



THE HONG KONG  
POLYTECHNIC UNIVERSITY

香港理工大學

Pao Yue-kong Library

包玉剛圖書館

---

## Copyright Undertaking

This thesis is protected by copyright, with all rights reserved.

**By reading and using the thesis, the reader understands and agrees to the following terms:**

1. The reader will abide by the rules and legal ordinances governing copyright regarding the use of the thesis.
2. The reader will use the thesis for the purpose of research or private study only and not for distribution or further reproduction or any other purpose.
3. The reader agrees to indemnify and hold the University harmless from and against any loss, damage, cost, liability or expenses arising from copyright infringement or unauthorized usage.

If you have reasons to believe that any materials in this thesis are deemed not suitable to be distributed in this form, or a copyright owner having difficulty with the material being included in our database, please contact [lbsys@polyu.edu.hk](mailto:lbsys@polyu.edu.hk) providing details. The Library will look into your claim and consider taking remedial action upon receipt of the written requests.

# **Algorithmic Development of an Optimal Path Computation Model Based on Topographic Map Features**

Tang, Yuet Fong Mandy

Master of Philosophy

Department of Land Surveying and Geo-Informatics  
The Hong Kong Polytechnic University

September, 2005

A thesis submitted in partial fulfilment of the requirements for  
the Degree of Master of Philosophy



Pao Yue-kong Library  
PolyU · Hong Kong

## CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

\_\_\_\_\_ (Signed)

TANG YUET FONG MANDY (Name of Student)



## Contents

<b>Acknowledgements .....</b>	<b>9</b>
<b>Chapter 1 Introduction .....</b>	<b>10</b>
1.1 Background.....	10
1.2 Objectives .....	12
1.3 Scope.....	13
1.4 Significance .....	13
1.5 Methodology .....	14
1.6 Thesis Organization .....	16
<b>Chapter 2 A Review of Graph Theories .....</b>	<b>17</b>
2.1 Fundamental Theories.....	17
2.1.1 Basic Graph Definition .....	17
2.2 Data Structure for Graphs and Networks.....	19
2.2.1 Topology .....	19
2.2.2 Shortest Path Problems and Algorithms .....	20
<b>Chapter 3 Network Analyse in GIS.....</b>	<b>24</b>
3.1 An Overview of Network Analysis .....	24
3.2 Previous Studies on GIS-T .....	25
3.3 Road Network Data Representation and Modeling .....	28
3.4 Problems with Using Arc-Node Data Model.....	37
<b>Chapter 4 An Evaluation of Motion Planning Methods .....</b>	<b>40</b>
4.1 Definition of Motion Planning.....	40
4.2 Problem Statement and Definitions .....	41
4.3 Big O Notion.....	42
4.4 Construction of Topological Networks for Motion Planning.....	43
4.4.1 Roadmap (Skeleton) Methods .....	43
4.4.2 Cell Decomposition Methods .....	44
4.4.3 Potential Fields Method.....	47
5.1 Prototype Specification.....	50
5.1.1 Data Requirement.....	50
5.2 Conceptual Model.....	54
5.3.1 Local Level – Exact Cell Decomposition Implementation .....	57
5.3.2 Global Level – Triangulation and Generation of a Global Graph.....	61
<b>Chapter 6 Discussion and Conclusion .....</b>	<b>67</b>
6.1 Achievements .....	67
6.2 Discussion of Result .....	68
6.2.1 Limitation of the Algorithm .....	68
6.2.2 Flexibility of Exact Cell Decomposition in GIS Navigation .....	70
6.2.3 Running Time of the Package .....	71



6.3 Limitation of Project.....	72
6.3.1 Data Availability .....	72
6.3.2 Software.....	72
6.4 Conclusion and Future Work.....	72
<b>REFERENCE .....</b>	<b>775</b>
<b>BIBLIOGRAPHY .....</b>	<b>77</b>
<b>APPENDIX – Pseudocode .....</b>	<b>79</b>



## List of Figures

### CHAPTER 1

Figure 1. 1 Generation of road centerlines in ArcInfo. The Complicated junction is flagged (blue) in order to digitize manually later (Arc/Info Help). .....	11
Figure 1. 2 Topographic features: street blocks (yellow), building blocks (green) and bridges (brown) .....	12

### CHAPTER 2

Figure 2. 1 Mathematical definition of a graph.....	18
Figure 2. 2 Incidence matrix of a graph .....	20
Figure 2. 3 Classification of shortest path problems (Evans and Minieka, 1992) .....	20
Figure 2. 4 Dijkstra's algorithm example (Evans, J.R. and Minieka, E., 1992, pp.85) .....	22

### CHAPTER 3

Figure 3.1 Physical and logical description of GIS-T data modeling.....	30
Figure 3.2 Two different modelling techniques for traffic island.....	32
Figure 3.3 Different representations of a roundabout .....	32
Figure 3.4 Single line (left) and multi-lines approach .....	33
Figure 3.5 Single node representation.....	34
Figure 3.6: 4-nodes and 8-nodes representation .....	35
Figure 3.7 Simple network with traffic flow direction .....	36
Figure 3. 8 Database structure of network .....	36
Figure 3. 9 Non-planar network and planar network .....	39

### CHAPTER 4

Figure 4. 1 Example of visibility graph and its network representation .....	44
Figure 4. 2 Decomposing the safe configuration space into horizontal slices.....	45
Figure 4. 3 Forming a graph by using the mid-points of each line segment.....	45
Figure 4. 4 Forming a graph by using the centroids of cells.....	45
Figure 4. 5 Quadtree decomposition .....	47
Figure 4. 6 Potential Fields .....	478

### CHAPTER 5

Figure 5. 1 Street block (green) and building block (yellow) .....	51
Table 5. 1 Table of testing dataset .....	52
Figure 5. 4 The implementation of the prototype .....	56
Figure 5. 5 The generation of discrete connected graphs by using Exact Cell Decomposition .....	61
Figure 5. 6 Triangulation all the nodes of the resulting graph.....	62
Figure 5. 7 Filtering of redundant links .....	62
Figure 5. 8 The completed graph .....	63
Figure 5. 9 Add additional dataset to the graph .....	64
Figure 5. 10 The removal of link segments by erasing with bridges .....	64
Figure 5. 11 Completed graph with bridges added .....	65
Figure 5. 12 Resulting path from source A to destination B.....	65
Figure 5. 13 Generalization of the resulting path.....	66



**CHAPTER 6**

Figure 6. 1 Links created along bridges .....	68
Figure 6. 2 Incorrect links connecting the ground and bridges.....	69
Figure 6. 3 Incomplete graph in concave graph.....	69
Figure 6. 4 Implementation of Exact Cell Decomposition algorithm in Mong Kok	71



Abstract of thesis entitled

**Algorithmic Development of an Optimal Path Computation Model Based on  
Topographic Map Features**

Submitted by Tang Yuet Fong, Mandy

for the master degree of philosophy in Geomatics  
at The Hong Kong Polytechnic University in October 2005

Navigation applications have been at the heart of much GIS network analysis research. Path finding problems have attracted widespread research interests with different applications such as Logistics applications, Infrastructure Planning and Travel Demand analysis. To support such applications, specific data requirements are required. The conventional arc-node model is commonly and widely utilized to model the complexity of network elements in a logical way. The reason is clear. For example, in an arc-node road centerline network model, arcs correspond to segments that are the conduits for transportation and nodes correspond to intersections connecting arcs together. However, a road centerline is not a natural feature that can be found on the ground or map. The generation and maintenance of centerlines are difficult and tedious because these are only imaginary lines. Human judgment and manual digitization are often essential in generating these lines.

To minimize the effort of producing and maintaining a network of additional linear features, an alternative approach is suggested. Path finding method is independent of any arc-node data structure. The network model for such computation is solely based on feature outlines as appeared on the topographic maps. In other words, outlines or symbology of relevant features like road margins, building outlines,





and footbridges will directly be used to model the path finding network. This project therefore aims at investigating the rationale and logistics behind in developing such a model with pedestrian walking as an application example. Horizontal cell decomposition and triangulation algorithm are the main aspect of the study. In the prototype, users are required to input the topographic map features available, specify the semantics of these features, and identify the origin and destination points. Without the need to generate extra lines of arc-node structure, this study will investigate the automatic generation of an optimal path in terms of computation.



## Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor Dr. Lilian Pun for giving me the opportunity to explore my knowledge in PolyU again. I did have an enjoyable experience with the topic she suggested. I also thank for her guidance and constructive comments throughout this research and for reviewing this thesis.

I am deeply grateful to Prof. Christopher Gold, who was my advisor of my guided study. He helped me to understand the graph theory and algorithms and showed me the right track of this thesis.

I warmly thank my two special friends Ela and Krzysiek for providing me encouragement and constant support. I think their presence in Hong Kong were the best thing that could happen to me and my thesis. I am especially obliged to Krzysiek for his valuable and untiring help whenever I was in need. Buzi Buzi to Ela and Krzysiek.

I would have never finished this thesis without the encouragement and help of many friends and colleagues to whom I am very much indebted. I would like to thank Alice, Carol, Ida, Kay, Monchhchi, Moon, Rebecca, Samuel, Shing, Stanley, Tracy, Yeung and XuZhu for being around and sharing several good times together during my stay in PolyU.

Finally, I would like to thank my mom and my family for their support. Without the support of my family, I would have never been here at this point of my life.



## Chapter 1

## Introduction

### **1.1 Background**

With the development of geographical information systems (GIS) technology, network and transportation analyses within a GIS environment have become a common practice in many application areas. For instance, many previous research studies focus on vehicle navigation to find the optimum/shortest path for drivers before or during a journey.

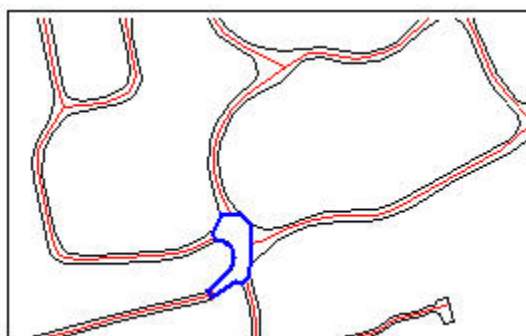
The most important issue of path finding is the algorithm. The most famous algorithm for solving shortest path problem is Dijkstra's algorithm. Nearly all path calculations use some form of this method (Ralston 2002) to perform the same calculations but differ with respect to implementation. Dijkstra's algorithm is a node-based algorithm. It solves the problem of finding the shortest path from a point to a destination with the considerations of the attribute values associated with arcs. Therefore, the network should be in a node-arc structure form to implement this algorithm.

The most common and convenient way to represent a road network is the node-arc representation. Generally, nodes correspond to street intersections while arcs correspond to street segments between intersections. In many applications, network arcs of so called road centerlines are digitized to lie between the centers of physical road margins. In order to support routing application, all road centerlines should be snapped together to form a connected road network. This connectivity property of node-arc representation of road network allows the system to traverse through the network efficiently with the support of network data storage structure.



The problem of arc-node representation will be provided more in details in Chapter three.

However, road centerlines are not natural such that they can be found on the ground. This means traditional data capturing techniques like surveying, photogrammetry and remote sensing are unlikely to capture this kind of data also. Manual digitization from paper maps or photos is the most common way to generate the road centerlines that lie between road margins. Although many GIS software are intended to automate the process of creating the centerline, the result could be unsatisfactory when some complicated cases are met (Figure 1.1). Therefore, the generation of centerlines could be cumbersome and time consuming.



**Figure 1. 1 Generation of road centerlines in ArcInfo. The Complicated junction is flagged (blue) in order to digitize manually later (Arc/Info Help).**

To avoid extra effort needed of centreline generation, the author proposes to use map features directly to perform the path finding analysis. Although no centerlines are not shown on a paper map, the user can determine accessible paths based on both spatial features (e.g. vehicular roads, walking paths, buildings) and their descriptions (e.g. slope, stairs, turning directions) without the existence of road centerlines. Therefore, it should be possible to find a path from base map features



such as footbridges/elevated walkways, footpaths, kerb lines, pavements, steps, subways and road/rail tunnels etc. provided in a digital map (Figure 1.2).



**Figure 1. 2 Topographic features: street blocks (yellow), building blocks (green) and bridges (brown)**

### **1.2 Objectives**

This project investigates the algorithmic development of an optimal path computation model based on topographic map features. In traditional networking GIS application, arc node data structure is the dominant data representation method. However, this arc node data structure could be unfamiliar and unnatural for general users, especially non-GIS people. The generation of this huge dataset is tedious and maintenance of its topology is arduous. The aim of this research project is to develop an algorithm to perform path finding based on topographical map features, so that users need not undertake tedious digitization nor understand the complicated arc node data structure. Users can just base on their understanding of map features, set the rules to these features, input the origin and destination to the designed system. The system will then be able to generate a graph based on the input topographic



based map features and to compute an optimal path indicating the walking direction between the chosen origin and destination.

### **1.3 Scope**

It is understandable that the complexity to compute a path for pedestrians is different from that for a driver. To calculate an optimal/shortest path for a driver, factors like turns, number of lanes, traffic lights and restricted time for certain zones have to be taken into consideration. However, less complexity is required to calculate a walking path for pedestrians. Pedestrians are supposed to walk freely on all walking path features (like pavements, stairs, elevated walkways, subways) and on restricted places of roads (e.g. zebra crossings, traffic light areas). To start researching into this new model, a relatively simple model of walking paths for pedestrians is studied first.

To avoid making this investigation unnecessarily excessive, it is assumed that pedestrians will observe road regulations. The criterion for paths is based on the legality to suggest a safe path for pedestrians. Furthermore, the resulting path would be considered as a “shorter” or an “easier” path rather than the “shortest” path. Unlike a driving path, a walking path includes elevated walkways, steps or subways. The surface distance of steps, for example, will be longer than that displayed on a map. Hence, it is hard to define the term “shortest” in this case. Instead, it would be more meaningful to suggest an easier path for pedestrians.

### **1.4 Significance**

The objective of a transportation system is to improve individual accessibility. However, most of the Geographical Information Systems for navigation are mainly



focused on vehicle navigation, individual accessibility is less common. This research could be important and useful for those people who are interested in personal navigation.

In addition, the findings of this thesis would be a new algorithm which is suitable for performing path finding analysis based on map features. This algorithm could be further investigated, say, with the considerations of modeling the facilities (e.g. stairs, lifts) inside buildings such that the walking path could become more realistic to have both “horizontal” and “vertical” movements of pedestrians. On the other hand, the modeling of facilities/paths for the disabled could facilitate the resulting walking path to exclude obstacles like stairs for them. Hopefully, the idea of this project could be extended to a more complicated scenario to model turns and other attribute data associated with base map features and finally to handle the path finding problem for drivers without using road centerlines in the future.

### ***1.5 Methodology***

The methodology will be divided into three parts – data modelling, algorithm exploring and programming. The first part is to further study the data model of road network and to design a new data model for this study.

The primary spatial data required in this project will be a set of digital data showing land features of buildings, roads, pavements, walkways and transportation infrastructure of a selected study area in Tai Po. These specific features (as shown in the table below) will support or aid the path finding functionality:



Features	Feature Representation
Footbridge/elevated walkway	Polygon/Line
Footpath	Polygon/Line
Road margin line	Line
Pavement line	Line
Island	Polygon/point
Zebra crossing	Polygon/ Line

To start modelling the map features, it is suggested to start from the node-arc representation, which is the most common and familiar data model in path finding analysis. In fact, it does not matter of what geometry these features originally are. To perform path finding, all points, lines or polygons can be converted to the node-arc topology if necessary in the model. The most important requirement is the recognition of different semantic features on the map, their characteristics with regard to ‘walking’ and their spatial relationships with each other.

In a conventional road network, all arcs are well connected by nodes. An optimum path between two nodes can be calculated by traversing the topology. However, there is no such connectivity characteristic between base map features. So the main difficulty of this project is the computation of walking path with disconnected road features. To deal with this particular problem, modification of the Exact Cell Decomposition and Triangulation methods are used to compute walking path based on topographic features such as buildings and street blocks. Furthermore, the proposed algorithm is implemented and verified in a Visual Basic program.

To conduct this research, it is assumed that a large scale base map should be used to achieve a high level of detail. Among different topographic map features, this project focuses on finding walking paths based on map features. However,





movement and facilities such as lifts, elevators and stairs inside buildings will not be taken into account. To make the investigation more feasible, mapped features and data from the Survey and Mapping Office, the Transport Department and the Buildings Department will all be considered.

### **1.6 Thesis Organization**

This thesis is arranged into six chapters beginning with an introductory chapter. Chapter one outlines the background information of this topic and seeks to show the significance of the idea of optimum path finding for pedestrians. The following three chapters then review some literature of relevant materials of this project. Chapter two covers the fundamental graph theory and classical shortest path algorithm. The relationship between graph theory and GIS network analysis application can be found in Chapter three. It also presents the current approach and problem of path finding in GIS. Chapter four, the last chapter of literature review introduces some algorithms in motion planning problem. Advantages and disadvantages of each algorithm will be covered here.

