ANALYSIS OF DIJKSTRA'S AND A* ALGORITHM TO FIND THE SHORTEST PATH

AMANI SALEH ALIJA

A thesis submitted in
fulfillment of the requirements for the award of the
Degree of Master of Computer Science (Software Engineering)

Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia

SEPTEMBER, 2015

ABSTRACT

There are so many algorithms used to find the shortest path such as Dijkstra, A* algorithm, Genetic algorithm, Floyd algorithm and Ant algorithm. In this study, two algorithms will be focused on. This study compares the Dijkstra's, and A* algorithm to estimate search time and distance of algorithms to find the shortest path. It needs the appropriate algorithm to search the shortest path. Therefore, the purpose of this research is to explore which is the best shortest path algorithm by comparing the two types of algorithms. Such that it can be used to solve the problem path, search to analyze their efficiency in an environment based on two dimensional matrix which is best because of lookup time. This study implements the algorithm in visual C++ 2008 and design interface for the algorithms that allow the user to find the shortest path with the search time and the distance by determine the size of map, starting node and the destination node. The experimental result showed the search time of A* algorithm is faster than Dijkstra's algorithm with average value 466ms and the distance is same of the both of algorithms.

ABSTRAK

Terdapat banyak algoritma yang digunakan untuk mencari laluan terpendek seperti Dijkstra dan algoritma A*, algoritma genetic, algoritma Floyd dan algoritma Ant. Untuk kajian ini, dua jenis algoritma akan ditekankan. Kajian ini akan membandingkan masa carian daripada Dijkstra dan algoritma A * untuk mengetahui algoritma yang paling pantas apabila mencari laluan terpendek. Ia perlu algoritma yang sesuai untuk mencari laluan terpendek. Oleh itu, tujuan kajian ini adalah untuk meneroka laluan algoritma terpendek yang terbaik dengan membandingkan dua jenis algoritma yang boleh digunakan untuk menyelesaikan masalah laluan carian untuk dianalisa kecekapannya. Lingkungan berdasarkan matriks dua dimensi adalah yang terbaik oleh kerana masa pencariannya. Kajian ini akan menggunakan algoritma visual C++ 2008 dan reka bentuk antaramuka untuk algoritma yang membolehkan pengguna untuk mencari laluan terpendek dengan carian masa dan jarak dengan menentukan saiz peta, nod permulaan dan destinasi nod.Keputusan experimental menunjukkan carian masa bagi algoritma A* adalah lebih pantas daripada algoritma Djikstra dengan nilai purata 466ms dan mempunyai jarak yang sama. Bagi kebanyakkan komputer moden, memori tidak begitu penting, dan jika ia penting sekalipun, kawasan yang tidak digunakan boleh disembunyikan dan dimuaturun bila diperlukan. Untuk membandingkan jalan algoritma terpendek di antara dua algoritma yang digunakan, satu antaramuka telah direkabentuk. Ini adalah supaya perbandingan dapat ditentukan dan algortima yang mana yang lebih sesuai untuk mencari laluan terpendek.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

Path finding is defined as the process of moving an object from its earlier position to the final position. Different application areas used Path Finding Algorithms (PFA). These include Games and Virtual Tours, Driverless Vehicles, Robot Motion and Navigation.

Path finding is usually described as a process of finding a path between two points in a certain environment. In most cases the objective is to find the shortest path possible, which would be optimal i.e., the shortest, cheapest or simplest. Several criteria such as, path which imitates path chosen by a person, path which requires the lowest amount of fuel, or from two points A and B through point C is often found relevant in many path finding tasks.

Finding the shortest path is the most difficult issue in many fields, starting with navigational systems, artificial intelligence and ending with computer simulations. Although these fields have their own specific algorithms, there are many general purpose path finding algorithms that are applied successfully. However, it remains unclear what benefits certain algorithm have in comparison with others.

Shortest path algorithms are currently used widely. They are the basis of some problems such as network flow problems, tree problems and other related problems. They decide the minimum cost of travel of the problems production cycle, the shortest path in an electric circuit or the most reliable way.

The internet is a vast field where the shortest path algorithm is usually applied. The Internet problems comprise data package transmissions with minimal time or using the most reliable path.

This research attempts to make an implementation of a shortest path algorithm by using A* algorithm and compare it with Dijkstra's algorithm on different criteria, including search time and the distance were implemented to analyze their efficiency in an environment based on 2 dimensional matrix.

A* algorithm is an algorithm that is widely used in path finding and graph traversal. The process of plotting a resourcefully traversal path between points is called nodes. A* traverses the chart and follows the lowermost known path, keeping a sorted priority queue of alternate path sections along the system. If at any position, a segment of the path being traversed has a higher cost as compared to another encountered path segment, it leaves the higher-cost path segment and traverse low cost path segment instead this procedure continues until the goal is reached.

The search space can be reduced by the use of an efficient heuristic function. Without heuristic or when the heuristic function equals to zero, A* becomes Dijkstra‟s path finding algorithm. In addition, if it is extremely high, A* turns into BFS. Therefore, the heuristic function plays a vital role in controlling the behavior of A*. If the heuristic function gives a very little value, then A* will become slow to find the shortest path. If heuristic evaluation is very high value, then A* will become very fast. It shows that the tradeoff between speed and accuracy of the algorithm is dependent on heuristics. Therefore, a heuristic should be chosen very cautiously, keeping in mind this tradeoff. A heuristic that is specific to the problem should be used in algorithms.The time complication of A* depends on the heuristic.

Considering the worst case scenario, the number of nodes expanded is exponential in the length of the solution, but it is polynomial when the questspace is a tree, there is a single goal state and the empirical function h.

Dijkstra's algorithm is also known as the single-source shortest path problem. It computes the length of the shortest path from the source to each of the vertices on the plot. The single basis shortest path problem can be defined as: Let G= {V, E} be a focused weighted graph with V having the set of vertices. The exceptional vertex s in V, with s as the source and let for each edge e in E. Edge Cost(e),is the length of edge e. All the weights in the plotought benon-negative. Before going in depth about Dijkstra’s procedure talk in detail about directing-weighted graph. Directed graph can

be defined as an ordered pair G: = (V, E) where V is a set, having elements called vertices or nodes and E is a set of ordered pairs of vertices, called directed edges, arcs, or arrows. Directed graphs are also known as a digraph.

