A New Data Structure and Algorithm for Spatial Network Representation

by

FUNG Tze Wa

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Philosophy

In

Department of Systems Engineering and Engineering Management

©The Chinese University of Hong Kong
June 30, 2003

Abstract

Most existing spatial data formats are proprietary. This makes it difficult to develop new spatial algorithms requiring low level data access for real spatial data. To deal with the problem, this thesis applies Geography Markup Language (GML), a variant of eXtensible Markup Language (XML) for real spatial data. We suggest an indexing scheme for GML spatial data.

A spatial network is one type of spatial data with network characteristics. Although a spatial network has many applications, little literature specifically studies the data structure of spatial network. We propose an algorithm, called Connected Page Algorithm (CPA), which is used to cluster a spatial network. In our experiments we find that we can save unnecessary calculations in network operations such as shortest path problems in this clustered network. In addition, we try to extend the CPA and propose an algorithm called Nearest Neighbor in Connected Page Algorithm (NNCPA).

摘要

大多數的空間數據結構是專利的，引致利用真實的空間數據(spatial data)作發展新的的空間演算法(spatial algorithms)出現很多困難。本篇論文使用地理標記語言(Geography Markup Language, GML) ——一種可擴充標記語言(eXtensible Markup Language, XML)於空間數據的應用。我們提出一個索引方式於地理標記語言形式的空間數據。

空間網絡為其中一種空間數據而有網絡的特質。空間網絡有多方面的應用，但針對空間網絡數據結構的研究則不多。我們提出一個數據演算法名為連接頁演算法(Connected Page Algorithm, CPA)以群集空間網絡。我們的實驗發現很多的網絡計算如最短路徑問題(Shortest Path Problem)能省去於這群集網絡。另外，我們嘗試擴展連接頁演算法之應用於最近鄰查詢(Nearest Neighbor Queries)。

Acknowledgements

During my M. Phil study I have had many people's support.

First of all, I would like to show my gratitude to Professors C.H. Cheng and Janny M.Y. Leung. Without their guidance, this thesis cannot be completed. Also I would like to thank all professors, administrative and technical staff in the Department of Systems Engineering and Engineering Management. They provide a good environment and atmosphere for learning and research works here.

Next, I would like to say special thanks to Jacky Wong, Ada Ng, Lam Ngok, Ah Kin, Ah Wah since they have helped me a lot in my research work. Also many people have expressed their concern for me including Ida Hui, Angie Lim, Gabriel Fung, Fiona Choi, Ah Fan and here I would like to say thanks to all of them.

Last but not the least, I must thank to my parents for their nourishment and support.

Fung Tze Wa

June, 2003

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Technology in Relational Database Management Systems (DBMS or RDBMS) has developed rapidly in the past 20 years. With the advancement of database technology and algorithms, current database systems can handle extremely large amount of data and become the core of information systems.

Out of many database implementations, relational databases by far are the most widely used in the industry. A Relational Database is very efficient in storing and retrieving "relational" data, for example, account balance and transactions of customers in a banking company, invoice and customer information in a logistics company, etc. The application of database technology has been extended to Data Mining, Spatial Databases, Multimedia Databases, etc.

The capabilities of Spatial Databases have been enhanced from an traditional approach to store spatial data just for geographic information systems (GIS) to many real world applications such as web-based digital map access, map display in Wireless Application Protocol (WAP), i-Mode and General Packet Radio Service (GPRS) phone or personal digital assistant (PDA).

Although the application of spatial databases has become common and is able to satisfy the access needs of most spatial data users, request improvement in many areas, particularly in spatial indexing mechanism, is still needed. Most of the spatial data formats are proprietary, (i.e., a data format structure is not made available to the public and data in a certain format can only be used by the corresponding software). This limits the advancement in spatial data storage and also in the spatial indexing method needed to fulfill individual user's needs. To solve this problem, we will use eXtensible Markup Language (XML) format for spatial data storage and retrieval.

Based on XML, we build spatial data and spatial databases. Related issues such as indexing methods of spatial data, query in spatial networks and nearest neighbor queries are discussed.

1.2 Motivation

To demonstrate the motivation of our research, let us consider one of the most popular applications of spatial data – web-based digital map access. The most comprehensive web-based digital maps in Hong Kong are www.ypmap.com and www.centamap.com.

These two web sites have comprehensive map and related map and other commercial information for users to search. Although the users may find information on the websites abundant, they often find that the response time is exceptionally long.

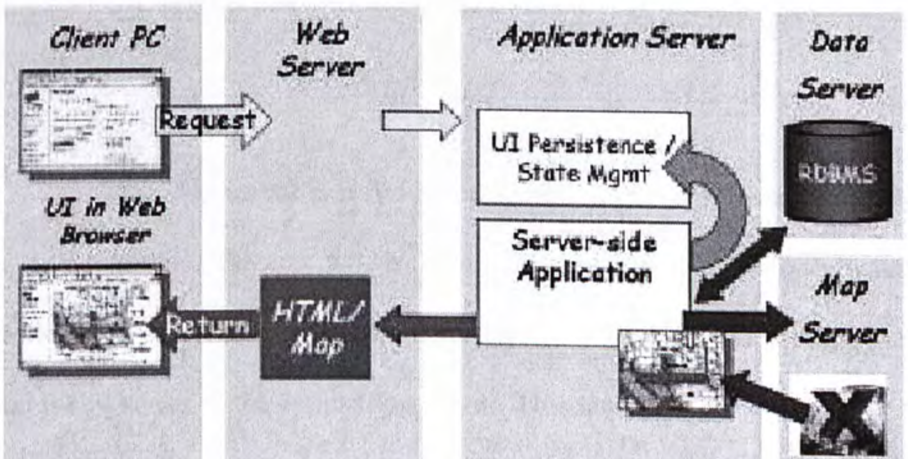


Fig. 1.1: Server side architecture of web server which support digital map [Mapinfo Corporation (2002a)]

Fig. 1.1 is a server side architecture of MapXtreme, a product from Mapinfo Corporation which supports web-based digital map applications development. This server side architecture is adapted for most web-based digital map servers. It is mainly divided into the user-needed information between two servers. Non-spatial data, such as building name and address, are stored in a database server. Spatial data which is usually in a proprietary format, such as building polygon, are stored in a map server.

This architecture has several problems, although it is commonly used:

1. The Map Server and Database Server are independent. Simple spatial queries, for example, locating the map and address of a building by the building name, must involve the map server, database server and also application server.

2. The Map server is a file-based server. It can only return the whole layer that contains the area which the user requests. If the user only wants a small region for information from many layers, the map server would have to send many layers to the application server. This increases the workload of the servers and explains why the response time is often long.

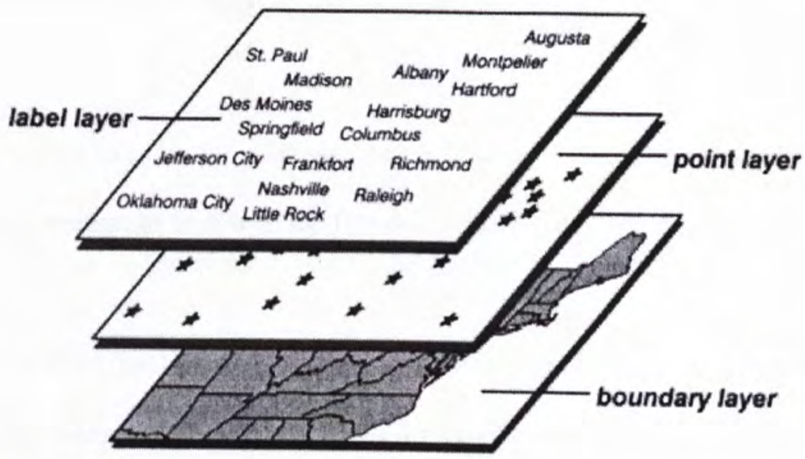


Fig. 1.2: Computer maps are organized into layers. You may treat a layer as a transparency. Each transparency contains different aspects of the whole map. Transparencies are stacked on top of one another to provide needed information. [Mapinfo Corporation (2002b)]

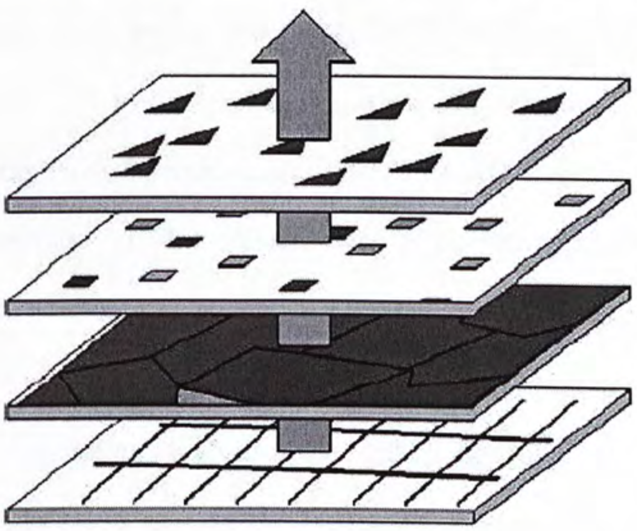


Fig. 1.3: By stacking these layers one on top of the other, a complete map can be built. [Mapinfo Corporation (2002b)]

1.3 Purposes of this Research

Our objective is to increase the efficiency of spatial databases. Specially, we attempt to deal with the following issues:

1. Since most spatial data formats are binary and proprietary, many spatial indexing mechanism cannot directly be used in existing and real spatial data, for example, digital map data from Lands Department, HKSAR Government, (http://www.info.gov.hk/landsd/) (Lands Department only provide .e00, ASCII, DGN and DXF which can only open by ESRI (http://www.esri.com), Integraph (http://www.intergraph.com/) or AutoCAD (http://www.autocad.com) products. For TIFF format, it can be opened by any imaging software but it is only a graphic, not spatial data.) This thesis would try to convert those real spatial data to Geography Markup Language (*GML*), an extension of eXtensible Markup Language (*XML*), so that the existing and new spatial indexing mechanism can be built on real spatial data.

2. The characteristics of XML are openness and vendor-neutrality. It is in text format and self-explanatory. So GML spatial data can be stored in any file or database systems. Therefore, when the user would need to

query a GML map with address, i.e. spatial and non-spatial simultaneously, only one database system is needed.

3. For applications (such as finding Shortest Path, solving Vehicle Routing Problem, planning electricity, water and drainage systems, etc) in Hong Kong, graph representation (i.e. node and arc) is needed. Although these graph information is not available from Lands Department, HKSAR Government, these graph information(e.g. road network in Hong Kong) can be retrieved by Geography Markup Language (*GML*). Other problems, such as Nearest Neighbor Problem, can also be studied in GML spatial data.

1.4 Contribution of this Research

This thesis mainly focuses on XML indexing for spatial data. To the best of our knowledge, this is the first work to deal with this issue. Our work contributes to the literature in the following ways:

1. Through eXtensible Stylesheet Language (XSL), XML documents may be transformed to a format suitable for display in different devices, such as web, Personal Digital Assistant (PDA) and mobile phone. Most existing application servers can only handle one device. The reason is that different devices have different displays, resolutions and hardware

limitations. (For example, Wireless Application Protocol (WAP) phone can only receive several KiloBytes in a WAP page.)

2. By converting the digital map data which is in a binary and proprietary format from HKSAR government to GML format (text format and vendor neutral), different existing and new spatial indexing mechanism can be easily applied. In this thesis, spatial indexing mechanisms such as STR Packed R-tree is implemented with different parameters.

3. Another contribution in this thesis is on spatial network. Although the road center line (RG1000) provided by Lands Department, HKSAR Government is not in graph representation (node and arc), graph representation can be retrieved by converting the road center line (RG1000) in GML format. The storage of road center line (RG1000) is not efficient using the existing spatial indexing mechanism for some network operations, e.g. shortest path queries. A new algorithm called Connected Page Algorithm (CPA) is introduced. Our modification is based on the characteristics of spatial network to improve the retrieval and update of spatial data.

1.5 Outline of the Thesis

The remaining chapters of this thesis are organized in this way. Chapter 2 reviews spatial access methods such as R-Tree, R*-Tree and R+-Tree. The literature on spatial network and nearest neighbor queries would also be investigated.

Chapter 3 provides the details on the data studied in this thesis. Digital Map data from Lands Department, HKSAR Government would be used in this thesis. Brief illustration of XML, GML and XML indexing is also used to explain the process of conversion of data format to GML format. Based on the raw GML data, additional information can be retrieved which is helpful for spatial indexing and graph representation of spatial network.

Chapter 4 describes the implementation of STR Packed R-Tree in GML spatial data. It is well-known that STR Packed R-Tree is the most efficient for storing static general spatial data. However, the experimental result for spatial storage is good for building polygons, but not the road center line, i.e. spatial network data. The reason is that spatial network data is inter-related (the end point of a line segment would also be an end point of another line segment), so the overlapping situation of non-leaf level in STR Packed R-Tree is much worse in spatial network data than other general spatial data.

Based on the result from Chapter 4, Chapter 5 will study Connectivity-Clustered Access Method (CCAM) and a newly proposed algorithm called Connected Page Algorithm (CPA). A simple heuristic which is specifically designed for spatial network to try to improve the query time of Dijkstra's method would also be introduced.

Based on shortest path queries, chapter 6 will extend Connected Page Algorithm to application of nearest neighbor queries.

The final chapter of this thesis concludes with questions we are dealing with and research findings. Our previous work on web-based digital map is discussed and future research directions will be outlined.

Chapter 2

Literature Review and Research Issues

2.1 Introduction

In order to handle spatial data efficiently, as required in computer aided design and geo-data applications, a database system needs an index mechanism that will help it retrieve data items quickly according to their spatial locations.

In this chapter, the literature on index mechanism (or spatial data access method) would be discussed. Our review is intended to provide an overview. Detailed discussion on the literature relevant to our work will be given in subsequent chapters.

There are three fundamental spatial data types, illustrated in Fig 2.1, namely points, lines, and polygons. Points represent an object (e.g. lamp). Line segments represent a road segment, electrical wire, etc. Polygons represent a building or structure.

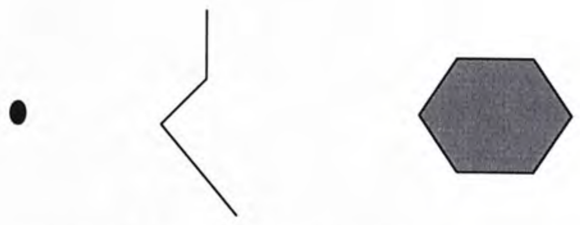


Fig 2.1 The fundamental spatial data types (point, line, polygon).

Spatial data types have complex structures. Consider, for example, the line illustrated in Fig 2.1: such a line is represented by a list of connected points or line segments, but the number of points to represent a spatial object is not fixed. (e.g. The point object in Fig 2.1 can be represented by one point coordinates. The line object in Fig 2.1 can be represented by four point coordinates. The polygon object in Fig 2.1 can be represented by six point coordinates. For some other complex spatial objects, it may consist of hundreds of points.) In other words, its representation in a table (according to the relational model) or a class (according to the object-oriented model) is not straightforward. Spatial operators (e.g., overlap, within) are also more computational expensive than the traditional ones (equality, inequality, string truncation, etc.)

Processing spatial queries, such as checking whether two spatial objects are overlap or not, is very complex. For example, if we want to check whether the following line segments and polygons overlap or not (Fig. 2.2), there will be many tedious calculations for comparison. (e.g. We need to check whether there is any point in the line segment bounded by the polygon or not.)

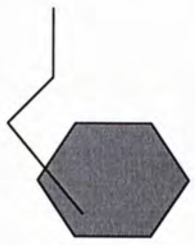


Fig 2.2: Checking if these two objects overlap is a complex task.

A generally accepted solution is the method of minimum bounding rectangle (MBR), which is the smallest rectangle with axis-parallel sides that completely covers the object. According to that, any n-dimensional object is approximated by 2n numeric values, corresponding to the lower-left and upper-right corner (or lower-right and upper-left corner) coordinates. In this thesis we only study 2-dimensional objects. Fig 2.3 is an example of using MBR to check whether two objects overlap.

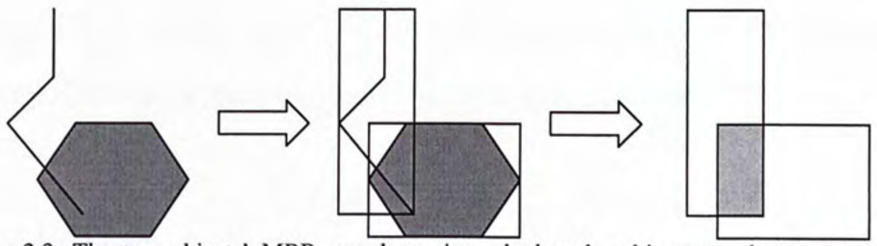


Fig 2.3: The two objects' MBR can determine whether the objects overlap or not can be checked easily.

Some situations make MBR produce a wrong test result. Fig 2.4 illustrates one such example:

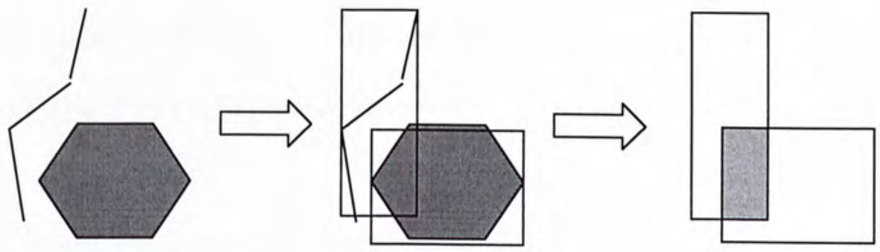


Fig 2.4: False checking by MBR

Although MBR approximations cannot guarantee the relationship of two spatial objects, it can avoid excessive spatial calculations. The literature discusses in this thesis uses MBR as basis to develop spatial access methods.

2.2 Spatial Access Methods

Spatial Access Methods include space driven algorithms and data driven algorithms. In this chapter we review data driven algorithms as the research direction of this thesis concentrates on data driven methods. In appendix we review some space driven algorithms as some applications use space driven algorithms for storage and query of spatial data.

