The Summarization of Hierarchical Data with Exceptions

by

Shaofeng Bu

M.Eng., Xi'an Jiaotong University, P.R.China, 2001

B.Eng., Xi'an Jiaotong University, P.R.China, 1998

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard

The University of British Columbia

October 2004

Library Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Shaofeng Bu
Name of Author *(please print)*

05/10/2004
Date (dd/mm/yyyy)

Title of Thesis: The Summarization of Hierarchical Data with Exceptions

Degree: Master of Science Year: 2004

Department of Computer Science
The University of British Columbia
Vancouver, BC Canada

Abstract

In many applications of OLAP or data warehouse, users need to query data of interest, such as a set of data that satisfies specific properties. A normal answer to such query just enumerates all the interesting cells. This is the most accurate but not the most informative method. Summarizations need to be done in order to return more concise descriptions of these interesting cells to the users. MDL approach has been applied on the hierarchical data to get concise descriptions. However in many cases the descriptions are not concise enough to the users. Another method, GMDL, can generate much shorter descriptions, but the GMDL descriptions are not truly pure. The motivation of our research is to overcome the disadvantages in the above methods.

In this thesis, we bring up a methodology that focuses on generating the summarization with exceptions of the hierarchical data. We extend the MDL approach to include some exceptions in the description. The exceptios are some uninteresting cells. The result shows that the description with exceptions is pure, which means that the description only covers "interesting cells". We call this new approach MDLE, i.e. MDL with exceptions. Our new approach aims to find the shortest description with exceptions to cover all "interesting cells". Firstly, we study two simple cases that can be solved in polynomial time and we give the algorithms. Secondly, we prove that MDL with exceptions is an NP-Hard problem in general cases and we propose three heuristics. Finally, we show some experiments that we have done to compare MDLE with MDL and GMDL. The experiment results show that MDLE generates more concise descriptions than MDL and meantime MDLE gets shorter descriptions than GMDL when the white-ratio is low or there are some red cells.

Contents

List of Figures

Acknowledgements

I would like to express sincere gratitude to my supervisor, Dr. Raymond T. Ng, for his helpful suggestion and constant encouragement throughout the course of my research work. Without his guidance and inspiration, this research work would never have been possible.

I highly appreciate the help from Dr. Laks V.S. Lakshmanan. Without the fruitful discussions with him and his excellent advice, this research work would not be done in time. I would also like to thank Dr. Will Evans, who was a member of my supervisory committee, for his kindness and valuable instructions.

I would like to thank Dr. George K. Tsiknis for his reviewing on my thesis. His comments and suggestions are very helpful and valuable.

I would like to thank all the members in database system lab, especially Dr. Ramesh Ganesh and Xiaodong Zhou who gave me lots of good advices in my research work.

I am very grateful to Dr. Ming Huo who gave me many good suggestions for my thesis writing.

Finally, I would like to express my deeply thankfulness to my parents. Without their endless support, understanding, patience and suggestion, I could not overcome the difficulties in my life and study.

SHAOFENG BU

The University of British Columbia
October 2004

Chapter 1

Introduction

With the increase of data accumulated in the application databases and the demand to analyze transaction data for decision support, data warehousing and on-line analytical processing(OLAP) have been studied and widely used in the last decade. In contrast to the on-line transaction processing(OLTP) applications, data warehousing and OLAP systems deal with historical, summarized and consolidated data, which are more meaningful and useful than detailed, individual records[18][3]. Data analysis applications generally aggregate data across several different dimensions in order to find useful or meaningful, unusual or hidden trends or patterns[6], and the data in such applications is stored under the multidimensional model. For example, in a hospital data warehouse, the data for patients might include birthday, sex, occupation and birthplace as interesting dimensions. The data warehousing and OLAP applications present the interesting data to users by a multidimensional model, which is a data cube. Dimensions can often be represented in hierarchical structure[3]. Birthplace can be a city-province-country hierarchy. Birthday is a day-month-year hierarchy.

In hierarchy each ancestor or upper level node is the summarization of its

descendant nodes or lower level nodes. Given a multidimensional hierarchical data cube, how to represent an interesting data set in this cube by a more concise and meaningful method is a good research problem and helpful to find unusual or hidden patterns from the data set. For example in a 2-dimensional data cube which is in the form of a 10×10 matrix, there are 10 leaf nodes in each dimension and the data cube contains 100 data cells. A user's query needs to select all the data cells that satisfy the user defined property p. Suppose the query returns 80 single data cells to the user. The problem is how to express all these interesting data cells. Shall we enumerate them or do some summarization? The summarization of the data has two advantages. First, the length of the expression is much shorter. Second, the expression contains the hierarchy information and is more meaningful to the user.

1.1 Motivating Example and Problems Statement

Data summarization has been studied in [1][11][19][14]. The goal of the summarization is to use a concise description to express a set of data. In a multidimensional data cube, a data cell is defined as a tuple that contains a leaf node in each dimension. A data region is defined as a tuple that contains some internal non-leaf nodes in some dimensions. Both data cells and data regions are axis-parallel rectangular regions. When users apply any query on the data cube, the system returns all the data cells that satisfy the properties defined by users. All these data cells are interesting cells to the user and are marked as blue cells, while the rest of the cells are non-blue cells.

A summary is a non-redundant covering of interesting cells including maximal axis-parallel rectangular regions[1]. We also call such a covering a description for all interesting cells. In the hierarchical data cube, a description contains data cells

and data regions. The length of a description is defined as the cardinality of this description, which is the number of the included data cells and the included data regions.

Minimum Description Length(MDL) was firstly proposed in [8]. MDL is to find the most concise description for a set of data cells. An algorithm for MDL approach in the hierarchical data cube, i.e. MDL-Tree algorithm, has been proposed in [11]. MDL-Tree algorithm is to find the unique description with the minimum length for a set of interesting data cells in the pure hierarchical data cube. The authors also generalized MDL approach to GMDL to decrease the length of the description. In GMDL approach, data cells are grouped into three categories, i.e. "interesting cells", "don't care cells" and "undesirable cells". Contrast to the MDL description, which only includes "interesting cells", a GMDL description can include some "don't care cells" to cover all the "interesting cells". The GMDL description has shorter length than the MDL description.

Before we bring out the statements of the problems with which this thesis is concerned, let us study an example.

Example 1.1.1. In order to see how we can summarize interesting data cells in a multidimensional hierarchical data cube, we begin with an example from sales department. The hierarchical structure is the same as in [11]. Clothes and location are the two dimensions. The measure is still the sale dollar amount. Each dimension has the hierarchical structure showed in Figure 1.1.

For hierarchical data, a data cell corresponds to a tuple of leaf level values in each dimension[11], such as *(Vancouver, skirts)* and *(Chicago,jackets)*. A data region is a tuple that contains internal non-leaf nodes in some dimensions. *(Vancouver men's)* and *(northwest, jackets)* are data regions. All the data cells that satisfy the

user defined property are interesting cells and we call them 'blue cells' and mark them with '◯' in the figure. All unmarked cells are uninteresting cells and are called non-blue cells.

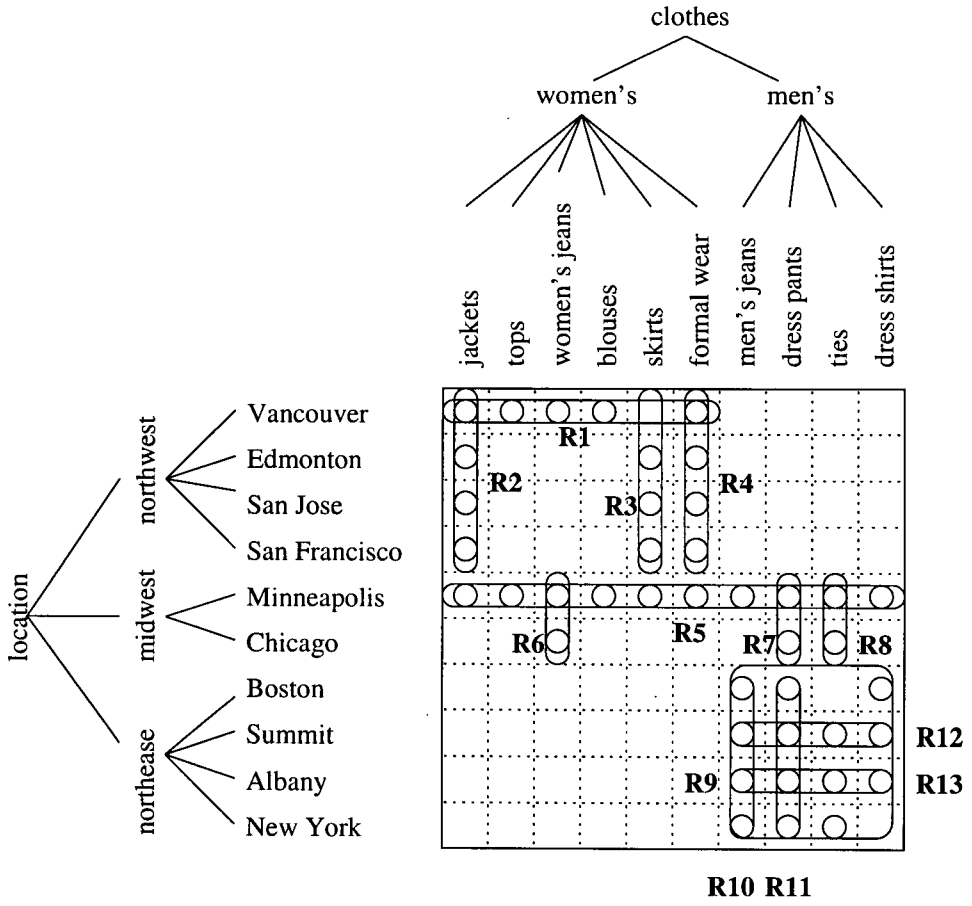


Figure 1.1: Motivating Example

Suppose the company manager still wants to summarize all "hot" items this year. The manager still defines a cell as blue if the sales amount in this year is at least 2 times that of last year. The data of 2004 is different from the data of 2002 in [11]. We find *(Vancouver,skirts),(Boston,ties)* and *(New York,dress shirts)* are not blue any more, which means these three items are not 'hot' items this year and

do not satisfy the property defined by the manager. So we cannot get R_1, R_3, R_9 in Figure 1.1 for 2004. MDL approach merges children regions to their parent region if all these children regions contain only blue cells. For example, $(Minneapolis, ties)$ and $(Chicago, ties)$ are merged into region R_8, i.e. $(midwest, ties)$. For the data of this year, the MDL approach generates a covering containing 10 regions, i.e. $R_2, R_4, R_5, R_6, R_7, R_8, R_{10}, R_{11}, R_{12}, R_{13}$, and 8 single blue cells that are not covered by these 10 regions. This covering contains 10 regions and 8 cells and we say the length of this covering is 18.

Now the question here is whether we can do better. Can we generate a shorter covering? We know the reason that R_1, R_3 and R_9 are not in the MDL covering is that *(Vancouver,skirts),(Boston,ties)* and *(New York,dress shirts)* are not blue this year. We define $\{R_1 - (Vancouver, skirts)\}$ as all the data cells in region R_1 except *(Vancouver,skirts)*, and we similarly define $\{R_3 - (Vancouver, skirts)\}$ and $\{R_9 - (Boston, ties) - (NewYork, dressshirts)\}$. Therefore we can generate a new covering containing 9 regions, i.e. $R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8, R_9$, with the exceptions of *(Vancouver,skirts),(Boston,ties)* and *(New York,dress shirts)*. Now the new covering includes 9 regions and 3 cells(as exceptions). The length of the new covering is 12. □

From this example we know that if we include some non-blue cells in the covering then we can generate a shorter covering for the blue cells. At the same time, this kind of covering is more informative and meaningful to users. For instance, in Figure 1.1, we use $\{R_1 - (Vancouver, skirts)\}$ to express the blue cells instead of enumerating all of them. Therefore users can know all the data cells in R_1 satisfy the query except $(Vancouver, skirts)$. This is much more intuitive and meaningful than only enumerating all blue cells.

In this thesis, we focus on the multidimensional data cube with hierarchy in each dimension. We call this kind of data cube the multidimensional hierarchical data cube. The hierarchy associated with data cube is a tree hierarchical structure, which is the common hierarchy in many real application systems.

A description with some exceptions is to cover all the blue cells by including some non-blue cells. We cover all blue cells by a union of regions and blue cells with the exception of some non-blue cells. This description is still a covering for the blue cells. The included non-blue cells are also called holes in the covering. Given a description with exceptions, the length of this description is defined as the cardinality of the description, i.e. the number of the contained regions and blue cells plus the number of included non-blue cells. MDL with exceptions(MDLE) is to find the shortest description with exceptions.

1.2 Contributions

This thesis studied the problem on how to do summarization with exceptions on pure hierarchical data, which is called MDL with exceptions for pure hierarchical data. MDL summarization has been studied and MDL-Tree algorithm have been proposed in [11]. This thesis extends MDL covering to include holes in order to shorten the length of the covering. The major contributions of this thesis are:

- We prove the MDL with exceptions is an NP-Hard problem. Even though MDL summarization can be solved in time linear in the size of the hierarchies[11] [19] we can see that when we just include data cells as exceptions in the description this problem becomes NP-Hard.

- Two tractable cases are studied and the algorithms are given. We show two

simple cases that are tractable and solvable. We study these two cases and analyze why they can be solved in polynomial time. We also propose algorithms for each case.

- We propose three heuristics. We show some examples to illustrate the reason why this problem is hard to solve. We have three heuristics. Experiments results are showed to compare our heuristics with MDL-Tree algorithm and GMDL approach[11]. These three heuristics generate much shorter descriptions than MDL-Tree algorithm. They also work better than GMDL when the white ratio is not too high and work much better than GMDL when there is a fraction of red cells.

1.3 Outlines of Thesis

The following parts of this thesis is organized as:

- Chapter 2 - Background and Related Work.
- Chapter 3 - Complexity Analysis. In this chapter we discuss the complexity of MDLE problem and we prove that this is an NP-Hard problem. We also study the 1-dimensional case which is tractable.
- Chapter 4 - Heuristics and Experiments. We study more about this NP-Hard problem and propose three heuristics: Greedy, Dynamic Programming, and Quadratic Programming. We also show the comparison amongst these three heuristics, MDL-Tree algorithm and GMDL-Tree algorithm by experimental results.

- Chapter 5 - Conclusions and Future Work. We give the summarization of this thesis and propose some ideas for further work.

Chapter 2

Background and Related Work

2.1 Rectangle Covering Problem

The MDL covering problem is similar to the rectangle covering problem which is to find the minimum number of axis-parallel rectangles to cover a 2-dimensional rectilinear polygon. Rectangle covering problem has been studied by many researchers and it is shown an NP-Hard problem in [16]. The rectangle covering for polygons without holes has been studied by [4]. In [10] the authors propose an $O(\sqrt{\log n})$ factor approximation algorithm for covering a rectilinear polygon with holes using axis-parallel rectangles.

Our problem is different from the rectangle covering problem. This thesis studies the hierarchical data. Therefore, the "rectangle regions" in the MDL with exceptions can only be a tuple that contains a node from the hierarchy in each dimension other than an arbitrary region. The other difference is that we deal with multidimensional not only 2-dimensional data.

2.2 MDL and GMDL for Hierarchical Data

In [11], the authors proposed a generalized MDL approach for summarization, which is called GMDL. In many cases, it is hard to define strict determinations for query properties. The user may define what kind of cells are definitely "interesting" and what kind of cells are definitely "undesirable". All other cells are called "don't care" cells. The interesting cells are named as "blue cells". The undesirable cells are called "red cells". The "don't care" cells are "white cells". GMDL tries to shorten the length of the covering by allowing some white cells in the covering. By adding white cells, some regions in MDL covering can be merged into big regions and then the total number of the regions decreases.

Firstly, the authors of [11] applied GMDL approach on the spatial data and proposed four GMDL algorithms to summarize spatial data with white cells. Secondly, this paper studied hierarchical data and brought out a MDL-Tree algorithm whose running time is linear to the sizes of the hierarchies. Then based on MDL-Tree algorithm, this paper proposed GMDL-Tree algorithm for hierarchical data. From the experiments we know that the GMDL approach generates more concise covering for blue cells.

In this thesis we focus on the MDL with exceptions on pure hierarchical data. Now let us discuss more about MDL-Tree algorithm and GMDL-Tree algorithm for hierarchical data as they were presented in [11].

2.2.1 MDL-Tree Algorithm

MDL-Tree algorithm finds the shortest MDL covering for a set of blue cells. MDL-Tree algorithm visits nodes in each hierarchy. Then regions in lower level are merged into regions in higher level if all children regions of a node contain only blue cells.

From Example 1.1.1 we already know that the covering generated by MDL-Tree algorithm is still not short enough. In this thesis we extend MDL-Tree algorithm to allow some holes in a covering. For example, if a node has 10 children cells and 9 of those children cells are blue. The MDL-Tree algorithm can not merge its 9 children to a bigger region because one of its children is not a blue cell. Therefore the covering for those 9 blue cells returned by MDL-Tree algorithm enumerates those blue cells and the length of this covering is 9. But MDL with exceptions(MDLE) can generate a shorter covering by including that non-blue child as a hole in the covering. The covering is in the form of this node excepting its non-blue child cell and the length of this covering is 2.

2.2.2 GMDL-Tree Algorithm

In [11], the authors proposed GMDL approach to generate more concise covering than MDL approach. GMDL approach allows the covering for blue cells to contain some white cells and then more regions can be merged into bigger regions. It is good to get shorter coverings for users, but the covering becomes not pure or not precise by including some white cells. MDLE is different from The GMDL approach. MDLE also allows non-blue cells in the covering, but the non-blue cells are in the exception part of the covering. So the user knows clearly about which cells are included in the covering and the covering is pure and precise.

On the other hand, the non-blue cells in MDLE covering can be white cells or red cells. Therefore, if a node has 10 children cells, which contains 9 blue cells and 1 red cell, then the covering generated by GMDL-Tree algorithm enumerates those 9 blue cells. MDL with exceptions(MDLE) generates a shorter covering by including that red cell as a hole in the covering. The covering is in the form of this

node except its red child cell. The length of this covering is 2.

2.3 Automatic subspace Clustering

The problem of automatic subspace clustering has been studied in [1]. The authors proposed an algorithm, CLIQUE, to find clusters embedded in subspaces of high dimensional data. CLIQUE also applied MDL-based pruning to decide which subspaces are interesting. The cluster descriptions in the form of DNF expressions returned by CLIQUE have been minimized for easy understanding. The work in [1] is related to our work. The difference is that we study the pure hierarchical data.

2.4 Concise Descriptions

In [13], the authors formalized the MDL approach to generate the concise descriptions for subsets of structured sets. They argued that the MDL problem for simple hierarchies(1-dimensional hierarchy) can be solved in polynomial time and they proposed an algorithm for this case. Then they proved that the MDL problem for multidimensional hierarchies is NP-Hard.

Our problem has two important different aspects from [13].

Firstly, the length of a description in [13] is defined as the number of nodes used in the description. For example, in a 2-dimensional hierarchical data cube, $'a'$ is a node in one dimension and $'1'$ is a node in the other dimension. The description to cover $(a, 1)$ in [13] is $S_1 = a \cdot 1$. The length of S_1 is 2 because S_1 contains two nodes. In the user's view, the important thing is how many data cells are included in the description. We need only to use $(a, 1)$ to express this data cell and the length is 1. In this thesis the length of a description is defined as the number of axis-parallel

rectangular regions, which are data cells or data regions in a data cube, contained in this description.

