Journal of Information Engineering and Applications
ISSN 2224-5782 (print) ISSN 2225-0506 (online)
Vol.3, No.9, 2013

Shortest Route Finding using an Object-oriented Database Approach

Anwar M. A.
Al Ghurair University UAE
anwar@agu.ac.ae

Abstract
The huge and complex road networks in a country make it challenging to find the best route for roaming around. Therefore, it is necessary to arrange the data in such a way that its retrieval, especially a part of the road or road sub-network , should be efficient and needless data processing is avoided. This paper presents an object-oriented road network database model in which we divide the road into road segments and road network into road sub-networks, into levels according to the administrative in which a country is divided administratively. The implementation of the model for shortest route finding has proved that model is efficient and effective.
Keywords: Geographic Information Systems, Dijkstra's algorithm, Road Subdivision, Composite and Containment Hierarchies

1. Introduction

The management, representation and evaluation of spatial data in information systems are a complex task and have gained a lot of importance during the last decades. The Geographic Information Systems (GIS) are progressively being used in public administration, sciences and businesses. The core of the GIS is the geographic database system [1]. The standard database systems are built to meet the needs of the business applications and such systems are not suitable for geographic database applications. The insufficient expressive power of relational systems leads to unnatural data models and to inefficient query processing. Therefore, various research groups have developed a large numbers of concepts and techniques for improving single aspect of a geographic database. Example is the design of spatial data models for managing large sets of spatial objects.
In this paper, we will describe the modeling and implementation of an Object-oriented road network data model that will be advantageous for solving shortest path problems integrating several concepts and techniques. The route finding is known as the shortest path problem in the network theory. There are many algorithms such as Dijkstra's algorithm [2], usually used to solve this type of problems. These algorithms are general and are not efficient for route finding [3]. However, the limitation of these algorithms can be overcome by associating common sense knowledge about the geographic information of the road network and of the area in which route will be searched. The object-orientated concepts ion such as generalization, aggregation relationships and different hierarchical arrangements of the data are very much used full in associating such knowledge with the road network.
The paper is divided into (1) road network database design and (2) route finding in this road network database. In section 2 we discuss some shortest route finding algorithms and show why these algorithms need the different types of knowledge to compute shortest route. Section 3 describes the design of the Road Network Data Model (RNDM). Section 4 explains how route is searched in an Object-oriented Road Network Database and in section 5 we evaluate our approach and in section 6 related works is presented. In last we present conclusion of our work.

2. Shortest Path Algorithms

The shortest path algorithm is modeled as finding the shortest path between two nodes in weighted and directed network $\boldsymbol{G} = (\boldsymbol{N}, \boldsymbol{A})$, where $\boldsymbol{N}$ is a node set and $\boldsymbol{A}$ is an arc set as shown in Figure 1. A weight $\boldsymbol{l_{ij}} \geq \boldsymbol{0}$ is associated with each arc $(\boldsymbol{i}, \boldsymbol{j}) \in \boldsymbol{A}$ which may be the length of the arc.

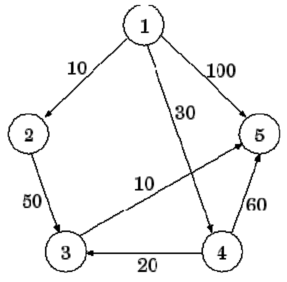

Figure 1: Weighted Graph

In the route finding context, the network is the road network, nodes are the road junctions and arcs are road segments. The length of each arc could be the distance or travelling time between two adjacent junctions. A review the two approaches for the shortest route find is presented in next subsection but it has been noted that neither of these approaches are suitable for route findings.

