 杉浦健人, 脇田佑希子,「時空間データウェアハウスにおける差分演算について」, 第15回情報科学技術フォーラム (FIT 2016), 2D-4, 2016年9月.
- 趙セイ, 石川佳治, 杉浦健人, 脇田佑希子,「時空間データ分析のための差分ヒストグラム構築手法」, 第9回データ工学と情報マネジメントに関するフォーラム (DEIM 2017), G1-2, 2017年3月.
- 河井悠佑, 杉浦健人, 趙セイ, 石川佳治,「配列指向DBMSを用いた避難シミュレーションデータの格納と分析」, 情報処理学会第79回全国大会, 1L-03, 2017年3月.
- 趙セイ, 石川佳治, 河井悠佑, 杉浦健人,「配列DBMSにおける時空間データの差分分析について」, 第10回データ工学と情報マネジメントに関するフォーラム (DEIM 2018), C6-3, 2018年3月.
- 李セイ, 石川佳治, 趙セイ, 杉浦健人,「逆最近傍問合せに基づくデマンドヒートマップの連続的な更新手法」, 第16回情報科学技術フォーラム (FIT 2017), D-012, 2017年9月.
- 安田健人，河井悠佑, 趙セイ，杉浦健人，石川佳治,「配列DBMSにおける空間スキャン統計量の計算手法」, 情報処理学会全国大会, 4L-5, 2018年3月.
- 野田昌太郎，河井悠佑, 趙セイ，杉浦健人，石川佳治,「大規模データ分析のための可視化手法に関する検討」, 情報処理学会全国大会, 6L-7, 2018年3月.