Secondly, product expressions are allowed in [13]. For example, in a 2-dimension hierarchical data cube, $\{a, b, c, d\}$ is from one dimension and d is the parent of $\{a, b, c\}$, while $\{1, 2, 3\}$ is from the other dimension and 3 is the parent of $\{1, 2\}$. In [13] $S = (a + b) \times 2$ is a valid expression to cover two cells $(a, 2)$ and $(b, 2)$. The length of S in [13] is 3 because S uses three nodes, i.e. a, b and 2. But in user's view, the expression including $(a + b)$ is not pure hierarchical data any more, because $(a + b)$ treats the data as spatial data again, which has been studied in [1]. In this thesis, we study pure hierarchical data and we do not allow product expressions in the descriptions.

Chapter 3

Complexity Analysis

In this chapter we will give the formal definitions of MDL with exceptions for pure hierarchical data. We study the 1-dimensional hierarchy firstly and give an algorithm to get the optimal description in this case. Then we prove MDL with exceptions is an NP-Hard problem in a 2-dimensional hierarchical data cube.

3.1 One-Dimensional Hierarchy: A Tractable Case

3.1.1 Basic Definitions

Definition 3.1.1. An $l-level$ hierarchy is in the form of $h = \{h_{11}, h_{21}, ..., h_{2n_2}, ..., h_{l1}, ..., h_{ln_l}\}$.

- h_{11} is the root;
- for any $1 \leq i \leq l$, n_i is the number of nodes in level i;
- for any $2 \leq i \leq l$, h_{ij} has one parent $h_{i-1,k}$ in level $i-1$. □

Let us use $Cld(h_{ij})$ to represent the children of h_{ij} and $Des(h_{ij})$ to present all the descendants of h_{ij}.

Definition 3.1.2. A node $h_{ij,1\leq i\leq l,1\leq j\leq n_i}$ is a leaf iff $Cld(h_{ij}) = \emptyset$ □

For any leaf node h_{ij} we have $Cld(h_{ij}) = Des(h_{ij}) = \emptyset$.

Without loss of generality, let us assume only the nodes in level l do not have children. This is also the very common hierarchical structure in many real applications. In fact, if a node in level 1 to $l-1$, for example h_{ij}, has no children, we can add a new node as its child and repeat this adding procedure till all nodes in level 1 to $l-1$ have at least one child. The new added nodes will not affect the hierarchical structure. Under this assumption all and only the nodes in level l are leaves.

Let us use $Leaves(h)$ to present the set of leaves of h, then

$$Leaves(h) = \{h_{ij}|1 \leq i \leq l, 1 \leq j \leq n_i, Cld(h_{ij}) = \emptyset\} = \{h_{lj}|1 \leq j \leq n_l\}$$

The data in a data warehouse or an OLAP system is generally modelled in a multidimensional structure [3] [6], commonly called a data cube. In this thesis we only study data cubes with a hierarchical structure in each dimension.

Definition 3.1.3. In a 1-dimensional hierarchical data cube with hierarchy h:

- Data Cell is a leaf node, such as h_{lk}, $for\ some\ 1 \leq k \leq n_l$;
- Data Region is a non-leaf node, such as h_{ij}, $for\ some\ 1 \leq i \leq l-1, 1 \leq j \leq n_i$.

□

Definition 3.1.4. Given a set of data regions and data cells, A, we have the following definitions:

1. The length of A is defined as the number of regions and cells in A, that is $|A| = |\{h_{ij}|h_{ij} \in A\}|$

2. $Cell(A)$ is the set of all data cells covered by A.

$$Cell(A) = \{h_{lk} | h_{lk} \in A \text{ or } \exists h_{ij} \in A, h_{lk} \in Des(h_{ij})\}$$

□

Now we define the operations between two sets of data regions and data cells. We use '+' to represent the union operation and '−' to represent the difference operation.

Definition 3.1.5. Given two sets of data regions and data cells, A, B, we have the following definitions:

1. $A = B \Longleftrightarrow Cell(A) = Cell(B)$

2. $Cell(A + B) = Cell(A) + Cell(B) = \{h_{lk} | h_{lk} \in Cell(A) \cup Cell(B)\}$

3. $Cell(A - B) = Cell(A) - Cell(B) = \{h_{lk} | h_{lk} \in Cell(A), h_{lk} \notin Cell(B)\}$

□

Blue cells in a data cube are all those cells that satisfy properties defined by users, or we call them interesting cells to users. All other data cells are called non-blue cells. When we give an interesting set D, we always call the data cells in D as blue cells and data cells not in D as non-blue cells. We use a property 'color' to represent whether a cell is blue. For example, $h_{lk}.color = blue$ means cell h_{lk} is a blue cell.

Now let us discuss more about a data region h_{ij}. We can cover the blue cells in h_{ij} by:

1. enumerating all the blue cells, which is in the form of

$$S_1 = \{h_{lk} | h_{lk} \in Cell(h_{ij}), h_{lk}.color = blue\}$$

and the length of S_1 is

$$S_1 = |\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color = blue\}|$$

or

2. using h_{ij} with exceptions of all non-blue cells in $Cell(h_{ij})$, which is in the form of

$$S_2 = h_{ij} - \{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color \neq blue\}$$

and the length is of S_2 is

$$|S_2| = 1 + |\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color \neq blue\}|$$

The difference between the length of S_1 and the length of S_2, i.e.$|S_1| - |S_2|$, is the number of regions and cells can be saved to cover the blue cells by S_2. We call this difference as the benefit of h_{ij}.

Definition 3.1.6. Given a data region $h_{ij}, 1 \leq i \leq n_{l-1}$, its benefit is defined as:

$$\begin{aligned} Benefit(h_{ij}) &= |\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color = blue\}| \\ &\quad -(|\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color \neq blue\}| + 1) \\ &= |\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color = blue\}| \\ &\quad -|\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color \neq blue\}| - 1 \end{aligned}$$

□

Example 3.1.1. The example in Figure 3.1 is a 1-dimensional hierarchy with three levels. The root is $h_{11} = Canada$ which has three children $BC, Alberta, Ontario$ in the second level. This hierarchy has ten leaves in the third level. The leaves are data

cells, such as *Vancouver, Victoria, Toronto....* The non-leaf nodes are data regions, such as *Canada, Alberta.*

Given a set $A = \{Victoria, Alberta, Toronto\}$, the length of A is 3, and we have

$$Cell(A) = \{Victoria, Edmonton, Calgary, Toronto\}$$

If $B = \{Vancouver, Edmonton, Waterloo\}$ then

$$Cell(A + B) = \{Victoria, Vancouver, Edmonton, Calgary, Toronto, Waterloo\}$$

$$Cell(A - B) = \{Victoria, Calgary, Toronto\}$$

Given a subset of leaves, D, with seven elements,

$$D = \{Vancouver, Richmond, Victoria, Edmonton, Toronto, Waterloo, London\}$$

We mark these seven cells as blue and the other three cells as non-blue. So BC has three children which are blue. In order to cover these three blue cells in BC, we can

- enumerate all blue cells as { *Vancouver, Richmond, Victoria*} and the length is 3, or
- use a description with exceptions as $\{BC\text{-}Burnaby\}$ and the length is 2.

So we only need 2 regions/cells in $\{BC\text{-}Burnaby\}$ to cover those 3 blue cells, and we can save 1 region/cell in the covering. Therefore $Benefit(BC) = 3 - (1 + 1) = 1$.

□

3.1.2 Problem Statements

Definition 3.1.7. A description with exceptions for an interesting data set D in a hierarchical data cube h is in the form of $S = U_1 - U_2$ where U_1 is a set of data

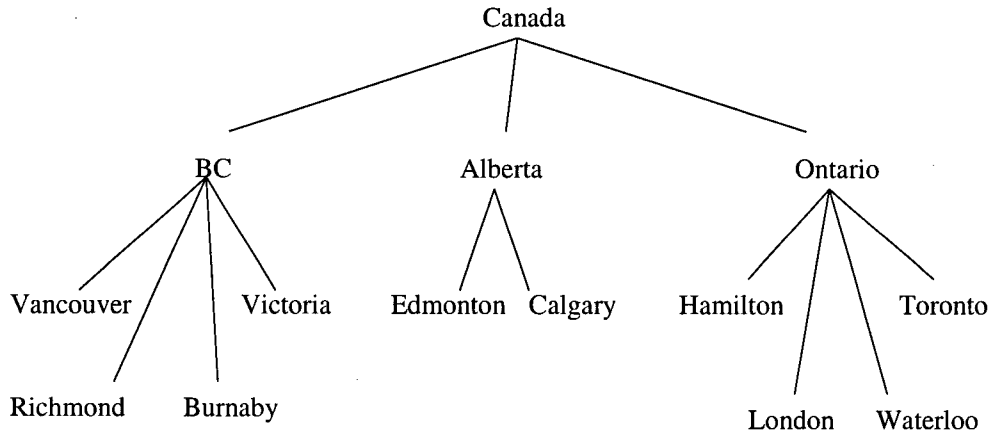


Figure 3.1: An Example for 1-Dimensional Hierarchy

regions and data cells, U_2 is a set of data cells, and S covers exactly the data cells in D, i.e. $Cell(S) = D$. □

From the definition of the description with exceptions in Definition 3.1.7, U_2 can only contain data cells. A description for D is a covering of all data cells in D. The data cells in U_2 are called holes in this covering of D. We only enumerate all the holes in a covering instead of doing any summarization on holes.

Definition 3.1.8. For a description $S = U_1 - U_2$ for D in h, its length is defined as $|S| = |U_1| + |U_2|$, and its benefit is defined as $Benefit(S) = |D| - |S|$. □

$|D|$ is the number of interesting cells. The length of S is the number of regions and cells in S to cover the blue cells in D. So the benefit of S is the number of regions and cells that are saved to cover the interesting cells D by the description S.

Let us use S_{Opt} to denote the description with the minimum length amongst all the descriptions with exceptions for D, and we say that S_{Opt} is an optimal description for D in h. For a given instance, $|D|$ is constant, so based on Definition 3.1.8

S_{Opt} has the maximum benefit too.

Definition 3.1.9 (MDL with Exceptions: MDLE). MDL with exceptions is to find an optimal description S_{opt}, which has the minimum length and the maximum benefit, for D in h. □

Example 3.1.2. Let us still use the example in Figure 3.1.

The interesting data set D is

$$D = \{Vancouver, Richmond, Victoria, Edmonton, Toronto, Waterloo, London\}$$

In order to cover D, we can use a description as

$$\begin{aligned} S_1 &= \{(BC - Burnaby) + Edmonton + (Ontario - Hamilton)\} \\ &= \{BC + Edmonton + Ontario\} - \{Burnaby + Hamilton\} \end{aligned}$$

The length of S_1 is $|S_1| = 3 + 2 = 5$, so $Benefit(S_1) = |D| - |S_1| = 7 - 5 = 2$.

Or we can use another description,

$$S_2 = Canada - \{Burnaby + Calgary + Hamilton\}$$

The length of S_2 is $|S_2| = 1 + 3 = 4$, so $Benefit(S_2) = |D| - |S_2| = 7 - 4 = 3$. S_2 is the description with the minimum length for D in h. □

3.1.3 Algorithm for MDL with Exceptions

Before we go to the algorithm for MDL with exceptions, let us see how to compute the benefit of a region based on its children's benefits.

Any data region h_{ij}, *for some* $1 \leq i \leq l-2$, is a node in level 1 to $l-2$. So h_{ij}'s children are in some level from 2 to $l-1$ and all its children are not leaves.

From Definition 3.1.6 we know

$$
\begin{aligned}
Benefit(h_{ij}) &= |\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color = blue\}| \\
&\quad -|\{h_{lk}|h_{lk} \in Cell(h_{ij}), h_{lk}.color \neq blue\}| - 1 \\
&= \sum_{h_{i+1,c} \in Cld(h_{ij})} |\{h_{lk}|h_{lk} \in Cell(h_{i+1,c}), h_{lk}.color = blue\}| \\
&\quad - \sum_{h_{i+1,c} \in Cld(h_{ij})} |\{h_{lk}|h_{lk} \in Cell(h_{i+1,c}), h_{lk}.color \neq blue\}| - 1 \\
&= \sum_{h_{i+1,c} \in Cld(h_{ij})} (|\{h_{lk}|h_{lk} \in Cell(h_{i+1,c}), h_{lk}.color = blue\}| \\
&\quad\quad - |\{h_{lk}|h_{lk} \in Cell(h_{i+1,c}), h_{lk}.color \neq blue\}| - 1) \\
&\quad + |Cld(h_{ij})| - 1 \\
&= |Cld(h_{ij})| - 1 + \sum_{h_{i+1,c} \in Cld(h_{ij})} Benefit(h_{i+1,c}) \qquad (3.1)
\end{aligned}
$$

From Equation 3.1 we can see that the benefit of a node in level 1 to $l-2$ can be computed from its children's benefits. The number of h_{ij}'s children is $|Cld(h_{ij})|$. After we aggregate its children regions to h_{ij}, all those children are covered by h_{ij}. So the new gained benefit is $|Cld(h_{ij})| - 1$. The sum of this new gained benefit and all benefits from children regions is the benefit for h_{ij}, which has been exactly presented in the Equation 3.1.

From Definition 3.1.6 and Equation 3.1 we have the following bottom-up algorithm for MDLE problem in a 1-dimensional hierarchy.

Algorithm 3.1.1. Bottom-Up Algorithm for MDL with exceptions in 1-dimensional hierarchy.

Input: a hierarchy h and a data set D;

Output: the optimal description for D;

1. Initial two stacks T and V;

   */*T is the stack of descriptions and V is the stack of benefits of descriptions associated with nodes. We use $T(h_{ij})$ to present the description and $V(h_{ij})$ to present the benefit of the description associated with node h_{ij} */*

2. Browse the hierarchy h by post-order;

   (a) For each node $h_{l-1,j}, 1 \leq j \leq n_{l-1}$:

      i. Compute the benefit based on Definition 3.1.6 and push $Benefit(h_{l-1,j})$ into stack V;

      ii. If $(Benefit(h_{l-1,j}) > 0)$

      push $S_{l-1,j} = h_{l-1,j} - \{h_{lk} | h_{lk} \in Cell(h_{l-1,j}), h_{lk}.color \neq blue\}$ into stack T;

      Else

      push $S_{l-1,j} = \{h_{lk} | h_{lk} \in Cell(h_{l-1,j}), h_{lk}.color = blue\}$ into stack T;

      End If

   (b) For each node $h_{ij}, 1 \leq i \leq l-2, 1 \leq j \leq n_i$

      i. Compute the benefit by Equation 3.1:

$$Benefit(h_{ij}) = |Cld(h_{ij})| - 1 + \sum_{h_{i+1,c} \in Cld(h_{ij})} Benefit(h_{i+1,c})$$

      ii. If $(Benefit(h_{ij}) > \sum_{h_{i+1,c} \in Cld(h_{ij}) \& V(h_{i+1,c}) > 0} V(h_{i+1,c}))$

      push $Benefit(h_{ij})$ into stack V;

      push $S_{ij} = h_{ij} - \{h_{lk} | h_{lk} \in Cell(h_{ij}), h_{lk} \neq blue\}$ into stack T;

      Else

      push $\sum_{h_{i+1,c} \in Cld(h_{ij}) \& V(h_{i+1,c}) > 0} V(h_{i+1,c})$ into stack V;

      push $S_{ij} = \bigcup_{h_{i+1,k} \in Cld(h_{ij})} T(h_{i+1,c})$ into stack T;

End If

iii. Popup h_{ij}'s children descriptions and values from T and V;

3. return $T(h_{11})$.

□

Example 3.1.3. The example in Figure 3.2 is a 4-level hierarchy. All blue cells are marked by ◯. The following table is the procedure to build the optimal description.

region	benefit	value in V	description in T
s	$3-1-1=1$	1	$s-d$
t	$1-1-1=-1$	-1	e
u	$3-1-1=1$	1	$u-j$
x	$1+(-1)+1+3-1=3$	3	$x-d-f-j$
v	$3-1-1=1$	1	$v-n$
w	$1-3-1=-3$	-3	o
y	$1+(-3)+2-1=-1$	1	$(v-n)+o$
z	$3+(-1)+2-1=3$	4	$(x-d-f-j)+(v-n)+o$

- node s. $Benefit(s) = 1$, we can use $(s-d)$ to cover the three blue cells, $\{a,b,c\}$. So $T(s) = s-d$ and $V(s) = 1$;

- node t. $Benefit(t) = -1$, if we use $(t-f)$ to cover the blue cell $\{e\}$, the length is 2; if we just use e then the length is 1. So $T(t) = e$;

- node x. $Benefit(x) = 3$, this is the benefit for the description $S_1 = x-d-f-j$. The sum of its children's positive values in V is 2, and this is the benefit for the description $S_2 = (s-d)+e+(u-j)$. Therefore we can gain more benefit

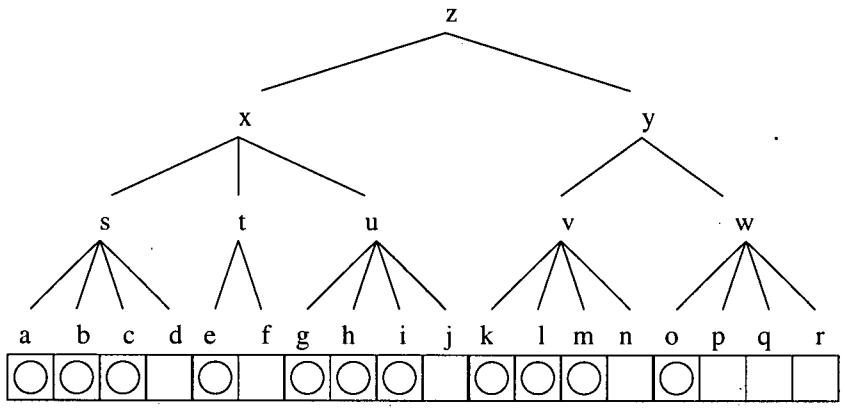


Figure 3.2: Example for Bottom-Up Algorithm

with S_1, which aggregates children regions to x. It is clear that the optimal description for x is $(x - d - f - j)$;

- node y. $Benefit(y) = -1$, and the sum of its children's positive values in V is 1. This means we can gain more benefit if we just use the optimal descriptions of its children to express y instead of aggregating children regions to y. So the optimal description for y is the union of its children's optimal description, that is $(v - n) + o$;

- node z. $Benefit(z) = 3$, and the sum of its children's positive values in V is 4, so the optimal description for z is the union of its children's optimal description, that is $(x - d - f - j) + (v - n) + o$. This is also the optimal description for z with respect to the set of blue cells in Figure 3.2.

□

Example 3.1.3 indicates that the bottom-up algorithm in Algorithm 3.1.1 generates an optimal description for each non-leaf node. We have the following theorem about the optimality of Algorithm 3.1.1.

Theorem 3.1.1. Algorithm 3.1.1 generates an optimal description for data set D. □

Proof. We show that description generated in Algorithm 3.1.1 is optimal for each node.

- For a node in level $l-1$, i.e. $h_{l-1,j}, 1 \leq j \leq n_{l-1}$: $h_{l-1,j}$'s children are all leaves. We can use two methods to cover the blue cells in $Cell(h_{l-1,j})$.

  1. enumerating the blue cells and we can not get any benefit;
  2. by $h_{l-1,j}$ with the exception of all non-blue cells, which is in the form as

  $$h_{l-1,j} - \{h_{lk} | h_{lk} \in Cell(h_{l-1,j}), h_{lk}.color \neq blue\}$$

  and the benefit is $Benefit(h_{l-1,j})$.

  Step2(a) of Algorithm 3.1.1 stores the description with more benefit. The description pushed into T is the optimal description for $h_{l-1,j}$.