1.2 Problem Statement

Path finding generally refers to finding the shortest path between any two locations. Many existing algorithms are designed precisely to solve the shortest path problem such as Genetic, Floyd algorithm.

The method proposed in this study is A* algorithm and Dijkstra's, will probably find the shortest path solution in a very short amount of time and minimum distance. A* algorithm, a kind of informed search, is widely used for finding the shortest path, because the location of starting and ending point is taken into account beforehand. The A* algorithm is a refinement of the shortest path algorithm that directs the search towards the desired goal. A* use the heuristic function to speed up the runtime. The general purpose of heuristic algorithm is to find an optimal solution where the time or resources are limited. Dijkstra algorithm is simple and excellent method for path planning. Dijkstra's algorithm chooses one with the minimum cost until found the goal, but the search is not over as it calculates all possible paths from starting node to the goal, then choose the best solution by comparing which way had the minimum distance. Dijkstra's algorithm is a special case of A* star algorithm where heuristic is zero. It remains current because it is realistically fast and relatively easy to implement.

1.3 Objectives of the Research

For the aim of this research to be achieved, the following objectives should be fulfilled:

(i) To apply A* and Dijkstra 's algorithm to find the shortest path.

(ii) To compare A* and Dijkstra'salgorithm based on distanceand time consumption for the shortest path.

1.4 Scope of Research

The research will focus onlytocompare two algorithm A* and Dijkstra's algorithm based ontime to search and minimum distance on the shortest path in four casesto see how the both of algorithm work in small and big size of search: 8*8 ,22*22, 33*33, 10*10 in two dimensional grid and implement the algorithms inVisual C++2008.

1.5 Motivation

Dijkstra's Algorithm is efficient in finding the shortest possible path despite having a disadvantage of time wasting in exploring directions that are not promising. Greedy Best First Search explores in promising ways, but not necessarily find the shortest possible path efficiencies. The A* algorithm uses the actual distance from the start as well as the estimated distance to the goal. The algorithm is successful because it gathers some information used by Dijkstra's algorithm (supporting vertices that are near to the starting point) and information that Best- First-Search uses (favoring vertices that are close to the goal). It is similar to Dijkstra's procedure in that it can be used to find the shortest path. It is also comparable to Best-First-Search in that it can use a heuristic to guide itself.

1.6 Research Outline

This thesis consists of five chapters. Chapter 1 is an overview and main objectives of the project. It consists of the scope of work covered and the objectives of the project.After this introduction chapter, basic Methods and related work are briefly explained in chapter 2.Chapter 3 discusses the methodology and tools to obtain the entire objectives of this project.Chapter 4 explains the implementation and detailed steps used in this work. Chapter 5 discusses the result and compares it by running time for each method. Chapter 6 includes the objectives achieved, future work, and conclusion.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Before designing the application and the system environment, this chapter gives insight intotechnique used in this project.Furthermore, general uses that are related to thisproject arespecified at the end of this chapter.

2.2 Applications of Path Finding Algorithms

Path finding algorithms are useful in theareaof robotic manipulation, as it can be used to control a robot around difficult terrain without the need of human intervention (Carsten J, 2007). Itwould be profitable if the robot were ona differentplanet like Mars, in which some topography must be avoided, but due to the extremedistances involve, guidingit completely through remote control would be difficult (too much delay in the radio transmission)(Obara T *et al.*, 1994).It could also be useful if the robots are to be operated underwater, where radio waves could not get to it. It also find applications in almost any case where a vehicle needs to go somewhere, while avoiding obstacles, without human intervention.(David M. *et al.*, 2004)other use is in computer games where something needs to be moved from one place to another avoiding any walls or other difficulties in the way.

The algorithms might also be used in determiningthe shortest path to drive betweentwopointson a map. Which is the best way to route an e-mail through computer network or shortest path to run telephone wires through existing circuits.

2.2.1 PFAs in Games and Virtual Tours

Path finding is an important part of game programming. Game typesmove according to the path they (or computer) calculate. There are some algorithms used in the gamechanging with the complexity and purpose of the path calculation. Themostalgorithm found currently in games todayare the A* Algorithm.

"What makes the A* algorithm so appealing is that it is guaranteed to find the best path between any initial point and any ending point, assuming that a path exists." (David M. *et al*., 2004).

Users directed in a virtual building or virtual duplicateof a real building, for example a museum, can see the artworks without having to go to a physical place. This is moralfor eluding traveling long distances and attaining more people to show the works. Users can arrange their visit path and algorithm calculates the route thenthe virtual tour begins.Shafie& Hassan (2004) work on this scenario and developed an application capable of doing path planning process..They used an A* algorithm to find the path. The final path can be seen in a 3D environment (Figure 2.1).

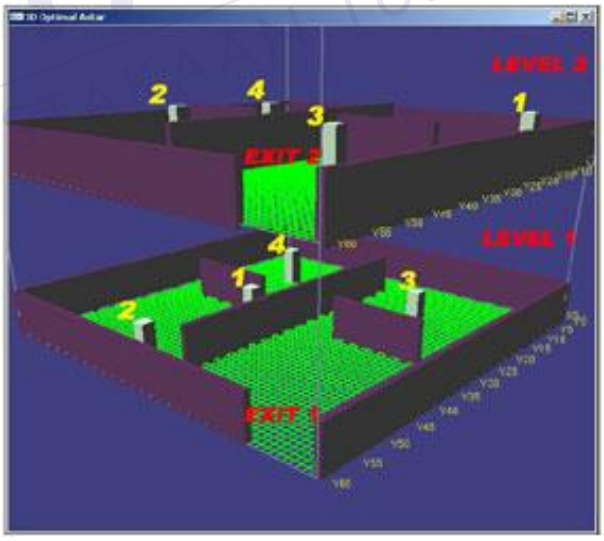

Figure 2.1: 3D Model of Latiff and Hassan 2004
(M. Shafie, & R. Hassan, 2004)

Autonomous triangulation of virtual characters, is also an important task to work on.It is required to prevent obstacles and proceed to their paths without any impairment.In a virtual environmentnot like the robotic navigation, there is no data coming from sensors, there exist the site database only. (Frꝋhlich & Kullmann, 2002)haveusedA* algorithm studies, on the environmental modeling of the area and resolved that the uniform-sized grid cell methodology is a better choice, alsostatedthat A* algorithm works well in the environments where the world isflat,whichsuits our environment well.