2.2.1 R-Tree

One of the first access methods created to support extended objects is Guttman's R-tree [Guttman (1984)]. The R-tree is a height-balanced tree which is the natural extension of the B-tree for k-dimensions. Objects are represented in the R-tree by their minimum bounding rectangle (MBR). R-tree is characterized by the following properties:

1. Every leaf node contains between *m* and *M* index records, unless it is the root (where $m \leq M / 2$).
2. For each index record (*I*, tuple-identifier) in a leaf node, *I* is the minimum bounding rectangle that spatially contains the k-dimensional data object represented by the indicated tuple.
3. Every nonleaf node has between *m* and *M* children, unless it is a root.
4. For each entry (*I*, child-pointer) in a nonleaf node, *I* is the minimum bounding rectangle that spatially contains the rectangles in the child node.
5. The root node has at least two children, unless it is a leaf.
6. All leaves appear on the same level.

7. All MBRs have sides parallel to the axis of a global coordinate system.

In order to efficiently process spatial queries, one needs specific access methods relying on a data structure called an index.

Each node in the tree corresponds to a disk page. A leaf node consists of a number of entries with format (*I*, tuple-id), where *I* is an MBR, and tuple-id is the unique identifier for the tuple in the database holding the object corresponding to that MBR. *I* is represented as $I = (I_0, ..., I_{k-1})$, where I_i is a closed, bounded interval [a,b] along direction *i*.

Nonleaf nodes are composed of a number of entries of the format (*I*, child-pointer) where *I* is the MBR for all rectangles in the lower node entries pointed to by child-pointer. Each node in the tree can have a maximum of M entries and a minimum *m* (where $m \leq M/2$) entry, unless it is the root. The root node has at least two children, unless it is a leaf.

Point and range queries can be processed in a top-down recursive manner on R-tree. The query point (or region) is tested first against each entry (*I*, child-pointer) in the root. If the query point is inside (or query region overlaps with) *I*, then the search algorithm is applied recursively on entries in the R-tree node pointed to by the child-pointer. This process stops after reaching the leaves of R-tree. The selected entries in leaves are used to retrieve the records.

The PointQuery algorithm (Fig. 2.5) with an R-tree is given as follows.

RT-PointQuery (P:point): set (oid)

```
begin
    result = $\phi$
    // Step 1: Traverse the tree from the root, and compute SL, the
    // set of leaves whose dr contains P
    SL = RtreeTraversal (root, P)
    // Step 2: scan the leaves, and keep the entries that contains P
    for each L in SL do
        // Scan the entries in the leaf L
        for each e in L do
            if (e.mbb contains P) then result += {e.oid}
        end for
    end for
    return result
end
```

Fig. 2.5 RT-PointQuery

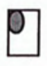
The window query algorithm is a straightforward generalization of the point query in which the "contains P" predicate is replaced by the "intersect W" predicate, where W is the window argument. The larger the window, the larger the number of nodes to be visited.

Search performance depends on two parameters: *coverage* and *overlap*. Coverage of a level of the tree is the total area covered by all MBRs of all nodes at that level. This way, coverage is an indirect measure of dead space area, or empty space covered by the tree. Overlap of a level of the tree is the total area of space covered by more than one rectangle associated with nodes at that level. Overlap may make it necessary to visit more than one node of the tree to find an object. This problem associated with the R-tree means that a worst case performance of search operations cannot be estimated, even if an attempt is made to minimize overlap.

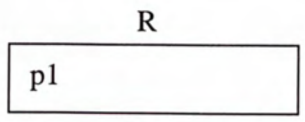
The following example illustrates the indexing of spatial objects.

Example of R-tree with M=4:

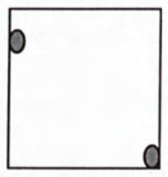
Inserting point 1:



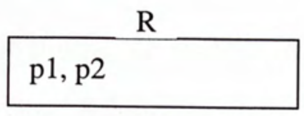
R: p1



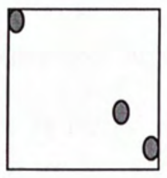
Inserting point 2:



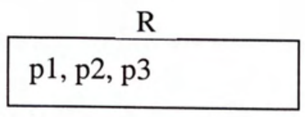
R: p1, p2



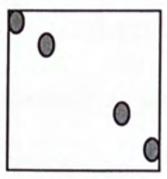
Inserting point 3:



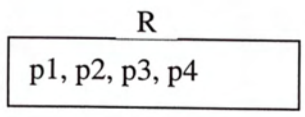
R: p1, p2, p3



Inserting point 4:



R: p1, p2, p3, p4



Inserting point 5:

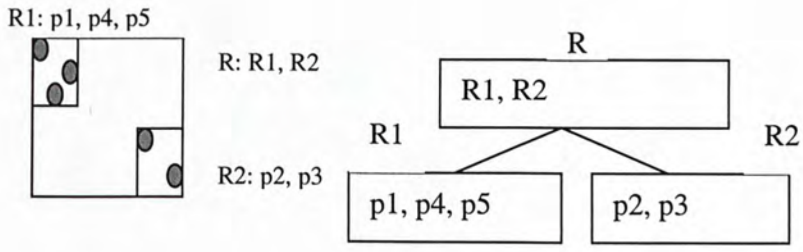


Fig. 2.6: Example of R-tree

2.2.2 R*Tree

R*tree [Beckmann, et al. (1990)] is a variant of R-tree and it supports several improvements to the insertion algorithm. Essentially, these improvements aim at optimizing the following parameters: (1) node overlapping, (2) area covered by a node, and (3) perimeter of a node's directory rectangle.

The R-tree split algorithm first initializes the two groups with the two entries that are as far as possible from each other, then assigns each of the remaining entries to a group. The R*tree approach is different in the sense that it assumes the split to be performed along one axis (say, horizontal), and explores all possible distributions of objects above or below the split line. Fig 2.7 illustrates the difference between R-tree and R*Tree

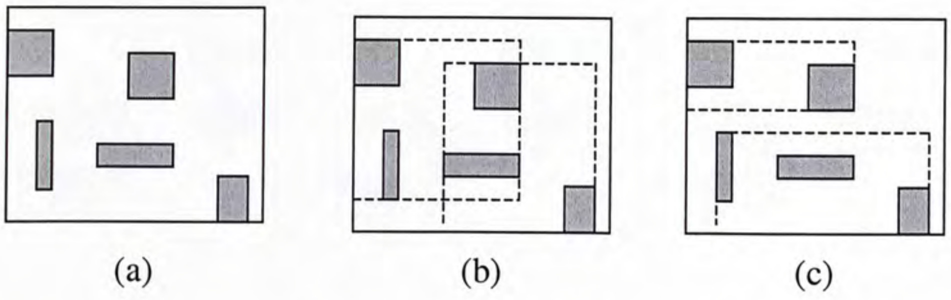


Fig 2.7: Splitting strategies: overflowing node (a), a split of the R-tree (b), and a split of the R*tree (c)

Another improvement of R*tree is the forced reinsertion algorithm. The insertion order can dramatically influence the quality of the R-tree organization. R*tree tries to avoid the degenerated cases by reinserting some entries whenever a node overflows.

Assume (see Figure 2.8) that rectangle 8 is inserted in the tree of (a). Node v overflows, and the R-tree split algorithm will only perform a local reorganization with a rather important node overlapping (b). The R*tree tries to avoid splitting by reinserting the rectangles in node v and finds that removing of rectangle 4 in node v can save a lot of dead space in node v. The reinsertion algorithm proceeds as follows:

- Remove 4 from node v.
- Compute the new bounding box of node v.
- Reinsert 4, starting from the root, and insert in node u.
- Now entry 8 can be inserted in node v, and no split occur.

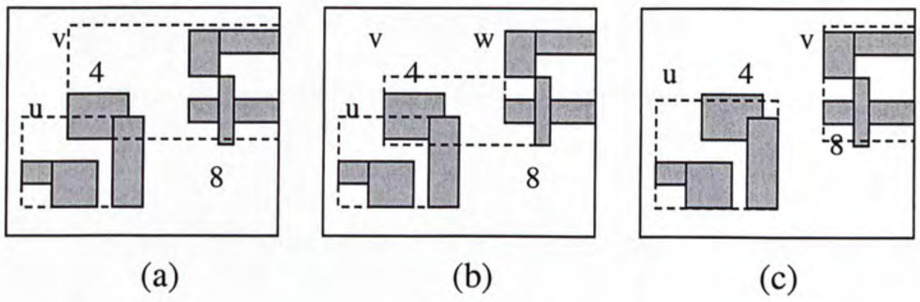


Fig 2.8: The R*tree reinsertion strategy: insertion of 8 (v overflows) (a), a split of the R-tree (b), and R*tree forced reinsertion of 4 (c).

2.2.3 R+ Tree

In R+tree [Sellis, et al. (1987)], the directory rectangles at a given level do not overlap. This has as a consequence that for a point query a single path is followed from the root to a leaf. R+tree is defined as follows:

- The root has at least two entries, except when it is a leaf.
- The directory rectangle (*dr*) of two nodes at the same level cannot overlap. The point query performance benefits from the nonoverlapping of *dr*.
- If node N is not a leaf, its *dr* contains all rectangles in the subtree rooted at N.
- Two *dr* in the same level cannot be overlapped. If a spatial object covers two or more *drs*, this spatial object will be indexed in all *drs* that this spatial object covered.

- A rectangle of the collection to be indexed is assigned to all leaf nodes the *drs* of which it overlaps. A rectangle assigned to a leaf node N is either overlapping N.*dr* or is fully contained in N.*dr*.

Figure 2.9 illustrates an R+tree. Note that objects 8 and 12 referenced twice. Object 8 is overlapping leaves p and r, whereas object 12 is overlapping leaves p and q. Note also that both at the leaf level and at the intermediate level, node *drs* are not overlapping.

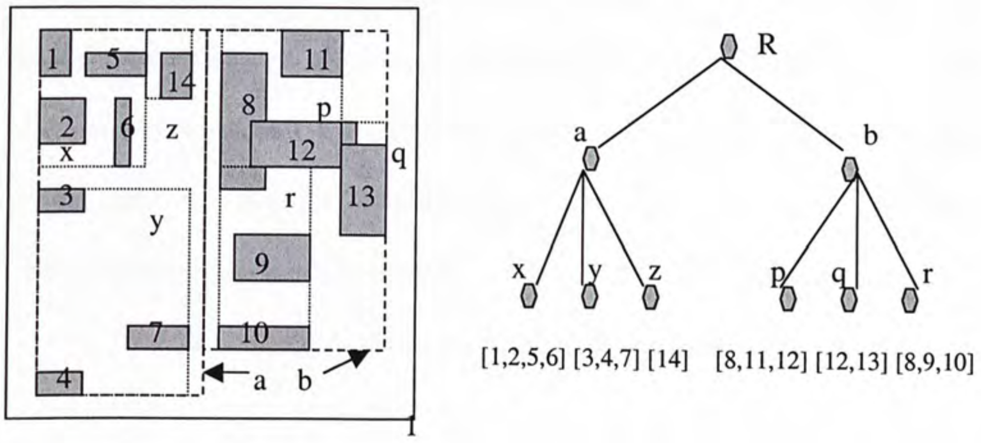


Fig 2.9: The R+tree

2.3 Spatial Network Analysis

A spatial network is one type of spatial data which consists of network relations. Applications of spatial network include transportation networks, drainage systems, electricity systems, etc. In addition to the ordinary insert, update and query operations in spatial data, a spatial network also includes shortest path queries and other network related operations.

[Shashi, et al. (1997)] has evaluated Connectivity-Clustered Access Method for Networks and Networks Computations (CCAM). Shashi defines a spatial network as follows:

"A spatial network is a special kind of graph, with nodes located in a two-dimensional or three-dimensional Euclidean space. Unlike raster and vector area, spatial network data is characterized by rich connectivity. A spatial network G = (N, E) consists of a node set N and an edge set E. Each element u in N is associated with a pair of real numbers (x, y) representing the spatial location of the node in an Euclidean plane. Edge set E is a subset of the cross product N*N. Each element (u, v) in E is an edge that joins node u to node v. There are attributes associated with the nodes and edges."

Details of CCAM are illustrated in Chapter 5. The major research contribution of this thesis is also related to spatial network.

2.4 Nearest Neighbor Queries

Nearest Neighbor Queries is an important application in geographical information systems. For example, finding the nearest gas station, car park, etc. Nearest Neighbor Queries is also essential in delivery planning systems such as finding the nearest depot to deliver the product to particular customer and determine the nearest parking place to deliver the product to the customer.

Below is the nearest neighbor queries by R-tree proposed in Roussopoulos, et al. (1995), Papadopoulos, et al. (1997), Cheung, et. al. (1998), Tao, et al. (2002).

Define (i)Point ***P***: the query point

(ii)Rectangle ***R***: the directory rectangle or minimum bounding rectangle in R-tree

(iii)Vertex *V*: the nearest vertex of ***R*** from ***P***.

(iii)***Min-distance(P, R)*** = 0 if P is inside ***R*** or on boundary of ***R***

Min-distance(P, R) = Euclidean distance between ***P*** and any edge of ***R*** if ***P*** is outside ***R***

(iv)***Min-Max-distance (P, R)*** is the distance of ***P*** from the farthest point on any face of the ***R*** containing vertex ***V***

The search algorithm for nearest neighbor starts with the root node of the R-tree and traverses the tree. For example, a breadth first traversal of the R-tree will visit MBRs of the children of the interior nodes of current node for pruning using the above rules. The remaining children will be expanded in the next iteration. The final iteration will have a set of leaf nodes (database object level) from the MBRs that survive level-wise pruning. The algorithm will need to compute the distance of each leaf from query point P to determine the nearest neighbor.

2.5 Summary

In this chapter we have reviewed some representative literature:

(i) Spatial Access Method – R-Tree, R+Tree, R*Tree

(ii) Spatial Network Analysis

(iii) Nearest Neighbor Queries

In the following chapters, the above algorithms would be applied to develop new algorithms.

Chapter 3

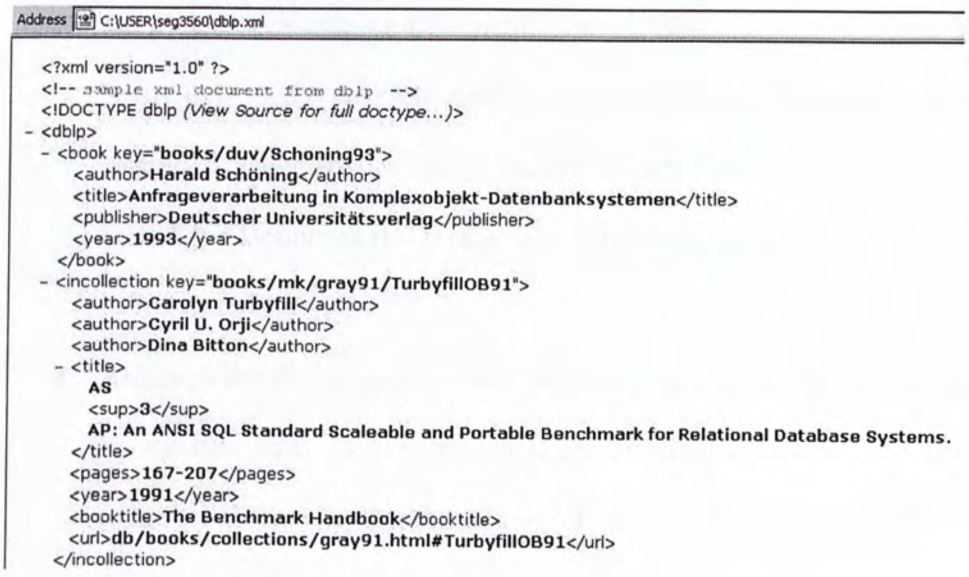
Data preparation

3.1 Introduction – XML, GML, XML Indexing

eXtensible Markup Language (XML), according to www.webopedia.com, is defined as follows: "XML is a specification developed by the World Wide Web Consortium (W3C). XML is a pared-down version of Standard Generalized Markup Language (SGML), designed especially for web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations."

Here we use an example to illustrate one popular usage of XML and illustrate the benefits of using XML:

In http://www.informatik.uni-trier.de/~ley/db/, two files(dblp.xml and dblp.dtd) are available for download among 380,000 articles. Parts of dblp.xml and dblp.org are shown in Fig. 3.1 and Fig. 3.2, respectively:



```
Address C:\USER\seg3560\dblp.xml

  <?xml version="1.0" ?>
  <!-- sample xml document from dblp -->
  <!DOCTYPE dblp (View Source for full doctype...)>
- <dblp>
  - <book key="books/duv/Schoning93">
      <author>Harald Schöning</author>
      <title>Anfrageverarbeitung in Komplexobjekt-Datenbanksystemen</title>
      <publisher>Deutscher Universitätsverlag</publisher>
      <year>1993</year>
    </book>
  - <incollection key="books/mk/gray91/TurbyfillOB91">
      <author>Carolyn Turbyfill</author>
      <author>Cyril U. Orji</author>
      <author>Dina Bitton</author>
    - <title>
        AS
        <sup>3</sup>
        AP: An ANSI SQL Standard Scaleable and Portable Benchmark for Relational Database Systems.
      </title>
      <pages>167-207</pages>
      <year>1991</year>
      <booktitle>The Benchmark Handbook</booktitle>
      <url>db/books/collections/gray91.html#TurbyfillOB91</url>
    </incollection>
```

Fig 3.1 Part of dblp.xml

```
<!ELEMENT dblp (article|inproceedings|proceedings|book|incollection|
              phdthesis|mastersthesis|www) *>
<!ENTITY % field
"author|editor|title|booktitle|pages|year|address|journal|volume|number|month|url|ee|cdr

<!ELEMENT article        (%field;)*>
<!ATTLIST article
                         key CDATA #REQUIRED
                         reviewid CDATA #IMPLIED
                         rating CDATA #IMPLIED
>

<!ELEMENT inproceedings (%field;)*>
<!ATTLIST inproceedings key CDATA #REQUIRED>

<!ELEMENT proceedings    (%field;)*>
<!ATTLIST proceedings    key CDATA #REQUIRED>

<!ELEMENT book           (%field;)*>
<!ATTLIST book           key CDATA #REQUIRED>
```

Fig 3.2 Part of dblp.dtd

Even a person has little or even no knowledge of XML, he or she can get the details of particular articles by browsing the dblp.xml files. For example, according to Fig 3.2, it describes what type of materials is saved in dblp (articles, inproceedings, proceedings, book, etc.). For each article, it must contain the key and other fields, such as author, editor, title, etc.