- For a node in level 1 to $l-2$, i.e. $h_{ij}, 1 \leq i \leq l-2, 1 \leq j \leq n_i$: we have already generated optimal descriptions for its children. We can use two methods to cover the blue cells in $Cell(h_{ij})$ too. One is including h_{ij} in the description. The other is not including h_{ij}.

  1. Including h_{ij}: we only permit data cells in U_2 (Definition 3.1.7) so the description is $S = h_{ij} - \{h_{lk} | h_{lk} \in Cell(h_{ij}), h_{lk}.color \neq blue\}$ and the benefit is $Benefit(h_{l-1,j})$.
  2. Not including h_{ij}: we do not aggregate the children regions to h_{ij}. From the Definition 3.1.1 we know the hierarchy is a tree structure and any two nodes in the same level do not have overlapping children sets. So we can

deal with h_{ij}'s children region independently and use the union of children's description to express h_{ij}. The description in stack T is the optimal description for each child. $S_{ij} = \bigcup_{h_{i+1,k} \in Cld(h_{ij})} T(h_{i+1,c})$ is the description for this case and has benefit $\sum_{h_{i+1,c} \in Cld(h_{ij}) \& V(h_{i+1,c})>0} V(h_{i+1,c})$.

Step2(b) of Algorithm 3.1.1 stores the description with the more benefit between case 1 and case 2. The description pushed into T is the optimal description for h_{ij}.

Therefore, Algorithm 3.1.1 creates an optimal description for each node and the output for h_{11} is an optimal description for D in h. □

Theorem 3.1.2. The running time of MDL with exceptions for a 1-dimensional hierarchy h is $O(|h|)$. □

Proof. In Algorithm 3.1.1, for each node in level $l-1$ we need to find its children to compute the benefit. So all leaves will be visited once. Each node in level $l-1$ to 1 has two 'push' and 'pop' operations with stack T and stack V. Therefore Algorithm 3.1.1 can be done in $O(|h|)$. This means the time complexity of MDLE is linear to the number of nodes in h. □

3.2 NP-Completeness for 2-Dimensional Hierarchy

We have seen that MDLE in a 1-dimensional hierarchy is tractable and we proposed an algorithm to get a description with the minimum length. In this section we discuss the case when the data cube has a 2-dimensional hierarchical structure.

3.2.1 Basic Definitions

We have shown the definition for 1-dimensional hierarchy in Section 3.1.1. The related concepts of 2-dimensional hierarchy are similarly defined in this section.

Definition 3.2.1. A 2-dimensional hierarchical data cube is in the form of $H =< H_1, H_2 >$ where both H_1 and H_2 are hierarchies defined in Definition 3.1.1. □

Definition 3.2.2. In a 2-dimensional hierarchical data cube $H =< H_1, H_2 >$, and for some $x_1 \in H_1, x_2 \in H_2$, we have

- (x_1, x_2) is a data cell iff $x_1 \in Leaves(H_1)$ *and* $x_2 \in Leaves(H_2)$;
- (x_1, x_2) is a data region iff $x_1 \notin Leaves(H_1)$ *or* $x_2 \notin Leaves(H_2)$.

□

Definition 3.2.3. Given a data region (x_1, x_2) in $H =< H_1, H_2 >$, $Cell(x_1, x_2)$ is the set of all data cells covered by (x_1, x_2),

$$Cell(x_1, x_2) = \{(x_1', x_2') | x_1' \in Leaves(H_1), x_2' \in Leaves(H_2), x_1' \in Des(x_1), x_2' \in Des(x_2)\}$$

□

In order to prove the NP-Completeness of MDLE for a 2-dimensional hierarchy, we only consider the 2-level hierarchy. We can represent a data cube with 2-dimensional 2-level hierarchical structure by a matrix and call one dimension as row and the other as column. We define the 2-dimensional 2-level hierarchical data cube in a more simple way in Definition 3.2.4.

Definition 3.2.4. Given a 2-dimensional 2-level hierarchial data cube $H = R \times C$, $R = \{r_1, r_2, ..., r_{n_r}\}$, r_0 is the parent of $r_i, 1 \leq i \leq n_r$, $C = \{c_1, c_2, ..., c_{n_c}\}$, c_0 is the parent of $c_j, 1 \leq j \leq n_c$. For some $r_i \in R, c_j \in C$,

- $d_{ij} = (r_i, c_j)$ is a data cell if $r_i \in R, c_j \in C$;
- $d_{ij} = (r_i, c_j)$ is a data region if $r_i = r_0$ *or* $c_j = c_0$;

□

In a 2-dimensional 2-level hierarchical data cube, only r_0 and c_0 are not leaves. All nodes in R and C are leaves. The nodes in R are called rows. The nodes in C are called columns.

The benefit of a row in $H = R \times C$ is similarly defined as Definition 3.1.6.

Definition 3.2.5. A row region $r_i \in R$ is defined as $r_i = (r_i, c_0) = \{(r_i, c_j) | c_j \in C\}$, and the benefit of r_i is:

$$\begin{aligned} Benefit(r_i) &= |\{d_{ij} | d_{ij} \in Cell(r_i, c_0), d_{ij}.color = blue\}| \\ &\quad -|\{d_{ij} | d_{ij} \in Cell(r_i, c_0), d_{ij}.color \neq blue\}| - 1 \\ &= |\{d_{ij} | d_{ij}.color = blue, 1 \leq j \leq n_c\}| \\ &\quad -|\{d_{ij} | d_{ij}.color \neq blue, 1 \leq j \leq n_c\}| - 1 \end{aligned}$$

□

The benefit of a column is similarly defined as the benefit of a row in Definition 3.2.5.

Example 3.2.1. The example in Figure 3.3 is a 7×7 2-dimensional 2-level hierarchical data cube. The hierarchical structure have been showed in Figure 3.3. The data cells marked with '◯' are blue cells. This data cube has 49 data cells and the interesting data set D has 28 cells. Rows are $R = \{a, b, c, d, e, f, g\}$. Columns are $C = \{1, 2, 3, 4, 5, 6, 7\}$. r_0 is 'i', which is parent of all nodes in R. c_0 is '8', which is

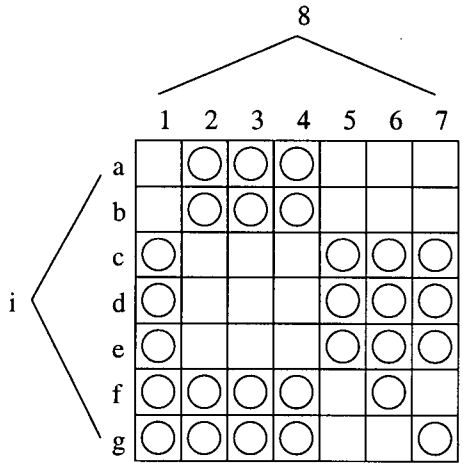


Figure 3.3: An Example for 2-dimensional 2-level hierarchy

parent of all nodes in C. $(a, 2)$ is a data cell. $(i, 2)$ and $(b, 8)$ are data regions. $(i, 1)$ contains 5 blue cells and 2 non-blue cells. So $Benefit(i, 1) = 5 - 2 - 1 = 2$.

□

3.2.2 Problem Statements

Problem 1: MDL with Exceptions

The definition of the description for a data set D in a 2-dimensional 2-level hierarchical data cube is similarly defined as the description in a 1-dimensional hierarchy which is given in Definition 3.1.7. The only difference is that the data cell and data region are in the forms defined in Definition 3.2.4.

Definition 3.2.6. Given a 2-dimensional 2-level hierarchical data cube H and an

interesting data set D, the description S for D is in the form as

$$S = U_1 - U_2 = \bigcup(r_i, c_j) - \bigcup(r_k, c_l),$$
$$0 \leq i \leq n_r, 0 \leq j \leq n_c$$
$$1 \leq k \leq n_r, 1 \leq l \leq n_c$$

and S covers exactly all the data cells in D, i.e. $Cell(S) = D$. □

The length and benefit of a description is similarly defined as Definition 3.1.8.

The problem of MDLE, MDL with exceptions, is defined in the same way as Definition 3.1.9.

Example 3.2.2. Let us still use the example in Figure 3.3.

Firstly we compute the benefits for each row and column and show them in the following table.

rows	length	benefit	columns	length	benefit
(f,8),(g,8)	3	2	(i,1)	3	2
(c,8),(d,8),(e,8)	4	0	(i,2),(i,3),(i,4),(i,6),(i,7)	4	0
(a,8),(b,8)	5	-2	(i,5)	5	-2

The optimal description with the minimum length is:

$$\begin{aligned} S_{Opt} &= \{(c,8) + (d,8) + (e,8) + (i,2) + (i,3) + (i,4)\} \\ &\quad -\{(c,2) + (c,3) + (c,4) + (d,2) + (d,3) + (d,4) + (e,2) + (e,3) + (e,4)\} \\ &\quad +(f,1) + (g,1) + (f,6) + (g,7) \end{aligned}$$

The length of S_{Opt} is 19. So the benefit of S is

$$Benefit(S_{Opt}) = |D| - |S_{Opt}| = 28 - 19 = 9$$

S_{Opt} is the description with the minimum length and the maximum benefit

□

From this example we can see $(f,8),(g,8)$ and $(i,1)$ have the most benefits in all rows and columns, but they are not in the optimal description. Even though rows $(c,8),(d,8),(e,8)$ and columns $(i,2),(i,3),(i,4)$ do not have positive benefit (benefits are 0), we can get more benefits by selecting all of them. Therefore a row/column with positive benefit may be not included in the optimal description and a row/column with non-positive benefit may be included in the optimal description.

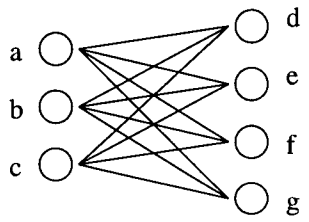
In a 1-dimensional hierarchy, when we generate the description for a region from its children regions, we aggregate all its children (Step2(a) in Algorithm 3.1.1) or just unite the optimal descriptions of the children regions with positive benefits (Step2(b) in Algorithm 3.1.1). Each way keeps the optimality in the children regions. But in a 2-dimensional 2-level hierarchical data cube, the intersections between rows and columns makes it impossible to keep the optimality from lower level regions. This induce the MDLE problem to an NP-Complete problem in a 2-dimensional hierarchical data cube. We show the proof in the next section.

3.2.3 NP-Completeness Proof Part I: CWE Bipartite Graph

In this part, we show the proof that MDLE problem is an NP-Complete problem. We firstly prove that it is NP-Complete to find a maximum induced subgraph in a complete edge-weighted (CEW) bipartite graph, which has 2-value weights on edges. We prove the NP-Completeness of CEW from a known NP-Complete problem: CLIQUE. Then we prove the MDLE problem is NP-Complete from CEW.

Problem2: CEW Bipartite Graph

Definition 3.2.7. The complete edge-weighted (CEW) bipartite graph with 2-level weights is in the form of $G_e = (V,E), V = V_1 \cup V_2$, for any $v_i \in V_1, v_j \in V_2, e =$



Weights on Edges

	d	e	f	g
a	−1	1	−1	1
b	−1	1	−1	−1
c	−1	−1	1	−1

Figure 3.4: Complete Edge-Weighted Bipartite Graph

$(v_i, v_j) \in E$, the weights on edges have two values, i.e. $wt(e) \in \{w_1, w_2\}$, and at least one of w_1, w_2 is negative. □

Definition 3.2.8. For any subgraph $G'_e = (V', E')$ in G_e induced by $V' \subseteq V$, the weight of G'_e is defined as:

$$wt(G'_e) = \sum_{e \in E'} wt(e)$$

□

Definition 3.2.9. The maximum induced subgraph problem in CEW bipartite graph is to find a subgraph $G'_e = (V', E')$ induced by $V' \subseteq V$ such that $(wt(G'_e) + |V'|)$ is maximized. □

Example 3.2.3. Figure 3.4 is an example of CEW bipartite graph. $G_e = (V, E), V = V_1 \cup V_2, V_1 = \{a, b, c\}, V_2 = \{d, e, f, g\}$ and the weights on edges have been shown in the figure.

If $V' = \{a, b, e\}$, then the weight of the induced subgraph G'_e is $wt(G'_e) = 2$, and $wt(G'_e) + |V'| = 2 + 3 = 5$.

If $V'' = \{a, b, e, g\}$, then the weight of the induced subgraph G''_e is $wt(G''_e) = 2$, and $wt(G'_e) + |V''| = 2 + 4 = 6$.

□

After we check all the induced subgraphs of G_e, we know that G''_e is the maximum induced subgraph in G_e. In fact, because of interactions between the two level weights on edges and the cardinality of vertices set, it is hard to find a maximum induced subgraph in a CEW bipartite graph.

Now we prove the induced subgraph problem in CEW bipartite graph (Definition 3.2.9), to find a subgraph G'_e induced by V' in G_e such that $wt(G'_e)+|V''| \geq K$, is an NP-Complete problem. In order to prove this, we make a reduction, which is triggered by the reduction in the proof of BEB problem in [7][12], from the clique problem which is a well known NP-Complete problem.

Theorem 3.2.1. Clique is a known NP-Complete problem[16][5].

Instance: Graph $G = (V, E)$, positive integer $K \leq |V|$.

Question: Does G contain a clique of size K or more, i.e., a subset $V' \subseteq V$ with $|V'| \geq K$ such that every two vertices in V' are joined by an edge in E?

□

Lemma 3.2.1. The induced subgraph problem in a CEW bipartite graph is an NP-Complete problem.

Instance: A complete edge weighted(CEW) bipartite graph $G_e = (V_1, V_2, E)$, $wt(e) \in \{w_1, w_2\}$, and at least one of w_1, w_2 is negative; an integer K.

Question: Does G_e contain an induced subgraph $G' = (V', E')$ such that $|V'| + wt(G') \geq K$?

□

Proof. The reduction is from the clique problem.

Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$. Construct a complete bipartite graph $G_e = (V, V', E')$ with $V' = V$ and the set of edges $E' = \{(v, u)|v \in V, u \in V'\}$. The weight function wt is defined as:

For any $e = (v, u) \in E$:

$$wt(e) = \begin{cases} 0 & v = u \\ 0 & e \in E \\ -1 & e \notin E \end{cases}$$

We now show that G contains a clique of size K or more if and only if G_e contains an induced subgraph $G'_e = (V'', E'')$ such that $|V''| + wt(G'_e)| \geq 2K$.

Firstly, if G_e contains an induced subgraph $G'_e = (V'', E'') = (V_b, V_b, E'')$ such that $|V''| + wt(G_e) \geq 2K$ or more (from the construction of G_e we know that the two vertices sets V and V' have the same properties, so if any $v \in V$ is contained in G_e, the correspondent vertex $v' \in V'$ is included in G_e too. That is why $V' = V_b \cup V_b$). The weights on edges in G'_e has two cases:

1. $\forall e \in E', wt(e) = 0$

   The induced subgraph by V_b in G is a clique. Because no edge in G_e has weight -1, this means all nodes in V_b are connected in G by the definition of weight function $wt(e)$. $|V'| + wt(G_e) = 2|V_b| \geq 2K$. The clique induced by V_b in G is of size K or more.

2. $\exists e \in E', wt(e) = -1$

   If $e' = (v', u') \in E', wt(e') = -1$, then v', u' is not connected in G. Let us delete v' from V_b and the three edges $(v', v'), (v', u'), (u', v')$ from E'. $wt(v', v') = 0, wt(v', u') = wt(u', v') = -1$. At the same time, we need delete all other edges incident on v', and those edges have weight 0 or -1. Therefore the total weight of all deleted edges is -2 or less. So the left over biclique still keeps $|V'| + wt(E') >= (|V'| - 2) + (wt(E') - (-2)) \geq 2K$. After we delete all the

edges with weight -1, the left over edges only have weight 0. It comes back to case 1.

Therefore if we can find an induced subgraph $G'_e = (V'', E'')$ in G_e such that $|V'| + wt(G'_e) \geq 2K$, we can also find a clique in G whose size is K or more.

On the other hand, let us suppose G contains a clique of size K or more. Let us use V_c to present the set of vertices in this clique, so $|V_c| \geq K$. The induced subgraph in G_e is $G'_e = (V', E') = (V_c, V'_c, E'), V'_c = V_c, E' = \{(v, u) | v \in V_c, u \in V'_c\}$. Because V_c induces a clique in G, any two nodes in V_c are connected. Therefore all edges in E' have weight 0. So for G'_e we have

$$\begin{aligned} |V'| + wt(G'_e) &= 2 \times |V_c| + \sum_{v \in V_c, u \in V'_c, e=(v,u) \in E} wt(e) \\ &= 2 \times |V_c| + \sum_{v \in V_c, u \in V'_c, e=(v,u) \in E} (0) \\ &= 2 \times |V_c| \geq 2K \end{aligned}$$

Clique is an NP-Complete problem, so it is NP-Hard to find a subgraph G'_e induced by V'' in G_e such that $|V''| + wt(G'_e) \geq K$.

Given an induced subgraph $G'_e = (V'', E'')$ in G_e there exists a polynomial time algorithm that can check whether $|V''| + wt(G''_e) \geq 2K$. So this problem is in NP.

Now we have proved that this problem is an NP-Complete problem.

□

Example 3.2.4. An example for the proof of Lemma 3.2.1 is given in Figure 3.5. Graph $G = (V, E), V = \{1, 2, 3, 4\}, E = \{(1, 2), (1, 4), (2, 4), (2, 3), (3, 4)\}$.

We can construct a complete bipartite graph $G_e = (V, V, E')$ and only edges $(1, 3)$ and $(3, 1)$ have weight -1, all other edges have weight 0.

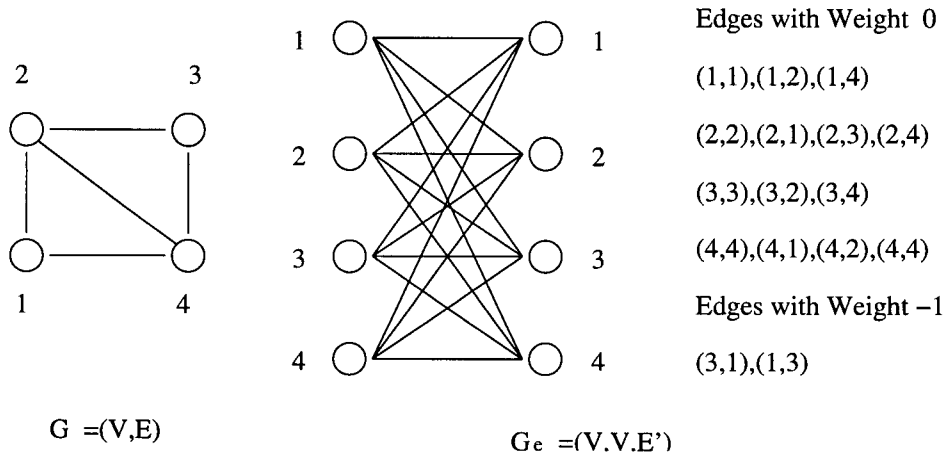


Figure 3.5: Reduction From Clique

In G, vertices $\{1, 2, 4\}$ induce a clique of size 3. Then in graph G_e, the subgraph G'_e induced by $V' = \{1, 1, 2, 2, 4, 4\}$ has six vertices and all edges have weight 0. So for G'_e, $|V'| + wt(G'_e) \geq 6 = 2 \times 3$.

On the other hand, if we find an induced subgraph $G'_e = (V', E')$ in G_e such that $|V'| + wt(G'_e) \geq 2K$, let us discuss two cases:

- $\forall e \in E', wt(e) = 0$.