2.2.2 Robot Motion and Navigation

Mobile robots moving in an outdoor or indoor environment must have their course scheduling and navigational schemes in order to be able to find their direction. Commonly, the navigation and pathfinding units are placed on the robots so that can move by themselves. It is imperative to have this property, if the usage of these robots is considered. In the military, detection of hazardous or explosive materials and discovery and investigation of unknown areas are very suitable for this type of robots. Razavian and Sun (2005) proposed a new algorithm called Cognitive Based Adaptive Path Planning Algorithm (CBAPPA) comparing it with A* Algorithm and Focused D* Algorithm (D*, aDynamic A* algorithm which is resulting from the A* Algorithm and has some improved added capabilities used for autonomous units),used it to observe the behaviour of biological units and paid attention to the behaviour of ignoring the irrelevant information from the surroundings, trying toreachthetarget speedily. This method may not use optimum paths, but the results were efficient. This algorithm is also prepared for self-processing units, which can be a good example for our future works.

Koenig and Likhachev (2002) announces another new algorithm called “D* Lite” which is a variant of D* Algorithm for improved fast planning in unknown terrain. D* algorithm does better than A* in unknown areas when used in self-directed robots. D* Lite is more effective than D* because it is simpler, shorter and easier to understand. D* and D* Lite algorithms appraise their knowledge about the terrain when they move into it and continuously perform planning of the track. This method

is quite different from our environment, but can be very useful when it is implemented system in various (may be unknown) conditions.

It is stated in Artificial Intelligence (AI) for the Game Developers book(David M. et al., 2004) that: "Pathfinding problem is thought as solved, future works are for making the algorithm fast and more efficient." Studies give more effective and faster algorithms to use and implement in different areas.

2.2.3 Driverless Vehicles

Another area of applicationof the path finding algorithms is the programmed vehicles that will be able to discover its way without making contact with the impediments.In addition, its primary usage is in the military, but it can be implemented in diverse areas of the natural life.

In the future, can hunt for the lost outdoor adventurers, hikers or can be very beneficial in case of natural disasters, which the sensors have.
(Yahja et al.,2000) introduced the D* algorithm which stands for Dynamic A* which they used framed-quadtrees (Figure 2.2) in constructing their environment and data structures, and verified their algorithm in as elf-directed vehicle (Figure 2.3). Their algorithm continuously transforms itself when it gets fresh information about the terrain.

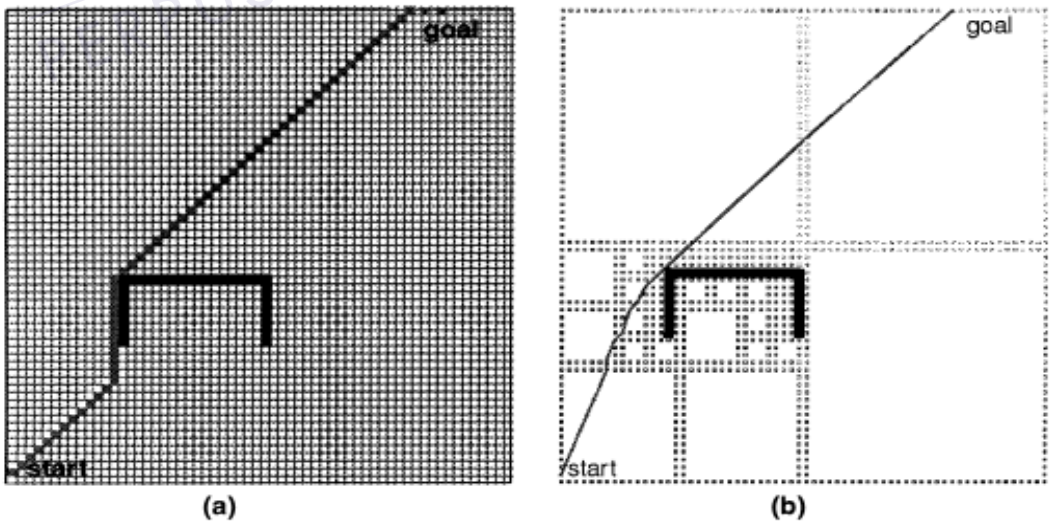

Figure 2.2: (a) Grid (b) Framed-Quadtree approach in D* Algorithm

According to (Yahja*et al.*,2000), if accurate and complete maps were obtainable, it would be appropriate to use a normal search method such as A*.

Figure 2.3: Autonomous vehicle used in the D* algorithm trial

In the United States, there was a race ordered by The Defense Advanced ResearchProjects Agency (DARPA), that contestants were the autonomous ground vehicles from around the globe. The race was held twice until 2007; one in 2004 where none of the participants were capable to complete the course and the second one was in 2005. In 2005, 4 vehicles completed the 132-mile desert track, and the winner is a VWTouareg (Figure 2.4) which was developed by Stanford Racing Team(http://www.stanfordracing.org). This car has a processing system t`o calculate its route while on the road. Onboard, computers control the vehicle from start to finish, and there was not any intervention from the race team. It constantly modifies the path according to the information it gets from its sensors while it moves on the route.

Figure 2.4: Champion DARPA Challenge race 2005

2.2.4 Transportation Networks

In road networks, it is getting more important to find a path to the final end point. If a person is new to a location, sample time can be was ted in locating the end point.

There exist some products established to overcome this difficulty, providing a map of the region. After entering the starting and the final destination, it is likely to acquire the shortest possible track.

Experimental studies were performed to select the best algorithm for using in the path finding process. For instance study by Jacob *et al* (1999) from Los Alamos National Laboratory who compared the shortest path algorithms by doing some experimental analysis on big database (Figure 2.5), as sessed the Dijkstra, A* and modified A* algorithms with respect to the calculation time on the real network solution quality, affluence of execution and the extensibility of the algorithm,Found out that A* algorithm has a better time efficiency.

