An Optimistic Routing Based Web Localization System

Aye Aye Mar, Nang Soe Soe Aung
Computer University (Lashio)
ayeayemar02@gmail.com

Abstract

Dijkstra algorithm among the shortest-path algorithms is to solve the shortest-path problem. Dijkstra's algorithm is probably the best-known and thus most implemented shortest path algorithm. It is based on a priority queue data structure. The "World Wide Web" is a collection of interconnected documents with images and other information stored on computers around the globe. Internet and Web Information Systems (WWW) is an international, archival, peer-reviewed journal that covers all aspects of the Web, including issues related to architectures, applications, Internet and Web information systems, and communities. This system searches a shortest path from the start point to the goal points by creating any nodes and searches shortest path from single south nodes to all nodes by using Dijkstra algorithm. This system intends to find the shortest-path and updates information and calculates shortest path again.

1. Introduction

Transit network design can be divided into two main groups, corresponding to the two following different approaches: (a) analytical models supplying a synthetic analysis of network performances and (b) optimization procedures for solving the network design problem by determining transit routes and associated frequencies and locating main transit centers Since it is well known that transit network design is a non-convex, problem solution methods are based on heuristic procedures. Although each method has peculiar characteristics, almost all of them apply the following steps :(1) building of a set of lines by following a given criterion; (2) development of the base network; (3) selection of a feasible sub-optimal solution; (4) improvement of the solution. This system represents the heuristic procedures for searching the optimal solution. For small networks and in the main important works, the selection of the best routes is done by deterministic methodology guided by rules and criterion. . As computer science and technology has progressed, computer is used in various fields like medicine, research, space, etc. The application of evolutionary computation techniques for the solution of optimization problems is now the major area of research. Optimization problems reveal the fact that the formulation of engineering design problems involves linear terms for constraints and objective function but certain other problems involve nonlinear terms for them. In some problem, the terms are not explicit functions of the design variables. Unfortunately, there does not exit a single optimization algorithm which will work in all optimization problems equally efficiently. Some algorithms perform better on one problem, but may perform poorly on other problems. That is the optimization literature contains a large number of algorithms, each suitable to solve a particular type of problem. Algorithms are very useful in real life applications. The choice of a suitable algorithm for an optimization problem is, to a large extent, dependent on the user's experience in solving similar problems. Efficiency is a pervasive theme throughout this area.

1.1 Related Work

P.Biswas, P.K.Mishra and N.C.Mahanti showed to perform an empirical study of the practical computational efficiency of the class of shortest path algorithms based on re-distribute heaps. They study documents that the re-distributive heap does not emerge as the best method in terms of actual computation efficiency in spite of the fact that it displays the best theoretical efficiency [4].N.Jing, Y. W. Huang, and A.Elke proposed the HEPV (Hierarchical Encoded Path View) approach that addresses those issues while guaranteeing path optimality. This solution interesting in efficiently determining the next link for the desired path rather than the complete path all at once [3]. Edward

P.F.Chan & Ning Zhang described a disk-based algorithm for finding shortest paths in a large network system. To show that it is scalable to large network systems and is adaptable to different computing environment, seven states in Tiger/ Line files are extracted as test cases and are experimented on machines with different configurations. The running time for finding the shortest path depends primarily on the power of the power of the underlying system [2]. Galen ***C.*** Hunt ', Maged M. Michael *, Srinivasan Parthasarathy I , Michael L. Scott showed a new algorithm that uses multiple mutual exclusion locks to allow consistent concurrent access to array-based priority queue heaps. They described that the new algorithm provided reasonable performance on small heaps, and significantly superior performance on large heaps under high levels of contention [5].