Here we can summarize the benefits of using XML:

- XML is indeed a text file with a structured format possessing self-described characteristics. The format is described in the corresponding Data Type Definition (DTD) file. The data stored in a DTD file is also called metadata.
- Although the storage space would be larger for a XML file than other binary file, XML is vendor-neutral and XML file can be read and modified through any text editor. This turns out to be effective in dealing with the compatibility problem. The storage space problem of a XML file can be alleviated by using some compression mechanism.

Geography Markup Language is "an XML encoding for the transport and storage of geographical information, including both spatial and nonspatial properties of geographic features" [GML]. It is a recommended standard to encode or mark up spatial and nonspatial information in XML format by OGC. GML provides a standard way to encode spatial features, feature properties, feature geometries, and the location of the feature geometries based on a standard spatial data model.

Fig. 3.3 is a sample GML file that is extracted from Lands Department, HKSAR Government:

```
<?xml version='1.0' encoding='UTF-8'?>
<dataset xmlns="http://www.safe.com/xml/namespaces/fmegml2"
xmlns:fme="http://www.safe.com/xml/namespaces/fmegml2"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.safe.com/xml/schemas/fmegml2.xsd">
<schemaFeatures>
<gml:featureMember>
<Feature>
<featureType>30</featureType>
<property fme:name="igds_class">fme_decimal(5,0)</property>
<property fme:name="igds_color">fme_decimal(5,0)</property>
<property fme:name="igds_graphic_group">fme_decimal(5,0)</property>
<property fme:name="igds_style">fme_decimal(5,0)</property>
<property fme:name="igds_weight">fme_decimal(5,0)</property>
</Feature>
</gml:featureMember>
</schemaFeatures>
<dataFeatures>

<gml:featureMember>
<Feature>
<featureType>30</featureType>
<property fme:name="igds_class">0</property>
<property fme:name="igds_color">1</property>
<property fme:name="igds_graphic_group">0</property>
<property fme:name="igds_style">0</property>
<property fme:name="igds_weight">0</property>
<property fme:name="gml2_coordsys"></property>
<gml:lineStringProperty>
<gml:LineString gml:srsName=""><gml:coordinates>838920.253,827495.65,0
838943.247,827483.115,0 838954.949,827461.57,0</gml:coordinates></gml:LineString>
</gml:lineStringProperty>
</Feature>
</gml:featureMember>
</dataFeatures>
</dataset>
```

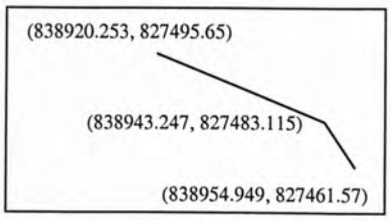


Fig 3.3a: The line segment represented by the GML file.

Fig 3.3: A sample GML file

In Fig 3.3, the GML file contains only one linestring (i.e. line) that is composed of three points (i.e. (838920.253,827495.65), (838943.247,827483.115), (838954.949,827461.57)). The graphical representation of this line string is in Fig 3.3a.

In search, query, and update of a GML file, the searching time will be quite large (O(n)) if no indexing is used. (This is because we need to compare one by one of each spatial object stored for query or update. Hence, we need an index scheme in the GML file to facilitate the spatial index method as mentioned in Chapter 2.

By finding the minimum bounding rectangle (MBR) of each spatial object, we can make index for each spatial object, for example, the index of the object represented in Fig 3.3 is as follows (Fig. 3.4):

```
<featureMember>
  <linestring_key>lk14806</linestring_pk>
  <Box>
    <min_x_coord>838920.253</min_x_coord>
    <max_x_coord>838954.949</max_x_coord>
    <min_y_coord>827461.57</min_y_coord>
    <max_y_coord>827495.65</max_y_coord>
    <centre_x>838937.601</centre_x>
    <centre_y>827478.610</centre_y>
  </Box>
  <start_x>838920.253</start_x>
  <start_y>827495.65</start_y>
  <end_x>838954.949</end_x>
  <end_y>827461.57</end_y>
  <absolute_distance>48.634</absolute_distance>
  <relative_distance>48.634</relative_distance>
</featureMember>
```

Fig 3.4: index of the spatial object (line segment) shown in Fig 3.3

In the index, it contains the following items:

(i) linestring_key: a unique spatial object key, e.g. lk14806

(ii) Box: minimum bounding rectangle (MBR) with lower-left, upper-right and center coordinates

(iii) Coordinates of two end-points

(iv) Distance of the line string and distance of the two end points

The above index can be used to search (i) a unique key and (ii) by coordinates key. (e.g. by any data-driven algorithm such as R-Tree)

3.2 Spatial data from Lands Department

This thesis uses the real spatial data from Lands Department, HKSAR Government. Table 3.1 is the table showing the geographical data from Lands Department.

Code	Name	Scale	Area purchased
B20000	Digital Topographic Map Database	1:20000	Whole Hong Kong area (16 sheets)
BG1000	Geo-Reference Database – Building Name and Address	1:10000	7SE, 7SW, 7NE, 7NW, 11SE, 11SW, 11NE, 11NW
SG1000	Geo-Reference Database – Site Polygon	1:10000	7SE, 7SW, 7NE, 7NW, 11SE, 11SW, 11NE, 11NW
RG1000	Geo-Reference Database – Road Centre Line	1:10000	7SE, 7SW, 7NE, 7NW, 11SE, 11SW, 11NE, 11NW

Table 3.1: Digital Map data used in this research
(Ref: http://www.info.gov.hk/landsd/mapping/web/page/d_m_prod_new.htm)

The data format provided by Lands Department are all in proprietary format.

(http://www.info.gov.hk/landsd/mapping/web/page/d_format_new.htm)

We use Feature Manipulation Engine (FME) (http://www.safe.com/products/fme/index.htm) to convert the data from ArcInfo Ungenerated format(.e00) to Geography Markup Language (.xml).

3.3 Graph representation for Road Network data

One type of data available in Lands Department is road centre line (RG1000). Fig. 3.5 is an example.

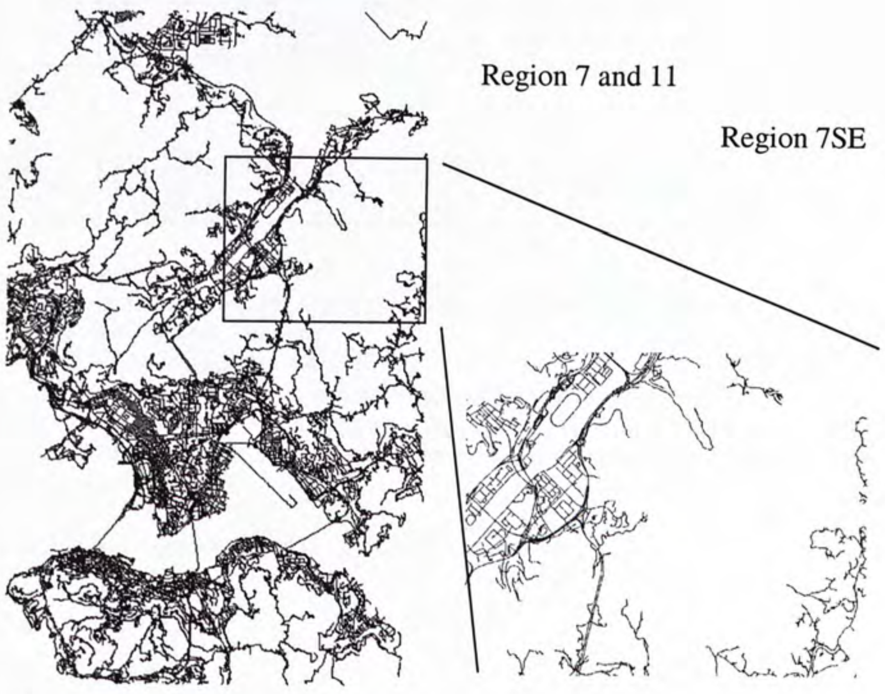


Fig 3.5: RG1000 Spatial data from HKSAR Government

The data format of RG1000 is in linestring (line) format. It stores all the line data like the format shown in Fig. 3.4. Indeed RG1000 contains the road network data and we can make network operations on it. But it do not contain the necessary information for network operations such as nodes and length of linestring. We cannot effect network operations on it directly.

However, we can retrieve the network representation of RG1000 spatial data as following node and arc table (Table 3.2 and Table 3.3):

Primary Key of Node	X_Coord	Y_Coord	Foreign Key for Road	No. of Node Connected	Connected or not
0	830000.000	823316.956	0	1	1
1	830000.000	823331.336	1	1	1
2	830000.000	823351.209	2	1	1
3	830000.000	823483.587	3	1	1
4	830000.000	823513.003	4	1	1
			⋮		
13792	845000.000	832203.152	34168	1	0
13793	845000.000	832221.393	34169	1	0

(total 13794 records in node table)
Primary Key of Node: Unique key of each node (end point of line segment)
X_Coord: X-Coordinates of node
Y_Coord: Y-Coordinates of node
Foreign Key for Road: Set Relations to the primary key in the road table
No. of Node Connected: This column store the no. of arc connected with this node
Connected or not: This column stores whether this node is connected with the major road network or not.

Table 3.2: Node table prepared from BG1000 spatial data

Primary Key of Start Node	Primary Key of End Node	X_Coord of Start Node	Y_Coord of Start Node	X_Coord of End Node	Y_Coord of End Node	Absolute distance	path length
0	103	830000.000	823316.956	830104.688	823349.207	109.543	109.643
1	141	830000.000	823331.336	830187.380	823424.393	209.215	213.177
2	98	830000.000	823351.209	830096.751	823378.116	100.423	100.423
3	362	830000.000	823483.587	830699.882	823634.481	715.964	772.779
4	75	830000.000	823513.003	830054.567	823537.665	59.881	60.200
				⋮			
13792	13734	845000.000	832203.152	844968.833	832160.401	52.906	55.512
13793	13653	845000.000	832221.393	844859.687	832221.758	140.313	198.639

(total 34170 records in arc table)
Primary Key of Start Node: Unique key of start node (end point of line segment)
Primary Key of End Node: Unique key of end node (end point of line segment)
X_Coord: X-Coordinates of start node
Y_Coord: Y-Coordinates of start node
X_Coord: X-Coordinates of end node
Y_Coord: Y-Coordinates of end node
Absolute distance: the distance between two end points in metres
Path length: the length of the road

Table 3.3: Arc table prepared from BG1000 spatial data

Since RG1000 data do not contain the directed information (one-way or two-way road), we assume all the arcs are undirected (two-way road) and we formulate the directed network by duplicating an arc with swapping of start node and end node. Therefore, we have 17085 (no. of records of RG1000 in our studied area) x 2 = 34170 arcs.

We would use the above real spatial data for trying different spatial data access algorithms in the following chapters. The major different of this proposed indexing mechanism is that it can be use for spatial query as it has the Minimum Bounding Rectangle (MBR) in each index page and also the primary key of each spatial object which can be used for mapping of non-spatial data such as building name and address.

3.4 Summary

In this chapter, we discussed how to prepare data in a format suitable for our research. Spatial data will be represented in GML, that is a variant of XML. An index scheme is also defined for quick accesses.

Chapter 4

XML Indexing for Spatial Data

4.1 Introduction

In the previous chapters we have described the existing vendor-specific spatial data formats. They are not suitable for the study of our spatial access methods. By converting those vendor-specific spatial data formats to vendor-neutral Geography Markup Language [*GML*], we would be able to develop and study spatial access methods in this chapter.

The spatial access method algorithms presented in chapter 2 for inserting entries in an R-tree, R*-tree, R+-tree are dynamic in the sense that they enable concurrent insertions and deletions in an already existing R-tree. However, the evolution of the structure over time does not allow one to optimize the space utilization and thus might lead to a degradation in performance as a number of insertions and deletions has been performed.

In the static case, when the collection of rectangles is stable over time, one can pre-process the data before creating an associated R-tree. Several algorithms, called packing algorithms, have been proposed for a R-tree. The packed R-tree is described below [Leutenegger, et al. (1996)]:

1. Preprocess the data file so that the r rectangles are ordered in $\lceil r/b \rceil$ consecutive groups of b rectangles, where each group of b is intended to be placed in the same leaf level node. Note that the last group may contain fewer than b rectangles.
2. Load the $\lceil r/b \rceil$ groups of rectangles into pages and output the (MBR, page-number) for each leaf level page into a temporary file. The page numbers are used as the child pointers in the nodes of the next higher level.
3. Recursively pack these MBRs into nodes at the next level, proceeding upwards, until the root node is created.

4.2 STR Packed R-Tree

STR Packed R-Tree is a Packed R-Tree with a specific sorting order. The following is the description of STR Packed R-tree [Leutenegger, et al. 1996]:

"Consider a k-dimensional data set of r hyper-rectangles. A hyper-rectangle is defined by k intervals of the form $[A_i, B_i]$ and is the locus of points whose i-th coordinate falls inside the i-th interval, for all $1 \le i \le k$.

STR is best described recursively with $k = 2$ providing the base case. (The case $k = 1$ is already handled well by a regular B-tree.) Accordingly, we first consider a set of rectangles in the plane. The basic idea is to "tile" the data space using $\sqrt{r/n}$ vertical slices so that each slice contains enough rectangles to pack roughly $\sqrt{r/b}$ nodes. Once again we assume coordinates are for the center points of the rectangles. Determine the number of leaf level pages $P = \lceil r/b \rceil$ and let $S = \left\lceil \sqrt{P} \right\rceil$. Sort the rectangles by x-coordinates and partition them into S vertical slices. A slice consists of a run of $S \cdot b$ consecutive rectangles from the sorted list. Note that the last slice may contain fewer than $S \cdot b$ rectangles. Now sort the rectangles of each slice by y-coordinate and pack them into nodes by grouping them into runs of length b (the first b rectangles into the first node, the next b into the second node, and so on).

The case $k > 2$ is a simple generalization of the approach described above. First, sort the hyper-rectangles according to the first coordinates of their center. Then divide the input set into $S = \left\lceil P^{\frac{k-1}{k}} \right\rceil$ slabs, where a slab consists of a run of $b \cdot \left\lceil P^{\frac{k-1}{k}} \right\rceil$ consecutive hyper-rectangles from the sorted list. Each slab is now processed recursively using the remaining $k - 1$ coordinates (i.e., treated as a k – 1-dimensional data set)."

The following section illustrates the implementation of STR Packed R-tree for spatial data.

4.2.1 Implementation

We use the data shown in Fig 3.5 (Road Centre Line, RG1000 of Region 7 and 11. The data cover the whole Kowloon Peninsula and more than half of Hong Kong Island). We prepare the index of the data as the structure shown in Fig 3.4.

The parameters which are needed for applying STR Packed R-tree include:

1. The total number of line string in RG1000 of Region 7 and 11 are 17085, i.e. r = 17085;
2. It is a 2-dimensional data, i.e. k = 2;

3. Assume that 3 objects are saved in a directory rectangle, b = 3;

4. Number of leaf level pages, $P = \lceil r/b \rceil = \lceil 17085/3 \rceil = 5695$;

5. Number of vertical slices, $S = \left\lceil \sqrt{P} \right\rceil = \left\lceil \sqrt{5695} \right\rceil = 76$.

To recursively pack the MBRs to next level, we have the following leaf level (Table 4.1):

r = 17085, k = 2, b = 3

Non-leaf Level	1	2	3	4	5	6	7	8
P	5695	1899	633	211	71	24	8	3
S	76	44	26	15	9	5	3	1

Table 4.1: Leaf level pages, P and vertical slices, S in different non-leaf level

Tree representation of our experiment is shown in Figure 4.1 (for b = 3):

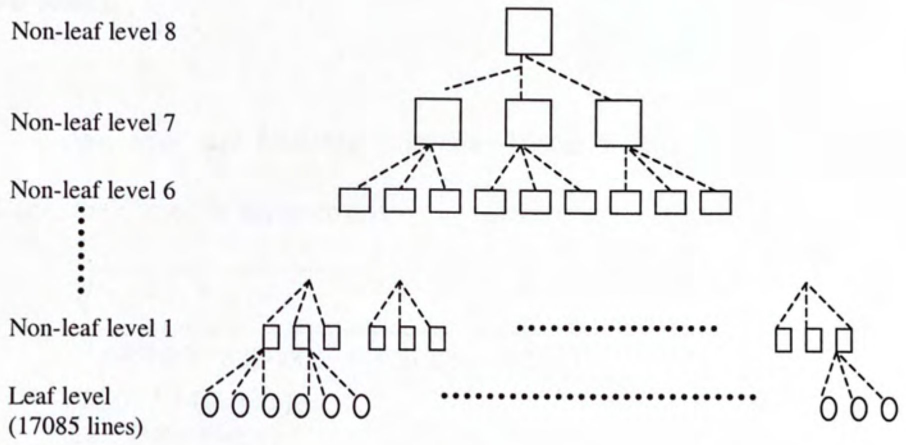


Fig 4.1: Graphical representation of STR Packed R-tree

We implement the STR Packed R-tree in Java 1.3.1 with operating system Redhat 7.2 running on a desktop computer. The configuration of the computer is AMD Duron 1.1G CPU, 256MB SD-RAM with an IBM 9.1G SCSI Harddisk.

4.2.2 Experimental Result

As it is mentioned earlier, the data we study cover the whole Kowloon Peninsula and more than half of Hong Kong Island. This area consists of the most urban area of Hong Kong although it only covers about one-eighth of the whole territory. The region ranges from x-coordinates with (830000 - 845000) and y-coordinates (812000 - 836000). The coordinates system using is Hong Kong 1980 Grid System and the distance can be easily calculated by this coordinates system. For example, suppose we have two points (830000, 812000) and (845000, 836000) and we want to find the distance between the two points.

