THE JOURNAL OF NAVIGATION (2013), **66**, 435–447.
doi:10.1017/S0373463313000076

Adaptive Road Candidates Search Algorithm for Map Matching by Clustering Road Segments

Ming Ren and Hassan A. Karimi

(*Geoinformatics Laboratory, School of Information Sciences, University of Pittsburgh, USA*)
(E-mail: mng_ren@yahoo.com)

Map matching is an important algorithm for any location-based service, especially in navigation and tracking systems and services. Identifying the relevant road segments accurately and efficiently, given positioning data, is the first and most important step in any map matching algorithm. This paper proposes a new approach to searching for road candidates by clustering and then searching road segments through a constructed hierarchical clustering tree, rather than using indexing techniques to query segments within a fixed search window. A binary tree is created based on the hierarchical clustering tree and adaptive searches are conducted to identify candidate road segments given GPS positions. The approach was validated using road maps with different scales and various scenarios in which moving vehicles were located. Both theoretical analysis and experimental results confirm that the proposed approach can efficiently find candidate road segments for map matching.

KEY WORDS

1. Map matching. 2. Hierarchical clustering tree. 3. Binary tree. 4. Road candidates search.

Submitted: 5 July 2012. Accepted: 15 February 2013.

1. INTRODUCTION. Navigation and tracking systems have been a research topic for the past several years. Regardless of the underlying geo-positioning sensors (e.g., Global Positioning System (GPS) or Wireless Fidelity (Wi-Fi)) considered in these systems, they cannot provide actual locations of moving objects with changes in time, area, and weather. This is the reason why map matching is needed as the fundamental task in navigation and tracking systems to estimate moving objects (e.g., vehicles) using position data and spatial networks. The first step in a map matching algorithm involves identification of the correct road segment on which a vehicle is travelling. Thereafter, estimating the location of a vehicle on that segment is the second step in a map matching algorithm. Finding the correct road segment requires the identification of a set of candidate road segments, based on a received GPS position, followed by comparison of those candidates to decide the most likely segment. Therefore, finding the set of candidate road segments is imperative in map matching

algorithms and involves two types of situation. The first is initializing a search range when the first position data is received. The other is continuously updating the search window for new positions. The worst case, from a search standpoint, is that when the search window covers an entire road network. Searching a large network is time consuming, which may delay the process of map matching. Identifying the segment on which a vehicle is moving not only demands correctness, but demands efficiency as well. For navigation, which processes in real-time, efficiency in finding the correct segment is especially important.

A common approach to this first step in map matching is initially to reduce the search range and then to find the correct segment. The correct segment is among those that are close to the given GPS position. Given this assumption, search time is reduced by using a small portion (window) of a large network. All links within this window are treated as candidates for further analysis during the map matching process. A widely used process for creating search windows is to create a buffer, centred at a given GPS position, and identify the road segments within it as candidates. How to define the buffer window and how to quickly search those candidate links within the window has been a research topic amongst the navigation community.

In this paper, we present a new algorithm to address the problem of efficiently finding candidate road segments in road networks given the GPS positions of a moving vehicle. The algorithm is based on a bottom-up approach that builds a road segment-clustering tree to achieve both adaptability and efficiency in finding candidate segments.

The main innovations of this paper are: the introduction of a segment-clustering tree algorithm, which clusters segments by both geometry (distance) and topology (connectivity) in road networks, and an adaptive technique to create and update search windows for map matching in navigation systems. The proposed algorithm avoids the redundant searching that is inherent in conventional spatial indexing techniques.

The rest of the paper is structured as follows. Section 2 discusses background and related works. Section 3 introduces road segment-clustering trees and describes their data representation. Section 4 describes the query and update algorithms used to find relevant road segment candidates. Section 5 discusses performance analysis and experimental evaluation. Finally, conclusions and future research are discussed in Section 6.