The implementation of the prototype is discussed in Chapter five. The descriptions of the algorithm are presented in two parts: local and global levels. At the local level, an exact cell decomposition algorithm is used to generate graphs within each street blocks whereas at the global level, triangulation is used to have a complete graph for routing. The last chapter, Chapter six, concludes the current status of this project and proposes a future work schedule.



## Chapter 2 A Review of Graph Theories

For transport network, a clear arc and node definition is important. As Rodrigue (2002) stated, some transport networks like maritime and air networks may place less importance on arcs representation because they are not normally defined clearly. Other network models such as those for telecommunication, electrical or pipe networks would define arcs and nodes differently. In general, graph theories can be used to represent all network features and a review of the graph theory is noteworthy. With the understanding of graph theory, we will move further to the classical shortest path problem and algorithms which are built on graphs/networks. To enhance the readability of this thesis, some related terminologies and concepts will be highlighted and explained.

### 2.1 Fundamental Theories

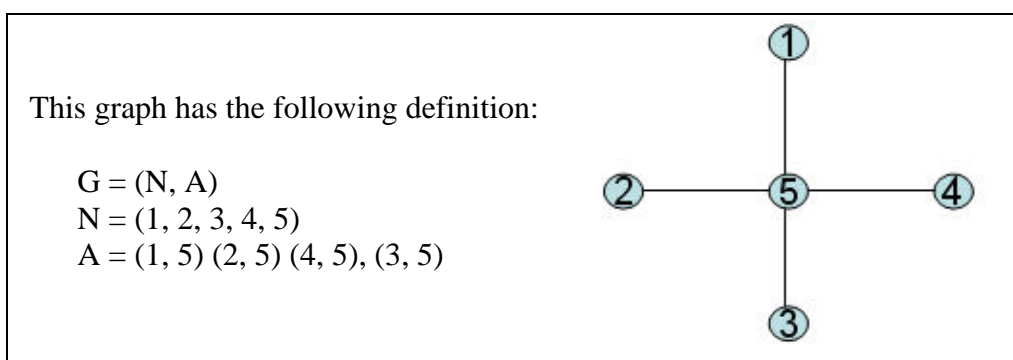
#### 2.1.1 Basic Graph Definition

The famous “Königsberg Bridges” problem is generally viewed as the origin of graph theories. The city of Königsberg (now Kaliningrad) in East Prussia was divided into four parts by the Pregel River, and these four parts were joined by seven bridges. Citizens of Königsberg wanted to know whether it would be possible to take a walk that crossed each of the seven bridges exactly once. , the famous Swiss mathematician, built a representative graph and proved that such a walk was impossible. In 1736, his solutions formed the basis of the graph theory and paved the way for the development of path finding algorithms.

A formal definition of a graph quoted from Martin (1979) is stated as below (Figure 2.1): “A graph  $G = (N, A)$  consists of (i) a finite set  $N = \{n_1, n_2, \dots, n_n\}$ ,



whose elements are called nodes, and (ii) a subset  $A$  of the Cartesian product  $A \times A$ , the elements of which are called arcs.” A similar up-to-date definition of a graph can be seen from Carter and Price (2001): “A graph is a structure consisting of a set of nodes (vertices, points, or junctions) and a set of connections called arcs (edges, links, or branches).” It is noted that the terms node, vertex, point or junction can be used interchangeably in graph theory terminology. It is also the same for the terms arc, edge or link. In this chapter, the terms **arc** and **node** will be used when referring to a graph.



**Figure 2. 1 Mathematical definition of a graph**

However, more than a graph, a complicated structure called a network is required to perform analysis in diverse problems in different areas application like transportation, telecommunications, electrical and operations research. In fact, this complicated structure can be thought of as a graph but the values associated with arcs or nodes in the graph might represent distances, costs or other possible parameters. Hence, according to Carter and Price (2001), a network can be defined as “a directed connected graph that is used to represent or model a system or a process”. A graph (digraph) is said to be directed if the orientation or direction of an arc is specified, otherwise it is undirected. As can be seen, all the graphs/networks referred to in this thesis are directed.



## 2.2 Data Structure for Graphs and Networks

### 2.2.1 Topology

For a graph consisting of a set of arcs and nodes, the relationship between the arcs and their respective nodes is referred to as the network topology. This generally can be represented by a rectangular array of numbers called matrix stating the presence or absence of a relationship between network elements and other variables (Bell and Iida, 1997). Here the simplest and more frequently used matrix – the *adjacency matrix* and the *incidence matrix* will be described. For adjacency matrix, it involves the adjacency of vertices, whereas the incidence matrix involves the incidence of vertices and edges.

Referring to Evans and Minieka (1992), the definition of an adjacency matrix  $A$  of a directed graph without loop, with  $m$  vertices labeled  $1, 2, \dots, m$  and  $n$  edges labeled  $a, b, c, \dots, n$  is a  $m \times m$  matrix in which  $a_{ij}=1$  if there exists an arc  $(i, j)$  from node  $i$  to node  $j$ , and 0 otherwise. On the other hand, the incidence matrix  $N$  will be a  $m \times n$  matrix (Figure 2.2) in which

$N_{ij} = + 1$ , the node associated with row is the tail of the arc associated with column  
 $= - 1$ , the node associated with row is the head of the arc associated with column  
 $= 0$ , otherwise

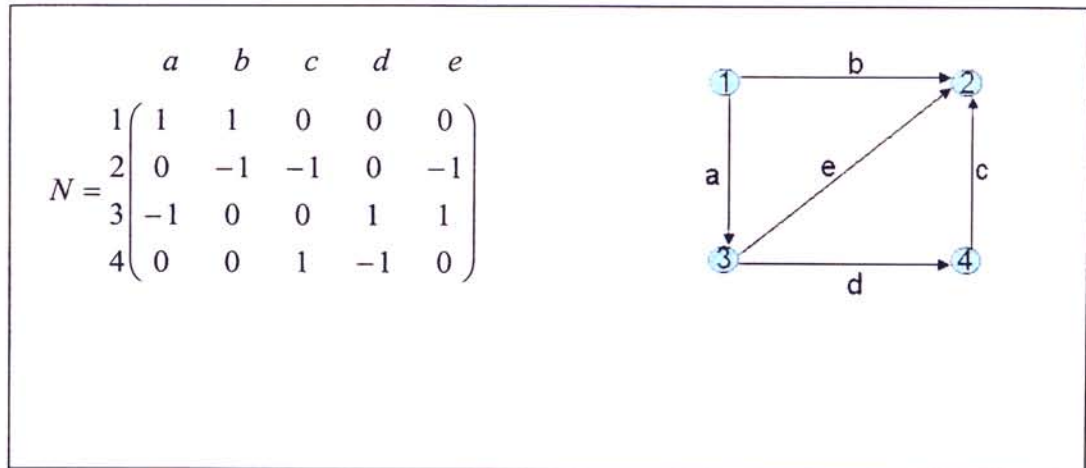


Figure 2. 2 Incidence matrix of a graph

### 2.2.2 Shortest Path Problems and Algorithms

The classical problems of finding shortest path have been extensively studied among different kinds of network optimization problems. Evans and Minieka (1992) classified shortest path problems into the following different types (Figure 2.3):

- I. Ordinary Path Lengths
  - A. Unconstrained
    - 1. Shortest path
      - a. Shortest path between 2 specific nodes
      - b. Shortest path from one node to all others
      - c. Shortest path between all nodes
    - 2. k-shortest path (1<sup>st</sup> shortest, 2<sup>nd</sup> shortest and so on)
  - B. Constrained
    - 1. Shortest path that includes specified nodes
    - 2. Shortest path that includes a specified number of arcs
- II. Generalized Path Lengths
  - A. Turn Penalties
  - B. Length as a function of the path
  - C. Algebraic related problems

Figure 2. 3 Classification of shortest path problems (Evans and Minieka, 1992)

The difference between categories I and II is that Ordinary Path Lengths refer to the sum of the lengths of individual arcs whereas those path lengths with more complicated functions of arc lengths such as turn penalties are categorized in Generalized Path Lengths. The problem of finding optimum path for pedestrians will



be classified as an unconstrained problem which concerns the physical distance of path, feasibility and legality between locations.

To determine the shortest path in a network, an efficient algorithm is needed. Over the years, many significant shortest path algorithms have been developed. One of the most common algorithms called Dijkstra's algorithm which can be found in most GIS software will be discussed.

### **Dijkstra's Algorithm**

One of the important algorithms in solving the shortest path problem is Dijkstra's algorithm, first published by E.W. Dijkstra in 1959. This is a label setting algorithm to find the shortest path from a single source node(s) to all other nodes in a network. The following example illustrates that label setting algorithm is an iterative processing. At each iteration, it assigns tentative labels to those nodes adjacent to a particular permanently labeled node and computes the shortest distances of these tentative labeled nodes. At the end of each iteration, the algorithm set one tentative label to permanent and the iteration continues until the shortest path to the destination node is found. Permanent labels here represent the actual shortest path distances in the optimal solution. The implementation of Dijkstra's algorithm (Figure 2.5) is described below with an example (Figure 2.4).

#### **Step 1 Initialization**

For a directed graph  $G = (N, A)$ , let node  $s$  and node  $t$  be the source and destination nodes respectively. Initially, label node  $s$  as permanent. Assign  $d(x)$  to each node  $x$  to denote the tentative distance between  $s$  and  $x$ , where  $d(s) = 0$  and  $d(x) = \infty$  for all  $x \neq s$ .



### Step 2 Tentative labeling

For each unlabeled nodes  $x$  in the forward star of  $s$ , re-compute  $d(x)$  as follows:

$d(x) = \min\{d(x), d(s)+a(s,x)\}$ , where  $a(s,x)$  denotes the arc incident with the nodes  $s$  and  $x$ .

### Step 3 Permanent labeling

Label the node with the minimum value of  $d(x)$  and its corresponding arc.

### Step 4 Termination

Terminate the algorithm if the destination  $t$  has been permanently labeled. Otherwise, repeat steps 2 and 3.

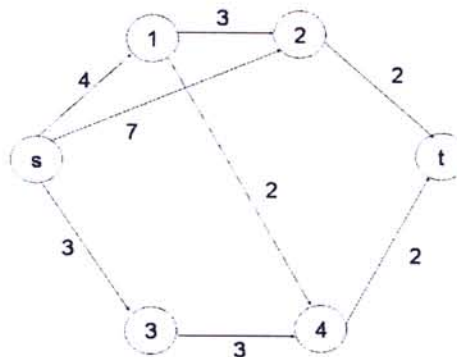
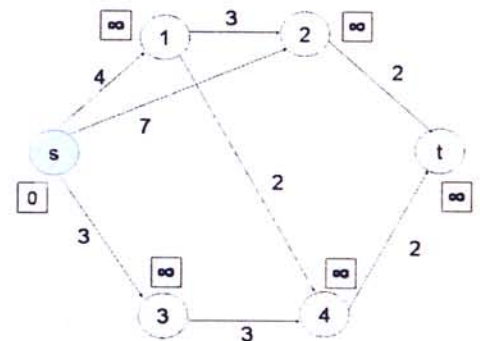
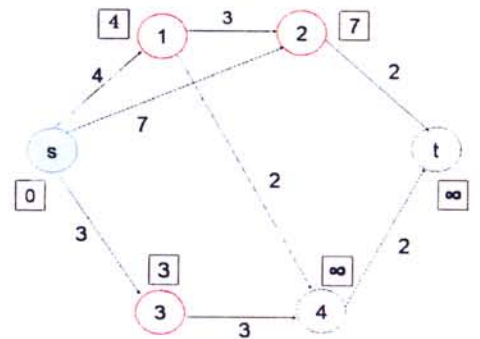


Figure 2. 4 Dijkstra's algorithm example (Evans, J.R. and Minieka, E., 1992, pp.85)

Label the source node  $s$  and assign tentative distance equal to infinity for all the vertices in the graph



Calculate the tentative distances for the unlabeled nodes which are incident to node  $s$ . (Highlight as red circles)





Since the minimum distance on any unlabeled node is node 3, therefore node 3 and arc(s, 3) are labeled.

Re-calculate the tentative distances for all the unlabeled nodes which are incident to node 3.

And since the minimum distance on any unlabeled node is node 1, therefore node 1 and arc (s, 1) are labeled.

Re-calculate the tentative distances for all the unlabeled nodes which are incident to node 1.

And since the minimum distance on any unlabeled node is node 4, therefore node 4 and arc (3, 4) are labeled.

Re-calculate the tentative distances for all the unlabeled nodes which are incident to node 4.

And since the minimum distance on any unlabeled node is node 2, therefore node 2 and arc (s, 2) are labeled.

Re-calculate the tentative distances for all the unlabeled nodes which are incident to node 2.

Thus, node t has been labeled at last. Also, arc (4, t) is labeled.

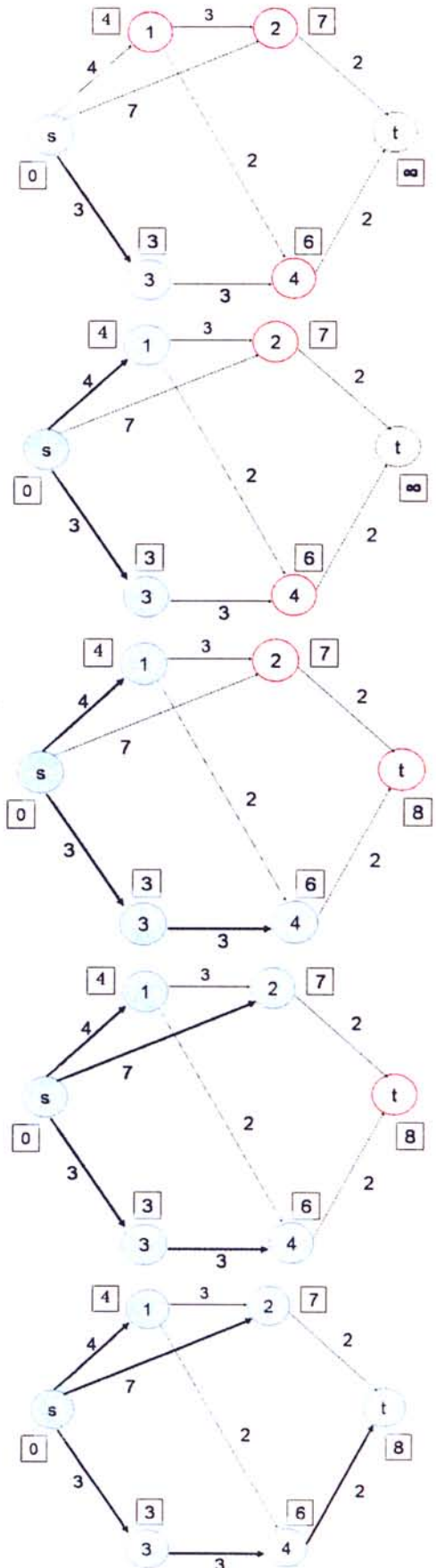


Figure 2.5: The execution of Dijkstra's algorithm





## **Chapter 3            Network Analyse in GIS**

The previous chapter explained the fundamental concepts of graph theory, the shortest path problem and its algorithm. In this chapter, we will focus on the application of graph theory in network analysis in GIS-T. It commences with a brief overview of network analysis in GIS, followed by a detailed description of the shortest path problem in GIS. The concurrent data modeling technique and its corresponding problems will be covered in this chapter as well.

### ***3.1 An Overview of Network Analysis***

As can be seen from the previous chapter, the graph theory developed a significant topological and mathematical presentation of the structure and nature of networks. However, some other tools are needed to support the use of this graph theory and its algorithm in a real-world application. To find the shortest path of a transportation network, the algorithm cannot be worked out unless the network is modeled and stored digitally in an information system. Clearly, a Geographical Information System (GIS) has the inherent advantage of storing, updating, manipulating, retrieving, visualizing and analyzing of different kinds of spatial data models. With the integration of transportation data in GIS, it provides an opportunity to analyze the data more comprehensively. Indeed, GIS-T or the application of GIS in transportation was considered one of the most rapid growing areas in GIS in the 1960s (Goodchild, 2000).

According to Lo and Yeung (2002), a network in GIS is defined as “a set of linear features that are interconnected”. Common examples of networks include highways, railways, city streets, rivers, transportation routes (e.g. transit, school



buses, garbage collection and mail delivery), and utility facilities (e.g. electricity, telephone, water supply, and sewage). Regarding different aspects and application context, network analysis techniques are rather well developed in today's GIS to handle complex analyse of address geocoding, closest facility, resource allocation and shortest path. However, it is not the scope of this thesis to discuss all these in detail except the shortest path problem.