2.1. Dijkstra's Algorithm

The path finding problem has been an attractive research topic and got considerable attention in the past. There are many algorithms devised for its solution but the Dijkstra's algorithm [2] is perhaps the earliest and also one of the most effective algorithms for the shortest path finding problem. The other algorithms mostly are variations of the Dijkstra's algorithm. This algorithm can solve the route finding problem without any additional helping information. This algorithm solves the problem in $\boldsymbol{O}((|\boldsymbol{A}| + |\boldsymbol{N}|)\boldsymbol{log}|\boldsymbol{N}|)$ steps by using a binary heap [4], where A is the number of arcs and $|N|$ is the number of nodes in a network. The pseudo code of the algorithm is given below.

```
// Let v1 be the origin vertex and initialize W and ShortDist[u] as
  W := {v1}
  ShortDist[v1] :=0
  FOR each u in V - {v1}
    ShortDist[u] := T[v1,u]
// Now repeatedly enlarge W until W includes all vertices in V
  WHILE W <> V
// Find the vertex w in V - W at the minimum distance from v1
    MinDist := INFINITE
    FOR each v in V - W
      IF ShortDist[v] < MinDist
        MinDist = ShortDist[v]
        w := v
      END {if}
    END {for}
// Add w to W
    W := W U {w}
// Update the shortest distance to vertices in V - W
    FOR each u in V - W
      ShortDist[u] := Min(ShorDist[u],ShortDist[w] + T[w,u])
  END {while}
```

The Dijkstra's algorithm can solve route finding problems alone but in a complex road network with thousands of roads and cross points it will take a long time to find the shortest path for the algorithm will be time consuming in terms of computation because there no need to search through the whole network in order to find the solution. For example, if someone wants to find a route with source and destination nodes in a city then it is not necessary to search the whole road network and a search only in this city road sub-network will provide a solution. Another drawback of the Dijkstra's algorithm is that it can also produce the solutions which are not suitable for human users due the reason that shortest path may use many minor roads as part of the solution.

2.2. Knowledge-based Techniques

Knowledge-based problem solving [4] emphasizes the use of human problem solving strategies on a computer. It takes advantage of insight knowledge of the human on a particular problem solving domain, and tries to imitate the human problem solving process. This technology has been used to solve many real life problems.

The human beings know the knowledge about the road network and geographical knowledge about various places to isolate the part of the whole network that may contain solution. However, they are not good at searching for the best solution in the isolated area. Thus, it can be said that route finding by using human knowledge alone will not be efficient. It can efficiently isolate the part of the network containing the solution. After that, a search algorithm has to be used to find the best solution in the isolated area.

Therefore we can conclude that each individual technique does not provide a good technique for route finding, however, each of these has its advantages in solving the problems. Analysis of the advantages and disadvantages shows that they can easily help each other. In order to use the knowledge about road network and the geographical knowledge we use object-oriented data model and store all the required knowledge in objects as responsibilities of each object.

3. Designing the Road Network Data Model (RNDM)

The identification of classes, arranging them in different types of hierarchies and allocating data and behavior to the classes is an important task in object-oriented data model. In the following subsection we present these tasks.

3.1. Identification of Classes

The major classes in the RNDM are divided into (1) base classes and (2) concrete classes. A base class is an abstract class that is used to represent a template for a geographic object. It canNOt be instantiated. Typically, there will be subclasses of it which will inherit its properties down the hierarchy. A concrete class, on the other hand, can be instantiated with real world objects. The base class is: GeoObject. GeoObjects is a template for the geographic objects in the model.

The analysis of the requirement specifications of the shortest path problem shows that most of the queries pertaining to the RNDM would be asked with reference to certain well understood geographic objects in the model, such as country, prefecture, or city. The prefecture, and city etc., are the container geo-objects that contain other container and non-container geo-objects. Roads and rivers are the examples of non-container geo-objects. Thus the geographic objects in this application can be broadly divided into container geo-objects which contain other geographic objects and non-container geo-objects which don't contain any other geo-object. Due to this fact, the following two abstract classes are defined; ContainerGeoObject class is defined to represent container geo-objects in the model. In order to add real life container geographic objects like country, prefecture, and city in the model |mboxconcrete subclasses *Country, Prefecture, City, and Neighborhood* of the subclass ContainerGeoObject are defined. NonContainerGeoObject class is defined to represent the non-container objects in the model.