2. Shortest Path

A shortest-path involves a weighted, possibly directed graph described by a set of edges and vertices. Is given a start University, the goal is to find the shortest existing path between the start University and any of the other University in the graph. Each path, therefore, will have the minimum possible sum of its component edges' weights. Formally, one tends to think of as the 'length' of an edge is known as its 'weight'. Thus, a graph whose edges are all of equal length is unweighted, whereas a graph with edges of differing lengths is weighted. The term 'weight' is used because graphs are not limited to representing locations on a plane or in space; consequently edges could represent time, cost, and so on, generally a quantity which is to be kept minimal when going from any University to another. There are many techniques for solving the shortest path problem; some of the better known algorithms are Dijkstra's and Bellman-Ford. So, this system implements the Dijkstra algorithm for search routing of Computer Universities network topology

3. Routing Algorithm and Routing Table

Shortest path problems are popular among the most studied network flow optimization problems, with interesting applications in various fields. In shortest-path routing algorithms, the links distances of Computer Universities are decided based on the particular routing metrics (eg. delay, queue length, throughput, and hops). The arrangement and connectivity of nodes and links of a network is referred to as its topology. Routing algorithm makes decision that determines the route taken by each packet as it travels through the University.

Routing algorithm dynamically update its knowledge of the network based on a traffic monitoring and the detection of configuration changes or failures. The routing table contains a single entry for each possible destination, showing the next University that a packet must take towards its destination.

The shortest path problem is a fundamental problem with numerous applications. The shortest-path problem involves a weighted, possibly directed graph described by the set of edges and vertices. The constrained shortest path problem has attracted considerable attention from different research communities: operations research, computer science, and telecommunications. A star topology is designed with each University connected directly to a central network. Star topology is used each University connect because each device is inherently isolated by the link that connects it to the hub.

3. 1 Dijkstra Algorithm

An algorithm solves the single-source shortest path problem. The distance of a vertex *v* from a vertex *s* is the length of a shortest path between *s* and *v*. Dijkstra's algorithm computes the distances of all vertices from a given start vertex *s* [1].

Assumptions

- the graph is connected, the edges are directed
- The edge weights are non-negative

```
Dijkstra (G, w, s)   {
    for (each u Є V)
        d[u] = ∞;
    d[s]  =  0  ;
    pred[s] = nil;
    Q  = (queue with all vertices)
    While (Non-Empty (Q)) {
        u = Extract-Min (Q)
          for (each v  Є Adj[u] )  {
            if (d[u] + w (u, v) < d[v]) {
         [v] = d[u] + w (u, v);
              Decrease-Key (Q, v, d[v]);
            pred[v]  = u  ;       }}}}
```

4. Implementing Dijkstra Algorithm

The system implement with three phases. The first phase is finding the shortest path using Dijkstra Algorithm, the second phase is system design and the third phase is system implementation. In first phase Consider the diagram shown in Figure 1, Let G = (V,

E) be a weight digraph for Figure 1, that is, a set V of vertices, a set E of edges, and a real-valued weight function w : E→ R. If e = (u, v), write w (u, v) for w (e), that is, closest known vertex, u, and applying to all of its neighbors, v. Let Lashio University=S, Taunggyi University=A, Mandalay University=B, Mikhtila University=C and Magway University=D and so on.

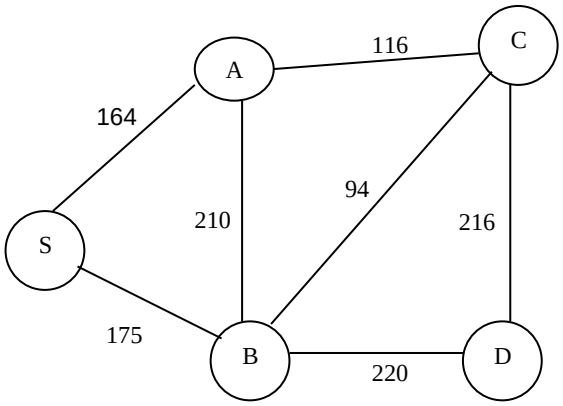

Figure 1. Sample graph

The length of a path p = < v_0 , v_1 , v_2 , …, v_k > is the sum of the weights of its constituent edges:

$$Length(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) \quad \text{Eq---------- (1)}$$