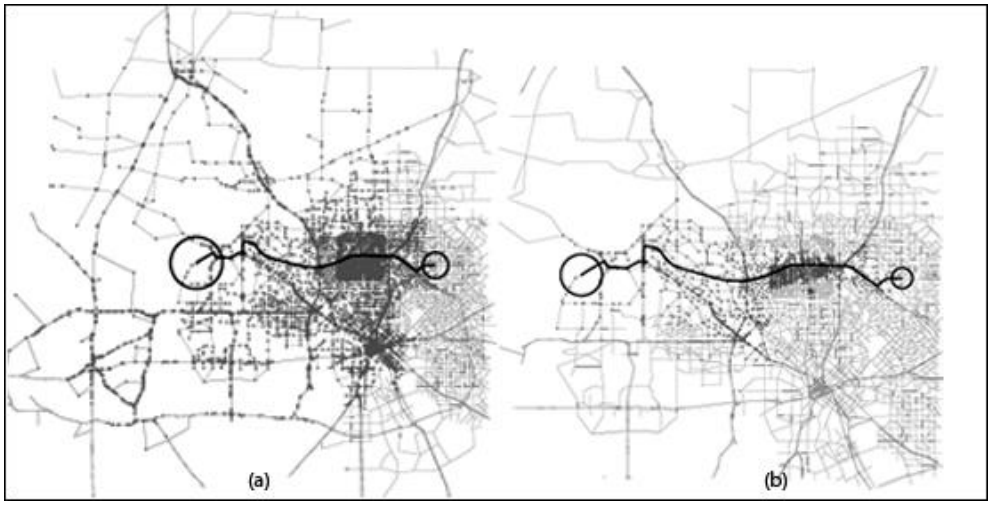

Figure 2.5: (a) Dijkstra's Search (b) A* search on road network of Dallas Ft-Worth urban area

Figures 2.5 showed that A* Search is also much more effectivewith respect to the number of nodes visited. Therefore, Dijkstra's Algorithm searches much more nodes than A* does.

2.2.5 Human Navigation

There are also some studies to help human navigation, especially for visually impaired and deaf-visually impaired people. For example a study by Kulyukin *et al* (2004) used RFID technology and the modified potential field's algorithm for human map reading. They used a robotic director, moving at moderate speeds, thereby detecting the free spaces about itself with sensor and laser range finders (Figure 2.6). Additionally, they sited RFID tags to the objects in the surroundings. The guide senses these tags and discovers its way between the objects. This approach is good for navigating in an unidentified area like anestate and identifying the path without any need for a map database of that area. It also gives the administrator to change the place of the objects without changing any scene database.

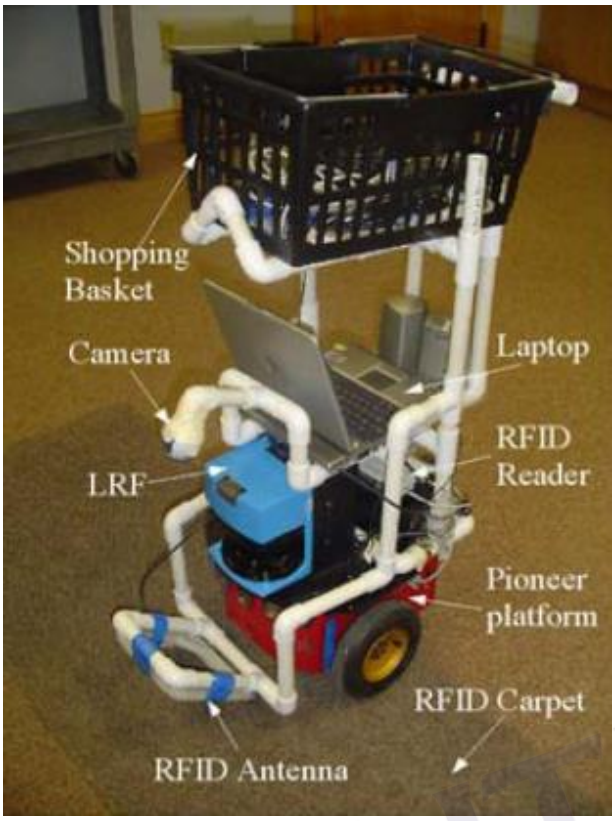

Figure 2.6: Kulyukin's Robotic Aid(Kulyuki *et al.*, 2004)

Amemiya, Yamashita, Hirota and Hirose (2004)used RFID technology, motionsensor together witha wearable computer and a wearable boundaryfor Finger-Braille (Figure 2.7) in order to interact with deaf-visually diminished people. Test area was covered ove 1300 RFID tags. When the user moves, the system does the calculations and gives directions to the user by the vibration alerts on fingers.

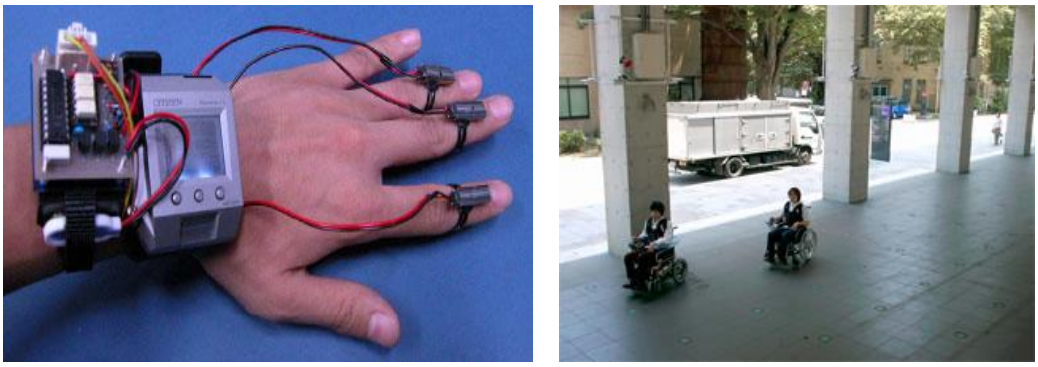

Figure 2.7: Wearable interface for Finger-Braille(Amemiya *et al.*,2004**)**

Beside RFID technology, Global Positioning System (GPS) and ultrasound technology were also used in some studies. Study conducted by Ran et al (2004)proposed a dissimilar system that can be used both outdoor and indoor. The system works with Differential Global Positioning System (DGPS) at outdoor, and with ultrasound positioning system indoors (Figure 2.8). User can shift between these two systems with a simple voice command. The system switches to the ultrasound positioning system indoor, and uses simple geometry calculations to find the location.