The class Road is an abstract subclass of the class NonContainerGeoObject that is used as a template for road objects in the model. In order to add real life road objects in the model concrete subclasses, *Expressway, CityExpressway, NationalRoad, PrefectureRoad, CityRoad, and MinorRoad* of the class Road have been defined. We also define a concrete subclass *RoadSegment* of the class Road. To add real life road segment we define three subclasses*PrefectureSegment, CitySegment, and NeighbourhoodSegment* of the class RoadSegment. The class hierarchy is shown in Figure 2. The subclass Node represents cross point of two roads, start/end point of a road, or cross point of boundary of a geographic object and road. The subclass Link represents the part of the road from node to node in this model. To represent it a class Link has been defined. Link in the road network data model may be a ramp, a bridge or a tunnel.

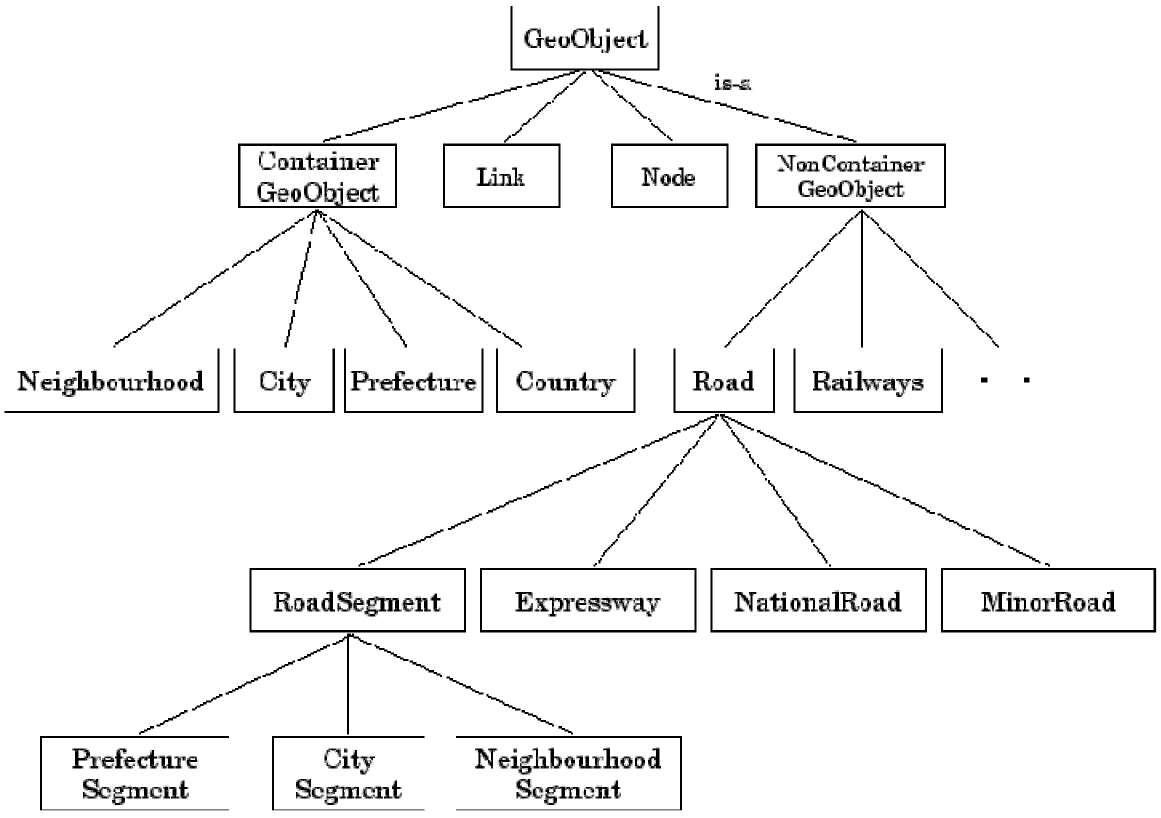

Figure 2: Class hierarchy

These geographic objects are arranged in a containment hierarchy with country containing prefectures, expressways, city expressways and national roads etc. and prefecture containing cities, prefecture roads, and prefecture segments of expressways, city expressways, and national roads. Similarly it can be explained down the hierarchy as shown in Figure 3. Due to the complexity of showing everything in one figure we used the word *group*. The *group* at country level means the group of the prefecture segments of expressways, city expressways, and national roads which make a road at country level. The *group* at prefecture level means the group of city segments of each road that makes a prefecture road or a prefecture segment of a road and so on. At the lowest level the *group* consists of the links that make the neighborhood road or neighborhood segment of any road.