### **3.2 Previous Studies on GIS-T**

Many reviews on the shortest path problem have been published since the end of 1950s (Han et al., 2001). An excellent overview of GIS-T was conducted by Goodchild (2000). His paper characterized GIS-T in 3 perspectives: a map view, navigation view and behavioral view (the study of behavior). With the understanding of the development of GIS-T data and applications in these three different views, the difficulties of adapting generic GIS approaches to the specific needs of GIS-T were highlighted. The author finally provided a list of six new research challenges ranging from the technical issues of unambiguous communication of location to the economic modeling of the creation and use of GIS-T information.

The shortest path algorithm is the key component in the shortest path problem. Reviews or evaluations of shortest path algorithm have been the subject of extensive research for many years (Vuren et al. 1987, Zhan and Noon 1998)

The merging of GIS and advanced technologies has yielded better data quality and more efficient algorithms. Another way to improve the overall analytical efficiency is through refinement of the data structure. This was illustrated by



Horowitz (1996). He explored the integration of GIS concept into transportation network data structure. This paper highlighted the advantages of using object oriented technology in tackling the burden transportation data structure. With both theoretical and practical considerations, the author derived a network structure composed of objects to deal with sophisticated network data models.

Sutton (1996) discussed the problems of integrating transportation data in GIS. The problems were presented in two categories: data attribution and network presentation issues. To enhance the data quality in terms of positional accuracy and attribute accuracy, a generalization process called conflation was suggested to integrate transportation data associated with networks. The author explored the advantages of data integration using a case study in Southern California by conflating the street centerlines with the transportation model networks.

Kirkby et al. (1999) had modified Dijkstra's shortest path algorithm and successfully implemented it in a 3D GIS environment. The aim of this study is to provide an interactive interface for the users to determine emergency service response time between locations within Okayama in Japan. To do this, a modified algorithm consisting of 2 modules to perform calculation and display functions was implemented into a 3D GIS. Factors affecting vehicle travel time such as travel speed, gradient and turning times were used to calculate the paths.

The above reviews included a classical problem to determine the path between a given origin and destination in a connected road network. Chen and Yang (1999) illustrated some other "non-classical" problems to calculate the shortest path



with additional constraints like traffic-lights. The paper simulated the operations of traffic-light control encountered at intersections of roads. A technique called “time windows” was used in this study. The scenario of cars waiting for traffic-lights to turn green was simulated in a specified time window. For each time window, routes were restricted such that only specified routes were allowed to pass through the junction. In addition, the authors developed a polynomial algorithm with a sequence of time windows for finding the optimum path in a traffic-light network.

In another research, Han et al. (2001) developed an optimizing algorithm path finding in vehicle navigation with the considerations of turn penalties and prohibitions. The authors represented the turn prohibition problem in a traffic network and analyzed key issues in the classical path finding algorithm such as network topology, data structure and the algorithm itself. Since the traditional shortest path algorithm was not designed to handle costs or prohibition associated with intersections in a road network, the traditional Dijkstra’s algorithm is modified based on a new proposed forward star data structure to solve the shortest path problem with turn limitations. The whole idea was tested on a large-scale Hong Kong road network data with 5351 nodes, 7117 arcs and 43575 turns.

It is not difficult to notice that transportation applications of GIS have been the target of many research efforts in recent years. Empirical studies have been reported on the findings of shortest paths for drivers over a network formed by road centerlines. However, it is perhaps surprising that there is but a dearth of research into finding the shortest paths for pedestrians.



### ***3.3 Road Network Data Representation and Modeling***

GIS-T studies can be divided into three categories: data representation, modeling and applications (Miller and Shaw, 2001). In this section, a detail discussion of data representation and the modeling of GIS-T will be provided.

#### **GIS-T Data Representation**

Compared with paper maps, GIS provides users a powerful tool to store, present and analyze geographical data digitally. In tradition, spatial data are presented on paper maps with the use of lines, symbols and color. In GIS, spatial information are geometrically stored in two ways: vector data in the forms of points, lines and polygons or raster data in the forms of cells. In this section, network modeling in these two data representations will be discussed in more detail.

Traditionally, vector representation has been the domain of network analysis in GIS. The transport network model is defined as a “line” graph which comprises a collection of nodes and arcs to form a continuous and connected network. The two primitives, nodes and arcs represent the junctions and segments of a road respectively. The links of the network are normally digitized to lie between the road margins and that is the reason the generic term “road centerlines” is introduced. Due to the way of representation and connectivity property of the vector network, complexities such as turns, restrictions, and lane information can be incorporated in the model easily. Also, its flexibility property enables a transportation system to be modeled as a planar or non-planar graph. This implies that overpasses can be considered as links intersecting with the underpasses in a planar vector model whereas in a non-planar network, they just cross each other and no extra nodes will be created at the intersections.



However, raster network modeling adopts a completely different approach to the road centerlines model. In the raster mode, roads are generalized in terms of regular cells in the network. Furthermore, traffic directions cannot be explicitly derived from the arc direction. The costs or impedances are normally encoded in separate layers. Therefore, network analysis in the raster model is generally performed across a vast number of layers. According to Husdal (2000), a raster based network model is more suitable to be used when the path finding application is carried out across a terrain surface.

### **GIS-T Data Modeling**

The term “data modeling” is defined as a process of interpreting reality by using both a real-world and a data model (Bernhardsen Tor, 2002). However, the perception of the real world is user-oriented. Different interpretations of the real world result from subjective perceptions of users although the real world is objective. It is clear that transportation data are the most complicated due to its complex physical structure and relationship with other geographical data like buildings. Thus, it is important to understand the multifaceted nature of GIS-T data before translating them into digital formats. According to Miller and Shaw (2001), transportation entity has physical descriptions and logical relationships with other transportation entities (Figure 3.1).



	Logical	Physical
Real	Legal Definitions - Route - State Trunk Network - Country Trunk Network - Street Network - Political Boundary	Actual facilities -Highways -Roads -Interchanges -Intersections
Virtual	Data Structures -Networks -Chains -Links -Nodes -Lattices	Data Values -Lines -Points -Polylines -Polygons -Attributes

**Figure 3.1 Physical and logical description of GIS-T data modeling**

As shown in Figure 3.1, the real/physical mode corresponds to the real world transportation facilities such as highways and interchanges whereas real/logical mode corresponds to the legal definition of transportation entities in the real world. One-to-many relationship can be derived between these two models. From a virtual world or database view, virtual/logical and virtual/physical models resemble the data structure and values respectively. Similarly, there are often one-to-many relationships between these two entities. For example, at a small scale map, a single line with modal-specific flow will be used instead of using two directed arcs in presenting a two-way road.

To represent the real-to-virtual relationship of a road network in GIS, a well known data model called arc-node model is used. In the arc-node model, road segments are abstracted as centerlines lying between the center of road margins and interchanges or intersections are modeled as nodes. Road centerlines may be directed or undirected. For directed ones, the traffic flow of a particular road segment is indicated by ordering its two-ended nodes. In general, a directed network is used to represent the traffic flow of a transportation system. As discussed in Chapter two, the



difference between a graph and a network is that a network can incorporate weights associated with each arc. Take a transportation system as an example, “weights” can be included in turns at junction, road width, speed limit, road class, registrations, number of lanes etc. A more detailed discussion of the data schema will be provided in the next section. It is understood that nodes in a road network represent interchanges or intersections whereas lines (centerlines) represent the physical road segments between the road margins. However, different definitions of roads, quality requirements or criteria would transform the real road network into different portrayals. Hence, different data modeling techniques are discussed as below. To clarify the differences, the discussions are divided into two main parts: linear data modeling and topological data modeling.

### **Network Linear Data Model Design**

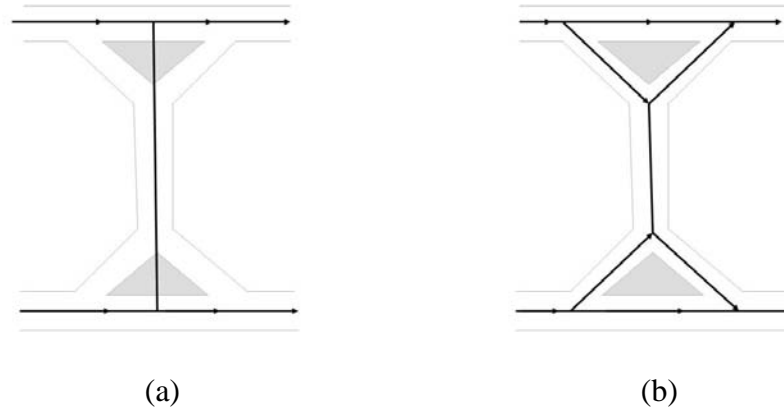
As the name implies, network linear data model design focuses on the linear modeling part of the transportation network. Factors that affect the way of linear feature modeling include the road geometry, number of lanes and the traffic flow direction.

In general, road segments are abstracted as single road centerlines connecting two end nodes. These single road centerlines are digitized at the center between the road margins representing one way or bi-directional road segments. Without a doubt, this approach is simple and does not capture critical geometry property. However, this approach not only reduces the positional accuracy but also introduces unrealistic display representation. In Figure 3.2 (a), the road centerline is digitized across the traffic island. For navigation application, this representation is definitely



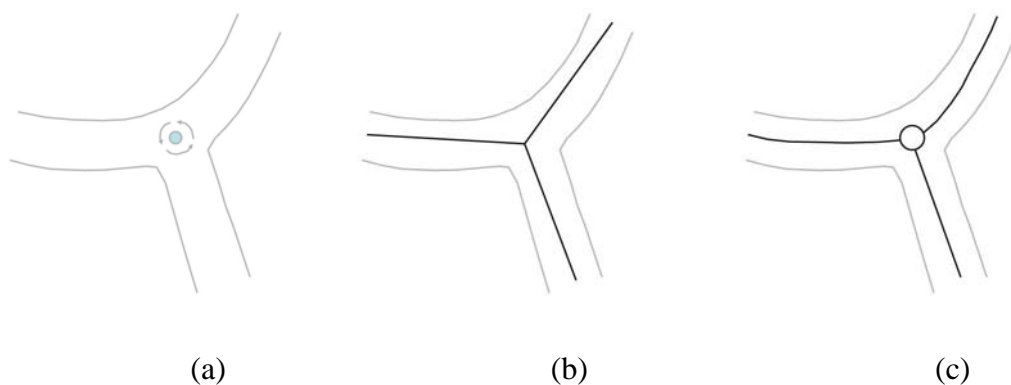


unacceptable. Misleading vehicle guidance may result because this traffic island may have major influence on the traffic flow (depending on the size of the traffic island). Therefore, the centerline should be split so as to indicate the left and right turns at the ends (Figure 3.2 (b)). This comes to another worthy topic here – the geometric issue.



**Figure 3.2 Two different modelling techniques for traffic island**

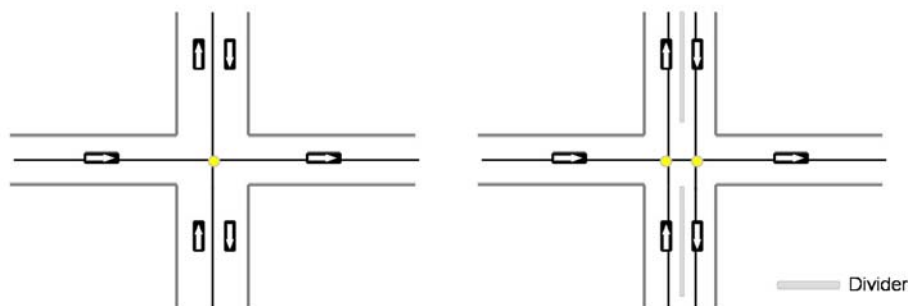
In fact, geometric problem is another critical theme in data modeling. Some complicated transportation infrastructures like roundabouts, traffic islands, slip roads, barriers or ramps etc. may introduce different modeling approaches. In case of a roundabout, there are two possible ways to model the following scenario (Figure 3.3 (a)).



**Figure 3.3 Different representations of a roundabout**



The simplest way is to model the roundabout is to aggregate it as a point connecting three arcs as shown in Figure 3.3(b). The second approach is to present the real traffic flow of the roundabout (Figure 3.3(c)). This indicates that it may be so large as to have an important impact on the traffic flow. Drivers need to make a right turn to enter the roundabout, travel a short distance on the circular track, and re-enter one of the roads again. As has been noted, one single line can be used to represent one way or bi-directional information, but in some circumstances like the existence of barriers, the single line is forced to split into two single lines (multi-line approach) to represent two different travel directions (Figure 3.4). Geometry is a key factor to determine the graphical representation, but the choices between modeling methods are also driven by other factors such as map scale, applications, accuracy requirement and analytical objectives.



**Figure 3.4 Single line (left) and multi-lines approach**

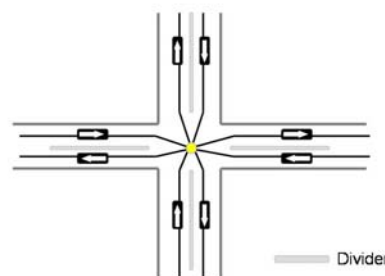
The traditional arc-node model can be enhanced by adding lane information on each road element. In conventional arc-node model, the relationship between lines and lanes are often one-to-one or one-to-many relationship. This implies that the relationship between lines and directions are one-to-one or one-to-many relationship also. But in lane-based road network model, arcs depict the number of lanes per road part. Clearly, each arc represents a single flow traffic direction only. For a modern



intelligent transportation system, lane information is important to provide information such as turn directions, instructions for avoidance of obstructions or restrictions in particular lanes, monitoring lane-specific flow for analysis and representing beginning and ending points for lanes for enhanced route guidance (Miller and Shaw 2001).

### Network Topological Data Model Design

Similarly, nodes correspond to road intersection can also be represented at different levels of resolution. As can be seen in Figure 3.5, the intersection is aggregated to a node without representing critical intersection property and turn impedances. To restrict some of the turns at particular intersection, additional tables are needed to store the constraints. Although this method is simple, it is not navigable due to its low positional accuracy.

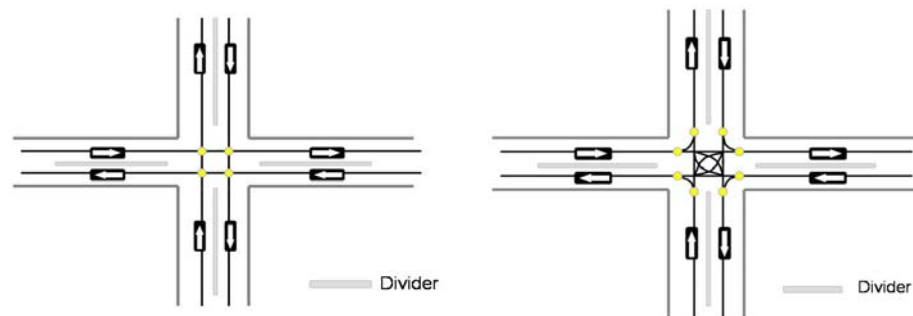


**Figure 3.5 Single node representation**

To capture the turn properties, an expanded model can be used. In Figure 3.6, the aggregated intersection is now expanded into 4 nodes connecting all extended road segments. Although this expanded structure also grows with the database with the increasing number of arcs and nodes, it provides a fairly good positional accuracy for navigation purpose. It is noted that additional tables are still required to store turn impedances. Another sophisticated 8-node data model contributes the highest level of resolution (Figure 3.6). Based on the 4 node network model, some transit links are



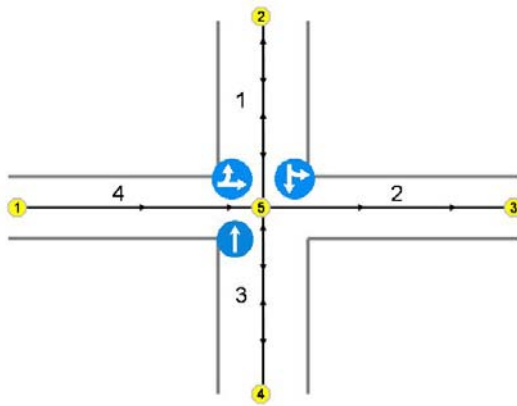
added. Since the turn restrictions are stored directly within arcs, the need of turntable is eliminated. However, this high level representation is time consuming to digitize and maintain.



**Figure 3.6: 4-nodes (left) and 8-nodes (right) representation**

### The Relational Databases

The same as other GIS applications, attribute data of transportation network are normally stored in relational databases. Figure 3.7 provides a simple network with traffic flow directions and turn information and the structure of its corresponding arc and node tables are shown. The general complexities of transportation system such as speed, road class and direction of traffic flow can be stored in an arc attribute table relating to each arc. However, ancillary relations such as turn information of U-turns, prohibited turns and restricted turns cannot be easily manipulated. An existing solution of incorporating this complexity is the use of turn tables. The turn table is normally stored as a separate table for storing intersection turning movements and constraints. When performing shortest path analysis in GIS, the optimum path can be derived implicitly by accessing this turntable.



