ON THE ORDERING OF MULTIATTRIBUTE DATA IN INFORMATION RETRIEVAL SYSTEMS

by

XIAN LIU

B.A.Sc, Xi'an Jiaotong University, 1982
M.A.Sc, Xi'an Jiaotong University, 1987
M.Math, University of Waterloo, 1992

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY in

THE FACULTY OF GRADUATE STUDIES

Department of Electrical Engineering

We accept this thesis as conforming to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

March, 1996

©Xian Liu, 1996
In presenting this thesis in partial fulfilment 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.

Department of Electrical Engineering
The University of British Columbia
Vancouver, Canada

Date April 20, 1996
Abstract

The requirement of ordering a set of multiattribute data arises frequently from spatial index processing and secondary key retrieval in modern information systems. The first application involves developing a single numerical index on a one-dimensional line for each point in multi-dimensional space, such that the spatial localizability can be preserved as best as possible. This work can be carried out by a mathematical transformation called spatial ordering. The Hilbert order has attained intensive interest in the literature due to its desirable performance. The lack of inexpensive encoding and decoding algorithms has been mentioned frequently in publications.

The second application, secondary key retrieval, involves two issues: determining the resolution for each dimension in a multi-dimensional hashed space and ordering data blocks on disks. Aho and Ullman proposed a constrained nonlinear programming model applied to the partial match query. However, the literature is relatively silent on the range query retrieval. Faloutsos showed that the order determined by the reflected Gray code can significantly improve the cluster distribution of the data blocks. It is recommended to design symmetric Gray codes to reduce the bias introduced by the reflected Gray code.
In this thesis, efficient analytical formulas are first derived to encode and decode the two-dimensional Hilbert order. Then the algorithms for the three-dimensional Hilbert order are discussed.

The encoding and decoding processes for n-dimensional Hilbert orders \((n \geq 4)\) would be considerably complicated and place a heavy computational burden on the application problems. Thus a new spatial order called the U order is proposed. Its major advantage is the significant simplicity of the encoding and decoding operations for multi-dimensional data. Performance assessments suggest that the U order is comparable with the Hilbert order and superior to most other orders. The performance analysis is followed by a formal mathematical description for the n-dimensional U order. Then a set of analytical encoding and decoding formulas are presented.

Besides the U order, several new spatial orders with the quadrant-recursive structure are also investigated. The goal is to find the candidates that are competitive with the Hilbert order in performance and with relative simple encoding and/or decoding formulas. Of these new orders, the Q4 order behaves best and its performance is very close to the Hilbert order. An analysis shows that its encoding and decoding algorithms only need 64.5% and 84.2% of operations of the corresponding algorithms of the Hilbert order.

The optimization of resolutions of multiattribute file systems is then discussed. Optimization models for the range query retrievals are presented by incorporating stochastic programming methodology. Three approaches are discussed. The here-and-now and the wait-and-see approaches seem to be applicable to the situation in
which range queries follow a relative simple distribution. Four typical paradigms are analyzed. One of them leads to a result that is similar to the one used in industry to evaluate the performance of disk drives. The scenario tracking approach appears to be more flexible and more appropriate for the problems where the probability distribution is not available. It also provides an adaptive technique to formulate non-conventional stochastic programming problems. Then a stochastic programming software package is developed and implemented. Numerical experiments show the coordinating effect of the scenario tracking methodology.

Finally, the symmetric Hamiltonian paths in a high-dimensional hypercube are discussed. A heuristic approach is applied to construct the symmetric Hamiltonian paths in the five-, six-, and seven-dimensional hypercubes. They are equivalent to the symmetrical Gray codes.

In summary, this thesis proposes three major approaches applied to the ordering of multiattribute data: a set of analytical encoding and decoding formulas for the two-dimensional Hilbert order, two new spatial orders with significant simple encoding and decoding formulas, and a set of optimization models for the range query problem.
Contents

Abstract ii

Table of Contents v

List of Tables x

List of Figures xii

Acknowledgement xviii

1 Introduction 1

2 The Hilbert Order 6

2.1 Overview 6

2.2 Metric Analysis of the 2D Hilbert Order 10

2.2.1 Range Query 15

2.3 Encoding and Decoding the 2D Hilbert Order 22

2.3.1 Encoding 23

2.3.2 Decoding 30
2.3.3 Implementation and Execution-Time Assessment .......... 33

2.4 Encoding and Decoding the 3D Hilbert Order ............... 37

2.4.1 Encoding ........................................ 38

2.4.2 Decoding ........................................ 49

2.5 Alternative Patterns of the Hilbert Order .................. 50

2.5.1 Configuration of Pattern L1 .......................... 53

2.5.2 Configuration of Pattern L2 .......................... 54

2.5.3 Configuration of Pattern L3 .......................... 58

2.5.4 Configuration of Pattern L4 .......................... 60

2.6 Summary ............................................. 62

3 The U Order ............................................ 64

3.1 Overview ........................................... 64

3.2 Metric Analysis of the 2D U Order ........................ 65

3.2.1 Partial Match Query ................................. 65

3.2.2 Range Query ...................................... 70

3.3 Neighbour Finding .................................... 73

3.4 Neighbour Proximity .................................. 80

3.5 Encoding and Decoding ................................ 84

3.6 The Analytic Formulas for the 2D U Order .................. 87

3.7 Extension to Three Dimensions ............................ 90

3.8 The Analytic Formulas for the 3D U Order .................. 94

3.9 The n-dimensional U Order ............................... 96
3.9.1 Definition .............................................. 97
3.9.2 Encoding and Decoding ................................. 101

3.10 Kronecker Algebraic Structures of 2D Spatial Orders ..... 103
3.10.1 The Z Order ............................................. 103
3.10.2 The U Order ............................................. 105
3.10.3 The X Order ............................................. 106

3.11 Discussion on Conventional Bit-Manipulation Techniques .... 109
3.12 Summary .................................................... 116

4 The Q Orders ................................................ 117
4.1 Overview ................................................... 117
4.2 The Q1 Order ............................................... 118
  4.2.1 Configuration ........................................... 118
  4.2.2 Metric Analysis of Range Query ...................... 122
4.3 The Q2 Order ............................................... 125
  4.3.1 Configuration ........................................... 125
  4.3.2 Metric Analysis of Range Query ...................... 126
4.4 The Q3 Order ............................................... 132
  4.4.1 Configuration ........................................... 132
  4.4.2 Metric Analysis of Range Query ...................... 135
4.5 The Q4 Order ............................................... 139
  4.5.1 Configuration ........................................... 139
  4.5.2 Metric Analysis of Partial Match Query .............. 142
4.5.3 Metric Analysis of Range Query ................. 143
4.6 The Q5, Q6, and Q7 Orders ...................... 147
4.7 Neighbour Finding and Proximity ................. 151
4.8 Encoding and Decoding the Q4 Order .............. 152
    4.8.1 Encoding .................................. 153
    4.8.2 Decoding .................................. 158
4.9 Summary ......................................... 159

5 Optimization of the Key Resolution for Range Query Retrieval 162
    5.1 Overview ..................................... 163
    5.2 Introduction to Stochastic Programming .......... 165
    5.3 The Here-and-Now Stochastic Programming Model . 168
    5.4 The Wait-and-See Stochastic Programming Model . 175
    5.5 The Scenario Tracking Stochastic Programming Model 178
    5.6 Optimization of the Section Capacity ............ 179
    5.7 Generalization to Uncertainty Models ............ 182
    5.8 Summary ..................................... 182

6 Optimization on the Bucket Distribution 184
    6.1 Overview ..................................... 184
    6.2 Gray Coordinate Lattice and Symmetric $H_n$ .... 189
    6.3 Summary ..................................... 195

7 Conclusion ......................................... 197
Appendix A
Implementation of Stochastic Programming Software and Numerical Experiments
List of Tables

2.1 Expected Number of Clusters for Resolution $r$ .................................. 12
2.2 Coordinates and $H$-Values at Level 1 .................................................. 40
2.3 Tables for Encoding the 3D Hilbert Order ............................................ 47

3.1 Expectation for Partial Match Query .................................................... 70
3.2 Expectation for $2 \times 2$ Range Query .................................................. 73
3.3 Neiboughood of the 8-connected Template .......................................... 77
3.4 Average Maximum Distance of Finding 8-Neighbors in a 2D Grid ............ 77
3.5 Average Total Distance of Finding 8-Neighbors in a 2D Grid ............... 78
3.6 Average Maximum Distance of Finding 26-Neighbors in a 3D Grid ........ 80
3.7 Average Total Distance of Finding 26-Neighbors in a 3D Grid ............. 81
3.8 Average Farthest Distance of the Neighbors in a 2D Grid ................. 82
3.9 Average Farthest Distance of the Neighbors in a 3D Grid ............... 84
3.10 Coordinates and $u$-values at Level 1 ................................................. 92

4.1 Expectation for Partial Match Query .................................................. 144
4.2 Behavior of the Average Case ............................................................. 148
4.3 Average Maximum Distance of Finding 8-Neighbors in a 2D Grid . . . . 151
4.4 Average Total Distance of Finding 8-Neighbors in a 2D Grid . . . . 152
4.5 Average Farthest Distance of the Neighbors in a 2D Grid . . . . . 153
4.6 Computational Costs for Resolution $r$ . . . . . . . . . . . . . . . 160

5.1 Coefficients $a_{ki}$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 174

6.1 Number of Hamiltonian Cycles in Hypercubes . . . . . . . . . . . 189
.1 Input Data Set 1: Frequencies of Fields to be Specified . . . . . . 212
.2 Input Data Set 2: Weights of Range Queries . . . . . . . . . . . . 214
.3 Optimal Values of the Single Scenario Model . . . . . . . . . . . . 214
List of Figures

2.1 The B Order of Resolution 2 ........................................ 7
2.2 The D Order of Resolution 2 ........................................ 7
2.3 The G Order of Resolution 2 ........................................ 7
2.4 The Hilbert Order of Resolution 2 ................................... 8
2.5 The K Order of Resolution 2 ........................................ 8
2.6 The R Order of Resolution 2 ........................................ 8
2.7 The RP Order of Resolution 2 ....................................... 9
2.8 The S Order of Resolution 2 ........................................ 9
2.9 The Z Order of Resolution 2 ........................................ 9
2.10 The Hilbert Order of Resolution 1 ................................. 12
2.11 The Hilbert Order of Resolution 2 ................................. 12
2.12 The Hilbert Order of Resolution 3 ................................. 13
2.13 Path of the Hilbert Order of Resolution 1 ....................... 16
2.14 Path of the Hilbert Order of Resolution 2 ....................... 16
2.15 Path of the Hilbert Order of Resolution 3 ....................... 17
2.16 Path of the Hilbert Order of Resolution 4 ....................... 18
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.17 Structure and Orientation of Level 1</td>
<td>25</td>
</tr>
<tr>
<td>2.18 Structure and Orientation of Level 2</td>
<td>25</td>
</tr>
<tr>
<td>2.19 Two Instances with Directional Vectors at Level 2</td>
<td>26</td>
</tr>
<tr>
<td>2.20 Indexed Directions</td>
<td>27</td>
</tr>
<tr>
<td>2.21 The Hilbert Order of Resolution 4</td>
<td>31</td>
</tr>
<tr>
<td>2.22 Comparison of Encoding Algorithms for the Hilbert Order</td>
<td>36</td>
</tr>
<tr>
<td>2.23 Comparison of Decoding Algorithms for the Hilbert Order</td>
<td>36</td>
</tr>
<tr>
<td>2.24 Structure and Orientation of Level 1</td>
<td>39</td>
</tr>
<tr>
<td>2.25 Structure and Orientation of Level 2</td>
<td>40</td>
</tr>
<tr>
<td>2.26 Two Instances with Directional Vectors at Level 2</td>
<td>43</td>
</tr>
<tr>
<td>2.27 Rotation from $U_2$ to $U_4$</td>
<td>44</td>
</tr>
<tr>
<td>2.28 Rotation from $U_7$ to $U_f$</td>
<td>44</td>
</tr>
<tr>
<td>2.29 Indexed Directions</td>
<td>48</td>
</tr>
<tr>
<td>2.30 Moore's Pattern of Resolution 2</td>
<td>51</td>
</tr>
<tr>
<td>2.31 Moore's Pattern of Resolution 3</td>
<td>51</td>
</tr>
<tr>
<td>2.32 Moore's Pattern of Resolution 4</td>
<td>52</td>
</tr>
<tr>
<td>2.33 Indexing L Patterns of Resolution 1</td>
<td>53</td>
</tr>
<tr>
<td>2.34 Pattern L1 of Resolution 2</td>
<td>54</td>
</tr>
<tr>
<td>2.35 Pattern L1 of Resolution 3</td>
<td>54</td>
</tr>
<tr>
<td>2.36 Pattern L1 of Resolution 4</td>
<td>55</td>
</tr>
<tr>
<td>2.37 Pattern L2 of Resolution 2</td>
<td>56</td>
</tr>
<tr>
<td>2.38 Pattern L2 of Resolution 3</td>
<td>56</td>
</tr>
</tbody>
</table>
3.16 The Three-Dimensional U Order of Resolution 2 ........................................ 91
3.17 Structure and Orientation of Level 1 .......................................................... 91
3.18 Structure and Orientation of Level 2 .......................................................... 93
3.19 Execution-Time Assessments for Encoding 3D Orders .............................. 94
3.20 Execution-Time Assessments for Decoding 3D Orders ............................ 95
3.21 The Four-Dimensional Hypercube ............................................................... 98
3.22 The Four-Dimensional U order of Resolution 1 .......................................... 99
3.23 The Z Order of Resolution 3 ........................................................................ 104
3.24 The X Order of Resolution 1 ........................................................................ 106
3.25 The X Order of Resolution 2 ........................................................................ 107

4.1 The Q1 Order of Resolution 1 ........................................................................ 119
4.2 Path of the Q1 Order of Resolution 1 ............................................................. 119
4.3 The Q1 Order of Resolution 2 ........................................................................ 120
4.4 Path of the Q1 Order of Resolution 2 ............................................................. 120
4.5 The Q1 Order of Resolution 3 ........................................................................ 121
4.6 Path of the Q1 Order of Resolution 3 ............................................................. 121
4.7 The Q2 Order of Resolution 1 ........................................................................ 126
4.8 The Q2 Order of Resolution 2 ........................................................................ 126
4.9 The Q2 Order of Resolution 3 ........................................................................ 127
4.10 Path of the Q2 Order of Resolution 1 ............................................................ 127
4.11 Path of the Q2 Order of Resolution 2 ............................................................ 128
4.12 Path of the Q2 Order of Resolution 3 ............................................................ 128
A.1 The Structure of the Software .......................... 213
Acknowledgement

The author wishes to acknowledge gratefully the valuable advice and guidance of Dr. Günther F. Schrack, who supervised this thesis throughout all stages. It is his research on quadtrees and the corresponding spatial order that inspired the author's initial interest in exploring a class of new hierarchical data structures. In the research progress, the author was always encouraged by Dr. Schrack's great patience and understanding. The author has also benefited by reading the up-to-date literature timely recommended by him. His meticulous scholarship on critically reading and discussing the penultimate draft of the thesis is especially appreciated. This research was, in part, supported by the Natural Sciences and Engineering Research Council (Research Grant 5-0148 to Dr. Schrack).

Sincere thanks should also be extended to the committee for co-supervising the research. Dr. Mabo Ito provided valuable suggestions to the macrostructure of this research. The queries from Dr. Don Gillies and Dr. Alain Fournier helped the author to refine some important concepts. The author has also benefited from the discussion with Dr. Rabab Ward in preparing the departmental examination. In the early phase of this research, Dr. David Kirkpatrick inspired the author to investigate and choose the appropriate methodology applied to the proposed subject.

At the final oral examination, several constructive suggestions were given by the Examination Committee. Especially, Dr. Jagadish carefully read the thesis and provided valuable comments. The author would like to express his sincere gratitude to all of the members of the Examination Committee.

The wife and daughter of the author have shown unusual patience and understanding during the period of this study. It is they who had to endure the prolonged absences and the chronic absent-mindedness. Without their support, it would have been impossible for the author to complete the research on schedule. The author would like to express his deepest gratitude to them.
Chapter 1

Introduction

The multi-dimensional lattice system is a fundamental model in modern computer sciences, especially in information systems. A record with \( n \) attributes can be mapped by hash functions to a point in an \( n \)-dimensional integer space. Thus a lattice can be defined to be the rectangular region that contains a subset of such points.

The requirement for ordering a set of multi-dimensional data arises frequently from two applications:

- Spatial Index Processing
- Secondary Key Retrieval

The first application requires developing a single numerical index on a one dimensional line for each point in multi-dimensional space, such that the spatial localizability can be preserved as best as possible. This work can be carried out by a mathematical transformation called spatial ordering. Usually, a parameter called \textit{resolution} is used to describe the granularity of the domain of a spatial order. For
example, a spatial order is of resolution $r$ if it is defined on a two-dimensional square image with $2^r \times 2^r$ pixels.

Interest in spatial orders has been high since the 1960s because of their broad applications to many fields in applied sciences. The most representative ones include Computer Graphics, Image Processing, Geographic Information Systems (GIS), Digital Halftoning, Mathematical Programming, and Image Encryption.

Our research concentrates on the application of spatial orders to information retrieval systems.

Faloutsos [17] showed that the Hilbert order exhibits satisfactory performance. His conclusion is supported by the mathematical analysis presented by Jagadish [31]. Recently, Abel and Mark [2] assessed empirically the relative merits of five orderings (row, row-prime, Hilbert, Morton and Gray-Morton) for four paradigmatic geographical data processing tasks in spatial analysis. They concluded that the Hilbert ordering deserves further investigation.

Several researchers have mentioned the lack of inexpensive encoding and decoding algorithms for the Hilbert ordering (e.g. [2], [10], [17], [18]).

The second application, secondary key retrieval, involves two issues: determining the resolution for each dimension in the multi-dimensional lattice system and ordering the data units on disks. Aho and Ullman [3] proposed a constrained nonlinear programming model applied to the partial match query. In a partial match query, a bit-string with $b_i$ bits called the field signature, is generated by the hash function $h_i$ with the specified field as its argument. The number of bits in a field signature
is called field resolution and it determines the cardinality of the hashed space. For instance, we may have:

\[
h_{name}(x) = \begin{cases} 
00 & \text{if } x \leq "G" \\
01 & \text{if } "G" < x \leq "N" \\
10 & \text{if } "N" < x \leq "T" \\
11 & \text{others}
\end{cases}
\]

In this example, the field resolution is 2 and the cardinality of the hashed space is 4. In Aho and Ullmans' model, the arguments are field resolutions and the objective function to be minimized is the expected value of data blocks that need to be retrieved from the secondary storage to answer a query.

The model proposed by Aho and Ullman presents an elegant mathematical formulation for the partial match query problem. The basic formulation has been developed into several variants. However, most models proposed are limited to the application of the partial match problem. The literature is relatively silent on the range query retrieval problem.

The performance of the secondary retrieval depends on the ordering of the data units on disks. Recently, Faloutsos [15] proposed a new approach for this issue. Using combinatorial analysis, he showed that the order determined by the Gray Code can significantly improve the cluster distribution of the records. The major open question that remains is the lack of symmetric Gray codes to reduce the bias introduced by the reflected Gray code.

We are motivated by the above issues.
This thesis is organized as follows. The Hilbert order is analyzed in Chapter 2. A recursive approach is presented first to analyze a typical paradigm. The approach is analytical, thus can be applied to other spatial orders. Next, a set of analytical formulas is derived to encode and decode the two-dimensional Hilbert order. The execution-time assessment suggests that they are faster than a look-up-table approach published previously. Then, the encoding and decoding algorithms for the three-dimensional Hilbert order are discussed. An approach based on the rotational matrices is presented. Since the encoding and decoding processes for high-dimensional Hilbert orders involve complicated operations, we propose a new spatial order, the U order, in Chapter 3. An analysis shows that the U order is superior to most other orders. The encoding and decoding formulas are presented. These operations are much simpler than those of the Hilbert order. In Chapter 4, several new spatial orders with the quadrant-recursive structure are investigated. The goal is to find the candidates that are competitive with the Hilbert order in performance and with relative simple encoding and/or decoding formulas. In Chapter 5, stochastic programming models for optimizing the resolutions of the multi-dimensional lattice system are proposed. The analysis concentrates on the range query problem. Three types of stochastic programming models are presented. Four paradigms are discussed. One of them leads to a result that is similar to the one used to evaluate the performance of disk drives. In Chapter 6, the symmetric Gray codes are analyzed. A heuristic approach is presented to construct the symmetric Hamiltonian paths in the five-, six-, and seven-dimensional hypercubes. Finally, a conclusion is given in Chapter 7. The development and imple-
mentation of the stochastic programming algorithms are described in Appendix A.
Numerical experiments are also reported.
Chapter 2

The Hilbert Order

2.1 Overview

A spatial ordering is a bijection relating a finite n-dimensional lattice to a finite set of natural numbers. Some well-known orders (shown with resolution $r = 2$), namely, B (Beyer) order, D (Cantor-diagonal) order, G (Gray) order, H (Hilbert) order, K (Knuth) order, R (Row) order, RP (Row-prime) order, S (Spiral) order, and Z (Morton) order are presented in Fig. 2.1 to Fig. 2.9.

Interest in spatial orders has been high since the 1960s because of their broad applications to many fields in applied sciences. The most representative ones are:

- **Computer Graphics, Image Processing and Geographic Information Systems (GIS):** The Z order corresponding to the quadtree and octree structures have attained high attention since 1970s. Samet has provided an excellent survey and presented detailed analyses in his two monographs ([60], [61]).
Figure 2.1: The B Order of Resolution 2

<table>
<thead>
<tr>
<th>15</th>
<th>7</th>
<th>13</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>11</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>2</td>
<td>10</td>
</tr>
</tbody>
</table>

Figure 2.2: The D Order of Resolution 2

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 2.3: The G Order of Resolution 2

<table>
<thead>
<tr>
<th>10</th>
<th>11</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>13</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Figure 2.4: The Hilbert Order of Resolution 2

Figure 2.5: The K Order of Resolution 2

Figure 2.6: The R Order of Resolution 2
Figure 2.7: The RP Order of Resolution 2

Figure 2.8: The S Order of Resolution 2

Figure 2.9: The Z Order of Resolution 2
• Digital Halftoning: Limb presented an Ordered Dither method in 1969 [37]. Beyer [6] proposed an optimal criterion based on the wavelength analysis of the dot patterns (the spatial order he designed is called B order in the foregoing). Knuth [36] claimed another class of spatial orders and initialized the research for Dot Diffusion (the spatial order he designed is called K order in the foregoing).

• Mathematical Programming [58].

• Image Encryption [8].

The literature shows that various spatial orders have found extensive applications in modern science and engineering. Our research will concentrate on the application of spatial orders to information retrieval systems.

Due to its desirable property, the Hilbert order has attained intensive interest in the literature. In this chapter, a recursive approach is presented first to analyze a typical application. The result matches the one reported by the algorithmical method suggested in the literature. Next, a set of analytical formulas based on Boolean algebra is proposed to encode and decode the two-dimensional Hilbert order. Finally, the encoding and decoding algorithms for the three-dimensional Hilbert order are discussed.

2.2 Metric Analysis of the 2D Hilbert Order

Orenstein [47] showed that the Z order is well suited for range queries. Faloutsos [17] studied the performance of the Hilbert order for range query and near neighbour
query problems. Evaluations of the performance of the Hilbert, Z, G, R and RP orders applied to spatial data systems were presented by Jagadish [31] and Abel [2].

The metrics they employed are based on the following rationale. A particular spatial order determines a linear traversal in the multi-dimensional space. It specifies the order with which the data blocks are allocated on disk. The appropriate measures of cost associated with a particular linear traversal are:

1. number of disk blocks fetched;

2. number of non-consecutive disk blocks fetched. It is preferred to fetch a set of consecutive disk blocks, which is called a "cluster" or "run", due to the seek time involved.

Jagadish [31] reported a set of statistics (see Table 2.1), which are based on two types of templates: $1 \times n$ region for partial match query, where $n = 2^r$, and $2 \times 2$ region for range query. Since it is preferred to process fewer clusters, Table 2.1 suggests that the Hilbert order exhibits good performance. In addition, since the Hilbert order exhausts a quadrant of a square before exiting it and does not include diagonal paths (like the Z order), the locality of the range query is well preserved. This provides, among others, the motivation to design more efficient encoding and decoding algorithms for the Hilbert order. Fig. 2.10 to Fig. 2.12 show the Hilbert order with resolution $r = 1, 2,$ and $3$, respectively.

It is believed that the range metric analysis is of central importance for understanding the inherent property of the Hilbert order and its significance for applications. However, a range metric analysis for this order is not presented in the literature.
<table>
<thead>
<tr>
<th>Order Type</th>
<th>Partial Match Query</th>
<th>Range Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>$2^{r-1} + 0.5$</td>
<td>2</td>
</tr>
<tr>
<td>RP</td>
<td>$2^{r-1} + 2^{-r-1}$</td>
<td>$2 - (2^r - 1)^{-1}$</td>
</tr>
<tr>
<td>Z</td>
<td>$1.5 \times 2^{r-1}$</td>
<td>2.625</td>
</tr>
<tr>
<td>G</td>
<td>$2^{r-1} + 2^{-r-1}$</td>
<td>2.5</td>
</tr>
<tr>
<td>H</td>
<td>$2^{r-1} + 2^{-r-1}$</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2.1: Expected Number of Clusters for Resolution $r$

Figure 2.10: The Hilbert Order of Resolution 1

Figure 2.11: The Hilbert Order of Resolution 2
due to its complexity. For this reason, in this chapter we present a comprehensive recursive approach for the range metric analysis of the Hilbert order. This approach will also serve as a general procedure to conduct a metric analysis for other spatial orders proposed in Chapter 3 and 4.

A record consisting of two attributes corresponds to a point in the two-dimensional Cartesian space and a collection of such records corresponds to a lattice. For convenience, in the following an equivalent system in which the points are represented by cells is considered. Without loss of generality, the discussion is applied to a square lattice system with $2^r \times 2^r$ cells, where the parameter $r$ is the resolution.

In a range query, the ranges of the attributes are specified in a multiple attribute record. Geometrically, a range query to the two-attribute record system corresponds to a subset of rectangular shape in the two-dimensional lattice system. Given a lattice
with $2^r \times 2^r$ cells, there is a large number of various rectangular subsets. Apparently it is impractical to conduct an exhaustive analytical investigation for all of them, thus a particular subset is to be selected as a representative to gain some mathematical insight.

A set of terminology covering some key concepts in the metric analysis is defined here.

**Definition 2.2.1** A *cluster* is a subset of records with contiguous addresses in the one-dimensional space. Equivalently, it is a subset of cells with contiguous indices in the two-dimensional lattice. Such a cluster can be represented graphically with a path that links the cells concerned.

**Definition 2.2.2** A *template* is a square region with $2 \times 2$ cells.

According to the number of clusters contained, there are four types of templates:

**Definition 2.2.3** A $C_k$ template ($k = 1, 2, 3, 4$) is a template in which the four cell indices form $k$ clusters.

For example, in Fig. 2.12,

- the template with indices 8, 9, 10, and 11 is a $C_1$ template;
- the template with indices 5, 6, 57, and 58 is a $C_2$ template;
- the template with indices 8, 11, 12, and 13 is also a $C_2$ template;
- the template with indices 10, 31, 32, and 53 is a $C_3$ template;
the template with indices 6, 9, 54, and 57 is a $C_4$ template.

In the rest of this chapter as well as Chapter 3 and 4, the range query associated with the template defined above will serve as a representative for the purpose of analytical analysis. The number of $C_k$ templates in a $2^r \times 2^r$ square image will be denoted $N_r^{(k)}(k = 1, 2, 3, 4)$.

A particular spatial order specifies a rule to allocate the records into a one-dimensional space. Consideration is to be given to the number of clusters that need to be retrieved in a query. A small number is preferred to a large number because of the access architecture of modern disk systems. The expected value of clusters is used as the metric to evaluate the performance.

### 2.2.1 Range Query

A graphical representation for the Hilbert order is more convenient for a metric analysis. The graphs of resolution 1, 2, 3 and 4 are depicted in Fig. 2.13 to Fig. 2.16, respectively.