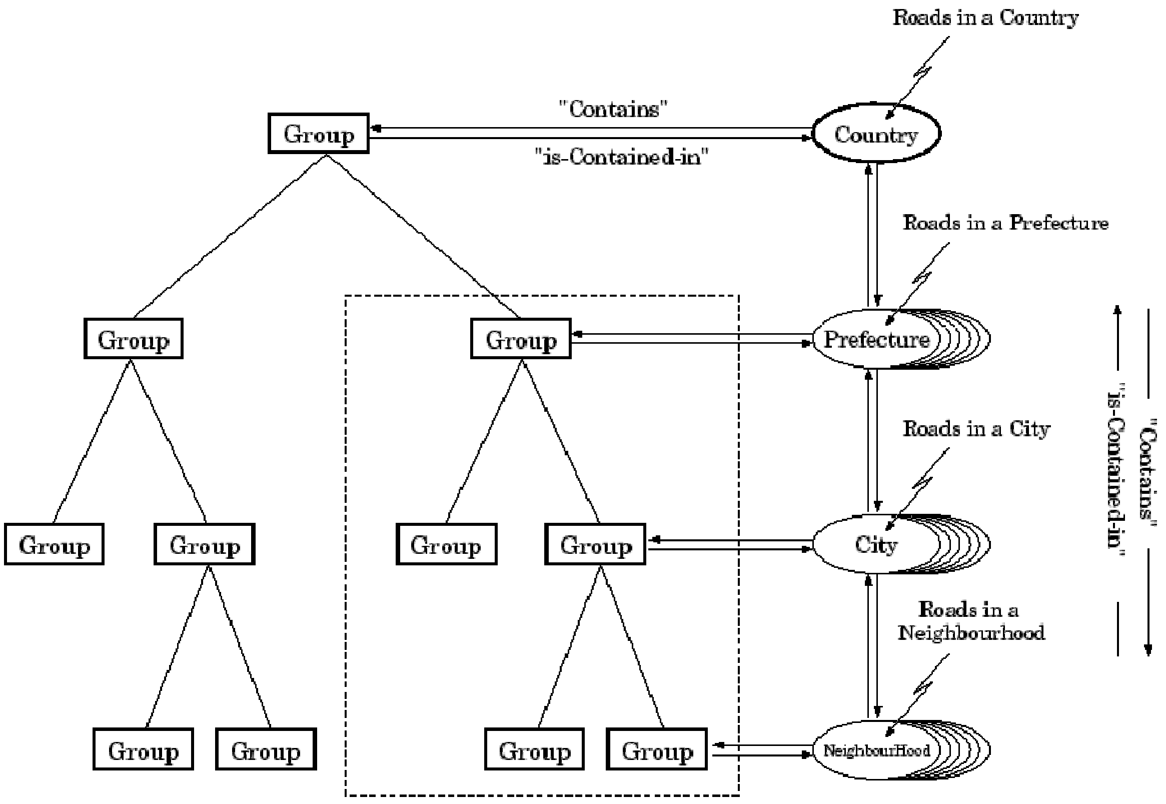

Figure 3: Containment hierarchy

At this stage in the design process it is noticed that non-container geo-objects can be divided into composite segments. For example, non-container geo-object such as, expressway is composed of prefecture segments of the expressway which are contained entirely by prefectures. And similarly, prefecture segments and prefecture roads are composed of city segments which are contained entirely by cities as shown in Figure 4.