**Figure 3.7 Simple network with traffic flow direction**

In Figure 3.8, the attributes of turns at node 5 are stored in a turn table with four U-turns (shaded in green) plus three directions from each edge to other edges. A total of 16 (the square of the number of arcs connecting at a particular node) possible turns are generated. A turn table can be used to assign impedance values to indicate different situations. For instance, a zero or a negative value is used to indicate that the turn is prohibited whereas 1 signifies that the turn is allowed. In this example, going straight from arc 3 to arc 1 is allowed while a left turn onto arc 4 and right turn onto arc 2 are prohibited.

**Arc Table**

ID	From Node	To Node	Speed	....
1	2	5	60	
2	5	3	60	
3	5	4	60	
4	1	5	60	

**Node Table**

ID	....
1	
2	
3	
4	
5	

**Turn Table -- Node 5**

From Arc	To Arc	Impedance
1	1	0
1	2	1
1	3	1
1	4	0
2	2	0
2	1	0
2	3	0
2	4	0
3	3	0
3	1	1
3	2	0
3	4	0
4	4	0
4	1	1
4	2	1
4	3	0

**Figure 3. 8 Database structure of network (include arc, node attributes and turntable)**



### **3.4 Problems with Using Arc-Node Data Model**

The conventional arc-node model is common and widely utilized to model the complexity of network elements in a logical way. The reason is clear. In an arc-node road network model, arcs correspond to road segments that are the conduits for transportation and nodes correspond to road intersections connecting arcs together. This means a **connected network** is consequently constructed to depict the complexities of a transport system. Since the connectivity relationship of arcs and nodes and turn restrictions are implemented in attributes tables and turn tables respectively, the optimum path between a source and a destination can be derived easily by traversing the topology of the road network, just like performing a “search” operation in a branching tree with a parent root (the source) and corresponding child nodes (possibilities of destinations).

However, as stated at the beginning of the thesis, the road centerline is not a natural feature that can be found on ground or map. Therefore, generating road centerlines or developing the node-arc data model for network analysis is both important and difficult. It is important because a road centerline seems to be an essential element of a transportation network as discussed above. It is difficult because it involves tedious digitizing work although centerline mapping technologies are evolving rapidly, from traditional map digitizing to GPS, photogrammetry and remote sensing, each of these technologies is associated with a performance range in terms of accuracy or resolution.

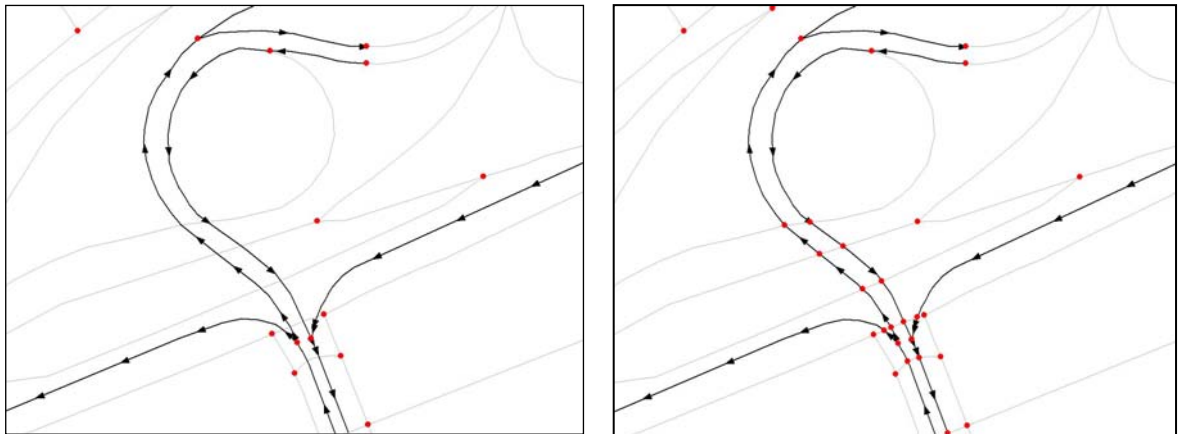
As noted previously, there are many ways to model a transport system. Different definitions of roads, quality requirements or criteria make it difficult to have a consistent representation of a transportation system. According to Dueker and



Butler (2000), there are several GIS data models used for transportation applications. Prominent examples include Geographic Data File Standard (GDF), National Cooperative Highway Research Program 20-27 (NCHRP) and Dueker and Butler's enterprise GIS-T data model. However, these kinds of specifications do not lead to consistency due to their differing definitions and criteria.

Besides, the maintenance of road centerlines is hard. It is because the geometry and position of centerlines are determined by the physical geometry of roads shown on a base map. Changes of physical geometry imply changes of road centerline geometry and its related non-spatial attribute data. These may include node and arc identifiers, impedances and turns. The existing arc-node representation of road network results in a huge volume of data model. As mentioned in the last section, a total of 16 possible turns are generated for a simple junction connecting 4 arcs. Even for a small territory like Hong Kong, its planar network consists of about 40,000 arcs and 30,000 nodes, with as many as 200,000 turn possibilities. Clearly, a substantial amount of time is required for data input, preparation and validation in order to maintain a good quality road network and its associated data model.

Another problem with the arc-node model is the enforcement of node at every intersection in a planar network. In Figure 3.9, additional nodes not only generate more turn possibilities at each junction, but also create more arcs and nodes in the network.



**Figure 3. 9 Non-planar network (left) and planar network (right)**

In addition, the planarity of network does not represent well the real world properties of transportation networks that contain features such as “underpass” and “overpass” (Miller and Shaw 2001). Placing an additional node at under- or overpass introduces a possibility of turning off the highway which does not make any sense in the real situation. These unrealistic turns can be addressed by assigning infinity impedance in turntable. However, this approach is an inefficient means of working around a problem.





## Chapter 4 An Evaluation of Motion Planning Methods

Without the use of road centerlines, a new idea for finding paths for pedestrians will be totally independent of the traditional arc node data model or algorithm. As discussed in the previous chapter, all path finding algorithms developed are based on connected graphs. To apply the existing path finding algorithms, the construction of a graph of a simple polygon (walking area) with obstacles (discrete building blocks) becomes the initial consideration of this study. In this chapter, the solutions of the motion planning problem will be provided.

### 4.1 Definition of Motion Planning

Motion planning is a specific problem of finding a motion for a mobile robot to move from a given starting position to a given destination in a “maze” – an environment or workspace which contains a set of obstacles. In this “maze”, the robot is programmed to walk through it and not collide with any of the obstacles. According to Ahrikencheikh (1994), motion planning problem is stated as follows:

*“Given a system of objects, a start and a goal configurations for these objects, and a set of obstacles which can be stationary or moving with known pattern, the objective is to find the best feasible trajectory for the system of objects to move from the start configuration to the goal configuration such that collision with the obstacles is avoided”*

In this study, we assume that the workspace is static, that is, the obstacles do not move. Therefore, we will focus on the configuration of the workspace with stationary obstacles and basic motion planning approaches. The hardware and software components of a mobile robot will not be discussed here.



#### **4.2 Problem Statement and Definitions**

Suppose that a robot  $A$  is a point that can be translated into the two-dimensional Euclidean space  $\mathbb{R}^2$ . A configuration  $q$  of  $A$  is a specification of the position of  $A$  denoted by  $(x, y) \in \mathbb{R}^2$ . With provided geometric description and a configuration  $(x, y)$ , the position of a robot can be defined exactly in a plane. The configuration space is the space of all the possible configurations of the robot. A *free space* is that subset of the configuration space in which the robot does not intersect any obstacles.

The motion planning problem is to create a data structure, called a map, which contains a connectivity graph and information about the workspace. Given an initial configuration and goal configuration, a route can be generated that the robot can follow to avoid any collisions with obstacles.

In the static domain, motion planning problems are usually solved in the following two steps:

1. Build a graph which is not occupied by any of the obstacles to represent the geometric structure of an environment.
2. Perform a graph search to find a component connecting the source and destination nodes.

Some basic approaches (Step 1) to deal with stationary obstacles in an environment will be described in the next section whereas reference for the graph search techniques (Step 2) can be found in Chapter 2.



### 4.3 Big O Notion

Before a detailed discussion of different algorithms, it is important to understand how complexity relates to the size of the problem itself. To sort a thousand records, an algorithm might need 2 seconds and 10 seconds to sort a million. Another algorithm might need only 3 seconds to sort a thousand numbers and 5 seconds to sort a million. In this situation, it is hard to declare one algorithm is better than the other. Instead, it depends on which kinds of data one was sorting.

The big O notation is useful when analyzing algorithms for their efficiency. For example, the running time of a program (or the number of steps it takes) to complete a problem of size  $n$  might be  $f(n) = 5n^2 - 10n + 19$ . As  $n$  grows larger, the  $n^2$  term will come to dominate, so that all other terms can be neglected. Furthermore, the coefficients will depend on the precise detail of the implementation and the hardware it runs on, so they should also be neglected. The big O notation captures what remains:  $f(n) \in O(n^2)$  and says that the algorithm has an order of  $n^2$  time complexity.

In increasing order of time there are:  $O(c)$  (where  $c$  is a constant),  $O(\log N)$ ,  $O(N)$ ,  $O(N \log N)$ ,  $O(N^2)$  followed by other polynomial times such as  $O(N^3)$  then you start getting into very slow algorithms such as the travelling salesman problem which takes  $O(N!)$  time. To determine which algorithm to adopt in this study, the algorithm performance will be taken into consideration.



#### **4.4 Construction of Topological Networks for Motion Planning**

Prior knowledge of the shapes of obstacles is required for constructing network representations for motion planning. The general idea consists of subdividing the free space into simple “cells” and representing each cell by a node of a network. A connected network is then constructed by connecting adjacent nodes in space.

There are many algorithms to solve the motion planning problem. In path planning, typical methods for creating a collision-free graph between a given origin and destination points cluttered with polygonal obstacles include roadmap methods, cell decomposition, and potential fields methods. The two feasible solutions of this study, roadmap and cell decomposition will be discussed in the following sections.

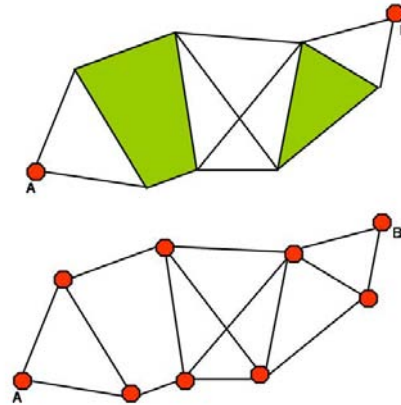
##### **4.4.1 Roadmap (Skeleton) Methods**

The roadmap approach is dependent upon the concepts of configuration space. This approach to path generation consists of reducing sets of feasible motions onto a network of one-dimensional curves, lying in the free space and representing on a roadmap. The cells are the critical vertices of obstacles. The network is formed by connecting each cell of an obstacle to each of another cell without entering the interior of any obstacles.

There are various possible definitions of a roadmap. The most common roadmap methods are the visibility graph, Voronoi diagram, silhouette, subgoal and freeway net. The earliest path planning methods was the visible graph method which was explored in 1969 (Latombe, 1991). The construction of a non-planar visibility



graph for a given polygonal environment has been a subject structure in planning path. With all the critical vertices of obstacles plus the given start and goal points, an edge results which consists of line segments connecting two vertices without passing through the interior of any of the obstacles (Figure 4.1).

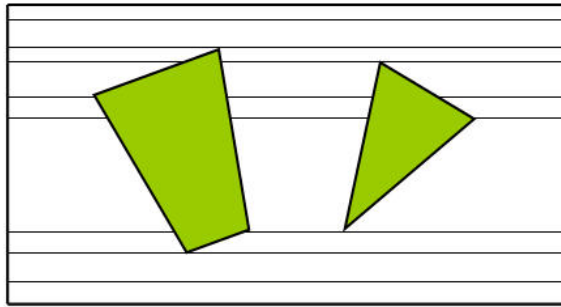


**Figure 4. 1 Example of visibility graph (Above) and its network representation**

The advantage of these roadmap methods is easy to understand. However, in view that the visibility graph is not planar, the resulting graph may consist of  $\Omega(n^2)$  edges, implying that it takes  $O(n^2)$  for constructing the visibility graph. Hence, one of the disadvantages of the roadmap methods is the brute generation of paths.

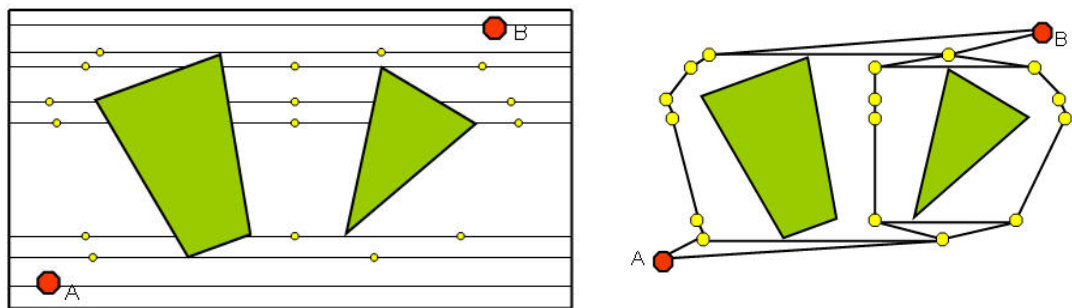
#### 4.4.2 Cell Decomposition Methods

The most extensively studied motion planning methods are the cell decomposition methods. They consist of decomposing the free space into non-overlapping cells (Figure 4.2), such that a connectivity graph can be easily generated by connecting paths between any two adjacent cells. Nodes of the graph are the cells extracted from the free space and two nodes are linked if and only if the two corresponding cells are adjacent.

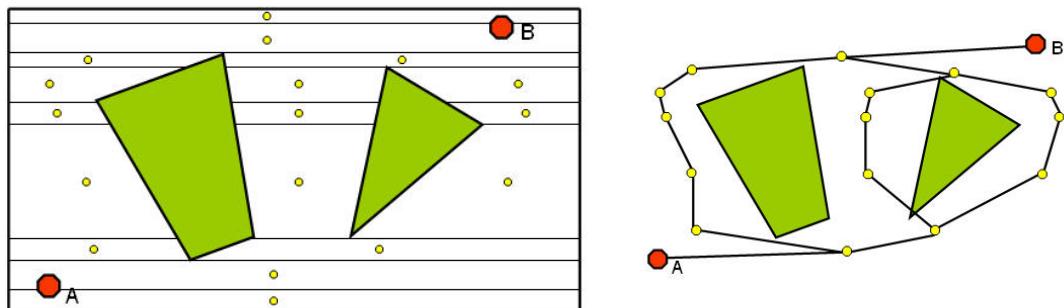


**Figure 4. 2 Decomposing the safe configuration space into horizontal slices**

Exact cell decomposition is the most well-known method. In this approach, the free configuration space outside the obstacles is divided into a number of non-overlapping horizontal or vertical cells. With the given starting and destination points, the polygonal environment are decomposed into cells by drawing the extending lines from each critical points of the obstacle. In most cases, the mid-point of each line segment that is adjacent in the configuration space is connected to form a connectivity graph (Figure 4.3). But in some studies, the centroids of cells are connected to form a graph instead (Figure 4.4).



**Figure 4. 3 Forming a graph (right) by using the mid-points of each line segment (left)**



**Figure 4. 4 Forming a graph (right) by using the centroids of cells (left)**

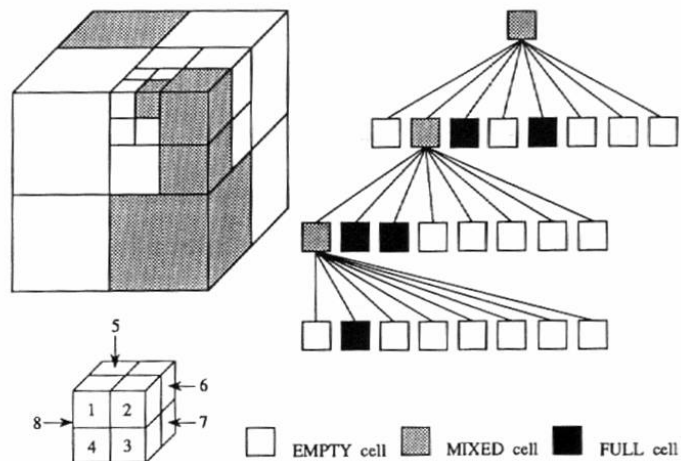


The first approach of using mid-points will create more links than the second one. It may lead to a longer run time of the algorithm but it yields a more accurate graph for routing. Figure 4.4 is not acceptable, because a link was created while cutting across the triangular obstacle on the right.