Figure 2.8: Drishti's navigation system(Ran *et al*., 2004)

Mostly, all of these systems are established on outdoor navigation which intended to aid the visually impaired people. These systems bring some benefits to users,which make their life easier.However, in order to make shopping available to these people this is need to implement some of that technology in one indoor system.
In all of these systems users have to carry or read some tools which have big capacities. This brings an additionalweight to the operator. These schemes should be improved in order for the integration of the visually impaired people inthe real life.
Feasible systems should comprise of small devices capable of doing all of the requested processes. They should be carried along without any problem.

2.3 Shortest Path Analysis

A shortest path difficulty is to find a route with the least travel cost from one or more roots to one or more destinations through a network (Panahi & Delavar, 2008). Shortest path analysis is important because of its wide range of applications in transportation (Lim & Kim, 2005). (Naqi*et al.*,2010)stated that the shortest path helps calculate the most optimal route, and optimal routing is the process of defining the best route to get from one location to another. The best route could be shorter or faster depending on how it is defined.

The shortest path can be computed either for a given start time or to find the start time and the path that leads to less travel time journeys.

2.4 Shortest Path Algorithm

Due to the nature of routing applications, there is the need forsupple and effective shortest path techniques, both from a processing time point of view and also in terms of the memory requirements. Unfortunately, prior research does not offer a perfectcourse for picking an algorithm when one faces the difficulty of computing shortest paths on real road networks.However, like most popular papers on Shortest Path algorithms, have concentrated their focus on algorithms that guarantee optimality and have worked on tuning data structures used in implementing these algorithms. (*Faramroze Engineer, 2001.).*

In road networks, it is more imperative to discover the way to the final destination. When an individualis new to a place, much time may be wasted inlocating the endpoint. There exist some products established to overcome this difficulty, providing a map of the region. After entering the starting point and the final location, it is possible to get the shortest path.

There are many algorithms that are normally used, ranging from Simplex to complex, in order to be able to solve theshortest path problem(Mustafa,2007).The modest approach is to walk openlyheaded for the goal until met with any kind of obstruction. When come across with any object, direction will be changed and it can be passed by tracing around the obstacle.

Additionally, there is also another algorithm which plans the whole path before moving anywhere. Best-first algorithm expands the nodes based on a heuristic approximation of the cost to the objective. Nodes, which are projected to give the best cost, are expanded first. The most commonly used algorithm is A* algorithm, which is a combination of theDijkstra's algorithm and the best-first algorithm (Mustafa, 2007).

2.4.1 Dijkstra's algorithm

Dijkstra's algorithm is an algorithm named after its developer, E. Dijkstra Oliver j in 1959. Woodman (2007)looks on theuntreatedneighbours of the node closest to the start, and sets or modernizes their distances (in terms of cost, not number of nodes) from the initial point.Dijkstra's Algorithm is a chart search algorithm that unravels the single-source shortest path delinquentfor a plot with non-negative edge path costs, producing a shortest path tree.

Dijkstra's algorithm or variations of it are the most commonly used route finding algorithm for solving the shortest path (Sadeghi-Niaraki *et al.*, 2011). Dijkstra's algorithm is sometimes called the single-source shortest path because it solves the single-source shortest-path difficulty on a subjective, directed graph (G = V, E) where

V is a set whose elements is called vertices (nodes, junctions, or intersections) and E is a set of ordered pairs of vertices entitled directed edges (arcs or road segments). To find a shortest path from a source s vertex or location to a destination location d, Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the sources that already been determined. Knowing that w is the edge weight, the edge is an ordered pair (u, v) and assuming w (u, v) ≥ 0 for each edge (u, v) ϵ E, the algorithm recurrently chooses the vertex u ϵ V – S with the least shortest-path approximation, adds u to S, and relaxes all edges leaving u (Puthuparampil, 2007).

The Dijkstraalgorithm enlarges the node that is at the extremefrom the initial node, so it finishesup "stumbling" into the goal node. Just like the breadth-first search, it is certain tofind the shortest path (Figure 2. 9) (Cormen *et al.*,2001).

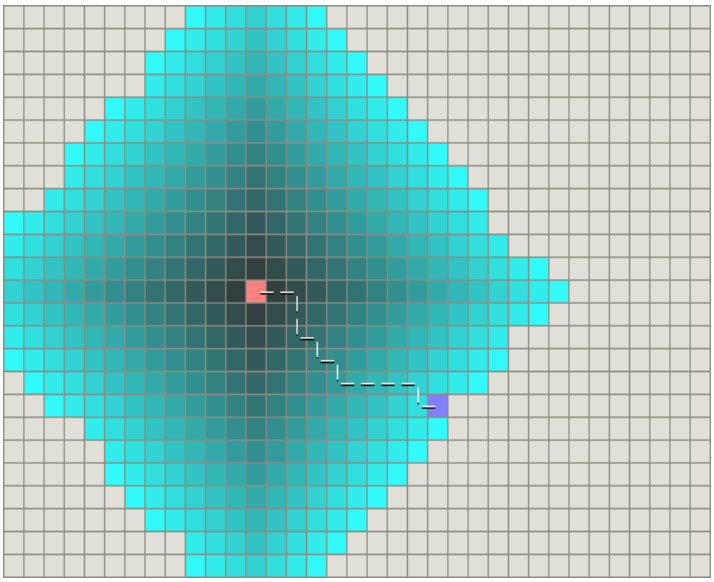

Figure 2.9: Dijkstra's Search

From the figure (Figure 2.10) shown the objective is to find the shortest paths from origin to all other nodes. Dijkstra's will assign zero to initial node and node A while assigning infinity to all other nodes that are not visited. It will then assign a value gradually to get smallest value up to the destination, which is node E.

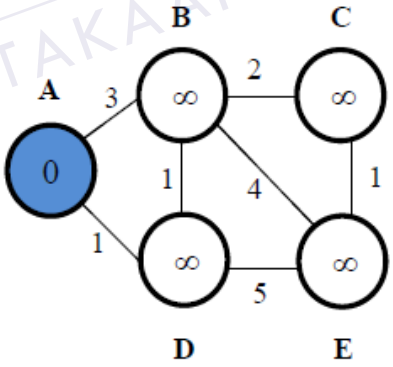

Figure2.10: First step of Dijkstra's Algorithm

Step 1: Node A is set to become current node. Zero is assigned to node A and infinity to all other nodes.