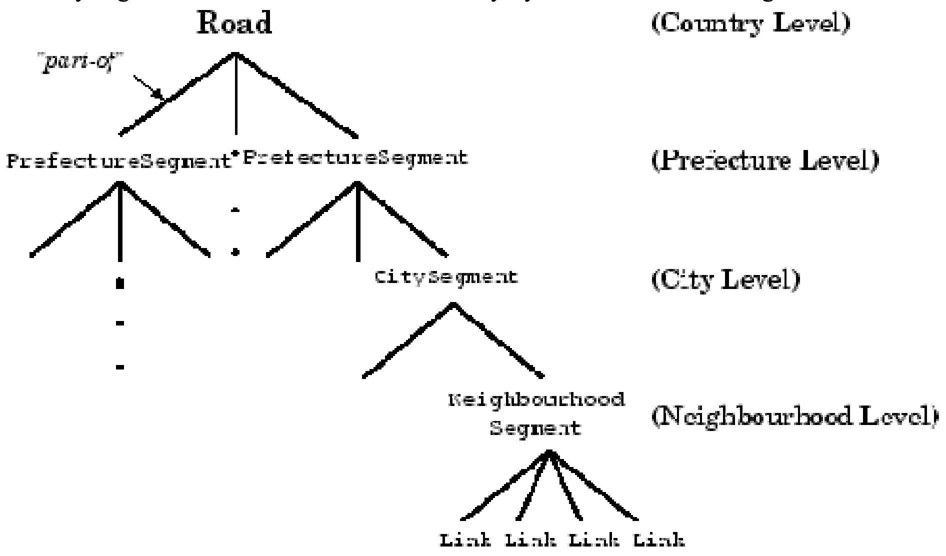

Figure 4: Composite hierarchy of Road

3.2. Significance of the Model's Components

We consider the classification of the geographic objects into container and non-container features as a semantically significant aspect of our model. While such a classification seems conceptually simple, it plays an important role in route finding problem. Under are the two characteristics of these classes:

- A container object at a particular level in the hierarchy can contain only one other container object and the container object must be of a different type than the container. Thus a prefecture directly contains cities and indirectly contains neighborhood, etc. However a prefecture does not contain another prefecture.
- A container object at a particular level in the hierarchy can contain one or more non-container objects, and the contained objects may be of different types of as long as they are not also contained by other container object. Thus, a prefecture contains a prefecture road and prefecture segment of highways, etc.

The hierarchy that results in corresponds to ``user perspective level''. Thus, a user can have ``country level'' perspective, ``prefecture level'' perspective, etc. Therefore the route finder system, first, will determine the user perspective level and then process according to that level.
One possible query that may be posed in route finder system is: ``Retrieve all the major roads in prefecture X.'' To process such a query, the processor can potentially check each link in road network and use some algorithm to decide whether it lies in prefecture X or not. This leads to inefficiency of the route finder system. Processing this type of query can be made more efficient, by using a geographic containment hierarchy in our model. Since each container object contains roads entirely contained in it and these contained roads have, also, been divided into different types therefore, the system can find the major roads in prefecture X by just invoking the method GetMajorRoads() defined in prefecture class. Hence there is no need to check each link to be in prefecture X and of type major road. This arrangement greatly reduces the computation time and is more efficient in terms of route finding performance.