The main advantage of this method is that it is efficient to implement. The number of cells is linear in the number of edges  $n$  and the decomposition can be obtained in time  $O(n \log n)$ . However, this method has some drawbacks. The cells are not very natural looking. As can be seen in Figure 4.3 and 4.4, the resulting graph tends to be long and skinny, which is not good for navigation purpose.

### **Approximate Cell Decomposition**

In this approach the safe configuration is decomposed into grid of cells where each cell in the grid contains a flag indicating it as free or occupied by obstacles. A partially filled cell is also marked as occupied. There are several ways to decompose a safe environment into smaller cells. Perhaps the most widely used technique is to compute a quadtree decomposition which is based on the principle of recursive decomposition of space. In quadtree decomposition, a coarse resolution is used to encode large homogeneous areas, while a finer resolution is used for areas of high variability (Figure 4.5).



**Figure 4. 5 Quadtree decomposition**

To capture every important detail, the resolution of the cell should be sufficiently high enough and hence results in a graph with many unnecessary nodes. Thus path finding is not very efficient. Similar to exact cell decomposition, it does not give rise to a natural way of describing the environment, but is very easy to implement. This method normally uses a recursive method to continue subdividing the cells until either a free path is obtained or a resolution limit is attained.

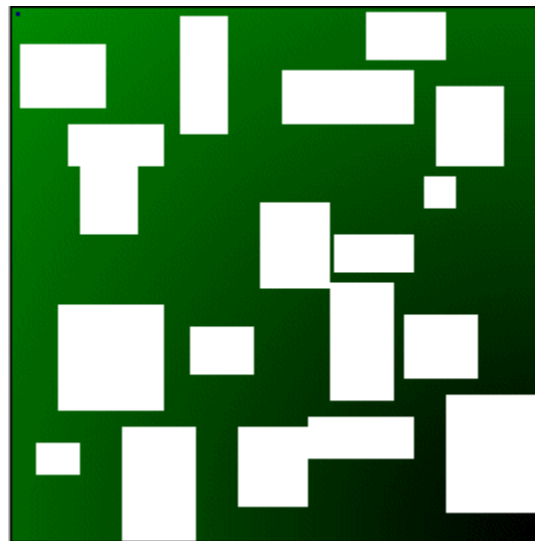
#### **4.4.3 Potential Fields Method**

Potential fields method is based on the idea of attraction and repulsion forces. Attraction is to pull the object toward the destination, while repulsion is to push the object away from the obstacles. To implement this idea, the safe configuration space is divided in squares, sized one pixel each. To start, we need to define a potential field. The term “potential” can be considered as a value to be assigned in each pixel to indicate the state of that pixel under some circumstances. Since our ultimate goal is to find a walking path from a source to a destination, hence the potential values are used as a decisive factor to “walk” in the safe configuration space.





An object is able to “walk” if its nearby pixels have a lower potential value than the one the object stands in at that moment. Based on this rules, the ‘potential’ of the starting location is made as high as possible and the ‘potential’ of the destination as low as possible. To avoid the object entering any obstacles, high ‘potential’ values will be assigned to them such that the object will never be tried to enter the obstacles. The potential field is illustrated in Figure 4.6. In this figure, darker green indicates a lower potential and obstacles are shown in white. The source is the black pixel in the top left corner and the destination is the red pixel in the lower right corner.



**Figure 4.6 Potential Fields**

[source: <http://www.gamedev.net/reference/articles/article1125.asp>]

The advantage of this algorithm is easy to implement. But it has several limitations. One major problem is that object may get ‘stuck’ in local minima, yet it is not our destination. The potential fields algorithm will be more complicated and less effective to have a complete solution of filling local minima.

The “optimal path computation model” in this study is in terms of computation efficiency. Among these methods, the exact cell decomposition is chosen for this



study due to the consideration of running time and ease of implementation. The resulting graph with optimal number of nodes and arcs is pleasant enough to indicate a direction for pedestrians to follow although it may not be really look natural.



## Chapter 5 An Implementation of Exact Cell Decomposition in GIS

The discussions from previous chapters illustrates the importance of road centerline and the evade effort in producing road centerline network for GIS network analysis. In the last chapter, we discussed some alternatives to construct graphs automatically. In this chapter, one of the possible approaches, Exact Cell Decomposition, will be implemented with topographic features in a prototype for routing purposes. Details of the implementation will be discussed.

### 5.1 Prototype Specification

A prototype of cell decomposition algorithm for finding pedestrian path is developed to implement and verify the cell decomposition approach in constructing graphs with the input of topographic map features. The prototype will be implemented in the ArcMap environment of ArcGIS 8.3 by ESRI which provides excellent mapping and analysis functions. A graph will be generated automatically in the shapefile format. Since the network analyst is not compatible in ArcGIS 8.3, ArcView 3.2a with network analyst is chosen for path finding due to its user friendly interface and its built-in Dijkstra's Algorithm.

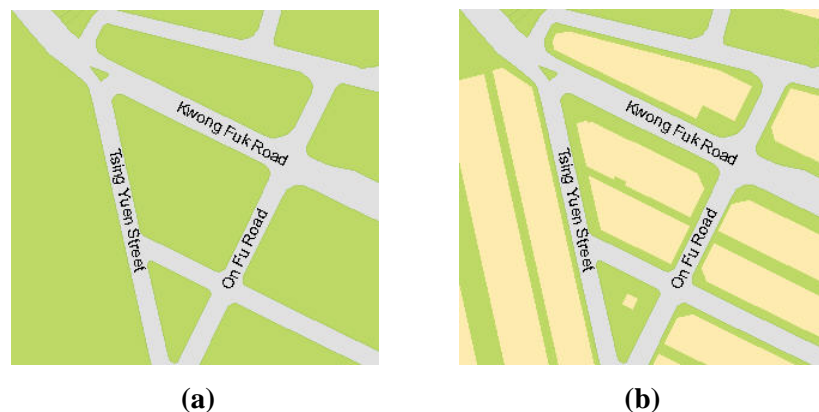
Development Environment	Windows XP and Solaris 9 ArcToolBox, ArcMap of ArcGIS 8.3, ArcView 3.2a, ArcInfo Workstation
Programming Language	Visual Basic and AML (Arc Macro Language)
Component	ArcObjects

#### 5.1.1 Data Requirement

To transform the motion planning problem into a topographical path planning problem, a configuration space is needed as the basic requirement. The procedure



involves converting a dataset with accessible areas for pedestrians such as ‘street block’ polygons to form a discrete local configuration space. A ‘Street block’ in this thesis is defined as an area bounded by one or more streets on all sides. Infrastructures like buildings are built on top of it and pedestrians can move freely around these infrastructures within each street block (Figure 5.1a). Another dataset containing buildings or podium polygons will be considered as obstacles for pedestrians (Figure 5.1b). These two datasets are compulsory to be input in the prototype. The implementation of the decomposition approach cannot be performed well without these two polygonal datasets due to the insignificant information provided.



**Figure 5. 1 Street block (green) and building block (yellow)**

Other supplementary datasets like bridge, subways and zebra- crossing will be used in this project also (Table 5.1). Most of the digital data for testing was obtained from the Land Information Center (LIC), Lands Department HKSAR. For data like zebra-crossings which cannot be obtained from Lands Department or other departments in the Hong Kong Government, these are digitized manually for testing.



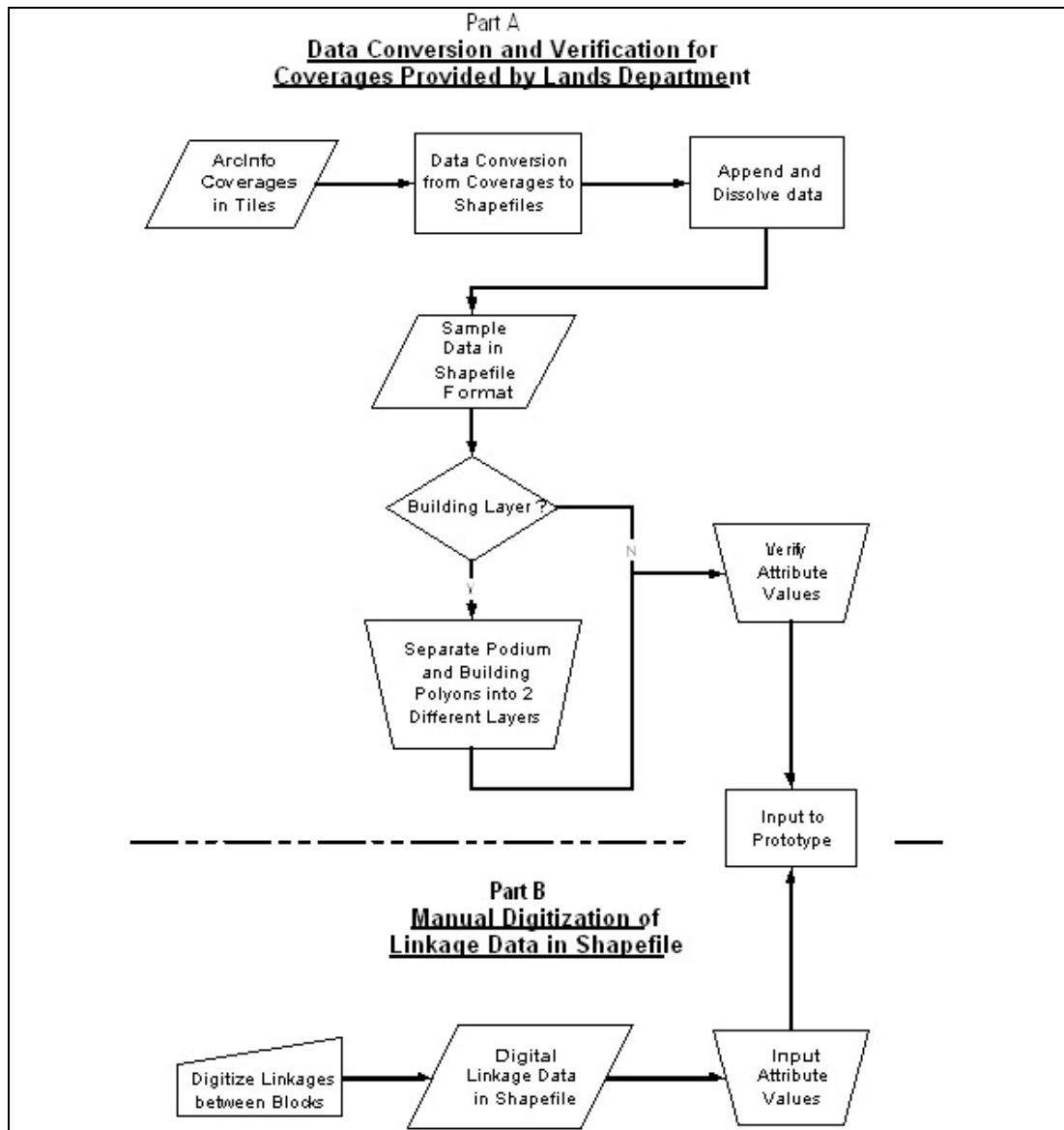
Data Name	Data Format	Geometry	Source	Description	Mandatory
Building	Arc/Info e00	Polygon	LIC	Buildings (obstacles)	Y
Roadfill	Arc/Info e00	Polygon	LIC	Accessible area, e.g. Street Blocks	Y
Bridgefill	Arc/Info e00	Polygon	LIC	Linkages which connect accessible area together (e.g. footbridge)	N
Links	Shapefile	Line	Digitized manually	Linkages which cannot be obtained from Lands Government	N

**Table 5.2 Table of testing dataset**

### Data Conversion and Preparation

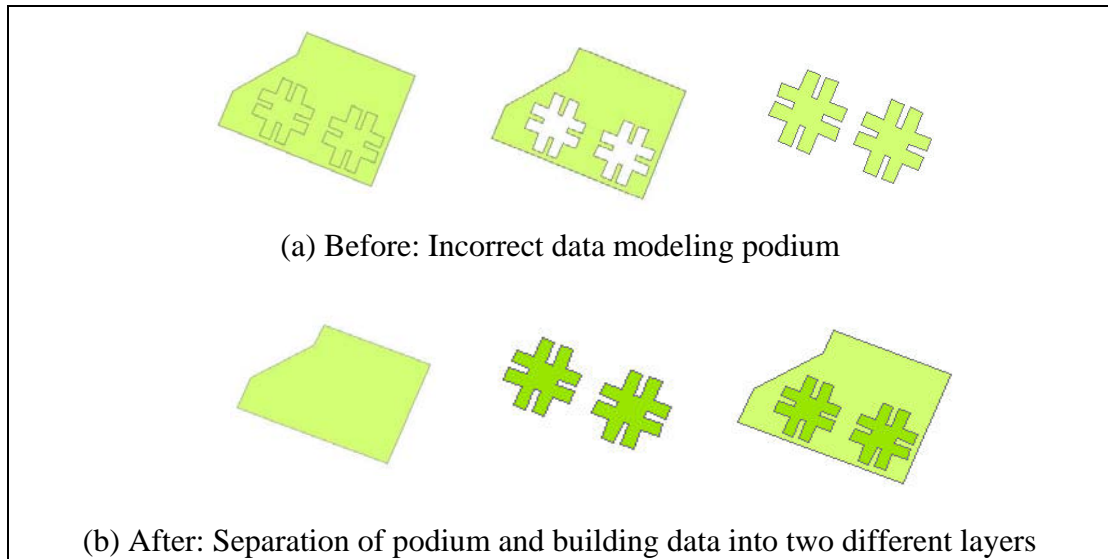
Figure 5.2 shows the data preparation process. To begin, all digital data in Arc/Info e00 format were converted into coverages in batch using the ArcToolBox. All tile based data were then combined into a single coverage and dissolved to merge adjacent polygons which cross different tiles. Next, coverages were then converted into the shapefile format and edited in ArcMap.

Basically, digital map data from any departments or organizations can be used as long as the features comply with the prototype requirement. To make the test data meet the prototype requirement, some data modifications are needed in order to support the implementation of the prototype. First, the building polygons of the LIC data were re-classified. Second, building polygons and podiums were re-digitized as solid polygons and separated into two layers (Figure 5.3(b)). In LIC data, building polygons and podiums are represented in the same layer (Figure 5.3(a)). This is due to the limitation of Arc/Info which does not support overlapping features. Podium in LIC data is digitized as a hollow polygon when there is/are building polygon(s) on top of it. For those sample data like linkages which are not available from Lands Department, data were digitized manually for testing.



**Figure 5. 2 Workflow of data preparation**

To test the algorithm, only a small area is needed. Some sample areas are chosen to reflect the complexity of real world.



**Figure 5. 3 Re-digitization of podium polygons**

## **5. 2 Conceptual Model**

The implementation of prototype was divided into two levels: local and global. Most of the computations was performed at the local level which involved the generation of connected graphs by using the exact cell decomposition within each discrete street block. Those separate graphs resulting from this level are sufficient enough to support the routing application in each single block. However, people movement is not restricted in only one street block. Therefore, triangulation was used at the global level in order to generate links to connect street blocks together. As can be seen in the next section, extra links were created after the triangulation. Since many redundant links were created in the triangulation process, all the links completely inside a street block were removed. Hence, only links connecting two different blocks were left. A complete connected graph was then created by joining these extra links to those graphs resulted at the local level. At the end of the algorithm, a graph would result and be stored in the shapefile format. With this graph, a path could be obtained without any difficulties in any software with the optimum path algorithm.



To transform a physical motion planning problem into a purely geometrical path planning problem, it should be clarified that all building blocks which include podiums are considered as inaccessible due to the unavailability of data. Actually, exact cell decomposition can also be applied at the podium level with the consideration of building blocks as obstacles. But without the information of entrance/exit point of podium, the connectivity of the graph within the podium and that within street block is hard to create. Furthermore, this study aims to generate the network automatically and generate a path to indicate the walking direction for pedestrians. Hence, it is assumed that the interested area should not cover a huge area and therefore, the graph in the prototype is designed to generate solutions on-the-fly.

### ***5.3 Detailed Description of the Algorithm***

The flow of the prototype is depicted in Figure 5.4. Before the algorithm starts, the user should input the two compulsory polygonal dataset: street and building blocks. Next, s/he is required to input the source and destination by clicking the locations on top of the map. The prototype is then able to determine an interested area as defined by the extent of that two input points. It is noted that the algorithm only concerns those features intersecting with the interested area. Hence the execution time of the exact cell decomposition and the path finding algorithms is independent of the number of features input by the users.

