Road Map Estimation by using the Single Source Shortest Path Algorithm (DIJKSTRA'S Algorithm)

Khin Thandar Tun
University of Computer, Monywa, Myanmar
thandar.monywa @gmail.com

Abstract

Traffic congestion is becoming a serious problem in more and more modern cities. Encouraging more private-vehicle drivers to use public transportation is one of the most effective and economical ways to reduce the ever increasing congestion problem on the roads. To make public transport services more attractive and competitive, providing travellers with individual travel advice for journeys becomes crucial. However, with the massive and complex network of a modern division, finding one or several suitable route(s) according to user preferences from one place to another is not a simple task. This paper focuses on the uses of the Dijkstra's algorithm to find the shortest path in the Sagaing Division. The implementation of the shortest path algorithm is presented by the undirected graphs. This paper is able to suggest unfamiliar public users to choose a route based on their preferences. Users can choose not only shorter route with more frequent station change but also longer route with less station change. This system can also calculate the solution path from the starting station to the intended station. Based on the solution path, the system suggests the shortest distance, minimum cost and minimum time of the paths for the user's convenience. In order to develop a utility, this paper has selected the Java Language.

Keywords: Dijkstra's algorithm, Graphs and Road Maps

1. Introduction

Roadway is an identifiable route, way or path, between two or more places. Roads are typically smoothed, paved or otherwise prepared to allow easy travel. Different type of path has distances, times and costs. Therefore, physical distance of two points is not enough to describe the path. Instead, a logical distance is used to represent the real distance as well as shortest path. In distributed algorithms, the computation of route is shared among the nodes with information exchanged between them as necessary. This paper describes a set of experiments, which use different levels of shortest path computations. Dijkstra's algorithm is strictly limited in its performances even if it uses multi-processor core. The Dijkstra's shortest path algorithm is the most commonly used to solve the single source shortest path problem today. Computing a shortest path from one node to another in an undirected graph is a very common task. Many techniques are known to speed up this algorithm heuristically, while optimality of the solution can still be guaranteed. This paper presents a brief overview of the application using the shortest path algorithm in the Sagaing Division of Myanmar consists of 14 provinces; or 7 states representing the areas of 7 main ethnic races and 7 divisions. But, in this paper, we study the paths of the cities and compute the shortest of the paths in the Sagaing Division. Early shortest path work has been done by Dijkstra, Bellman and Ford, Floyd and Warshall. Dijkstra's algorithm is used to find the shortest distance, the minimum time and the minimum cost of the path form the source city and the destination city. Shortest path is the path that is the least length between two vertices. It is the set of line has the shortest overall distance. The system exploits the undirected graphs in order to extract route in the given road map.

1.1. Path Finding Algorithms

There are many algorithms that are commonly known and used, ranging from simplex to complex, in order to be able to solve the path finding problem. The simplest approach is to walk directly towards the goal until met with any kind of obstacles. When met with any object, direction will be changed and it can be passed by tracing around the obstacle. This

type of algorithm is an example of the "visually impaired Search" since it does not know or have any information about the path. Some other examples of the visually impaired Search are Breadth-First Search, Dijkstra's Algorithm and Depth-First Search. There also exist some algorithms that plan the whole path before moving anywhere. Best-first algorithm expands nodes based on a heuristic estimate of the cost to the goal. Nodes, which are estimated to give the best cost, are expanded first. The most commonly used algorithm is A* algorithm, which is a combination of the Dijkstra algorithm and the best-first algorithm. The different algorithms work in different ways. The breadth-first search begins at the start node, and then examines all nodes one step away (here we have 4 different nodes for Manhattan distance calculation and 8 different nodes for Euclidean type of distance calculation), then all nodes two steps away, then three steps, and so on, until the goal node is found. This algorithm is guaranteed to find a shortest path as long as all nodes have a uniform cost.