The analysis is presented as a series of theorems:

**Theorem 2.2.1**

\[ N_r^{(1)} + N_{r-1}^{(1)} = \left( \frac{19}{48} \right)^r \frac{1}{3} \quad (r \geq 2) \]

**[proof]** It follows from Fig. 2.13 and Fig. 2.14 that $N_1^{(1)} = 1$ and $N_2^{(1)} = 4N_1^{(1)} + 1 = 5$. The latter is established due to the fact that, in addition to the four one-cluster templates arising from replicating the Hilbert order with resolution 1, a new one-cluster
template with a continuous path is formed. In general, \( N_k^{(1)} = 4N_{k-1}^{(1)} + 1 \) holds for \( k \) even and \( N_k^{(1)} = 4N_{k-1}^{(1)} \) holds for \( k \) odd \( (k = 2, 3, \ldots, r) \), according to the structure of the Hilbert order. With the intermediate variable \( T_k = N_k^{(1)} + N_{k+1}^{(1)} \) \( (k = 1, 2, \ldots, r-1) \), we have:

\[
T_k = 4T_{k-1} + 1 \quad k = 2, 3, \ldots, r-1
\]
Figure 2.15: Path of the Hilbert Order of Resolution 3

By iteration we have:

\[ T_k = 4^j T_{k-j} + \frac{4^j - 1}{3} \quad j = 1, 2, \ldots, k - 2 \]

The closed form is obtained by setting \( j \) to \( k - 2 \):

\[ T_k = 4^{k-2} T_2 + \frac{4^{k-2} - 1}{3} \]

\[ = \left( \frac{19}{12} \right) 4^k - \frac{1}{3} \]

Therefore,

\[ T_{r-1} = \left( \frac{19}{48} \right) 4^r - \frac{1}{3} \quad \square \]

**Theorem 2.2.2**

\[ N_r^{(2)} + N_{r-1}^{(2)} = \left( \frac{17}{32} \right) 4^r - \left( \frac{11}{8} \right) 2^r \quad (r \geq 2) \]

**[proof]** According to the structure of the Hilbert order (the graphs with resolution up to 4 are presented in Fig. 2.13 to Fig. 2.16), a sequence can be established as
Figure 2.16: Path of the Hilbert Order of Resolution 4

follows:

\[
\begin{align*}
N_1^{(2)} &= 0 \\
N_2^{(2)} &= 4N_1^{(2)} + 3 \\
N_3^{(2)} &= 4N_2^{(2)} + 8 \\
N_4^{(2)} &= 4N_3^{(2)} + 14 \\
N_5^{(2)} &= 4N_4^{(2)} + 30 \\
N_6^{(2)} &= 4N_5^{(2)} + 58 \\
&\ldots \ldots
\end{align*}
\]

where the first term of the right-hand side is according to the quadruple constructing process from level \(k - 1\) to level \(k\) \((k = 2, 3, \ldots, r)\), while the second term comes from the contribution of the additional new templates generated by the cells in the single
row or column which is one of the sides of the lower level Hilbert order. The subset consisting of all of the new templates is shaped like a Greek Cross with width of two cells.

With the intermediate variable $T_k = N_k^{(2)} + N_{k+1}^{(2)}$ ($k = 1, 2, \ldots, r - 1$), we have:

$$T_1 = 3$$

$$\ldots \ldots$$

$$T_k = 4T_{k-1} + 2^{k-1}b$$

$(k \geq 2)$

where $b = \frac{11}{2}$. Consequently, by iteration,

$$T_k = 4^iT_{k-i} + 2^{k-1}(2^i - 1)b$$

Let $j = k - 2$, then

$$T_k = (\frac{17}{8})4^k - (\frac{11}{4})2^k$$

Therefore,

$$T_{r-1} = (\frac{17}{32})4^r - (\frac{11}{8})2^r \quad \square$$

**Theorem 2.2.3**

$$N_r^{(3)} + N_{r-1}^{(3)} = 4^{r-1} - 2^r + 1 \quad (r \geq 2)$$

**[proof]** According to the structure of the Hilbert order a sequence can be established.

With the intermediate variable $T_k = N_k^{(3)} + N_{k+1}^{(3)}$ ($k = 1, 2, \ldots, r - 1$), we have:

$$T_1 = 1$$
\[ T_k = 4T_{k-1} + 2^{k+1} - 3 \quad (k \geq 2) \]

Consequently, by iteration,
\[ T_k = 4^j T_{k-j} + 2^{k+1}(2^j - 1) - (4^j - 1) \]

Let \( j = k - 2 \), then
\[ T_k = 4^k - 2^{k+1} + 1 \]

Therefore,
\[ T_{r-1} = 4^{r-1} - 2^r + 1 \quad \square \]

**Theorem 2.2.4**
\[ N_r^{(4)} + N_{r-1}^{(4)} = (\frac{7}{96})4^r - (\frac{5}{8})2^r + \frac{4}{3} \quad (r \geq 2) \]

[proof] According to the structure of the Hilbert order a sequence can be established. With the intermediate variable \( T_k = N_k^{(4)} + N_{k+1}^{(4)} \) \((k = 1, 2, \ldots, r - 1)\), we have:
\[ T_1 = 0 \]
\[ \cdots \cdots \]
\[ T_k = 4T_{k-1} + (\frac{5}{2})2^{k-1} - 4 \quad (k \geq 2) \]
By iteration,

\[ T_k = 4^i T_{k-j} + \left(\frac{5}{2}\right) 2^{k-1}(2^i - 1) - \left(\frac{4}{3}\right)(4^i - 1) \]

Let \( j = k - 2 \), then

\[ T_k = \left(\frac{7}{24}\right) 4^k - \left(\frac{5}{4}\right) 2^k + \frac{4}{3} \]

Therefore,

\[ T_{r-1} = \left(\frac{7}{96}\right) 4^r - \left(\frac{5}{8}\right) 2^r + \frac{4}{3} \quad \square \]

**Corollary 2.2.1** When \( r \) is large, the expectation of the number of clusters per range query tends to:

\[ R_r = 2 \]

**[proof]** There are \((2^k - 1)^2\) templates in a lattice system with \(2^k \times 2^k\) cells \((k = 1, 2, \ldots, r)\). Let \( p_1, p_2, p_3, \) and \( p_4 \) denote the occurrence frequencies of the templates with one, two, three, and four clusters in the lattice of resolution \( r \). Then we have:

\[ p_1 = \lim_{r \to \infty} \frac{N_r^{(1)} + N_{r-1}^{(1)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} = \frac{19}{60} \]

\[ p_2 = \lim_{r \to \infty} \frac{N_r^{(2)} + N_{r-1}^{(2)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} = \frac{17}{40} \]

\[ p_3 = \lim_{r \to \infty} \frac{N_r^{(3)} + N_{r-1}^{(3)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} = \frac{1}{5} \]

21
\[ p_4 = \lim_{r \to \infty} \frac{N_r^{(4)} + N_{r-1}^{(4)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} \]
\[ = \frac{7}{120} \]

Consequently, when \( r \) is large, the expectation of the number of clusters per range tends to:

\[ R_r = 1p_1 + 2p_2 + 3p_3 + 4p_4 \]
\[ = \frac{19}{60} + \frac{17}{20} + \frac{3}{5} + \frac{7}{30} \]
\[ = 2 \]

The statistics obtained above will be compared with other typical spatial orders applied in data processing systems later.

### 2.3 Encoding and Decoding the 2D Hilbert Order

A space-filling curve, defined on an integer lattice, defines a corresponding order, e.g., the Hilbert order. By numbering the points visited sequentially on the curve, location codes are defined. For plotting purposes, several sequential algorithms for the Hilbert order (and others) have been proposed for generating the coordinates in sequence ([24], [65]). For many applications in information retrieval systems, however, random-access algorithms are required, i.e., encoding and decoding without previous context or historical information. An encoding algorithm derives the location code for a given coordinate point, while a decoding algorithm derives the coordinates from a given location code.
It appears that only the algorithms proposed by Fisher [18] and Jagadish [31] are random-access algorithms. Fisher's algorithms are serial-bit processes and use two short look-up tables. The algorithms are stated in terms of the occam language. Jagadish only presents the encoding algorithm and also uses look-up tables.

In this section, encoding and decoding algorithms are proposed which do not depend on look-up tables; they are stated in terms of an analytical formulation; in addition, they appear to be faster than Fisher's according to experimental evidence.

2.3.1 Encoding

The Hilbert orders and curves of various resolutions have been depicted in Fig. 2.10 to Fig. 2.16. In this section, the iteration rules to construct the Hilbert order are described first. These rules establish the basis for deriving the analytic encoding and decoding formulas.

The Hilbert order is to be established in a subset $D = \{0 \leq x < 2^r, 0 \leq y < 2^r\}$ in the two-dimensional integer space. The parameter $r$ is the specified resolution or level of the spatial order. Small values of $r$ correspond to low resolution or low levels.

Given the coordinates of a particular point $P$ in the lattice system $D$, the corresponding value of $P$ in the Hilbert order ($H$-value in brief) is to be determined. This process is called encoding.

Let the coordinate values $x$ and $y$ be expressed by bit-strings, $x = x_{r-1} \ldots x_1 x_0$ and $y = y_{r-1} \ldots y_1 y_0$, and the corresponding $H$-value be expressed in terms of quaternary digits ($h$-digits in brief), $h = 4^{r-1} \times h_{r-1} + \ldots + 4^1 \times h_1 + 4^0 \times h_0$. At level $i$
(i = 1, 2, ..., r), given \( x_{r-i} \) and \( y_{r-i} \), the \( i \)-th most significant digit of the coordinates \( x \) and \( y \), \( h_{r-i} \), the \( i \)-th most significant digit of the H-value is to be determined.

Some terminology is required in the following. A unit is an item specifying the indexing strategy for four quadrants of a square region. It can be represented by a simple graph with three edges and four nodes labeled by indexes in terms of quaternary digits (\( n \text{-} digits \) in brief), as shown in Fig. 2.17. Two important features are associated with a unit: orientation and aspect. The orientation indicates the current rotation state of a unit, while the aspect specifies the relation between \( h \text{-} digits \) and \( n \text{-} digits \).

The nodes of a unit are successively indexed clockwise, but the sequence of \( h \text{-} digits \) may be either clockwise or counterclockwise. The orientation of the unit at the lowest level (called level 1, due to the correspondence to the resolution \( r = 1 \)) is called the principal orientation of the Hilbert order.

Both orientation and aspect determine the structure of a unit. However, the orientation is the key feature in the recursive construction process and its iteration can be isolated from the effect of the aspect. According to the inherent property of the Hilbert order, the units generated from node 0, 1, 2, 3 of the unit at level \( k \) (\( k = 1, 2, \ldots, r \)) always rotate 90 degrees clockwise, 0 degree, 0 degree, and 90 degrees counterclockwise, to the orientation of the unit at level \( k \), respectively.

Given a unit with a specific orientation, the aspect affects only the current digit of the H-value. The aspect of a unit generated from the nodes 0 and 3 of the unit at the lower level is always reversed, while a unit generated from node 1 and 2 is
The analysis for the orientation will be developed in the two-dimensional Cartesian coordinate system. As a reference point, the coordinate origin is set at the bottom left of the lattice. The principal structure at level 1 is specified as shown in Fig. 2.17. This structure determines the overall appearance of the order considered.

![Figure 2.17: Structure and Orientation of Level 1](image)

Level 2 is generated by constructing four units, labeled $U_v$ ($v = 0, 1, 2, 3$), from the corresponding indexed nodes of the unit at level 1 (see Fig. 2.18). The indexes of nodes are obtained by concatenating a quaternary digit to the index of the unit. Note the nodes in these units are still indexed progressively clockwise.

![Figure 2.18: Structure and Orientation of Level 2](image)

The orientation of a unit is specified by a normalized vector $m$. According to
the specified arrangement of the Hilbert order relative to the coordinate system, 
\[ m \in \{(0,1)^T, (-1,0)^T, (0,-1)^T, (1,0)^T\} \] \(^1\). Two units with their normal vector are demonstrated in Fig. 2.19.

![Figure 2.19: Two Instances with Directional Vectors at Level 2](image)

The iteration of the orientation from the current level to the next level could be characterized by multiplying vector \( m \) by a two-dimensional rotation matrix. If the positive rotation direction is defined counterclockwise, then a rotation with angular shift \( \theta \) can be represented by:

\[
R(\theta) = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) \\
\sin(\theta) & \cos(\theta)
\end{bmatrix}.
\]

For instance,

\[
R(90^\circ) = \begin{bmatrix}
0 & -1 \\
1 & 0
\end{bmatrix}.
\]

Since matrix multiplication may involve many operations, we propose an alternative approach called the *directional index method*. Based on this method, analytic formulas for encoding and decoding will be derived.

\(^1\)The superscript "T" means transposition. In the following, it will be ignored for simplicity.
The idea of indexing the directions is suggested by the fact that the rotational freedom comprises only the four principal axis directions, namely, $x, y, -x,$ and $-y,$ thus they are sufficient to characterize a unit's orientation. Therefore, a relation between the directional vector $m$ and an index $v$ can be defined (see Fig. 2.20):

<table>
<thead>
<tr>
<th>vector $m$</th>
<th>index $v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(0,1)^T$</td>
<td>0</td>
</tr>
<tr>
<td>$(-1,0)^T$</td>
<td>1</td>
</tr>
<tr>
<td>$(0,-1)^T$</td>
<td>2</td>
</tr>
<tr>
<td>$(1,0)^T$</td>
<td>3</td>
</tr>
</tbody>
</table>

Since $v$ has four possible values, it can be represented by two bits $a$ and $b$. Similarly,

![Indexed Directions](image)

Figure 2.20: Indexed Directions

two bits $n_{2k-1}$ and $n_{2k-2}$ are used to represent an $n$ digit. From here on, the following notation will be used for Boolean variables $p$ and $q$:

- $p$ and $q$ is denoted as $pq$,
- $p$ inclusive or $q$ is denoted as $p + q$,
- $p$ exclusive or $q$ is denoted as $p \oplus q$,
the complement of \( p \) is denoted as \( \bar{p} \).

According to the inherent structure of the Hilbert order being discussed, \( n_{2k-1} \) and \( n_{2k-2} \) can be expressed in terms of \( a, b, x_{k-1} \) and \( y_{k-1} \):

\[
\begin{align*}
n_{2k-1} &= \bar{a}bx_{k-1} + \bar{a}by_{k-1} + a\bar{b}x_{k-1} + ab\bar{y}_{k-1} \\
         &= \bar{b}(a \oplus x_{k-1}) + b(a \oplus y_{k-1}) \\
n_{2k-2} &= \bar{b} \oplus x_{k-1} \oplus y_{k-1}
\end{align*}
\]

\( k = 1, 2, \ldots, r. \)

The iteration of \( v \) can be carried out by either multiplying or adding an incremental factor that is governed by the inherent structure of a particular Hilbert order. In the following the addition approach is chosen by reason that it appears to facilitate analysis as well as implementation. Denote the incremental factor by \( dv \), the transition rules may then be explicitly presented by two \( if \) clauses:

\[
\begin{align*}
if \ n\_digit = 0 \ then \ dv &= 3; \\
else if \ n\_digit = 3 \ then \ dv &= 1; \\
else \ dv &= 0;
\end{align*}
\]

Since \( dv \in \{0, 1, 3\} \), two bits are sufficient to represent it. With the two bits denoted as \( c \) and \( d \), the relation defined by the \( if \) clauses above may be written as a table

\[
\begin{array}{|c|c|c|}
\hline
n_{2k-1} & n_{2k-2} & c & d \\
\hline
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 \\
\hline
\end{array}
\]

28
which in turn can be formulated analytically as two Boolean expressions:

\[ c = \bar{n}_{2k-1} \bar{n}_{2k-2} \]  
\[ d = n_{2k-1} \oplus \bar{n}_{2k-2} \]  

The updated value \( v' \) of the directional index \( v \) is obtained by the addition:

\[ v' = (v + dv) \mod 4 \]  

Let \( v' \) be represented by two bits \( a' \) and \( b' \), then Eq.(2.3) can be formulated as:

\[ a' = (bd) \oplus a \oplus c \]  
\[ b' = b \oplus d \]  

Substituting (2.1) and (2.2) into (2.4) and (2.5) yields

\[ a' = [b(n_{2k-1} \oplus \bar{n}_{2k-2})] \oplus a \oplus (\bar{n}_{2k-1} \bar{n}_{2k-2}) \]  
\[ b' = b \oplus n_{2k-1} \oplus \bar{n}_{2k-2} \]  

In order to obtain the final expressions of \( a' \) and \( b' \) in terms of \( a, b, x_{k-1}, \) and \( y_{k-1}, \) tedious algebraical calculations need to be performed which will not be presented here. The iteration process of the directional index \( v \) can be analytically expressed as as follows:

\[ a_{r-1} = 0 \]  
\[ b_{r-1} = 0 \]  
\[ a_{j-1} = a_j(x_j \oplus y_j) + b_jx_jy_j + \bar{b_j} + x_j + y_j \]  
\[ b_{j-1} = b_j(a_j \oplus \bar{x_j}) + \bar{b_j}(a_j \oplus \bar{y_j}) \]  

\[ j = r - 1, \ldots, 2, 1. \]
To determine the H-value of a particular point with two integer coordinates \( x \) and \( y \), the aspect of the units have to be taken into consideration. Let the two bits of an \( h \)-digit be denoted by \( h_{2k} \) and \( h_{2k+1} \), respectively. Then one has

\[
\begin{align*}
    h_{2k} &= x_k \oplus y_k \\
    h_{2k+1} &= \bar{a}_k \bar{b}_k x_k + \bar{a}_k b_k \bar{y}_k + a_k \bar{b}_k \bar{x}_k + a_k b_k y_k \\
    &= \bar{b}_k (a_k \oplus x_k) + b_k (a_k \oplus y_k)
\end{align*}
\]

\( k = 0, 1, \ldots, r - 1 \).

The order of this algorithm is \( 31r \).

As an example, the Hilbert order of resolution \( r = 4 \) is generated as shown in Figure 2.21.

### 2.3.2 Decoding

Decoding is the inverse process of encoding: given the H-value of a particular point, the corresponding coordinate pair \( (x, y) \) is to be determined. An analytical formula for decoding can be derived by the same methodology as applied to encoding. Let an \( h \)-digit be represented by the two bits \( h_{2k-1} \) and \( h_{2k-2} \), and an \( n \)-digit be represented by the two bits \( n_{2k-1} \) and \( n_{2k-2} \). Then one has

\[
\begin{align*}
    n_{2k-1} &= b \oplus h_{2k-1} \\
    n_{2k-2} &= b \oplus h_{2k-2}
\end{align*}
\]

\( k = 1, 2, \ldots, r \)

where \( a \) and \( b \) are the two bits of the directional index \( v \).
<table>
<thead>
<tr>
<th>85</th>
<th>86</th>
<th>89</th>
<th>90</th>
<th>101</th>
<th>102</th>
<th>105</th>
<th>106</th>
<th>149</th>
<th>150</th>
<th>153</th>
<th>154</th>
<th>165</th>
<th>166</th>
<th>169</th>
<th>170</th>
</tr>
</thead>
<tbody>
<tr>
<td>83</td>
<td>82</td>
<td>93</td>
<td>92</td>
<td>99</td>
<td>98</td>
<td>109</td>
<td>108</td>
<td>147</td>
<td>146</td>
<td>157</td>
<td>156</td>
<td>163</td>
<td>162</td>
<td>173</td>
<td>172</td>
</tr>
<tr>
<td>80</td>
<td>81</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>110</td>
<td>111</td>
<td>144</td>
<td>145</td>
<td>158</td>
<td>159</td>
<td>160</td>
<td>161</td>
<td>174</td>
<td>175</td>
</tr>
<tr>
<td>79</td>
<td>76</td>
<td>75</td>
<td>74</td>
<td>117</td>
<td>116</td>
<td>115</td>
<td>112</td>
<td>143</td>
<td>140</td>
<td>139</td>
<td>138</td>
<td>181</td>
<td>180</td>
<td>179</td>
<td>176</td>
</tr>
<tr>
<td>78</td>
<td>77</td>
<td>72</td>
<td>73</td>
<td>118</td>
<td>119</td>
<td>114</td>
<td>113</td>
<td>142</td>
<td>141</td>
<td>136</td>
<td>137</td>
<td>182</td>
<td>183</td>
<td>178</td>
<td>177</td>
</tr>
<tr>
<td>65</td>
<td>66</td>
<td>71</td>
<td>70</td>
<td>121</td>
<td>120</td>
<td>125</td>
<td>126</td>
<td>129</td>
<td>130</td>
<td>135</td>
<td>134</td>
<td>185</td>
<td>184</td>
<td>189</td>
<td>190</td>
</tr>
<tr>
<td>64</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>122</td>
<td>123</td>
<td>124</td>
<td>127</td>
<td>128</td>
<td>131</td>
<td>132</td>
<td>133</td>
<td>186</td>
<td>187</td>
<td>188</td>
<td>191</td>
</tr>
<tr>
<td>63</td>
<td>62</td>
<td>49</td>
<td>48</td>
<td>47</td>
<td>44</td>
<td>43</td>
<td>42</td>
<td>213</td>
<td>212</td>
<td>211</td>
<td>208</td>
<td>207</td>
<td>206</td>
<td>193</td>
<td>192</td>
</tr>
<tr>
<td>60</td>
<td>61</td>
<td>50</td>
<td>51</td>
<td>46</td>
<td>45</td>
<td>40</td>
<td>41</td>
<td>214</td>
<td>215</td>
<td>210</td>
<td>209</td>
<td>204</td>
<td>205</td>
<td>194</td>
<td>195</td>
</tr>
<tr>
<td>59</td>
<td>56</td>
<td>55</td>
<td>52</td>
<td>33</td>
<td>34</td>
<td>39</td>
<td>38</td>
<td>217</td>
<td>216</td>
<td>221</td>
<td>222</td>
<td>203</td>
<td>200</td>
<td>199</td>
<td>196</td>
</tr>
<tr>
<td>58</td>
<td>57</td>
<td>54</td>
<td>53</td>
<td>32</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>218</td>
<td>219</td>
<td>220</td>
<td>223</td>
<td>202</td>
<td>201</td>
<td>198</td>
<td>197</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>9</td>
<td>10</td>
<td>31</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>229</td>
<td>228</td>
<td>227</td>
<td>224</td>
<td>245</td>
<td>246</td>
<td>249</td>
<td>250</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>8</td>
<td>11</td>
<td>30</td>
<td>29</td>
<td>24</td>
<td>25</td>
<td>230</td>
<td>231</td>
<td>226</td>
<td>225</td>
<td>244</td>
<td>247</td>
<td>248</td>
<td>251</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>13</td>
<td>12</td>
<td>17</td>
<td>18</td>
<td>23</td>
<td>22</td>
<td>233</td>
<td>232</td>
<td>237</td>
<td>238</td>
<td>243</td>
<td>242</td>
<td>253</td>
<td>252</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>234</td>
<td>235</td>
<td>236</td>
<td>239</td>
<td>240</td>
<td>241</td>
<td>254</td>
<td>255</td>
</tr>
</tbody>
</table>

Figure 2.21: The Hilbert Order of Resolution 4
The transition rules for iterating the index $v$ are the same as applied to the encoding process. Substituting (2.8) and (2.9) into (2.6) and (2.7) yields

$$a' = [b((b \oplus h_{2k-1}) \oplus (\bar{b} \oplus h_{2k-2}))) \oplus a \oplus ((\bar{b} \oplus h_{2k-1})(\bar{b} \oplus h_{2k-2}))$$

$$b' = b \oplus ((b \oplus h_{2k-1}) \oplus (\bar{b} \oplus h_{2k-2}))$$

Extensive operations in Boolean algebra need to be performed to obtain a simplified analytical formula set. The detailed derivation is not reproduced here. The result is

$$a' = a \oplus (\bar{h}_{2k-1} \bar{h}_{2k-2})$$

$$b' = b \oplus h_{2k-1} \oplus \bar{h}_{2k-2}$$

To determine the values of the coordinates $x$ and $y$ from the $h\_digits$, the aspect of the units has to be considered. According to the relation between $n\_digit$ and $h\_digit$ (Eq. (2.8), (2.9)),

$$x_{k-1} = \bar{a} \bar{b}h_{2k-1} + \bar{a}b(\bar{h}_{2k-1} \oplus h_{2k-2}) + abh_{2k-1} + ab(h_{2k-1} \oplus h_{2k-2})$$

$$= (b\bar{h}_{2k-2}) \oplus a \oplus h_{2k-1}$$

$$y_{k-1} = \bar{a}b(h_{2k-1} \oplus h_{2k-2}) + \bar{a}b\bar{h}_{2k-1} + a\bar{b}(\bar{h}_{2k-1} \oplus h_{2k-2}) + abh_{2k-1}$$

$$= (b + h_{2k-2}) \oplus a \oplus h_{2k-1}$$
Finally, the decoding formula can be expressed as follows:

\[
\begin{align*}
    a_{r-1} &= 0 \\
    b_{r-1} &= 0 \\
    a_{j-1} &= a_j h_{2j} + \overline{h_{2j}} (a_j \oplus \overline{h_{2j+1}}) \\
          &= a_j \oplus (h_{2j} + h_{2j+1}) \\
    b_{j-1} &= \overline{b_j} \oplus h_{2j} \oplus h_{2j+1} \\
          &\quad j = r - 1, \ldots, 2, 1.
\end{align*}
\]

\[
\begin{align*}
    x_k &= (b_k h_{2k}) \oplus a_k \oplus h_{2k+1} \\
    y_k &= (b_k + h_{2k}) \oplus a_k \oplus h_{2k+1} \\
          &\quad k = 0, 1, \ldots, r - 1.
\end{align*}
\]

The order of this algorithm is $19r$.

### 2.3.3 Implementation and Execution-Time Assessment

The operands in the foregoing formulas are binary digits. Some languages, however, support the bit-wise logical operations *and*, *or*, *exclusive or*, and *complement* on an entire machine word.

Therefore, two integers, $h_{\text{even}}$ and $h_{\text{odd}}$, are introduced. They comprise the bits at the even and odd digit positions of the location code, respectively. Thus, when interleaved bit-wise, they will yield the location code.

Assuming that $x, y, h_{\text{even}}$ and $h_{\text{odd}}$ are declared as integers, the encoding algorithm
can be implemented in the C language as follows:

```c
mask = (1 << res) - 1;
heven = x ^ y;
temp1 = x & y;
temp2 = (~ x & ~ y) & mask;
a = 0;
b = 0;
for (i = 0; i < res - 1; ++i)
{
    a = ((a & heven) | (b & temp1) | (~ b & (temp2))) >> 1;
    b = (((b & (a ^ ~ x)) | (~ b & (a ^ ~ y))) & mask) >> 1;
}
hoed = (~ b & (a ^ x)) | (b & (~ a ^ y));
```

The decoding algorithm becomes:

```c
mask = (1 << res) - 1;
a = 0;
b = 0;
temp = heven ` hodd;
for (i = 0; i < res - 1; ++i)
{
    a = (a ^ (~ (heven | hodd))) >> 1;
    b = ((~ b & mask) ^ temp) >> 1;
}
x = (b & ~ heven) ^ a ^ hodd;
y = (b | heven) ^ a ^ hodd;
```

The proposed algorithms as well as the algorithms presented in [18] were implemented in the C language. For small values of the resolution, the overhead cost of the system is significant. Thus we choose r=7 as the starting point. The time is evaluated in normalized values. The base used is the average time for encoding (or decoding) a
point over 4’ points.

It should be mentioned that the performance of the algorithms proposed in this section and Fisher’s algorithms differs only by a constant. A simple analysis for the implemented codes gives the following results:

- **Encoding:**
  
  our algorithm: 21r
  
  Fisher’s algorithm: 23r
  
  Thus, the ratio of the constants is $23/21 \approx 1.095$, while the experiment gives 1.2 approximately.

- **Decoding:**
  
  our algorithm: 16r
  
  Fisher’s algorithm: 23r
  
  Thus, the ratio of the constants is $23/16 = 1.4375$, while the experiment gives 1.36 approximately.

The time assessment employs a C library function “clock”. Another C library function “times” is also used, and the simulation result is similar. It is not reported here to save space. The programs have been run on SPARC stations 1, 2, IPC, and IPX. The data reported are from IPX. The results of execution-time tests for encoding and decoding are shown in Figure 2.22 and Figure 2.23, respectively.
Figure 2.22: Comparison of Encoding Algorithms for the Hilbert Order

Figure 2.23: Comparison of Decoding Algorithms for the Hilbert Order
2.4 Encoding and Decoding the 3D Hilbert Order

The representation of three-dimensional data is important because of its crucial function in solid modeling in computer aided design and computer graphics ([22], [61]), as well as the reported applications to spatial database management and image processing ([2], [17]).

We believe that it is worthwhile to develop algorithms transforming three-dimensional data to a one-dimensional ordering and vice versa.

The significance of the three-dimensional Hilbert ordering is suggested by the active research in two dimensions reported in the literature ([2], [10], [17], [18], [24], [31], [39], [65], etc.). However, little has been published in the literature on three-dimensional orders.

In this section, algorithms for the three-dimensional Hilbert order are discussed. A digit-wise approach which maps the coordinates \((x, y, z)\) of a particular point to the corresponding value in the three-dimensional Hilbert ordering and vice versa is described. The discussion for encoding is presented first. A small set of terminology is defined at the beginning, followed by the specification of geometric primitives. The constructing rule is then explored by means of the top-down approach. As the consequence of such a hierarchical process, an iterative algorithm is designed. The decoding process is described in the next section.
2.4.1 Encoding

The main purpose of this section is to describe the iteration rules applied to construct the Hilbert order.