After the interested area is defined, the exact cell decomposition is then implemented in each street block. The implementation of this algorithm in each street block is considered at a local level. The interim result of the local level is further processed at the global level. This aims to create extra linkages to connect those





graphs created at the local level. Detailed descriptions of these two-level processes can be found in the next section. As shown in Figure 5.4, the prototype is processed in three different environments, but this prototype can be tested in ArcGIS on Windows platform if the Tin and Network analyst extension are available.

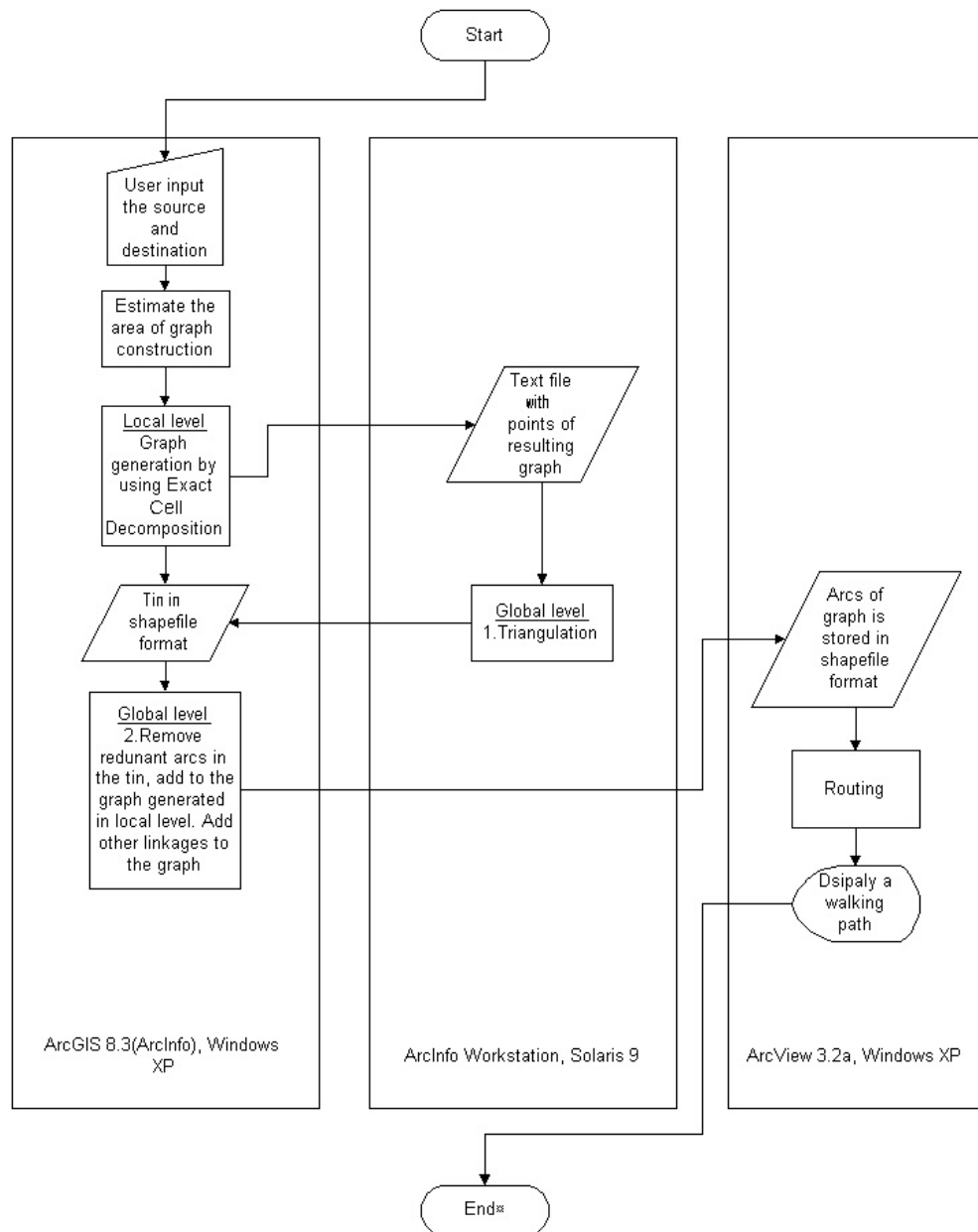


Figure 5. 3 The implementation of the prototype

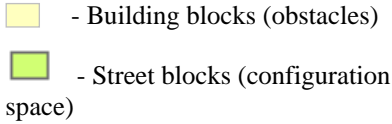
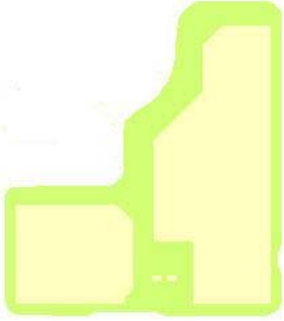
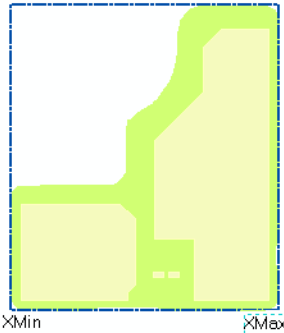
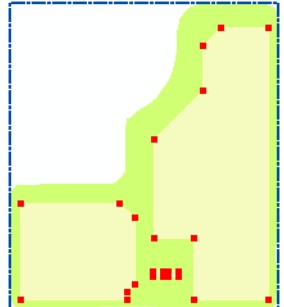


### 5.3.1 Local Level – Exact Cell Decomposition Implementation

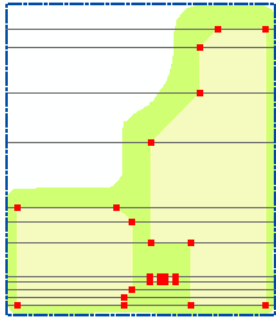
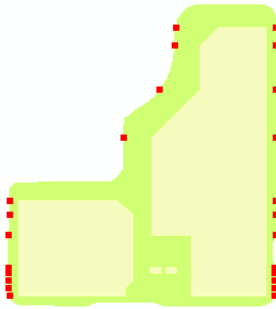
At the local level, the algorithm aims at constructing discrete skeleton maps automatically within each street block by using exact cell decomposition with provided base map features. The data to be input at this level are street block (StBk) and building block (Bldg) features and the resulting graph will be stored in two separate files in the shapefile format. Instead of using a large area to illustrate the algorithm, a single street block with some building polygons was selected to demonstrate how the algorithm works. Pseudocode of the algorithm is provided in the appendix.

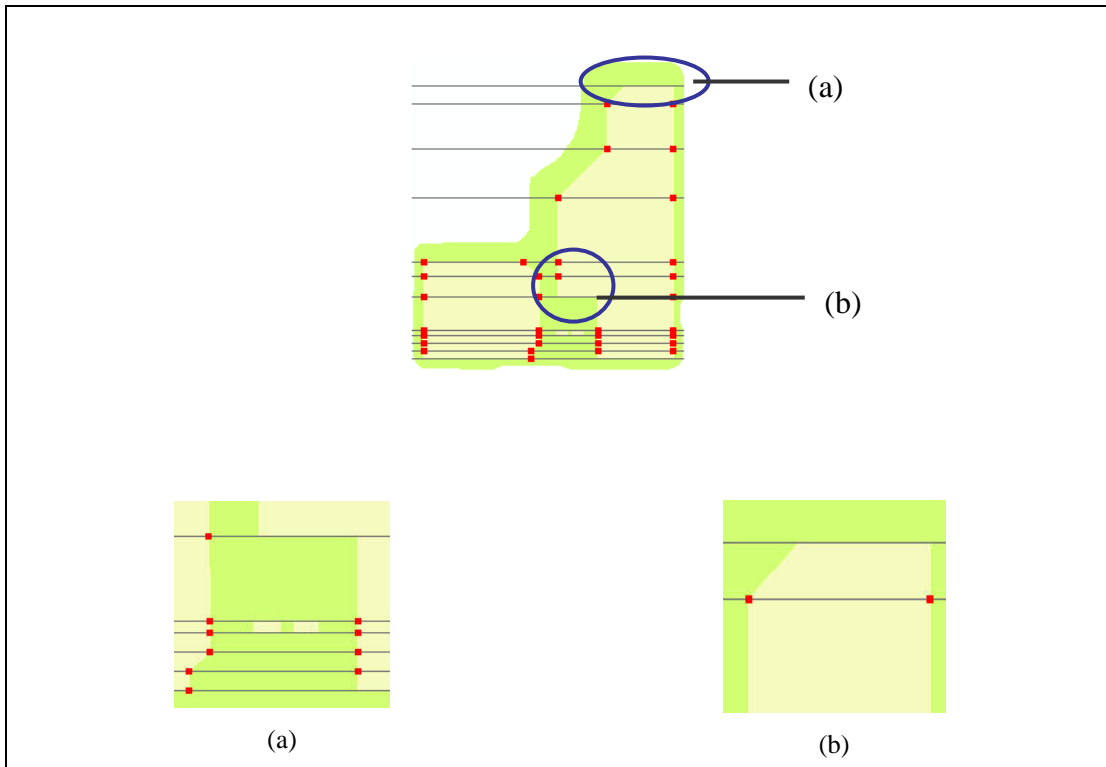
Input features: Street Block (StBk), Building Block (Bldg)

Output data: Two shapefiles containing arcs and nodes of resulting graph separately

	
<p><u>Step 1 – Get the extent of particular street block</u></p> <p>To have a rectangular configuration space, get the XMin XMax, YMin and YMax of the geometry extent.</p>	
<p><u>Step 2 – Get all the critical points of obstacles</u></p> <p>For each obstacle inside the configuration space, get all the critical vertices <math>(x_i, y_i)</math> as shown in red and store in a geometry bag (PtColl).</p>	

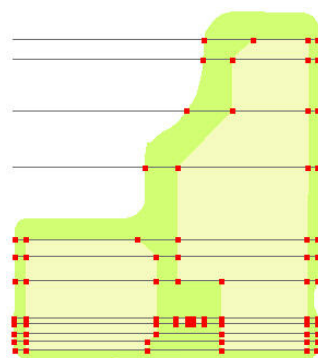


<p><u>Step 3 – Draw Horizontal Lines at Each Critical Point</u></p> <p>Draw a horizontal line (<math>L_n</math>) at each critical vertex, set the from-point and to-point coordinates of <math>L_n</math> as <math>(X_{Min}, y_i)</math> and <math>(X_{Max}, y_i)</math> respectively. If there are 2 critical points with the same <math>y</math> coordinate, two exactly <math>L_n</math> would be drawn. Store all the lines <math>L_n</math>s in <math>L_nColl</math>.</p>	
<p><u>Step 4 – Get the Intersection Points Between Each Line and Street Block</u></p> <p>Intersect <math>L_n</math> with the configuration space. For each <math>L_n</math>, there should be two intersection points after overlaying with the street block. Add all these intersection points in <math>PtColl</math> also.</p>	
<p><u>Step 5 – Data Cleaning</u></p> <p>Now, we have two geometry bags: one containing all the critical vertices of obstacles and the intersection points lying on the boundary of the configuration space and the other containing all the horizontal lines. It is noted that some geometric objects in these two geometry bags are duplicated. In the next step, all the geometries in these two geometry bags will be sorted and duplicated objects removed.</p>	
<p><u>Step 6 – Interest Lines with the Building Block</u></p> <p>All the lines <math>L_n</math> in <math>L_nColl</math> are to intersect with building blocks within the street block. The red points in the figure below represent the resultant intersection points which are stored in <math>pNode</math>. As can be seen, the ‘intersect’ spatial analysis function in ArcGIS fails to compute the intersection points of two parallel lines. This means if the input line (<math>L_n</math>) is exactly parallel to one side of the building block, the computation maybe incomplete as shown in (a) and (b).</p>	



### Step 7 – Draw all Critical Points

To obtain all the intersection points, all the points in pNode and all the critical points resulted in Step 4 were combined together. Then all the duplicated points were removed and all the points needed for further computation were stored in a new geometry bag (pNewUnion).

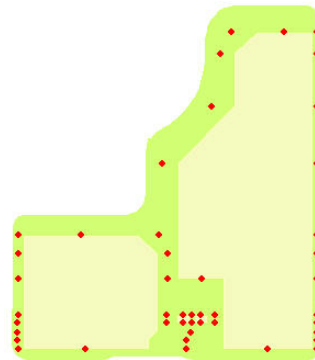


The mid-points between each pair of points are then calculated. With all these mid-points, a graph can be created. The generation of a graph involves two procedures: the generation of horizontal lines and the generation of vertical lines.



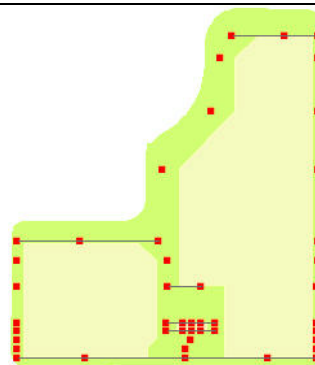
Step 8 – Compute Mid-Points

Overlay each Ln in LnColl with pNewUnion, hence every time there is a series of points with the same y-coordinates are extracted. The coordinates of the mid-points can be calculated by using the mid-point formula:  $\{(x1 + x2) / 2, (y1 + y2) / 2\}$ . All the coordinates of these mid-points are then exported to a text file for triangulation. A series of points is passed to the next step to draw horizontal lines.



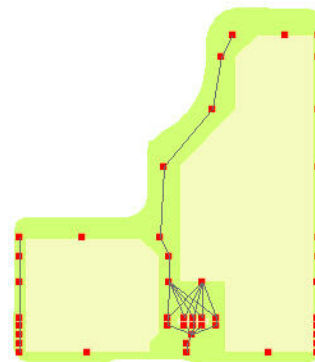
Step 9 – Draw Horizontal Lines

A horizontal line is drawn for each pair of point. But only those horizontal lines without intersecting with any obstacles features are considered as part of the resulting graph.

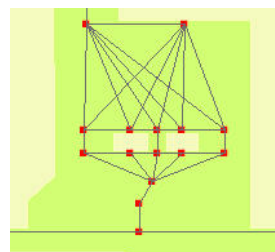
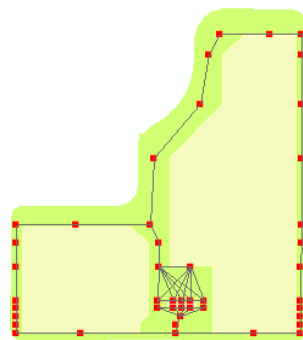


Step 10 – Draw Vertical Lines

Each time, two series of points (pPtColl1 and pPtColl2) are processed. For each point in pPtColl1, vertical lines are drawn to join all the points in pPtColl2. Similar to step 9, only those vertical lines without overlapping with any obstacles features are taken into consideration.

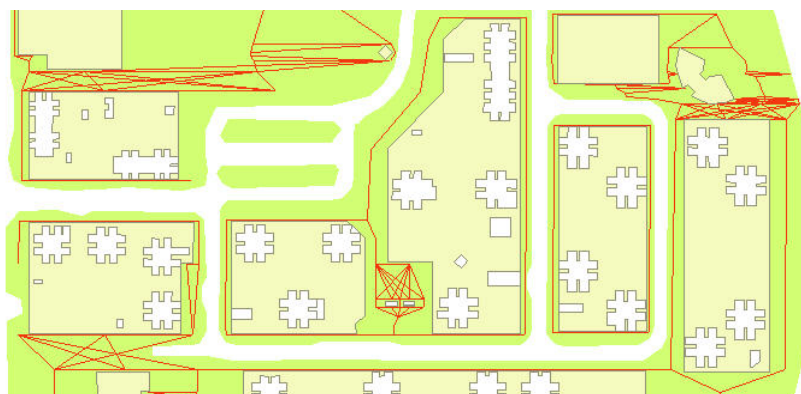


Step 11 – Complete the Graph





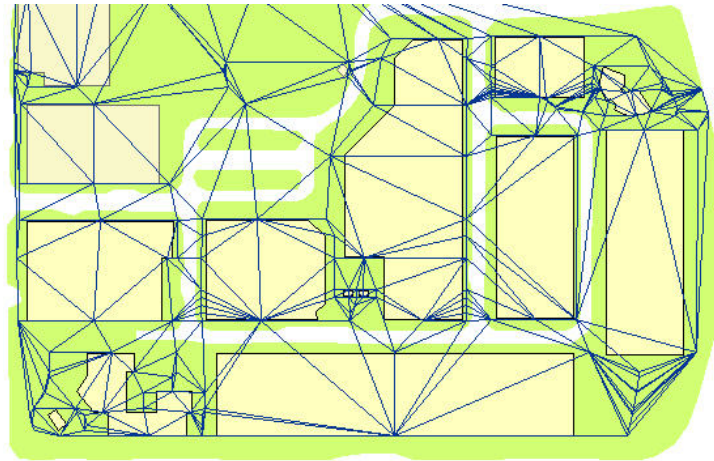
Taking all horizontal and vertical lines, a connected graph will result. The arcs in the graph are then converted from the graphics format into a shapefile. At the end of this local level processing, the x-y coordinates of all the mid-points are exported to a text file for triangulation. Figure 5.5 shows the discrete graphs after the implementation at the local level.