The bi-directional breadth-first search is where two breadth-first searches are started simultaneously, one at the start and one at the goal, and they keep searching until there is a node that both searches have examined. The final path is the combination of the path from the start to the intersection node, and the path from the goal to the intersection node.

1.2. Shortest Path System

Shortest path algorithms are applied to automatically find directions between physical locations, such as driving directions on web mapping websites like Mapquest or GoogleMap. In graph theory, the shortest path problem is finding a path between two vertices (or nodes) such that the sum of the weights of its constituent edges is minimized. An example is finding the quickest way to get from one location to another on a road map; in this case, the vertices represent locations and the edges represent segments of road and are weighted by the time or cost needed to travel that segment. There are many systems to extract data from database. In some of them, many forms of user interfaces are used shortest path that is the least length between two vertices. It is the set of lines has the shortest overall distance. There are several different algorithms that find the shortest path between two vertices in a weighted graph. Early shortest path work has been done by Dijkstra, Bellman and Ford, Floyd and Warshall. Shortest path problems are inevitable in road maps applications as city in optional routings have to be found. As the traffic condition among a city changes from time and there are usually a huge amounts of requests occur at any moment. Any shortest path algorithm must examine each edge in the graph at least once, since any of the edges could be a shortest path.

1.2.1. Shortest Path

Shortest path is the path that is the least length between two vertices. It is the set of lines has the shortest overall distance.

1.3. Dijkstra's algorithm

Dijkstra's algorithm is called the single-source shortest path. It is also known as the single source shortest path problem. It computes length of the shortest path from the source to each of the remaining vertices in the graph. Dijkstra's algorithm solves the single-pair, single-source, and single-destination shortest path problems. Djikstra's algorithm solves the problem of finding the shortest path from a source to a destination. It turns out that one can find the shortest paths from a given source to all vertices in a graph in the same time. In fact, this algorithm can be used to deliver the set of edges connecting all vertices such that the sum of the edge lengths from the source to each node is minimized. Dijkstra's algorithm is used to find the shortest path from the source city and the destination city. This paper presents an algorithm discovered by the Dutch mathematician E. Dijkstra in 1959 [4]. Edsger Dijkstra has been one of the most forceful proponents of programming as a scientific discipline. He has made fundamental contributions to the areas of operating systems, including deadlock avoidance, programming languages. The algorithm shortest path, as first given by Edsger Dijkstra makes use of these observations to determine the length of the shortest path from V to all other vertices in G. The version we will describe solves this problem in undirected weighted graph where all the weights are positive. It is easy to adapt it to solve shortest path problems in directed graphs. In performing Dijkstra's algorithm, it is sometimes more convenient to keep track of labels of vertices in each step using a table instead of redrawing graph in each step. Many problems can be modeled using graphs with weights assigned to their edges. The problem of finding shortest path plays a central role in the design and analysis of networks shortest path

problems can be solve once an appropriate cost is assigned to each link, reflecting its available bandwidth, for example. The cost of a path is defined to be the sum of the costs of the edges in the path. There are various algorithms for finding the shortest path is the edge in a network that are characterized by a single non-negative additive metric. The most popular shortest path algorithm is Dijkstra's algorithm, which is used in Internet's Open Shortest Path First (OSPF) routing procedure.

In many applications, the system may wish only to find shortest paths form one vertex (the source). In fact, it may be described to find only the shortest path between two particular vertices, but there is no know algorithm for that problem that is more efficient in the worst case than the best single source algorithm (Dijkstra's algorithm). Moreover, if the system only wish to know to which there exists a path from the source, the problem is trivial and can solved by a number of algorithm with operate in once, and an edge graph.