The Hilbert order is to be established in a subset $D = \{0 \leq x < 2^r, 0 \leq y < 2^r, 0 \leq z < 2^r\}$ in the three-dimensional integer space.

The top-down approach is to be applied to construct an encoding formula for the Hilbert order. The cubic lattice is partitioned and indexed similarly to the two-dimensional case.

A unit is an item specifying the indexing strategy for eight octants of a cubic region. It can be represented by a set of $2 \times 2 \times 2$ cells, or by a simple graph with seven edges and eight nodes, as shown in Fig. 2.24. The latter will be used in this section, since it appears more convenient to present the unit’s two important features: orientation and aspect. The orientation specifies the current rotation state of a unit, which can be represented by two normalized vectors. Given a unit with a specified orientation, the aspect affects the current digit of the H-value only. According to the inherent property of the Hilbert order, it is observed that the aspect of a unit generated from the nodes 0, 1, 6 and 7 of the unit at the lower level is always reversed, while a unit from nodes 2, 3, 4 and 5 remains the same. Consequently, a Boolean variable $ds$ can be employed as the iteration factor:

\[
\begin{align*}
ds &= false \\
if \ node &= 0 \ or \ 1 \ or \ 6 \ or \ 7 \\
then \ ds &= true
\end{align*}
\]
where node is the node index determined at the last level.

The analysis for the orientation is to be developed in the three-dimensional Cartesian coordinate system. As a reference point, the coordinate origin is set at the front bottom left of the lattice concerned. The frame of the lowest level (called level 1, due to the correspondence to the resolution r = 1) is specified as shown in Fig. 2.24. This frame determines the overall appearance of the ordering considered. Analytically, the relation between the coordinates and H-values at level 1 is shown in Table 2.2.

Level 2 of resolution r = 2 is generated by constructing eight units, labeled with \( U_k \) (\( k = 0, 1, 2, 3, 4, 5, 6, 7 \)), from the corresponding indexed nodes in the unit at level 1 (see Fig. 2.25).

The orientation of a unit is specified by two normalized vectors \( m \) and \( n \), orthogonal to each other. Two units with their normal vector are shown in Fig. 2.26. Note that the orientation of \( U_2 \) has been chosen as the principal orientation of the Hilbert order discussed.

The iteration of the orientation from the current level to next level is characterized
Table 2.2: Coordinates and H-Values at Level 1

<table>
<thead>
<tr>
<th>xbit</th>
<th>ybit</th>
<th>zbit</th>
<th>hdigit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 2.25: Structure and Orientation of Level 2
by a three-dimensional rotation matrix. On a specified plane, the positive rotation direction is defined counterclockwise. For instance, on the x-y plane, a rotation with angular shift $\theta$ can be represented by:

$$R(\theta) = \begin{bmatrix}
    \cos(\theta) & -\sin(\theta) & 0 \\
    \sin(\theta) & \cos(\theta) & 0 \\
    0 & 0 & 1
\end{bmatrix}.$$ 

Since there are eight rotation states for the units, eight rotation matrices are needed. These matrices can be indexed according to the node indexes at level 1. Thus, the normal vectors specifying the orientation of the unit $U_k$ at level 2, can be obtained by multiplying matrix $R_{k2}$ to the normal vectors specifying the orientation of the unit at level 1. The first and second subscript specify the final and initial states of a unit in a transition, respectively. The second subscript (2) also indicates the fact that the orientation of unit at level 1 is the same as the unit 2 at level 2. These matrices are:

$$R_{02} = \begin{bmatrix}
    0 & 1 & 0 \\
    -1 & 0 & 0 \\
    0 & 0 & 1
\end{bmatrix},$$

$$R_{12} = \begin{bmatrix}
    0 & 0 & 1 \\
    1 & 0 & 0 \\
    0 & 1 & 0 \\
    -1 & 0 & 0
\end{bmatrix}.$$
\[
R_{22} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix},
\]

\[
R_{32} = \begin{bmatrix}
0 & 0 & -1 \\
-1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix},
\]

\[
R_{42} = \begin{bmatrix}
0 & 0 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix},
\]

\[
R_{52} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix},
\]

\[
R_{62} = \begin{bmatrix}
0 & 0 & -1 \\
0 & 1 & 0 \\
1 & 0 & 0
\end{bmatrix},
\]

\[
R_{72} = \begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}.
\]
The transition from the unit at level 1 to the units at level 2 then can be characterized by means of rotation matrices. For instance, unit $U_7$ is obtained by the following two equations:

$$m_7 = R_{72}m_2$$
$$n_7 = R_{72}n_2$$

where $m_7 = [-1, 0, 0]^T$, $n_7 = [0, 0, 1]^T$, $m_2 = [0, 1, 0]^T$, and $n_2 = [0, 0, 1]^T$.

Although the transition from level 1 to level 2 can be perceived visually, the recursive generating rules have to be found for constructing the higher level frames. The non-commutativity of matrix multiplication will have to be taken into account. Geometrically, this property can be interpreted by the statement that two unit-pairs with the same relative rotation may not be characterized by the same rotation matrix. An example illustrating this situation is shown in Fig. 2.27 and Fig. 2.28.

Figure 2.26: Two Instances with Directional Vectors at Level 2
Figure 2.27: Rotation from $U_2$ to $U_4$

Figure 2.28: Rotation from $U_7$ to $U_f$
It can be shown that the matrix determining the rotation from \( U_2 \) to \( U_4 \) is
\[
R_{42} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.
\]

Now consider another initial state \( U_7 \). The final state \( U_f \) is to be reached such that the relative rotation between \( U_7 \) and \( U_f \) is the same as the one between \( U_2 \) and \( U_4 \). One might expect that the corresponding rotation matrix, \( R_{f7} \), is the same as \( R_4 \). Unfortunately, this is not the case. It can be easily verified that
\[
R_{f7} = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ -1 & 0 & 0 \end{bmatrix}.
\]

In general, the problem can be stated as follows: Given an initial state \( U_a \) and a matrix \( R_{ka} \) that determines the rotation from \( U_a \) to \( U_k \), the same relative rotation from another initial state \( U_b \) to the corresponding final state \( U_k \) needs to be characterized. The solution is a compound matrix. By matrix algebra, one can prove the following equation:
\[
R_{k_b} = R_{ba} R_{ka} R_{ba}^{-1} \tag{2.10}
\]

Eq. (2.10) is crucial to develop a robust iterative algorithm. The encoding algorithm for the three-dimensional Hilbert ordering would have an obscure appearance but for employing Eq. (2.10). There exist, however, some variant formulations of Eq. (2.10). They are mathematically equivalent, but may be of significant difference in operating
efficiency. It is not intended to discuss this topic here, since the main purpose of this section is to present a compact and, more importantly, comprehensible algorithm.

To determine the H-value of a particular point with three integer coordinates \(x, y,\) and \(z,\) a set of tables are introduced (see 2.3). These tables are constructed according to the complete set of units comprising the frame concerned. At level 2, there are seven distinct units. The types of other units at the higher levels are specified by an auxiliary procedure. The verification follows the foregoing iterative rules. It has been found that there are fourteen new units at level 3 and three new ones at level 4. The total of twenty-four types is sufficient to comprise the ordering discussed this section.

An H-value is expressed in terms of octal digits, namely, \(h = 8^{r-1} h_{r-1} + \ldots + 8^1 h_1 + 8^0 h_0.\) At level \(i (i = 1, 2, \ldots, r),\) given \(x_{r-i}\) and \(y_{r-i},\) the \(i-\text{th most significant digit}\) of the coordinates \(x\) and \(y,\) \(h_{r-i},\) the \(i-\text{th most significant digit}\) of the H-value is determined from an appropriate table. The iterated directional vector \(m\) and \(n\) are used as the key to select the table needed. The tables are subscripted as follows. Six numerals 0, 1, 2, 3, 4 and 5 are assigned to the six axis-directions in the three-dimensional Cartesian space, i.e., directions of \(x, -z, y, -x, z,\) and \(-y,\) respectively (Fig. 2.29). For instance, the unit with vectors \(m\) and \(n\) aligning axes \(y\) and \(x\) is associated with the table with index 20.

Then the table corresponding to the unit with its normal vectors aligning the directions with indexes \(j\) and \(k\) is denoted as table \(t_{jk}.\) For instance, table \(t_{24}\) is associated the unit with vectors \(m\) and \(n\) aligning axes \(y\) and \(z,\) respectively. The tables follow: In software the vectors \(m\) and \(n\) are initialized to \([0, 1, 0]\) and \([0, 0, 1]\),
Table 2.3: Tables for Encoding the 3D Hilbert Order

<table>
<thead>
<tr>
<th>$z_{bit}$</th>
<th>$x_{bit}$</th>
<th>$y_{bit}$</th>
<th>$t_01$</th>
<th>$t_02$</th>
<th>$t_04$</th>
<th>$t_05$</th>
<th>$t_{10}$</th>
<th>$t_{12}$</th>
<th>$t_{15}$</th>
<th>$t_{20}$</th>
<th>$t_{21}$</th>
<th>$t_{23}$</th>
<th>$t_{24}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$z_{bit}$</th>
<th>$x_{bit}$</th>
<th>$y_{bit}$</th>
<th>$t_{31}$</th>
<th>$t_{32}$</th>
<th>$t_{34}$</th>
<th>$t_{35}$</th>
<th>$t_{40}$</th>
<th>$t_{42}$</th>
<th>$t_{43}$</th>
<th>$t_{45}$</th>
<th>$t_{50}$</th>
<th>$t_{51}$</th>
<th>$t_{53}$</th>
<th>$t_{54}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

h – digit Tables

h – digits Tables, continued

Table 2.3: Tables for Encoding the 3D Hilbert Order
respectively, according to the orientation specified at level 1.

An example is provided here. Given the resolution $r = 2$, the $H$-value corresponding to the point with coordinates $x = 1, y = 1, z = 2$ is to be determined. First, the integers in base 10 are converted to base 2, namely, $(x, y, z) = (01, 01, 10)$. According to the initial state of vectors $m$ and $n$, the table with index 24 is used. The most significant digits of $x, y$ and $z$ are extracted as key, i.e., $(0, 0, 1)$. Thus from Table 24 the octal digit 3 is obtained. This digit implies that rotation matrix $R_{32}$ needs to be used to update the directional vectors. $m$ and $n$ then become $[0, 0, 1]$ and $[-1, 0, 0]$, respectively. Consequently, the table with index 43 is visited at level 2. Since the key is $(1, 1, 0)$, consisting of the second most significant digit of $(x, y, z)$, the octal digit 0 is retrieved from Table 43. Therefore, the $H$-value in base 10 is $8^1 \times 3 + 8^0 \times 0 = 24$.

The encoding process for $r > 2$ will need to iterate eight rotation matrices at each level. At level $k$, among eight units $U^k_j, j = 0, \ldots, 7$, there is one and only one unit, say, $U^K$, generating eight child-units $U^{k+1}_j, j = 0, \ldots, 7$. The generating scheme is
inherited from that constructing $U^k$ itself and its seven siblings from their parent, a unit at level $k - 1$. It is Eq. 2.10 that characterizes the heredity.

We end this section by indicating that the octree, a data structure widely used in computer graphics, spatial data analysis, and image processing [61], may be regarded as a special case of the hierarchical structure described above, since no rotation is needed at the levels of the octree when generating child-units.

2.4.2 Decoding

The methodology is the same as that used for encoding, i.e., the top-down approach. The decoding process begins at the most significant digit of the given H-value, denoted as $h_{r-1}$, in base 8 system. This digit specifies the octant to which the point belongs at level 1, and determines how to construct eight child-octants from the current octant. Then $h_{r-1}$ is used as the key to extract the most significant digits of $x$, $y$, and $z$ from an appropriate table. Next, the directional vectors are updated by the appropriate rotation matrix determined by $h_{r-1}$. At level 2, the second most significant digit of the H-value, denoted as $h_{r-2}$, is used to extract the second most significant digits of $x$, $y$, and $z$. The visited table is determined by the updated directional vectors. The process is repeated until the least significant digits) of $x$, $y$, and $z$ are reached.

To illustrate the process, consider an example. Given the resolution $r = 2$, the coordinate triple $(x, y, z)$ corresponding to the point with the H-value 38 is to be determined. First, the decimal 38 is converted to octal notation $(h_1 h_0) = 46$. According to the initial states of vector $m$ and $n$, the table with index 24 is visited with $h_1$ as
key. Thus the bit-triple (1,0,1) is obtained. The value of $h_1$ implies that rotation matrix $R_{42}$ should be used to update the directional vectors. $m$ and $n$ then become [0,0,1] and [1,0,0], respectively. Consequently, the table with index 40 is visited at level 2. Since the key is $h_0 = 6$, the bit-triple (0,1,1) is obtained. Therefore, in decimal form, $x = 2^1 \times 1 + 2^0 \times 0 = 2$, $y = 2^1 \times 0 + 2^0 \times 1 = 1$, $z = 2^1 \times 1 + 2^0 \times 1 = 3$.

The tables used for decoding are the same ones used for encoding, except that the roles of the key and the contents are exchanged. For implementation, one may employ a small inverted list or, most simply, introduce two sets of tables. The table size for encoding is $2 \times 2 \times 2$ and for decoding are $8 \times 3$. The issue of software efficiency is not to be investigated here.

In this section, a hierarchical approach to encode and decode the Hilbert ordering is described. The main purpose of this section is to present the iterative property of the three-dimensional Hilbert ordering. Future work will focus on designing efficient algorithms.

### 2.5 Alternative Patterns of the Hilbert Order

The Hilbert order discussed above is constructed by setting the starting and ending points at the bottom left and the bottom right of the lattice, respectively. Alternative geometric patterns can be generated by rotating the square lattice 90°, 180°, or 270° counterclockwise in the Cartesian coordinate system. All of them can be regarded as instances of a topological class with the feature that the ordering starts at a corner and ends at an adjacent corner. The literature concerning the Hilbert order has almost
exclusively referred to this particular class. However, there exist other topological classes. Moore [44] reported such a class. The graphical representations of resolution 1 of Moore’s pattern is the same as shown in Fig. 2.13. The graphical representations of resolution 2, 3 and 4 are depicted in Fig. 2.30 to Fig. 2.32.

Figure 2.30: Moore’s Pattern of Resolution 2

Figure 2.31: Moore’s Pattern of Resolution 3

In this section, we present other four topological classes, named pattern $L1$, $L2$, $L3$, and $L4$, respectively. They have not been found in the literature.

The configurations of these patterns can be analytically described in terms of complex algebra. Let $\mathbb{C}$ be the complex space. Consider the region $\mathbb{C}_+ = \{z | \Re(z) \geq 0\}$,
We will construct the frame of the orders in $C_+$. Given
resolution $r$, consider a square $D$ with its lower-left corner at the origin and with side
$2^r$, partitioned into four quadrants and indexed by 0, 1, 2, and 3, respectively. The
sequence 0, 1, 2, 3 also specifies $L$ patterns of resolution 1 (Fig. 2.33). The reference
path for establishing the configuration of $L$ patterns of resolution 1 in $D$ is the same
as shown in Fig. 2.13.
2.5.1 Configuration of Pattern L1

The generation of the configuration of pattern L1 from resolution 1 to 2 is determined by the following transformations:

\[ T_0z = \frac{1}{2}(1 + j - z) \]
\[ T_1z = \frac{1}{2}(j + z) \]
\[ T_2z = \frac{1}{2}(1 + j + z) \]
\[ T_3z = 1 + \frac{1}{2}(j - z) \]

where \( j \) is the imaginary number \( \sqrt{-1} \) and \( T_k \) is the transformation that maps \( D \) onto its quadrant \( k \) (\( k = 0, 1, 2, 3 \)).

The generation of the configuration from resolution \( k \) to \( k+1 \) (\( k \geq 2 \)) is determined by the following transformations:

\[ T_0z = \frac{1}{2}j\bar{z} \]
\[ T_1z = \frac{1}{2}(j + z) \]
\[ T_2z = \frac{1}{2}(1 + j + z) \]
\[ T_3z = 1 + \frac{1}{2}j(1 - \bar{z}) \]

Figure 2.33: Indexing L Patterns of Resolution 1
The graphical representations of pattern L1 of resolution 2, 3 and 4 are depicted in Fig. 2.34 to Fig. 2.36.

Figure 2.34: Pattern L1 of Resolution 2

Figure 2.35: Pattern L1 of Resolution 3

2.5.2 Configuration of Pattern L2

The generation of the configuration of pattern L2 from resolution 1 to 2 is determined by the following transformations:

\[ T_{0z} = \frac{1}{2}(j + \bar{z}) \]
Figure 2.36: Pattern L1 of Resolution 4

\[
\begin{align*}
T_1z &= \frac{1}{2}[1 + j(1 + z)] \\
T_2z &= \frac{1}{2}[1 + j(2 - z)] \\
T_3z &= \frac{1}{2}(1 + j + \bar{z})
\end{align*}
\]

The transformations of the configuration from resolution \(k\) to \(k + 1\) (\(k \geq 2\)) are the same as for pattern L1.

The graphical representations of pattern L2 for the resolutions 2, 3 and 4 are depicted in Fig. 2.37 to Fig. 2.39.
Figure 2.37: Pattern L2 of Resolution 2

Figure 2.38: Pattern L2 of Resolution 3
Figure 2.39: Pattern L2 of Resolution 4
2.5.3 Configuration of Pattern L3

The generation of the configuration of pattern L3 from resolution 1 to 2 is determined by the following transformations:

\[
\begin{align*}
T_0 z &= \frac{1}{2} j \bar{z} \\
T_1 z &= \frac{1}{2} (j + z) \\
T_2 z &= \frac{1}{2} (1 + j + z) \\
T_3 z &= 1 + \frac{1}{2} (j - z)
\end{align*}
\]

The transformations of the configuration from resolution $k$ to $k + 1$ ($k \geq 2$) are the same as for pattern L1.

The graphical representations of pattern L3 for the resolutions 2, 3 and 4 are depicted in Fig. 2.40 to Fig. 2.42.

![Pattern L3 of Resolution 2](image)

Figure 2.40: Pattern L3 of Resolution 2
Figure 2.41: Pattern L3 of Resolution 3

Figure 2.42: Pattern L3 of Resolution 4
2.5.4 Configuration of Pattern L4

The generation of the configuration of pattern L4 from resolution 1 to 2 is determined by the following transformations:

\[
\begin{align*}
T_0z &= \frac{1}{2}(j + \bar{z}) \\
T_1z &= \frac{1}{2}[1 + j(z + 1)] \\
T_2z &= \frac{1}{2}[1 + j(2 - z)] \\
T_3z &= \frac{1}{2}[1 + j(1 - z)]
\end{align*}
\]

The transformations of the configuration from resolution \( k \) to \( k + 1 \) \((k \geq 2)\) are the same as for pattern L1.

The graphical representations of pattern L4 for the resolutions 2, 3 and 4 are depicted in Fig. 2.43 to Fig. 2.45.

Figure 2.43: Pattern L4 of Resolution 2
Figure 2.44: Pattern L4 of Resolution 3

Figure 2.45: Pattern L4 of Resolution 4
2.6 Summary

Diverse applications of the Hilbert order have been reported in the literature. Many researchers have mentioned the lack of inexpensive encoding and decoding algorithms. In this chapter, a recursive approach is presented first to analyze a typical application. The result matches the one obtained by the algorithmical method suggested in the literature. Our approach is analytical, thus can be applied to other spatial orders. Next, a set of analytical formulas based on Boolean algebra is proposed to encode and decode the two-dimensional Hilbert order. The derivation process is conducted by extensive Karnaugh-map operations. An execution-time assessment suggests that they are faster than the ones published previously. Then, the encoding and decoding algorithms for the three-dimensional Hilbert order are discussed. An approach based on rotational matrices is proposed. Since a large number of Boolean variables is involved, the Karnaugh-map technique is inappropriate to derive irreducible analytic formulas. Other systematic methods like Quine-McCluskey Method ([42], [51], [52]) could be employed, but may lead to complicated mathematical operations. Therefore, it appears that the matrix approach is an acceptable method. Extensive simulations have been carried out on a SPARC station. The rotational-matrix algorithms can also applied to other spatial orders such as the U order and the X order discussed in the following chapter.

The two-dimensional Hilbert order may have alternative patterns. Moore [44], an American mathematician, reported such a pattern in 1900. In this chapter, we proposed other four patterns of the Hilbert order. To the author's best knowledge,
these four patterns have not been reported elsewhere. In a two-dimensional lattice system of resolution 2, there are in total six possibilities of the quadrant-exhaustive and rook-connected patterns. In this sense, the original pattern, the Moore's pattern, and patterns L1 through L4 comprise a complete set of the Hilbert order. The applicability of these alternative patterns may be paradigm-dependent. For example, the Finite-Element-Method (FEM) in electromagnetic field analysis would benefit from a particular pattern, since there are various boundary conditions that may make sense to “wrap around” the finite grid in defining a region. For the issue on computational complexity, it is anticipated that these alternative patterns behave similarly since the structure difference among them lies only in the iteration from resolution 1 to resolution 2. In Chapter 4, these patterns will be compared with each other for some typical applications in spatial data processing systems.

The encoding and decoding processes for n-dimensional Hilbert orders (n ≥ 4) would be considerably complicated and place a heavy computational burden on the user. Therefore, in the next chapter, we propose a new spatial order called the U order.
Chapter 3

The U Order

3.1 Overview

It has been shown that the Hilbert order is fairly sophisticated and that it is a non-trivial task to derive and implement encoding and decoding algorithms. In large data processing systems, the response time for an interactive query is an important performance index. A fast addressing technique to point to positions of data sections is desired. In this chapter, a new spatial order called the $U$ order is proposed. Its major advantage is a significant simplicity of the encoding and decoding operations for multi-dimensional data. We will show that the U order is comparable with the Hilbert order and that it is superior to most other orders by four paradigms, namely, partial match query, range query, eight-connected neighbour-finding, and neighbour proximity evaluation. Then, the encoding and decoding algorithms are discussed. Next, the U order is extended to the $n$-dimensional space. A formal mathematical
description is given. A set of analytical encoding and decoding formulas is also presented. Then, the Kronecker algebraic structures of the two-dimensional U order as well as two other orders are briefly described. Finally, we show that conventional bit-manipulation techniques, such as bit-interleaving, bit-centering, bit-concatenation, and bit-complementation, are not sufficient for encoding most applied spatial orders.

3.2 Metric Analysis of the 2D U Order

In this section, the performance of the U order applied to partial match queries and range queries in a two-dimensional attribute space is analyzed. The expected value of clusters (defined in Section 2.2) is used as the metric to evaluate the performance. The two-dimensional U orders of resolution 1, 2, and 3 are shown in Fig. 3.1, Fig. 3.2, and Fig. 3.3, respectively.

3.2.1 Partial Match Query

In a partial match query, one of the attributes is specified in a two-attribute record while the other is not. Geometrically, the set of the selected records corresponds to a line parallel to one of the coordinate axes in the two-dimensional Cartesian space.
Figure 3.2: The U Order of Resolution 2

Figure 3.3: The U Order of Resolution 3
A graphical representation for the U order is more amenable for a metric analysis. The graphs for the resolutions 1, 2, and 3 are shown in Fig. 3.4 to Fig. 3.6, respectively.

Figure 3.4: Path of the U Order of Resolution 1

Figure 3.5: Path of the U Order of Resolution 2

The results are collected in the following theorems:

**Theorem 3.2.1** The total number of clusters arising from the horizontal type of partial match query in the lattice with resolution \( r \) is given by:

\[
H_r = 2^{2r-1}
\]
[proof] Consider the U orders shown in Fig. 3.4, Fig. 3.5, or Fig. 3.6. A horizontal type of partial match query specifies a row of cells. There are $2^{r-1}$ clusters in each row. Each cluster consists of two cells with contiguous indices. Therefore, the total $2^r$ rows give $H_r = 2^r \times 2^{r-1} = 2^{2r-1}$ clusters. □

**Theorem 3.2.2** The total number of clusters arising from the vertical type of partial match query in the lattice with resolution $r$ is given by:

$$V_r = \frac{2^{2r+1} + 1}{3}$$

[proof] It is observed from Fig. 3.4 that there are three clusters for $r=1$, one with two contiguously indexed cells and two with a single cell each. This leads to $V_1 = 3$, where $V_r$ stands for the total number of clusters by vertical selection, for the resolution $r$. The U order with $r=2$ is constructed by replicating the one with $r=1$. However, there are two clusters being combined into one (Fig. 3.5), hence $V_2 = 4V_1 - 1$. In general, $V_k = 4V_{k-1} - 1$ ($k = 2, 3, \ldots, r$). Consequently, by iteration we have:
The closed form is obtained by setting $k$ to $r - 1$:

$$V_r = 4^k V_{r-k} - \frac{4^k - 1}{3}$$

$$= \frac{2^{2r+1} + 1}{3} \quad \square$$

**Corollary 3.2.1** The expectation of the number of clusters per partial match query is given by:

$$P_r = \frac{7 \times 2^r}{12} + \frac{1}{6 \times 2^r}$$

**[proof]** There are $2^r$ horizontal selections and $2^r$ vertical selections altogether. Therefore, the expectation of the number of clusters per query is:

$$P_r = \frac{H_r + V_r}{2^r + 2^r}$$

$$= \frac{2^{2r-1} + (2^{2r+1} + 1)/3}{2^{r+1}}$$

$$= \frac{7 \times 2^r}{12} + \frac{1}{6 \times 2^r}$$

$$\approx 1.167 \times 2^{r-1} \quad \square$$

The statistic obtained above is compared with those of two other typical spatial orders which are applied in data processing systems, namely, the Z order and Hilbert order [31] (see Table 3.1). It is observed that the U order offers a desirable compromise. In the next section the range query will be analyzed.
<table>
<thead>
<tr>
<th>Spatial Order</th>
<th>The Expectation of Cluster Number Per Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>$1.5 \times 2^{r-1}$</td>
</tr>
<tr>
<td>U</td>
<td>$1.167 \times 2^{r-1}$</td>
</tr>
<tr>
<td>H</td>
<td>$1.0 \times 2^{r-1}$</td>
</tr>
</tbody>
</table>

Table 3.1: Expectation for Partial Match Query

3.2.2 Range Query

The terminology used in the following has been defined in Chapter 2. The properties concerned are presented as a series of theorems.

Theorem 3.2.3

$$N_r^{(1)} = \frac{4^r - 1}{3}$$

[proof] It follows from Fig. 3.4 and Fig. 3.5 that $N_1^{(1)} = 1$ and $N_2^{(1)} = 4N_1^{(1)} + 1 = 5$, respectively. The latter is established due to the fact that, in addition to the four one-cluster templates arising from replicating the U order with resolution 1, a new one-cluster template with contiguous indices 6, 7, 8, and 9 is formed. In general, $N_k^{(1)} = 4N_{k-1}^{(1)} + 1 \ (k = 2, 3, \ldots, r)$. Consequently, by iteration we have:

$$N_r^{(1)} = 4^k N_{r-k}^{(1)} + \frac{4^k - 1}{3} \quad k = 2, 3, \ldots, r - 1$$

The closed form is obtained by setting $k$ to $r - 1$:

$$N_r^{(1)} = 4^{r-1} N_1^{(1)} + \frac{4^{r-1} - 1}{3}$$

70
\[ N_r^{(2)} = \frac{2^{2r-1} + 1}{3} - 2^{r-1} \]

**Theorem 3.2.4**

\[ N_r^{(2)} = \frac{4^r - 1}{3} \]

**[proof]** It follows from Fig. 3.4 and Fig. 3.5 that \( N_1^{(2)} = 0 \) and \( N_2^{(2)} = 1 \), respectively.

In general,

\[ N_k^{(2)} = 4N_{k-1}^{(2)} + b_k \quad (k = 2, 3, \ldots, r) \]

where the first term of the right-hand side derives from the quadruple construction process from level \( k - 1 \) to level \( k \), while the second term stems from the contribution of the additional new templates generated by the cells in the single row or column which is one of the sides of the lower level U order. The subset consisting of all of the new templates is shaped like a *Greek Cross* with width of two cells. A sequence can be established as follows:

\[ b_k = 2^{k-1} - 1 \quad (k = 2, 3, \ldots, r) \]

Consequently, by iteration we have:

\[ N_r^{(2)} = 4^kN_{r-k}^{(2)} + 2^{r-1}(2^k - 1) - \frac{4^k - 1}{3} \quad k = 2, 3, \ldots, r - 1 \]

Then the theorem is proved by setting \( k \) to \( r - 1 \). □

**Theorem 3.2.5**

\[ N_r^{(3)} = \frac{4^r - 1}{3} - 2^{r-1} \]
[proof] It follows from Fig. 3.4 and Fig. 3.5 that \( N_1^{(3)} = 0 \) and \( N_2^{(3)} = 3 \), respectively.

In general,

\[
N_k^{(3)} = 4N_{k-1}^{(3)} + c_k \quad (k = 2, 3, \ldots, r)
\]

To obtain \( c_k \), the Greek Cross region described in Theorem 3.2.4 is investigated again.