We may use Distance Formula (based on Pythagorean Theorem). Hence, the Cartesian distance d is

$$\begin{aligned} &= \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \\ &= \sqrt{(845000 - 830000)^2 + (836000 - 812000)^2} \\ &= 28301.94 \text{ (meters)} \\ &= 28.3 \text{ (kilometers)} \end{aligned}$$

The area we study is

$$
\begin{aligned}
&= |(x_a - x_b) * (y_a - y_b)| \\
&= |(845000 - 830000) * (836000 - 812000)| \\
&= 3.6 \times 10^8 \text{ (sq. meters)} \\
&= 360 \text{ (sq. kilometers)}
\end{aligned}
$$

Fig. 4.2 is the sum of the area of the minimum bounding rectangles (MBR) in non-leaf level for STR Packed R-Tree for *b* = 3: (The blue or upper line is the sum of area of MBR. The pink or lower line is the whole area (i.e. 360 sq. km) for the sample data. (i.e. The total sum of MBR in the non-leaf level with more than 360 sq. km means the overlapping of MBR).

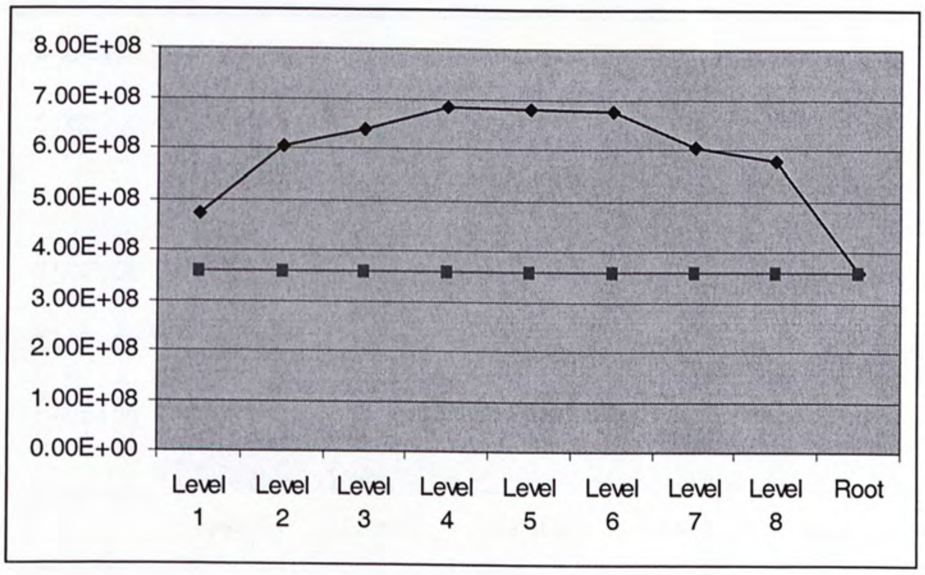


Fig 4.2: Sum of area covered for MBR in different non-leaf level for b = 3.

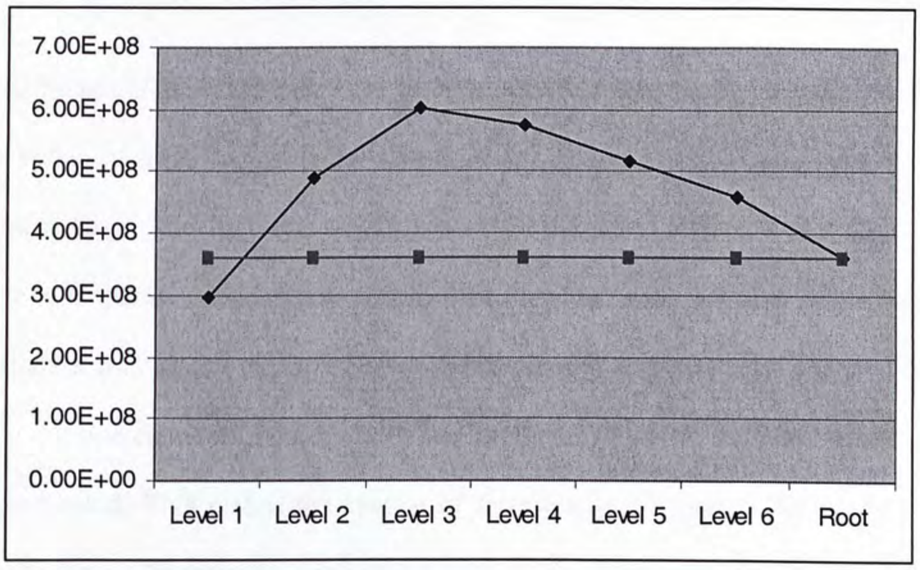


Fig 4.3: Sum of area covered for MBR in different non-leaf level for b = 5.

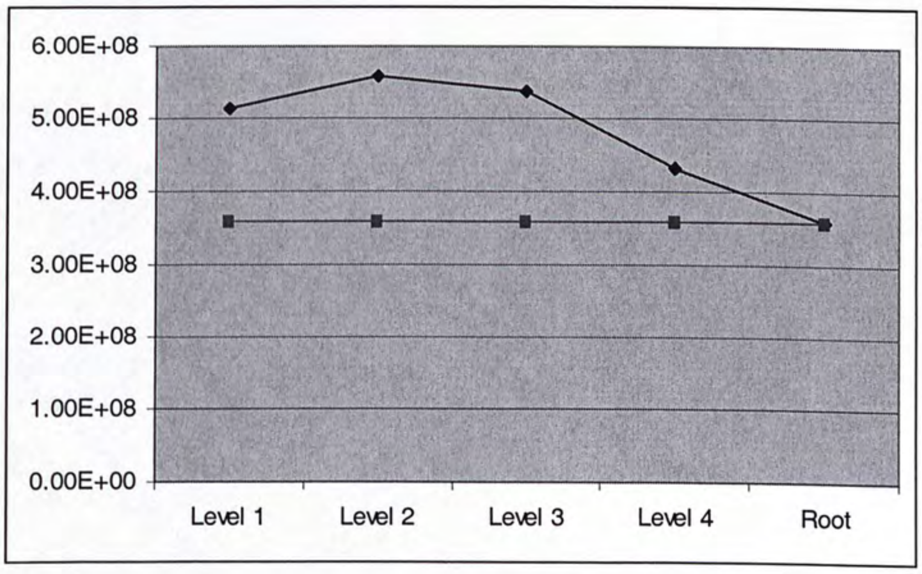


Fig 4.4: Sum of area covered for MBR in different non-leaf level for b = 10

Fig. 4.2 – 4.4 show the implementation of STR Packed R-Tree with different b, i.e. different number of spatial objects saved in each non-leaf page. By increasing b the overlapping area in each non-leaf level would be smaller.

Although STR Packed R-Tree is optimized for storage of spatial data, the overlap of STR Packed R-Tree for RG1000 data of Lands Department is still quite large. This indicates that in general STR Packed R-Tree is optimized for general points and polygon spatial objects which have no inter-relationship. But for line spatial object which is the element of a spatial network, the MBR of the line elements overlaps with one another, as most of the line element are connected. This makes the chance of false query (for details please refer to Fig. 2.5 in Chapter 2) much larger than other point or region MBR. Let us illustrate the difference for storage of line spatial objects and other spatial objects by the following example from our real data (Fig 4.5):

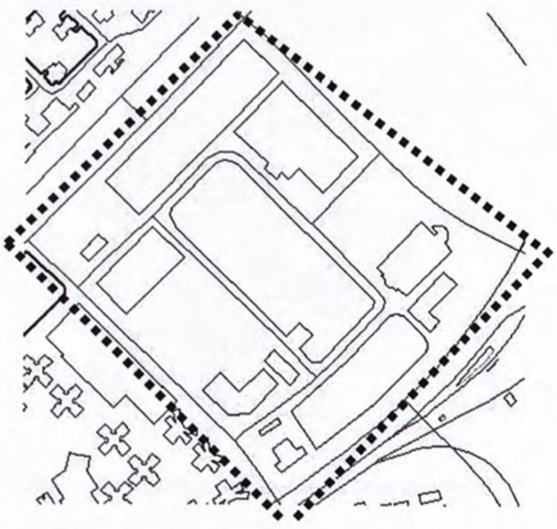


Fig 4.5: Map data from Siu Lek Yuk, Shatin

Above region (Fig 4.5) consists of two layers: Road Center Line (RG1000) (Fig. 4.6a) and Building layer (one layer in B20000) (Fig. 4.6b). And we illustrate the example by only the dotted region:

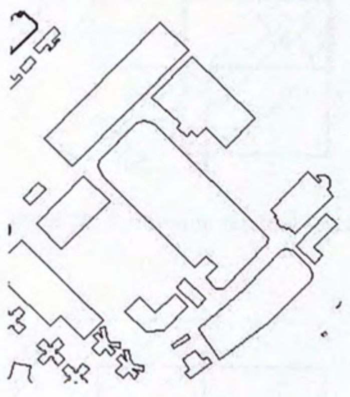


Fig. 4.6a: Building layer of B20000

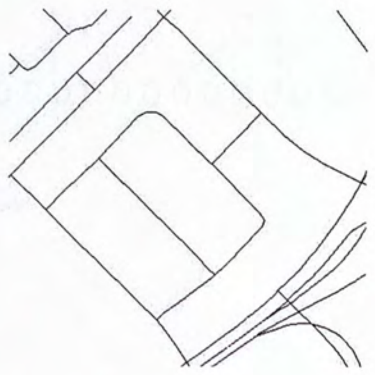


Fig. 4.6b: Road Centre Line

Fig. 4.7a-d illustrates for building up the STR Packed R-Tree for Fig 4.6a (assume b = 3):

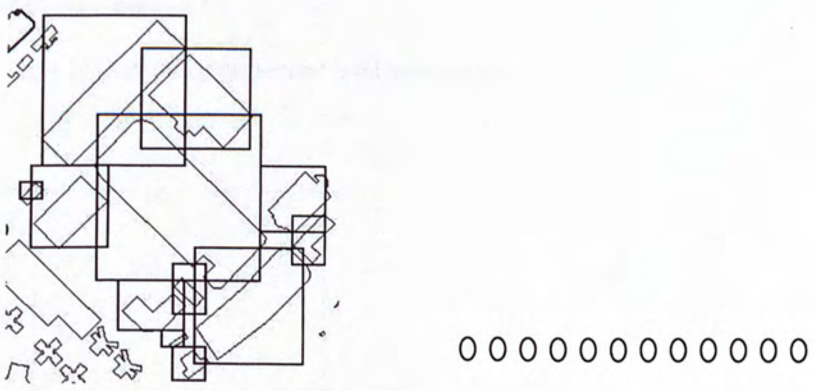


Fig. 4.7a: Finding MBRs for 12 spatial objects

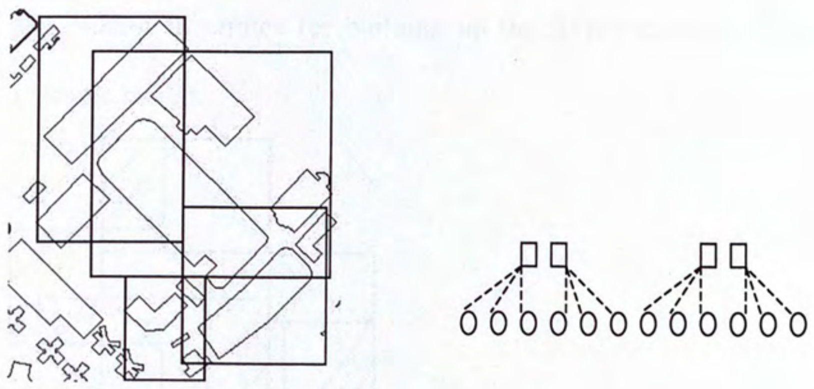


Fig 4.7b: Retrieving the first level non-leaf node

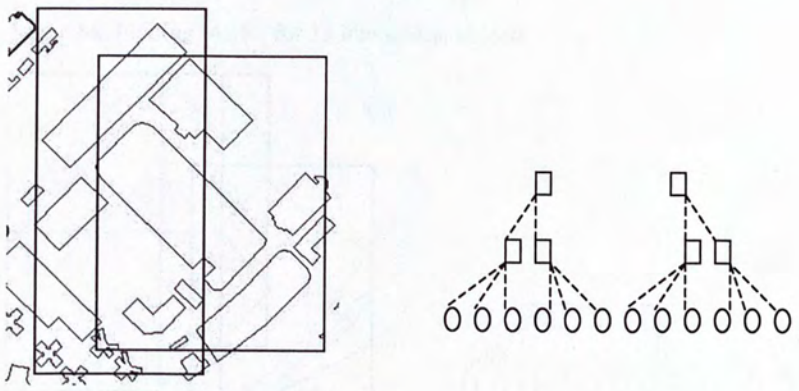


Fig 4.7c: Retrieving the second level non-leaf node

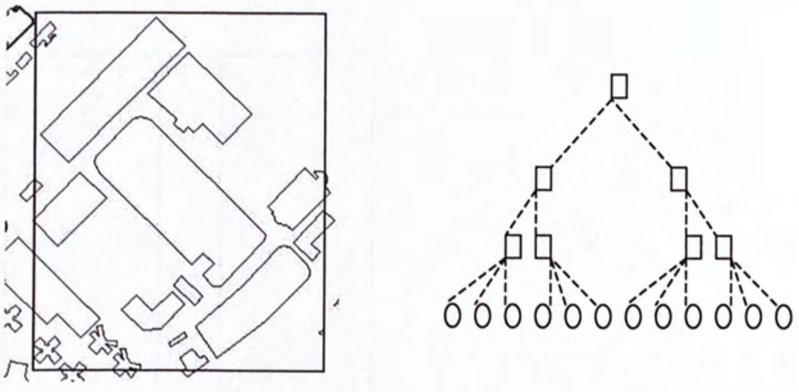


Fig 4.7d: Retrieving the third level non-leaf node (root)

Fig. 4.8a-d illustrates for building up the STR Packed R-Tree for Fig 4.6b (assume b = 3):

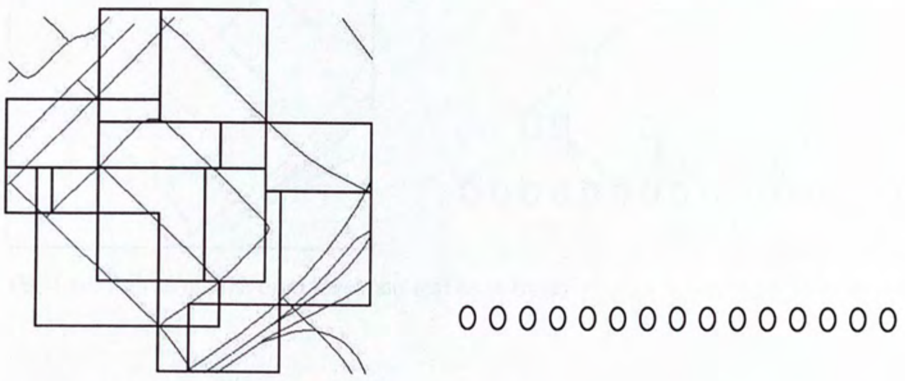


Fig. 4.8a: Finding MBRs for 15 line spatial objects

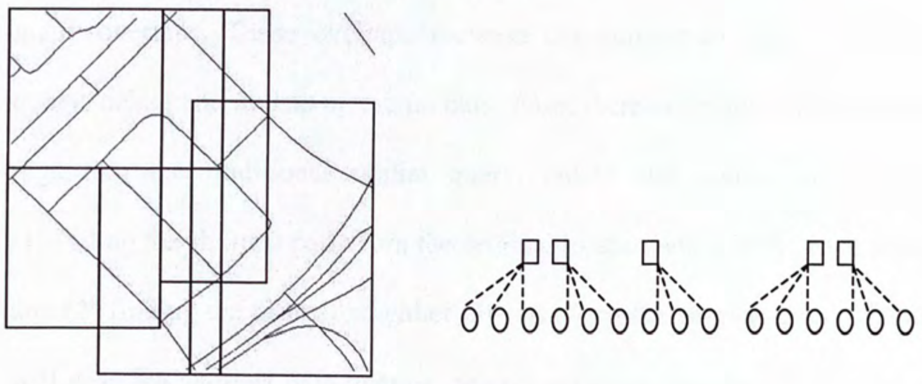


Fig 4.8b: Retrieving the first level non-leaf node

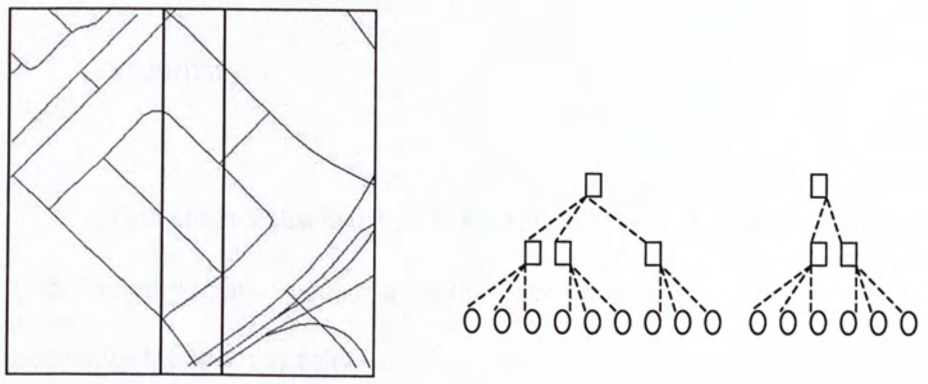


Fig 4.8c: Retrieving the second level non-leaf node

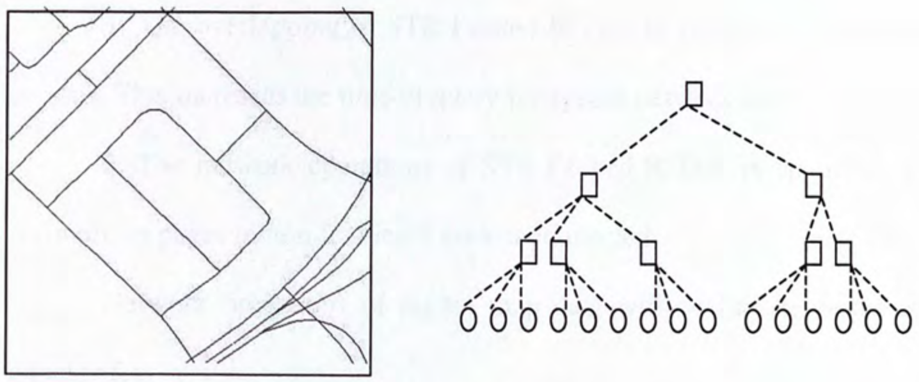


Fig 4.8d: Retrieving the third level non-leaf node (root)

Fig 4.8a-d show that building up STR Packed R-Tree for line string causes many overlaps. These overlaps increase the number of false retrieval for query, delete and update of spatial data. Also, there are many other operations regarding the traditional spatial query, delete and update, for example, (1)finding the shortest path from the original to destination in the road network and (2) finding the nearest neighbor objects along the shortest path. Chapter 5 will describe shortest path queries, nearest neighbor queries and also the new algorithms for road network spatial data.