Since any algorithm which does not "Look at'' all edges cannot be correct, it is not hard to believe that O (e) is the best we can hope for the Dijkstra's shortest path algorithm is the most commonly used to solve the single source shortest path problem today. For a graph G(V,E), where V is the set of vertices and E is the set of edges, the running time and finding a path between two vertices varies. The running time of Dijkstra's algorithm on a graph G=(V,E) is $O(V^2)$. The more the number of nodes in a road map increase, it's very difficult to compute in using Dijkstra's algorithm because it needs pervious calculation results. Dijkstra's algorithm is strictly limited in its performance even if it uses multi-processor core. Dijkstra's algorithm proceeds by finding the length of the shortest path from source vertex, the length of the shortest path from source to a second vertex, and so on, until the length of the shortest path from source to destination is found.

1.4. Graph

Many have proven to be an extremely useful tool for analyzing situations involving a set of elements in which various pairs of elements are related by some property. Many problems can be modeled with paths formed by traveling along the edges of graphs for instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model. Problems of efficiently planning routes for mail delivery, garbage pickup, diagnostics in computer networks, and so on, can be solved using models that involve paths in graphs. We begin by defining the basic terminology of graph theory that deals with paths. Graphs are used to solve problems in many fields. Graphs may be used to represent the highway structure of a state or country with vertices representing cities and edges representing sections of highway. Graphs are discrete structures consisting of vertices and edges that connect these vertices. Involving distances can be modeled by assigning distances between cities to the edges. Problems involving fares can be modeled by assigning fares to the edges. Graph that has a number assigned to each edge are called weighted graphs. Weighted graphs are used to modes computer networks. Involving weighted graphs ask for the circuit of the shortest total length that visits every vertex of a complete graph exactly once. An edge weight is also referred to an edge length or edge cost or edge time. The length of a path is defined to be the sum of the lengths of the edges on that path rather than the number of edges. The cost of a path is defined to be the sum of the costs of the edges in the path. The starting vertex of the path will be referred to as the source and the last vertex the destination. There are several different types of graphs that differ with respect to the kind and number of edges that can connect a pair of vertices. Problems in almost every conceivable discipline can be solved using graph models. This system will show how graphs can be used to solve many types of problems, such as computing the number of different combination of paths between two cities.

1.4.1. Undirected Graph

Undirected graph is a graph that consisting of vertices that represents the computers, towns and undirected edges that represent telephone lines, networks line, road map, where each edge connects two distinct vertices and no two edges connect the same pairs of vertices. Many problems can be modeled using graphs with weights assigned to their edges. As the following example, we can set up the undirected graph model by representing cities by vertices and road by edges. Problems involving distances can be modeled by assigning distances between cities to the edges [4].

2. Related Work

M. Tommiska and J. Skytt reviewed the shortest-path algorithm which directly or indirectly informs their work. They first introduced Dijkstra's shortest path algorithm, the fundamental algorithm for other

SP algorithm for extended problems. They then introduced several approaches to solving the problems. This algorithm is applied point-to-point shortest path calculation on Road networks [3], geographical information system(GIS), telecommunication networks and Reconfigurable Hardware[2], Public Transport Information system[1] and so on.

Ulrich presented an efficient algorithm for fast and exact calculation of shortest paths in graphs with geometrical information in nodes (co-ordinates), e.g. road networks. The method was based on preprocessing and therefore best suited for static graphs, i.e., graphs with fixed topology and edge costs. In the preprocessing phase, the network was divided into regions and edge flags are calculated that indicate whether an edge belongs to a shortest path into a given region. In the path calculation step, only those edges need to be investigated that appropriate flag. They compared this method to a classical Dijkstra's implementation using USA road networks with travel times and report on speedup, preprocessing time, and memory needed to store edge flags [5].

3. Characteristics of the algorithm

This algorithm is computed the shortest path of the road map. But this algorithm is applied to find the shortest path of the airline or the networking paths. The problem with Dijkstra's algorithm is that vertices are selected in increasing distance from the source, a task that is at least as hard as sorting *n* numbers.

Purpose is single source shortest path problem: the shortest path between two points or all pairs shortest path problem: the shortest paths between one point and all vertices and computation.