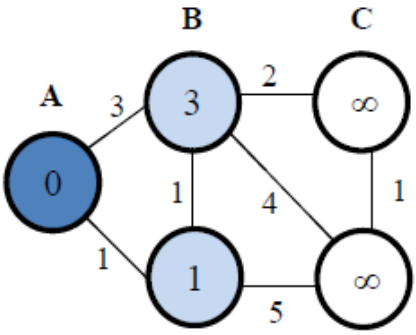

Figure2.11: Second step of Dijkstra's Algorithm

Step 2: Consider all unvisited neighbors and tentative distance will be calculated. Previously recorded value will be replaced since new value less than infinity.

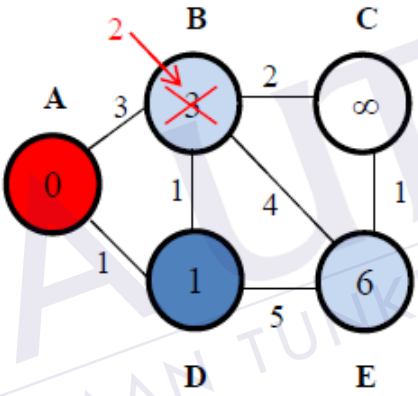

Figure 2.12: Third step of Dijkstra's Algorithm

Step 3: Since all neighbors of node A havebeen taken into account, it is struck as visited and will not be tested again.

The next least distance from node A, node D now will be marked as current node. Itsneighbouring nodes will be updated with the new minimal distance value.

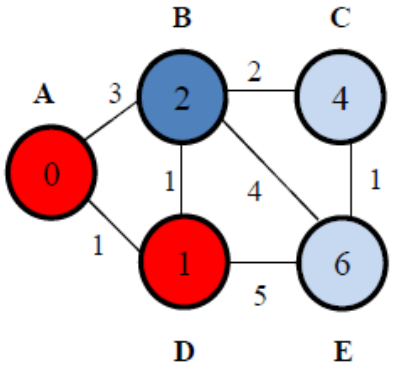

Figure 2.13: Step 4 of Dijkstra's Algorithm

Step 4: Since all neighbors of node D have been taken into account, it is marked as visited and will not be checked over.

The next minimal distance from node D, node B will now be marked as current node. Its neighboring nodes will be updated with the new minimal distance value.

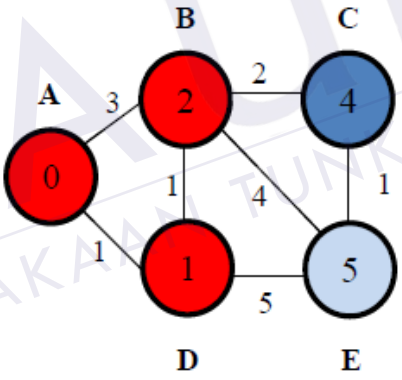

Figure 2.14: Fifth step of Dijkstra's Algorithm

Step 5: Since all neighbours of node B have been accounted for, it is marked as visited and will not be testedover.

The next available minimal space from node B, which is node C now will be taken as present node. Its neighbouring nodes will be updated with afresh minimal distance value.

Step 5: Since all neighbours of node B have been accounted for, it is marked as visited and will not be tested .

The next available minimal space from node B, which is node C now will be taken as present node. Its neighbouring nodes will be updated with a fresh minimal distance value.

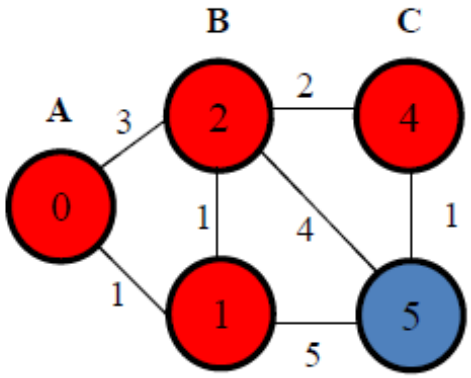

Figure2.15: Step 6 of Dijkstra's Algorithm

Step 6: Meanwhile, all neighbours of node C have been taken into account, it is marked as visited and will not be checked .

The next shortest distance from node C is node E, which will be chosen as current node. Since all the nodes have been visited, the shortest route from node A to node E is found.

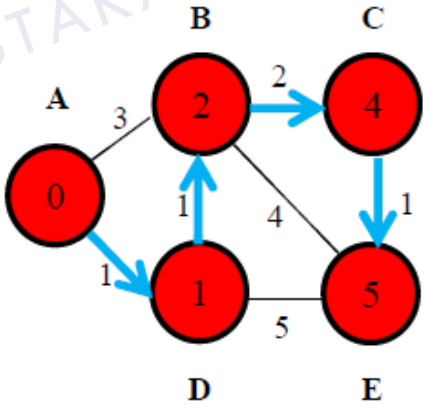

Figure2.16: Shortest path by Dijkstra's Algorithm

The shortest distance from Node A to Node E is:

A → D → B → C → E

2.4.2 The Depth-First Search Algorithm

The depth-first search spreads nodes (it extends a node's descendants before its siblings) until it either reaches the goal or a certain cutoff point, it then goes onto thenextlikelyroute.

The best-first search algorithm is anempirical search algorithm, that cantake knowledge about the plot into account. It is comparable to Dijkstra's algorithm, butit goes to the node nearby to the goal, rather than the node that is extreme from the start.

2.4.3 A* Algorithm

The algorithm was first described in *1968 by Peter Hart, Nils Nilsson, and Bertram Raphael*.It is a universal space-search algorithm that can be used to find the clarificationsto many problems,with shortest pathasone of such.It has been used in several real-time strategy games and is perhaps the most popular shortest path algorithm.

A* is the most popular choice for pathfinding, because it is fairly flexible and can be used in a wide range of contexts. (R.Anbuselvi &R.S.Bhuvaneswaran, 2009).

A* algorithm uses a starting point and anendpoint point to produce the desired path, if it exists (Figure 2.17). In figure 2.17, the cell marked with "O" is the startingpoint/node and the cell marked with "X" is the destination. The white squares arewalkable nodes and the black ones are walls, shelves or any other obstacles.