4.3 Summary

This chapter discusses STR Packed R-Tree and its implementation in both ordinary spatial data and spatial network data (i.e. road network). We concludes the result as follows:

1. The overlapping of STR Packed R-Tree in spatial network is quite serious. This increases the time of query for spatial network data

2. The network operations of STR Packed R-Tree in spatial data are difficult, as pages in non-leaf level are not connected.

Network operations of digital map data will be further discussed in chapter 5.

Chapter 5

Spatial Network

5.1 Introduction

Spatial Data Accesses such as insert, update and delete are the most primitive operations in many geographical applications. With the advance of spatial research, the spatial technology has extended to other areas of practical applications including Intelligent Transportation System (ITS), Transport Information System (TIS), Scheduling and Distribution for Vehicle Routing System, electricity, water and drainage system, etc.

Besides simple spatial queries such as finding the location of a building, many applications also make use of network operations on spatial data, for example, finding the shortest path in a road network, minimizing the cost (e.g., the distance traveled) for delivery by vehicles in a road network, etc. These types of applications are referred to as spatial network applications.

A spatial network refers to a network containing spatial data. The main difference between a spatial network from other spatial data is that line segments are usually inter-related. For instance, the end point of one line segment is also the end point of another line segment. This characteristic is different from other ordinary spatial data. For instance, building polygon that is not inter-related. In Chapter 4, we find that the storage of spatial elements of a spatial network is not as efficient as other ordinary spatial data. Also some modifications in the access methodology can be made to help improve the efficiency of spatial accesses needed in accessing a spatial network.

A spatial network is also a network, i.e. it has all the network characteristics. Algorithms solving shortest path problem, maximum flow problem, and minimum cost flow problem can be applied to a spatial network to model the real world problems. One such real world problem may involve solving traffic congestion. All these applications are developed using spatial data and help us in route planning, and electricity and drainage systems planning, etc.

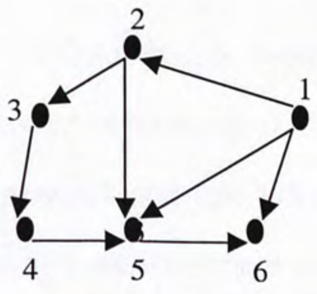
On the other hand, applications of a spatial network usually require more information than that provided by these theoretical algorithms. For example, the optimal path found by the shortest path algorithm such as Dijkstra's algorithm can only get the shortest distance and also the index keys of line segments of the optimal path. However, the user of route planning would like to have the graphical representation of the optimal path and probably the nearby buildings to identify the exact locations he/she wants. All these applications require spatial accesses and nearest neighbor queries (that will be discussed in Chapter 6).

Indeed for spatial network operations such as finding the optimal path in a digital map, most existing digital map applications would only retrieve the corresponding layer and programming developments are needed in application servers to find the optimal path as well as the nearby spatial objects (refer to Fig. 1 in Chapter 1 for details). Although it is a generic architecture (i.e., the spatial database only stores three types of spatial objects: point, line and polygon), it is not very efficient for complex spatial query operations such as shortest path queries and nearest neighbor queries.

In the next section of this chapter, we review a connectivity-clustered access method for networks and network computations [Shekhar, et al. (1997)]. In Section 3, we provide the definition of Shortest Path and its implementation in spatial networks based on the existing spatial access methods (e.g. STR Packed R-Tree). These techniques are useful in understanding our new algorithm. In Section 4, we propose a new algorithm called Connected Page Algorithm that is specially for a spatial access method in a spatial network Section 5 proposes a modified shortest path algorithm that can be implemented for spatial network.

5.2 Connectivity-Clustered Access Method (CCAM)

Since a line segment in a road network consists of two end points (i.e., nodes), CCAM [Shekhar, et al. (1997)] firstly stores the node data, coordinates, successor list, and predecessor list. A successor list (predecessor list) contains a set of outgoing (incoming) edges, each of which is represented by the node-id of its end (start) node and the associated edge cost. The successor list is also called the adjacency list, and is used in network computations. The predecessor list is used in updating the successor list in Insert() and Delete() operations. CCAM refers to the neighbor list of a node x as the set of nodes whose node-id appears in the successor list or predecessor list of x. CCAM notes that records do not have fixed formats because the size of the successor list and predecessor list varies across nodes.



Node

Nid	x	y	Successors	Predessors
1	---	---	(2, 5, 6)	()
1	---	---	(3, 5)	(1)
1	---	---	(4)	(3)
1	---	---	(5)	(3)
1	---	---	(6)	(2, 1)
1	---	---	()	(1, 5)

Fig. 5.1: Node table for CCAM

Assume that the node relation is physically clustered as shown in Fig. 5.1 according to the value of the node-id and the disk page size is two tuples for the node relation. In other words, node 1 and 2 share a page. There are two other pages, one with nodes (3,4) and other with nodes (5,6). This makes one unsplit edge (1,2) and two split edge (1,5) and (1,6). If the nodes can be clustered to minimize the number of unsplit edges, then it will reduce the I/O cost of network operations.

To check the number of split and unsplit edges, a term, called Connectivity Residue Ratio (CRR) can be introduced:

CRR = Total number of unsplit edges / Total number of edges.

It can be shown that maximizing the CRR will minimize the average I/O cost of network operations [Shekhar, et al. (1997)]. As an example, consider the CRR for the example in Fig. 5.1. For pages ((1,2), (3,4), (5,6)), the CRR is 3/8 = 0.375. If each page accommodates three nodes, then higher CRR can be achieved. For example, CRR ((1,2,3), (4,5,6)) is 4/8 = 0.5 and CRR ((1,5,6), (2,3,4)) is 5/8 = 0.625.

CCAM assigns nodes to the data page by a graph partitioning approach, which tries to maximize the connectivity residue ratio (CRR). Each data page is kept at least half full whenever possible. Records of the data file are not physically ordered by node-id values. A primary index cannot be created without renaming the nodes to encode disk-page information in the node-id, and it requires additional overhead during update operations. Therefore, a secondary index can be created on top of the data file with an index entry for each record in the data file.

If networks are embedded in geographic space, (x, y) coordinates for each node are also stored in the record. Then a spatial indexing scheme on the (x, y) coordinates can be used as the secondary index. This secondary index can support point and range queries on spatial databases.

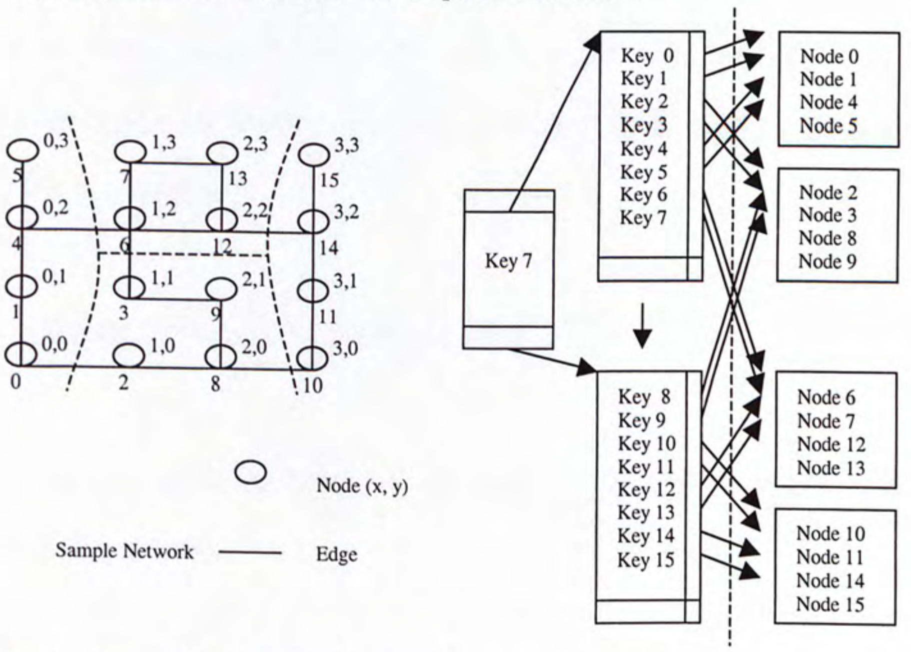


Fig. 5.2: Clustering and storing a sample network (key represents spatial order).

In Fig. 5.2, a sample network and its CCAM are shown. The left half of Fig. 5.2 shows a spatial network. Nodes are annotated with the node-id (an integer) and geographical coordinates (a pair of integers). To simplify the example, the node-id is an integer representing the Z-order of the (x, y) coordinates. For example, the node with the coordinates (1, 1) gets a node-id of 3. The solid lines that connect the nodes represent edges. The dashed lines show the cuts and partitioning of the spatial network into data pages. There exists a cut on edge e(u, v) if node u and node v fall into different partitions. The partitions are (0, 1, 4, 5), (2, 3, 8, 9), (6, 7, 12, 13), and (10, 11, 14, 15). The right half of Fig. 5.2 shows the data pages and the secondary index. We note that the nodes are clustered into data pages by CCAM, using a graph partitioning approach. Nodes in the same partition set are stored on the same data page. They are not physically ordered by their node-id values. A secondary index ordered by node-id is used to facilitate the Find() operation. The secondary index in this example is a B+ tree on the Z-order of (x, y) coordinates of each node.

5.3 Shortest Path in Spatial Network

To solve the shortest path problem in a spatial network, firstly we need to define the problem.

In shortest path problem, we consider a network $G=(N,A)$ with an *arc length* (or *arc cost*) c_{ij} associated with each arc $(i,j)\in A$. The network has a distinguished node s, called the source. Let $A(i)$ represent the arc adjacency list of node i and let $C=\max\{c_{ij}:(i,j)\in A\}$. We define the length of a directed path as the sum of the lengths of arcs in the path. The shortest path problem is to determine for every non-source node $i\in N$ a shortest length directed path from node s to node i. Alternatively, we might view the problem as sending 1 unit of flow as cheaply as possible (with arc flow costs as c_{ij}) from node s to each of the nodes in $N-\{s\}$ in an un-capacitated network. This viewpoint gives rise to the following linear programming formulation of the shortest path problem (Fig. 5.3).

$$Minimize \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.3a}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = \begin{cases} n-1 & \text{for } i = s \\ -1 & \text{for all } i \in N-\{s\} \end{cases} \tag{5.3b}$$

$$x_{ij} \geq 0 \text{ for all } (i,j)\in A. \tag{5.3c}$$

Fig 5.3: Linear programming formulation of shortest path problem

For a spatial network (e.g., a road network), all arc length must be positive. Hence, all algorithms for shortest path problem can be applied on spatial network. Some algorithms, such as Dijkstra's algorithm [Dijkstra (1959)], cannot be applied for solving shortest path problems with negative arc lengths.

The pseudo code of Dijkstra's algorithm is given in Fig 5.4:

```
algorithm Dijkstra;
begin
        $S := \Phi; \overline{S} = N;$
        $d(i) := \infty$ for each node $i \in N$;
        $d(s) := 0$ and pred(s) := 0;
        while $|S| < n$ do
        begin
                let $i \in \overline{S}$ be a node for which $d(i) = \min\{d(j) : j \in \overline{S}\}$;
                $S := S \cup \{i\}$;
                $\overline{S} := \overline{S} - \{i\}$;
                for each $(i, j) \in A(i)$ do
                        if $d(j) > d(i) + c_{ij}$ then $d(j) := d(i) + c_{ij}$ and
$pred(j) := i$;
        end;
end;
```

Fig.5.4: Pseudo Code of Dijkstra's algorithm

We implement the Dijkstra's Algorithm in C Programming Language (gcc 2.96) with operating system Redhat 7.2 running on a desktop computer. The configuration of the computer is AMD Duron 1.1G CPU, 256MB

SD-RAM with an IBM 9.1G SCSI Harddisk. We randomize 100 origin-destination (O-D) pairs and search the shortest path from the above node and arc table. The result is shown in Table 5.1:

Primary Key of Start Node	Primary Key of End Node	No. of iterations	Shortest Path Distance	Absolute Distance
11455	5463	3930	10877.903	7318.271
10687	10893	98	1146.848	594.809
12400	2789	9850	15351.885	11182.973
4684	10475	10370	13410.543	9809.265
3930	7602	4138	5935.353	4509.505
			⋮	
11368	12596	1773	6719.771	3501.052
11888	11337	5384	10081.420	5638.070

100 records

(total 100 O-D pairs)
Primary Key of Start Node: Unique key of start node
Primary Key of End Node: Unique key of end node
No. of iterations: No. of nodes visited before finding the shortest path (i.e. *S* in Fig. 5.4)
Shortest path distance: the shortest path distance of start and end points in metres
Absolute distance: the distance between start and end points in metres

Table 5.1: Result of calculating shortest path distance from 100 O-D pairs

The total iteration for finding these 100 O-D pairs is 708296. And the total time for finding the 100 O-D pairs shortest path is 2140 seconds. On average each shortest path would need around 7083 iterations and 21 seconds.

The time complexity of the original implementation of Dijkstra's algorithm is $O(n^2)$. Other implementation of Dijkstra's algorithm such as Fibonacci heap implementation have time complexity $O(m + n \log n)$ but it needs specific data structure which is different from ours. Other algorithms such as Floyd-Warshall Algorithm has the time complexity of $O(n^3)$. Indeed, to solve the problems in the network with 13794 nodes and 34170 arcs is a very tedious operation.

Some applications which require road networks for their problems have a different method to deal with this complex problem. Wong, et al. (2002) try to formulate the layered networks for Hong Kong road network.

Layered network is a directed network $G = (N, A)$ with a specified source node s and a specified sink node t if we can partition its node set N into k layers $N_1, N_2, \ldots, N_k$ so that $N_1 = \{s\}$, $N_k = \{t\}$, and for every arc $(i, j) \in A$, nodes i and j belong to adjacent layers (i.e., $i \in N_l$ and $j \in N_{l+1}$ for some $1 \le l \le k-1$). Fig. 5.5 is the graphical representation of the layered network model used in Wong, et al. (2002):

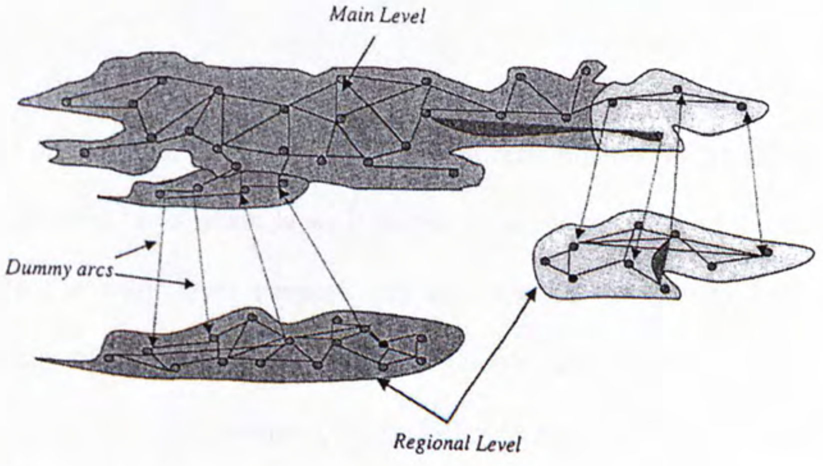


Fig. 5.5: Layered road network from Wong, et al. (2002)

In Wong, et al. (2002), the main level layer contains 691 nodes and 1932 directed arcs which cover the main roads for the whole Hong Kong territory. Other regional level layers contain no more than 200 nodes and 500 directed arcs. This can save many tedious calculations in finding the shortest path in the whole road network by calculating three shortest paths with hundreds of nodes.

Layered network is a possible solution for managing a large scale network such as the road network for the whole Hong Kong territory. It can be also easily applied to existing geographic information system which is described in Fig. 2 of chapter 1. However, a solution of shortest path problem in layered network may not be the optimal solution for the original network since one or more of the connected nodes must be used. The accuracy of shortest path depends on clustering of the large scale networks and the number of connected nodes between main level layer and regional level layers.

Another problem to apply layered network is that its design must be based on the main level layer first. But the determination of which arc should be contained in the main layer is also a problem: Decrease of the number of arc in the main layer network can decrease the complexity but also the accuracy of network operations. For example, applying the following main level layer (Fig. 5.6) which consists of main roads of Hong Kong territory with around 50 nodes is also feasible but may not be optimal for network operations:

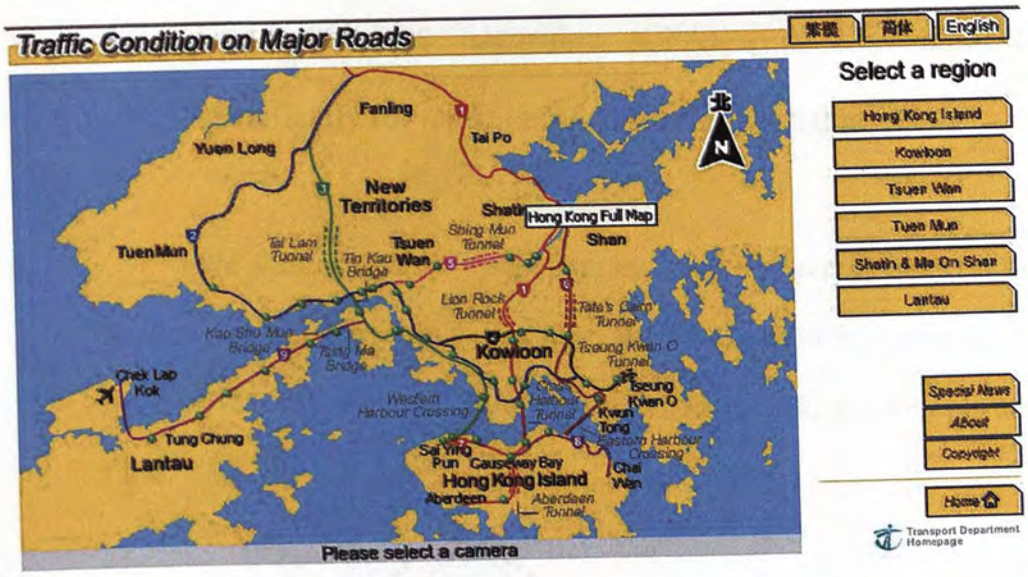


Fig. 5.6: Hong Kong major roads from traffic department, HKSAR Government
http://traffic.td.gov.hk/snapshots/eng/index.htm

Also, there is still not a standard layered network design for Hong Kong road network. (Digital Map RG1000 data from Lands Department only divides the data in region, but not in main level and regional level.) To implement a layered network for Hong Kong road network requires excessive computations.