  For example G'_e is induced by $V' = \{2, 3, 4, 2, 3, 4\}$, $|V'| = 6$, and all edges in E' have weight 0, so $wt(G'_e) = 0$, $|V'| + wt(G'_e) = 6$. This means in G there exists one clique of size 3, which contains vertices $\{2, 3, 4\}$.

- $\exists e \in E', wt(e) = -1$. For example G'_e contains all vertices of G_e. $|V'| + wt(G'_{1e}) = 8 + (-2) = 6$. The weights of edge $(1, 3)$ and edge $(3, 1)$ are -1. Now let us delete vertices $\{3, 3\}$ from G'_e, and all the edges incident on $\{3, 3\}$, which are $(3, 3), (3, 1), (3, 2), (3, 4)$ and $(1, 3), (2, 3), (4, 3)$. The total weight of those deleted edges are -2. The left over subgraph is $G''_e = (V'', E'')$, and $V'' = V' - \{3, 3\} = \{1, 2, 4, 1, 2, 4\}$, $wt(G''_e) = wt(G'_e) - (-2) = -2 - (-2) = 0$,

so

$$|V''| + wt(G''_e) = 6 + 0 = 6$$

Now G''_e only contains $\{1, 2, 4, 1, 2, 4\}$, so $\{1, 2, 3\}$ induces a clique of size 3 in graph G.

□

3.2.4 NP-Completeness Proof Part II: MDLE

Before we show the proof of the NP-Completeness of MDLE, let us discuss about the form of the description for a data cube first.

From Definition 3.2.6 we know a description for the data set D in a 2-dimensional 2-level hierarchical data cube H is in the form of:

$$S = U_1 - U_2 = \bigcup(r_i, c_j) - \bigcup(r_k, c_l),$$
$$0 \leq i \leq n_r, 0 \leq j \leq n_c$$
$$1 \leq k \leq n_r, 1 \leq l \leq n_c$$

Based on the data regions in U_1, the descriptions can be classified into two cases:

1. $U_1 = \{(r_0, c_0)\}$ and $S = (r_0, c_0) - \{d_{ij} | d_{ij} \in Cell(r_0, c_0), d_{ij}.color \neq blue\}$. The whole region, (r_0, c_0), with the exceptions of all the non-blue cells are the blue cells in D. For a given instance it is directly to get this description and its benefit is constant.

$$|S| = 1 + |\{d_{ij} | d_{ij} \in Cell(r_0, c_0), d_{ij}.color \neq blue\}|$$

$$\begin{aligned} Benefit(S) &= |D| - |S| \\ &= |\{d_{ij} | d_{ij} \in Cell(r_0, c_0), d_{ij}.color = blue\}| \\ &\quad -|\{d_{ij} | d_{ij} \in Cell(r_0, c_0), d_{ij}.color \neq blue\}| - 1 \end{aligned}$$

2. $U_1 = \{(r_i, c_j) | 1 \leq i \leq n_r, 1 \leq j \leq n_c\}$ and the description is in the form of

$$\begin{aligned} S &= U_1 - U_2 \\ &= \bigcup_{r_i \in R_1, R_1 \subseteq R} r_i + \bigcup_{c_j \in C_1, C_1 \subseteq C} c_j \\ &\quad -\{d_{ij} | d_{ij} \in Cell(R_1) \cup Cell(C_1), d_{ij}.color \neq blue\} \\ &\quad +\{d_{ij} | d_{ij} \notin Cell(R_1) \cup Cell(C_1), d_{ij}.color = blue\} \end{aligned} \tag{3.2}$$

S has three parts. Firstly, S contains all the rows in R_1 and columns in C_1. Secondly, S excepts the non-blue cells contained in R_1 and C_1. Now S covers all and only the blue cells contained in R_1 and C_1. Finally, by adding the blue cells not contained in R_1 and C_1, S covers all the blue cells in D. The intersections between rows and columns will make the selections of rows and columns complicated(as showed in Example 3.2.2). For a given instance it is hard to find a description in this form with the most benefit directly. In the following part of Section 3.2.4, we only consider the description in the form of Equation 3.2.

Based on Lemma 3.2.1, we prove the MDLE problem is NP-Hard by a reduction from CEW subgraph problem.

Theorem 3.2.2. The following problem is NP-Complete:

Instance: A data cube $H = R \times C$, an interesting data set D and a positive integer K.

Question: Does there exist a description S in the form of

$$\begin{aligned} S = &\bigcup_{r_i \in R_1, R_1 \subseteq R} r_i + \bigcup_{c_j \in C_1, C_1 \subseteq C} c_j - \{d_{ij} | d_{ij} \in Cell(R_1) \cup Cell(C_1), d_{ij}.color \neq blue\} \\ &+\{d_{ij} | d_{ij} \notin Cell(R_1) \cup Cell(C_1), d_{ij}.color = blue\} \end{aligned}$$

which has benefit K or more?

□

Proof. Given an instance of complete edge-weighted(CEW) bipartite graph, $G_e = (V, E), V = V_1 \cup V_2, e \in E, wt(e) = \pm 1$ and a positive integer K', we have known from Lemma 3.2.1 that it is NP-Complete to find a subgraph $G_e = (V', E')$ induced by V' in G_e such that $|V'| + wt(B) \geq K'$.

We construct a data cube H' based on graph G_e:

1. $H' = V_1 \times V_2$, H' is a $|V_1| \times |V_2|$ data cube: it has $|V_1|$ rows and $|V_2|$ columns;

2. for $e = (v_i, v_j), v_i \in V_1, v_j \in V_2$, the weight function $wt(e)$ is:

$$\begin{cases} \text{if } wt(e) = 1, & \text{mark } d_{ij} = (v_i, v_j) \text{ in } H' \text{ as a non-blue cell;} \\ \text{if } wt(e) = -1, & \text{mark } d_{ij} = (v_i, v_j) \text{ in } H' \text{ as a blue cell.} \end{cases}$$

From the weight function $wt(e)$ we know that a blue cell in H' corresponds to an edge with weight -1 in G_e and a non-blue cell in H' corresponds to an edge with weight 1 in G_e. For a row/column in data cube H', $v \in V_1 \cup V_2$, the benefit of v can be computed in the following way:

$$\begin{aligned} Benefit(v) &= |\{d_{ij} | d_{ij} \in Cell(v), d_{ij}.color = blue\}| \\ &\quad -|\{d_{ij} | d_{ij} \in Cell(v), d_{ij}.color \neq blue\}| - 1 \\ &= |\{e = (v,u) | e \in E, wt(e) = -1\}| \\ &\quad -|\{e = (v,u) | e \in E, wt(e) = 1\}| - 1 \\ &= \{- \sum_{e=(v,u)\in E, wt(e)=-1} wt(e)\} - \{\sum_{e=(v,u)\in E, wt(e)=1} wt(e)\} - 1 \\ &= -1 - \sum_{e=(v,u)\in E} wt(e) \end{aligned} \tag{3.3}$$

For any $v_i \in V_1, v_j \in V_2$, the description with exceptions to cover the blue

cells in v_i and v_j is

$$S_{ij} = v_i + v_j - \{d_{ij} | d_{ij} \in Cell(v_i) \cup Cell(v_j), d_{ij}.color \neq blue\}$$

The total benefit of $v_i \& v_j$ is the number of blue cells in $Cell(v_i) \cup Cell(v_j)$ minus the length of S_{ij},

$$\begin{aligned}
Benefit(v_i \& v_j) &= |\{d_{kl} | d_{kl} \in Cell(v_i) \cup Cell(v_j), d_{kl}.color = blue\}| \\
&\quad -(|\{d_{kl} | d_{kl} \in Cell(v_i) \cup Cell(v_j), d_{kl}.color \neq blue\}| + 2) \\
&= (|\{d_{il} | d_{il} \in Cell(v_i), d_{il}.color = blue\}| \\
&\quad -|\{d_{il} | d_{il} \in Cell(v_i), d_{il}.color \neq blue\}| - 1) \\
&\quad +(|\{d_{kj} | d_{kj} \in Cell(v_j), d_{kj}.color = blue\}| \\
&\quad -|\{d_{kj} | d_{kj} \in Cell(v_j), d_{kj}.color \neq blue\}| - 1) \\
&\quad + \begin{cases} -1 & d_{ij} \in Cell(v_i) \cap Cell(v_j), d_{ij}.color = blue \\ 1 & d_{ij} \in Cell(v_i) \cap Cell(v_j), d_{ij}.color \neq blue \end{cases} \\
&= Benefit(v_i) + Benefit(v_j) \\
&\quad + \begin{cases} -1 & d_{ij} \in Cell(v_i) \cap Cell(v_j), d_{ij}.color = blue \\ 1 & d_{ij} \in Cell(v_i) \cap Cell(v_j) v_j, d_{ij}.color \neq blue \end{cases}
\end{aligned} \tag{3.4}$$

From Equation 3.4 we can know that, given a row $v_i \in V_1$ and a column $v_j \in V_2$, the intersecting cell, i.e. d_{ij}, will be counted twice in $Benefit(v_i) + Benefit(v_j)$. So total benefit of v_i and v_j can be computed in two ways based on the color of d_{ij}:

- if $d_{ij}.color = blue$, then d_{ij} appears twice in the positive part of $Benefit(v_i)$ and $Benefit(v_j)$. So the total benefit is $Benefit(v_i) + Benefit(v_j) - 1$;
- if $d_{ij}.color \neq blue$, then d_{ij} appears twice in the negative part of $Benefit(v_i)$ and $Benefit(v_j)$. So the total benefit is $Benefit(v_i) + Benefit(v_j) + 1$.

Given a description S' generated from R'_1, C'_1 where $R'_1 \in V_1, C'_1 \in V_2$, S' is in the form of

$$\begin{aligned} S' &= \bigcup_{r_i \in R'_1} r_i + \bigcup_{c_j \in C'_1} c_j - \{d_{ij} | d_{ij} \in Cell(R'_1) \cup Cell(C'_1), d_{ij}.color \neq blue\} \\ &+ \{d_{ij} | d_{ij} \notin Cell(R'_1) \cup Cell(C'_1), d_{ij}.color = blue, \} \end{aligned}$$

The benefit of S' is generated by using rows in R'_1 and columns in C'_1 to cover the blue cells in those rows and columns. So based on Equation 3.3, Equation 3.4 and the construction of H', the benefit of S' can be rewritten as: (Let $V' = R'_1 \cup C'_1, E' = \{(v,u) | v,u \in V'\}, V'' = V - V', E'' = \{(v,u) | v,u \in V''\}$)

$$\begin{aligned} Benefit(S') &= \sum_{r_i \in R'_1} Benefit(r_i) + \sum_{c_j \in C'_1} Benefit(c_j) \\ &+ \sum_{d_{ij} \in Cell(r_i) \cap Cell(c_j), d_{ij}.color = blue} (-1) + \sum_{d_{ij} \in Cell(r_i) \cap Cell(c_j), d_{ij}.color \neq blue} (1) \\ &= \sum_{v \in V'} Benefit(v) + \sum_{v,u \in V', e=(v,u), wt(e)=-1} wt(e) + \sum_{v,u \in V', e=(v,u), wt(e)=1} wt(e) \\ &= \sum_{\forall v \in V'} \{-1 - \sum_{e=(v,u) \in E} wt(e)\} + \sum_{\forall e=(v,u) \in E'} wt(e) \\ &= -|V'| - \{2 \sum_{\forall e=(v,u) \in E'} wt(e) + \sum_{v \in V', u \in V-V', e=(v,u) \in E} wt(e)\} \\ &+ \sum_{\forall e=(v,u) \in E'} wt(e) \\ &= -|V'| - \{\sum_{v,u \in V', e=(v,u) \in E'} wt(e) + \sum_{v \in V', u \in V-V', e=(v,u) \in E} wt(e)\} \\ &= -|V'| - \sum_{v \in V', u \in V, e=(v,u) \in E} wt(e) \\ &= -(|V| - |V''|) - \{\sum_{v,u \in V, e=(v,u) \in E} wt(e) - \sum_{v,u \in V-V', e=(v,u) \in E} wt(e)\} \\ &= -\{|V| + \sum_{v,u \in V, e=(v,u) \in E} wt(e)\} \\ &+ \{|V''| + \sum_{v,u \in V'', e=(v,u) \in E''} wt(e)\} \end{aligned} \tag{3.5}$$

From Definition 3.2.8 the total weight of $G_e = (V, E)$ is:

$$wt(G_e) = \sum_{e=(v,u),e\in E} wt(e) \tag{3.6}$$

The total weight of a subgraph $G'_e = (V'', E'')$ in graph G_e is:

$$wt(G''_e) = \sum_{e=(v,u),e\in E''} wt(e) \tag{3.7}$$

Applied Equation 3.6 and Equation 3.7 to Equation 3.5 we can easily get that:

$$Benefit(S') = -\{|V| + wt(G_e)\} + \{|V''| + wt(G'_e)\} \tag{3.8}$$

Therefore if we can find R'_1, C'_1 that can generate a description with benefit $K'' = -\{|V| + wt(G_e)\} + K'$ or more for D, then we can easily build the subgraph $G'_e = (V'', E'')$ in G_e induced by $V'' = V - R'_1 \cup C'_1$ such that $|V''| + wt(G'_e) \geq K'$.

On the other hand, if we can find a subgraph $G'_e = (V'', E'')$ in G_e induced by $V'' = V - R'_1 \cup C'_1$ such that $|V''| + wt(G'_e) \geq K'$, from Equation 3.8 we can build a description based on R'_1, C'_1 that has benefit K'' or more in the form of:

$$\begin{aligned} S \;=\; & \bigcup_{r_i\in R'_1} r_i + \bigcup_{c_j\in C'_1} c_j - \{d_{ij} | d_{ij} \in Cell(R'_1) \cup Cell(C'_1), d_{ij}.color \neq blue\} \\ & + \{d_{ij} | d_{ij} \notin Cell(R'_1) \cup Cell(C'_1), d_{ij}.color = blue\} \end{aligned}$$

Because it is NP-Complete to fine an induced subgraph G'_e in G_e such that $|V''| + wt(G'_e) \geq K'$, it is NP-hard to find the description with benefit K or more for data set D in data cube H.

Now we show that the MDLE problem is in NP.

Given a data cube $H = R \times C$, an interesting data set D and a positive integer K, for any description S for D, the algorithm will first check whether this

description has covered all the blue cells and does not cover any non-blue cells. Then the benefit of S will be computed and the algorithm will check whether the benefit is greater than K. This can be performed in polynomial time. So this problem is also in NP.

Therefore this is an NP-Complete problem. □

Example 3.2.5. An example is given in Figure 3.6 and Figure 3.7.

In Figure 3.6, the complete bipartite graph $G = (V, E)$ has weights on edges. $|V| = 9, wt(G) = -4$. Let $K = 9$, that is we want to find an induced subgraph $G' = (V', E')$ such that $|V'| + wt(G') \geq 9$.

We can construct a 2-dimensional 2-level data region D as in Figure 3.7. The data cells are marked by '◯' are blue cells and correspond to the edges with weight -1 in graph G. The unmarked data cells are non-blue cells and correspond to the edges with weight 1 in graph G. $K' = -(|V| + wt(G)) + K = -(9-4)+9 = 4$, that is to find a description with benefit 4.

We can find a description with benefit 4 for D as :

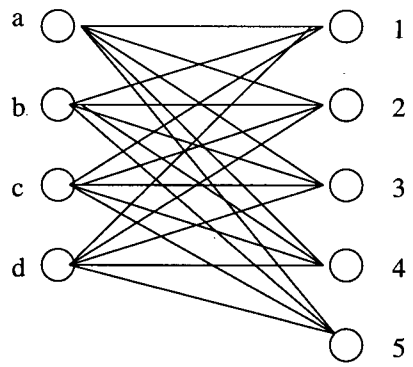
$$S = a + d + 1 + 3 + (b, 5) - \{(a, 1) + (a, 3) + (d, 3)\}$$

$$Benefit(S) = |D| - |S| = 12 - 8 = 4$$

S contains rows $R_1 = \{a, d\}$ and columns $C_1 = \{1, 3\}$. Based on this description, we can construct the induced subgraph $G' = (V', E'), V' = V - R_1 \cup C_1 = \{b, c, 2, 4, 5\}, wt(G') = 5 - 1 = 4$, so $|V'| + wt(G') = 5 + 4 = 9$.

□

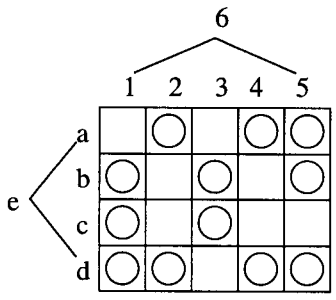
The problem in Theorem 3.2.2 is the decision problem. The optimization version of this problem is to find a description with the maximum benefit(equivalently, with the minimum length) for a data set.



Weights on Edges

	1	2	3	4	5
a	1	−1	1	−1	−1
b	−1	1	−1	1	−1
c	−1	1	−1	1	1
d	−1	−1	1	−1	−1

Figure 3.6: CEW Bipartite Graph:G=(V,E)



Data Cells marked by ◯ are blue cells.
These cells correspond to the edges with weight −1.

Unmarked cells non−blue cells.
These cells correspond to the edges with weight 1.

Figure 3.7: Correspondent Data Cube

Corollary 3.2.1. MDLE is an NP-Hard problem, i.e.it is NP-Hard to find a description with exceptions of the maximum benefit(equivalently,of the minimum length) for a 2-dimensional 2-level hierarchical data cube. □

We have shown that in a 2-dimensional 2-level data cube, the MDLE problem is already NP-Hard. So for the general case, in a multi-dimensional hierarchical data cube, this problem is also an NP-Hard problem.

Corollary 3.2.2. MDLE is an NP-Hard problem in a multidimensional hierarchical data cube. □

Chapter 4

Heuristics and Experiments

In previous chapter, we have analyzed the complexity of the MDLE problem and we have proved that it is an NP-Hard problem. In this chapter we study one tractable case and then propose three heuristics for the MDLE problem. We also compare these three heuristics for MDLE with MDL-Tree Algorithm and GMDL method.

4.1 A Tractable Case: No Shared Holes

Before we go to heuristics for MDLE, let us study a tractable case.

4.1.1 Preliminaries

In a 2-dimensional 2-level hierarchical data cube, given a row r_i and a column c_j, the intersecting cell is the data cell covered by r_i and c_j, i.e. $d_{ij} = (r_i, c_j)$.

From Equation 3.4 in the proof of Theorem 3.2.2, we know that,

- if d_{ij} is a blue cell, then the total benefit of r_i and c_j is

$$Benefit(r_i) + Benefit(c_j) - 1$$

- if d_{ij} is a non-blue cell, then the total benefit of r_i and c_j is

$$Benefit(r_i) + Benefit(c_j) + 1$$

We call the intersecting cell of a row and a column as the shared cell of this row and this column. For example, $d_{ij} = (r_i, c_j)$ is the shared cell of row r_i and column c_j. A shared cell can be a blue cell or a non-blue cell, which is the reason that the selections between rows and columns are hard. If a shared cell is a non-blue cell then we call it a shared hole.

Given an interesting data set D in a 2-dimensional 2-level hierarchical data cube $H = R \times C$, a description S for D is $(R_1 \in R, C_1 \in C)$:

$$\begin{aligned} S \;=\; & \bigcup_{r_i \in R_1} r_i + \bigcup_{c_j \in C_1} c_j - \{d_{ij} | d_{ij} \in Cell(R_1) \cup Cell(C_1), d_{ij}.color \neq blue\} \\ & + \{d_{ij} | d_{ij} \notin Cell(R_1) \cup Cell(C_1), d_{ij}.color = blue\} \end{aligned}$$

A row r_i is in S iff $r_i \in R_1$. S covers the rows in $R_1 \subseteq R$ and the columns in $C_1 \subseteq C$. If rows in R_1 and columns in C_1 do not share any non-blue cells, i.e.$\forall d_{ij} \in Cell(R_1) \cap Cell(C_1), d_{ij} = blue$, then we say S is a description without shared holes.