A sequence can be derived as follows:

\[
c_k = 2^{k-1} + 1 \quad (k = 2, 3, \ldots, r)
\]

Then the theorem is proved similarly to the proof of Theorem 3.2.4. □

**Theorem 3.2.6**

\[
N_r^{(4)} = \frac{2^{2r-1} + 4}{3} - 2^r
\]

[proof] It follows from Fig. 3.4 and Fig. 3.5 that \( N_1^{(4)} = 0 \) and \( N_2^{(4)} = 0 \), respectively.

In general,

\[
N_k^{(4)} = 4N_{k-1}^{(4)} + d_k \quad (k = 2, 3, \ldots, r)
\]

Once more, the Greek Cross region needs to be considered to deduce \( d_k \). It can be shown that:

\[
d_k = 2^k - 4 \quad (k = 2, 3, \ldots, r)
\]

from which the theorem follows similarly to the proof of Theorem 3.2.4. □

**Corollary 3.2.2** When \( r \) is large, the expected value of clusters per range tends to:

\[
R_r = \frac{7}{3} \approx 2.333
\]
Table 3.2: Expectation for $2 \times 2$ Range Query

[proof] There are $(2^r - 1)^2$ templates in a lattice system with $2^r \times 2^r$ cells. Therefore, the expectation of the number of clusters per range query is:

$$R_r = \lim_{r \to \infty} \frac{1N_r^{(1)} + 2N_r^{(2)} + 3N_r^{(3)} + 4N_r^{(4)}}{(2^r - 1)^2} = \frac{7}{3} \square$$

The statistics obtained above are compared with those of the three other typical spatial orders applied in data processing systems, namely, the Z order, the G order, and the Hilbert order (see Table 3.2). The data of expected values for the Z, G, and Hilbert orders were initially reported in [31]. It is observed that the U order offers a desirable compromise.

### 3.3 Neighbour Finding

In a 2D cellular representation system, the neighborhood of a cell is defined as a subset of cells that are 4-connected or 8-connected adjacent. Finding the neighbors
of a specified cell is an elementary operation in many applications. For instance, to enhance a 2D image with random noise, it is a prerequisite to examine the attributes of the eight neighbors. Another example involves the evaluation of the gradient from the elevation values of the cells in the neighborhood. Given a spatial order, a traversal path is specified in the spatial domain and a sequential order of positions in which the cells reside is determined on the secondary storage. After the transformation, the locality established by a set of cells in the same neighborhood in the high-dimensional spatial domain is no longer preserved: the adjacent cells are scattered in the 1D space.

For a 2D system of resolution \( r \), the scattering can be characterized by the following function:

\[
F = \sum_{i=0}^{4r-1} d(i)
\]

where \( d(i) \) is a metric scaling the linear span of the neighborhood of the cell with index \( i \). Mathematically, any norm-like formulations can be employed for this purpose. For instance, we may have:

\[
d(i) = \sum_{j \in N(i)} |i - j| \overset{\text{def}}{=} d_1(i)
\]

or

\[
d(i) = \text{maximum}_{j \in N(i)} |i - j| \overset{\text{def}}{=} d_{\text{max}}(i)
\]

or

\[
d(i) = \text{maximum}_{j1, j2 \in N(i)} |j1 - j2| \overset{\text{def}}{=} d_s(i)
\]

where \( N(i) \) stands for the neighborhood of the cell with index \( i \) and its configuration is application-dependent. In the following, we choose the 8-connected template and
use both $d_{\text{max}}$ and $d_1$ to evaluate the performance of the spatial orders concerned. For 8-connected applications, the neighborhood is defined as shown in Fig. 3.3, where "C" stands for the centroid, the position in which the current cell resides, and "N" means neighbor cells concerned, respectively.

The metric $d_{\text{max}}$ is the maximum linear span between the centroid and its adjacent cells. In the case of tape systems, the time for retrieving the neighborhood depends directly on $d_{\text{max}}$. In the case of disk systems, $d_{\text{max}}$ is also relevant to the retrieval time, plus the effect of track positions. Thus $d_{\text{max}}$ can be chosen as an appropriate metric to assess the scattering degree. The metric $d_1$ is the sum of the linear spans between the centroid and its adjacent cells.

In real circumstances, the performance of the neighbor-finding paradigm is associated with both the block size in the secondary storage and the capacity of the main memory. However, for purposes of machine-independent analysis, it is reasonable to assume that each cell occupies one block. Then the metric $d_{\text{max}}$ or $d_1$ is measured in block units.

We have designed and implemented a set of algorithms to conduct the performance evaluation for the U order, the Z order, and six patterns of the Hilbert order. The relative merits of these orders are assessed computationally for the 8-connected neighbor-finding problem over the range of resolution $r$ from 2 through 8. The average values of $d_{\text{max}}$ and $d_1$ over the set of $2^r \times 2^r$ cells are numerically reported in Table 3.4 and Table 3.5, respectively. Figure 3.7 and Figure 3.8 also graphically present the results of the Hilbert order (original pattern), the U order, and the Z orders. It is
concluded that the performance of the U order is between the Hilbert order and the Z order.

Two other spatial orders with structures that are not quadrant-exhaustive, namely, the the R (Row) order and the RD (Row-Diagonal) order, have also been evaluated for comparison (Table 3.4 and Table 3.5). The Row order has been shown in Fig. 2.6. The RD order of resolution 2 is illustrated in Fig. 3.9. It appears that, for the 8-connected neighbour finding problems, both the R order and the RD order behave well. However, the R order presents significant asymmetry to horizontal and vertical directions and may introduce large variance to query retrieval applications. The RD order is symmetrical diagonally, but the average number of clusters in a 2 × 2 template tends to 3, which is higher 50% than the Hilbert order. In addition, neither the R order nor the RD order presents satisfactory results when applied to neighbour proximity problems (see Section 3.4).

In the 3D cellular representation system, cells can be adjacent along a face, an edge, or a vertex. Since an cell has eight vertices, twelve edges, and six faces, the neighborhood of an interior cell can be defined as a subset of cells that are 26-connected adjacent. For the cells that are on the boundary of the system, if the wrap-around effect is not an issue, then their neighborhood cardinality is dependent on the particular location of the cell, but the configuration is similar.

A set of algorithms has been designed and implemented to conduct the performance evaluation for the Hilbert order (original pattern), the Z order, and the U order. The relative merits of these orders are assessed computationally for the 26-
Table 3.3: Neighborhood of the 8-connected Template

<table>
<thead>
<tr>
<th>Resolution</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>4.88</td>
<td>8.97</td>
<td>16.99</td>
<td>33.00</td>
<td>65.00</td>
<td>129.00</td>
<td>257.00</td>
</tr>
<tr>
<td>RD</td>
<td>5.88</td>
<td>11.84</td>
<td>22.90</td>
<td>44.44</td>
<td>87.22</td>
<td>172.61</td>
<td>343.30</td>
</tr>
<tr>
<td>Z</td>
<td>5.25</td>
<td>12.25</td>
<td>27.16</td>
<td>57.96</td>
<td>120.70</td>
<td>247.41</td>
<td>501.93</td>
</tr>
<tr>
<td>U</td>
<td>5.50</td>
<td>13.38</td>
<td>30.53</td>
<td>66.29</td>
<td>139.34</td>
<td>287.03</td>
<td>584.40</td>
</tr>
<tr>
<td>H</td>
<td>5.75</td>
<td>14.03</td>
<td>32.14</td>
<td>69.68</td>
<td>146.07</td>
<td>300.17</td>
<td>609.68</td>
</tr>
<tr>
<td>M</td>
<td>6.13</td>
<td>14.00</td>
<td>32.13</td>
<td>69.66</td>
<td>146.06</td>
<td>300.17</td>
<td>609.68</td>
</tr>
<tr>
<td>L4</td>
<td>6.00</td>
<td>14.14</td>
<td>32.27</td>
<td>69.82</td>
<td>146.22</td>
<td>300.33</td>
<td>609.84</td>
</tr>
<tr>
<td>L2</td>
<td>5.88</td>
<td>14.28</td>
<td>32.41</td>
<td>69.98</td>
<td>146.39</td>
<td>300.50</td>
<td>610.01</td>
</tr>
<tr>
<td>L3</td>
<td>6.19</td>
<td>14.88</td>
<td>34.09</td>
<td>73.91</td>
<td>154.87</td>
<td>318.12</td>
<td>645.91</td>
</tr>
<tr>
<td>L1</td>
<td>6.50</td>
<td>15.44</td>
<td>35.72</td>
<td>77.81</td>
<td>163.35</td>
<td>335.74</td>
<td>681.82</td>
</tr>
</tbody>
</table>

Table 3.4: Average Maximum Distance of Finding 8-Neighbors in a 2D Grid

77
<table>
<thead>
<tr>
<th>Resolution</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>RD</td>
<td>16.25</td>
<td>38.06</td>
<td>81.02</td>
<td>166.50</td>
<td>337.25</td>
<td>678.63</td>
<td>1360.13</td>
</tr>
<tr>
<td>Z</td>
<td>15.50</td>
<td>37.50</td>
<td>83.19</td>
<td>176.70</td>
<td>366.13</td>
<td>747.51</td>
<td>1513.61</td>
</tr>
<tr>
<td>R</td>
<td>16.50</td>
<td>40.25</td>
<td>88.13</td>
<td>184.06</td>
<td>376.03</td>
<td>760.02</td>
<td>1526.68</td>
</tr>
<tr>
<td>U</td>
<td>16.50</td>
<td>41.63</td>
<td>94.53</td>
<td>203.32</td>
<td>424.05</td>
<td>868.75</td>
<td>1761.06</td>
</tr>
<tr>
<td>H</td>
<td>16.50</td>
<td>42.38</td>
<td>97.34</td>
<td>210.59</td>
<td>440.40</td>
<td>903.35</td>
<td>1832.55</td>
</tr>
<tr>
<td>M</td>
<td>17.00</td>
<td>43.00</td>
<td>98.00</td>
<td>211.25</td>
<td>441.06</td>
<td>904.02</td>
<td>1833.21</td>
</tr>
<tr>
<td>L4</td>
<td>17.00</td>
<td>43.00</td>
<td>98.00</td>
<td>211.25</td>
<td>441.06</td>
<td>904.02</td>
<td>1833.21</td>
</tr>
<tr>
<td>L2</td>
<td>17.00</td>
<td>43.00</td>
<td>98.00</td>
<td>211.25</td>
<td>441.06</td>
<td>904.02</td>
<td>1833.22</td>
</tr>
<tr>
<td>L3</td>
<td>17.75</td>
<td>45.44</td>
<td>103.86</td>
<td>223.96</td>
<td>467.49</td>
<td>957.87</td>
<td>1941.93</td>
</tr>
<tr>
<td>L1</td>
<td>19.00</td>
<td>48.50</td>
<td>110.38</td>
<td>237.34</td>
<td>494.59</td>
<td>1012.40</td>
<td>2051.34</td>
</tr>
</tbody>
</table>

Table 3.5: Average Total Distance of Finding 8-Neighbors in a 2D Grid
Figure 3.7: Average Maximum Distance of Finding Neighbours in a 2D Lattice

Figure 3.8: Average Total Distance of Finding Neighbours in a 2D Lattice
connected neighbor-finding problem over the range of resolution $r$ from 1 through 6. The average values of $d_{\text{max}}$ and $d_1$ over the set of $2^r \times 2^r \times 2^r$ cells are numerically reported in Table 3.6 and Table 3.7, respectively. It is concluded that the performance of the U order is between the other two.

![Table 3.6: Average Maximum Distance of Finding 26-Neighbors in a 3D Grid](image)

3.4 Neighbour Proximity

The neighbor proximity problem, also called nearest neighbor problem, has been an active research topic in the field of computational geometry. Its solution process
Table 3.7: Average Total Distance of Finding 26-Neighbors in a 3D Grid

<table>
<thead>
<tr>
<th>Resolution</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>21.00</td>
<td>166.38</td>
<td>913.38</td>
<td>4266.69</td>
<td>18444.64</td>
<td>76693.16</td>
</tr>
<tr>
<td>U</td>
<td>21.00</td>
<td>164.50</td>
<td>900.44</td>
<td>4210.77</td>
<td>18224.71</td>
<td>75823.30</td>
</tr>
<tr>
<td>Z</td>
<td>21.00</td>
<td>157.50</td>
<td>848.44</td>
<td>3945.45</td>
<td>17039.35</td>
<td>70839.63</td>
</tr>
</tbody>
</table>

serves as a fundamental operation for core modules in many geographical information systems ([9], [14]). The problem involves searching a subset of cells with the specified attributes such that the distance between the given subset and the current cell is a minimum. In the literature, various metrics have been proposed for defining the distance ([56], [63]). Of them, the City-block metric is one of the most widely used metrics. Based on it, the performance of the Hilbert order and the Z order of resolutions from 1 through 4 was reported in [17], where the authors evaluated the average farthest distance from each cell to a subset defined as the collection of cells such that the difference between their indices and the current cell's index is not greater than a specified threshold. The rationale of this metric is based on the fact that less cells need to be visited to find the nearest neighbors if the cells with similar indices are closer each other in the cellular system.

We have designed and implemented a set of algorithms to evaluate the performance for the U order, Z order, and six patterns of the Hilbert order. Similarly to the model described in [17], $2^{r-1}$ and the number in the particular order is chosen as the threshold and the index, respectively. The results for two dimensions over the set
of $2^r \times 2^r$ cells ($r = 1, 2, \ldots, 7$) are numerically reported in Table 3.8. Figure 3.10 also graphically presents the results of the Hilbert (original pattern), the U order, and the Z orders. The results for three dimensions over the set of $2^r \times 2^r \times 2^r$ cells ($r = 1, 2, \ldots, 6$) are numerically reported in Table 3.9 and graphically presented in Figure 3.11, respectively. It is clear that the U order behaves better than the Z order.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>1.00</td>
<td>2.00</td>
<td>3.25</td>
<td>4.88</td>
<td>7.54</td>
<td>10.82</td>
<td>16.04</td>
</tr>
<tr>
<td>L2</td>
<td>1.00</td>
<td>2.00</td>
<td>3.25</td>
<td>4.88</td>
<td>7.54</td>
<td>10.81</td>
<td>16.04</td>
</tr>
<tr>
<td>L3</td>
<td>1.00</td>
<td>2.00</td>
<td>3.27</td>
<td>4.89</td>
<td>7.53</td>
<td>10.81</td>
<td>16.04</td>
</tr>
<tr>
<td>L4</td>
<td>1.00</td>
<td>2.00</td>
<td>3.27</td>
<td>4.89</td>
<td>7.53</td>
<td>10.81</td>
<td>16.04</td>
</tr>
<tr>
<td>H</td>
<td>1.00</td>
<td>2.00</td>
<td>3.28</td>
<td>4.89</td>
<td>7.53</td>
<td>10.81</td>
<td>16.05</td>
</tr>
<tr>
<td>M</td>
<td>1.00</td>
<td>2.00</td>
<td>3.28</td>
<td>4.90</td>
<td>7.53</td>
<td>10.81</td>
<td>16.05</td>
</tr>
<tr>
<td>U</td>
<td>1.00</td>
<td>2.63</td>
<td>4.06</td>
<td>7.28</td>
<td>10.76</td>
<td>17.25</td>
<td>24.26</td>
</tr>
<tr>
<td>Z</td>
<td>1.50</td>
<td>2.75</td>
<td>4.84</td>
<td>7.91</td>
<td>12.15</td>
<td>18.48</td>
<td>27.01</td>
</tr>
<tr>
<td>R</td>
<td>1.50</td>
<td>3.13</td>
<td>6.19</td>
<td>12.22</td>
<td>24.23</td>
<td>48.24</td>
<td>96.25</td>
</tr>
<tr>
<td>RD</td>
<td>1.50</td>
<td>3.38</td>
<td>7.41</td>
<td>15.41</td>
<td>31.42</td>
<td>63.44</td>
<td>127.45</td>
</tr>
</tbody>
</table>

Table 3.8: Average Farthest Distance of the Neighbors in a 2D Grid

We have shown that the U order exhibits satisfactory characteristics for several paradigms frequently arising from applications of spatial data processing. Thus the
Figure 3.10: Average Farthest Distance of the Neighbours in a 2D Lattice

Figure 3.11: Average Farthest Distance of the Neighbours in a 3D Lattice
3.5 Encoding and Decoding

The main purpose of this section is to describe the iteration rules applied to construct the U order. These rules are important since they establish the basis for deriving the analytic encoding formula and designing efficient algorithms.

The U ordering is to be established in a subset $D = \{0 \leq x < 2^r, 0 \leq y < 2^r\}$ in the two-dimensional integer space.

Similar to the Hilbert order, the analysis for the U order is based on the top-down...
approach. First, the whole lattice is partitioned into four quadrants and indexed in base 4. Then each quadrant is recursively processed identically. The first and second levels are illustrated in Fig. 3.12 and Fig. 3.13, respectively.

\[
\begin{array}{cc}
3 & 2 \\
0 & 1 \\
\end{array}
\]

Figure 3.12: The First Level

\[
\begin{array}{cccc}
33 & 32 & 23 & 22 \\
30 & 31 & 20 & 21 \\
03 & 02 & 13 & 12 \\
00 & 01 & 10 & 11 \\
\end{array}
\]

Figure 3.13: The Second Level

The units of the U order can be defined similarly as the Hilbert order (see Section 2.3), but with simpler features. Instead of multiform states, now both the orientation and the aspect of a unit are invariable. For the specified U order considered in this section, the “mouth” of a unit is oriented to the negative x axis and the indexing order of the nodes is counterclockwise in the two dimensional Cartesian coordinate system (see Fig. 3.4 to Fig. 3.6).

According to the inherent property of the U order, the recursive construction process does not involve any rotations. Consequently, the encoding algorithm designed
for the Hilbert order is applicable to the U order by deactivating the switching tests and the iteration of rotations. Indubitably, this simplicity considerably reduces the encoding time.

The encoding algorithm is implemented similarly to that of the Hilbert order. A series of execution-time assessments for various resolutions has been conducted. The performance of the encoding algorithm has been compared with that of the Hilbert order; the results are depicted in Fig. 3.14.

The discussion above is also applicable to the decoding process. The performance comparison between the U order and the Hilbert order is presented in Fig. 3.15.
3.6 The Analytic Formulas for the 2D U Order

In this section, we derive the analytical encoding and decoding formulas for the two-dimensional U order. The discussion is related to the U order shown in Fig. 3.1 to Fig. 3.6.

The relation defined between the u-value and the \((x, y)\) values in the U order can be expressed in a tabular form in terms of bits. Consider the case of \(r = 2\) first, then \(x = 2^1x_1 + 2^0x_0, \ y = 2^1y_1 + 2^0y_0, \) and \(u = 2^1u_1 + 2^0u_0.\) The following table is associated to Fig. 3.2.
With Boolean algebra, a set of analytic formulas can be derived from the relation defined in the table:

\[ u_3 = y_1 \]
\[ u_2 = x_1 \oplus y_1 \]
\[ u_1 = y_0 \]
\[ u_0 = x_0 \oplus y_0 \]

It follows the hierarchical structure of the U order that the u-values with resolution \( r \) are determined by

\[ u_{2k+1} = y_k \quad (3.1) \]
\[ u_{2k} = x_k \oplus y_k \quad (3.2) \]

\[ k = 0, 1, \ldots, r - 1. \]

The order of this algorithm is \( 5r \).
Equation (3.1) and (3.2) can be implemented conveniently in modern computer languages. For instance, one can write the following expressions in C:

\[ u_{odd} = y \]
\[ u_{even} = x \land y \]

where \( u_{odd} \) and \( u_{even} \) are the words consisting of the bits at odd and even positions in the expression of a u-value in base 2, respectively. The final step is to interleave the two machine words \( u_{odd} \) and \( u_{even} \) to generate a single word \( u \). One of the simplest approaches is to use a loop structure. However, the interleaving process can considerably sped up by employing well-designed data structures. Schrack [62] has designed a set of sophisticated interleaving algorithms and the numerical simulations demonstrate remarkable performance.

The methodology for deriving the analytic decoding formula is the same as that used for encoding. Decoding is the inverse operation of encoding, namely, given the u-value of a particular point, the corresponding coordinate pair \((x,y)\) is to be determined. For a given resolution \( r \), the decoding formulas become

\[ x_k = u_{2k+1} \]  \hspace{1cm} (3.3)
\[ y_k = u_{2k} \oplus u_{2k+1} \]  \hspace{1cm} (3.4)

\[ k = 0, 1, \ldots, r - 1. \]

The order of this algorithm is also \( 5r \).

Just as the Delta Modulation (DM) is comparable with the Pulse Code Modulation (PCM) in digital communication, the principal advantage of the U order over the Hilbert order is the simplicity of its implementation. Later we will show that this
simplicity holds in n-dimensional space and it is in the form of a prefix calculation, a typical structure widely applied in parallel processing.

3.7 Extension to Three Dimensions

The significance of the two-dimensional U order is suggested by the satisfactory performance revealed in the foregoing section. The discussion presented in this section extends the analysis from two dimensions to three dimensions. The encoding and decoding algorithms are developed parallelly to those for the Hilbert order.

The U order is to be established in a subset $D = \{0 \leq x < 2^r, 0 \leq y < 2^r, 0 \leq z < 2^r\}$ in the three-dimensional integer space. Fig. 3.16 shows the U order of resolution 2.

The terminology to be used below has been defined in Section 2.3. A unit of the three-dimensional U order can be represented by a set of $2 \times 2 \times 2$ cells, or by a simple graph with seven edges and eight nodes, as shown in Fig. 3.17. The latter will be used in this section, since it appears more convenient to present the order’s hierarchical features.

The analysis for the orientation is to be developed in the three-dimensional Cartesian coordinate system. As a reference point, the coordinate origin is set at the front bottom left of the lattice concerned. The frame of the highest level (called level 1, at resolution $r = 1$) is specified as shown in Fig. 3.17. This frame determines the overall appearance of the ordering considered. Analytically, the relation between the coordinates and u-values at level 1 is shown in Table 3.10.
Figure 3.16: The Three-Dimensional U Order of Resolution 2

Figure 3.17: Structure and Orientation of Level 1
Table 3.10: Coordinates and u-values at Level 1

<table>
<thead>
<tr>
<th>xbit</th>
<th>ybit</th>
<th>zbit</th>
<th>udigit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

Level 2 of resolution $r = 2$ is generated by constructing eight units, labeled $U_k$ ($k = 0, 1, 2, 3, 4, 5, 6, 7$); from the corresponding indexed nodes in the unit at level 1 (see Fig. 3.18).

Again, according to the inherent property of the U order, the recursive construction process does not involve rotations. Therefore, the encoding algorithm designed for the Hilbert order is applicable to the U order by deactivating the switching tests and the iteration of rotation matrices. This simplicity considerably reduces the encoding and decoding time.

The encoding algorithm is implemented corresponding to the Hilbert order. A series of execution-time assessments for various resolutions has been conducted. The performance of the encoding algorithm has been compared with that of the Hilbert order, the results are shown in Fig. 3.19.

The discussion above applies as well to the decoding process. The performance
Figure 3.18: Structure and Orientation of Level 2
comparison between the U order and the Hilbert order is presented in Fig. 3.20.

3.8 The Analytic Formulas for the 3D U Order

Basically the encoding and decoding processes discussed in the preceding section can be regarded as a look-up-table approach. However, unlike the Hilbert order, a single look-up-table is sufficient for a specified U order. The following discussion is related to the U orders shown in Fig. 3.17 to Fig. 3.18.

The analytic formulas can be derived much as for the case of two dimensions. First, the relation defined between the u-value and the \((x, y, z)\) values in the U order is expressed in a tabular form in terms of bits. Then employing Boolean algebra,
Figure 3.20: Execution-Time Assessments for Decoding 3D Orders

the analytic formulas are derived from the table. To save space, only the result is presented here without listing the derivation details:

\[ u_{3k+2} = x_k \]  
\[ u_{3k+1} = x_k \oplus z_k \]  
\[ u_{3k} = x_k \oplus y_k \oplus z_k \]  

\[ k = 0, 1, \ldots, r - 1 \]

Equations 3.6 and 3.7 can be conveniently implemented in the C language. For instance,

\[ u_a = x \gg y \gg z \]
\[ u_b = x \gg z \]
\[ u_c = x \]

where \( u_a \) and \( u_b \) are the machine words consisting of the bits at the \( 3k \)-th and the
(3k + 1)-th digit positions (k = 0, 1, ..., r − 1), respectively. The final step is to interleave three words u_a, u_b and u_c to generate a single word u.

The methodology for deriving the analytic decoding formula is the same as that used for encoding. Decoding is the inverse operation of encoding, namely, given the u-value of a particular point, the corresponding coordinate triple (x, y, z) is to be determined. Given resolution r, the decoding formulas are written as follows:

$$x_k = u_{3k+2}$$  \hspace{1cm} (3.8)

$$y_k = u_{3k} \oplus u_{3k+1}$$  \hspace{1cm} (3.9)

$$z_k = u_{3k+1} \oplus u_{3k+2}$$  \hspace{1cm} (3.10)

$$k = 0, 1, \ldots, r - 1$$

It is the simplicity inherent in the structure and the developed efficient interleaving algorithms that make the U order competitive with other spatial orders, especially for the high-dimensional cases.

### 3.9 The n-dimensional U Order

The high-dimensional lattice system is a fundamental model in modern computer sciences, especially in database systems. A record with n attributes can be mapped to a point in n-dimensional integer space $Z^n$. Thus a lattice can be defined to be the rectangular region that contains a subset of such points. The cardinality of the i-th attribute is the resolution of the i-th dimension of the lattice (i = 1, ..., n).

The significance of the high-dimensional U order is suggested by the satisfied per-
formance revealed in the proceeding sections. In this section, a formal mathematical
definition is given to the $n$-dimensional U order. Then both the encoding and the
decoding formulas are presented.

3.9.1 Definition

Some terminology needs to be introduced before presenting a mathematical defini­
tion for the high-dimensional U order. These concepts can be found in most college
textbooks on graph theory (e.g., [27]).

A graph $G$ is a set $(V, E)$, where $V$ is a finite set and $E$ is a binary relation on
$V$. If it is needed from the context, the sets $V$ and $E$ are denoted $V(G)$ and $E(G)$,
respectively, to stress the correspondence to a particular graph $G$. The elements of
$V$ are called nodes. The cardinality of a node is not necessarily unity. A node whose
cardinality is more than one is called a complex node, while when its cardinality is one
it is called a unitary node. The elements of $E$ are called edges. The product $G_1 \times G_2$
of two graphs $G_1$ and $G_2$ is the graph with node set $V_1 \times V_2$, where the cross means
Cartesian product, such that two nodes $(a_1, a_2)$ and $(b_1, b_2)$ of $G_1 \times G_2$ are adjacent if
and only if either $a_1 = b_1$ and $a_2 b_2 \in E(G_2)$ or $a_2 = b_2$ and $a_1 b_1 \in E(G_1)$. A complete
graph is a graph in which there exists an edge for every pair of nodes. The complete
graph with two nodes is denoted $K_2$.

The $n$-dimensional hypercube $Q^n$ is defined recursively as follows:

$$Q^1 = K_2$$

$$Q^n = Q^1 \times Q^{n-1}$$
Thus $Q^n$ is a graph $G = (V, E)$ with $2^n$ nodes and $n2^{n-1}$ edges. $Q^n$ can be labeled or unlabeled. In the former case, each vertex is labeled with a bit string $(b_1b_2\ldots b_n)$, where $b_i \in \{0, 1\}$. Two vertices of $Q^n$ are adjacent if and only if their labels differ in exactly one bit. A Hamiltonian path is a path passing through every node exactly once. Fig. 3.21 shows a four-dimensional hypercube.

Let $Q^n(V)$ be the $n$-dimensional cube with the elements in the set $V$ as its nodes, then the Hamiltonian path in $Q^n(V)$ is denoted $H(Q^n, V)$. Let $V_1$ be the set of unitary nodes. With the concepts introduced above, the U order can be recursively defined as follows:

**Definition 3.9.1** The $n$-dimensional U order of resolution 1 is the order determined by sequentially indexing the nodes of $H(Q^n, V_1)$;

Let $V_r = \{H_j(Q^n, V_{r-1}), j = 1, 2, \ldots, 2^n\}$. Then the $n$-dimensional U order of resolution $r$ is the order determined by sequentially indexing the nodes of $H(Q^n, V_r)$ and
incorporating the orders defined for resolution $r - 1$.

As an instance, the four-dimensional U order of resolution 1 and $H(Q^4, V_1)$ is shown in Fig. 3.22.

Figure 3.22: The Four-Dimensional U order of Resolution 1

One of the interesting properties of high-dimensional U orders is the anti-Gray property. By anti-Gray it is meant that the large values of the Hamming distance between adjacent lines in a code are obtained. The following anti-Gray code was
proposed by Hamming [26]:

<table>
<thead>
<tr>
<th>$g_3$</th>
<th>$g_2$</th>
<th>$g_1$</th>
<th>$g_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Let $d(k) = \text{row}(k) \oplus \text{row}(k + 1) \ (k = 1, 2, \ldots, 16 \ (mod\ 16))$, then the anti-Gray code gives: $\sum_{k=1}^{16} d_k = 56$.