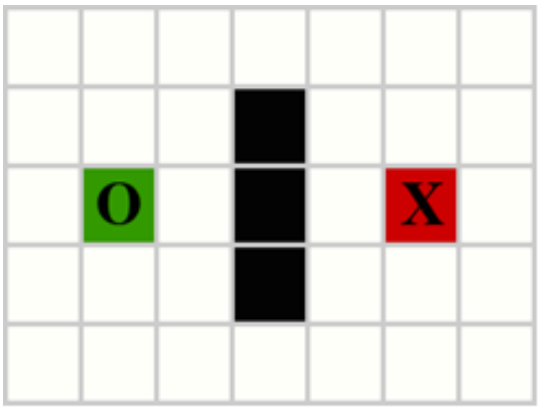

Figure 2.17: Starting and Destination Points in A* Search Algorithm

A* algorithm starts to perform by looking at the starting node first and thenexpansion to the surrounding nodes (Figure 2.18).

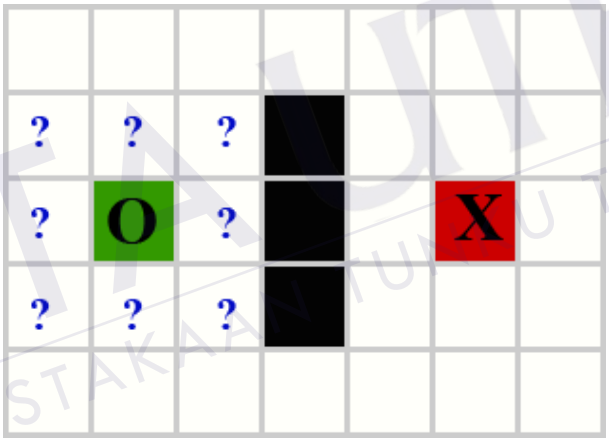

Figure 2.18: Starting the A* Search Algorithm

This operation will be carried on till the endpoint node is reached. In order to have an idea of the nodes, which will be used in the search, A* algorithm requests a way to keep track of the nodes. the nodes to be scrutinized are held in a list, called an Open List. At the beginning, place the starting node to the Open List and after inspecting all of its surrounding nodes move it from the Open List and place in another list named the Closed List (Figure 2.19). Closed List holds the nodes that are visited and there is no need to re-visit its members. When building the Open List, The algorithm checks if the node is walk able. If the node is not walkable, it is not added to the Open List.

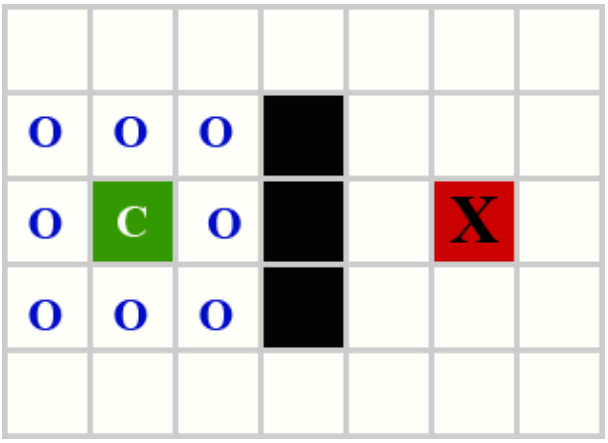

Figure 2.19:Putting the neighbouring cells to the Open List

This procedure prepared for one cell is the main iteration over one A* loop. However, need to track some supplementary information. Need to know how the nodes are linked . Although the Open List maintains a list of adjacent nodes, we need to know how the adjacent nodes are linked as well. Can do this by tracking the parent node of each node in the Open List. A node's parent is the single node that the user steps from in order to get to its current location. On the first iteration over the loop, each node will point to the starting node as its parent (Figure 2.20).

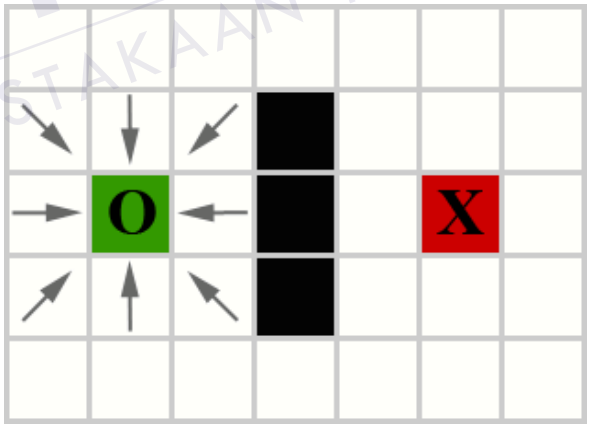

Figure 2.20: Parent relation of Starting Node

Willuse the parent links to trace a path back to the starting node when reach the destination. At this point the process was recommenced. Now have to choose a new node to check from the Open List.At the first iteration there is only a single node in the Open List. Now have eight nodes in the Open List, and the node which will first

be inspected is determined by assigning a score to each node. This score, f(n), is the combination of two scores:

$$f(n) = g(n) + h(n)$$

Where g(n) is the cost of the path from the starting node to any node n, h(n) is the heuristic estimated cost from any node n to the goal.

Selectthe node with the lowest score. Use a Priority Queue to build the OpenList, and the nodes that will be added to Open List is sorted by this score. So whenpop an element from this Queue, Always get the node with the lowest score.There are some well-known heuristics used in scoring.

Calculate each node's score by adding the cost of getting there from the starting location to the heuristic value, which is an estimate of the cost of getting from the given node to the final destination. Use this score when determining which tile to check next from the Open List. Check the tiles with the lowest cost. In this case, a lower cost will equate to a shorter path.

The g(n) value shown in each open node is the cost of getting there from the starting node. In this case, each value is 1 because each node is just one step from the starting node. The h(n) value is the heuristic. The heuristic is an estimate of the number ofsteps from the given node to the destination node. Do not take obstacles intoconsideration when determining the heuristic. Do not examined the nodes between the current node and the destination node, so do not really know yet if they contain any obstacles. At this point simply want to determine the cost, assuming that there are no obstacles. The final value isf(n), which is the sum of g(n) and h(n). This is the cost of the node. It represents the known cost of getting there from the starting point and an estimate of the remaining cost to get to the destination.

Heuristics used in A* algorithm have some variations, but for grid based maps there are three heuristic functions that work well.