If the optimal description for D in H is a description without shared holes then we say that D does not contain shared holes.

In the following part of Section 4.1 we only discuss the case that does not contain shared holes. This means for any $r_i \in R_1$ and $c_j \in C_1$, $d_{ij} = (r_i, c_j)$ is a blue cell and the total benefit of r_i and r_j is

$$Benefit(r_i \& r_j) = Benefit(r_i) + Benefit(r_j) - 1 \tag{4.1}$$

From Equation 4.1 we know that in a data cube without shared holes, the selection

of a row and a column decreases the sum of the benefit of this row and the benefit of this column by 1. Therefore we can get the following theorem:

Theorem 4.1.1. In a 2-dimensional 2-level data cube without shared holes, the rows and columns contained in S_{Opt} have non-negative benefits. □

Proof. From Equation 3.5 we know the benefit of S_{Opt} is (R_{Opt} is the set of rows covered by S_{Opt} and C_{Opt} is the set of columns covered by S_{Opt}):

$$\begin{aligned} Benefit(S_{Opt}) &= \sum_{r_i \in R_{Opt}} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) \\ &\quad + \sum_{d_{ij} \in Cell(R_{Opt}) \cap Cell(C_{Opt}), d_{ij}.color = blue} (-1) \\ &= \sum_{r_i \in R_{Opt}} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) \\ &\quad - |R_{Opt}| \times |C_{Opt}| \end{aligned} \tag{4.2}$$

Suppose there is a row with negative benefit in S_{Opt}, such as $r_o \in R_{Opt}$ and $Benefit(r_o) < 0$. Let $R'_1 = R_{Opt} - r_o$. The new description S' generated from R'_1 and C_{Opt} is

$$\begin{aligned} S' &= \bigcup_{r_i \in R'_1} r_i + \bigcup_{c_j \in C_{Opt}} c_j - \{d_{ij} | d_{ij} \in Cell(R'_1) \cup Cell(C_{Opt}), d_{ij}.color \neq blue\} \\ &\quad + \{d_{ij} | d_{ij} \notin Cell(R'_1) \cup Cell(C_{opt}), d_{ij}.color = blue\} \end{aligned}$$

The benefit of S' is

$$\begin{aligned} Benefit(S') &= \sum_{r_i \in R'_1} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) \\ &\quad + \sum_{d_{ij} \in Cell(R'_1) \cap Cell(C_{Opt}), d_{ij}.color=blue} (-1) \\ &= \sum_{r_i \in R'_1} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) - |R'_1| \times |C_{Opt}| \\ &= \sum_{r_i \in R_{Opt}} Benefit(r_i) - Benefit(r_o) + \sum_{c_j \in C_{Opt}} Benefit(c_j) \\ &\quad -(|R_{Opt}| - 1) \times |C_{Opt}| \\ &= Benefit(S_{Opt}) - Benefit(R_o) + |C_{Opt}| \end{aligned} \tag{4.3}$$

Because $Benefit(r_o) < 0$, $-Benefit(R_o) + |C_{Opt}| > 0$, therefore $Benefit(S') > Benefit(S_{Opt})$. This contradicts with the definition of the optimal description that S_{Opt} is the description with the maximum benefit. Therefore the assumption that S_{Opt} contains a row with negative benefit is not true. So S_{Opt} does not contain rows with negative benefits. For the same reason, S_{Opt} does not contains columns with negative benefits □

In a 2-dimensional 2-level hierarchical data cube, any two rows intersect with the same set of columns. Therefore we can deal with two rows with the same benefits similarly. From Equation 4.2 in the proof of Theorem 4.1.1, only the rows and columns with non-negative benefits can be selected into the optimal description. So we have the following lemma:

Lemma 4.1.1. If two rows, r_1, r_2, have the same non-negative benefits, then we can merge r_1, r_2 into a group. Both or none of them are selected into the optimal description, i.e. $r_1 \in R_{Opt} \iff r_2 \in R_{Opt}$.(We have the similar conclusion for columns.) □

Proof. 1. $r_1 \in R_{Opt} \Longrightarrow r_2 \in R_{Opt}$:

Let $R'_1 = R_{Opt} - r_1$. For the optimal description S_{Opt}, we have

$$\begin{aligned} Benefit(S_{Opt}) &= \sum_{r_i \in R_{Opt}} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) - |R_{Opt}| \times |C_{Opt}| \\ &= \sum_{r_i \in R'_1} Benefit(r_i) + Benefit(r_1) + \sum_{c_j \in C_{Opt}} Benefit(c_j) \\ &\quad -(|R'_1| + 1) \times |C_{Opt}| \\ &= \sum_{r_i \in R'_1} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) - |R'_1| \times |C_{Opt}| \\ &\quad +Benefit(r_1) - |C_{Opt}| \end{aligned} \tag{4.4}$$

Let us use S' to present the description generated from R'_1 and C_{Opt}, then

$$Benefit(S') = \sum_{r_i \in R'_1} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) - |R'_1| \times |C_{Opt}| \tag{4.5}$$

Applied Equation 4.5 to Equation 4.4, we have

$$Benefit(S_{Opt}) = Benefit(S') + Benefit(r_1) - |C_{Opt}|$$

Because S_{Opt} is the optimal description, $Benefit(S_{Opt}) \geq Benefit(S')$. Therefore we have

$$Benefit(r_1) - |C_{Opt}| \geq 0 \tag{4.6}$$

$Benefit(r_1) = Benefit(r_2)$, so $Benefit(r_2) - |C_{Opt}| \geq 0$.

Now let us suppose r_2 is not in the optimal description. Let $R''_1 = R_{Opt} + r_2$,

then the benefit of the description generated from R_1'' and C_{Opt} is

$$\begin{aligned} Benefit(S'') &= \sum_{r_i \in R_1''} Benefit(r_i) + \sum_{c_j \in C_{Opt}} Benefit(c_j) - |R_1''| \times |C_{Opt}| \\ &= \sum_{r_i \in R_{Opt}} Benefit(r_i) + Benefit(r_2) + \sum_{c_j \in C_{Opt}} Benefit(c_j) \\ &\quad -(|R_{Opt}| + 1) \times |C_{Opt}| \\ &= Benefit(S_{Opt}) + (Benefit(r_2) - |C_{Opt}|) \\ &\geq Benefit(S_{Opt}) \end{aligned} \tag{4.7}$$

We need to discuss two cases about Equation 4.7 here:

- $Benefit(S'') = Benefit(S_{Opt})$: S'' has the same benefit as S_{Opt}. It does not matter whether r_2 is in S_{Opt}, S_{Opt} remains the same benefit. So r_2 can also be included in the optimal description;
- $Benefit(S') > Benefit(S_{Opt})$: S' has more benefit than S_{Opt} , which is contradict with the definition of optimal description, therefore the assumption that r_2 is not in the optimal description is not true. So r_2 is in the optimal description.

2. $r_2 \in R_{Opt} \Longrightarrow r_1 \in R_{Opt}$ can be similarly proved.

□

From Lemma 4.1.1, in a 2-dimensional 2-level hierarchical data cube we can group rows with the same non-negative benefits into a group. We have the following definitions for a group.

Definition 4.1.1. For a group of rows, g, (without of loss of generality, we only consider groups with rows here), we have the following definitions:

1. Cardinality of g is defined as the number of contained rows:

$$|g| = |\{r_i | r_i \in g\}|$$

2. Benefit of g is defined as the benefit of a row in g:

$$Benefit(g) = Benefit(r_k | r_k \in g)$$

3. Total benefit of g is the sum of benefits of all rows in g:

$$TotalBenefit(g) = \sum_{r_i \in g} Benefit(r_i) = |g| \times Benefit(g)$$

□

After we group all rows into different groups based on the benefits of rows, we need to find a method to select groups into the optimal description. The following theorem shows a rule to select groups depended on the benefits of groups.

Theorem 4.1.2. Given two groups of rows, g_1, g_2, and $Benefit(g_1) > Benefit(g_2)$, if rows contained in g_2 are covered by S_{Opt} then rows contained in g_1 are covered by S_{Opt} too; if rows contained in g_1 are not covered by S_{Opt}, then neither the rows contained in g_2.

$$\begin{cases} g_2 \subseteq R_{Opt} \Longrightarrow g_1 \subseteq R_{Opt} \\ g_1 \not\subseteq R_{Opt} \Longrightarrow g_2 \not\subseteq R_{Opt} \end{cases}$$

□

Proof. Suppose $r_1 \in g_1$, $r_2 \in g_2$. From the Definition 4.1.1 we know $Benefit(r_1) > Benefit(r_2)$, so $Benefit(r_1) - |C_{Opt}| > Benefit(r_2) - |C_{Opt}|$.

- $g_2 \subseteq R_{Opt}$, so $r_2 \in R_{Opt}$. From Equation 4.6 we have $Benefit(r_2) - |C_{Opt}| > 0$, so $Benefit(r_1) - |C_{Opt}| > 0$ too. Based on Equation 4.7 r_1 is included in R_{Opt}. From Lemma 4.1.1, we know $g_1 \subseteq R_{Opt}$;

- $g_1 \not\subseteq R_{Opt}$, from Equation 4.6, we have $0 > Benefit(r_1) - |C_{Opt}| > Benefit(r_2) - |C_{Opt}|$. Therefore r_2 is not included in the optimal description, neither g_2.

□

4.1.2 Algorithm

Based on Lemma 4.1.1, Theorem 4.1.2 and Definition 4.1.1, we have the following algorithm to find optimal descriptions for a 2-dimensional 2-level hierarchical data cube:

Algorithm 4.1.1. Matrix Filling Algorithm

Input:a 2-dimensional 2-level data cube $H = R \times C$ without shared holes, and an interesting data set D.

1. Compute the benefit for each row/column;
2. $\forall r_i, r_j \in R$, if $Benefit(r_i) = Benefit(r_j) \geq 0$, then put r_i, r_j into one group (group columns in the same way);
3. Order row groups in descendant order of benefit. p is the number of row groups.

$$G_R = \{g_{r_i} | 1 \leq i \leq p, \forall 1 \leq k < l \leq p, Benefit(g_{r_k}) > Benefit(g_{r_l})\}$$

Column groups are ordered in the same way. q is the number of column groups.

$$G_C = \{g_{c_j} | 1 \leq j \leq q, \forall 1 \leq k < l \leq q, Benefit(g_{c_k}) > Benefit(g_{c_l})\}$$

4. Build a $(p+1) \times (q+1)$ matrix M as:

$$
\begin{aligned}
M_{ij} &= \sum(\text{total benefit of row groups from 1 to i}) \\
&\quad + \sum(\text{total benefit of column groups from 1 to j}) \\
&\quad - |\text{blue cells covered both by row groups and column groups}| \\
&= \sum_{1 \le k \le i} TotalBenefit(g_{r_k}) + \sum_{1 \le l \le j} TotalBenefit(g_{c_l}) \\
&\quad - \sum_{1 \le k \le i} |g_{r_k}| \times \sum_{1 \le l \le j} |g_{c_l}| \\
&= \begin{cases} 0 & i, j = 0 \\ M_{i-1,0} + TotalBenefit(g_{r_i}) & j = 0 \\ M_{0,j-1} + TotalBenefit(g_{c_j}) & i = 0 \\ M_{i-1,j} + M_{i,j-1} - M_{i-1,j-1} - |g_{r_i}| \times |g_{c_j}| & i \times j \neq 0 \end{cases}
\end{aligned} \tag{4.8}
$$

Output: $M_{i_{max}j_{max}} = max\{M_{ij}, 0 \le i \le p, 0 \le j \le q\}$ *is the benefit of the optimal description. The optimal description covers the rows in* $R_{Opt} = \{r_k | r_k \in g_{r_i}, 1 \le i \le i_{max}\}$, *columns in groups* $C_{Opt} = \{c_l | c_l \in g_{c_j}, 1 \le j \le j_{max}\}$.

$$
\begin{aligned}
S_{Opt} &= \bigcup_{r_i \in R_{Opt}} Benefit(r_i) + \bigcup_{c_j \in C_{Opt}} Benefit(c_j) \\
&\quad - \{d_{kl} | d_{kl} \in Cell(R_{Opt}) \cup Cell(C_{Opt}), d_{kl}.color \neq blue\} \\
&\quad + \{d_{kl} | d_{kl} \notin Cell(R_{Opt}) \cup Cell(C_{Opt}), d_{kl}.color = blue\}
\end{aligned}
$$

□

Example 4.1.1. The example in Figure 4.1 is a 2-dimensional 2-level hierarchical data cube without shared holes. All interesting cells are marked as ◯. We compute the benefits for rows and columns, and then order and group them by benefits:

group	rows	length	benefit	group	columns	length	benefit
g_{r_1}	(g,9),(h,9)	2	5	g_{c_1}	(i,1)	2	5
g_{r_2}	(f,9)	3	3	g_{c_2}	(i,2)	3	3
g_{r_3}	(e,9)	4	1	g_{c_3}	(i,3),(i,4)	4	1

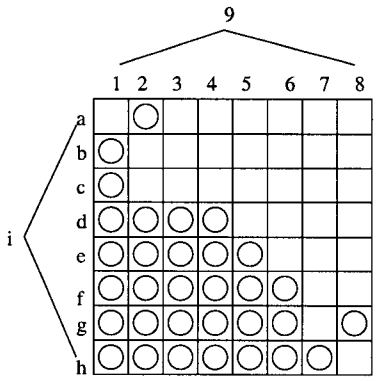


Figure 4.1: Data Region without Shared Holes

This example has 3 row groups and 3 column groups. We can fill a 4×4 matrix M to get the optimal description.

$$M = \begin{pmatrix} 0 & 5 & 8 & 10 \\ 10 & 13 & 14 & 12 \\ 13 & 15 & 15 & 11 \\ 14 & 15 & 14 & 8 \end{pmatrix}$$

- $M_{0,0} = 0$: means we do not select any row or column and only enumerate all blue cells. So $R_1 = C_1 = \emptyset$ and the benefit is 0;
- $M_{0,1} = 5$: means we select the columns in g_{c_1}, i.e. $(i, 1)$ and the benefit is 5;
- $M_{0,2} = 8$: means we select the columns in g_{c_1} and g_{c_2}, i.e. $\{(i, 1), (i, 2)\}$ and the benefit is $M_{0,1} + Benefit(i, 2) = 5 + 3 = 8$;

- $M_{0,3} = 10$: means we select the columns in g_{c_1}, g_{c_2} and g_{c_3}, i.e. $\{(i,1),(i,2),(i,3),(i,4)\}$ and the benefit is

  $$M_{0,2}+Benefit(i,3)+Benefit(i,4) = M_{0,2}+TotalBenefit(g_{c_3}) = 8+1\times 2 = 10$$

- $M_{2,1} = 15$: means we select the rows in g_{r_1}, g_{r_2} and the columns in g_{c_1}. $R_1 = \{(f,9),(g,9),(h,9)\}, C_1 = \{(i,1)\}$. Therefore the benefit of this selection is

  $$\sum_{r_i \in R_1} Benefit(r_i) + Benefit(i,1) - |R_1| \times |C_1| = 5+5+3+5-3 = 15$$

  At the same time we can see that

  $$M_{1,1} + M_{2,0} - M_{1,0} - |g_{r_2}| \times |g_{c_1}| = 13+13-10-1\times 1 = 15$$

  Therefore $M_{2,1} = M_{1,1} + M_{2,0} - M_{1,0} - |g_{r_2}| \times |g_{c_1}|$.

From the matrix M, we know that the maximum value is 15 and we can construct the optimal descriptions from M. For $M_{max} = M_{2,1} = 15$, all rows in groups g_{r_1}, g_{r_2} and columns in groups g_{c_1} are in the optimal description S_{Opt}:

$$\begin{aligned} S_{Opt} &= (h,9)+(g,9)+(f,9)+(i,1) \\ &\quad -\{(g,7)+(h,8)+(f,7)+(f,8)+(a,1)\} \\ &\quad +(a,2)+(d,2)+(d,3)+(d,4)+(e,2)+(e,3)+(e,4)+(e,5) \end{aligned}$$

The length of S_{Opt} is 17. The data set D has 32 data cells. So the benefit of this description is $32-17 = 15 = M_{2,1}$.

$M_{2,1} = M_{3,1} = M_{2,2} = 15$, so the optimal description is not unique. This example has three optimal descriptions and each of them has benefit of 15. □

Time Complexity:We can see that the run time of this algorithm is polynomial. Let $|R| = n_r, |C| = n_c$. To order and group all the rows and columns can be done in $O(n_r \lg n_r + n_c \lg n_c)$. And to fill the matrix can be done in $O(n_r \times n_c)$.

Theorem 4.1.3. Algorithm 4.1.1 generates an optimal description for data set D in the 2-dimensional 2-level data cube $H = R \times C$ without shared holes. □

Proof. Firstly, rows with the same benefits are grouped into a group based on Lemma 4.1.1. From Theorem 4.1.2 we know the groups with more benefit are selected before the groups with less benefit are selected. Therefore we order row groups in the descendant order of benefit. Columns are grouped and ordered in the similar way. Given a group of rows, g_{r_i}, the groups of rows that have more benefits than g_{r_i} are $\{g_{r_k}|1 \leq k < i\}$. If g_{r_i} is selected then all groups in $\{g_{r_k}|1 \leq k \leq i\}$ should be selected too. For the same reason, if a group of columns, g_{c_j}, is selected, all groups in $\{g_{c_l}|1 \leq l \leq j\}$ should be selected too. Let $R_1 = \{r_p|r_p \in g_{r_k}, 1 \leq k \leq i\}$ and $C_1 = \{c_q|c_q \in g_{c_l}, 1 \leq l \leq j\}$, the description generated from R_1 and C_1 has benefit as

$$\begin{aligned} & \sum_{r_p \in R_1} Benefit(r_p) + \sum_{c_q \in C_1} Benefit(c_q) - |R_1| \times |C_1| \\ = & \sum_{1 \leq k \leq i} TotalBenefit(g_{r_k}) + \sum_{1 \leq l \leq j} TotalBenefit(g_{c_l}) \\ & - \sum_{1 \leq k \leq i} |g_{r_k}| \times \sum_{1 \leq l \leq j} |g_{c_l}| \end{aligned}$$

In matrix M, we use M_{ij} to save the benefit of the description when we select the rows in groups $\{g_{r_k}|1 \leq k \leq i\}$ and the columns in groups $\{g_{c_l}|1 \leq l \leq j\}$. Therefore, only if we find the maximum value amongst all elements in M we can generate a description with the maximum benefit for D.