On the other hand, an instance of the four-dimensional U order can be written as
follows:

<table>
<thead>
<tr>
<th></th>
<th>u₃</th>
<th>u₂</th>
<th>u₁</th>
<th>u₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

It gives \( \sum_{k=1}^{16} d_k = 54 \).

The anti-Gray type codes can be applied to design multiple disk allocation for partial match queries ([1], [16]).

### 3.9.2 Encoding and Decoding

In this section, the analytic encoding and decoding formulas of the \( n \)-dimensional U order are presented. For encoding, both serial and parallel versions are given.

Consider a point \( p \) in the \( n \)-dimensional integer space. Let the coordinates of \( p \) be \((x_1, x_2, \ldots, x_n)\). The bits of \( x_j \) \((j = 1, 2, \ldots, n)\) are expressed as \((x_{j,r-1} x_{j,r-2} \ldots x_{j,0})\),
where \( r \) is the specified resolution. The bits of \( u \)-values are expressed as \( (u_{n,r-1} u_{n,r-2} \ldots u_{n,0}) \).

Then the serial encoding formula of the U order can be written as follows:

\[
\begin{align*}
    u_{n,k+n-1} &= x_{1,k} \\
    u_{n,k+n-j} &= u_{n,k+n-j+1} \oplus x_{j,k} \\
    j &= 2, 3, \ldots, n, \quad k = 0, 1, \ldots, r - 1
\end{align*}
\]

The parallel encoding formula can be expressed as follows:

\[
\begin{align*}
    u_{n,k+n-1} &= x_{1,k} \\
    u_{n,k+n-j} &= x_{1,k} \oplus x_{2,k} \ldots \oplus x_{j,k} \\
    j &= 2, 3, \ldots, n, \quad k = 0, 1, \ldots, r - 1
\end{align*}
\]

The decoding formula can be expressed as follows:

\[
\begin{align*}
    x_{n,k} &= u_{n-1+k,n} \\
    x_{n-j,k} &= u_{n-j+k,n} \oplus u_{n-j-1+k,n} \\
    j &= 1, 2, \ldots, n - 1, \quad k = 0, 1, \ldots, r - 1
\end{align*}
\]

It is important that the presented encoding formula of the U order is in the form of a *prefix calculation*, a typical structure widely applied in parallel processing [11]. The complexity of many primitive tasks, such as binary addition, list ranking, etc., can be effectively reduced by introducing the prefix calculation. We also showed the "anti-Gray" property of the U order. All of these characteristics suggest that the potential of the U order is not limited to the application of spatial data processing only.
3.10 Kronecker Algebraic Structures of 2D Spatial Orders

3.10.1 The Z Order

The Z order has been used widely in image processing and computer graphics because it corresponds to the quadtree structure. The location code of the Z order for \( r > 2 \) can be simply constructed by interleaving the bits of the \( y \) coordinate with the bits of the \( x \) coordinates:

\[
N_Z^{(3)} = w^T b
\]

where

\[
w = [2^{2r-1}, 2^{2r-2}, \ldots, 2^1, 2^0]^T
\]

\[
b = [y_{r-1}, x_{r-1}, \ldots, y_0, x_0]^T
\]

Fig. 3.23 shows the Z order for \( r = 3 \).

In this section, we propose an alternative formulation by using Kronecker algebra ([25]). Define

\[
Z_1 = \begin{bmatrix}
2 & 3 \\
0 & 1
\end{bmatrix}
\]

\[
I_1 = \begin{bmatrix}
1 & 1 \\
1 & 1
\end{bmatrix}
\]
Then the Kronecker Products of $Z_1$ and $I_1$, and of $I_1$ and $Z_1$ are

$$M = Z_1 \otimes I_1 = \begin{bmatrix} 2 & 2 & 3 & 3 \\ 2 & 2 & 3 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$N = I_1 \otimes Z_1 = \begin{bmatrix} 2 & 3 & 2 & 3 \\ 0 & 1 & 0 & 1 \\ 2 & 3 & 2 & 3 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$
The matrix of the Z order for \( r = 2 \) can be obtained by concatenating \( M \) and \( N \) entry by entry:

\[
Z_2 = M \otimes N = \begin{bmatrix}
22 & 23 & 32 & 33 \\
20 & 21 & 30 & 31 \\
02 & 03 & 12 & 13 \\
00 & 01 & 10 & 11 & \text{(base4)} \\
10 & 11 & 14 & 15 \\
08 & 09 & 12 & 13 \\
02 & 03 & 06 & 07 & \text{(base10)} \\
00 & 01 & 04 & 05
\end{bmatrix}
\]

It can be shown that

\[
Z_3 = (Z_2 \otimes I_1) \otimes (I_2 \otimes Z_1)
\]

The Z order of higher resolution can be constructed recursively.

### 3.10.2 The U Order

In this section, an alternative construction method based on the Kronecker algebra for the two-dimensional U order is presented to demonstrate its recursive property.

The U order for \( r = 1 \) and \( r = 2 \) can be expressed by the matrices:

\[
U_1 = \begin{bmatrix}
3 & 2 \\
0 & 1
\end{bmatrix}
\]
It can be shown that

\[ U_2 = (U_1 \otimes I_1) \otimes (I_1 \otimes U_1) \]

\( U_r (r > 2) \) can be constructed recursively.

3.10.3 The X Order

The Z order corresponds to the quadtree structure. It is recursive, or, equivalently, exhausts a quadrant or sub-quadrant of a square before exiting it. However, besides the Z order, the U order and the X order can also represent the quadtree structure. More importantly, only these three orders can represent quad-tree structures in the sense of topology. The former has been discussed. The latter is briefly discussed in this section. The X orders of resolution 1 and 2 are shown in Fig. 3.24 and Fig. 3.25, respectively.

Figure 3.24: The X Order of Resolution 1
The X order for $r = 1$ and $r = 2$ can be expressed by the matrices:

$$X_1 = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}$$

$$X_2 = \begin{bmatrix} 10 & 9 & 6 & 5 \\ 8 & 11 & 4 & 7 \\ 2 & 1 & 14 & 13 \\ 0 & 3 & 12 & 15 \end{bmatrix}$$

It can be shown that

$$X_2 = (X_1 \otimes I_1) \odot (I_1 \otimes X_1)$$

$X_r (r > 2)$ can be constructed recursively in this fashion.

It is interesting to define the dual of the X order as:

$$\tilde{X}_2 = (I_1 \otimes X_1) \odot (X_1 \otimes I_1)$$
$\hat{X}_2$ happens to represent a typical spatial order applied to digital halftoning. In digital halftoning, three approaches are well known — ordered dithering [32], dot diffusion [36], and error diffusion [21]. Among them, the first two approaches involve using spatial orders, which will be called the B order and the K order.

The B order for $r = 1$ and $r = 2$ can be represented by the matrices [32]:

$$N_1^{(2)} = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

$$N_2^{(2)} = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$
In terms of the Kronecker operators, it can be shown that

\[ N_2^{(2)} = (I_1 \otimes N_1^{(2)}) \otimes (N_1^{(2)} \otimes I_1) \]

Comparing this formulation with that of \( \hat{X}_2 \) (Section 3.10.3), we find that \( N_2^{(2)} \) happens to be the clockwise rotated version of \( \hat{X}_2 \)! This observation enhances the belief that there is an inherent relation between the X order and the B order.

The analytical encoding and decoding formulas for the \( n \)-dimensional X order can be constructed similarly to those for the U order. The details will not be discussed further.

### 3.11 Discussion on Conventional Bit-Manipulation Techniques

In this section, we will show that bit-interleaving, bit-centering, bit-concatenating, and bit-complementing are not sufficient for encoding the Hilbert, U, and X orders.

*Bit-interleaving* is a method to map a spatial point in the Cartesian coordinate system into a one-dimensional sequence. Usually, the desired one-dimensional sequence consists of non-negative integers, no matter what number base is used. The elements in this sequence are called *location codes*.

Consider a point \((x, y)\) in the two-dimensional Cartesian coordinate system. Its coordinates can be expressed with base 2 as \( x_{r-1}x_{r-2} \ldots x_1x_0 \) and \( y_{r-1}y_{r-2} \ldots y_1y_0 \), respectively, where \( r \) is the resolution of the image, \( x_i, y_i \in \{0, 1\} \), and \( i = 0, 1, \ldots, r-1 \). By inserting the bit sequence \( y_i \) into \( x_i \), we have \( y_{r-1}y_{r-1} \ldots y_0x_0 \), called *1-interleaving*,...
or \( y_{r-1}y_{r-2}x_{r-1}x_{r-2} \ldots y_1y_0x_1x_0 \), called \( 2\)-interleaving, etc. In the following, 1-interleaving will be simply called “interleaving” since it is the most popular technique to construct location codes.

It is not possible to derive formulas to generate the location codes of the H, U, and X orders by using the interleaving technique only. Even other bit-manipulating techniques, such as centering, concatenation or complementation, are not sufficient for this purpose.

Consider the case of \( r = 2 \) to introduce the concepts of new bit-manipulating techniques. Given a spatial point \((x, y)\) in the two-dimensional Cartesian coordinate system, a four bit set, viz., \((x_1 x_0, y_1 y_0)\) needs to be considered. The total number of their permutations is 4! or 24. The 24 instances can be partitioned to form three classes, which will be named the **interleaving class**, the **concatenating class** and the **centering class**.

The three classes and their instances are shown below:

**Interleaving:**

\[
\begin{align*}
&y_1x_1y_0x_0 & x_1y_1x_0y_0 \\
&y_1x_0y_0x_1 & x_1y_0x_0y_1 \\
&y_0x_1y_1x_0 & x_0y_1x_1y_0 \\
&y_0x_0y_1x_1 & x_0y_0x_1y_1
\end{align*}
\]

**Concatenating:**

\[
\begin{align*}
&y_1y_0x_1x_0 & x_1x_0y_1y_0 \\
&y_1y_0x_0x_1 & x_1x_0y_0y_1 \\
&y_0y_1x_1x_0 & x_0x_1y_1y_0 \\
&y_0y_1x_0x_1 & x_0x_1y_0y_1
\end{align*}
\]
By complementing one or more bits, more spatial orders can be constructed. Since we are considering the four-bit set \( x_1, x_0, y_1, y_0 \), there are \( 2^4 \) possible bit-complement operation for each instance out of the \( 4! \) permutations. Consequently, the total number of spatial orders is \( 4! \times 2^4 = 384 \). It is interesting that the property discussed above is just the one of the hyperoctahedral group \( O_n \) of order \( 2^n n! \) \([23]\) for \( n = 4 \). The typical symmetry operation of \( O_n \) consists of one of the \( n! \) possible permutations of the coordinates of the \( n \)-tuple, followed by one of the \( 2^n \) possible complementations.

Note that so far we have implicitly used the weight set \( \{ 2^3, 2^2, 2^1, 2^0 \} \). This is just one instance of an integer subset \( \{ w_3, w_2, w_1, w_0 \} \). In general, we have:

**Definition 3.11.1** For resolution \( r = 2 \), an M-code is the location code formulated as the combination of bits of the \( x \)-coordinate and \( y \)-coordinate:

\[
N_{base10}^{(3)} = w_3 b_3 + w_2 b_2 + w_1 b_1 + w_0 b_0
\]

where

\[
b_i \in \{ x_1, x_0, y_1, y_0, \overline{x_1}, \overline{x_0}, \overline{y_1}, \overline{y_0} \}
\]

\[
x_i, y_i \in \{ 0, 1 \}
\]

and the bit set \( b_3, b_2, b_1, b_0 \) is one instance of 384 permutations.

Below, we will show that the M-code system can not generate the H, U, and X orders.

Some concepts in linear algebra need to be introduced before investigating the M-code. The details can be found in text books on linear algebra, e. g., [4].
A system \( L \) of linear equations is \textit{inconsistent} if it has no solutions, and it is \textit{consistent} if there is at least one solution. \( L \) is \textit{overdetermined} if the number of its equations is greater than the number of unknowns.

\textbf{Lemma 3.11.1} A system of linear equations \( Aw = b \) is consistent if and only if the rank of the matrix \( A \) equals the rank of the augmented matrix \([A|b]\).

We begin the analysis with the following definition:

\textbf{Definition 3.11.2} Given a spatial order corresponding to an image with resolution \( r \), its \textit{characteristic matrix} is a \( 4^r \times 2^r \) 0-1 Boolean matrix, with the ordinal numbers expressed in bits as its rows, and the order of such rows is determined by the row scanning order from the bottom of this spatial order.

By this definition, the \textit{characteristic matrix} of the Z order with \( r = 2 \) is:

\[
N_Z^{(1)} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]
It is convenient to formulate the M-code in the form of linear equation systems. For example, the Z order bit-interleaving formulation can be written as:

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
8 \\
4 \\
2 \\
1
\end{bmatrix} = \begin{bmatrix}
0 \\
1 \\
4 \\
5 \\
2 \\
3 \\
6 \\
7 \\
8 \\
9 \\
12 \\
13 \\
10 \\
11 \\
14 \\
15
\end{bmatrix}
\]

The system shown above may be represented by the matrix equation \(Aw = b\), where \(A\) is the characteristic matrix and \(b\) is the right hand side vector. Note that in this case, the system \(Aw = b\) is over determined, although not necessarily consistent. An analysis of the solution existence follows. For simplicity, it is carried out for the \(r = 2\)
case. However, the methodology and conclusion can be generalized to arbitrary values of $r$ by induction without difficulty.

In the case of $r = 2$, the 384 different bit-manipulating structures can be partitioned to form 16 classes such that each class consists of 4! instances. For each class, the 4! instances are just permutations of columns of $A$ and such permutations do not change the rank of $A$ or $[A|b]$. For this reason, only one instance needs to be analyzed from among the 4! ones. Picking one instance from each of the 16 classes, we obtain:

\[
\begin{align*}
&y_1x_1y_0x_0 \ y_1x_1y_0\bar{x}_0 \ y_1x_1\bar{y}_0x_0 \ y_1x_1\bar{y}_0\bar{x}_0 \\
&y_1\bar{x}_1y_0x_0 \ y_1\bar{x}_1y_0\bar{x}_0 \ y_1\bar{x}_1\bar{y}_0x_0 \ y_1\bar{x}_1\bar{y}_0\bar{x}_0 \\
&\bar{y}_1x_1y_0x_0 \ \bar{y}_1x_1y_0\bar{x}_0 \ \bar{y}_1x_1\bar{y}_0x_0 \ \bar{y}_1x_1\bar{y}_0\bar{x}_0 \\
&\bar{y}_1\bar{x}_1y_0x_0 \ \bar{y}_1\bar{x}_1y_0\bar{x}_0 \ \bar{y}_1\bar{x}_1\bar{y}_0x_0 \ \bar{y}_1\bar{x}_1\bar{y}_0\bar{x}_0
\end{align*}
\]

We can write 16 characteristic matrices, denoted as $N_1^{(1)}, N_2^{(1)}, \ldots, N_{16}^{(1)}$, respectively. For example, $y_1\bar{x}_1\bar{y}_0x_0$ generates $N_7^{(1)}$ as follows:

\[
N_7^{(1)} = \begin{bmatrix}
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 \\
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1
\end{bmatrix}
\]
Let the right-hand side vectors corresponding to the H, U, X orders be $b_H, b_U, b_X$, respectively, then we have: 

1: 

$$b_H = [0, 1, 14, 15, 3, 2, 13, 12, 4, 7, 8, 11, 5, 6, 9, 10]^T$$

$$b_U = [0, 1, 4, 5, 3, 2, 7, 6, 12, 13, 8, 9, 15, 14, 11, 10]^T$$

$$b_X = [0, 3, 12, 15, 2, 1, 14, 13, 8, 11, 4, 7, 10, 9, 6, 5]^T$$

where the superscript "T" denotes transposition.

It is easy to find that the rank of $N_i^{(1)}$ is 4 while the rank of $[N_i^{(1)}|b_t]$ is 5, where $i = 1, 2, \ldots, 16, t \in \{H, U, X\}$. Therefore, it follows from Lemma 3.11.1 that there exist no solutions of the weight vector $w$. Consequently, no M-code exists for the H, U, and X orders.

The significance of above analysis is that it demonstrates that there does not exist any weight vector $w$ such that $w$ is a solution. Particularly, $w = [8, 4, 2, 1]^T$ or a permutation can not be a solution. This implies that bit-interleaving, bit-centering, bit-concatenation, and bit-complementation are not sufficient for encoding the Hilbert, U, and X orders. Therefore, other mathematical systems have to be used to derive the analytical formulas for this purpose, as we discussed in the foregoing sections.

The analysis for the decoding process of the H, U, and X orders is parallel to the analysis presented in this section. The conclusion is similar.

In addition, we have investigated the D order (Fig. 2.2), RP (Fig. 2.7), and S order (Fig. 2.8). The above conclusion applies to them as well.

---

1The results shown were obtained by using the MATLAB system.
3.12 Summary

The encoding and decoding processes for n-dimensional Hilbert orders \( n \geq 4 \) is quite complicated and places a heavy computational burden on the user. In this chapter, we proposed a new spatial order called the U order. Its performance is assessed analytically and empirically through four paradigmatic data processing tasks in spatial analysis: the partial match query, the range query, the eight-connected neighbour-finding, and the neighbour proximity evaluation. These paradigms have frequently been chosen as typical applications to evaluate the performance of a particular spatial order (e.g., [17], [31]). The result suggests that the U order is comparable with the Hilbert order and superior to others.

A formal description for the n-dimensional U order is presented, whereas this topic is considerably more complicated for the Hilbert order. Several encoding and decoding methods are discussed. It is concluded that the major advantage of the U order is the significant simplicity of the encoding and decoding operations for multi-dimensional data.

Future research may involve the U order applied to quadtree and octree structures as well as investigations into its parallel algorithms.
Chapter 4

The Q Orders

4.1 Overview

The U order described in the preceding chapter is of advantage for computing ef­ficiency, at a small cost of the performance. In this chapter, we will investigate several new spatial orders. Our goal is to find candidates that are competitive with the Hilbert order in performance yet require relative simple encoding and/or decoding formulas. Attention will be paid to the spatial orders with a quadrant-recursive structure, i.e., one which exhausts a quadrant of a square before exiting it. Such orders will be called the Q orders in the following.

In this chapter, in Sections 4.2 to 4.5, a recursive analysis is presented for four members of the Q order family, the Q1, Q2, Q3, and the Q4 orders. They differ in performance. Of the four orders, it appears that the Q4 order is the best. Its performance is very close to that of the Hilbert order. Thus, in Section 4.8, the
algorithms of the Q4 order are constructed. An analysis shows that its encoding and decoding algorithms only need 64.5% and 84.2% of operations of the corresponding algorithms of the Hilbert order. Finally, the relative merits of various orders are tabulated in Section 4.9.

4.2 The Q1 Order

First, the configuration of the Q1 order is established, then its performance applied to range queries in a two-dimensional attribute space is analyzed. The expected value of clusters (defined in Section 2.2) is used as a metric to evaluate the performance.

4.2.1 Configuration

Let $\mathbb{C}$ be the complex space. Consider again the region $\mathbb{C}_+ = \{z|\Re(z) \geq 0, \Im(z) \geq 0, z \in \mathbb{C}\}$ to construct the frame of the Q1 order in $\mathbb{C}_+$. Given resolution $r$, consider a square $D$ with its lower-left corner at the origin and with side $2^r$, partitioned into four quadrants and indexed by 0, 1, 2, and 3, respectively. The sequence 0, 1, 2, 3 also specifies the Q1 order of resolution 1 (Fig. 4.1). To establish the configuration of the Q1 order, a path is introduced in $D$ as a reference (Fig. 4.2).

The generation of the configuration from resolution 1 to 2 then can be determined by the following transformations:

$$T_0z = \frac{1}{2}(1 + j - z)$$
$$T_1z = \frac{1}{2}(j + z)$$

118
\[ T_2z = \frac{1}{2}(1 + j + z) \]
\[ T_3z = 1 + \frac{1}{2}(j - z) \]

where \( j \) is the imaginary number and \( T_k \) is the transformation that maps \( D \) onto its quadrant \( k (k = 0, 1, 2, 3) \).

The Q1 order of resolution 2 and its reference path is shown in Fig. 4.3 and Fig. 4.4, respectively.

The configurations of higher resolutions are determined by recursively applying the transformations defined above. The Q1 order of resolution 3 and its reference path is shown in Fig. 4.5 and Fig. 4.6, respectively.

\[
\begin{array}{cc}
1 & 2 \\
0 & 3
\end{array}
\]

Figure 4.1: The Q1 Order of Resolution 1

Figure 4.2: Path of the Q1 Order of Resolution 1
Figure 4.3: The Q1 Order of Resolution 2

Figure 4.4: Path of the Q1 Order of Resolution 2
Figure 4.5: The Q1 Order of Resolution 3

Figure 4.6: Path of the Q1 Order of Resolution 3
4.2.2 Metric Analysis of Range Query

The terminology used in the following has been defined in Chapter 2. The graphs of resolution 1, 2 and 3 of the Q1 order are shown in Fig. 4.2 to Fig. 4.6, respectively. Then, the following theorems capture the properties of interest.

Theorem 4.2.1

\[ N_r^{(1)} = \left( \frac{5}{16} \right)^4 r \quad (r \geq 2) \]

[proof] From Fig. 4.2 to Fig. 4.6 it follows that \( N_1^{(1)} = 1, N_2^{(1)} = 4N_1^{(1)} + 1, \) and \( N_3^{(1)} = 4N_2^{(1)} + 0. \) No new links are introduced for higher resolutions, hence \( N_k^{(1)} = 4N_{k-1}^{(1)} \quad (k \geq 3). \) Therefore,

\[ N_r^{(1)} = 4^i N_{r-i}^{(1)} \quad (i = 1, \ldots, r - 2) \]

from which the closed form is obtained by setting \( i \) to \( r - 2: \)

\[ N_r^{(1)} = 4^{r-2} N_{2}^{(1)} \]

\[ = \left( \frac{5}{16} \right)^4 r \quad \square \]

Theorem 4.2.2

\[ N_r^{(2)} = \left( \frac{3}{8} \right)^4 r - \left( \frac{5}{4} \right)2^r \quad (r \geq 2) \]

[proof] According to the structure of the Q1 order (the graphs with resolution up to 3 are presented in Fig. 4.2 to Fig. 4.6), a sequence can be established as follows:
Then the theorem is proved by recurrence. □

Theorem 4.2.3

\[ N_r^{(3)} = \left(\frac{3}{16}\right)4^r \quad (r \geq 2) \]

[proof] According to the structure of the Q1 order, a sequence can be established as follows:

\[ N_1^{(3)} = 0 \]
\[ N_2^{(3)} = 3 \]
\[ N_3^{(3)} = 12 \]

\[ \cdots \quad \cdots \]

\[ N_k^{(3)} = 4N_{k-1}^{(3)} + b_k \]
\[ b_k = 2b_{k-1} \]

\[ (k \geq 4, b_3 = 10) \]

In general, we have \( N_k^{(3)} = 4N_{k-1}^{(3)} \) \((k \geq 3)\), since no new links are introduced into the graphs of resolution higher than 2. Then the theorem is proved by recurrence. □

Theorem 4.2.4

\[ N_r^{(4)} = \left(\frac{1}{8}\right)4^r - \left(\frac{3}{4}\right)2^r + 1 \quad (r \geq 1) \]
According to the structure of the Q1 order, a sequence can be established as follows:

\[
\begin{align*}
N_1^{(4)} &= 0 \\
N_2^{(4)} &= 0 \\
N_3^{(4)} &= 3 \\
N_4^{(4)} &= 21 \\
\vdots & \quad \vdots \\
N_k^{(4)} &= 4N_{k-1}^{(4)} + d_k \\
d_k &= d_{k-1} + \left(\frac{3}{8}\right)2^k \\
(k \geq 4, d_3 = 3)
\end{align*}
\]

Then the theorem is proved by recurrence. \( \square \)

**Corollary 4.2.1** When \( r \) is large, the expectation of the number of clusters per template query tends to:

\[
R_r = \frac{17}{8} = 2.125
\]

There are \((2^r - 1)^2\) templates in a lattice system with \(2^r \times 2^r\) cells. Let \(p_1, p_2, p_3,\) and \(p_4\) denote the occurrence frequencies of the templates with one, two, three, and four clusters in this lattice system. Then we have:

\[
p_1 = \lim_{r \to \infty} \frac{N_r^{(1)}}{(2^r - 1)^2} = \frac{5}{16}
\]

Similarly, we have:

\[
p_2 = \frac{3}{8}
\]
Consequently, when $r$ is large, the expectation of the number of clusters per template tends to:

$$R_r = 1p_1 + 2p_2 + 3p_3 + 4p_4$$

$$= \frac{17}{8}$$

$$= 2.125$$

The statistic obtained above will be compared with other spatial orders (Table 4.2).

4.3 The Q2 Order

In this section, the configuration of the Q2 order is established, then its performance applied to range queries in a two-dimensional attribute space is analyzed. The expected value of clusters (defined in Section 2.2) is used as a metric to evaluate the performance.

4.3.1 Configuration

The configuration of the Q2 order is determined by the following transformations:

$$T_0z = \frac{1}{2}(1 + jz)$$

$$T_1z = \frac{1}{2}[1 + j(1 + z)]$$
\[ T_2z = 1 + j(1 - \frac{1}{2} \bar{z}) \]
\[ T_3z = 1 + \frac{1}{2} j(1 - \bar{z}) \]

where \(j\) is the imaginary number and \(T_k\) is the transformation that maps \(D\) onto its quadrant \(k\) \((k = 0, 1, 2, 3)\).

The Q2 orders of resolution 1 to 3 and the corresponding reference paths are shown in Fig. 4.7 to Fig. 4.12.

\[
\begin{array}{ccc}
1 & 2 \\
0 & 3 \\
\end{array}
\]

Figure 4.7: The Q2 Order of Resolution 1

\[
\begin{array}{cccc}
6 & 7 & 9 & 8 \\
5 & 4 & 10 & 11 \\
2 & 3 & 13 & 12 \\
1 & 0 & 14 & 15 \\
\end{array}
\]

Figure 4.8: The Q2 Order of Resolution 2

### 4.3.2 Metric Analysis of Range Query

The graphs of resolution 1, 2 and 3 of the Q2 order are depicted in Fig. 4.10 to Fig. 4.12, respectively. The terminology used in the following has been defined in
Figure 4.9: The Q2 Order of Resolution 3

Figure 4.10: Path of the Q2 Order of Resolution 1
Figure 4.11: Path of the Q2 Order of Resolution 2

Figure 4.12: Path of the Q2 Order of Resolution 3
Chapter 2.