**Figure 5. 4 The generation of discrete connected graphs by using Exact Cell Decomposition**

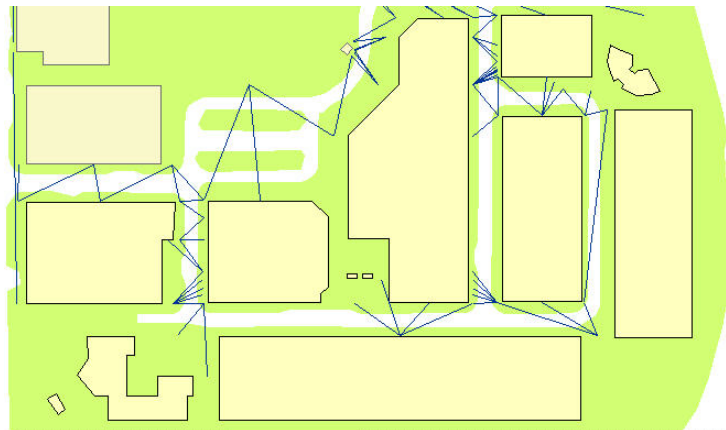
### **5.3.2 Global Level – Triangulation and Generation of a Global Graph**

As can be seen from Figure 5.5, a graph is formed within each space of local configuration. Hence, triangulation in global level is used to create extra connectors to link the discrete graphs together. The text file resulted from the local level processing that contains the x and y coordinates of the mid-points was first imported to the ArcInfo workstation to generate a point coverage. All the points in this coverage are then triangulated to form a tin model (Figure 5.6)



**Figure 5. 5 Triangulation all the nodes of the resulting graph**

Redundant links were removed in the next step. Since triangulation aims to produce additional links to join two local configuration spaces together, those links that are completely within the local configuration space and links that intersect with obstacles are both considered as redundant arcs. After the removal of links, the result can be seen in Figure 5.7.



**Figure 5. 6 Filtering of redundant links**

All the resulting links (marked in blue) were then merged with the graphs from the local level (marked in red) as shown in Figure 5.8. Since the graphs were formed with the same set of points, the links were well snapped upon merging to form a complete graph. In fact, this preliminary graph of the optimal route for



pedestrians is established and ready for routing application at this point. In the real environment, pedestrians, when walking between several street blocks, are required to cross the roads via links like bridges or zebra-crossings for the sake of road safety. Yet these may not appear in the LIC dataset, so there is a need to supplement these features for more accurate routing.'

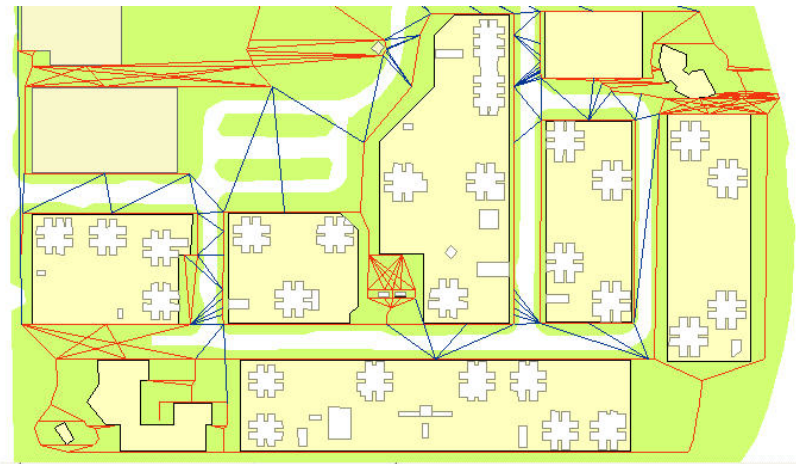


Figure 5.7 The completed graph

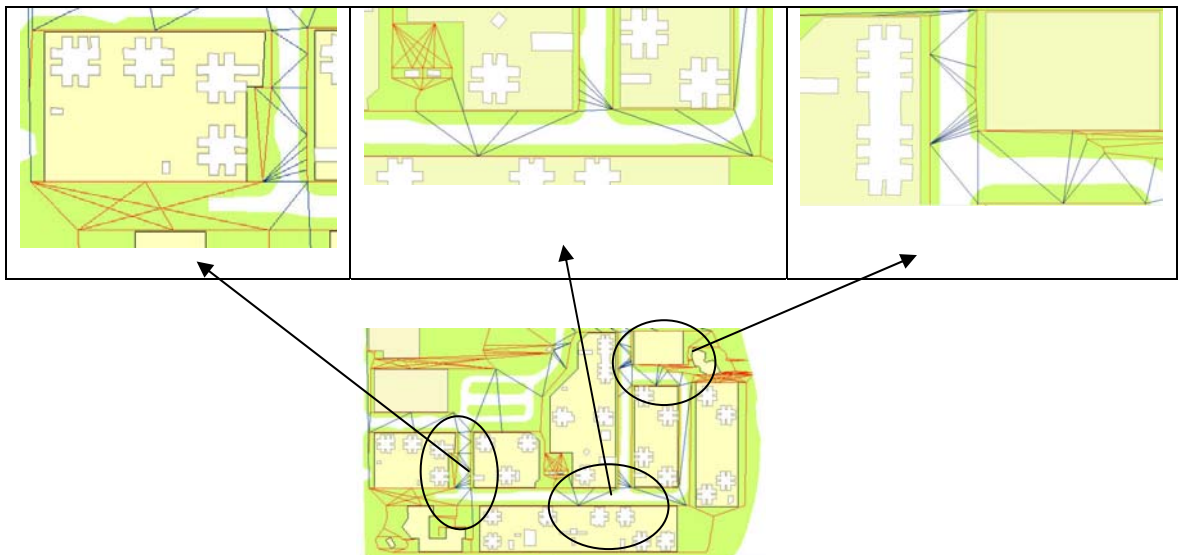
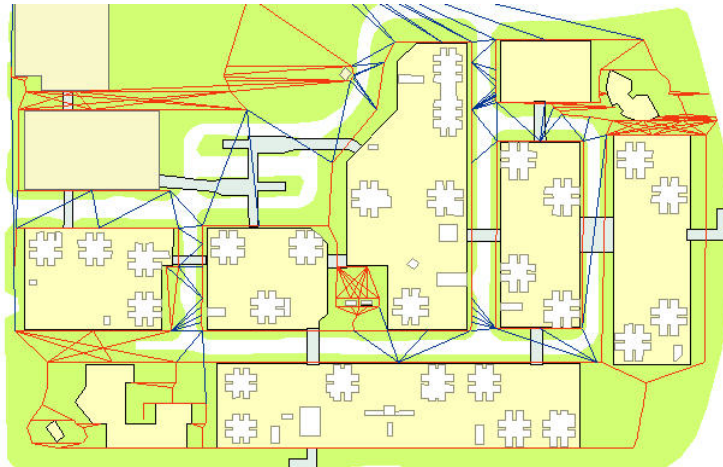


Figure 5.9 shows some footbridges (in blue) which are available to be added to the existing graph. To add bridges onto the existing graph as one kind of connectors, bridge polygons are intersected with the graphs and those overlapping

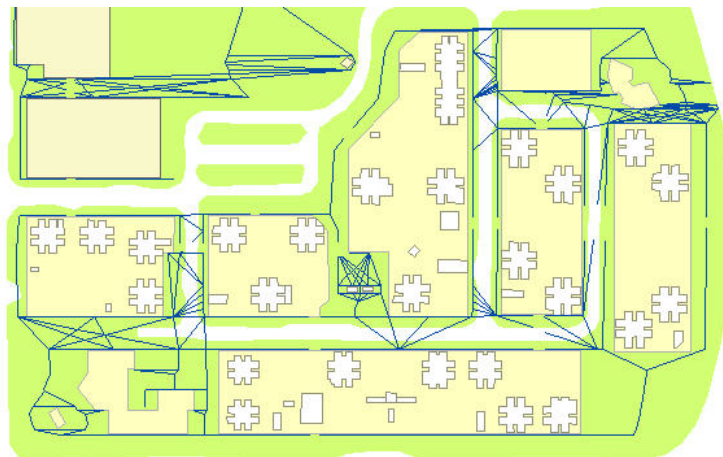




arc segments are then removed by erasing the bridge-symbolized polygons (Figure 5.10)

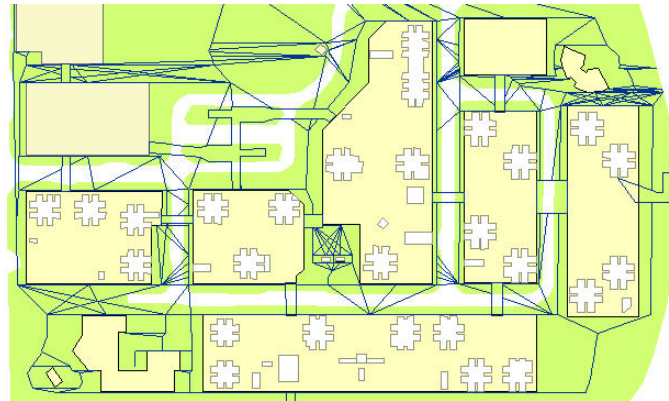


**Figure 5. 8 Add additional dataset to the graph**



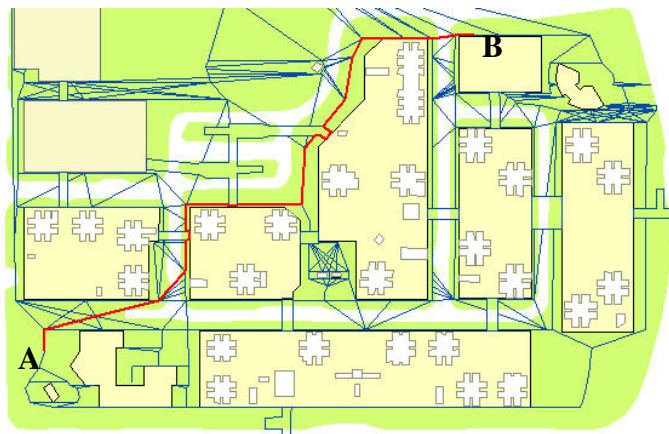
**Figure 5. 9 The removal of link segments by erasing with bridges**

After erasing the overlapping arc segments, all the arcs from the bridge layer are then appended to the graph as shown in Figure 5.11.



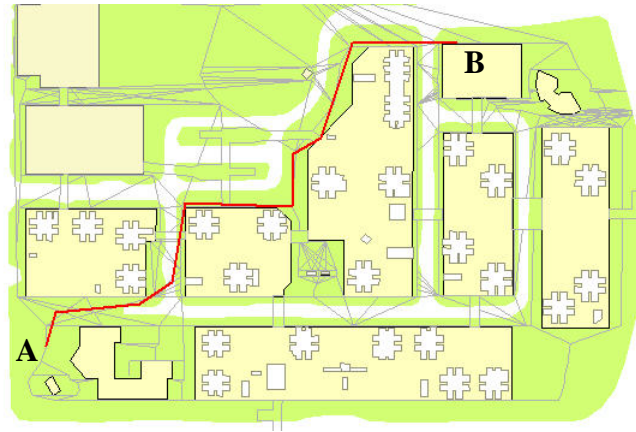
**Figure 5.10 Completed graph with bridges added**

The resultant graph is then imported into ArcView 3.2a with the network analyst extension to perform best route analysis. The best route based on the user-defined source A and destination B is mark in red in Figure 5.12. The computed path aims to show pedestrians the walking direction. When comparing this path with reality, it does suggest a reasonable and accessible path to indicate the user to walk from A to B.



**Figure 5.11 Resulting path from source A to destination B**

Compared with the graph in Figure 5.12, the path in Figure 5.13 is slightly generalized at 0.1m tolerance by using the Douglas-Peucker Algorithm which is embedded in a built-in generalization tool in ArcGIS. Instead of following the geometry of bridges as shown in Figure 5.12, the path in figure 5.13 is smoother. This makes the result more presentable and proper.



**Figure 5. 12 Generalization of the resulting path**

This chapter shows inspiring results being achieved by implementing the exact cell decomposition, triangulation and spatial analysis in deriving a walking path for pedestrians. Discussions of the results will be covered in the next chapter.



## **Chapter 6**                      **Discussion and Conclusion**

In the last chapter, we discuss the details of the implementation algorithm. This chapter describes the evaluation and the analysis of the proposed algorithm. The aim of the evaluation is to find out the reliability of the resulting path and the flexibility of the proposed algorithm. Furthermore, the limitations of the algorithms will be discussed and further development of this project will be suggested.

### **6.1 Achievements**

In Chapter 5, a prototype is implemented to allow users to input topographic map features and identify the source and destination. A prototype (which appears as a black box to general users) is performed to generate traversable links automatically in accessible area by using exact cell decomposition algorithm along with other spatial analyses. The ability of this algorithm is further expanded with triangulation to connect all the graphs generated by exact cell decomposition algorithm within each street block. The resulted routes or network can be exported into other GIS formats if users need to make further analysis.

The encouraging results mentioned in the last chapter suggest that the finding path is possible to be used automatically by using topographic map features even without any manually pre-prepared network. Although only a few cases are selected to demonstrate how the algorithm works, those cases are composed most of the complexities in the real world such as irregular shape of buildings, several street blocks and different levels of structures such as subways and footbridges. In the



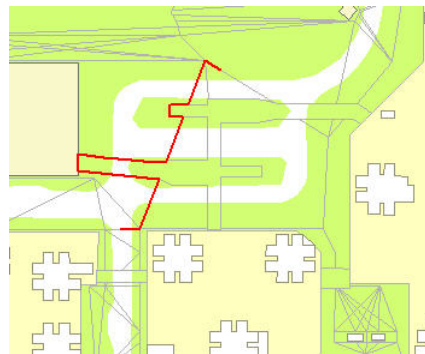
future, there should be no problem to cater for more complexed 3D structures into account.

Exact cell decomposition is a recognized algorithm to generate a collision-free path in a 2D environment with polygonal obstacles. Furthermore, the aim of this research study is to generate walking path for pedestrians. The accuracy of a walking path is less important compared with that of a driving path. The graphs generated in the prototype should be reliable enough for pedestrians to be traversed.

## **6.2 Discussion of Result**

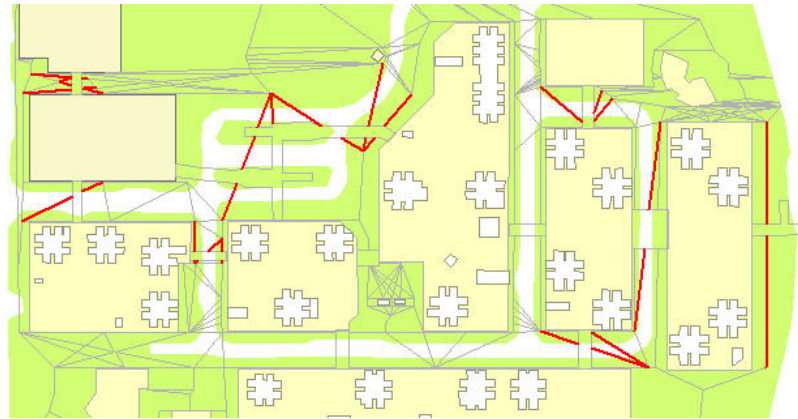
### **6.2.1 Limitation of the Algorithm**

Features like footbridges, building blocks and street blocks etc. are taken into consideration in the prototype. Misleading path may result due to insufficient data input. Referring to Figure 6.1, the optimal way for user to walk from A to B is to cross the bridge.



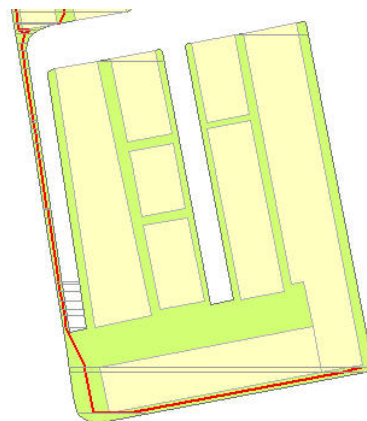
**Figure 6. 1 Links created along bridges**

But since the entrance or exit points of bridges are not available to the prototype, pedestrian is now suggested to cross the bridge in a very inappropriate way. In our world, there should be some entrance or exit points between on-ground features and elevated features. Hence the links highlighted in Figure 6.2 should not be connected along the bridge but the two ends of the bridge features.