2.4.3.1 Manhattan Distance

The standard heuristic for a square grid is the Manhattan distance. Look at cost function and find the minimum cost D for moving from one space to an adjacent

space. In the simple case, set D to be 1. The heuristic on a square grid where can move in 4 directions should be D times the Manhattan distance:

```
function heuristic(node) =
    dx = abs(node.x - goal.x)
    dy = abs(node.y - goal.y)
    return D * (dx + dy)
```

Where D represents the cost of moving one node to one of its neighbours.

Manhattan Distance (Figure 2.21) allows to move in four directions (North, South, East and West).

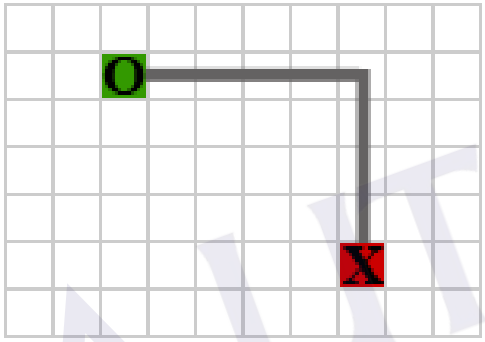

Figure 2.21: Manhattan distance calculation

The Manhattan heuristic is computed by adding the differences in the x and y components together. The advantage of using this heuristic is that, it is computationally inexpensive, and it can run faster than the others. The major disadvantage of the Manhattan heuristic is that it tends to overestimate the actual minimum cost to the goal (unless 4-adjacency is used) which means that the road being found may not be an optimal solution. If are not interested in an optimal solution, but just a good one, then using an overestimating heuristic can speed up theroad finding.

REFERENCES

Adam A. Razavian, Sun J. (2005). Cognitive Based Adaptive Path Planning Algorithmfor Autonomous Robotic Vehicles, *Southeast Con 2005 Proceedings, 8-10*.

Amemiya T., Yamashita J., Hirota K. and Hirose M. (2004). Virtual leading blocks for thedeaf-visually impaired: a real-time way-finder by verbal-nonverbal hybridinterface and high-density RFID tag space.

Benaicha Ramzi, Taibi Mahmoud. (2013).Dijkstra Algorithm Implementation On Fpga Card For Telecom Calculations.

Benjamin Chong Min Fui.(2012). A Comparative Study Of Maze Solving Algorithm For An Autonomous Mobile Robot.

Busra Ozdenizci, Kerem , Vedat Coskun, Mehmet N. Aydin. (2011). "Development of an Indoor Navigation System Using NFC Technology ".

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). Single-source shortest Paths :Introduction to algorithms. *2nd ed. Cambridge, MA: MIT Press,581-635.*

Carsten J. (2007).Global Path Planning on board the Mars Exploration Rovers. *IEEE Aerospace Conference.*

David M. Bourg, Seeman G.(2004). AI for Game Developers, O'Reilly, Chapter 7.

Edward M. Measure, David Knapp, Terry Jameson, and Andrew Butler. (2009). Automated Routing of U\nmanned Aircraft Systems (UAS).

Hart P. E., Nilsson N. J., Raphael B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths.

Jacob R., Marathe M. and Nagel K. (1999). A Computational Study of Routing Algorithms for Realistic Transportation Networks, ACM Journal of ExperimentaAlgorithms Vol 4 No 6.

Koenig S. and Likhachev M. (2002). Improved Fast Replanning for Robot Navigation in Unknown Terrain. *Proceedings of the 2002 IEEE, International Conference on Robotics & Automation,Washington DC.*

Kulyukin V., Gharpure C., Nicholson J. and Pavithran S. (2004). RFID in Robot-Assisted Indoor Navigation for the Visually Impaired. Proceedings Of lEEE/RSJ International Conference on Intelligent Robots and Systems.

Liang Dai. (2005). Fast Shortest Algorithm for Road Network and Implementation.

Leo Willyanto Santoso, Alexander Setiawan, Andre K. Prajogo. (2010).Performance Analysis of Dijkstra, A* and Ant Algorithm for Finding Optimal Path: Case Study Surabaya City Map.

M. Shafie Abd. Latiff, and R. Hassan.(2004). An Efficient Virtual Tour- A Merging of Path Planning and Optimization. Work with Computing Systems Conference, Kuala Lumpur.

Manh Hung V. Le , Dimitris Saragas , Nathan Webb.(2009). Indoor Navigation System for Handheld Devices", 2009.

Mustafa Kilinçarslan. (2007). Implementation Of A Path Finding Algorithm For The Navigation Of Visually Impaired People.

Neha Choubey and Bhupesh Kr.Gupta. (2013). Optimal route computation for online public transport enquiry system.

Obara T., Yamamoto K., Ura T., Maeda H., Yamato H. (1994). Development of an Autonomous Underwater Vehicle R1 with a Closed Cycle Diesel Engine, Proc.

Oliver J. Woodman. (2007). An introduction to inertial navigation.

Puthuparampil, M. (2007). Report Dijkstra's Algorithm [online]. Unpublished Presentation, Computer Science Department, New York University. Available from:http://www.cs.nyu.edu/courses/summer07/G22.2340001/Presentations/Puthuparampil.pdf. (October, 2014).

Ran L., Helal S. and Moore S., Drishti. (2004): An Integrated Indoor/Outdoor visually impaired Navigation System and Service. In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PerCom.04), pp.23-30.

R.Anbuselvi , R.S.Bhuvaneswaran. (2009). simulation of path finding algorithm a Bird's Eye perspective.

Rahul Kala· Anupam Shukla · Ritu Tiwarihave. (2010).Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning.

Sadeghi-Niaraki, A., Varshosaz, M., Kim, K., and Jung, J. (2011). Real world representation of a road network for route planning in GIS. Expert Systems with Applications, 38 (10), 11999-12008.

T. Frohlich and D. Kullmann. (2002). Autonomous and Robust Navigation for SimulatedHumanoid Characters in Virtual Environments. Proceedings of the First International Symposium on Cyber Worlds (CW'02).

Xiao Cui & Hao Shi.(2011). Direction oriented pathfinding in video games.

Yahja A., Singh S., Stentz A. (2000). An efficient on-line path planner for outdoor mobile robots, Robotics and Autonomous Systems.

YU Yongling, ZONG Sisheng, SHI Jinfa. (2010). Dynamic Labeling Algorithm Design forElectronic Map.