**Theorem 4.3.1**

\[ N_r^{(1)} = \left(\frac{3}{8}\right)4^r \quad (r \geq 2) \]

**Proof** From Fig. 4.10 to Fig. 4.12 it follows that \( N_1^{(1)} = 1, N_2^{(1)} = 4N_1^{(1)} + 2, \) and \( N_3^{(1)} = 4N_2^{(1)} + 0. \) Again, no new links are introduced for higher resolutions, hence \( N_k^{(1)} = 4N_{k-1}^{(1)} \) \((k \geq 3)\). Therefore,

\[ N_r^{(1)} = 4^iN_{r-i}^{(1)} \quad (i = 1, \ldots, r - 2) \]

As before, the closed form is obtained by setting \( i \) to \( r - 2 \):

\[ N_r^{(1)} = 4^{r-2}N_2^{(1)} = \left(\frac{3}{8}\right)4^r \quad \square \]

**Theorem 4.3.2**

\[ N_r^{(2)} = \left(\frac{1}{8}\right)4^r - \left(\frac{1}{2}\right)2^r \quad (r \geq 2) \]

**Proof** According to the structure of the Q2 order (the graphs with resolution up to 3 are presented in Fig. 4.10 to Fig. 4.12), a sequence can be established as follows:

\[
\begin{align*}
N_1^{(2)} &= 0 \\
N_2^{(2)} &= 0 \\
N_3^{(2)} &= 4 \\
\vdots & \quad \vdots \\
N_k^{(2)} &= 4N_{k-1}^{(2)} + b_k \\
b_k &= 2b_{k-1} \\
(k &\geq 4, b_3 = 4)
\end{align*}
\]

129
Then the theorem is proved by recurrence. □

**Theorem 4.3.3**

\[ N_r^{(3)} = \left( \frac{3}{8} \right) 4^r - \left( \frac{3}{4} \right) 2^r \quad (r \geq 2) \]

**[proof]** According to the structure of the Q2 order, a sequence can be established as follows:

\[
\begin{align*}
N_1^{(3)} & = 0 \\
N_2^{(3)} & = 3 \\
\cdots & \cdots \\
N_k^{(3)} & = 4N_{k-1}^{(3)} + c_k \\
c_k & = 2c_{k-1} \\
& \quad (k \geq 3, c_2 = 3)
\end{align*}
\]

Then the theorem is proved by recurrence. □

**Theorem 4.3.4**

\[ N_r^{(4)} = \left( \frac{1}{8} \right) 4^r - \left( \frac{3}{4} \right) 2^r + 1 \quad (r \geq 2) \]

**[proof]** According to the structure of the Q2 order, a sequence can be established as follows:

\[
\begin{align*}
N_1^{(4)} & = 0 \\
N_2^{(4)} & = 0 \\
N_3^{(4)} & = 3 \\
\cdots & \cdots \\
N_k^{(4)} & = 4N_{k-1}^{(4)} + d_k \\
d_k & = d_{k-1} + \left( \frac{3}{8} \right) 2^k \\
& \quad (k \geq 4, d_3 = 3)
\end{align*}
\]

Then the theorem is proved by recurrence. □
Corollary 4.3.1 When \( r \) is large, the expectation of the number of clusters per template query tends to:

\[
R_r = \frac{9}{4} = 2.25
\]

[proof] There are \((2^r - 1)^2\) templates in a lattice system with \(2^r \times 2^r\) cells. Let \(p_1, p_2, p_3,\) and \(p_4\) denote the occurrence frequencies of the templates with one, two, three, and four clusters in this lattice system. Then we have:

\[
p_1 = \lim_{r \to \infty} \frac{N_r^{(1)}}{(2^r - 1)^2} = \frac{3}{8}
\]

Similarly, we have:

\[
p_2 = \frac{1}{8}, \quad p_3 = \frac{3}{8}, \quad p_4 = \frac{1}{8}
\]

Consequently, when \( r \) is large, the expectation of the number of clusters per template tends to:

\[
R_r = 1p_1 + 2p_2 + 3p_3 + 4p_4 = \frac{9}{4} = 2.25 \quad \square
\]

The statistic obtained above will be compared with other spatial orders (Table 4.2).
4.4 The Q3 Order

In this section, the configuration of the Q3 order is established, then its performance applied to range queries in a two-dimensional attribute space is analyzed. The expected value of clusters (defined in Section 2.2) is used as a metric to evaluate the performance.

4.4.1 Configuration

The configuration of the Q3 order is determined by the following transformations:

\[
T_0z = \frac{1}{2} j \bar{z} \\
T_1z = \frac{1}{2} j (1 + \bar{z}) \\
T_2z = 1 + j (1 - \frac{1}{2} \bar{z}) \\
T_3z = 1 + \frac{1}{2} j (1 - \bar{z})
\]

where \( j \) is the imaginary number and \( T_k \) is the transformation that maps \( D \) onto its quadrant \( k \) (\( k = 0, 1, 2, 3 \)).

The Q3 order of resolution 1 to 3 and the corresponding reference paths are shown in Fig. 4.13 to Fig. 4.18.

\[
\begin{array}{cc}
1 & 2 \\
0 & 3 \\
\end{array}
\]

Figure 4.13: The Q3 Order of Resolution 1
Figure 4.14: The Q3 Order of Resolution 2

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 4.15: The Q3 Order of Resolution 3

<table>
<thead>
<tr>
<th>31</th>
<th>28</th>
<th>27</th>
<th>24</th>
<th>39</th>
<th>36</th>
<th>35</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>29</td>
<td>26</td>
<td>25</td>
<td>38</td>
<td>37</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>21</td>
<td>22</td>
<td>41</td>
<td>42</td>
<td>45</td>
<td>46</td>
</tr>
<tr>
<td>16</td>
<td>19</td>
<td>20</td>
<td>23</td>
<td>40</td>
<td>43</td>
<td>44</td>
<td>47</td>
</tr>
<tr>
<td>15</td>
<td>12</td>
<td>11</td>
<td>8</td>
<td>55</td>
<td>52</td>
<td>51</td>
<td>48</td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>10</td>
<td>9</td>
<td>54</td>
<td>53</td>
<td>50</td>
<td>49</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>57</td>
<td>58</td>
<td>61</td>
<td>62</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>56</td>
<td>59</td>
<td>60</td>
<td>63</td>
</tr>
</tbody>
</table>
Figure 4.16: Path of the Q3 Order of Resolution 1

Figure 4.17: Path of the Q3 Order of Resolution 2

Figure 4.18: Path of the Q3 Order of Resolution 3
4.4.2 Metric Analysis of Range Query

The graphical representation for the Q3 order is more convenient for the metric analysis. The graphs of resolution 1, 2 and 3 are demonstrated in Fig. 4.16 to Fig. 4.18, respectively. The terminology used in the following has been defined in Chapter 2. The properties concerned are narrated as a series of theorems.

**Theorem 4.4.1**

\[ N_r^{(1)} + N_{r-1}^{(1)} = \left(\frac{23}{48}\right)4^r - \frac{2}{3} \quad (r \geq 2) \]

[**proof**] It follows from Fig. 4.16 and Fig. 4.17 that \( N_1^{(1)} = 1 \) and \( N_2^{(1)} = 4N_1^{(1)} + 2 = 6 \). The latter is established due to the fact that, in addition to the four one-cluster templates arising from replicating the Q3 order with resolution 1, a new one-cluster template with a continuous path is formed. In general, \( N_k^{(1)} = 4N_{k-1}^{(1)} + 2 \) holds for \( k \) even and \( N_k^{(1)} = 4N_{k-1}^{(1)} \) for \( k \) odd \((k = 2, 3, \ldots, r)\), according to the structure of the Q3 order. With the intermediate variable \( T_k = N_k^{(1)} + N_{k+1}^{(1)} \) \((k = 1, 2, \ldots, r-1)\), we have:

\[
\begin{align*}
T_1 &= N_1^{(1)} + N_2^{(1)} \\
T_2 &= N_2^{(1)} + N_3^{(1)} \\
&= 4(N_1^{(1)} + N_2^{(1)}) + 2 \\
&= 4T_1 + 2 \\
\cdots \\
T_k &= 4T_{k-1} + 2 \\
&\quad (k = 2, 3, \ldots, r-1)
\end{align*}
\]

Then by iteration we have:

\[
T_k = 4^jT_{k-j} + \frac{2(4^j - 1)}{3} \quad (j = 1, 2, \ldots, k-2)
\]
The closed form is obtained by setting \( j \) to \( k - 1 \):

\[
T_k = 4^{k-1}T_1 + \frac{2(4^{k-1} - 1)}{3}
\]

\[
= \left(\frac{23}{12}\right)4^k - \frac{2}{3}
\]

Therefore,

\[
T_{r-1} = \left(\frac{23}{48}\right)4^r - \frac{2}{3}
\]

**Theorem 4.4.2**

\[
N_r^{(2)} + N_{r-1}^{(2)} = \left(\frac{13}{32}\right)4^r - \left(\frac{3}{2}\right)2^r + 2
\]

\[
(r \geq 3)
\]

[proof] According to the structure of the Q3 order (the graphs with resolution up to \( 3 \) are presented in Fig. 4.16 to Fig. 4.18), a sequence can be established. With the intermediate variable \( T_k = N_k^{(2)} + N_{k+1}^{(2)} \) \((k = 1, 2, \ldots, r - 1)\), we have:

\[
T_1 = 2
\]

\[
T_2 = 10
\]

\[
\ldots \quad \ldots
\]

\[
T_k = 4T_{k-1} + 3 \cdot 2^k - 6
\]

\[
(k \geq 3)
\]

Then the theorem is proved by recurrence. □

**Theorem 4.4.3**

\[
N_r^{(3)} + N_{r-1}^{(3)} = \frac{4^r}{24} - \frac{2}{3}
\]

\[
(r \geq 2)
\]
[proof] According to the structure of the Q3 order a sequence can be established.

With the intermediate variable \( T_k = N_k^{(3)} + N_{k+1}^{(3)} \) \((k = 1, 2, \ldots, r - 1)\), we have:

\[
\begin{align*}
T_1 &= 0 \\
T_k &= 4T_{k-1} + 2 \\
&\quad (k \geq 2)
\end{align*}
\]

Then the theorem is proved by recurrence. \(\square\)

**Theorem 4.4.4**

\[
N_r^{(4)} + N_{r-1}^{(4)} = \left(\frac{31}{96}\right)4^r - \left(\frac{3}{2}\right)2^r + \frac{4}{3}
\]

\((r \geq 3)\)

[proof] According to the structure of the Q3 order a sequence can be established.

With the intermediate variable \( T_k = N_k^{(4)} + N_{k+1}^{(4)} \) \((k = 1, 2, \ldots, r - 1)\), we have:

\[
\begin{align*}
T_1 &= 1 \\
T_2 &= 10 \\
\ldots & \ldots \\
T_k &= 4T_{k-1} + 3 \cdot 2^k - 4 \\
&\quad (k \geq 3)
\end{align*}
\]

Then the theorem is proved by recurrence. \(\square\)

**Corollary 4.4.1** When \( r \) is large, the expectation of the number of clusters per range query tends to:

\[
R_r = \frac{13}{6}
\]

\( \approx 2.167 \)
[proof] There are $(2^k - 1)^2$ templates in a lattice system with $2^k \times 2^k$ cells ($k = 1, 2, \ldots, r$). Let $p_1, p_2, p_3,$ and $p_4$ denote the occurrence frequencies of the templates with one, two, three, and four clusters in the lattice of resolution $r$. Then we have:

$$
\begin{align*}
    p_1 &= \lim_{{r \to \infty}} \frac{N_r^{(1)} + N_{r-1}^{(1)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} \\
    &= \frac{23}{60} \\
    p_2 &= \lim_{{r \to \infty}} \frac{N_r^{(2)} + N_{r-1}^{(2)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} \\
    &= \frac{13}{40} \\
    p_3 &= \lim_{{r \to \infty}} \frac{N_r^{(3)} + N_{r-1}^{(3)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} \\
    &= \frac{1}{30} \\
    p_4 &= \lim_{{r \to \infty}} \frac{N_r^{(4)} + N_{r-1}^{(4)}}{(2^r - 1)^2 + (2^{r-1} - 1)^2} \\
    &= \frac{31}{120}
\end{align*}
$$

Consequently, when $r$ is large, the expectation of the number of clusters per range tends to:

$$
R_r = 1p_1 + 2p_2 + 3p_3 + 4p_4 \\
\approx \frac{13}{6} \\
\approx 2.167 \quad \square
$$

The statistic obtained above will be compared with other spatial orders (Table 4.2).
4.5 The Q4 Order

In this section, the configuration of the Q4 order is established, then its performance applied to range queries in a two-dimensional attribute space is analyzed. The expected value of clusters (defined in Section 2.2) is used as a metric to evaluate the performance.

4.5.1 Configuration

The configuration of the Q4 order is determined by the following transformations:

\[
T_0z = \frac{1}{2}(1 + jz)
\]
\[
T_1z = \frac{1}{2}[1 + j(z + 1)]
\]
\[
T_2z = \frac{1}{2}[1 + j(2 - z)]
\]
\[
T_3z = \frac{1}{2}[1 + j(1 - z)]
\]

where \( j \) is the imaginary number and \( T_k \) is the transformation that maps \( D \) onto its quadrant \( k (k = 0, 1, 2, 3) \).

The Q4 order of resolution 1 to 3 and the corresponding reference paths are shown in Fig. 4.19 to Fig. 4.24.

\[
\begin{array}{cc}
1 & 2 \\
0 & 3 \\
\end{array}
\]

Figure 4.19: The Q4 Order of Resolution 1
Figure 4.20: The Q4 Order of Resolution 2

\[
\begin{array}{cccc}
6 & 7 & 8 & 9 \\
5 & 4 & 11 & 10 \\
2 & 3 & 12 & 13 \\
1 & 0 & 15 & 14 \\
\end{array}
\]

Figure 4.21: The Q4 Order of Resolution 3
Figure 4.22: Path of the Q4 Order of Resolution 1

Figure 4.23: Path of the Q4 Order of Resolution 2

Figure 4.24: Path of the Q4 Order of Resolution 3
4.5.2 Metric Analysis of Partial Match Query

In a partial match query, one of the attributes is specified in a two-attribute record while the other is not. Geometrically, the set of the selected records corresponds to a line parallel to one of the coordinate axes in the two-dimensional Cartesian space.

The graphical representation for the Q4 order is more convenient for the metric analysis. The graphs of resolution 1, 2 and 3 have been demonstrated in Fig. 4.22 to Fig. 4.24, respectively.

Let \( R_r \) be the total number of clusters in the lattice of \( 2^r \times 2^r \). The result is narrated as the following theorem:

**Theorem 4.5.1** The total number of clusters arising from the partial match query in the lattice of resolution \( r \) is given by:

\[
R_r = \left( \frac{67}{64} \right)^4 \quad (r \geq 3)
\]

**[proof]** It is observed from Fig. 4.22 that there are five clusters for \( r=1 \). This leads to \( R_1 = 5 \). For the Q4 order with \( r=2 \), in addition to the four copies arising from replicating the graph with \( r=1 \), there are three new links introduced (Fig. 4.23), hence \( R_2 = 4R_1 - 3 \). Similarly, one has \( R_3 = 4R_2 - 1 \). However, no new links are introduced for \( r \geq 4 \). Therefore, \( R_k = 4R_{k-1} \) (\( k \geq 4 \)). Consequently,

\[
R_r = 4R_{r-1} = 4^2R_{r-2} = \ldots = 4^iR_{r-i} \quad (i = 1, \ldots, r - 3)
\]

142
The closed form is obtained by setting $i$ to $r - 3$:

$$R_r = 4^{r-3} R_3$$
$$= \left(\frac{67}{64}\right)4^r \quad \Box$$

**Corollary 4.5.1** The expectation of the number of clusters per partial match query is:

$$P_r = \left(\frac{67}{64}\right)2^{r-1}$$

**[proof]** There are $2^r$ horizontal selections and $2^r$ vertical selections altogether. Therefore, the expectation of the number of clusters per query is:

$$P_r = \frac{R_r}{2^r + 2^r}$$
$$= \left(\frac{67}{64}\right)2^{r-1}$$
$$\approx 1.047 \times 2^{r-1} \quad \Box$$

The statistic obtained above is compared with those of the U, Z, and Hilbert order (Table 4.1). It is observed that the performance of the Q4 order is close to the Hilbert order, and is better than other two orders. In the next section the range query will be analyzed.

### 4.5.3 Metric Analysis of Range Query

The terminology used in this section has been defined in Chapter 2. The properties concerned are narrated as a series of theorems:
<table>
<thead>
<tr>
<th>Spatial Order</th>
<th>The Expectation of Cluster Number Per Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>$1.5 \times 2^{r-1}$</td>
</tr>
<tr>
<td>U</td>
<td>$1.167 \times 2^{r-1}$</td>
</tr>
<tr>
<td>Q4</td>
<td>$1.047 \times 2^{r-1}$</td>
</tr>
<tr>
<td>H</td>
<td>$1.0 \times 2^{r-1}$</td>
</tr>
</tbody>
</table>

Table 4.1: Expectation for Partial Match Query

**Theorem 4.5.2**

$$N_r^{(1)} = \left(\frac{25}{64}\right)4^r \ (r \geq 3)$$

**Proof** It follows from Fig. 4.22 to Fig. 4.24 that $N_1^{(1)} = 1, N_2^{(1)} = 4N_1^{(1)} + 2,$ and $N_3^{(1)} = 4N_2^{(1)} + 1$. No new links are introduced for higher resolutions, hence $N_k^{(1)} = 4N_{k-1}^{(1)} \ (k \geq 4).$ Therefore,

$$N_r^{(1)} = 4^i N_{r-i}^{(1)} \ (i = 1, \ldots, r - 3)$$

The closed form is obtained by setting $i$ to $r - 3$:

$$N_r^{(1)} = 4^{r-3} N_3^{(1)}$$
$$= 25 \times 4^{r-3}$$
$$= \left(\frac{25}{64}\right)4^r$$

**Theorem 4.5.3**

$$N_r^{(2)} = \left(\frac{21}{64}\right)4^r - \left(\frac{5}{4}\right)2^r \ (r \geq 3)$$
[proof] According to the structure of the Q4 order (the graphs with resolution up to 3 are presented in Fig. 4.22 to Fig. 4.24), a sequence can be established as follows:

\[
\begin{align*}
N_1^{(2)} &= 0 \\
N_2^{(2)} &= 1 \\
N_3^{(2)} &= 11 \\
\cdots & \cdots \\
N_k^{(2)} &= 4N_{k-1}^{(2)} + b_k \\
b_k &= 2b_{k-1} \\
(k \geq 5, b_4 = 20)
\end{align*}
\]

Then the theorem is proved by recurrence. □

Theorem 4.5.4

\[
N_r^{(3)} = \left(\frac{5}{64}\right)^4 r \quad (r \geq 3)
\]

[proof] According to the structure of the Q4 order, a sequence can be established as follows:

\[
\begin{align*}
N_1^{(3)} &= 0 \\
N_2^{(3)} &= 1 \\
N_3^{(3)} &= 5 \\
\cdots & \cdots 
\end{align*}
\]

In general, we have \(N_k^{(3)} = 4N_{k-1}^{(3)}\) (\(k \geq 4\)), since no new links are introduced into the graphs of resolution higher than 3. Then the theorem is proved by recurrence. □

Theorem 4.5.5

\[
N_r^{(4)} = \left(\frac{13}{64}\right)^4 r - \left(\frac{3}{4}\right)^2 r + 1 \quad (r \geq 3)
\]
[proof] According to the structure of the Q4 order, a sequence can be established as follows:

\[
\begin{align*}
N_1^{(4)} &= 0 \\
N_2^{(4)} &= 1 \\
N_3^{(4)} &= 8 \\
N_4^{(4)} &= 41 \\
&\vdots \quad \vdots \\
N_k^{(4)} &= 4N_{k-1}^{(4)} + d_k \\
d_k &= d_{k-1} + \left(\frac{3}{8}\right)^2 \\
&\quad (k \geq 5, d_4 = 9)
\end{align*}
\]

Then the theorem is proved by recurrence. □

**Corollary 4.5.2** When \( r \) is large, the expectation of the number of clusters per template query tends to:

\[
R_{ept} = \frac{67}{32}
\]

\[
= 2.09375
\]

[proof] There are \((2^r - 1)^2\) templates in a lattice system with \(2^r \times 2^r\) cells. Let \(p_1, p_2, p_3,\) and \(p_4\) denote the occurrence frequencies of the templates with one, two, three, and four clusters in this lattice system. Then we have:

\[
p_1 = \lim_{r \to \infty} \frac{N_r^{(1)}}{(2^r - 1)^2}
\]

\[
= \frac{25}{64}
\]

Similarly, we have:

\[
p_2 = \frac{21}{64}
\]

146
Consequently, when $r$ is large, the expectation of the number of clusters per template tends to:

$$R_{opt} = 1p_1 + 2p_2 + 3p_3 + 4p_4$$

\[
\begin{align*}
p_3 &= \frac{5}{64} \\
p_4 &= \frac{13}{64}
\end{align*}
\]

The statistic obtained above is compared with those of the G, U, Z, and Hilbert order (Table 4.2). It is observed that the performance of the Q4 order is close to the Hilbert order.

### 4.6 The Q5, Q6, and Q7 Orders

An eight-variable integer programming model can be constructed to determine the optimal spatial order(s) with quadrant-exhaustive and single-type hierarchical structures. Single-type implies that the iteration rule is identical for all resolutions. To simplify the analysis, consider the orders developed from the rook-connected path in the system of resolution 1 only (as shown in Fig. 4.2, for example). Let this path be called the basic path. The sub-ordering of each quadrant in the system of resolution 2 can be generated from this basic path by rotating it $0^\circ, 90^\circ, 180^\circ, \text{ and } 270^\circ$, with two possible indexing sequences each. Therefore, two integer variables are required to define the sub-ordering for each quadrant, one for the rotation state and another for
### Spatial Order

<table>
<thead>
<tr>
<th>Spatial Order</th>
<th>The Expectation of Cluster Number Per Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>2.625</td>
</tr>
<tr>
<td>G</td>
<td>2.5</td>
</tr>
<tr>
<td>U</td>
<td>2.333</td>
</tr>
<tr>
<td>Q1</td>
<td>2.125</td>
</tr>
<tr>
<td>Q2</td>
<td>2.25</td>
</tr>
<tr>
<td>Q3</td>
<td>2.167</td>
</tr>
<tr>
<td>Q4</td>
<td>2.094</td>
</tr>
<tr>
<td>H</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Table 4.2: Behavior of the Average Case
the indexing sequence. Their domains can be defined as \{0, 90, 180, 270\} and \{0, 1\}, respectively. Since the basic path has eight distinct ordering possibilities, there are in total \(8^4 = 4096\) ordering instances for resolution 2. The objective function can be any metric that is meaningful to characterize the performance of a particular order applied to query retrievals, for example, the metrics \(d_{\text{max}}, d_1,\) or \(d_s\) defined in Section 3.3. Instead of trying to solve this type of integer programming problem, which may have to rely on some enumeration techniques, in this section we investigate a subset of orders that are rook-connected at resolution 2.

There are only six possible rook-connected patterns in a \(2 \times 2\) lattice system. Three of them, namely, the Hilbert order (original pattern), the Q1 order, and the Q4 order, have been discussed. In the following, the remaining three are named the Q5, Q6, and Q7 order, respectively. Their configuration for resolution 2 are the same as those of patterns L2, L3, and L4 of the Hilbert order, but the iterations for resolution 3 are different. To save space, the mathematical descriptions are omitted here and only the paths are illustrated in Fig. 4.25.
Figure 4.25: Paths of the Q5, Q6 and Q7 Orders of Resolution 3
4.7 Neighbour Finding and Proximity

A set of algorithms has been designed and implemented for the purpose of evaluating the performance of the Q orders. Their relative merits are assessed computationally for the 8-connected neighbor-finding problem over the range of resolution \( r \) from 2 through 8. The average values of \( d_{\text{max}} \) and \( d_1 \) over the set of \( 2^r \times 2^r \) cells are reported in Table 4.3 and Table 4.4, respectively.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>5.75</td>
<td>14.03</td>
<td>32.14</td>
<td>69.68</td>
<td>146.07</td>
<td>300.17</td>
<td>609.68</td>
</tr>
<tr>
<td>Q5</td>
<td>5.88</td>
<td>14.41</td>
<td>32.41</td>
<td>69.96</td>
<td>146.30</td>
<td>300.34</td>
<td>609.80</td>
</tr>
<tr>
<td>Q7</td>
<td>6.00</td>
<td>14.59</td>
<td>32.98</td>
<td>71.32</td>
<td>149.42</td>
<td>307.03</td>
<td>623.67</td>
</tr>
<tr>
<td>Q4</td>
<td>6.13</td>
<td>14.63</td>
<td>33.02</td>
<td>71.51</td>
<td>150.01</td>
<td>308.52</td>
<td>627.05</td>
</tr>
<tr>
<td>Q3</td>
<td>6.13</td>
<td>14.88</td>
<td>33.53</td>
<td>72.12</td>
<td>150.65</td>
<td>309.16</td>
<td>627.74</td>
</tr>
<tr>
<td>Q2</td>
<td>6.13</td>
<td>14.86</td>
<td>33.61</td>
<td>72.31</td>
<td>150.95</td>
<td>309.57</td>
<td>628.10</td>
</tr>
<tr>
<td>Q6</td>
<td>6.19</td>
<td>15.17</td>
<td>35.20</td>
<td>76.92</td>
<td>161.89</td>
<td>333.36</td>
<td>677.94</td>
</tr>
<tr>
<td>Q1</td>
<td>6.50</td>
<td>16.00</td>
<td>37.91</td>
<td>83.90</td>
<td>177.90</td>
<td>367.91</td>
<td>749.96</td>
</tr>
</tbody>
</table>

Table 4.3: Average Maximum Distance of Finding 8-Neighbors in a 2D Grid

The Q orders have also been compared with each other for the proximity problem. The scheme is similar to that described in Section 3.4. The results over the set of \( 2^r \times 2^r \) cells (\( r = 1, 2, \ldots, 7 \)) are reported in Table 4.5.
4.8 Encoding and Decoding the Q4 Order

Of the seven Q orders proposed above, the Q4 order behaves best on the average for the 2 × 2 templates. For other applications like neighbor finding and proximity problems, the Q4 order also exhibits satisfactory performance. On the other hand, of the seven Q orders, only the Q1 and Q4 orders do not involve the aspect iteration. This property implies that these algorithms may be simpler than the other ones. However, the behaviour of the Q1 order is not as good as the Q4 order. Thus we choose the Q4 order as a representative and derive its encoding and decoding algorithms. These algorithms do not depend on look-up tables. The methodology is similar to that used for the Hilbert order.

Table 4.4: Average Total Distance of Finding 8-Neighbors in a 2D Grid
Table 4.5: Average Farthest Distance of the Neighbors in a 2D Grid

4.8.1 Encoding

The Q4 orders and curves of various resolutions have been depicted in Fig. 4.19 to Fig. 4.24. In this section, the iteration rules to construct the Q4 order are described first. These rules establish the basis for deriving the analytic encoding and decoding formulas.

The Q4 order is to be established in a subset $D = \{0 \leq x < 2^r, 0 \leq y < 2^r\}$ in the two-dimensional integer space.

Given the coordinates of a particular point $P$ in the lattice system $D$, the corresponding value of $P$ in the Q4 order ($W$ - value in brief) is to be determined. This process is called encoding.
Similar to the Hilbert order, the analysis for the Q4 order is based on the top-down approach. First, the whole lattice is partitioned into four quadrants and indexed in base 4. Then each quadrant is recursively processed in the same way. The first and second levels are illustrated in Fig. 4.26 and Fig. 4.27, respectively.

![Figure 4.26: The First Level](image)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

![Figure 4.27: The Second Level](image)

<table>
<thead>
<tr>
<th>12</th>
<th>13</th>
<th>20</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>10</td>
<td>23</td>
<td>22</td>
</tr>
<tr>
<td>02</td>
<td>03</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>33</td>
<td>32</td>
</tr>
</tbody>
</table>

Let the coordinate values $x$ and $y$ be expressed by bit-strings: $x = x_{r-1} \ldots x_1 x_0$ and $y = y_{r-1} \ldots y_1 y_0$, and the corresponding W-value be expressed in terms of quaternary digits ($w$-digits in brief): $w = 4^{r-1} x_{r-1} + \ldots + 4^1 x_1 + 4^0 x_0$. At level $i$ ($i = 1, 2, \ldots, r$), given $x_{r-i}$ and $y_{r-i}$, the $i$-th most significant digit of the coordinates $x$ and $y$, the $i$-th most significant digit of the W-value (denoted by $w_{r-i}$) is to be determined.

The terminology to be used has been defined in Section 2.3. For example, a unit is
an item specifying the indexing strategy for four quadrants of a square region. It can be represented by a simple graph with three edges and four nodes labeled by indexes in terms of quaternary digits \(n\_digit\) in brief, as shown in Fig. 4.28. For the Q4 order, the units rotate, thus their orientations need to be iterated. Their aspects, however, do not change. In other words, the nodes of a unit are always indexed clockwise. According to the structure of the Q4 order, the units generated from node 0 and 1 of the unit at level \(i\) \((i = 1, 2, \ldots, r)\) always rotate 90 degrees counterclockwise, while the units generated from node 2 and 3 always rotate 90 degrees clockwise.

The orientation of the unit at the level 1 is called the principal orientation of the Q4 order. The analysis for the orientation will be developed in the two-dimensional Cartesian coordinate system. The structure of level 1 is specified as shown in Fig. 4.28. This structure determines the overall appearance of the order considered.

![Figure 4.28: Structure and Orientation of Level 1](image)

Level 2 is generated by constructing four units, labeled \(U_v\) \((v = 0, 1, 2, 3)\), from the corresponding indexed nodes of the unit at level 1 (Fig. 4.29). The indexes of nodes are obtained by concatenating a quaternary digit to the index of the unit. Note the nodes in these units are indexed progressively clockwise (Fig. 4.27).
The orientation of a unit is specified by a normalized vector $m$. According to the specified arrangement of the Q4 order relative to the coordinate system, $m \in \{(0,1)^T,(-1,0)^T,(0,-1)^T,(1,0)^T\}$. Two units with their normal vector are demonstrated in Fig. 4.30.

Similarly to the Hilbert order, we use a directional index to conduct the encoding process.

The relation between the directional vector $m$ and an index $v$ has been shown in Fig. 2.20. Since $v$ has four possible values, it can be represented by two bits $a$ and $b$. Similarly, two bits $n_{2k-1}$ and $n_{2k-2}$ are used to represent an $n$-digit.