4. Route Finding Algorithm

This section presents the problem solving method in the proposed approach. It consists of four main steps. Below is the algorithm.

```
Step 1: Get the type of the road on which source node (S) is located by invoking appropriate method
        in source node.
    if this type matches with the major road
        then goto step 2
    else
        1. Find the index number of the neighborhood in which (S) is located by invoking appropriate method
           in source node. ( we call this neighborhood as source neighborhood )
        2. Retrieve the road network in the source neighborhood.
Step 2: Get the type of the road on which destination node (D) is located by invoking appropriate method in
        destination node.
    if this type matches with the major road
        then goto step 3
    else
        1. Find the index number of the neighborhood in which (D) is located by invoking appropriate
           method in the destination node. (we call this neighborhood as destination neighborhood)
        2. Retrieve the road network in the destination neighborhood.
Step 3:
    if both, the source and the destination nodes, are on minor roads, then
            1. Find the name of the the container geo-object containing source neighborhood and destination
               neighbourhood.
            2. Retrieve the major road network in the container geo-object.
            3. goto step 4.
    else if both, the source and the destination nodes, are on major roads, then
            1. Find the name of the the container geo-object containing S and D.
            2. Retrieve the major road network in the container geo-object.
            3. goto step 4.
    else if one (S or D) of the source or destination nodes is on the minor road and the other is on major road,
    then
            1. Find the name of the container geo-object containing neighborhood that contains the node
               which is on the minor road, and the other node that is on major road.
            2. Retrieve the major road network in the container geo-object
Step 4: run the shortest path algorithm such as Dijkstra's algorithm, with this augmented road network.
```

Now, let us do analysis of the complexity of the algorithm. When the entire network is in the hard disk then road network in the source and the destination neighborhood is transferred into the main memory in a constant time.

(1) and (2) in step 1 and step 2 can be done in a constant of time because each node has an index number of the neighborhood containing it. (1) in step 3 is also done in a constant of time because each neighborhood or container geo-object also knows the name of its container geo-object. Retrieving road network in neighborhood and container geo-object is also done in a constant of time as we have designed our road network in such a way that it contains the road network at each level as a group of links. Main computation is done in step 4. Since the number of nodes and arcs (links) has been substantially reduced, it will run much faster.

Now, we will present a complete example of problem solving in our proposed approach. Suppose a user would like to go from source *S* to destination *D* shown in Figure 5. The road network (thin dotted lines are minor roads and thin solid lines are boundaries of container geo-objects) is retrieved in neighborhood containing *S* and *D* and major road network is retrieved in the geo-object containing source and destination neighborhood. And then this augmented road network with knowledge such as, peak hours, road construction, road facilities, etc., in the source and destination neighborhood and the geo-object containing these neighborhoods is used to search the shortest path between *S* and *D* i.e. *S – 3 – 2 – D' – D*.

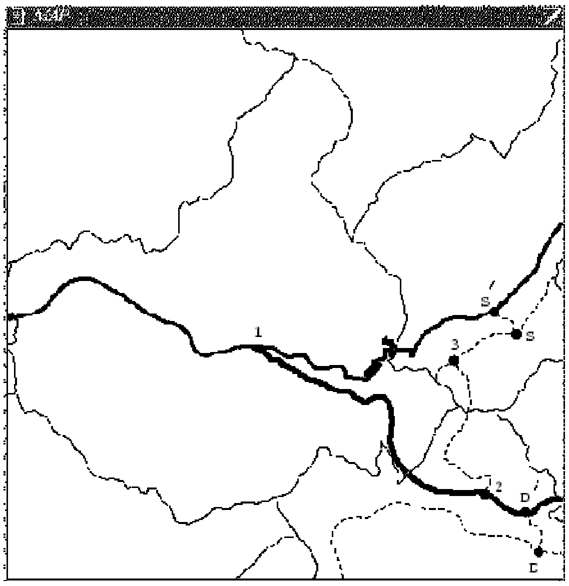

Figure 5: A Route Finding Example

5. Evaluation

The purpose of designing such road network data model is to provide a helpful base for solving the route finding problem. We believe we have succeeded. We argue in the following lines how it is supportive in resolving the route finding problems.

Firstly, we have divided the road network in a country into road sub-networks according to the administrative areas in which a country is divided. Any temporary information/caution about the road sub-network for example, repair of roads etc., in any administrative neighborhood can be put into that neighborhood enhancing the responsibilities of that neighborhood. This type of information will avoid not usable paths as a result. It also saves a great deal of system memory space because only the required neighborhood of interest with the required types of roads will be loaded into the memory from hard disk. This is also possible in case of source and destination points in different neighborhoods because only the road sub-networks in both of the neighborhoods and the major road network in the geo-object containing source and destination neighborhoods will be loaded into the main memory.