**Figure 6. 2 Incorrect links connecting the ground and bridges**

Second, the algorithm does not seem to work well when the configuration space is in concave shape. Referring to the implementation of exact cell decomposition algorithm as described in step 3 and step 4 in section 5.3, horizontal lines are drawn at each critical point with different y-coordinates of building polygons and all the lines are then intersected with the street blocks. When a line intersects with a convex polygon, it is obvious that two intersection points and one line are resulted. The algorithm has no problem to handle this. However, when a line intersects with a concave polygon, more than two intersection points and lines are created depending on the complexity of the concave polygon. Extra points and lines generated make variance in the algorithm computation. As can be seen in Figure 6.3, the graph is only generated on the left of the concave polygon but not on the right.



**Figure 6. 3 Incomplete graph in concave graph**



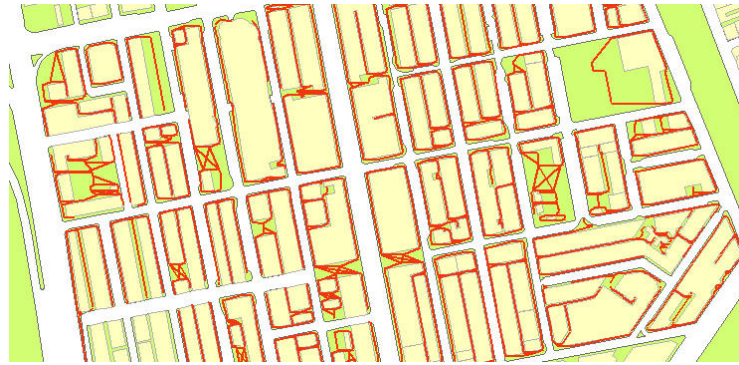
### 6.2.2 Flexibility of Exact Cell Decomposition in GIS Navigation

Exact cell decomposition algorithm is chosen to implement in this research project. It demonstrated that it is one of the solutions to generate a traversable links with obstacles. This algorithm was originally a solution of motion planning. In robot motion planning, a robot is designed to move in a virtual environment which is normally factitious made. The complexity and uniformity of the sampling obstacles are relatively simplest than the real world. As illustrated in the last chapter, the complex shape of obstacles results an intricate graph and many redundant arcs are generated. This graph might seem very perplexing to those unfamiliar with the proposed algorithm. This is not only reduces the efficiency of path finding algorithm, but also required applicable generalization to make the unnatural looking to be more pleasant.

Through the implementation of the prototype, it demonstrates that cell decomposition only works well for those geometries with uniform shapes. It seems that cell decomposition method can work well in those areas with less complicated building structure like industrial areas and Mongkok (Figure 6.4) in Hong Kong. Buildings in residential areas are normally too complicated and results a lot of critical points of shape for computations.



(a)



(b)



(c)

**Figure 6. 4 Implementation of Exact Cell Decomposition algorithm in Mong Kok (a) Mong Kok (b) Resulted graph (c) Details of the graph**

### 6.2.3 Running Time of the Package

As mentioned in Chapter 4, the running time of the exact cell decomposition is  $O(n \log n)$ . The performance of this algorithm is considered as probably acceptable. However, the prototype included not only this algorithm, but also triangulation, other overlay analyzes, optimal path algorithm and graphic to feature conversion functions etc. It is quite hard to estimate the running time of the prototype, especially that the prototype is implemented in different platforms and environments. To shorten the response time, the generation of graphs can be done in an offline mode instead.





### **6.3 Limitation of Project**

#### **6.3.1 Data Availability**

With no doubt, every successful GIS application project lies on the data availability and the quality of the data source. With respect to the objectives of this project, some supplementary data such as zebra-crossing and entrance points of buildings etc. is relatively unavailable for this research project. Although some of them are manually digitized, the credibility of resulting path is reduced.

#### **6.3.2 Software**

The major limitation in this project is due to the software used. As can be seen in Figure 5.4, the algorithm is first developed in ArcGIS 8.3 which is not fully compatible. At the end of the local level, a set of points is exported to other computer with Tin extension to perform the triangulation. The tin is then imported back to the prototype and continue the implementation. But since the network analyst is not yet developed in ArcGIS 8.3, the path finding part is forced to work in other software ArcView 3.2a. However, the implementation of the prototype will be simpler if there is a development environment which can provide both tin and network analyst components (e.g. ArcGIS 9.0 or above). The triangulation and the path finding can all be done in one environment and the existing process of importing and exporting data to different environments can be omitted.

### **6.4 Conclusion and Future Work**

The current project addresses and presents a new method for computing the optimal path with a connectivity graph and information about the path and topographic features. All the input to the proposed model is only a digital map with its associate topographic features, without the pre-prepared of any network. The



model is based on the exact cell decomposition methods to generate a connectivity graph by connecting a path between any two adjacent horizontal cells. The possible routes can be generated to provide useful information for pedestrians in determining their ways. The main advantages of this new method is the identification of users' familiarized land features from a digital map by themselves so as to obtain the desired results of path finding. The model and the customized program automatically handle and generate the routes and information at real time, without the generation and maintenance of the arc-node network data. This prototype could be further investigated, say, with the considerations of modeling the entrance point of buildings and facilities (e.g. lift, stairs) inside buildings such that the walking path could become more realistic to have both "horizontal" and "vertical" movement of pedestrians.

In this stage, the application of the model is on searching an optimal path for pedestrians. Further work will be done to apply the model for driving path. It is understandable that the accuracy required for a driving path is higher than with a walking path. Therefore, some sophisticated assumptions or considerations are needed to take into account in the algorithm. For this particular case, the algorithm should consider a road width, road speed, road direction and the existence of barriers etc. To have a more feasible and flexible model to solve path finding or planning in various applications, further analytical and theoretical works and studies are required to explore the feasibility of the proposed algorithm.

This study introduces a new idea of using exact cell decomposition methods to generate paths automatically with topographic map features to find paths for



pedestrians. In fact, the main objective of a transportation system is to improve individual accessibility. However, most navigation application in GIS mainly focuses on vehicle navigation, so the concern of individual accessibility is relatively less common. This research is a significant and useful initial step for researchers to investigate and contribute themselves in this area.



## REFERENCE

1. Ahrikencheikh, Cherif (1994). Network theory for efficient optimized motion planning, Ann Arbor, Mich. : U.M.I., 1994, c1993
2. Bell M.G. H. and Iida Yasunori (1997). Transportation Network Analysis. Chichester, New York : John Wiley & Sons
3. Bernhardsen Tor (2002). Geographic information systems: an introduction 3<sup>rd</sup> Ed. New York, N.Y. : John Wiley & Sons
4. Carter Michael W. and Price Camille C. (2001) Operations Research: A Practical Introduction. Boca Raton : CRC Press
5. Chen Yen Liang and Yang Hsu Hao (1999). Shortest Path in Traffic-light Networks. Transportation Research Part B 34 241-253.
6. Dueker Kenneth J. and Bulter J.Allison (2000). A Geographic Information System Framework for Transportation Data Sharing. Transportation Research Part C 8 3-26.
7. Evans James R. and Minioka Edward (1992). Optimization Algorithms for Networks and Graphs, Second Edition. Marcel Dekker, Inc. New York
8. GameDev.net – Motion Planning Using Potential Fields  
Available at:  
<http://www.gamedev.net/reference/articles/article1125.asp> (Accessed on 22th July 2006)
9. Goodchild Michael F. (2000). GIS and Transportation: Status and Challenges. GeoInformatica 4:2, 127-139, 2000.
10. Han Gang, Jang Jie and Chen Jun (2001). Optimizing Path Finding in Vehicle Navigation Considering Turn Penalties and Prohibitions. ISPRS, Vol.34, Part 2W2, “Dynamic and Multi-Dimensional GIS”, Bangkok, May 23-25, 2001. pp.118-122
11. Horowitz Alan.J (1996). Integrating GIS Concepts into Transportation Network Data Structures. Transportation Planning and Technol., Vol. 21, pp. 139-153.
12. Husdal, J. (2000) How to make a straight line square - Network analysis in raster GIS. Unpublished, thesis for the MSc in GIS, University of Leicester, UK.



Available at:

<http://husdal.com/mscgis/thesis/> (Accessed on 21st June 2006)

13. Kirkby S.D., Pollitt S.E.P. and Eklund P.W (1996). Implementing a Shortest Path Algorithm in a 3D GIS Environment. 96' Spatial Data Handling Conference, Delph, The Netherlands, August 12-16, 1996, pp.7b31-42
14. Lo C.P. and Yeung Albert K.W. (2002). Concepts and Techniques of Geographic Information Systems. Upper Saddle River, NJ: Prentice Hall
15. Martin A. (1979). Graph Theory and Network Flow. Clarendon Press, Oxford
16. Miller Harvey J. and Shaw Shih Lung (2001). Geographic Information Systems for Transportation: Principles and Applications. New York : Oxford University Press, 2001
17. Rodrigue Jean Paul (2002). Graph Theory: Definition and Properties  
Available at:  
<http://people.hofstra.edu/geotrans/eng/ch2en/meth2en/ch2m1en.html>  
(Accessed on 5th May 2004)
18. Sutton John (1996). Data Attribution and Network Representation Issues in GIS and Transportation. Transportation Planning and Technol., Vol.21, pp.25-44.
19. Vuren Van T. and Jansen G.R.M (1998). Recent Developments in Path Finding Algorithms: A Review. Transportation Planning and Technology, 1988, Vol.12, pp.57-71
20. Zhan F.Benjamin and Noon Charles E. (1998). Shortest Path Algorithms: An Evaluation using Real Road Networks. Transportation Science Vol.32, No.1, pp.65-73
21. Latombe Jean Claude. (1991) Robot Motion Planning. Boston : Kluwer Academic Publishers
22. Nadine Tschichold Gurman and Nora H. Sleumer. (1999) Exact Cell Decomposition of Arrangements used for Path Planning in Robotics. ETH Zürich, Technical Report 329. Available at:  
<http://webdoc.sub.gwdg.de/ebook/ah/2000/ethz/tech-reports/3xx/abstracts/abstract329.html> (Accessed on 19th December 2004)



## BIBLIOGRAPHY

1. Buckley Fred, Harary Frank (1990) Distance in Graphs. Addison-Wesley Publishing Company
2. Carre, Bernard (1979). Graphs and Networks. Oxford : Clarendon Press ; New York : Oxford University Press
3. Chrisman Nicholas (1997). Exploring Geographic Information Systems 2<sup>nd</sup> Ed. New York ; Chichester : John Wiley
4. Goodchild Michael F. (1996). Geographic Information Systems and Disaggregate Transportation Modeling. NCGIA Symposium 1-10, June 7-8
5. Longley Paul A., Goodchild Michael F., Maguire David J., and Rhind David W. (2001). Geographic Information Systems and Science. Chichester, England : Wiley
6. Papows Jeffrey and Max Stevens-Guille (2002). Optimal Routing Requires More than Maps. GeoWorld, January 2002, pp.34-36.
7. Ralston Bruce A. (2000). GIS and ITS Traffic Assignment: Issues in Dynamic User-Optimal Assignments. Geoinformatica 4:2, 231-242
8. Sutton John and Gillingwater David (1997). Geographic Information Systems and Transportation – Overview. Transportation Planning and Technol., Vol.21, pp. 1-4.
9. Thill Jean-Claude (2000). Geographic Information Systems for Transportation in Perspective. Transportation Research Part C 8 pp.3-12.
10. Transportation Research Board – National Research Council (1992) Vehicle Routing, Traveler ADIS, Network Modeling, and Advanced Control Systems. Washington, D.C. National Academy Press.
11. Transportation Research Board – National Research Council (1992). Transportation Planning, Programming, Land Use, and Applications of Geographic Information Systems. Washington, D.C. National Academy Press.
12. Transportation Research Board – National Research Council (1995). Artificial Intelligence and Geographical Information. Washington, D.C. National Academy Press.



13. Transportation Research Board – National Research Council (1999) Artificial Intelligence and Geographical Information. Washington, D.C. National Academy Press.
14. Verbyla David L. (2002). Practical GIS Analysis. London ; New York : Taylor & Francis
15. Wu Yi Hwa, Miller Harvey J. and Hung Ming Chih (2001). A GIS-based Decision Support System for Analysis of Route Choice in Congested Urban Road Networks. *Journal of Geographical Systems* 3:3-24.



## APPENDIX – Pseudocode

### Procedure Construct\_Graph (StBk, Bldg) {

```
    For each feature p in StBk do {
        Compute XMax, XMin of its envelop
        For each segment S in particular p do {
            Store all starting point CPt (x, y) in PtColl for each S // all critical vertices of p
            Construct a line Ln through each CPt, where
                Ln.FromPoint.x = xMin - 1
                LnFromPoint.y = CPt.y
                LnToPoint.x = xMax + 1
                LnToPoint.y = CPt.y
            Intersect Ln with p and store all intersection points in PtColl
            Construct and store Ln for each pair of intersection points in LnColl
        }until S in p is nothing
    }until p in StBk is nothing
    If LnColl is not nothing then
        Set SortLn = Sorted LnColl by y coordinates
        Set ClearLn = Cleaned SortLn without any duplicated lines
        Set SortPt = Sorted PtColl by x coordinates
        Set ClearPt = Cleaned SortPt without any duplicated points
        Set pNode = ConstructNode (ClearLn, Bldg)
        Set pUnionPt = pNode union PtColl
        Set pNewUnionPt = fix (the y-coordinates of each point in pUnionPt)
        Set pConstructLn = ConstructLn(pClearLn, pNewUnion, Bldg, p)
    End if
}
```

### Function ConstructNode (ClearLn, Bldg) {

```
    For each line Ln in ClearLn do {
        Set a temporary variable pTempLn for storing each Ln in ClearLn
        For each feature p in Bldg do {
            Set pTempLn = Ln
            Intersect pTempLn with p
            Store all intersection points in pNode
        }until p in Bldg is nothing
    }until Ln in ClearLn is nothing
} Return pNode
```





**Function ConstructLn (pClearLn, pNewUnion, Bldg, StBk) {**

```
For each Ln in pClearLn do {
    Set Ln1 = new polyline, where
    Ln1.FromPoint.x = Ln.FromPoint.x
    Ln1.FromPoint.y = Fix(Ln.FromPoint.y)
    Ln1.ToPoint.x = Ln.ToPoint.x
    TestY = Ln1.ToPoint.y = Fix(Ln.ToPoint.y)
    Intersect Ln1 with pNewUnion and store the result in pPtColl_1
    For each feature p in Bldg do {
        bmin = bmax = false
        //bmin and bmax are used to indicate if offset is needed for the resulting path
        Construct the convex hull of p and get its ymin and ymax
        If TestY = ymin Then
            bmin = True
            Exit do
        End if

        If TestY = ymax Then
            bmax = True
            Exit do
        End if

    }until p in Bldg is nothing
    If bmin = True Then
        Set pPtColl2 = CalMidPt(pPtColl1, p, "bmin")
    ElseIf bmax = True Then
        Set pPtColl2 = CalMidPt(pPtColl1, p, "bmax")
    Else
        Set pPtColl2 = CalMidPt(pPtColl1, p, "false")
    End if
}until Ln in pClearLn is nothing

// Draw horizontal lines and store in pTmpHorColl
Set pTmpHorColl = DrawHorLn(pPtColl2, Bldg)
For each Ln in pTmpHorColl {
    Add each Ln in pGraphLn
}until Ln in pTmpHorColl is nothing

Set pTmpHorColl = nothing
Set pTmpHorColl = DrawVertLn(pPtColl1, pPtColl2, Bldg, StBk)
```



```
For each Ln in pTmpHorColl {
    Add each Ln in pGraphLn
}until Ln in pTmpHorColl is nothing
} Return pGraphLn
```

**Function DrawHorLn (pPtColl1, Bldg) {**

```
Set DrawHorLn = New GeometryBag
For i = 0 to pPtColl1.PointCount - 2
    Set pHorLn = New Polyline
    Set pHorLn.ToPoint = pPtColl1.Point(i)
    Set pHorLn.FromPoint = pPtColl1.Point(i+1)
    If pHorLn is not intersect of any feature of Bldg then
        Add pHorLn in DrawHorLn geometry bag
    End if
Next i
} Return DrawHorLn
```

**Function DrawVertLn (pPtColl1, pPtColl2, Bldg, StBk) {**

```
Set DrawVertLn = New GeometryBag
For i = 0 to pPtColl1.PointCount - 1
    For j = 0 to pPtColl2.PointCount - 1
        Set pVertLn = New PolyLine
        Set pVertLn.FromPoint = pPtColl1.Point(i)
        Set pVertLn.ToPoint = pPtColl2.Point(j)
        If pVertLn is not intersect of any feature of Bldg and StBk then
            Add pVertLn in DrawVertLn
        End if
    Next j
Next i
} Return DrawVertLn
```