Now we show that Equation 4.8 in Algorithm 4.1.1 generate the correct benefits. Algorithm 4.1.1 has four equations to fill matrix M:

- i,j=0. This means we do not select any row or column and only enumerate the

blue cells in D, so we cannot get any benefit from this description. Therefore $M_{ij} = 0$;

- j=0. This means we only select rows in groups $g_{r_k}, 1 \leq k \leq i$. So the benefit of this description is $\sum_{1 \leq k \leq i} TotalBenefit(g_{r_k})$. We also have $M_{i-1,0} = \sum_{1 \leq k \leq i-1} TotalBenefit(g_{r_k})$, so

$$M_{ij} = M_{i0} = \sum_{1 \leq k \leq i} TotalBenefit(g_k) = M_{i-1,0} + TotalBenefit(g_{r_i})$$

- i=0. This case can be similarly proved as above case when $j = 0$;

- $i \times j \neq 0$. We select the rows in groups $g_{r_k}, 1 \leq k \leq i$ and the columns in groups $g_{c_l}, 1 \leq l \leq j$, and the benefit is

$$\begin{aligned} M_{ij} &= \sum_{1 \leq k \leq i} TotalBenefit(g_{r_k}) + \sum_{1 \leq l \leq j} TotalBenefit(g_{c_l}) \\ &\quad - \sum_{1 \leq k \leq i} |g_{r_k}| \times \sum_{1 \leq l \leq j} |g_{c_l}| \end{aligned}$$

At the same time, we have

$$\begin{aligned} M_{i-1,j} &= \sum_{1 \leq k \leq i-1} TotalBenefit(g_{r_k}) + \sum_{1 \leq l \leq j} TotalBenefit(g_{c_l}) \\ &\quad - \sum_{1 \leq k \leq i-1} |g_{r_k}| \times \sum_{1 \leq l \leq j} |g_{c_l}| \\ M_{i,j-1} &= \sum_{1 \leq k \leq i} TotalBenefit(g_{r_k}) + \sum_{1 \leq l \leq j-1} TotalBenefit(g_{c_l}) \\ &\quad - \sum_{1 \leq k \leq i} |g_{r_k}| \times \sum_{1 \leq l \leq j-1} |g_{c_l}| \\ M_{i-1,j-1} &= \sum_{1 \leq k \leq i-1} TotalBenefit(g_{r_k}) + \sum_{1 \leq l \leq j-1} TotalBenefit(g_{c_l}) \\ &\quad - \sum_{1 \leq k \leq i-1} |g_{r_k}| \times \sum_{1 \leq l \leq j-1} |g_{c_l}| \end{aligned}$$

We can get

$$\begin{aligned} M_{i-1,j} + M_{i,j-1} - M_{i-1,j-1} &= \sum_{1 \le k \le i} TotalBenefit(g_{r_k}) \\ &\quad + \sum_{1 \le l \le j} TotalBenefit(g_{c_l}) \\ &\quad - \sum_{1 \le k \le i} |g_{r_k}| \times \sum_{1 \le l \le j} |g_{c_l}| + |g_{r_i}| \times |g_{c_j}| \end{aligned}$$

So we have

$$M_{ij} = M_{i-1,j} + M_{i,j-1} - M_{i-1,j-1} - |g_{r_i}| \times |g_{c_j}|$$

Therefore, after we find $M_{i_{max}j_{max}} = max\{M_{ij}, 0 \le i \le p, 0 \le j \le q\}$ then let $R_{Opt} = \{r_k | r_k \in g_{r_i}, 1 \le i \le i_{max}\}$ and let $C_{Opt} = \{c_l | c_l \in g_{c_j}, 1 \le j \le j_{max}\}$, so we can construct a description with benefit $M_{i_{max}j_{max}}$ as

$$\begin{aligned} S_{Opt} &= \bigcup_{r_i \in R_{Opt}} Benefit(r_i) + \bigcup_{c_j \in C_{Opt}} Benefit(c_j) \\ &\quad - \{d_{kl} | d_{kl} \in Cell(R_{Opt}) \cup Cell(C_{Opt}), d_{kl}.color \neq blue\} \\ &\quad + \{d_{kl} | d_{kl} \notin Cell(R_{Opt}) \cup Cell(C_{Opt}), d_{kl}.color = blue\} \end{aligned}$$

S is an optimal description for D in H. □

4.1.3 Multi-Level Hierarchy Also Hard

From Algorithm 4.1.1 We know that the MDLE problem can be done in polynomial time in a 2-dimensional 2-level hierarchical data cube without shared holes. But how about a multilevel or multidimensional hierarchical data cube? Now let us analyze an example first.

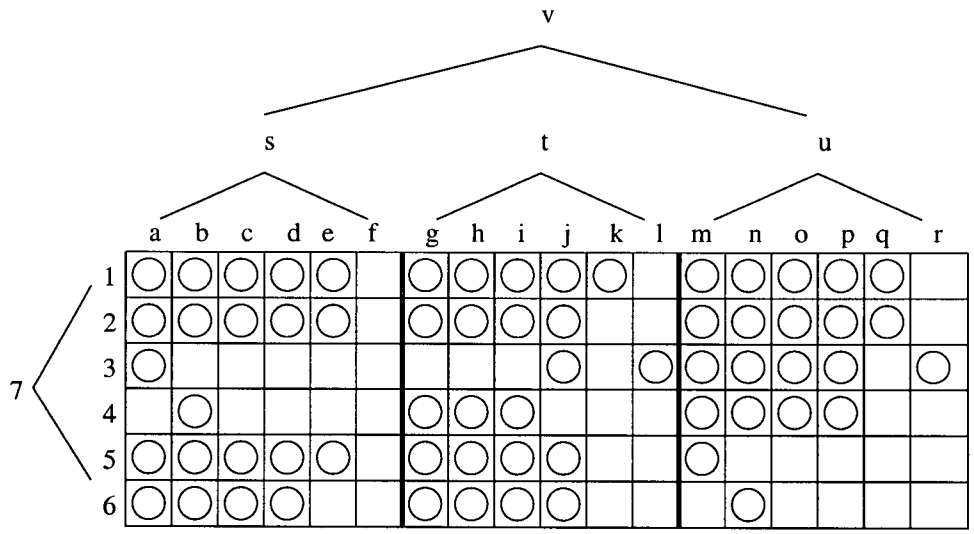


Figure 4.2: Multi-Level Hierarchical Data Cube

Example 4.1.2. For the example in Figure 4.2, the optimal description is

$$
\begin{aligned}
& (1,v) - \{(1,f) + (1,l) + (1,r)\} \\
+ \;& (2,s) + (5,2) + (6,s) + (3,a) + (4,b) - \{(2,f) + (5,f) + (6,e) + (6,f)\} \\
+ \;& (7,g) + (7,h) + (7,i) + (7,j) + (3,l) - \{(3,g) + (3,h) + (3,i) + (4,j)\} \\
+ \;& (2,u) + (3,u) + (4,u) + (5,m) + (6,n) - \{(2,r) + (3,q) + (4,q) + (4,r)\}
\end{aligned}
$$

In the proof of Lemma 4.1.1 and Theorem 4.1.2, an important property we used is that all rows intersect with the same columns, i.e. all rows in R_{Opt} intersect with all columns in C_{Opt}. But in this example, row $(1,v)$ and row $(2,s)$ intersect with different sets of columns. At the same time, row $(6,s)$ and row $(2,t)$ have the same benefit, column $(7,g)$ and column $(7,a)$ have the same benefit, but $(6,s)$ and $(2,t)$ are not in one group, and neither $(7,g)$ and $(7,a)$. Because the optimal description only contains $(6,s),(7,g)$. □

The consequence of losing this property is that we cannot order and group

rows and columns by benefit any more. Then Lemma 4.1.1 and Theorem 4.1.2 are not true any more. Therefore Algorithm 4.1.1 does not work either. So in a multilevel or multidimensional hierarchical data cube, MDLE remains NP-Hard.

4.2 Heuristics

4.2.1 Basic Definitions

We already have definitions for the 1-dimensional hierarchical data cube and the 2-dimensional hierarchical data cube. In this section we present some definitions for a more general case: the multi-dimensional multi-level hierarchical data cube.

Definition 4.2.1. A multi-dimensional hierarchical data cube is in the form of $H =< H_1, H_2, \ldots, H_n >$ where $H_i, 1 \leq i \leq n$, is an l_i level hierarchy defined in Definition 3.1.1 □

Definition 4.2.2. In data cube $H =< H_1, H_2, \ldots, H_n >$,

- Data cell is defined as $(x_1, x_2, \ldots, x_n)$ if $\forall x_i, x_i \in Leaves(H_i)$;
- Data region is defined as $(x_1, x_2, \ldots, x_n)$ if $\exists x_i, x_i \notin Leaves(H_i)$;

□

Definition 4.2.3. A parent region is a region that only has a non-leaf node in one dimension and this non-leaf node is the parent of leaves. It can be defined as:

$$\{(x_1, x_2, \ldots, x_n) | \exists i, \forall x' \in Cld(x_i), x' \in Leaves(H_i), \forall 1 \leq j \leq n, j \neq i, x_j \in Leaves(H_j)\}$$

□

4.2.2 A Greedy Heuristic

Greedy method is a very common heuristic for NP-Hard problems. From the previous chapter we know that MDLE tries to minimize the length of the description to describe a given data set and maximize the benefit of the description. So the main goal of MDLE is to decrease the regions and cells used to cover the interested data set. The benefit of a description is the number of saved regions and cells if we use this description to cover the interesting data set. We can order regions in the descendant order of the benefit. Each time the region with the most benefit is selected till all the left over regions have negative benefits.

After the selections of regions we mark all the non-blue cells in these regions as blue, then we apply MDL-Tree algorithm in [11] on the data cube. MDL-Tree algorithm generates the unique optimal MDL description for blue cells in the data cube. This MDL description with the exceptions of the non-blue cells in the selected regions is the MDLE description generated by our greedy heuristic.

Algorithm 4.2.1. Greedy Algorithm

Input: a multi-dimensional data cube H and an interesting data set D

Output: a description S for D

1. Initialize $S = D, E = S' = E' = \emptyset$; initialize an empty hash table Q;

   /S and S' are sets of blue cells; E and E' are sets of holes. $S - E$ is a description for D. Each time we select a region x, S' contains the blues in S but not in x. E' contains holes in E and x. $S' - E'$ is the new description by selecting x.*/*

2. Compute the benefit of each parent region;

Insert regions into hash table Q: for each region, the hash key is computed from its benefit and the number of holes in it. */*A region with more benefit always has greater hash key; for two regions with the same benefits, the region containing less holes has the greater hash key.*/*

3. Select the region, x, with the greatest hash key in Q, and let

$$S' = S - \{x' | x' \in x, x'.color = blue\}$$

$$E' = E + \{x'' \in x, x''.color \neq blue\}$$

4. If $|S'| + |E'| < |S| + |E|$ then

   Let $S = S', E = E'$;

   End If

5. Delete x from Q;

6. If Q is not empty then go to Step.3;

7. For each cell in E, find the corresponding cell in H and mark it as blue. Apply MDL-Tree algorithm on new blue cells in H and the description generated by MDL-Tree is stored in S;

8. $S - E$ is the MDLE description for D in H.

□

Complexity: Based on Definition 3.1.1, n_{l_i-1} is the number of nodes in level $l_i - 1$ of hierarchy H_i and n_{l_j} is the number of nodes in level l_j of hierarchy H_j. The number of parent regions in a data cube H is

$$\sum_{1 \leq i \leq n} \{n_{l_i-1} \times \prod_{1 \leq j \leq n, j \neq i} n_{l_j}\}$$

For any hierarchy H_i each node in level l_i has a parent in level $l_i - 1$, so we have $n_{l_i-1} \leq n_{l_i}$. Let us use $N = \prod_{1 \leq i \leq n} n_{l_i}$ to present the number of data cells in data cube H.

$$\begin{aligned} & \sum_{1 \leq i \leq n} \{n_{l_i-1} \times \prod_{1 \leq j \leq n, j \neq i} n_{l_j}\} \\ \leq & \sum_{1 \leq i \leq n} \{n_{l_i} \times \prod_{1 \leq j \leq n, j \neq i} n_{l_j}\} \\ = & \sum_{1 \leq i \leq n} \prod_{1 \leq j \leq n} n_{l_j} \\ = & \ n \times N \end{aligned}$$

We use a hash table to store the regions and the hash key is computed by the benefit of this region. The region with more benefit has greater hash key, and the hash table size is less than $n \times N$.

Therefore the time complexity of greedy algorithm is $O(n \times N)$ in the worst case. □

We notice that we restrict the candidate regions to parents of leaves. We can also extend the candidates to grandparents, great grandparents or higher level regions. But we need more work to order them. Because a grandparent region has more benefit than one of its children does not mean that we really should choose the grandparent before we choose its children. In order to compensate, we apply MDL-Tree algorithm on the data cube to merge the parents regions to upper level after we finish the selection.

Example 4.2.1. For the example in Figure 4.3, first we compute the benefits for parent regions. Then we select regions in the descending order of benefit.

region	length	benefit	holes	region	length	benefit	holes
(e,6)	1	3	-	(d,10)	2	2	(d,5)
(e,1)	2	1	(a,1)	(e,2)	2	1	(b,2)
(e,3)	2	1	(b,3)	(a,11)	2	1	(a,8)
(e,8)	2	1	(a,8)	(c,10)	3	0	(c,4),(c,5)

$(e,6)$ has the most benefit in all regions so we select it first. The second region is $(d,10)$. Now the current S is:

$$\begin{aligned} S &= (e,6)+(d,10)-(d,5) \\ &\quad +(a,2)+(a,3)+(b,1)+(b,5)+(c,1)+(c,2)+(c,3) \\ &\quad +(a,7)+(a,9)+(b,8)+(c,8)+(d,8) \end{aligned}$$

The length of S' is 15, $|S'| = 15$. Now the next region is $(e,1)$. If we select $(e,1)$ then we can delete $(b,1)$ and $(c,1)$ from S but we need to add $(e,1)$ and hole $(a,1)$, so the length of the description does not decrease. Therefore we cannot select $(e,1)$, neither $(e,2),(e,3)$. After the selection of $(a,11),(e,8)$ the algorithm stops, and the output of the algorithm is

$$\begin{aligned} S &= (d,10)-(d,5)+(a,2)+(a,3)+(b,1)+(b,5)+(c,1)+(c,2)+(c,3) \\ &\quad +(e,6)+(a,11)+(e,8)-(a,8) \end{aligned}$$

The length of S is 13 and the total benefit is 7. □

4.2.3 Dynamic Programming

Dynamic programming is also a widely used method to approximately solve NP-Complete problems. The core part of dynamic programming is that a problem can be decomposed to several sub-problems and answers to those sub-problems lead to

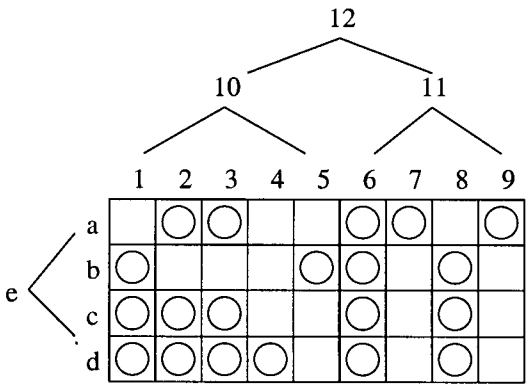


Figure 4.3: Example for Heuristics

an solution to this problem[17]. The multi-dimensional hierarchical data cube has such a property that we can use dynamic programming as a good heuristic method.

Given a data cube $H =< H_1, H_2, \ldots, H_n >$ and an interesting data set D, for any data region $x = (x_1, x_2, \ldots, x_n)$, if x_i is not a leaf, i.e. $x_i \notin Leaves(H_i)$, we can generate a description for x from the descriptions of the children regions of x_i, that is

$$S(x) = \bigcup_{x_{ik} \in Cld(x_i)} S(x_{ik}) \tag{4.9}$$

and the length of $S(x)$ can be computed by

$$|S(x)| = |\bigcup_{x_{ik} \in Cld(x_i)} S(x_{ik})| = \sum_{x_{ik} \in Cld(x_i)} |S(x_{ik})|$$

$S(x)$ is a description generated from children regions and we call $S(x)$ as a local solution for x.

For data region x we can also use the following description to cover the blue cells in x,

$$S = x - \{x' | x' \in Cell(x), x'.color \neq blue\} \tag{4.10}$$

The length of S is

$$|S| = 1 + |\{x'|x' \in Cell(x'', x'.color = blue\}|$$

S is to cover all blue cells in x by x with the exceptions of all non-blue cells in x. We call S the global solution for x.

For each region x, we find all local solutions and the global solution, then we use the solution with the minimum length as the description for x. This procedure is showed in the following algorithm.

Algorithm 4.2.2. Dynamic Programming

Input: $H =< H_1, H_2, \ldots, H_n >$, *and an interesting data set D;*

1. Build an n dimensional matrix $M = |H_1| \times |H_2| \times \ldots \times |H_n|$, $|H_i|$ is the number of nodes in hierarchy H_i;

   Build an n dimensional matrix P in the same way.

2. Label the nodes in each hierarchy H_i by post-order: the first leaf is labelled as '1' and the root is labelled as $|H_i|$. We use the label of a node as the index of this node in the corresponding dimension of a matrix.

   /*For example, for region $x = (x_1, x_2, \ldots, x_n)$, the value of x in M is the element $M(index(x_1), index(x_2), \ldots, index(x_n))$. In order to present the values in M more clearly, we only use $M(x_1, x_2, \ldots, x_n)$ to present the value in M for x. */

3. For any data cell $x = (x_1, x_2, \ldots, x_n)$, if x is a blue cell, then set $M(x_1, x_2, \ldots, x_n) = 1$, otherwise $M(x_1, x_2, \ldots, x_n) = 0$;

4. For any data region $x = (x_1, x_2, \ldots, x_n)$, we get the value of $M(x_1, x_2, \ldots, x_n)$ by the following methods:

(a) for any $1 \leq i \leq n$, if $x_i \notin Leaves(H_i)$, compute the length of the local solution(Equation 4.9) on dimension i:

$$\sum_{x_{i_k} \in Cld(x_i)} M(x_1, x_2, .., x_{i_k}, \ldots, x_n) \tag{4.11}$$

(b) compute the length of the global solution(Equation 4.10) of x:

$$1 + |\{x' | x' \in Cell(x), x'.color \neq blue\}| \tag{4.12}$$

$M(x_1, x_2, \ldots, x_n)$ is the minimum value of Equation 4.11 and Equation 4.12.

5. We get the value of $P(x_1, x_2, \ldots, x_n)$ based on the value of $M(x_1, x_2, \ldots, x_n)$:

$$P(x_1, x_2, \ldots, x_n) = \begin{cases} i & if\ M(x_1, x_2, \ldots, x_n) = Equation\ 4.11 \\ g & if\ M(x_1, x_2, \ldots, x_n) = Equation\ 4.12 \end{cases}$$

□

Complexity: We can see the two matrixes both have $|H_1| \times |H_2| \times \ldots \times |H_n|$ elements. So the total complexity of the dynamic programming is $O(|H_1| \times |H_2| \times \ldots \times |H_n|)$. □

The matrix M stores the length for each region $(x_1, x_2, \ldots x_n)$, and the matrix P stores the way to construct the description for the region. After we compute the length of the description for D, we need to construct the description based on matrix P.

Algorithm 4.2.3. Description:

Input: H, D, P, $x = (x_1, x_2, \ldots, x_n)$

Output: a description for region $x = (x_1, x_2, \ldots, x_n)$

1. if $P(x_1, x_2, \ldots, x_n) =' g'$, the description is

$$S = x - \{x' | x' \in x, x'.color \neq blue\}$$

2. if $P(x_1, x_2, \ldots, x_n) =' i', 1 \leq i \leq n$, the description is

$$S = \bigcup_{x_{i_k} \in Cld(x_i)} Description(x_1, x_2, \ldots, x_{i_k}, \ldots, x_n)$$

□

Complexity: This algorithm just splits the region to get the optimal descriptions. The complexity is $O(max|H_i|)$. □

Example 4.2.2. We still use the example in Figure 4.3. In the data cube $H = (H_1, H_2)$, let us assume $H_1 = \{a, b, c, d, e\}$ and $H_2 = \{1, 2, 3, 4, 5, 10, 6, 7, 8, 9, 11, 12\}$. We build two 5×12 matrixes M and P and we generate the elements by the method showed in Algorithm 4.2.2. The matrixes M and P are presented in the following tables.