3.1. Dijkstra's algorithm analysis

The system is used the Dijkstra's algorithm. The algorithm is the following:

begin

1. $S \leftarrow \{v_0\}$
2. $D[v_0] \leftarrow 0$;
3. *for each* v *in* $V \leftarrow \{v_0\}$ *do* $D[v] \leftarrow l(v_0, v)$
4. *while* $S \neq V$ *do*

   begin
5. *choose a vertex* w *in* $V - S$ *such that* $D[w]$ *is a minimum*;
6. *add* w *to* S;
7. *for each* v *in* $V - S$ *do*
8. $D[v] \leftarrow MIN(D[v], D[w] + l(w,v))$

end

end

S = the set such that the shortest path from the source of each vertex in S lies wholly in S

v_0 = a source vertex

v = each vertex in V, the minimum over all paths from v_0 to v

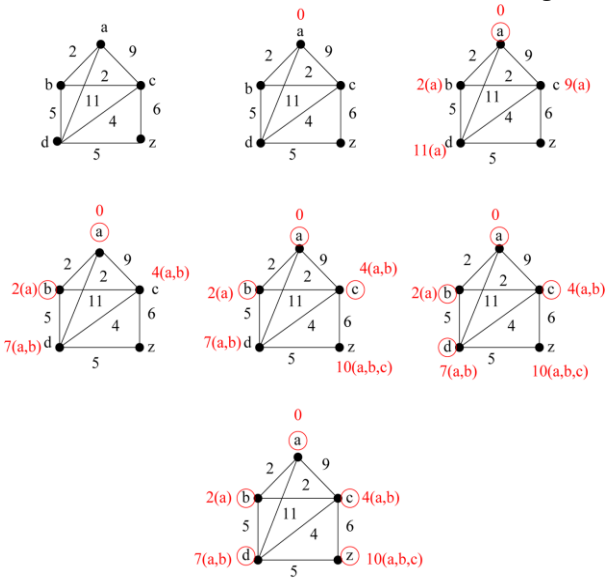
D[v] = the length of the shortest path from v_0 to v that lies wholly within S except for v itself

This algorithm computes the distance of the shortest path from the given source to each vertex and the desire source and destination.

The line 1 proves the size of the S that each v in S, D[v] is equal to the length of a shortest path from v_0 to v. The shortest path from v_0 to itself has length 0 and path from v_0 to v, wholly within S except for v, consists of the single edge (v_0, v). Thus line 2 and 3 correctly initialize the D array. Suppose vertex w is chosen on line 5. If D[w] is not the length of a shortest path from v_0 to w, then there must exit a shorter path P. The path P must contain some vertex other than w which is not in S. In line 6, the w add to the S. Let v be the first such vertex on P. But then the distance from v_0 to v is shorter than D[w] and moreover path to v lies wholly within S, except for v itself. In line 7 and 8, each vertex v in S is compared in the minimum function with D[v] and D[w] +1(w,v).

3.2. Sample of the Minimum Cost Estimation

Dijkstra's algorithm is a shortest path algorithm, meaning that it finds the length of the shortest path from a to a first vertex, the length of a shortest path from a to a second vertex, and so on until the length