Secondly, we have divided each road into composite segments in such a way that every segment of any road provides all the information about the entire road. This help in finding the city name through which a road is passing from a road object. Furthermore, we can also associate other information such as road facilities, public areas etc., in the road objects. The peak hours of traffic on a road are also added in each of the road and it will be helpful when a driver does not prefer to go by such a road and can select an alternate path. All this was possible by using powerful object orientation concepts such as containment and composite hierarchy.

6. Related Work

Shortest path problem has been an active area of research in network theory over the past three decades and many algorithms have been proposed. The emphasis has been on using topographic and geographic knowledge and reasoning to solve the problem. The major concern in Artificial Intelligence is spatial knowledge representation and reasoning. Shapiro [6] proposed a technique to use level structure of the road network to help searching for a path efficiently. This is similar to our minor and major road structure and it can be extended to any number of levels. We will explain inefficiency of this algorithm by an example. Let us suppose that source and the destination nodes are on the minor roads. Then, according to the algorithm proposed in [6] will search for the shortest path to a node $\boldsymbol{S'}$ in the major road network from the source node $\boldsymbol{S}$. It will then do again like this for the destination node to find a node $\boldsymbol{D'}$ on the major road network. Then it uses Dijkstra's algorithm for shortest path between $\boldsymbol{S'}$ and $\boldsymbol{D'}$ in the major road network.

This technique minimizes the time travelling on minor roads and also reduces the search space. However, the problem is that the two nodes $\boldsymbol{S'}$ and $\boldsymbol{D'}$ could be on the wrong major roads. Then the path found will not be shortest path. The path searched by this technique for source $\boldsymbol{S}$ and destination $\boldsymbol{D}$ will be $\boldsymbol{S - S' - 1 - 2 - D' - D}$, but it is clear from the Figure 5 that it is a longer path than the shortest path $\boldsymbol{S - 3 - 2 - D' - D}$ searched by the technique presented in this paper and hence could not be considered as good solution. Our approach solved this problem by using the minor and major road networks in the source and the destination neighborhood and only the major road network in the container geo-object containing both the neighborhoods. There is no chance of landing on wrong major roads.

Another advantage of the approach presented in this paper is that it partitions the whole network into smaller sub-networks contained in prefectures, cities and neighborhoods. This is important for a large road network.

Another related AI-based system is proposed in [3] in which the major road network is not pruned off and therefore, after deciding the grid sub-networks the shortest path algorithm searches in all the major road networks with minor road networks in grid networks. But, in our approach we also prune off the search space in major road network and give a substantially reduced major road network to the shortest path algorithm for search. This part of the major road network may be in a city containing source and destination neighborhood or a prefecture or a country at the highest level.

7. Conclusion

In this paper we have proposed an approach to solve the route finding problem in based on object-oriented road network database. We designed an object-oriented road network database to test the efficacy of the approach by applying the approach to an actual, complex and real-life route finding problem. The hierarchical arrangement of the data enables us to have road network in a neighborhood at low level while in another at higher level and therefore we can run Dijkstra's algorithm in multi-level road networks of interest.

Acknowledgement

We would like to thank Dr. Takaichi Yoshida and Zenrin and Co. for providing the road network data in Fukuoka prefecture of Japan. Without it, this research would be impossible.

References

1. T. Brinkhoff, H. Horn, H-P. Kriegel, and R. Schneider. "A storage and access architecture for efficient query processing in spatial database systems", In SSD, pages 357 - 376, 1993.
2. http://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html (accessed on May 10, 2013)
3. B. Liu and J. Tay., "Using knowledge about the road network for route finding", In IEEE Transactions on Systems, Man and Cybernetics, volume 27, July 1997.
4. R. Sedgewick. "Algorithms", Addison-Wesely., 1988.
5. F. Hayes-Roth, D. A. Waterman, and D. B. Lenat., "Building Expert Systems", Addison-Wesely, 1983.
6. J. Shapiro, J. Waxman, and D. Nir., "Level graphs and approximate shortest path algorithm", In Networks, volume 22, pages 691-717, 1992.