In the following section we will propose a new algorithm which can easily partition a spatial network such as the Hong Kong road network and the existing algorithms for network operations can also be easily applied with slight modifications.

5.4 Connected-Page Algorithm (CPA) - A New algorithm specifically for partitioning/clustering spatial networks

We illustrate the new algorithm by using the area of Siu Lek Yuk, Shatin (Fig. 5.7). There are totally 12 nodes and 15 arcs in these area. i.e. G = (N, A) = (12, 15). We name the nodes in order (a, b, …, k, l) and arcs in order (I, II, …, XV).

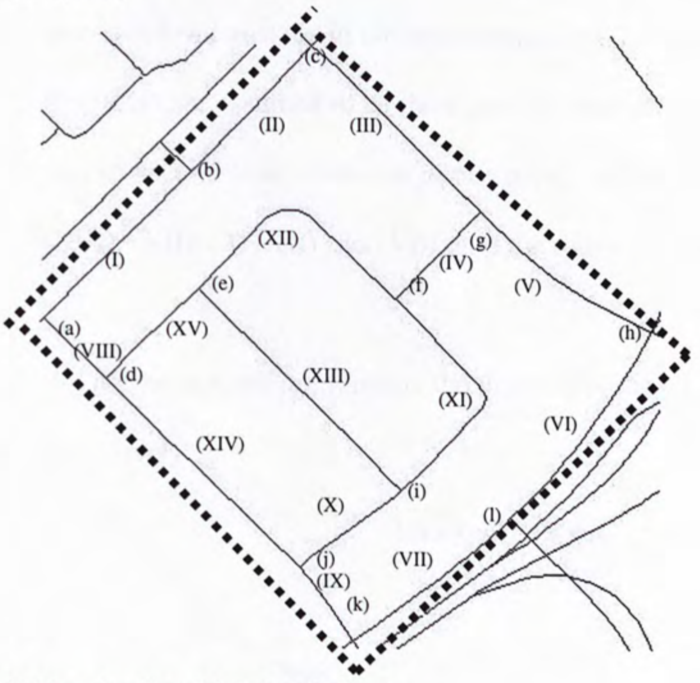


Fig. 5.7: Map data from Siu Lek Yuk, Shatin

Connected-Page Algorithm (CPA) for spatial network partition is given in the following. We assume that each page can store 4 nodes in maximum with all arcs connected for the 4 nodes.

Step 1: Randomize choosing a particular node (say (a)), find the nearest three nodes which is connected to node (a) (i.e. (b), (d) and (e))

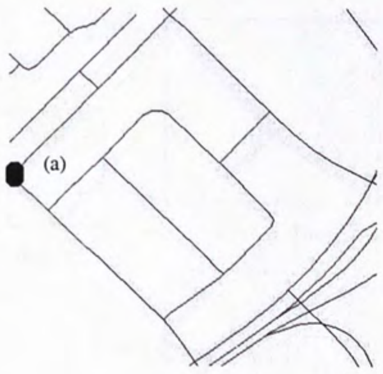
Step 2: Remove the node (a) and three nearest connected node ((b), (d) and (e)) in N. N_2 stores (b), (d) and (e). Node (a), (b), (d) and (e) and arcs (I), (VIII), (XV) are stored in one page (say Page (1)), also find the MBR of the page.

Step 3: Repeat the Step 1 and 2 by choosing a node in N_2 for step 1 until all nodes are stored in Pages (i.e. $N_2 = \varnothing$)

Step 4: Repeat the step 1 process by assuming the pages created in the previous level as node in the next level, (Page 1, 2 and 3 created in the first level are assumed to be three node in second level. And we found that in second level there are three "page" nodes and six arcs (XIV), (XIII), (XII), (XI), (II) and (VI)) until the root node is found.

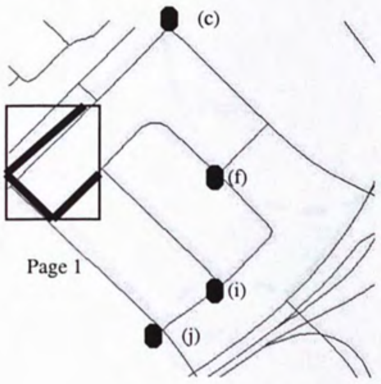
We use an example to illustrate the function of the algorithm.

Step 1: (first level)



Randomly choose a node (say node (a))

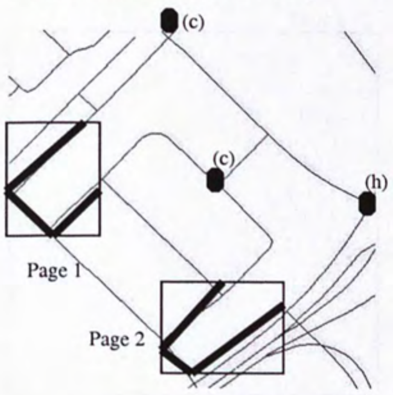
Step 2: (first level)



Page 1 stores nodes (a), (b), (d) and (e) and arc (I), (VIII) and (XV).

N_2 stores (c), (f), (i), (j).

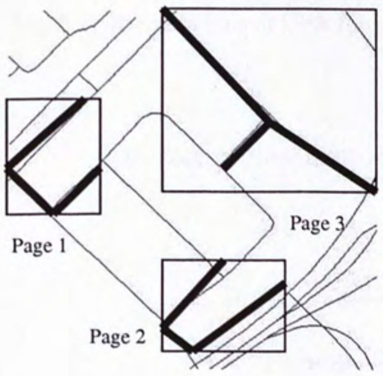
Step 1 and 2: (first level)



Page 2 stores nodes (i), (j), (k) and (l) and arc (VII), (IX) and (X).

N_2 removes (i), (j), and add (h). I.e.N_2 stores (c), (f) and (h).

Step 1 and 2: (first level)

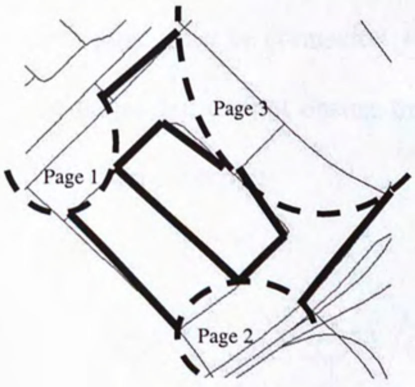


Page 3 stores nodes (c), (f), (g) and (h) and arc (III), (IV) and (V).

N_2 removes (c), (f), and (h). I.e.N_2 stores $\varnothing$.

Since all nodes are stored in different pages, next iterations will continue in second level

Step 4: (second level)

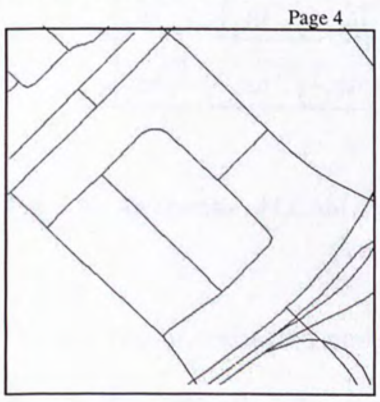


Assume Page 1, 2, 3 become dummy node in the next level, and then repeat step 1.

"Page 4" stores Page 1, 2, 3 and arc (XIV), (XIII), (XII), (XI), (II) and (VI).

Since all (dummy) nodes are stored , next iterations will continue in third level

Step 1: (third level)



Since "Page 4" is the root node, the algorithms can be stopped.

Fig. 5.8: The iterations of CPA for partitioning spatial networks

The tree representation of the new algorithm is:

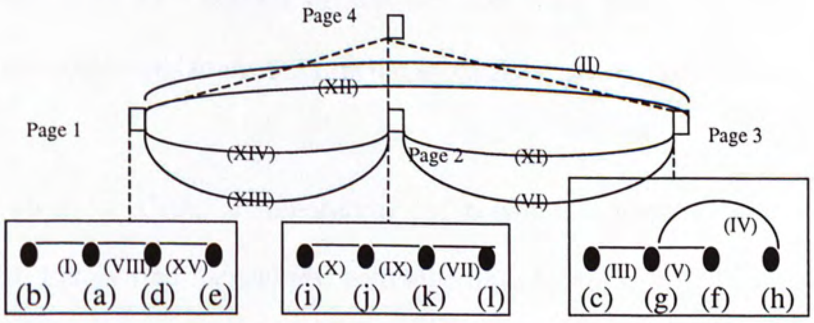


Fig. 5.9: Tree representation of CPA for partitioning spatial networks

The main difference of this new algorithm from CCAM is all nodes in each page must be connected. CCAM can only find the minimum cut between the pages but cannot ensure the nodes in the same pages are connected, Fig 5.10 is an example:

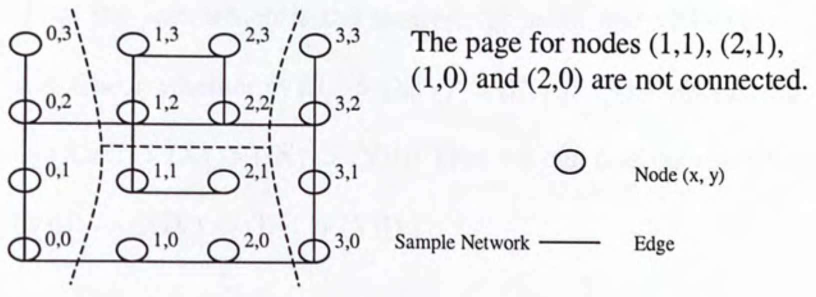


Fig. 5.10: An example of CCAM with a page have nodes not all connected.

The unconnected nodes within a page make some network operations failed. For example, if we want to find the shortest path or optimal path between the node (3,0) to (2,1), we would check the pages which contain these two nodes, although Page ((3,0), (3,1), (3,2), (3,3)) and Page ((1,1), (2,1), (1,0), (2,0)) are connected by arc ((2,0), (3,0)). The shortest path or optimal path could not be obtained by just consider these two pages. The whole network would need to visit to find the shortest path or optimal path.

But for CPA, all algorithms for network operations can be easily applied. Let us find the shortest path algorithm for (a) to (l). (This algorithm can also give the coordinates of origin and destination instead of specifying the index key of nodes.):

Step 1: Check which pages contained the origin and destination:

(i.e. Origin: Page 4(root) -> Page 1, Destination: Page 4(root) -> Page 2)

Step 2: Find the (shortest) path(s) connected for the origin page and destination page (i.e. arc (XIV) and (XIII)).

Step 3: Find the path which is the shortest for using path (XIV) and (XIII). (i.e. Check whether (VIII) -> (XIV) -> (IX) -> (VII) or (VIII) -> (XV) -> (XIII) -> (X) -> (IX) -> (VII)) Then we can find the shortest path is (VIII) -> (XIV) -> (IX) -> (VII).)

In the remaining section, we discuss insert and delete operations for CPA:

Insert:

If a node and corresponding arc(s) are inserted, firstly we need to check the nearest leaf pages (those pages contain nodes) can accommodate this new node, if not, then a new leaf page can be created. If only an arc is inserted, then the arc can be inserted in leaf pages or non-leaf pages. (If a new page is inserted, updating of arc connection is also needed in leaf pages.)

For example, node (m) and arc(XVI) is inserted (Fig. 5.11)

(arc (XVI) connect node (m) and (a):

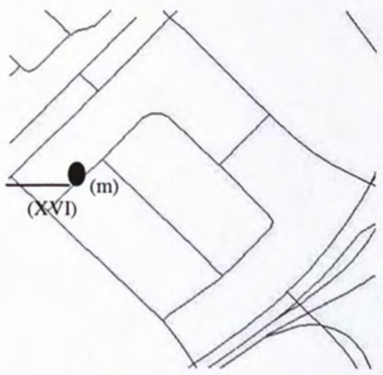


Fig 5.11: Insert new node (m) and arc (XVI)

The tree representation of CPA will be updated as follows (Fig. 5.12):

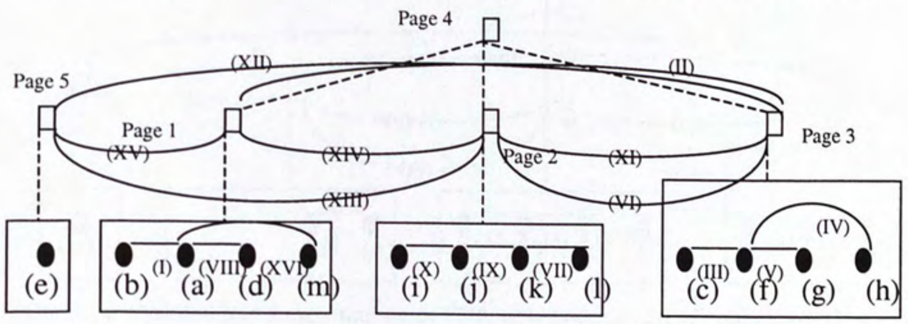


Fig 5.12: Updated tree representation of CPA after node (m) and arc (XVI) inserted

Delete:

If a node and/or corresponding arc(s) are deleted, we need to check whether the deleted node would change the connectivity within a leaf page and also the connectivity in non-leaf pages. A leaf page needs to split if the node in this page is not connected. If a page is no longer connected with other then this part of tree would need to move to the root node. (For this reason, a root node would require to have unlimited node connected.)

For example, node (a), arc(I) and arc(VIII) are deleted (Fig. 5.13):

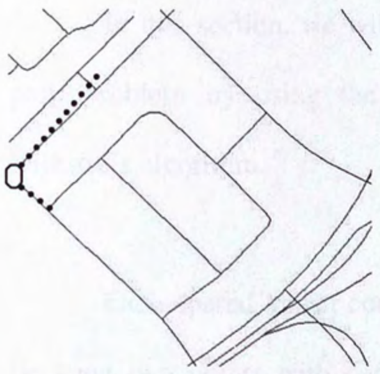


Fig 5.13: Delete node (a), arc (I) and arc(VIII)

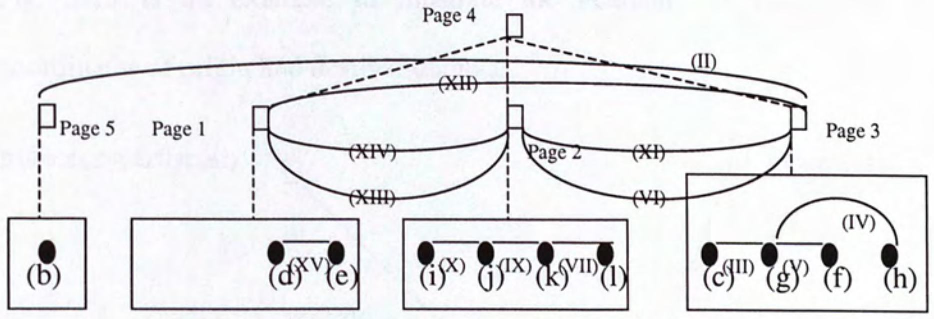


Fig 5.14: Updated tree representation of CPA after node (a), arc (I) and arc (VIII) are inserted

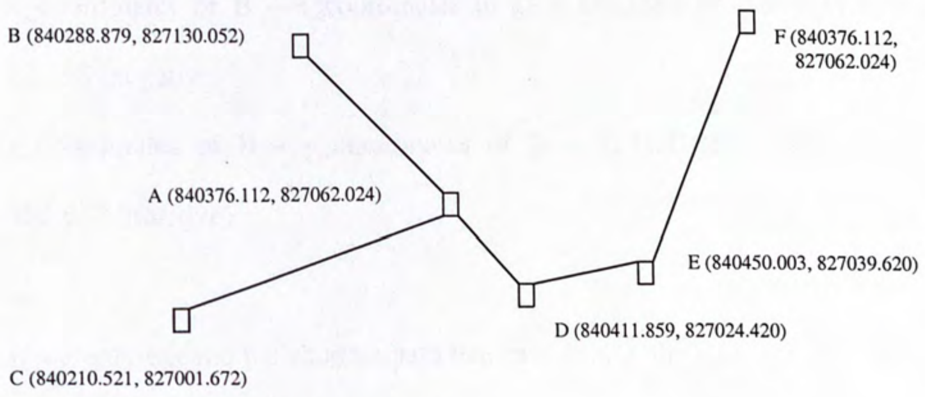
Indeed the insert and delete operation of CFA is quite complicated. But mostly spatial network would not have frequently changed and it is quite easy to rebuild the CFA tree after any insert or delete.

5.5 A New Simple heuristic for shortest path problem for spatial network

In this section, we will introduce a new simple heuristic for shortest path problem by using the characteristics of spatial network based on Dijkstra's algorithm.

Each spatial object contains coordinates. For finding the shortest path between two points with coordinates, we can compare the coordinates for searching.

Fig. 5.15 is an example to illustrate the heuristic by considering the coordinates of origin and destination node.



B (840288.879, 827130.052)
F (840376.112, 827062.024)
A (840376.112, 827062.024)
E (840450.003, 827039.620)
D (840411.859, 827024.420)
C (840210.521, 827001.672)

Fig. 5.15: A sample spatial network

If we want to find the path from node D to node B in Fig 5.15, by Dijkstra's algorithm, the path will be D -> A -> B. And the node iterated will be (D, A, E, B).