According to the inherent structure of the Q4 order, $n_{2k-1}$ and $n_{2k-2}$ can be expressed in terms of $a,b,x_{k-1}$ and $y_{k-1}$:

$$n_{2k-1} = \bar{a}\bar{b}x_{k-1} + \bar{a}by_{k-1} + a\bar{b}x_{k-1} + aby_{k-1}$$
\[ b(a \oplus x_{k-1}) + b(a \oplus y_{k-1}) \]
\[ n_{2k-2} = b \oplus x_{k-1} \oplus y_{k-1} \]
\[ k = 1, \ldots, r. \]

The iteration of \( v \) can be carried out by adding an incremental factor \( dv \):

\[ \text{if } n_{\text{digit}} = 0 \text{ or } 1 \quad \text{then } dv = 1; \]
\[ \text{else } \quad dv = 3; \]

Using a bit \( c \) to represent \( dv \), the relation defined by the \text{if} clauses above can be expressed as a table

<table>
<thead>
<tr>
<th>( n_{2k-1} )</th>
<th>( n_{2k-2} )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

which in turn can be formulated as a simple expression:

\[ c = n_{2k-1} \quad (4.1) \]

The updated value \( v' \) of the directional index \( v \) is obtained by the addition:

\[ v' = (v + dv) \mod 4 \quad (4.2) \]

Let \( v' \) be represented by two bits \( a' \) and \( b' \), then Eq.(4.2) can be formulated as:

\[ a' = a \oplus b \oplus c \quad (4.3) \]
\[ b' = \overline{b} \quad (4.4) \]
Substituting (4.1) into (4.3) yields

\[ a' = a \oplus b \oplus n_{2k-1} \]  

(4.5)

In order to obtain the final expressions of \( a' \) and \( b' \) in terms of \( a, b, x_{k-1}, \) and \( y_{k-1}, \) tedious algebraical calculations need to be performed. The derivation details are ignored here. The encoding formula is listed as below:

\[
\begin{align*}
  a_{r-1} &= 0 \\
  b_{r-1} &= 0 \\
  w_{2r-2} &= b_{r-1} \oplus x_{r-1} \oplus y_{r-1} \\
  w_{2r-1} &= \overline{b_{r-1}}(a_{r-1} \oplus x_{r-1}) + b_{r-1}(a_{r-1} \oplus y_{r-1}) \\
  a_{k-1} &= b_k y_k + \overline{b_k} x_k \\
  b_{k-1} &= \overline{b_k} \\
  w_{2k-2} &= b_{k-1} \oplus x_{k-1} \oplus y_{k-1} \\
  w_{2k-1} &= \overline{b_{k-1}}(a_{k-1} \oplus x_{k-1}) + b_{k-1}(a_{k-1} \oplus y_{k-1}) \\
  & \quad \text{for } (k = r - 1, r - 2, \ldots, 1).
\end{align*}
\]

The order of this algorithm is \( 20r \).

4.8.2 Decoding

Decoding is the inverse process of encoding: given the \( W \)-value of a particular point, the corresponding coordinate pair \((x, y)\) is to be determined. Let \( w_{\text{digit}} \) be represented by the two bits \( w_{2k-1} \) and \( w_{2k-2} \). The transition rules for iterating the index \( v \) are the same as applied to the encoding process, except using \( w_{\text{digit}} \) in the right
hand side:

\begin{align*}
    a_{r-1} &= 0 \\
    b_{r-1} &= 0 \\
    x_{r-1} &= a_{r-1} \oplus w_{2r-1} \oplus (b_{r-1}w_{2r-2}) \\
    y_{r-1} &= a_{r-1} \oplus w_{2r-1} \oplus (b_{r-1}w_{2r-2}) \\
    a_{k-1} &= a_k \oplus b_k \oplus w_{2k-1} \\
    b_{k-1} &= \overline{b_k} \\
    x_{k-1} &= a_{k-1} \oplus w_{2k-1} \oplus (b_{k-1}w_{2k-2}) \\
    y_{k-1} &= a_{k-1} \oplus w_{2k-1} \oplus (b_{k-1}w_{2k-2}) \\
\end{align*}

(k = r - 1, r - 2, \ldots, 1).

The order of this algorithm is 16r.

4.9 Summary

In applications of data retrieval, the Hilbert order exhibits good performance. However, its encoding and decoding processes involve many operations and the response time for interactive queries is long when the data volume is large. A fast addressing technique to point to positions of data blocks is desired. The U order is faster than the Hilbert order, but its performance is decreased somewhat. In this chapter, we proposed several new spatial orders and evaluated their performance. Among them, the Q4 order presents a simple structure. Thus it is selected as an applicable representative and its analytical encoding and decoding formulas are derived. The performance of the Q4 order has been assessed both analytically and computationally for several typical applications in spatial data processing systems. The result suggests that the
Q4 order behaves similarly to the Hilbert order.

An algorithm analysis has also been conducted to estimate the computational costs of several spatial orders and the results are listed in Table 4.6.

<table>
<thead>
<tr>
<th>Spatial Order</th>
<th>Encoding</th>
<th>Decoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>4r</td>
<td>4r</td>
</tr>
<tr>
<td>U</td>
<td>5r</td>
<td>5r</td>
</tr>
<tr>
<td>Q4</td>
<td>20r</td>
<td>16r</td>
</tr>
<tr>
<td>H</td>
<td>31r</td>
<td>19r</td>
</tr>
</tbody>
</table>

Table 4.6: Computational Costs for Resolution r

The uniform distribution of queries has been implicitly assumed throughout the foregoing discussions. For distributions that are non-uniform, the following model can be considered: given a range query that may correspond to an arbitrary polyhedron in an n-dimensional lattice system, the maximum linear span of the indices (i.e., location codes) in this polyhedron can be used as a metric to assess the scattering degree. Each range query may be associated with a probability. Therefore, the expectation and the variance can be determined given a particular spatial order. Thus the performance of spatial orders can be evaluated by comparing these statistics.

If the probability distribution changes over time, then the approach similar to the Scenario Tracking model (Chapter 5) can be applied. In general, the concerned time range can be partitioned into a series of phases, and each phase corresponds to a scenario. A set of scenario probabilities can be assigned to these phases. Then a
tracking model can be constructed to determine a spatial order that performs best over all scenarios.
Chapter 5

Optimization of the Key Resolution for Range Query Retrieval

In foregoing discussions, the resolution of the lattice systems is given and assumed to be identical for all dimensions. In this chapter, the optimization of the resolution is discussed. The analysis will concentrate on a typical application in information retrieval systems, the range query retrieval problem. The methodology and conclusion are expected to be applicable to other lattice models in data processing systems.
5.1 Overview

In modern file systems, data records are grouped into a set of sections based upon the values of attributes. A hashing function is a means of calculating the index of a section containing a given record from its key. Multiattribute hashing is an effective mechanism solving partial match query retrieval problems.

Suppose each record has \( n \) fields. In a multihashing structure, each field is associated with a distinct hashing function \( h_i \). The \( n \) hash values can be arranged according to interleaving, combination or permutation rules to generate a number. Mathematically, this is a mapping from \( n \) to 1. Usually the arrangement for the \( n \) hash values is carried out in terms of bits, rather than in terms of decimal digits. The resulting number is used as a pointer to a data section.

Interest in applying multihashing to partial match query retrieval has been high since the 1970s. The multiattribute hashing scheme was initially proposed by Rothnie and Lozano [57] as an alternative structure to invert files to improve a system’s performance. Rivest [54] analyzed the performance of non-binary attributes and discussed some hashing schemes by means of combinatorial design. Aho and Ullman [3] proposed a mathematical programming model to determine the optimal number of the bits allocated to the fields.

In a partial match query, a bit-string with \( b_i \) bits called the field signature, is generated by the hash function \( h_i \) with the specified field as its argument. The number of bits in a field signature is called field resolution and it determines the
cardinality of the hashed space. For instance, we may have:

\[
    h_{name}(x) = \begin{cases} 
    00 & \text{if } x \leq \text{"G"} \\
    01 & \text{if } \text{"G"} < x \leq \text{"N"} \\
    10 & \text{if } \text{"N"} < x \leq \text{"T"} \\
    11 & \text{others}
    \end{cases}
\]

In this example, the field resolution is 2 and the cardinality of the hashed space is 4.

The hashing process has a two-fold effect:

1. The domain of a field is subdivided into a finite set by decision thresholds, and

2. the range of the hash function is a finite discrete set of representation values.

In this chapter, the word "range" has two meanings. One is associated with range query, a concept defined in the attribute space, whereas the other stands for the set of the hashed values. The latter is used because of mathematical conventions.

For partial match query problems, Aho and Ullman [3] considered a storage space partitioned into \(2^B\) sections (\(B\) is a positive integer), each of equal size. Thus \(B\) bits are needed to determine a section address. They proposed a constrained nonlinear programming model to optimize the resolution for each field, based on the specified query probabilities. The objective function to be minimized is the expected value of data sections that need to be searched to answer a query.

Suppose that each record has \(n\) fields. Let \(p_i\) be the relative importance coefficient or weights attached to the \(i\)-th field. Let the field resolution to be determined be \(b_i\). A query \(q\) can be represented by a subset in the integer set \([1, 2, \ldots, n]\). There are \(2^n\)
such subsets. Aho and Ullman proposed the following model:

$$\begin{align*}
& \text{minimize } y = \sum_{q} [P_q \prod_{i \in q} 2^{b_i}] \\
& \text{subject to } \sum_{i=1}^{n} b_i = B
\end{align*}$$

(5.1)

where $P_q = \prod_{i \in q} p_i \prod_{i \notin q} (1 - p_i)$ and $q \subseteq \{1, 2, \ldots, n\}$.

The model characterized by (5.1) and (5.2) presents an elegant mathematical formulation for the partial match query optimization problem. This basic formulation has been developed and applied to various partial match query applications ([40], [41], [53], etc.).

Much research has been done on partial match query models. However, the literature is relatively sparse on the range query retrieval problem. We think that the optimization models for the range query problem can be nicely established by means of stochastic programming methodology. A brief introduction to stochastic programming is given in the next section.

### 5.2 Introduction to Stochastic Programming

A mathematical programming problem can be generally formulated as follows:

$$\begin{align*}
& \text{minimize } f(X) \\
& \text{subject to } g(X) \geq 0
\end{align*}$$

where $f : \mathbb{R}^n \to \mathbb{R}^1, g : \mathbb{R}^n \to \mathbb{R}^m$. $X$ is called the decision variable. $f(X)$ is called the objective function and $g(X)$ the constraint function. The feasible region of $X$ is
represented by the region enclosed by \( g(X) = 0 \). Usually, both \( f(X) \) and \( g(X) \) are expressed analytically in terms of \( X \) and a set of coefficients (e.g., price of goods, cost of transportation). Such a coefficient set is denoted \( \Omega \) in the following.

Conventional mathematical programming research deals with the optimization theory and methods on the assumption that each \( \omega \in \Omega \) is deterministic. *Stochastic programming* relaxes this assumption, namely, \( \Omega \) is regarded as a set of stochastic variables.

Tintner [64] described two kinds of problems:

- *subjective risk*: there exists a probability distribution of anticipation which is itself known with certainty, and

- *subjective uncertainty*: there is an *a priori* probability of the probability distributions themselves.

The first field leads to stochastic programming. In the literature, two classical types of stochastic programming models have been proposed ([13], [34]):

- the *wait-and-see* type (also called the passive or adaptive type), and

- the *here-and-now* type (also called the active or anticipative type).

The wait-and-see model concerns the situation in which the decision variable \( X \) is chosen after making an observation for each \( \omega \in \Omega \). In other words, the samples of realized \( \omega \) are available. The here-and-now model concerns the situation in which the decision variable \( X \) is chosen before making an observation for each \( \omega \in \Omega \). This
implies that the moments of $\omega$ can be incorporated into the formulations of $f(X)$ and $g(X)$.

The traditional tasks of the wait-and-see approach are to find the moments of the optimal solution of $X$ and/or the optimal value of $f(X)$, or, more generally, the distributions of theirs. Recently, Dembo [12] proposed an alternative task for the wait-and-see model, scenario tracking. The basic ideas can be considered as coming from original concepts of norm regulation or from the connotation of least square techniques.

Consider the following formulation of general stochastic programming problems:

$$\begin{align*}
\text{minimize} & \quad f(X, a_d, a_u) \\
\text{subject to} & \quad g(X, b_d, b_u) \geq 0
\end{align*}$$

where $f : \mathbb{R}^n \to \mathbb{R}^1, g : \mathbb{R}^n \to \mathbb{R}^m$; the subscripts $d,u$ stand for the deterministic and uncertain parameters respectively. A scenario can be defined as a particular realization of the uncertain data, $a_u$ and $b_u$, represented by $a_s$ and $b_s$. Thus, for each scenario $s \in S \equiv \{\text{set of all scenarios}\}$, the above problem reduces to the deterministic problem below, which will be referred to as the scenario subproblem:

$$\begin{align*}
y_s & \equiv \text{minimize} \quad f(X, a_d, a_s) \\
& \text{subject to} \quad g(X, b_d, b_s) \geq 0
\end{align*}$$

Corresponding to each scenario, a probability $p_s$ is assigned. In such models, a solution $X_f$ to a stochastic nonlinear system $g(X, b_d, b_s) \geq 0$ is said to be (norm) feasible if it minimizes $\sum_s p_s[|| \text{minimum} [0, g(X, b_d, b_s)] ||]$.  

167
Now, a coordination model can be constructed as:

\[
\text{minimize } \sum_{s} p_s [\| f(X, a_d, a_s) - y_s \| + \| \text{minimum}[0, g(X, b_d, b_s)] \|]
\] (5.3)

This formulation can be named \textit{tracking model} since it tracks the scenario solutions as closely as possible while still maintaining feasibility. The norm included can be the 1-norm or 2-norm.

It may be instructive to give some remarks to end this section. The essential of scenario optimization is the idea of coordination. Having realized this, we can construct diverse tracking models. In principle, any reasonable way in which scenario solutions can be combined into a single feasible solution to the underlying stochastic programming problem can be considered as a candidate. The suitable formulation of the tracking models should differ from application to application. In the following sections, we will show that the optimization models of range query retrieval can be nicely established by means of stochastic programming techniques.

5.3 The Here-and-Now Stochastic Programming Model

In applications, the domain of an attribute is not necessarily a subset in a numeral space (e.g., the integer space). It is, however, not difficult to design a transformation to map non-numeral items to numerals in a given application. Actually, this is the case that most hashing techniques assume. In the discussion to follow, therefore, this assumption is held.
Given a range query, one or more sequences of contiguous values of some fields are required. For example,

\[(25 \leq age \leq 40) \text{ and } (140 \leq weight \leq 160)\]  \hspace{1cm} (5.4)

It would be beneficial to give some interpretations to the philosophy implied by the formulation (5.1) before constructing a model for the range query problem. The objective function in (5.1) can be rewritten as follows:

\[
y = \sum_q P_q [\prod_{i \in q} 2^{b_i}]
\]

\[
= \sum_q P_q \left[ \prod_{i \in q} 2^{b_i} \prod_{i \notin q} 2^{b_i} \right]
\]

\[
= 2^B \sum_q P_q [\prod_{i \in q} 2^{-b_i}] \hspace{1cm} (5.5)
\]

Therefore, the minimization of (5.5) is similar to:

\[
\text{maximize} \quad 2^B \sum_q P_q [\prod_{i \in q} 2^{b_i}] \hspace{1cm} (5.6)
\]

Formulation (5.6) explicitly presents the intuition: the more frequently a field is specified, the more bits it should be have. In other words, the frequency \(P_q\) acts as a weight to these fields \(i \in q\). In applications, the weights may be contributed not only by frequencies but also by various other factors. For the range query problem, one of these factors is the size of range instances. Let the factor that characterizes the relative importance of range sizes be \(u_i\), which is a monotone decreasing function of the range size, then the model (5.1) becomes:

\[
\text{minimize} \quad y = \sum_q [\prod_{i \in q} u_i P_i \prod_{i \notin q} (1 - p_i) 2^{b_i}] \hspace{1cm} (5.7)
\]
Obviously the parameters \( u_i \) (\( i = 1, 2, \ldots, n \)) should be regarded as stochastic parameters for most practical range query problems. Consequently, the optimization model (5.7) is a stochastic programming model.

One major difficulty arising in range query problems is the large amount of range query instances. Let the cardinality of the domain of the \( i \)-th fields be denoted \( m_i \).

There may be two ways to classify range types. The first way considers both the size and the starting position of a range. By size it meant the cardinality of a range. For example, the size of \([2, 2]\) is 1. Consider the case that the domain is \( \{1, 2, 3, 4, 5, 6, 7, 8\} \). Then with numeral 1 as the starting point, there are \([1, 1], [1, 2], \ldots, [1, 8]\), in total eight instances. In general, with numeral \( j \) (\( 1 \leq j \leq m_i \)) as the starting point, there are altogether \( m_i - j + 1 \) instances. Thus the total number of all possible instances would be \( \sum_{j=1}^{m_i} (m_i - j + 1) = m_i(m_i + 1)/2 \). However, a query incorporating an \( m_i \)-size range specification for field \( i \) actually corresponds to unspecifying field \( i \). For instance, the following two queries are equivalent:

\[
\text{retrieve all records such that: } (25 \leq \text{age} \leq 40) \quad \text{and all weights}
\]

\[
\text{retrieve all records such that: } 25 \leq \text{age} \leq 40
\]

In other words, the \( m_i \)-size instance should be excluded from the calculation for the probability of events \( i \in q \). Consequently, the number of total instances is given by:

\[
N_i = \frac{(m_i - 1)(m_i + 2)}{2}
\]

The second way takes the size into account only. Thus there are \( m_i - 1 \) instances, each with size 1, 2, \ldots, \( m_i - 1 \), respectively.
In principle, a set of frequencies can be associated to each range instance in the attribute space. The number of them depends on the particular instance distribution. In this section, the here-and-now approach is discussed. Four frequency distributions of range types are analyzed as paradigm studies.

**Paradigm 1:**

Paradigm 1 considers the average size of ranges over all possible instances, taking both the size and the starting point into account. For field $i$, there are $m_i$ 1-size ranges, namely, $[1, 1], [2, 2], \ldots, [m_i, m_i]$. Similarly, there are $m_i - 1$ 2-size ranges, namely, $[1, 2], [2, 3], \ldots, [m_i - 1, m_i]$. In general, there are $(m_i - k + 1)$ $k$-size ranges ($k = 1, 2, \ldots, m_i - 1$). If assuming that all instances are equally likely to be specified, then the frequency of the $k$-size $(k = 1, 2, \ldots, m_i - 1)$ range is

$$f_k = \frac{m_i - k + 1}{N_i} = \frac{2(m_i - k + 1)}{(m_i - 1)(m_i + 2)}$$

i.e., the frequency follows a linear distribution. As a result, the average size of ranges is calculated as follows:

$$s_i = \sum_{k=1}^{m_i - 1} kf_k = \frac{m_i(m_i + 4)}{3(m_i + 2)} \approx \frac{m_i}{3}$$

**Paradigm 2:**

On the assumption that the range instances are uniformly distributed by con-
sidering both the size and the starting position, it has been shown that the 
frequency is a linear function of interval size $k$. Another possible situation is 
that the instances are uniformly distributed over the size only. Thus the fre-
quency is a constant against size $k$. Without the $m_i$-size interval, the frequency 
is given by

$$f_k = \frac{1}{m_i - 1} \quad (k = 1, 2, \ldots, m_i - 1)$$

and the average size is expressed as

$$s_i = \sum_{k=1}^{m_i-1} k f_k = \frac{m_i}{2}$$

Paradigm 3:

The partial match query problem has the special significance in that it has been 
regarded as a representative paradigm in applications and extensive research has 
been done. Thus it is instructive to distinguish the instances of the partial match 
query from the ones of the general range query family. Paradigm 3 considers 
the case of the uniform distribution of range instances, taking both sizes and 
starting positions into account, over the size set $k \in \{2, 3, \ldots, m_i - 1\}$. Given 
$f_1$, the following formulas are obtained:

$$f_k = \frac{2(m_i + 1 - k)}{(m_i + 1)(m_i - 2)} \quad (k = 2, \ldots, m_i - 1)$$

$$s_i = 1 \cdot f_1 + \sum_{k=2}^{m_i-1} k f_k$$
Paradigm 4:

The case of the uniform distribution of ranges about sizes only is considered in this paradigm. With $f_1$ specified, a set of formulas can be obtained as follows:

$$f_k = \frac{1}{m_i - 2} \quad (k = 2, \ldots, m_i - 1)$$

$$s_i = f_1 + \sum_{k=2}^{m_i-1} k f_k$$

$$= f_1 + \frac{1}{m_i - 2} \sum_{k=2}^{m_i-1} k$$

$$= f_1 + \frac{m_i + 1}{2}$$

At present, on the uniformity assumption, we have derived a set of mathematical expressions of the average size $s_i$. The uniformity assumption has frequently been applied in relevant studies. For example, assuming that all $2^n$ subsets for the binary reflected Gray code of order $n$ are equiprobable, Faloutsos [15] has showed that the improvement on the cluster number is up to 50% to the natural binary order. Another example based on the same assumption is the model used in industry to estimate the performance of different disk drives [59], where the average seek time for a disk is derived as the same time required to traverse one third of the tracks or cylinders. It is interesting that this result is similar to the one derived in Paradigm 1 shown above.
Table 5.1: Coefficients $a_{ki}$

To save text space, in the following we use a unified formula of $s_i$:

$$s_i = a_{1i}m_i + a_{2i} + \frac{a_{3i}}{m_i + a_{4i}}$$ (5.8)

The values of $a_{ki}$ ($k = 1, 2, 3, 4$) are given in Table 5.1.

By replacing $u_i$ by $f(s_i)$ in Eq. (5.7), where $f(s_i)$ is a monotone decreasing function of $s_i$, we have:

$$\text{minimize} \quad y = \sum \left[ \prod_{i \in q} f(s_i) p_i \prod_{i \in \bar{q}} (1 - p_i)^{2^h} \right]$$ (5.9)

The objective function $y$ in (5.9) can be rewritten by incorporating the equality constraint (5.2) as follows:

$$y = 2^B P \sum_{q} \left[ \prod_{i \in q} r_i f(s_i) 2^{-h_i} \right]$$ (5.10)

where $P = \prod_{i=1}^{n} (1 - p_i)$ and $r_i = p_i / (1 - p_i)$.

Since query $q \subseteq \{1, 2, \ldots, n\}$, the summation of $q$ is over all $2^n$ subsets. The empty product is considered unity by convention. Consequently, Eq. (5.10) becomes:

$$y = 2^B P \prod_{i=1}^{n} \left[ 1 + r_i f(s_i) 2^{-h_i} \right]$$ (5.11)
The minimization of the objective function $y$ characterized by Eq. (5.11) subject to equality constraint (5.2) can be solved by the classical Lagrangian multiplier method. To save space, the mathematical details are not presented here. The optimal solution is of closed form:

$$b^*_i = \frac{B}{n} + \log_2[r_i f(s_i)] - \frac{1}{n} \sum_{j=1}^{n} \log_2[r_j f(s_j)]$$

The optimal value of the objective function $y$ is:

$$y^* = P[2^\frac{B}{n} + \left(\prod_{j=1}^{n} r_j f(s_j)\right)^\frac{1}{n}]^n$$

### 5.4 The Wait-and-See Stochastic Programming Model

The here-and-now approach based on using the moments of the stochastic parameters is appropriate as long as analytic descriptions of the frequency distributions are available, especially for those cases with simple distributions like uniform or linear distributions, as shown in Section 5.3. In practice, however, the range size in the range query problem may be a stochastic variable with an unknown distribution. The moments are difficult or even impossible to be expressed in terms of $m_i$, the cardinality of the $i$-th field. In this section, the wait-and-see approach is investigated.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be an observation to the stochastic vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$, where $v_i$ is the range size of field $i$, and let $f(t)$ is a decreasing monotone function of
t, then model (5.7) is written as follows:

\[
\text{minimize } y = \sum_{q} \left[ \prod_{i \in q} f(x_i) p_i \prod_{i \notin q} (1 - p_i)^{2^{b_i}} \right]
\]  

(5.12)

Subject to equality constraint (5.2), the optimal solution of (5.12) can be determined by the Lagrangian multiplier method. The derivation details are ignored here to save space. The optimal solution is:

\[
b_i^* = \frac{B}{n} + \log_2(r_i) - \frac{1}{n} \sum_{j=1}^{n} \log_2(r_j) + \log_2[f(x_i)] - \frac{1}{n} \sum_{j=1}^{n} \log_2[f(x_j)]
\]  

(5.13)

Let us choose Paradigm 1 and 2 (see Section 5.3) as the representatives and use a unified formula to express their density distribution functions:

\[
d_i(t) = q_i t + c_i \quad 1 \leq t \leq m_i - 1
\]

where for the uniform distribution:

\[
q_i = 0,
\]

\[
c_i = \frac{1}{m_i - 1},
\]

and for the linear distribution:

\[
q_i = \frac{-2}{(m_i - 1)(m_i + 2)},
\]

\[
c_i = \frac{2(m_i + 1)}{(m_i - 1)(m_i + 2)}
\]

Consequently, the expected value of the optimal solution \(b_i^*\) can be calculated as follows:

\[
E(b_i^*) = \sum_{x_1=1}^{m_1-1} (q_1 x_1 + c_1) \ldots \sum_{x_i=1}^{m_i-1} [b_i^*(q_i x_i + c_i)] \ldots \sum_{x_n=1}^{m_n-1} (q_n x_n + c_n)
\]  

(5.14)
The availability of a closed form for Eq. (5.14) after substituting Eq. (5.13) into it depends on the specification of \( f(x_i) \). In general, given the density distribution function \( d_i(t) (i = 1, 2, \ldots, n) \), determining the expected value of the optimal solution may have to rely on numerical techniques to evaluate the following summation:

\[
E(b^*_i) = \sum_{x_1=1}^{m_1-1} [d_1(x_1)] \cdots \sum_{x_i=1}^{m_i-1} [b^*_i d_i(x_i)] \cdots \sum_{x_n=1}^{m_n-1} [d_n(x_n)] \tag{5.15}
\]

So far we have applied the here-and-now and wait-and-see methods to deal with the stochastic programming models arising from the optimization problem of the range query retrieval. It has been shown that both methods may involve finding the moments of stochastic variables. For the uniform or linear distributions, the here-and-now model does not involve complicated computations, while the wait-and-see model needs to evaluate a complicated summation. For this reason, the here-and-now model seems to have advantage over the wait-and-see model. Concerning applications, it appears to be appropriate to apply the here-and-now and wait-and-see methods to commercial data systems, where the assumption on the uniformity is usually justified. For engineering data systems, where diverse distributions are prevalent, the here-and-now and wait-and-see approaches may have to rely on numerical summation or integration techniques. In the next section, we discuss the third method — the scenario tracking method.
5.5 The Scenario Tracking Stochastic Programming Model

The conventional wait-and-see approach deals with the moments of the optimal solution of decision variables and/or the optimal value of the objective function, or, more generally, their distributions. Recently, the scenario tracking method was suggested to be an alternative task for the wait-and-see model [12]. The basic idea has been introduced in Section 5.2. In this section, this methodology is applied to the optimization of field resolutions.

Scenario tracking consists of two phases: the first is scenario optimization and the second is scenario coordination. Recall that a scenario is defined by a set of parameters \( (w_1, w_2, \ldots, w_n) \), a particular realization of a set of stochastic data \( (v_1, v_2, \ldots, v_n) \).

Thus model (5.12) is actually the formulation of the scenario optimization:

\[
\text{minimize} \quad y = \sum_{q} \left( \prod_{i \in q} f(w_i) p_i \prod_{i \notin q} (1 - p_i)^{2^{|i|}} \right)
\]

The optimal solution can be expressed as follows:

\[
b_i^* = \frac{B}{n} + \log_2(r_i) - \frac{1}{n} \sum_{j=1}^{n} \log_2(r_j)
\]

\[
+ \log_2[f(w_i)] - \frac{1}{n} \sum_{j=1}^{n} \log_2[f(w_j)]
\]

By substituting the above expression of \( b_i^* \) into the expression of the objective function \( y \), we obtain the optimal value:

\[
y^*_s = \{ \prod_{j=1}^{n} r_j f(w_j) \}^{\frac{1}{n}}
\]
where \( P = \prod_{i=1}^{n}(1 - p_i), r_i = p_i/(1 - p_i) \), and where the equality constraint Eq. (5.2) has been incorporated. The subscript "s" stands for scenarios.

Using the value \( y^*_s \), the scenario coordination model can be formulated as follows:

\[
\text{minimize } \quad y = \sum_s f_s \| y^*_s - P_s 2^B \prod_{i=1}^{n}(1 + f(w_{i,s})r_{i,s}2^{-b_i}) \| \quad (5.17)
\]

\[
\text{subject to } \quad \sum b_i = B \quad (5.18)
\]

The complete model may include some inequality constraints such as \( b_{i_{\text{min}}} \leq b_i \leq b_{i_{\text{max}}} \), etc.

An appropriate solution process for the optimization model characterized by (5.17) and (5.18) is constrained nonlinear programming. A set of algorithms has been developed and implemented. The description is given in the appendix.