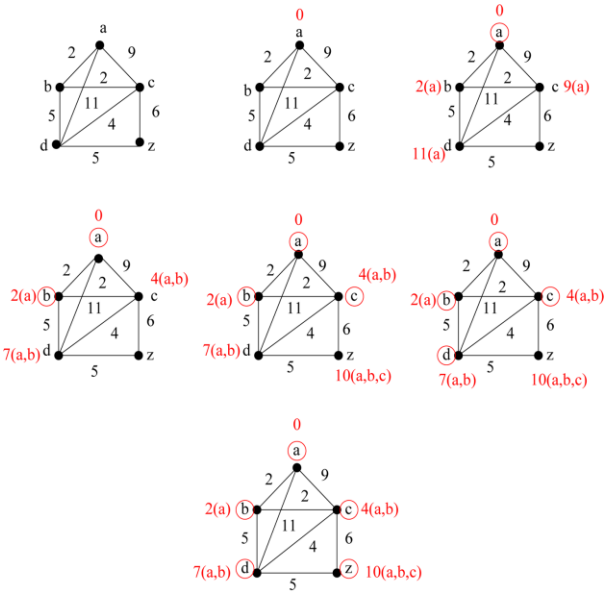


Figure 1. Minimum cost estimation

Dijkstra's Algorithm will be used to find the path of lowest cost from vertex a to vertex z in the weighted graph shown in Figure 1. While the graph is small enough so that the path of lowest-cost could be found simply by examining it, it was chosen this size for the purpose of the example. The figure shows the steps used by Dijkstra's Algorithm to find the lowest cost between a and z. At each iteration, the vertices of the distinguished set S_k, which is the set of vertices that have been chosen, are circled. The lowest cost path from a to each vertex containing only vertices from S_k (but not necessarily all of the vertices) is indicated for each iteration. The algorithm then terminates when z is circled. We will show that the path with the lowest cost from a to z is a; b; c; z with a cost of 10.

First, vertex a is labeled with 0. This is because at the 0th iteration the cost of the path from a to itself is 0, and so S0 = a.

Next look at the cost of all the paths from a to another vertex via a direct path. There is the path from a to b with a cost of 2, a to c with a cost of 9, and a to d with a cost of 11. Since the minimum cost is 2 along the path (a, b), vertex b is now circled and labeled with 2(a), and the distinguished set of vertices available for the next iteration is S1 = (a, b). Continuing on, this system will find the path of minimum cost from a to another vertex. Any path can be chosen containing one or more of the vertices from the distinguished set S1 and extending to a new vertex. One of the following two paths will be chosen: (a, b, c) with a cost of 4, or (a, d) with a cost of 11. Since the minimum is 4 along the path of (a, b, c) vertex c is circled and labeled with 4(a, b).

For the third iteration, a path starting at a, containing any of the following vertices, S2 = (a, b, c), and then extending to a new vertex will be chosen. There is the path (a, b, d) with a cost of 7, or the path (a, b, c, z) with a cost of 10. The path with a cost of 7 is chosen, so vertex d is circled and then labeled with 7(a, b).

For the fourth iteration there is only one more vertex to add, which is z. After examining the costs of all the different paths containing the vertices within the set S3 = (a, b, c, d) that extends to z, one can see that the path with the minimum cost is (a, b, c, z) with a cost of 10. Thus, in the final step, vertex z is circled and labeled with 10(a, b, c).

Hence it has been shown that the path from a to z with the minimum cost in this weighted graph is (a, b, c, z) of a cost of 10.

4. Proposed System design

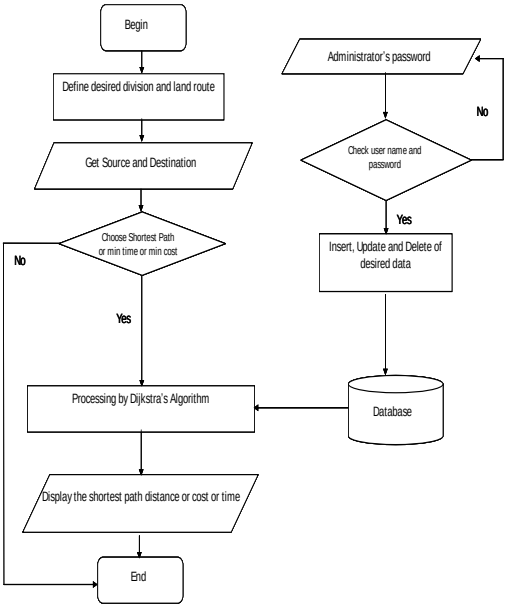


Figure 2. System flow diagram for the route map extraction

The system flow diagram is displayed the process of the system. User must define the desired division and land route. User can enter the source and destination cities. User can choose shortest distance or minimum cost or minimum time. The system will process the user desired and display results. Administrator is held the system by giving the administrator's password. If the password is false, the administrator is re-entered the password. If the

password is true, the administrator is inserted the new node (vertex) or edit the node and distance or delete the unwanted node or edit cost.