The distance from u to v, denoted d (u, v), is the length of the minimum length path if there is a path from u to v; is ∞ otherwise. Consider the problem of finding the shortest path between nodes S and D in the graph, maintain an estimate d[v] of the length d(s, v) of the shortest path for each vertex v. As shown in Table 1 and Table 2, initially d[s] =0 and all the other d[v] values are set to ∞. The dijkstra algorithm will process the vertices one by one in some order. The lengths of vertices are always d[v] > d (s, v) and d[v] equals to the length of a known path. As shown in Table 3 and Table 4, the algorithm picks the closest known vertex, A and B as Adj[s] = {A, B}, work on A and B and update information。The predecessor pointer pred [ ] is S for A vertex and B vertex. The distance d(S, A) is inserted into the priority queue and the distance d(S, B) into the priority queue.

Table1. Calculation for initialization

v	S	A	B	C	D
d[v]	0	α	α	α	α
Pred[v]	nil	nil	nil	nil	nil

Table 2. Priority queue

v	S	A	B	C	D
d[v]	0	α	α	α	α

Table 3. Calculation for step-1

v	s	A	B	C	D
d[v]	0	164	175	α	α
Pred[v]	nil	nil	nil	nil	nil

Table 4. Priority queue

v	S	A	B	C	D
d[v]	0	164	175	α	α

After Step 1, A has the minimum key in the priority queue. The algorithm extracts A with **ExtractMin ( )** key and picks the closest known vertices as Adj [A] = {B, C}, work on B and C update information. And then vertex A is removed from the priority queue with **DecreaseKey ( )**.

After Step 2, B has the minimum key in the priority queue. The algorithm extracts B with **ExtractMin ( )** key and picks the closest known vertices as Adj [B] = {A, C, D}, work on A, C and D update information. And then vertex B is removed from the priority queue with **DecreaseKey ( )**. Consider the diagram shown in Table 5 and Table 6, Step 2 has this path from S to C with the edge (A, C) and then gets another path from S to C with d [B] + w (B, C). If d[B] + w(B,C)< d[C], Dijkstra algorithm is called relaxation procedure, and then replace the old path < S,A, C > with the new shorter path < S, A, B, C > . In this algorithm is replaced d[B] + w(B,C) into d[C] and replaced B into pred[A] .

After Step 3, C has the minimum key in the priority queue. The algorithm extracts C with **ExtractMin ( )** key and picks the closest known vertices as Adj[C] = {D}, work on D and update information. And then vertex C is removed from the priority queue with **DecreaseKey ( )**.

Table 5. Calculation for step-3

v	s	A	B	C	D
d[v]	0	164	157	269	395
Pred[v]	nil	nil	nil	nil	nil

Table 6. Priority queue

v	C	D
d[v]	269	395

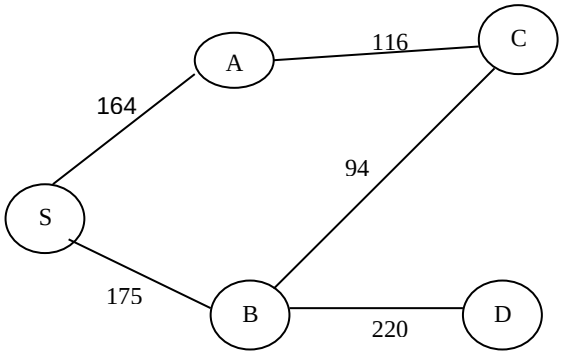

Figure 2. Shortest path tree for the system

After Step 4, D has the minimum key in the priority queue. The algorithm extracts D with **ExtractMin ( )** key and picks the closest known vertices as Adj [D] = {C}, work on D and update information. And then priority queue is empty, Q= Φ. The Dijksra algorithm are done shortest path as shown in Figure 4. The array pred[v] is used to build the shortest path tree. In Figure 1, there are many possible paths between S and D. They are SBD, SACD, SABD and SBCD such as, 395mile, 496mile, 594mile, and 485mile respectively. So the shortest path is 395 mile .The shortest paths of other nodes are described below Table 5.