However if we divide the network in four regions where the start node (D) is the origin:

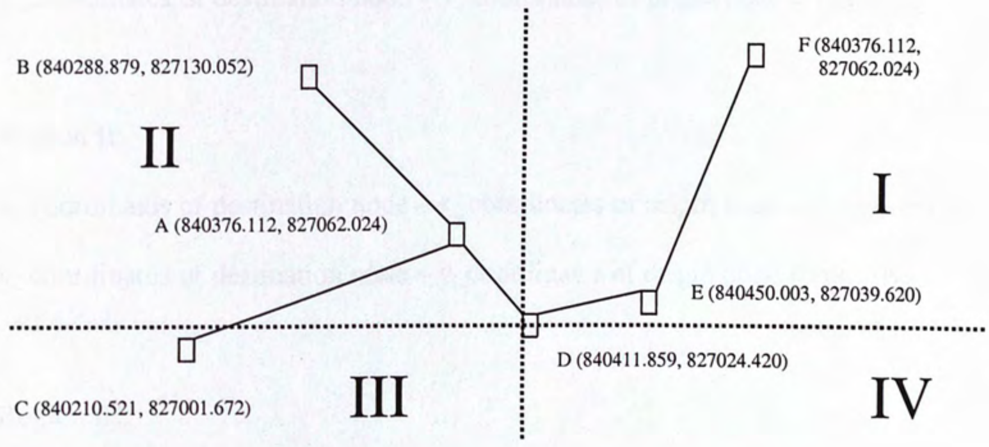

B (840288.879, 827130.052)
F (840376.112, 827062.024)
II
I
A (840376.112, 827062.024)
E (840450.003, 827039.620)
D (840411.859, 827024.420)
C (840210.521, 827001.672)
III
IV

Fig. 5.16: Cluster the network in Region I to IV

Comparing the coordinates of B and D:

x_coordinates of B – x_coordinates of D = 840288.879 – 840411.859 = -122.98 (negative)

y_coordinates of B – y_coordinates of D = 827130.052 – 827024.420 = 105.632 (positive)

If we only expand the shortest path tree in region 2, the path will also be D -> A -> B. And the node iterated (i.e. *S* in Fig. 5) will be (D, A, B). i.e. Iteration of E is saved.

In general, we can start the iteration in a particular region based on following conditions:

Region I:

x_coordinates of destination node – x_coordinates of origin node = (positive) y_coordinates of destination node – y_coordinates of origin node = (positive)

Region II:

x_coordinates of destination node – x_coordinates of origin node = (negative)

y_coordinates of destination node – y_coordinates of origin node = (positive)

Region III:

x_coordinates of destination node – x_coordinates of origin node = (negative)

y_coordinates of destination node – y_coordinates of origin node = (negative)

Region IV:

x_coordinates of destination node – x_coordinates of origin node = (positive)

y_coordinates of destination node – y_coordinates of origin node = (negative)

In the best case, this heuristic can saved third-fourth of the iterations compared with traditional Dijkstra's algorithm.

This heuristic uses the same algorithm of Dijkstra's algorithm. But in searching *i* in the line "let $i \in \overline{S}$ be a node for which $d(i) = \min\{d(j) : j \in \overline{S}\}$;" (Line 8 of Pseudo Code of Dijkstra's algorithm in Fig 5.4). The choice of *i* would only choose the node in the region specified.

Deficiencies in the heuristic:

If we search the path from D to C, the path cannot be found in region III. In this case, this heuristic fails. Even the path can be found, this heuristic cannot ensure that the path found is optimal, while the path found by Dijkstra's algorithm is an optimal path.

5.6 Summary

In this chapter we have proposed a new algorithm - Connected Page Algorithm (CPA) for storage of spatial networks. This algorithm based on the connectivity instead of proximity (e.g. STR Packed R-Tree) and we found that the overlapping area can be saved. CPA is also easier to implement than CCAM (Connectivity Clustered Access Method) and more suitable for network operations in spatial network. In the next chapter we would extend the usage of CPA. A simple heuristic which is based in the characteristics of spatial network (i.e. each spatial object has the corresponding coordinates) is also introduced.

Table 5.2 shows the comparison of R-Tree STR Packed R-Tree, CCAM and Connected Page Algorithm:

	R-Tree	STR Packed R-Tree	CCAM	Connected Page Algorithm
Data type being used	General spatial data	General spatial data	Spatial Network	Spatial Network
Inserting procedure	Dynamic	Static	Static	Static
Overlap of non-leaf MBR	Many overlap	Less overlap	---	No overlap
Preprocessi ng of data	No	No	Yes (network representations)	Yes (network representations)
Network computations	No support	No support	Support	Support
Multi-layer clustered	Yes	Yes	No	Yes

Table 5.2: Comparison of R-Tree, STR Packed R-Tree, CCAM and Connect Page Algorithm

Chapter 6

Nearest Neighbor Queries

6.1 Introduction

Nearest Neighbor Queries is an important application in geographical information systems. Examples include finding the nearest gas station, car park, etc. Nearest Neighbor is also essential in delivery planning systems. Examples include finding the nearest depot to deliver the product to particular customer and determine the nearest parking place to deliver the product to the customer.

The following is the nearest neighbor queries by R-tree proposed in Roussopoulos, et al. (1995), Papadopoulos, et al. (1997), Cheung, et al. (1998), Tao, et al. (2002).

Define (i)Point ***P***: the query point

(ii)Rectangle ***R***: the directory rectangle or minimum bounding rectangle(MBR) in R-tree or any other algorithms using MBR

(iii)Vertex *V*: the nearest vertex of ***R*** from ***P***.

(iii)***Min-distance(P, R)*** = 0 if P is inside ***R*** or on boundary of ***R***

Min-distance(P, R) = Euclidean distance between ***P*** and any edge of ***R*** if ***P*** is outside ***R***

(iv)***Min-Max-distance (P, R)*** is the distance of ***P*** from the farthest point on any face of the ***R*** containing vertex ***V***

Min-distance(P, R) represents the lower bound on the distance of any object inside ***R*** from ***P*** and it provides an optimistic ordering of subtrees in nearest neighbor search. ***Min-max-distance(P, R)*** guarantees that there is an object ***O*** inside rectangle ***R*** in the R-tree such that ***(O, P) ≤ Min-max-distance(P, R)***, and it provides a pessimistic ordering.

Search pruning strategies can be based on these measures, as well. For example, an MBR ***M*** can be eliminated if there is another MBR ***M'*** such that with min-distance ***(P, M) > Min-max-distance(P, R')***. An MBR ***M*** can also be eliminated if there is an object ***O*** such that distance ***(P, O) < min-distance(P, M)***. Finally an object ***O*** can be eliminated if there is an MBR ***M*** such that distance ***(P, O) > min-max-distance (P, M)***.

The search algorithm for nearest neighbor starts with the root node of the R-tree and traverses the tree. For example, a breadth first traversal of the R-tree will visit MBRs of the children of the interior nodes of current node for pruning using the above rules. The remaining children will be expanded in the next iteration. The final iteration will have a set of leaf nodes (database object level) from the MBRs that survive level-wise pruning. The algorithm needs to compute the distance of each leaf from query point P to determine the nearest neighbor.

6.2 Modified algorithm for Nearest Neighbor Queries

We can apply nearest neighbor queries to many situations such as finding the shortest path from one building to another in Hong Kong, we need to find the nearest nodes of an origin building and destination building from the Hong Kong Road Network.

It is a tedious operation if we need to find nearest nodes of origin building and destination building once we need to find the shortest path of these two buildings. Some literatures (e.g. [Wong, et al. 2002]) use a mapping table, i.e. setting many-to-one relationship for buildings to nodes. For example, Fig. 6.3 is a mapping table for buildings to nodes in Siu Lek Yuk, Shatin.

In Wong, et al. (2002), a distribution and delivery system named VANS is created. VANS involves to find the vehicle routing solution for customers address (buildings).

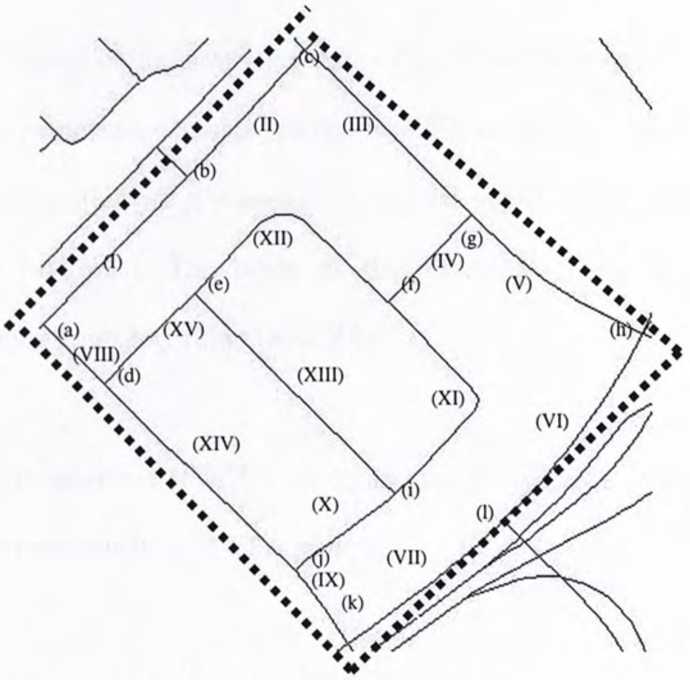


Fig. 6.1: Map data with node and arc index

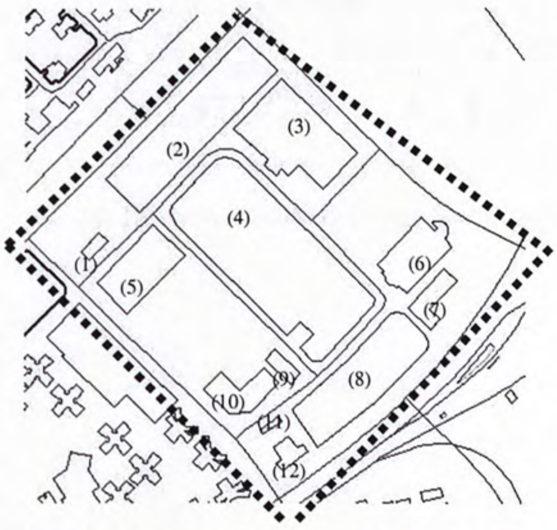


Fig. 6.2: Map data with polygon (building) index

Building	Node
1	d
2	b
3	g
4	i
5	e
6	f
7	i
8	l
9	i
10	j
11	j
12	k

Fig. 6.3: Mapping table for building to node

By using mapping table, if the shortest path from building (1) to (8) is needed, the shortest path from node (d) to (l) would be found instead.

Based on the new algorithm – Connected Page Algorithm (CPA), we extend the nearest neighbor queries for CFA to enhance the usage of CFA in more applications (e.g. mapping the spatial objects such as buildings to the spatial networks). The name of this extension is Nearest Neighbor in Connected Page Algorithm (NNCPA).

To illustrate NNCPA, we try to map the building shown in Fig. 6.2 to the tree representation of CPA (Fig. 6.4).

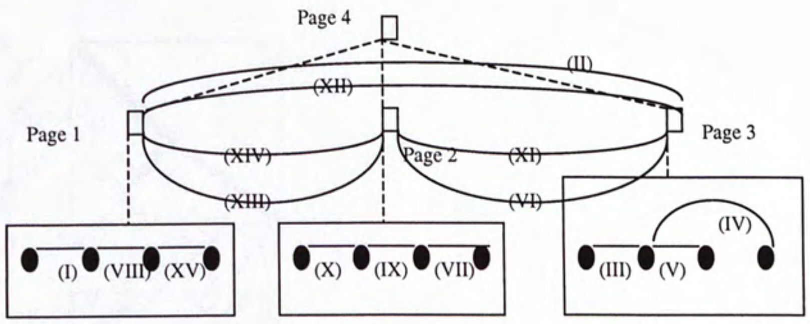


Fig. 6.4: Tree representation of CPA for partitioning spatial networks

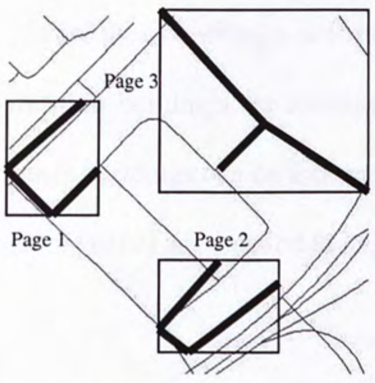


Fig 6.5: Creating the first level in CFA for Fig. 6.1

In generating the first level of CFA (Page 1, 2 and 3), we can compare the Minimum Bounding Rectangle (MBR) of building with the MBR of Page 1, 2 and 3. If any MBR of building can be included in any page, those building will be stored or indexed in that page. Fig 6.6 illustrates this comparison and indexing:

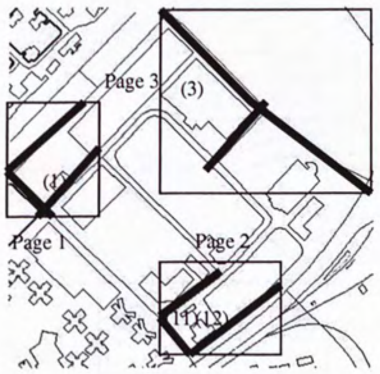


Fig 6.6: Comparing the first level in CPA for Fig. 6.1

In Fig 6.6, we can see the first level CPA page (Page 1, 2 and 3). Page 1 covers building (1). Page 2 covers building (11) and (12). Page 3 covers building (3). Since other buildings cannot be totally covered by the first level pages, these buildings will be continuously checked in the next level pages until all buildings are covered and indexed in CPA pages. In this example all other buildings can be indexed in the next level page, i.e. Page 4. i.e. Building (2), (4)-(10) are indexed in Page 4.

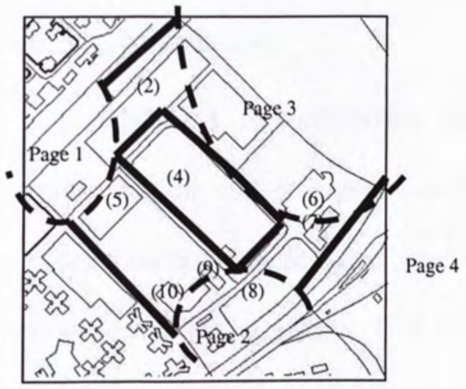


Fig 6.7: Comparing the second level in CPA for Fig. 6.1

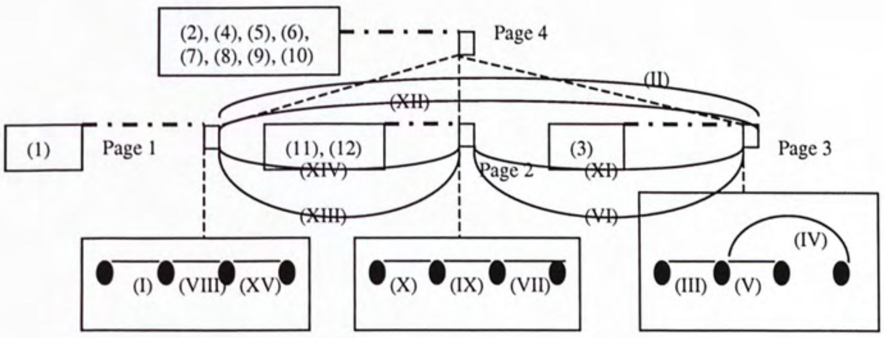


Fig. 6.8: Tree representation of CPA for partitioning spatial networks

In solving the shortest path in buildings of NNCPA, the operation is similar to solving shortest path of nodes in CPA. For example, if we want to find the shortest path from building (1) to building (12), we can find the indexed pages of building (1) and building (12), i.e. Page 1 to Page (2). Two paths, (XIV), (XIII) connect Page 1 and Page 2. And now we can choose the path between (XV) -> (XIII) -> (X) -> (IX) or (XIV) -> (IX) and find the shortest path between these two paths.

6.3 Summary

NNCPA is suitable for real-time road guidance for vehicles, e.g. transportation and emergency services. If one road is blocked by traffic congestion or traffic accident, there are many alternatives to choose which can also go to the destination. NNCPA is more flexible than just mapping the buildings to specific nodes and NNCPA can also save many calculations in network operations for spatial networks with thousands of nodes.

Chapter 7

Conclusion and Future Work

In this thesis we address spatial data and spatial databases. Since most spatial data are in proprietary formats, we propose to use Geography Markup Language (*GML*) for indexing scheme for spatial access which can be used in any spatial data access algorithms.

STR Packed R-Tree [Leutenegger, et al. (1996)] is efficient to store general point and polygon spatial data. But it is not efficient for line segments.

CCAM [Shekhar, et al. (1997)] focuses on the storage of a spatial network based on connectivity of spatial network. Based on CCAM, we propose a new algorithm - Connected Page Algorithm (CPA) for clustering a spatial network. We also illustrate how to extend the capabilities of CPA to nearest neighbor queries

7.2 Future Work

A spatial network we study in this thesis is in static mode. There are many real-life problems that is dynamic in nature. For example, a real road network consists of real time traffic information. To model the real road network for applications such as real time road guidance, additional algorithms and heuristics would be needed to develop for spatial network.

Dynamic insert, delete and update of spatial networks are also important operations in spatial networks. The challenge is not only optimized the network storage but also the spatial data query.

In this thesis we also deal with the shortest path problem for a spatial network. There are many other network operations such as traveling salesman problem, vehicle routing problem, maximum flow problem, minimum cut problem which can also be applied in a spatial network.

Geotools is one of the open source mapping toolkit for spatial data which support GML. To make some system implementation for spatial network, nearest neighbor queries we suggest making use of this toolkit. Fig. 7.1 shows a webpage which use *Geotools* for web-based spatial data visualization.