5. Implementation

5.1. Finding the shortest path

User can search by clicking the "Search Path" button. Then the user can choose the "Shortest Path" option or "Shortest Time" option or "Minimum cost" option. If the user selected the "Shortest Path" option, the system will display the shortest path with total cost and estimated time. If the user selected the "Shortest Time" option, the system will display the minimum time with total cost and total distances. If the user selected the "Minimum Cost" option, the system will display the minimum cost with total distance and estimated time.

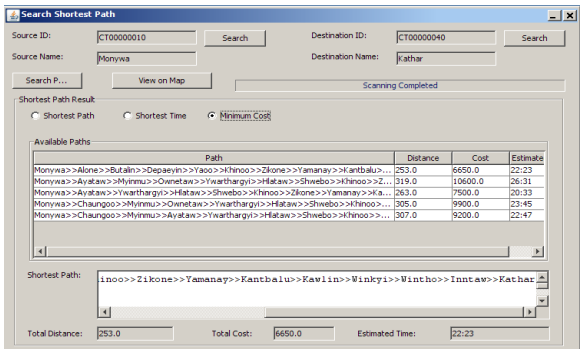


Figure 3. Display the minimum cost with total distance and estimated time

5.2. Displaying the Road Map

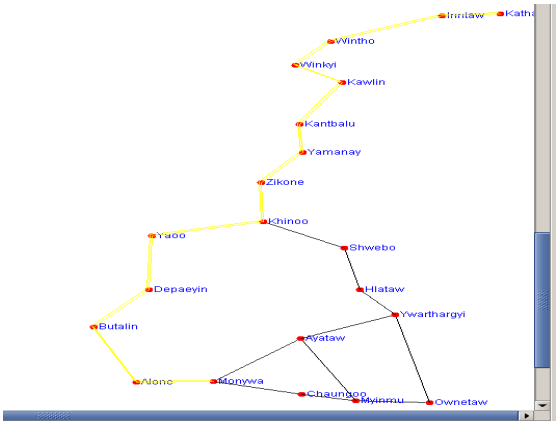


Figure 4. Display the shortest path from monywa to kathar with yellow path

If the user clicked the "View on Map" button, the possible paths will display. Then the shortest path will display with yellow path from the source city to the destination city.

6. Conclusion

The feasibility of using Dijkstra's algorithms to accommodate users' preferences for a public transportation network is discussed in this paper. The developed Dijkstra's algorithm is ideally expected to find a set of ranked shortest paths which can accommodate users' preferences easily and without many infeasible routes. This paper intends to deal with the difficulties faced when the user wants to visit the desired cities in the given road map. The Dijkstra's algorithm can be used to extract the desired route from a given road map. The undirected graph is used with the shortest path algorithm to calculate the solution path. By using this path, the user can get a chance to choose the desired path. For the unfamiliar public users, the system can help choosing the path from many existing routes as they wish. Furthermore, this paper supports the traveling process by giving the shortest distance, minimum cost and minimum time between desired source and destination city.

7. References

[1] A.J. Kelner and M. Brand, "Qtochastic Shortest Paths via Puasi-Bonvex Maximization Evdojia Nikolova", Cambridee MA, 2001.

[2] G. Seeman, M.D. Bourg, "*AI for Game Developers*", O'Reilly, Chapter 7, June 2004.

[3] J. Skytt and M. Tommiska,"Dijkstra's Shortest Path Routing Algorithm in Reconfigurable Hardware", 1995.

[4] R. Bellman,"*On A Routing Problem Quarterly of Applied Mathematics*", 1958.

[5] U. Lauther, "An Experimental Evaluation of Point-To-Point Shortest Path Calculation on Road networks with Precalculated Edge-Flags", 1998.