4.1. System Design

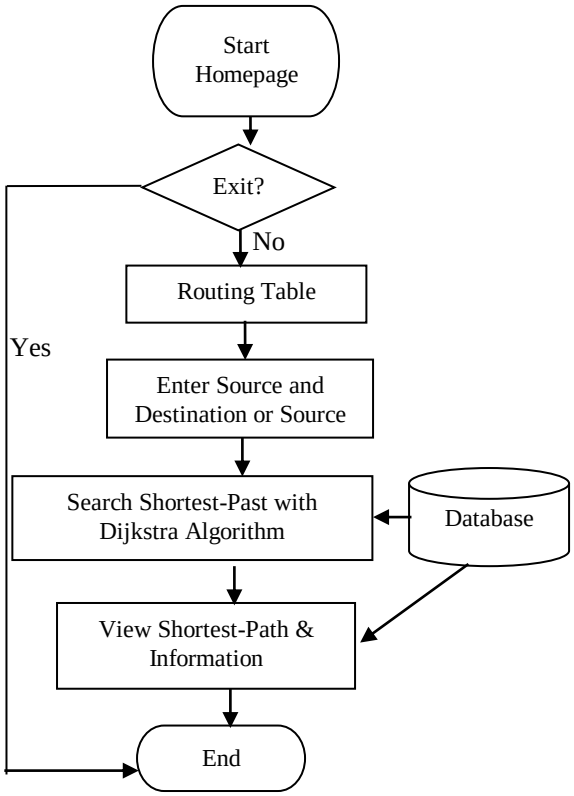

Figure 3.Overview of System Architecture

The heuristic transit network design method proposed here assumes as given the O/D matrix emend, the road network characteristics, and the vehicle capacity, as well as operating costs and users' costs. The basic framework of the model is established on the following three phases:

1. a heuristic algorithm to generate a set of feasible routes;
2. a genetic algorithm to find the optimal sub-set of routes with associated frequencies;
3. final improvement of the network configuration.

The figure shows overview of system architecture,. This system is implemented with three phases. The first phase shows the webpage of the system. The second phase show execution and creation of the network topology. The third is result of Routing table that describes shortest path and related information. Moreover, second phase needs to input source and destination. Therefore, the system executes by using Dijkstra Algorithm network design and outputs in the form of Routing table.

4.2. System Implementation

When the user starts the system, the main menu of the system will appear as in Figure 4.The main menu is Home page. Home page describes the detail

of the system. When the user clicks the Routing Table in the Home page the program will start.