M (index)		1	2	3	4	5	6	7	8	9	10	11	12
(index)	(node)	1	2	3	4	5	10	6	7	8	9	11	12
1	a	0	1	1	0	0	2	1	1	0	1	2	4
2	b	1	0	0	0	1	2	1	0	1	0	2	4
3	c	1	1	1	0	0	3	1	0	1	0	2	5
4	d	1	1	1	1	0	2	1	0	1	0	2	4
5	e	2	2	2	1	1	8	1	1	2	1	5	13

P (index)		1	2	3	4	5	6	7	8	9	10	11	12
(index)	(node)	1	2	3	4	5	10	6	7	8	9	11	12
1	a						2					g	2
2	b						2					2	2
3	c						2					2	2
4	d						g					2	2
5	e	g	g	g	2	2	2	g	1	g	1	2	2

We label nodes in each hierarchy by post-order. The label for each node is the index of this node in the corresponding dimension in matrixes M and P. For example, the index for node 10 is 6 and the index for node e is 5. M(5,6) is the length for region (e,10).

- $(d, 10)$. d is a leaf node and 10 is a non-leaf node. We compute the local solution for node 10:

$$\sum_{i \in Cld(10)} M(e, i) = 4$$

The global solution is

$$1 + |\{x | x \in Cell(d, 10), x.color \neq blue\}| = 1 + 1 = 2$$

Therefore $M(5, 6) = 2$ and $P(5, 6) =' g'$.

- $e, 10)$. Both of e and 10 are non-leaf nodes. So we need to compute two local solutions.

$$\sum_{i \in Cld(e)} M(i, 10) = M(a, 10) + M(b, 10) + M(c, 10) + M(d, 10) = 9$$

$$\sum_{j \in Cld(10)} M(e, j) = M(e, 1) + M(e, 2) + M(e, 3) + M(e, 4) + M(e, 5) = 8$$

The global solution is

$$1 + |\{x | x \in Cell(e, 10), x.color \neq blue\}| = 1 + 9 = 10$$

Therefore $M(e, 10) = 8$. 10 is a node in H_2, so $P(e, 10) =' 2'$.

- $(e, 12)$. Both of e and 12 are non-leaf nodes. So we need to compute two local solutions.

$$\sum_{i \in Cld(e)} M(i, 12) = M(a, 12) + M(b, 12) + M(c, 12) + M(d, 12) = 17$$

$$\sum_{j \in Cld(12)} M(e, j) = M(e, 10) + M(e, 11) = 13$$

The global solution is

$$1 + |\{x | x \in Cell(e, 12).x.color \neq blue\}| = 1 + 16 = 17$$

Therefore $M(e, 12) = 13$ and $P(e, 12) =' 2'$.

After we create the matrix P, we can create the description for D.

- $P(e,10) =' 2'$, $S(e,10)$ is the union of $S(e,1), S(e,2), S(e,3), S(e,4)$ and $S(e,5)$.

  $P(e,1) = P(e,2) = P(e,3) =' g'$, $S(e,1) = (e,1) - (a,1)$, $S(e,2) = (e,2) - (b,2)$, $S(e,3) = (e,3) - (b,3)$;

  $P(e,4) = P(e,5) =' 1'$, $S(e,4) = (d,4)$, $S(e,5) = (b,5)$;

  Therefore $S(e,10) = (e,1) - (a,1) + (e,2) - (b,2) + (e,3) - (b,3) + (d,4) + (b,5)$

- $P(e,11) = 2$, $S(e,11)$ is the union of $S(e,6), S(e,7), S(e,8)$ and $S(e,9)$

  $P(e,6) = P(e,8) =' g'$, $S(e,6) = (e,6)$, $S(e,8) = (e,8) - (a,8)$.

  $P(e,7) = P(e,9) =' 1'$, $S(e,7) = (a,7)$, $S(e,9) = (a,9)$.

  Therefore $S(e,11) = (e,6) + (e,8) - (a,8) + (a,7) + (a,9)$

- $P(e,12) =' 2'$, $S(e,12)$ is the union of $S(e,10)$ and $S(e,11)$, that is

$$\begin{aligned} S(e,12) &= (e,1) - (a,1) + (e,2) - (b,2) + (e,3) - (b,3) + (d,4) + (b,5) \\ &\quad +(e,6) + (e,8) - (a,8) + (a,7) + (a,9) \end{aligned}$$

□

From Example 4.2.1 and Example 4.2.2 we know the greedy heuristic and the dynamic programming heuristic generated different descriptions for the data cube in Figure 4.3. The optimal description for this data cube is:

$$\begin{aligned} S_{Opt} &= (e,1) - (a,1) + (e,2) - (b,2) + (e,3) - (b,3) + (d,4) + (b,5) \\ &\quad +(e,6) + (a,11) + (e,8) - (a,8) \end{aligned}$$

The length of the optimal description is 12 and the benefit is 8.

Let us compare the descriptions in Example 4.2.1 and Example 4.2.2 with the optimal description.

- $(e, 10)$: the optimal description for this region is $(e, 1) - (a, 1) + (e, 2) - (b, 2) + (e, 3) - (b, 3) + (d, 4) + (b, 5)$. The dynamic programming heuristic generated this description.

  In the greedy heuristic, the region selected in each loop, which has the most benefit amongst the parent regions, is the best choice for the current step, but maybe it is not the best choice for the whole cube. In this example, the greedy heuristic selected $(d, 10)$, which has benefit 2, into the description. But if we select $(e, 1), (e, 2)$ and $(e, 3)$, which have the same benefits of 1, then the total benefit is 3. Therefore the greedy heuristic may create an unoptimal description.

- $(e, 11)$: the optimal description for this region is $(e, 6) + (e, 8) - (a, 8) + (a, 7) + (a, 9)$. The greedy heuristic generated this description.

  The dynamic programming heuristic compares the local solution(Equation 4.9) with the global solution(Equation 4.10) and selects the description with the minimum length. For $(e, 11)$ the optimal description is the combination of a row $(a, 11)$ and columns $(e, 6), (e, 8)$, which is not contained in Equation 4.9 and Equation 4.10. This is the reason why the dynamic programming heuristic generates a description that has longer length than the optimal description sometimes.

4.2.4 Quadratic Programming

Preliminaries

Quadratic Function is in the form of

$$f(x) = \frac{1}{2} x^T H x + cx$$

x is a vector of variables, H is a matrix.

Given a function defined in a constrained space, this function is a convex function iff

$$f(ax + (1 - ay)) \leq af(x) + (1 - a)f(y), 0 \leq a \leq 1, x, y \in \text{constrained space}$$

If $f(x)$ is a convex quadratic function, then the quadratic programming is to minimize the value of $f(x)$ subject to linear equality and inequality constraints[9]. This kind of problem is called the convex quadratic programming and it is in the form of

$$\begin{aligned} Minimize \quad & f(x) = \tfrac{1}{2}x^T Hx + cx \\ s.t. \quad & Ax \geq b \\ & Bx = e \\ & x \geq lb \\ & x \leq ub \end{aligned}$$

In order to make sure that the quadratic function f has a minimal value in the constrained space, the function f should be a convex function. A matrix H is positive semi-definite iff for all $x \in R^n$, $x^T Hx \geq 0$. It has been proved that a quadratic function $f(x)$ is convex *iff* H is positive semi-definite[2].

Quadratic Programming for MDLE

In order to use quadratic programming to solve MDLE problem, we need to build the quadratic function and the linear constraints from the data cube H and the interesting data set D. Firstly let us study the 2-dimensional 2-level hierarchical

data cube defined in Definition 3.2.4. $H = R \times C$ and D is an interesting data set. A description S for D in the form of

$$\begin{aligned} S &= \bigcup_{r_i \in R_1} r_i + \bigcup_{c_j \in C_1} c_j - \{d_{ij} | d_{ij} \in Cell(R_1) \cup Cell(C_1), d_{ij}.color \neq blue\} \\ &\quad + \{d_{ij} | d_{ij} \in Cell(R_1) \cup Cell(C_1), d_{ij}.color = blue\} \end{aligned}$$

The benefit of S is

$$\begin{aligned} Benefit(S) &= \sum_{r_i \in R_1} Benefit(r_i) + \sum_{c_j \in C_1} Benefit(c_j) \\ &\quad + \sum_{d_{ij} \in Cell(R_1) \cap Cell(C_1), d_{ij}.color \neq blue} (1) \\ &\quad + \sum_{d_{ij} \in Cell(R_1) \cap Cell(C_1), d_{ij}.color = blue} (-1) \end{aligned}$$

Let us use X_i to present whether $r_i \in R_1$ for any $r_i \in R$: if $r_i \in R_1$ then $X_i = 1$, otherwise $X_i = 0$. Similarly we use Y_j to present whether $c_j \in C_1$ for any $c_j \in C$. Therefore we have the following equations: $(n_r = |R|, n_c = |C|)$

- the sum of benefits of rows in R_1 can be expressed as

$$\begin{aligned} \sum_{r_i \in R_1} Benefit(r_i)) &= \sum_{r_i \in R_1} 1 \times Benefit(r_i) + \sum_{r_i \notin R_1} 0 \times Benefit(r_i) \\ &= \sum_{1 \leq i \leq n_r} X_i \times Benefit(r_i) \end{aligned}$$

- for those blue cells in $Cell(R_1) \cup Cell(C_1)$, i.e. $d_{ij} = (r_i, c_j), r_i \in R_1, c_j \in C_1$, we have $X_i = 1, Y_j = 1$, therefore

$$\sum_{d_{ij} \in Cell(R_1) \cap Cell(C_1), d_{ij}.color = blue} (-1) = \sum_{r_i \in R_1, c_j \in C_1, d_{ij} = (r_i, c_j), d_{ij}.color = blue} (-1) \times X_i \times Y_j$$

For those blue cells not in $Cell(R_1) \cup Cell(C_1)$, $X_i \times Y_j = 0$, therefore

$$\sum_{d_{ij} \notin Cell(R_1) \cap Cell(C_1), d_{ij}.color = blue} (-1) \times X_i \times Y_j = 0$$

Combine these two equations, we can get

$$\sum_{d_{ij} \in Cell(R_1) \cap Cell(C_1), d_{ij}.color=blue} (-1) = (-1) \times \sum_{d_{ij}.color=blue} X_i \times Y_j$$

Similarly we can have the following two equations:

$$\begin{aligned} \sum_{c_j \in C_1} Benefit(c_j) &= \sum_{1 \le j \le n_c} X_i \times Benefit(c_j) \\ \sum_{d_{ij} \in Cell(R_1) \cap Cell(C_1), d_{ij}.color \neq blue} (1) &= 1 \times \sum_{d_{ij}.color \neq blue} X_i \times Y_j \end{aligned}$$

The benefit of S can be recomputed as

$$\begin{aligned} Benefit(S) = & \sum_{1 \le i \le n_r} X_i \times Benefit(r_i) + \sum_{1 \le j \le n_c} Y_j \times Benefit(c_j) \\ & +1 \times \sum_{d_{ij}.color \neq blue} X_i \times Y_j + (-1) \times \sum_{d_{ij}.color=blue} X_i \times Y_j \end{aligned} \quad (4.13)$$

The benefit function in Equation 4.13 is a quadratic function. X_i, Y_j are the variables. The constraints are $0 \le X_i \le 1, 0 \le Y_j \le 1$. The goal of MDLE is to maximize Equation 4.13, which is to minimize $f = -Beneifit(S)$. Therefore we can build the quadratic programming model as

$$\begin{aligned} Minimize \quad & f = -Benefit(S) \\ s.t. \quad & 0 \le X_i \le 1 \\ & 0 \le Y_j \le 1 \end{aligned}$$

Example 4.2.3. Let us look the example in Figure 4.4, the benefits of rows and columns are showed in the following table:

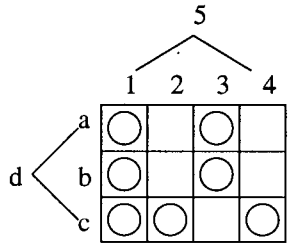


Figure 4.4: Example for Quadratic Programming

row	X	length	benefit	column	Y	length	benefit
(a,5)	X_1	3	-1	(d,1)	Y_1	1	2
(b,5)	X_2	3	-1	(d,2)	Y_2	3	-2
(c,5)	X_3	2	1	(d,3)	Y_3	2	0
				(d,4)	Y_4	3	-2

Let us use X_1, X_2, X_3 to represent the rows, Y_1, Y_2, Y_3 and Y_4 to represent the columns as showed in the table. We can get a description with the most benefit when we solve the following quadratic programming problem:

$$\begin{aligned} \textit{Minimize } f \;=\; & -(\; -X_1 - X_2 + X_3 \\ & +2Y_1 - 2Y_2 - 2Y_4 \\ & -X_1Y_1 + X_1Y_2 - X_1Y_3 + X_1Y_4 \\ & -X_2Y_1 + X_2Y_2 - X_2Y_3 + X_2Y_4 \\ & -X_3Y_1 - X_3Y_2 + X_3Y_3 - X_3Y_4 \;) \end{aligned}$$

$$\begin{aligned} s.t. \quad & 0 \le X_i \le 1 \\ & 0 \le Y_j \le 1 \end{aligned}$$

Transform f into the matrix form $f = -(\frac{1}{2}x^T Hx + cx)$:

x is a 7×1 matrix:

$$x = \begin{pmatrix} X_1 & X_2 & X_3 & Y_1 & Y_2 & Y_3 & Y_4 \end{pmatrix}^T$$

c is a 1×7 matrix:

$$c = \begin{pmatrix} -1 & -1 & 1 & 2 & -2 & 0 & -2 \end{pmatrix}$$

H is a 7×7 matrix:

$$H = \begin{pmatrix} 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

For this example, when $X_1 = 0, X_2 = 0, X_3 = 1, Y_1 = 1, Y_2 = 0, Y_3 = 1, Y_4 = 0$, the function f has the maximal value of 3 and the corresponding description is

$$S = (c, 5) + (e, 1) + (e, 3) - (c, 3)$$

□

Heuristic Algorithm

From Example 4.2.3 we know that we can use quadratic programming as a heuristic for MDLE problem. If the quadratic function is a convex function on the constrained space, we can reach a global optimality and find the minimum value for this function. Otherwise, the global optimality and the minimum value cannot be guaranteed.

In Example 4.2.3, $f = -(\frac{1}{2}x^T Hx + cx)$ is not a convex function and H is not a positive semi-definite matrix. So even quadratic programming is a good model for the 2-dimensional 2-level hierarchy, the global optimality cannot be reached for all cases. But we can use quadratic programming as a heuristic method for MDLE problem.

On the other hand, in Equation 4.13 we know that $X_i \times Y_j$ represents cell $d_{ij} = (r_i, c_j)$. A data cell in a 3-dimensional hierarchical data cube is in the form of $X_i \times Y_j \times Z_k$ and the benefit of a description is not a quadratic function anymore. Therefore in the algorithm, we only apply quadratic programming on the data region $x = (x_1, x_2, \ldots, x_n)$ such that x only contains non-leaf nodes in two dimensions. We can express the benefit of a description for x in the quadratic function as Equation 4.13.

Algorithm 4.2.4. Quadratic Programming:

Input: $H = \{H_1, H_2, \ldots, H_n\}$, *and an interesting data set D;*

1. Build an n dimensional matrix $M = |H_1| \times |H_2| \times \ldots \times |H_n|$, $|H_i|$ is the number of nodes in hierarchy H_i;

   Build an n dimensional matrix P in the same way.

2. Label the nodes in each hierarchy H_i by post-order: the first leaf is labelled as '1' and the root is labelled as $|H_i|$. We use the label of a node as the index of this node in the corresponding dimension of a matrix.

   /*For example, for region $x = (x_1, x_2, \ldots, x_n)$, the value of x in M is the element $M(index(x_1), index(x_2), \ldots, index(x_n))$. In order to present the values in M more clear, we only use $M(x_1, x_2, \ldots, x_n)$ to present the value in M for x. */

3. For those regions$(x_1, x_2, ..x_i, \ldots x_j, .., x_n)$ that contains two or more non-leaf nodes, we get the value of $M(x_1, x_2, \ldots, x_n)$ by the following methods:

   (a) for any $1 \leq i \leq n$, if $x_i \notin Leaves(H_i)$, compute the length of the local solution(Equation 4.9) on dimension i:

   $$\sum_{x_{i_k} \in Cld(x_i)} M(x_1, x_2, .., x_{i_k}, \ldots, x_n) \tag{4.14}$$

   (b) compute the length of the global solution(Equation 4.10) of x:

   $$1 + |\{x' | x' \in Cell(x), x'.color \neq blue\}| \tag{4.15}$$

   (c) if x contains two non-leaf nodes, then we need to call function Quadratic(x).

   $M(x_1, x_2, \ldots, x_n)$ is the minimum value of Quadratic(x), Equation 4.14 and Equation 4.15.

4. We get the value of $P(x_1, x_2, \ldots, x_n)$ based on the value of $M(x_1, x_2, \ldots, x_n)$:

$$P(x_1, x_2, \ldots, x_n) = \begin{cases} q & if \ \ M(x_1, x_2, \ldots, x_n) = Quadratic(x) \\ i & if \ \ M(x_1, x_2, \ldots, x_n) = Equation\ 4.14 \\ g & if \ \ M(x_1, x_2, \ldots, x_n) = Equation\ 4.15 \end{cases}$$

□

Algorithm 4.2.4 calls a function $Quadratic(x = (x_1, x_2, \ldots x_n))$. This is the function to do the optimization for quadratic functions.

Function 4.2.1. $Quadratic(x = (x_1, x_2, \ldots, x_n))$

Input: a data region x with two non-leaf nodes

*/*Suppose the two non-leaf nodes are x_k and x_l. */*

1. Build a $|Leaves(x_k)| \times |Leaves(x_l)|$ matrix M based on the data region $x = (x_1, x_2, \ldots, x_n)$, if a data cell is blue, then set the correspondent element in M to 1, otherwise 0;

2. Based on this matrix, compute all the needed arguments for Matlab function *quadprog*;

3. Call Matlab function *quadprog* to get the optimal solution;

4. Return the optimal result.

□

The Matlab function *quadprog* reaches a global optimality if the quadratic function is convex[15]. The function generated from the data region is not a convex quadratic function, therefore *quadprog* can only find one local solution in the polynomial time.

4.3 Experimental Results

In order to evaluate the three heuristic methods to summarize the hierarchical data with exceptions, we choose the hierarchy generator used in paper[19] to create hierarchical tree structure, which is the mimic structure used in data warehousing benchmarks.

4.3.1 MDL vs MDL with Exceptions

Figure 4.5 shows the experiment results of the three heuristics and MDL-Tree algorithm[11] are presented. The test data set is a 2-dimensional hierarchical data cube and in each dimension the fanout of the tree hierarchical structure is $1-4-25$,

which means the hierarchy has one root node that has 4 children and each child of the root has 25 children which are leaf nodes. So the whole data size is 10^4. The $x-axis$ is the number of cells randomly chosen to be blue; the $y-axis$ is the length of the description to represent those blue cells.