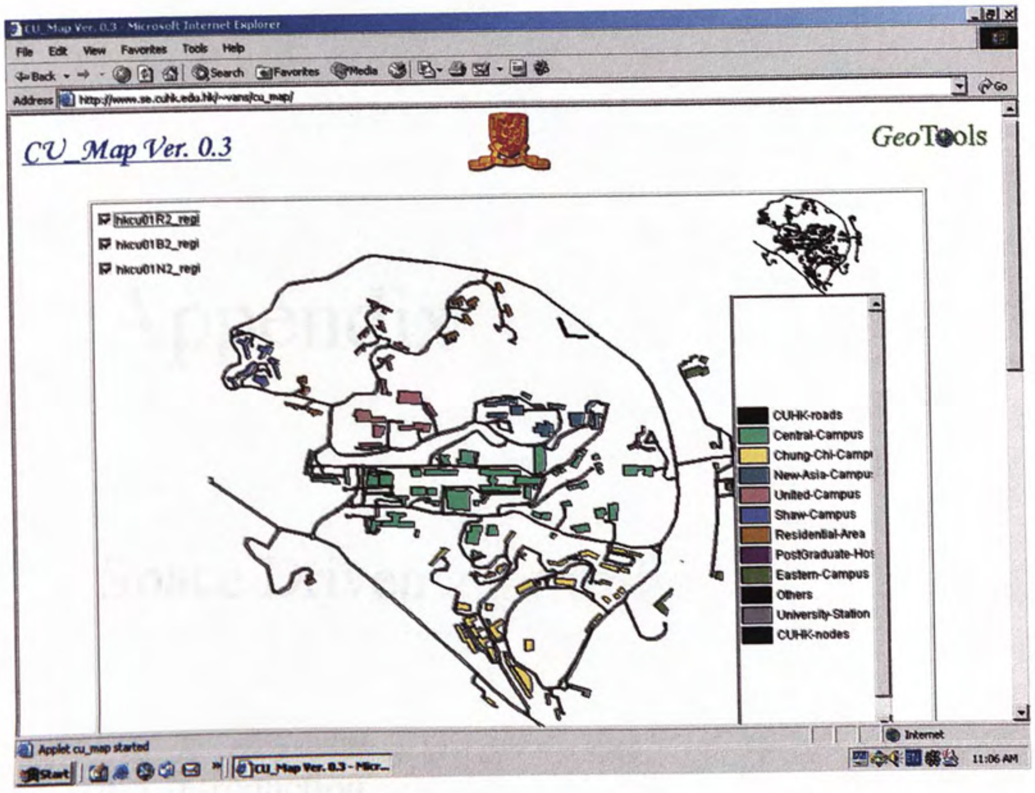


Fig.7.1: A webpage which make use of *Geotools* for spatial data visualization
http://www.se.cuhk.edu.hk/~vans/cu_map/

Appendix

Space Driven algorithms

A.1 Introduction

Spatial Access Methods include space driven algorithms and data driven algorithms. In chapter 2 we have reviewed data driven algorithms as the research direction of this thesis concentrates on data driven methods. In this appendix we discuss data driven algorithms. Three data driven algorithms are introduced: fixed grid, Z-curve, Hilbert Curve.

The grid file was initially designed for indexing objects on the value of several attributes. Unlike the B-tree, it is a multikey index that supports queries on any combination of these attributes. Linear structures enable a simple integration with the B+-tree of existing database management systems.

A.2 Fixed grid

In fixed grid [Bentley, et al. (1979), Nievergelt, et al. (1984)], the search space is decomposed into rectangular cells. The resulting regular grid is an n_x X n_y array of equal-size cells. Each cell *c* is associated with a disk page. Point *P* is assigned to cell *c* if the rectangle *c.rect* associated with cell c contains *P*. The objects mapped to a cell *c* are sequentially stored in the page associated with *c*.

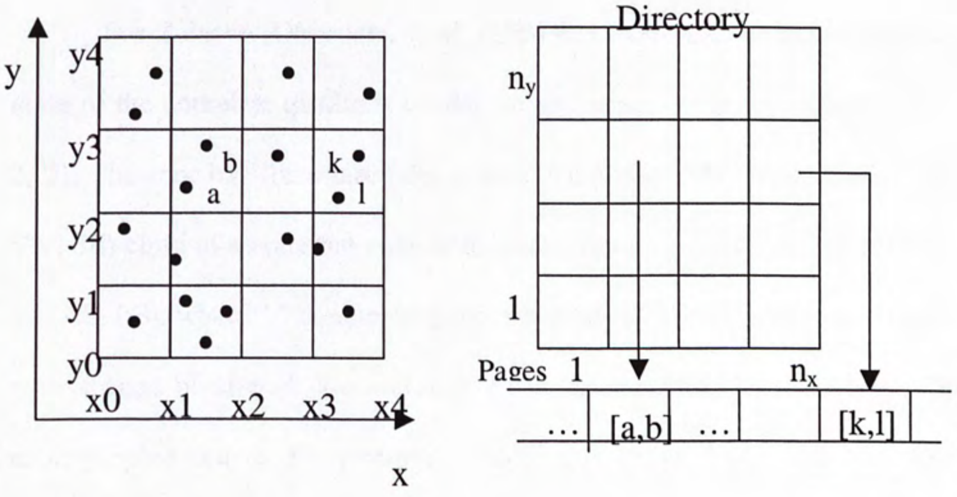


Fig. A.1: Fixed Grid

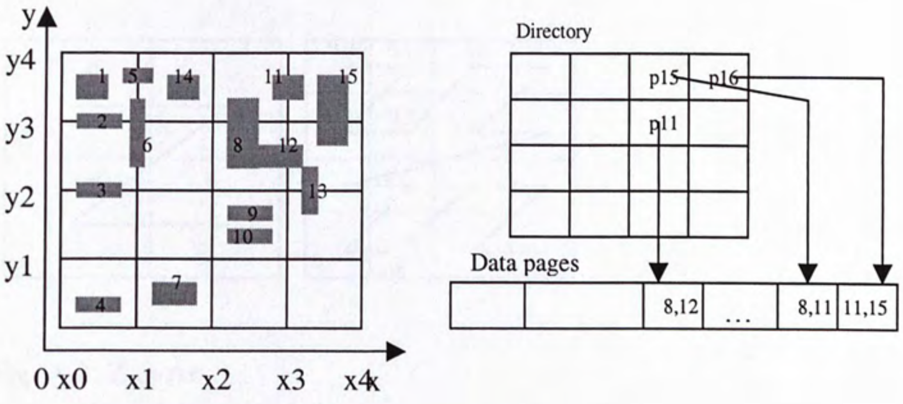


Fig. A.2: A fixed grid for rectangle indexing.

A.3 Z-curve

For Z-curve [Orenstein, et al. (1984)], a label is associated with each node of the complete quadtree, chosen among strings over the alphabet (0, 1, 2, 3). The root has for a label the empty string. The NW (respectively, NE, SW, SE) child of an internal node with label *k* has for a label *k*.0 (respectively, *k*.1, *k*.2, *k*.3), where "." denotes string concatenation. Then the cells are labeled with strings of size *d*. We can sort the cells according to their labels (in lexicographic order). For example, choosing a depth *d* = 3 and ascendant order, cell 212 is before cell 300 and after cell 21. The ordering NW, NE, SW, SE justifies its *z-order* name.

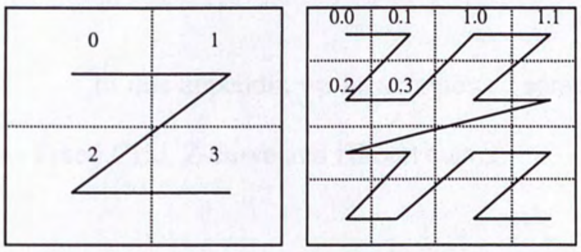


Fig. A.3: Z-curve

A.4 Hilbert curve

Hilbert curve likes Z-curve but the shape is Π but not Z. Unlike Z-curve, the Hilbert curve consists of segments of uniform length; that is, we never have to "jump" to a distant location while scanning the cells (Fig. A.4). It is easy to see that in both cases there exist some unavoidable situations in which two objects are close in the 2D space, but far from one another on the space-filling curve.

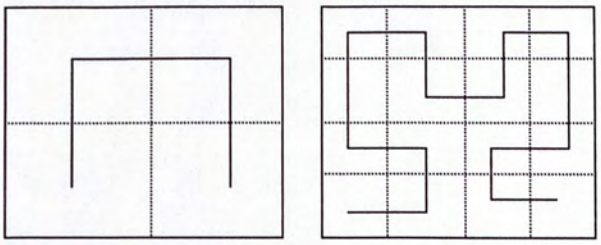


Fig. A.4: Hilbert curve

A.5 Conclusion

In this appendix we have reviewed some space-driven algorithms such as Fixed Grid, Z-curve and Hilbert curve.

Bibliography

Abel D., Ooi B. C. (1993). *Advances in spatial databases : Third International Symposium*, SSD '93, Singapore.

Ahuja R. K., Magnanti T. L., Orlin J. B. (1993). *Network Flow: Theory, Algorithms, And applications*, Prentice Hall.

Bailey T. C., Gatrell A. C. (1995). *Interactive Spatial Data Analysis*, Longman Scientific & Technical, New York.

Beckmann N., Kriegel H. P., Schneider R., Seeger B. (1990). *The R*Tree: An Efficient and Robust Access Method for Points and Rectangles*, In Proc. ACM SIGMOD Intl. Symp. on the Management of Data, pages 322-331.

Bentley J. L., Friedman J. H. (1979). *Data Structures for Range Searching*, ACM Computing Surveys, 11(4), 1979.

Bernhardsen T. (2002). *Geographic Information Systems, An Introduction*, John Wiley & Sons, Inc., New York.

Bertino E., Castano S., Ferrari E., Mesiti M. (1999). *Controlled Access and Dissemination of XML Documents*, In Proceedings of 2nd ACM Workshop on Web Information and Data Management,Kansas City (Missouri), pp. 22-27.

Cheung K., Fu A. (1998). *Enhanced Nearest Neighbor Search on the R-tree.* ACM SIGMOD Record, 27(3): 16-21.

Chou Y.L., Romejin H.E., Smith R.L. (1998). *Approximating Shortest Paths in Large-Scale Networks with an Application to Intelligent Transportation Systems*, INFORMS Journal on Computing, Vol. 10, No. 2, Spring 1998.

Dijkstra, E. (1959). *A note on two problems in connexion with graphs*, Numeriche Mathematics 1, pp.269-271.

Eggenhofer M. J., Herring J. R. (eds.) (1995). *Advances in spatial databases : 4th international symposium*, SSD '95, Portland, ME, USA.

Fu L., Rilett L.R. (1998). *Expected Shortest Paths in Dynamic and Stochastic Traffic Networks*, Transportation Research B, Vol. 32, No. 7, pp. 499-516.

Garcia R. Y. J., Lopez M. A., Leutenegger S. T. (1998). *On Optimal Node*

Splitting for R-trees, Proc. of VLDB, New York, USA.

Geotools - open source mapping toolkit from sourceforge.net *http://geotools.sourceforge.net*

Graves M. (2001). *Designing XML Databases*, Prentice Hall, USA.

Green D., Boossomaier T. (2002). *Online GIS and Spatial Metadata*, Taylor & Francis, London.

GML . http://opengis.net/gml/01-029/GML2.html, OGC Document Number: 01-029, OpenGIS® Implementation Specification, 20 February 2001.

Gunther O., Schek H. J. (eds.) (1991). *Advances in spatial databases : 2nd Symposium*, SSD '91, Zurich, Switerland.

Guting R. F. (1994). *GraphDB: Modeling and Querying Graphs in Databases*, Proc. of VLDB, Santiago, Chile.

Guting R. H., Papadias D., Lochovsky F. (eds.) (1999). *Advances in spatial databases* : 6th International Symposium, SSD'99 : Hong Kong, China.

Guttman A. (1984). *R-Trees: A Dynamic Index Structure for Spatial Searching*, In Proc. ACM SIGMOD Intl. Symp. on the Management of Data, pages 45-57.

Jagadish H.V. (1990). *On Indexing Line Segments*, Proceedings of the 16th VLDB Conference, Brisbane, Australia.

Jensen C. S., Schneider M., Seeger B., Tsotras V. J. (eds.) (2001). *Advances in spatial and temporal databases* : 7th International Symposium, SSTD 2001, USA.

Kha D.D., Yoshikawa M., Uemura S. (2001). *An XML Indexing Structure with Relative Region Coordinate*, 17th International Conference on Data Engineering April 02 – 06, 2001.

Kollios G., Gunopulos D., Tsotras Y. J. (1999). *On Indexing Mobile Objects*, in Proc. ACM PODS, 1999.

Leutenegger S.T., Edgington J. M., Lopez M. A. (1996). *STR: A Simple and Efficient Algorithm for R-Tree Packing*, In Proc. IEEE Intl. Conf. on Data Engineering (ICDE).

Lin H., Huang B. (2001). *SQL/SDA: A Query Language for Supporting*

Spatial Data Analysis and Its Web-based Implementation, IEEE Transaction of Knowledge and Data Engineering (TKDE), July/August, 2001.

Manolopoulos Y., Theodoridis Y., Tsotras V. J. (2000). *Advanced Database Indexing*, Kluwer Academic Publishers, London.

Mapinfo Corporation(2002a). *Mapinfo MapXtreme 4.0 documentation* http://www.mapinfo.com/common/docs/mapxtreme-java_edition-4.0-dev-pdf-none-eng/MXTJ40DevGuide.pdf

Mapinfo Corporation(2002b). *Mapinfo Professional 7.0 documentation* http://www.mapinfo.com/common/docs/mipro/mipro_70_users.pdf

Minieka E. (1978). *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, Inc. New York.

Newton P. W., Zwart P. R., Cavill M. E. (1992). *Networking Spatial Information Systems*, Belhaven Press, Britain.

Nievergelt, J., Hinterger, H., Sevick K. C. (1984). *The Grid File: An Adaptable Symmetric Multikey File Structure*, ACM Trans. on Database Systems, 9(1):38-71, 1984.

Orenstein J., Merrett T. H., (1984). *A Class of Data Structures for Associative Searching*, In Proc. ACM Intl., Symp. on Principles of Database Systems(PODS), 1984, pp. 181-190.

Pagel B. U., Six H. W., Winter M. (1995). *Window Query-Optimal Clustering of Spatial Objects*, PODS 1995: 86-94.

Papadias D., Theodoridis Y., Sellis T. K., Egenhofer M. (1995). *Topological Relations in the World of Minimum Bounding Rectangles: A Study with R-Trees*, In Proc. ACM Intl. Symp. on the Management of Data, pages 92-103.

Papadopoulos A., Manolopoulos Y. (1997). *Performance of Nearest Neighbor Queries in R-trees*, ICDT 1997, pp. 394-408.

The Paradise Team (1995). *Paradise: A Database System for GIS Applications*, SIGMOD Conference, San Jose, USA, 1995, pp.485.

Patel J., Yu J.B., Kabra N., Tufte K., Nag B., Burger J., Hall N., Ramasamy K., Lueder R., Ellmann C., Kupsch J., Guo S., Larson J., DeWitt D., Naughton J. (1997). *Building a Scalable Geo-Spatial DBMS: Technology, Implementation, and Evaluation*, SIGMOD Conference, AZ, USA, 1997, pp.336-347.

Peng Z. R., Tsou M.H. (2003). *Internet GIS, Distributed Geographic Information Services for the Internet and Wireless Networks*, John Wiley & Sons, Inc., Hoboken.

Richardson D. E., Oosterom P. V. (eds.) (2002). *Advances in spatial data handling : 10th International Symposium on Spatial Data Handling*, Berlin, New York. Springer.

Rigaux P., Scholl M., Voisard A. (2002). *Spatial Databases with Applications to GIS*, Morgan Kaufmann Publishers.

Roussopoulos N., Kelley S., Vincent F. (1995). *Nearest Neighbor Queries*, Proceedings of ACM Sigmod , May 1995.

Scholl M., Voisard A. (eds.) (1997). *Advances in spatial databases : 5th international symposium*, SSD '97, Berlin, Germany.

Sellis T., Roussopoulos N., Faloutsos C. (1987). *The R+-Tree: A Dynamic Index for Multi-dimensional Objects*, In Proc. Intl. Conf. On Very Large Data Bases (VLDB), pages 507-518.

Shekhar S., Chawla S. (2003). *Spatial Databases: A Tour*, Prentice Hall.

Shekhar S., Liu D.R. (1997). *CCAM: A Connectivity-Clustered Access Method for Networks and Network Computations*, IEEE Transactions on Knowledge and Data Engineering, Vol. 9, No. 1, pages 102-119.

Shi W., Goodchild M. F., Fisher P.F. (1999). *Proceedings of The International Symposium on Spatial Data Quality '99*, Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong.

Shi W., Goodchild M. F., Fisher P.F. (2003). *Proceedings of The 2nd International Symposium on Spatial Data Quality '03*, Department of Land Surveying and Geo-Informatics, The Hong Kong Polytechnic University, Hong Kong.

Tao Y., Papadias D., Shen Q. (2002). *Continuous Nearest Neighbor Search*, Proceedings of the 28th VLDB Conference, Hong Kong, China.

Tao Y., Papadias D. (2002). *Time-Parameterized Queries in Spatio-Temporal Databases*, ACM SIGMOD 2002, June4-6, Madison, Wisconsin, USA.

Wei Z. K., Oh Y. H., Lee J. D., Kim J. H., Park D. S., Lee Y. G., Bae H. Y. (1999). *Efficient Spatial Data Transmission in Web-Based GIS*, Kansas City, Missouri, United States, Pages: 38 – 42.

Wong J. C.F., Fung T.W., Cheng C.H., Leung J. M.Y. (2002). *A Map-based Decision-support System for Delivery Planning in Hong Kong*, Technical Report SEEM2002-03, Dept. of Systems Engineering and Engineering Management, The Chinese University of Hong Kong.

Wu W. Paper Link related with spatial data: *http://www-users.cs.umn.edu/~wuw/8705Paper.html*

XML http://www.w3.org/XML/

Yu H. B. (2001). *Data Organization for Routing on the Multi-modal Public Transportation System: A GIS-T Prototype of Hong Kong Island*, Thesis of Master of Philosophy, Department of Geography and Resources Management, The Chinese University of Hong Kong.

Zhan F.B., Noon C.E. (1998). *Shortest Path Algorithms: An Evaluation using Real Road Networks*, Transportation Science B, Vol. 32, No. 1, pages 65-73.