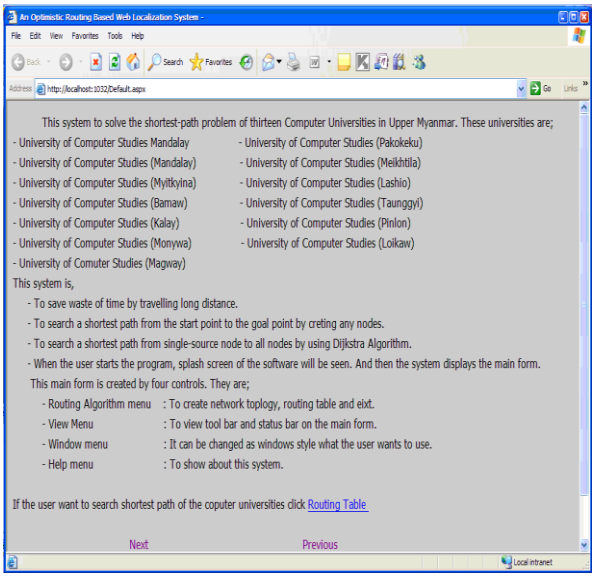

Figure 4. Home Page Form

When the user starts the program, splash screen of the software will be seen. And then the system displays the main form. This main form is created by four controls. They are; Routing Algorithm menu: To create network topology and routing table. View menu: To view tool bar and status bar on the main form. Window menu: It can be changed as windows style what the user wants to use. Help menu: To show about this system. In this section, the system allows the user to open Dijkstra Algorithm sub menu. In this system main menu form as shown in Figure 4.

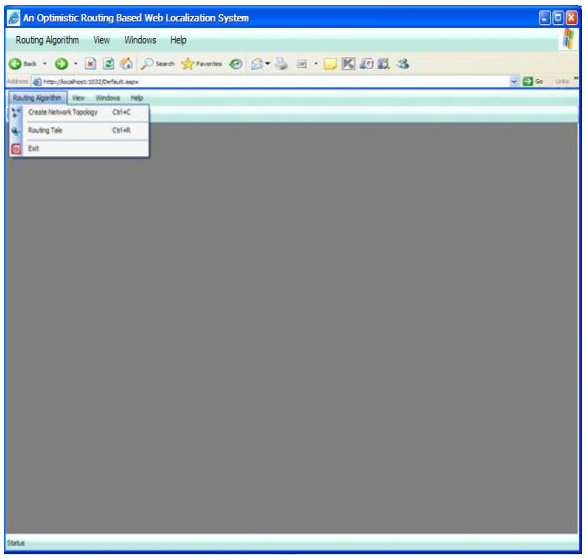

Figure 5. Main Menu Form

.In this system, there are two main potions. They are searching for routs between thirteen computer universities in upper Myanmar and creating network topology in them. In searching rout potion, a user can choose source and destination universities. The system searches the paths by using the Dijkstra algorithm and then displays network link of these universities and distances, routing table of network links, trees search view and then also displayed shortest path and its distance as shown in Figure 6.