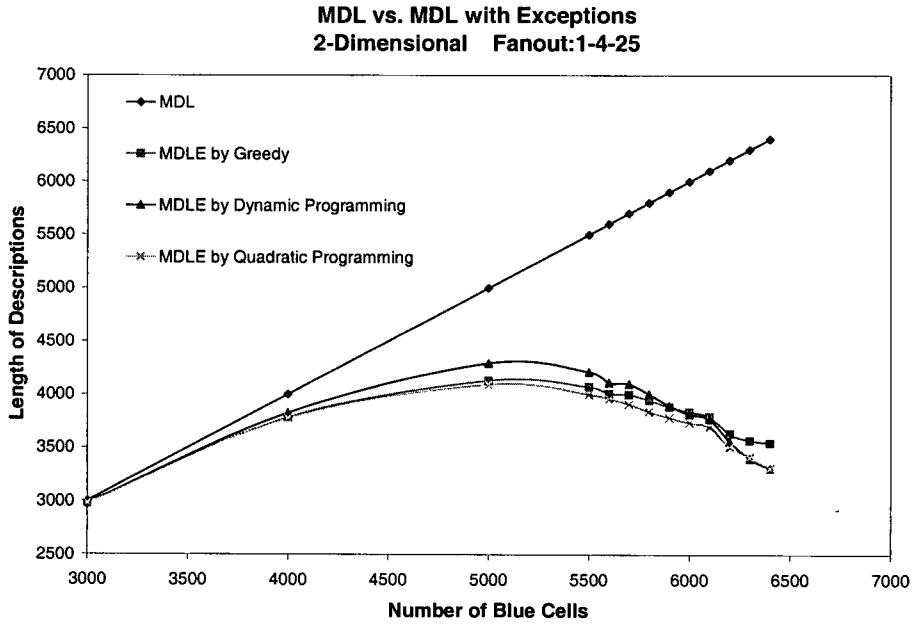


Figure 4.5: MDL vs. MDL with exceptions

From this figure, we can see that MDL with exceptions generates shorter descriptions than MDL algorithm. The smallest region in this structure contains 25 nodes (each parent of leaf-nodes has 25 children), so MDL algorithm need all these 25 cells as blue cells to summarize them to one region. This is the reason that MDL cannot get any gain when the blue cells are less than 6400. The three

heuristics all have better results than MDL. Around $5000 - 5500$ it is the peak area where the heuristics generate the longest descriptions. When the number of blue cells is less than 5000, the length of description is increasing with the number of blue cells. When the number of blue cells is greater than 5500, the length of description is decreasing. And the distance between MDL and heuristics w.r.t one x value is increasing with the number of blue, that is the more blue cells the more gain we can get.

Amongst these three heuristics, the quadratic programming always has better results than others. When blue cells is less than 6000, greedy is a little better than dynamic programming. When the number of blue is greater than 6000 the dynamic programming beats greedy algorithm. The reason is quadratic programming checks the combinations between rows and columns but the dynamic programming only checks the sum of rows or column without the combination. Greedy picks the region with the most benefit each time but loses the benefit from the tree structure which is considered well by quadratic programming and dynamic programming. Therefore when the blue ratio is high greedy is the worst one.

Running times of Heuristics

The running time of each heuristic is shown in the following table.

Heuristic	Running Time(secs)
Greedy	4
Dynamic Programming	5
Quadratic Programming	80

From the table of running time we know the greedy heuristic is the fastest one amongst these three heuristics.At the same time, the descriptions generated

by the greedy heuristic are just a little longer than the descriptions generated by the other two heuristics. In the following experiments we only compare the greedy heuristics with the GMDL-Tree algorithm.

4.3.2 GMDL vs MDL with Exceptions

Figure 4.6 is the comparison between GMDL[11] and MDL with exceptions. In GMDL-Tree algorithm white ratio means how many white cells are permitted to be covered. In this figure we only compare the GMDL with the greedy method for MDL with exceptions. From the figure we can see the greedy method generates longer descriptions at the beginning but it beats the GMDL with the increasing number of blue cells. When there are 6400 blue cells the greedy method works better than GMDL in H containing 15% white cells but still not good as GMDL in H containing 20% while cells.

From the definition of GMDL and MDL with exceptions we can see that GMDL description is not pure for blue cells because it contains lots of white cells. MDL with exceptions covers some rows and columns containing non-blue cells. But those non-blue cells are excluded in the exception part of the description. So the MDL with exceptions generates pure descriptions to express blue cells. On the other hand, as showed in Figure 4.7, when H contains some red cells the GMDL works much worse than MDL with exceptions.

Figure 4.7 is to compare MDLE with GMDL in H containing some red cells. In this experiment we set the red cells are 5% of total cells, that is there are 500 red cells overall. When the blue cells are more than 6000, MDL with exceptions works better than GMDL even if H contains 40% white cells, which is really high for real applications. If there are some red cells MDL with exception is better than GMDL.

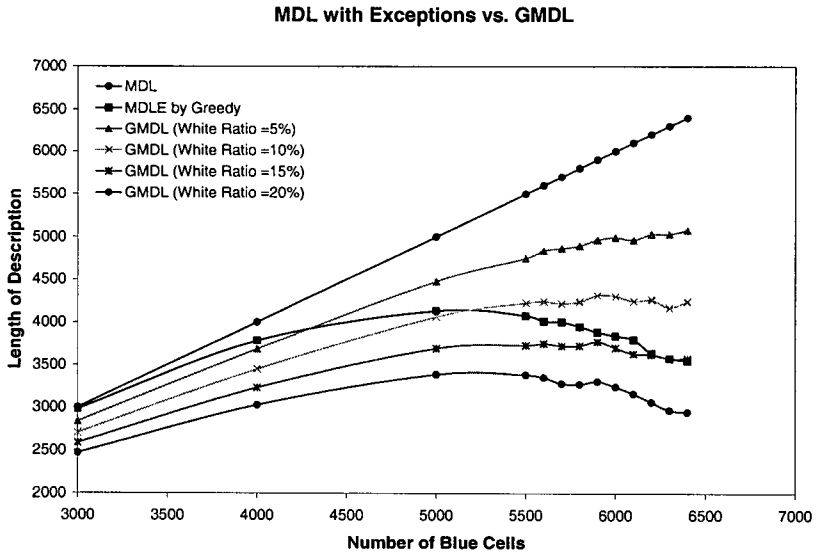


Figure 4.6: GMDL vs.MDL with exceptions

MDL with exceptions generates pure descriptions with much shorter length.

4.3.3 Comparison in 3-Dimensional Case

So far we only show the experiments in 2-dimensional hierarchical data cube. Now let us discuss the experiments in 3-dimensional hierarchical data cube. Figure 4.8 shows the results for 3-dimensional hierarchical data cube.

The tree hierarchical structure is $1-4-4-6$ which means this is a 4 level tree. The hierarchy has one root node which has 4 children, and each of those children has 4 children too. So the root has 16 grand-nodes. Each of those grand-nodes has 6 children which are leaf-nodes. So for each dimension there are 96 leaves and the data space is around 10^6.

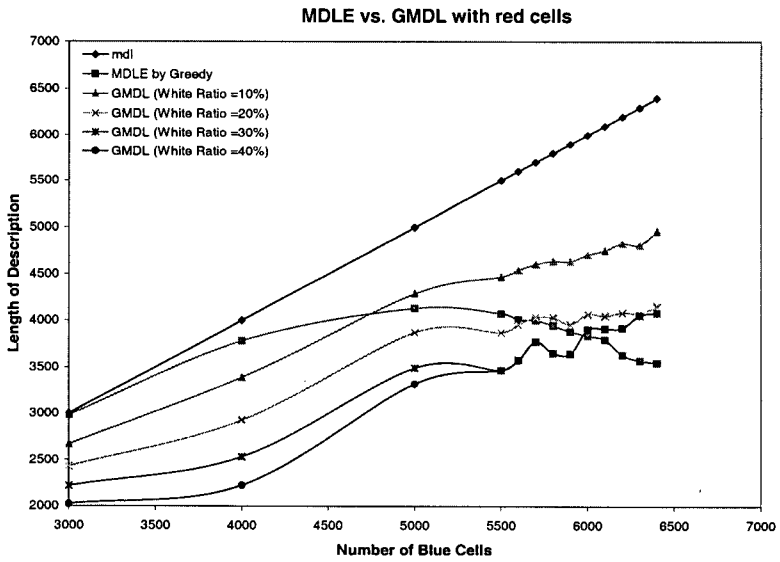


Figure 4.7: GMDL with red cells

From the figure we can see, MDL gets some benefits too because the smallest region only contains 6 nodes which makes it easy for MDL to summarize one region. The results in 3-dimensional hierarchical data cube is coincident with the 2-dimensional hierarchical data cube. Finally, quadratic programming also generates the shortest descriptions amongst the heuristics. Quadratic programming and dynamic programming are better than GMDL with white ratio 10%.

Comparison of Running Times

The running times of MDL-Tree, GMDL-Tree algorithm and the heuristics for MDLE are shown in the following table.

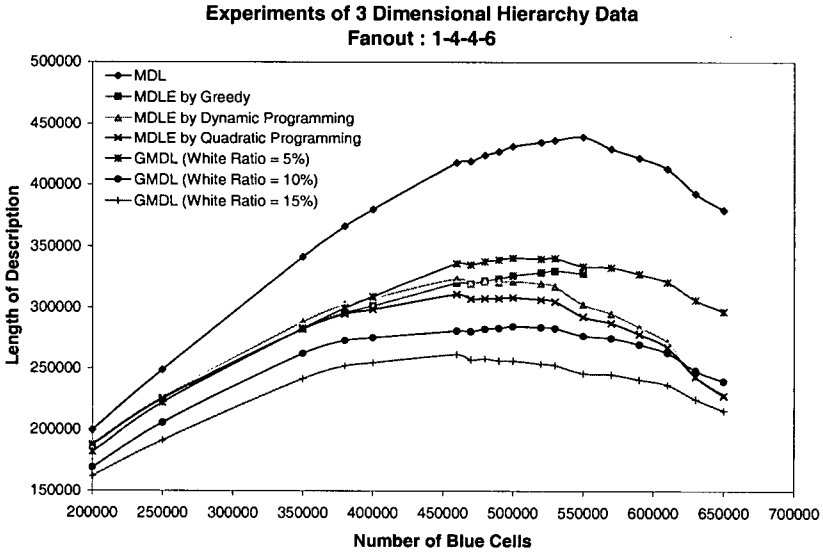


Figure 4.8: Experiments of 3-Dimensional Hierarchy

Algorithm	Running Time(secs)	
	2-dimensional fanout:1-4-25	3-dimensional fanout:1-4-4-6
MDL-Tree	4	12
GMDL-Tree	4	35
Greedy Heuristic	4	40
Dynamic Programming	5	70
Quadratic Programming	80	30000

From the table of running times we know that the greedy heuristic shortens the descriptions without taking much more time than GMDL-Tree algorithm. Dynamic programming is a little slower but it is still fast enough. Quadratic pro-

gramming is the slowest one. The reason is that we need call Matlab function 'quadprog' to find an optimal value for the quadratic function, which is the most important part of quadratic programming heuristic.

4.3.4 Hole Ratio

The last figure shows the ratio of holes in the descriptions of MDL with exceptions. Figure 4.9 represents the results. We can see from the figure that with the increasing number of blue cells, the description includes more and more holes.

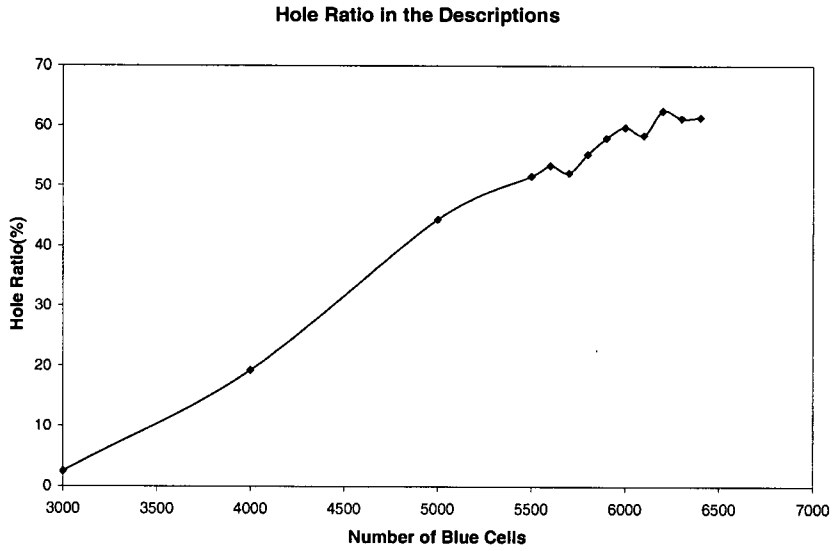


Figure 4.9: Hole Ratio

Chapter 5

Conclusions and Future Work

5.1 Summary and Conclusion

This thesis has studied the problem of data summarization with exceptions. MDL-Tree algorithm can find the concise description for a pure hierarchical data set in polynomial time. But for some cases MDL-Tree can not gain much benefit and the description is still too long. GMDL-Tree algorithm can generate much shorter descriptions but it contains lots of white cells which affects the description's impurity. In this thesis we define a new method that could generate a pure description which is much shorter than MDL and even GMDL in many cases. We name this new method as MDL with exceptions(MDLE), which means the description generated by this approach is in the form of the union of a set of blue cells and a set of data regions with the exception of non-blue cells in these regions. We call these included non-blue cells holes.

In this thesis we proved MDL with exceptions is an NP-Hard problem. We built a complete weighted bipartite graph that only has weights on edges(CEW bipartite graph). First we proved it is an NP-Hard problem to find a subset of

vertices which induces a subgraph that maximizes the addition of the weight of this subgraph and the cardinality of this vertices subset. Then we showed a reduction from CEW problem to MDL with exceptions.

Known the general MDL with exceptions is NP-Hard, we also studied some tractable cases. One of these tractable cases is the 1-dimensional hierarchical data cube. For this case we proposed a bottom-up algorithm to build the optimal description. The other case is a 2-dimensional 2-level hierarchical data cube without shared holes. We proposed a matrix-filling algorithm for this tractable case and also explain why MDLE remains NP-Hard when it goes to a multidimensional or multilevel hierarchical data cube.

We also proposed three heuristics for MDL with exceptions and compared them with MDL-Tree algorithm and GMDL-Tree algorithm. Amongst those three heuristics, quadratic programming has the best quality and always generates the shortest descriptions. Greedy method is the fastest one which generates just a little longer description. When the blue ratio is low, dynamic programming is not as good as greedy but if the blue ratio is high dynamic programming generates much shorter descriptions than greedy. The descriptions generated by those heuristics are pure, which means the description only covers blued cells. But with the increasing number of the blue cells, the description includes more and more holes, which may cause confusing to users.

5.2 Future Work

5.2.1 Summarization of Holes

From Figure 4.9 we already knew that when the blue cells are more than 6000, the hole ratio is higher than 60%. This means that if the length of the description is 3000 then there is more than 1800 holes included in the description. If the blue ratio is very high, then the descriptions contain too many holes. If we can do some summarization on those holes then we can get much shorter descriptions.

For a simple example in Figure 5.1, the description generated by MDL with exceptions is:

$$(5,a)+(5,b)+(5,c)+(5,d)+(1,g)-\{(1,a)+(1,b)+(1,c)+(1,d)\}$$

The length is 9. In this description there are 4 holes. If we can do some summarization on holes, we can get another expression as

$$(5,a)+(5,b)+(5,c)+(5,d)+(1,e)+(1,f)-\{(1,g)-(1,e)-(1,f)\}$$

$$=(5,a)+(5,b)+(5,c)+(5,d)-(1,g)+(1,e)+(1,f)$$

The length of this description is 7.

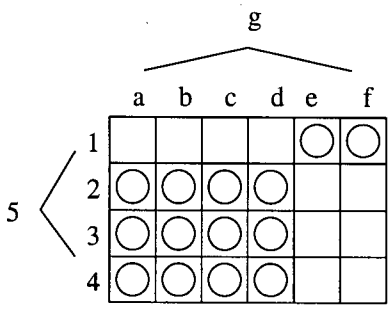


Figure 5.1: Example of Summarization on Holes

If we study this data region closer we can find another shorter description, which is

$$(5, g) - (1, g) - (5, e) - (5, f) + (1, e) + (1, f)$$

The length of this description is only 6.

From this example we can see that we can generate shorter descriptions and get more benefits by doing some summarizations on holes. Surely this problem is also a hard problem but we can try to propose some heuristics for it.

5.2.2 Representation of Holes

The descriptions generated by MDLE contain holes as exceptions. We define the length of each hole as 1, which is the same length as a blue cell or a region. But from the user's point of view it is different. For example, given two descriptions S_1 and S_2, S_1 contains 20 blue cells/regions and S_2 contains 10 blue cells/regions and 5 holes. The length of S_1 is 20. The length of S_2 is 15. Even S_2 has shorter length than S_1, maybe users still prefer to use S_1. Why does this happen? Because it is much more direct, easy and normal to read a description with only blue cells/regions. If the descriptions contain some holes, the user might need to know where these holes come from and need to know more clearly about the tree hierarchical structure to understand these descriptions. So it is natural to assign different length to a hole and a blue cell. We can assign a hole as length 1.5 or 2. This is to add some weights on cells. The other method is that we can limit the hole ratio in a description for easy understanding.

5.2.3 Approximation Rate

So far we have proposed three heuristics, Greedy, Dynamic Programming and Quadratic Programming. But we did not give any approximation rate of these methods. Further work should be done to find a good approximation algorithms with suitable approximation rate. Because the graph model(CEW) we used has some $2-$ *approximation* algorithms, it is a good start to find some approximation algorithms from these methods.

Bibliography

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *ACM SIGMOD*, pages 94–105, May 1998.

[2] John C.G.Boot. *Quadratic Programming: Algorithms, Anomalies, Applications.* Noth-Holland Publishing Company, 1964.

[3] S. CHAUDHURI and U. DAYAL. An overview of data warehousing and olap technology. *ACM SIGMOD Record*, 26(1):65–74, 1997.

[4] Joseph C. Culberson and Robert A. Reckhow. Covering polygons is hard. *Journal of Algorithms*, 17:2–44, 1994.

[5] G. Gambosi V. Kann A. Marchetti Spaccamela G. Ausiello, P. Crescenzi and M. Protasi. *Complexity and Approximations: Combinatorial Optimization Problems and Their Approximability Properties.* Springer, 1999.

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

[7] Dorit S. Hochbaum. Approximating clique and biclique problems. *Journal of Algorithms*, 29-1:174–220, 1998.

[8] J.Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[9] K.G.Murty. *Linear Complementarity, Linear and Nonlinear Programming.* Heldermann Verlag, 1988.

[10] V. S. Anil Kumar and H. Ramesh. Covering rectilinear polygons with axis-parallel rectangles. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing (STOC-99)*, pages 445–454, May 1999.

[11] Laks V.S. Lakshmanan, Raymond T. Ng, Christine Xing Wang, and Xiaodong. The generalized MDL approach for summarization. *Proceedings of the 28th VLDB Conference*, pages 766–777, 2002.

[12] P. Keskinocak M. Dawande and S. Tayur. On the biclique problem in bipartite graphs. In *GSIA working paper 1996-04*. Carnegie-Mellon University, 1997.

[13] Alberto O. Mendelzon and Ken Q. Pu. Concise descriptions of subsets of structured sets. In *ACM PODS*, pages 123–133, June 2003.

[14] Raymond T. Ng and Christine Xing Wang. Summarization using the generalized minimum description length. In *Technical Report*. Computer Science Department, University of British Columbia, 2000.

[15] Optimization Toolbox: quadprog. http://www.mathworks.com/access/helpdesk/help/toolbox/optim/quadprog.html.

[16] Michael R.Garey and David S.Johnson. *Computers and Intractablity: A Guide to the theory of NP-Completeness*. W.H.Freeman, 1979.

[17] Charles E. Leiserson Thomas H. Cormen and Ronald L.Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[18] W.H.Inmon. *Building the Data Warehouse*. QED Technical Publishing Group, 1992.

[19] XiaoDong Zhou. Generalized mdl approach for data summarization. *University of British Columbia, Canada*, 2002.