5.6 Optimization of the Section Capacity

In the foregoing discussion on the optimization of field resolution, the number of total data sections \( 2^B \) has been assumed to be a specified parameter. In this section, we discuss how to determine the optimal value of this parameter. Let \( R \) be the number of total records and \( S \) be the capacity of each data section. Then, the following relation holds:

\[ 2^B S = R \]

Therefore, changing \( B \) implies either changing \( R \) or changing \( S \). The former concerns dynamic file systems and is not to be covered here. The latter is usually constrained
by the particular storage structure. Although this issue is machine-dependent, it would be instructive to gain some mathematical insight.

Two more machine-dependent parameters, the time to locate the data section and the time to read it, need to be specified in order to proceed with the analysis. Let them be \( L \) and \( T \), respectively. Thus, given the optimized expectation of data sections (denoted \( y^* \)), the objective function with argument \( B \) is formulated as follows:

\[
 t = (L + TR2^{-B})y^* 
\]  

(5.19)

In the following, the cases corresponding to the here-and-now and wait-and-see approaches are discussed separately:

1. The here-and-now approach

The optimal value determined by the here-and-now model is:

\[
y^* = P(2^n + C)^n
\]  

(5.20)

where \( C = \left( \prod_{j=1}^{n} r_j f(s_j) \right)^{1/n} \)

The definition of \( s_j \) was given by Eq. (5.8).

Substituting (5.20) into (5.19) leads to:

\[
t = (L + TR2^{-B})P(2^n + C)^n
\]

Let the first derivative be zero, then we obtain the optimal solution of \( B \):

\[
B^* = \left( \frac{n}{n+1} \right) \log_2 \left( \frac{CRT}{L} \right)
\]
2. The wait-and-see approach

The optimal value determined by the wait-and-see model is:

\[ y^*_n = P\{2^n + \prod_{j=1}^{n} r_j f(x_j)\}^\frac{1}{n} \]  \hspace{1cm} (5.21)

The meaning of \( x_j \) was described in Section 5.4.

Substituting (5.21) into (5.19) leads to:

\[ t = (L + TR2^{-B}) P\{2^n + \prod_{j=1}^{n} r_j f(x_j)\}^\frac{1}{n} \]

Let the first derivative be zero, then we obtain the optimal solution of \( B \):

\[ B^* = \left( \frac{1}{n+1} \right) \{ n \log_2 \left( \frac{RT}{L} \right) + \sum_{j=1}^{n} \log_2 (r_j) + \sum_{j=1}^{n} \log_2 [f(x_j)] \} \]

Then its moment can be determined through the process similar to that described in Section 5.4.

In this section, we have briefly discussed how to optimize the capacity of data sections by means of the here-and-now and wait-and-see stochastic programming methods. The ST method is also applicable. In addition, the dynamic file model can be analyzed similarly. These topics are not to be discussed further, since the main subject of this section is on the optimization of field resolutions. The analytic methodology used in this section was initially described by Aho and Ullman [3] and was applied to the partial match retrieval problem.
5.7 Generalization to Uncertainty Models

The uncertainty problems concern the situation in which there is an \textit{a priori} probability of the probability distributions themselves [64]. Conventional stochastic programming techniques do not deal with models related to such problems. Consider the following situation: There are several scenarios for the probability distributions. In scenario 1, the range distribution of field $j$ follows the linear distribution; in scenario 2, it follows the uniform distribution; etc.

By specifying a set of scenario probabilities, we can formulate a tracking model for this problem:

\[
\begin{align*}
\text{minimize} & \quad \sum_{s} f_s \parallel \left\{ 2^B P \sum_{q \in Q} \prod_{i \in q} r_{i,s} f(s_{i,s}) 2^{-b_i} \right\} \\
& \quad - \left\{ P \left[ 2^B + \prod_{j=1}^{n} r_{j,s} f(s_{j,s}) \right]^\frac{n}{\hat{n}} \right\} \parallel \\
\text{subject to} & \quad \sum_{i=1}^{n} b_i = B
\end{align*}
\]  

(5.22)  

(5.23)

The model characterized by (5.22) and (5.23) provide an adaptive approach to solve uncertainty problems arising from range query retrievals.

5.8 Summary

Multiattribute hashing is an effective method dealing with partial match query retrievals. In this chapter, optimization models for the range query retrievals are proposed by incorporating stochastic programming methodology. Three approaches are discussed. The here-and-now and the wait-and-see approaches seem to be applica-
ble to the situation in which range queries follow a relative simple distribution. For the reason of computational complexity, the former seems to have advantage over the latter. Four paradigms are analyzed. The distribution models introduced have frequently been found in other relevant studies in computer sciences. One of them leads to a result that is similar to the one used in industry to evaluate the performance of disk drives. The scenario tracking approach appears to be more flexible and be appropriate for the problems where the probability distribution is not available. It also provides an adaptive technique to formulate non-conventional stochastic programming problems. A stochastic programming software package is developed and implemented. The description is given in the Appendix. Numerical experiments produced the desired results.

Future research may involve stochastic programming applied to range query problems with general linear or nonlinear constraints.
Chapter 6

Optimization on the Bucket Distribution

6.1 Overview

The performance of the multiattribute hashing system depends on how to order the data units on disks. In [15], Faloutsos proposed the following task: given a set of buckets containing records, determine their optimal or sub-optimal distribution on disks to minimize the retrieval time. With combinatorial analysis, he showed that the order determined by the Gray code can significantly improve the cluster distribution of the records. A major open question that remains is the lack of symmetric Gray codes to reduce the bias introduced by the reflected Gray code.

In the model concerned, the multiattribute hashing scheme is applied to each record to generate an integer, called the record signature. The buckets are determined by a partition for the domain of record signatures. A cluster for a given query is a set
of consecutive, qualifying buckets, surrounded by non-qualifying ones. The number of clusters is considered the performance metric.

The model is based on the following assumptions:
- Each bucket occupies exactly one disk block;
- the buckets are stored consecutively;
- the overflow problem is not considered.

A query can be represented by an \(n\)-tuple \((s_1, s_2, \ldots, s_n)\), where \(s_i \in \{0, 1, u\}\) and is constructed by the multiattribute hashing function. Such a tuple is called a query signature. The element \(u\) stands for “unspecified item”. For instance, we may have a query signature like “\(uu10\)” for \(n = 4\); thus four buckets with the signature 0010, 0110, 1010 and 1110. The allocation by the binary code introduces four clusters, while the reflected Gray code introduces only two (see below).

<table>
<thead>
<tr>
<th>binary code</th>
<th>reflected Gray code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>0001</td>
<td>0001</td>
</tr>
<tr>
<td>0010</td>
<td>0011</td>
</tr>
<tr>
<td>0011</td>
<td>0010</td>
</tr>
<tr>
<td>0100</td>
<td>0110</td>
</tr>
<tr>
<td>0101</td>
<td>0111</td>
</tr>
<tr>
<td>0110</td>
<td>0101</td>
</tr>
<tr>
<td>0111</td>
<td>0100</td>
</tr>
<tr>
<td>1000</td>
<td>1100</td>
</tr>
<tr>
<td>1001</td>
<td>1101</td>
</tr>
<tr>
<td>1010</td>
<td>1111</td>
</tr>
<tr>
<td>1011</td>
<td>1110</td>
</tr>
<tr>
<td>1100</td>
<td>1010</td>
</tr>
<tr>
<td>1101</td>
<td>1011</td>
</tr>
<tr>
<td>1110</td>
<td>1001</td>
</tr>
<tr>
<td>1111</td>
<td>1000</td>
</tr>
</tbody>
</table>

Assuming that all \(2^n\) subsets for the binary reflected Gray code with order \(n\) are
equiprobable, Faloutsos [15] has showed that the improvement on the cluster number is up to 50% to the natural binary order.

In the model discussed, the number of clusters determined by the reflected Gray code introduces an obvious bias over the attribute sets. For instance, the cluster distribution from left to right is (2, 2, 4, 8) for \( n = 4 \). Therefore, other Gray codes with a better distribution need to be investigated.

Since this topic involves some open questions in graph theory, a brief introduction may be helpful to explain the complexity.

The hypercube \( Q_n \) is an undirected regular graph \( G = (V, E) \) with \( |V| = 2^n \) and \( |E| = n2^{n-1} \). \( Q_n \) can be labeled or unlabeled. In the former case, each vertex is labeled with a bit string \( (b_1 b_2 \ldots b_n) \), where \( b_i \in \{0, 1\} \). Two vertices of \( Q_n \) are adjacent if and only if their labels differ in exactly one bit. A Hamiltonian path is a path that passes through every vertex in \( Q_n \) exactly once. A Hamiltonian cycle \( H_n \) is a closed Hamiltonian path. Thus, mathematically, the distinct types of Gray codes are equivalent to the Hamiltonian problems in the hypercube \( Q_n \).

The research on the Hamiltonicity of the hypercubes was initiated by Gilbert [23]. Based on the algebraic group theory, Gilbert classified the types of \( H_n \) \( (n > 2) \) into a finite number of classes:

1. Composite class,

   (a) ultracomposite subclass,

   (b) non-ultracomposite subclass,
2. non-composite class.

By enumeration, Gilbert has shown that there are nine distinct types of unlabeled $H_4$. Of them, five types are ultracomposite, including the reflected Gray code path, and three are non-ultracomposite. The last one is non-composite, which happens to present an even distribution of clusters, namely, (4, 4, 4, 4):

\[
\begin{array}{cccc}
g_3 & g_2 & g_1 & g_0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 
\end{array}
\]

In Gilbert's work, the sequence of the coordinate places in which the changes occur is used as a compact notation to specify a particular h-cycle. For instance, if the left-most bit is specified as the first bit, then the coordinate sequence

\[4 3 4 2 1 4 1 3 1 2 3 4 3 2 1 2\]
represents the above h-cycle.

In modern database systems, a moderate size of the secondary key retrieval problems usually needs \( n \) being above 4. Thus the \( H_n (n > 4) \) that present an even distribution of clusters (symmetric \( H_n \) in brief) deserve to be studied. Gilbert proposed a set of necessary and sufficient conditions (although not completely constructive) for the ultracomposite subclass as well as a set of sufficient conditions for the non-composite class. It is quite obvious from the geometric view, however, that symmetric \( H_n \) have to be a subclass of non-composite class. Several years later, Mills [43] reported a symmetric \( H_5 \) and proved that there exists a new class of \( H_n \) for \( n > 4 \) such that \( H_n \) does not traverse any \( Q_k (k = 2, \ldots, n - 1) \). We will prove, however, that his construction is neither necessary nor sufficient for symmetric \( H_6 \). Recently, Faloutsos has also mentioned that systematic ways for designing symmetric \( H_n \) for \( n > 4 \) are not known [15]. It appears that the latest development relevant to this topic is reported in [28]. Some impressive results are presented in Table 6.1, where \( h_n \) and \( H_n \) represent the Hamiltonian cycle in the labeled and unlabeled \( Q_n \), respectively. The analysis for \( n > 5 \) has not been reported, especially on the analysis of symmetrical \( H_n \). Thus how to construct systematically symmetric \( H_n \) is still unsolved.

From reviewing the literature of mathematics, we are aware that the Hamiltonicity problem of the hypercubes may lead to nontrivial graph-theoretic topics. Paying too much attention to the theoretical mathematical issues is beyond the scope of the proposed project. In this chapter, a heuristic approach to produce a set of high-dimensional symmetric Gray codes is presented. By constructing special non-
Table 6.1: Number of Hamiltonian Cycles in Hypercubes

<table>
<thead>
<tr>
<th>n</th>
<th>number of $H_n$</th>
<th>number of $h_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>1344</td>
</tr>
<tr>
<td>5</td>
<td>275065</td>
<td>906545760</td>
</tr>
</tbody>
</table>

composite $H_n$ with a few local adjustments, we will derive symmetric $H_n$ in the 5, 6 and 7-dimensional hypercubes. Our results show that Mills' construction is neither necessary nor sufficient for symmetric $H_n$ ($n = 5, 6, 7$).

6.2 Gray Coordinate Lattice and Symmetric $H_n$

The Gray coordinate lattice is a two-dimensional lattice with the coordinates indexed according to the order determined by a Gray code. We begin the discussion by considering the symmetric $H_4$ proposed in [23]. Its path is shown in Fig. 6.1 in the Gray coordinate system.

The planar graphical representation of $H_4$ may suggest the configuration of $H_6$, since the six bits of the labels can be evenly assigned to two coordinate axes, three for each. Based on this intuition, a developed path is constructed as shown in Fig. 6.2. This path represents a six-dimensional Hamiltonian path in $Q_6$ and the corresponding
Gray code can be written as follows:

<table>
<thead>
<tr>
<th>$g_5$</th>
<th>$g_4$</th>
<th>$g_3$</th>
<th>$g_2$</th>
<th>$g_1$</th>
<th>$g_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
1 1 1 1 0 0
1 1 0 1 0 0
0 1 0 1 0 0
0 1 1 1 0 0
0 0 1 1 0 0
0 0 1 1 0 1
0 0 1 1 1 1
0 0 1 1 1 0
0 0 1 0 1 0
0 0 1 0 1 1
0 1 1 0 1 1
0 1 0 0 0 1
1 1 0 0 1 1
1 1 1 0 1 1
1 1 1 0 1 0
1 1 1 1 1 0
1 1 1 1 1 1
1 1 1 0 1 1
0 1 0 1 1 1
0 1 0 1 1 1
0 1 0 1 1 0
1 1 0 0 1 1
1 1 0 0 1 0
0 1 0 0 1 0
0 1 1 0 1 0
0 1 1 1 1 0
0 1 1 1 1 1
0 1 1 1 1 0
1 1 1 0 0 1
0 1 0 1 1 1
0 1 0 1 1 0
1 1 0 1 0 0
1 1 1 1 0 1
1 1 1 1 0 1
1 0 1 1 0 1
1 0 1 1 1 1
```
The Gray code shown above has a much better cluster distribution (8, 8, 16, 8, 8,

\begin{verbatim}
1 0 1 1 1 0
1 0 1 0 1 0
1 0 1 0 1 1
1 0 1 0 0 1
1 1 1 0 0 1
1 1 0 0 0 1
0 1 0 0 0 1
0 1 1 0 0 1
0 0 1 0 0 1
0 0 0 0 0 1
0 0 0 0 1 1
0 0 0 0 1 0
0 0 0 1 1 0
0 0 1 1 1 1
0 0 0 1 0 1
0 0 0 1 0 0
0 0 1 0 1 0
0 0 1 0 1 1
0 0 0 1 1 1
0 0 0 1 0 0
0 0 0 1 0 0
\end{verbatim}

Figure 6.2: Approximate Symmetric $H_6$
than the reflected Gray code that has a distribution (2, 2, 4, 8, 16, 32). A further improvement is achieved by conducting a few local adjustment only, since the original path exhibits an approximate symmetry. By rearranging four pairs of rows, we obtain a symmetrical Gray code with distribution (10, 10, 12, 10, 10, 12). The coordinate sequence corresponding to this Gray code is written as follows:

\[
\begin{bmatrix}
3 & 2 & 3 & 1 & 3 & 2 & 3 & 6 & 5 & 6 & 4 & 5 & 3 & 2 & 3 & 1 \\
3 & 2 & 6 & 5 & 6 & 4 & 6 & 2 & 3 & 1 & 3 & 6 & 4 & 6 & 3 & 2 \\
5 & 2 & 1 & 5 & 6 & 1 & 4 & 1 & 3 & 4 & 6 & 5 & 1 & 2 & 5 & 6 \\
4 & 6 & 5 & 1 & 2 & 1 & 3 & 1 & 2 & 4 & 5 & 4 & 6 & 4 & 5 & 4
\end{bmatrix}
\]

A similar approach can be applied to construct the balanced Hamiltonian cycles in five or seven-dimensional hypercubes. Since five or seven bits can not be separated into two parts evenly, the resulted graphical representations are not squares (Fig. 6.3, Fig. 6.4).

![Approximate Symmetric H₅](image)

Figure 6.3: Approximate Symmetric $H_5$

The Gray code shown in Fig. 6.3 gives a cluster distribution (6, 8, 4, 6, 8) and its
The coordinate sequence is written as follows:

\[
\begin{bmatrix}
2 & 1 & 2 & 5 & 3 & 4 & 3 & 5 & 3 & 4 & 2 & 1 & 2 & 5 & 2 & 1 \\
4 & 1 & 2 & 5 & 3 & 2 & 3 & 1 & 3 & 5 & 4 & 1 & 4 & 2 & 4 & 5
\end{bmatrix}
\]

The Gray code shown above has a much better cluster distribution than the reflected Gray code that has a distribution \((2, 2, 4, 8, 16)\). To obtain the final symmetric Gray code with distribution \((6, 6, 6, 6, 8)\), only one pair of rows needs to be rearranged.

The Gray code shown in Fig. 6.4 gives a cluster distribution \((12, 16, 30, 6, 12, 16, 36)\). It is better than than the reflected Gray code that has a distribution \((2, 2, 4, 8, 16, 32, 64)\) and even further improvement is possible. The final distribution is \((18, 18, 18, 18, 18, 18, 20)\) and the coordinate sequence corresponding to the adjusted code is written as follows:

\[
\begin{bmatrix}
2 & 1 & 3 & 1 & 2 & 1 & 3 & 5 & 4 & 6 & 4 & 5 & 4 & 7 & 5 & 4 \\
5 & 6 & 5 & 4 & 5 & 7 & 2 & 1 & 3 & 1 & 2 & 1 & 3 & 7 & 5 & 6 \\
5 & 7 & 3 & 2 & 3 & 1 & 5 & 1 & 3 & 2 & 3 & 6 & 4 & 3 & 2 & 3 \\
4 & 1 & 4 & 6 & 5 & 1 & 5 & 7 & 1 & 6 & 1 & 3 & 2 & 3 & 6 & 7 \\
5 & 7 & 3 & 7 & 5 & 7 & 2 & 7 & 5 & 7 & 3 & 1 & 3 & 7 & 4 & 5 \\
4 & 7 & 6 & 7 & 4 & 7 & 1 & 7 & 2 & 7 & 6 & 2 & 3 & 6 & 1 & 6 \\
5 & 6 & 4 & 1 & 4 & 6 & 3 & 2 & 6 & 2 & 1 & 2 & 6 & 2 & 5 & 2 \\
4 & 6 & 4 & 7 & 4 & 6 & 4 & 5 & 4 & 7 & 6 & 2 & 1 & 2 & 3 & 7
\end{bmatrix}
\]

The procedures for local adjusting are better demonstrated by the developed code lines. To save space, the detail is omitted.

In this section, we have derived three balanced Hamiltonian cycles, for the 5, 6 and 7-dimensional hypercubes, respectively. The basic method is to construct a special non-composite h-cycle, followed by local adjustments. Finally, it should be mentioned that the constructed Hamiltonian cycles traverse \(Q_2\), due to the following lemma [43]:
Lemma 6.2.1 Let $P_j, P_{j+1}, P_{j+2},$ and $P_{j+3}$ be four consecutive vertices of a Hamiltonian cycle in $Q_n$. Let the coordinate sequence corresponding to the Hamiltonian cycle be $(A_1, \ldots, A_{2^n})$. Then $P_j, P_{j+1}, P_{j+2},$ and $P_{j+3}$ are the vertices of $Q_2$ if and only if $A_j = A_{j+2}$.

On the other hand, by starting from Mills' symmetric $H_6$ [43] and following Mills' construction procedure, an $H_6$ with distribution $(9, 12, 11, 14, 14, 4)$ resulted. Therefore, the subclass proposed by Mills is neither necessary nor sufficient for symmetric $H_n$.

![Figure 6.4: Approximate Symmetric $H_7$](image)

6.3 Summary

The symmetric Gray codes or the symmetric Hamiltonian paths in a high-dimensional hypercube can be applied to improve the performance of information retrieval sys-
tems. The Hamiltonian path in the four-dimensional hypercube has been reported in the literature. In this chapter, a heuristic approach is proposed to construct the the symmetric Hamiltonian paths in the five-, six-, and seven-dimensional hypercubes. The same methodology may be applied to the n-dimensional \((n \geq 8)\) cases.
Chapter 7

Conclusion

The requirement of ordering a set of multiattribute data arises frequently from spatial index processing and secondary key retrieval in modern information systems. The first application involves developing a single numerical index on a one-dimensional line for each point in multi-dimensional space, such that the spatial localizability can be preserved as best as possible. The two-dimensional Hilbert order has attained intensive interest in the literature due to its desirable performance. The lack of inexpensive encoding and decoding algorithms is mentioned several times in publications. Especially, analytical formulas have not been reported.

In this thesis, we propose analytical encoding and decoding formulas for the two-dimensional Hilbert order. The execution-time assessment suggests that they are faster than a look-up-table approach published previously [39]. Then, the encoding and decoding algorithms for the three-dimensional Hilbert order are discussed. An approach based on the rotational matrices is presented. Since a large number of
Boolean variables is involved, the Karnaugh-map technique is inappropriate to derive irreducible analytical formulas. Other systematic methods may lead to extensive mathematical operations. Therefore, for the three-dimensional case, it appears that the matrix approach is an acceptable method.

The encoding and decoding processes for n-dimensional Hilbert orders \((n > 4)\) would be considerably complicated and place a heavy computational burden on the application problems. Thus a new spatial order called the \(U\) order is proposed. Its major advantage is the significant simplicity of the encoding and decoding operations for multi-dimensional data. Performance assessments suggest that the \(U\) order is comparable with the Hilbert order and superior to many other orders. A performance analysis is followed by a formal mathematical description for the \(n\)-dimensional \(U\) order. Then a set of analytical encoding and decoding formulas are presented.

Besides the \(U\) order, several new spatial orders with a quadrant-recursive structure are also investigated. The goal is to find the candidates that are competitive with the Hilbert order in performance and with relative simple encoding and/or decoding formulas. Of these new orders, the \(Q4\) order behaves best and its performance is very close to that of the Hilbert order. An analysis shows that its encoding and decoding algorithms only need 64.5% and 84.2% of operations of the corresponding algorithms of the Hilbert order.

The second application, secondary key retrieval, involves two issues: determining the resolution for each dimension in a multi-dimensional hashed space and ordering data blocks on disks. Aho and Ullman proposed a constrained nonlinear programming
Faloutsos showed that the order determined by the reflected Gray code can significantly improve the cluster distribution of the data blocks. It is recommended to design symmetric Gray codes to reduce the bias introduced by the reflected Gray code.

In this thesis, optimization models for the range query retrievals are presented by incorporating a stochastic programming methodology. Three approaches are discussed. The here-and-now and the wait-and-see approaches seem to be applicable to the situation in which range queries follow a relative simple distribution. Four typical paradigms are analyzed. One of them leads to a result that is similar to the one used in industry to evaluate the performance of disk drives. The scenario tracking approach appears to be more flexible and more appropriate for the problems where the probability distribution is not available. It also provides an adaptive technique to formulate non-conventional stochastic programming problems. Then a stochastic programming software package is developed and implemented. Numerical experiments show the coordinating effect of the scenario tracking methodology.

Finally, the symmetric Hamiltonian paths in a high-dimensional hypercube are discussed. A heuristic approach is applied to construct the symmetric Hamiltonian paths in the five, six, and seven-dimensional hypercubes. They are equivalent to the symmetrical Gray codes.

In summary, in this thesis three major approaches applied to the ordering of multiattribute data are proposed: a set of analytical encoding and decoding formulas for the two-dimensional Hilbert order, two new spatial orders with significant simple
encoding and decoding formulas, and a set of optimization models for the range query problem.

The following issues are recommended as future research topics:

1. Quadtree and octree data structures based on the U order and the Q orders;

2. digital halftoning techniques based on the X order;

3. stochastic programming models of range query problems with general linear or nonlinear constraints.
Bibliography


Appendix A

Implementation of Stochastic Programming Software and Numerical Experiments

The general formulation for a nonlinear optimization model can be expressed as follows:

\[
\begin{align*}
\text{minimize} & \quad f(X) \\
\text{subject to} & \quad g(X) \geq 0
\end{align*}
\]

where \( f : \mathbb{R}^n \rightarrow \mathbb{R}^1 \), \( g : \mathbb{R}^n \rightarrow \mathbb{R}^m \). \( f(X) \) is called the objective function and \( g(X) \) the constraint function. The feasible region of \( X \) is represented by the region enclosed by \( g(X) = 0 \).

Most effective methods for solving the optimization model above are known as indirect methods in that the constrained problem is transformed to an unconstrained
The solving methodology consists of two levels:

- The technique of converting the constrained formulation to an unconstrained one, and

- the technique of minimizing the unconstrained problem.

One of the widely applied methods applied to level 1 is known as the exterior penalty function method ([5], [20]). The major advantage of the exterior penalty function method is that an infeasible starting point $X_0$ is allowed. It has, however, an inherent weakness. The value of the penalty factor, an iterated parameter, may become very large to force the optimization process to converge. To overcome this disadvantage of the exterior penalty function method, Hestenes [29] and Powell [49] independently proposed the augmented Lagrangian multiplier method. Later, Rockafellar [55] adapted the augmented Lagrangian multiplier method for inequality constrained problems and established some general computational properties.

In augmented Lagrangian multiplier method, the augmented objective function is formulated as:

$$L(X, r, h) = f(X) + r \sum_{i=1}^{n} \| \text{minimum}[0, g_i(X) + h_i/r] \|$$

where $r$ and $h_i$ are the penalty factor and the multiplier, respectively.

The augmented Lagrangian multiplier method shares with the exterior penalty function method the property that an infeasible starting point is allowed but does not suffer the problems the exterior penalty function method exhibits. As a relative new technique in optimization, augmented Lagrangian multiplier method has not
yet been widely applied to solve practical problems [38]. In this thesis, augmented Lagrangian multiplier method is implemented to solve the range query optimization model.

At level 2 of the solving process, the conjugate direction method ([5], [20]) is implemented. The conjugate direction method has obtained a good reputation regarding its robustness for solving complex nonlinear problems. It is expected to be appropriate for the range query optimization model because of the basic product formulation in the model.

The conjugate direction method requires searching the optimal points along several directions in the n-dimensional space. The parabolic interpolation method is chosen for carrying out this task as it exhibits good performance when incorporated in the conjugate direction method. The three major numerical algorithms described above as well as a few working subroutines are implemented to form a compact software package. Fortran is used as the implementation language since it is appropriate for scientific calculations.

The optimization package consists of a main program and six subroutines: ALMM, CDM, PIM, XSTEP, RE, and STOCH. A brief description follows:

1. STOCH — The subroutine incorporating stochastic parameters into the constructions of the objective function, the constraint function(s), and the argumented objective function.

2. XSTEP — The subroutine iterating the argument X and calling STOCH.

3. RE — The subroutine swapping data. It is a working routine to simplify the
program structure.

4. **PIM** — The subroutine executing the parabolic interpolation process and calling **XSTEP** and **RE**.

5. **CDM** — The subroutine performing the unconstrained minimization. Specifically, its functions are:
   
   (a) Call **PIM** to seek the optimal points along the initial set of conjugate directions;
   
   (b) after searching \(n\) directions, construct the \((n + 1)\)-th direction and use Powell’s criterion to determine if it is worthwhile choose this new direction to replace one of the previous directions;
   
   (c) check the convergence condition.

6. **ALMM** — the subroutine performing the augmented Lagrangian multiplier method. Specifically, its functions are:
   
   (a) Construct the initial set of conjugate directions;
   
   (b) call **STOCH** to calculate the objective function’s value at the starting point to estimate the base value used for normalization;
   
   (c) call **CDM**;
   
   (d) check the convergence condition.

The structure of the software is shown in Fig. A.1.
Numerical experiments were conducted on a SPARC Station IPX. A typical result is presented below. The total number of sections is specified to be $2^{20}$. The number of fields chosen is 5 (i.e., $n = 5$). Three scenarios are considered. The corresponding input data are recorded in Table A.1 and A.2.

The optimal values of the single scenario model (Eq. (5.16)), $y^*_s$, are presented in Table A.3, where $f_s$ is the specified probability of scenario $s$.

The optimal value obtained by minimizing the scenario tracking model (5.17) is $y^* = 251389$. The compensated amounts are 23.8%, -14.7%, and -21.0% for scenario 1, 2, and 3, respectively.
<table>
<thead>
<tr>
<th>Scenario</th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
<th>$p_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8</td>
<td>0.1</td>
<td>0.9</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
<td>0.2</td>
<td>0.8</td>
<td>0.1</td>
<td>0.4</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>0.2</td>
<td>0.7</td>
<td>0.2</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table A.1: Input Data Set 1: Frequencies of Fields to be Specified

<table>
<thead>
<tr>
<th>Scenario</th>
<th>$u_1$</th>
<th>$u_2$</th>
<th>$u_3$</th>
<th>$u_4$</th>
<th>$u_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.7</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>0.8</td>
<td>0.4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table A.2: Input Data Set 2: Weights of Range Queries

<table>
<thead>
<tr>
<th>Scenario</th>
<th>$f_s$</th>
<th>$y_s^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
<td>203068</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>294571</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>318367</td>
</tr>
</tbody>
</table>

Table A.3: Optimal Values of the Single Scenario Model