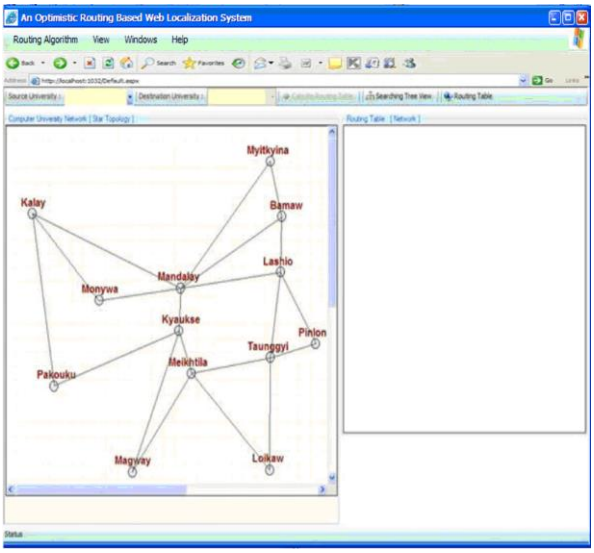

Figure 6. Routing Table Form

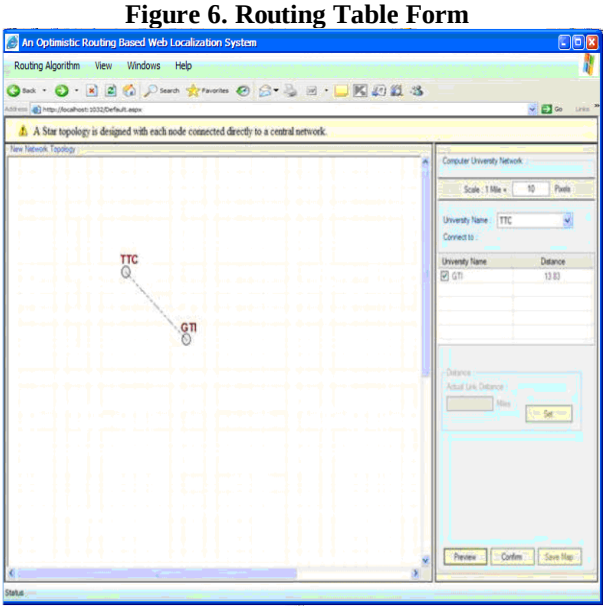

Figure 7. Create network topology

Figure 7 create network topology portion, a user can create the rebuild network topology in the system. The user can define scale rate pixel in mile, and then universities connect, the distance of the universities on the network topology. After that, the user can save the network topology. After saving the network

topology, this saving network uses to find the shortest path.

In this system calculating, system displayed about the information of routing table (To, Next Hop, Number of link and distance in miles) and shortest link and its distance. The searching result of the searching method and the straight lined distance can also be shown on its area. Then the optimal path of the map is also being displayed **as** shown in Figure 8.

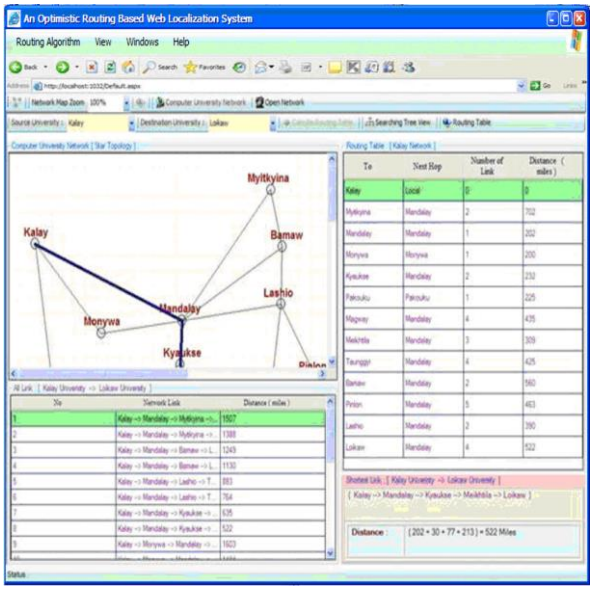

Figure 8. Dijkstra calculation form

5. Conclusions

The shortest path problem is to find a minimum-length (cost) path between a given pair. The shortest path problem involves a weighted and directed graph described by the set of edges and vertices. The goal is to find the shortest existing path between source vertex, s and any of the other vertices in the graph. This system solves the shortest path problem for any nodes track by using dijkstra algorithm. Dijkstra algorithm is frequently desired to find the shortest path between two nodes and than to finds the shortest path from a chosen source to a given destination. This system is single-source shortest paths problem. This system saves waste of time by traveling long distance and searches the shortest path for Upper Myanmar of Computer Universities.

6. References

[1] C. Mathieu, “Dijkstra’s shortest path algorithm**,** "CS157.

[2] P.F. Edward, N. Zhang, "Finding Shortest Paths in Large Network Systems,” Department of Computer Science University of Waterloo, May 3, 2001

[3] N.Jing, Y. W. Huang, A. Elke, "Hierarchical Optimization of Optimal Path Finding for Transportation Applications," in *Proc of the Conference on Information and Knowledge Management*, 1996. pp. 261-268**.**

[4] P. Biswas, P. K. Mishra and N. C. Mahanti, |"Computational Efficiency of Optimized Shortest Path Algorithms,” Department of Applied Mathematics, Birla Institute of Technology, Mesra (India*).*

[5] G.C. Hunt, M.M. Michael, S. Parthasarathy, and M.L. Scott. “An efficient algorithm for concurrent priority queue heaps，” In *Proc. Letters*, 60:151–157, 1996.