2. BACKGROUND AND RELATED WORKS.

Current research into searching for objects in spatial networks mainly uses spatial indexing techniques with the goal of improving the efficiency of queries (Tele Atlas, 2008; Zhao et al., 2001). Spatial indexing for spatial and spatio-temporal queries has been an active research topic over the past decades. Since 1984, when the R-tree was presented by Guttman (1984), several indexing techniques have been developed for efficient spatial queries. Basic indexing techniques with variant structures such as the R-tree, quadtree, B-tree, and grid (Zhao, 1997) have been employed in different experiments focused on spatial query efficiency (Lin 2008a and 2008b; Chen et al., 2009; Kalashnikov et al., 2002). We focus on the R-tree as a representative technique and discuss data structures and query algorithms in indexing and searching spatial objects in spatial databases.

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

R-tree was developed as an index structure for the efficient management of multi-dimensional and spatial data. Common operations performed on an R-tree include point location queries, range queries and nearest neighbour queries. For map matching, road segments, among other object locations, such as buildings, can be represented as polygon objects and stored on the leaf nodes of an R-tree. Each leaf node holds two items for each data record: one is the bounding box of the object and the other is to place the object. Non-leaf nodes of an R-tree can hold two items for each of its children: a bounding box of the child and a pointer to the child.

A number of bulk loading techniques were developed to build an R-tree (Bercken and Seeger, 2001). The Top-down Greedy Split (TGS) and the bottom-up approaches are the two predominant methods in use (Alborzi et al., 2007). A bottom-up approach builds an R-tree from the leaf nodes level to the upper level until it reaches the root node. In the lowest level, n data rectangles are sorted according to a predetermined sort order and m data rectangles are grouped in the upper level. The construction process is iterative from the bottom to the root of the R-tree. By contrast, a top-down approach starts with building the higher levels of the R-tree. The data rectangles are sorted according to a predetermined sort order and then split to build sub-trees for the children recursively down to the final leaf nodes. An order of objects must be predetermined in both approaches and stored in the tree. This causes a problem, which is that the pre-ordered leaf nodes cannot appropriately represent adjacent objects in a spatial space. There is a lack of efficiency when spatially adjacent objects in an R-tree are stored in multiple paths and queried as a group.

The first stage of a map matching algorithm involves the selection of candidate segments by searching a certain area centred on the given GPS position. This search process is often based on an indexing technique that traverses a tree, like R-tree, down to its leaf nodes to find those segments within the coverage of the search window. Since the topology of road networks is very important to match GPS positions onto road segments, those connected road segments around GPS positions are usually considered as candidate segments. However, spatial indexing techniques, such as the R-tree, do not consider the topology of road networks in constructing trees. In spatial indexing, it is likely that adjacent segments are stored in different sub-trees because the trees are constructed in a certain order (e.g., Hilbert order to build a Hilbert R-tree). As a result, the retrieval of candidate road segments in a given network requires several passes through the R-tree. The search complexity is significantly dependant upon the distribution patterns of road segments in a road network. Furthermore, the same road segment may be indexed more than once in the tree, requiring traversal of multiple paths. As a means of avoiding redundant searching in indexed spatial databases used for map matching, this paper presents a clustering technique to develop an adaptive candidate segments selection algorithm.

3. ROAD NETWORK DATA REPRESENTATION.

3.1. *Hierarchical Clustering Tree*. Unlike spatial indexing techniques, where objects are organized in a certain order, clustering techniques group objects by their characteristics. One important characteristic of road networks is the connectivity of road segments which can be used to cluster roads.

In the family of clustering algorithms for different types of applications, the two most common branches are the partitioning branch and the hierarchical branch

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

(Huang et al., 1998; Kotsiantis and Pintelas, 2004). Partitioning algorithms create a 'flat' decomposition of a data set into a set of clusters. Examples of partitioning algorithms include k-means, k-medoids and density-based algorithms. They generally need some input parameters which specify, either the number of clusters that a user intends to find, or a threshold for point density in clusters. However, it is difficult to determine what parameters are needed and what values they have, as the parameters may not even exist.

In contrast, hierarchical clustering algorithms do not actually partition a data set into clusters. Instead, an hierarchical representation of the data set is computed to reflect the possible hierarchical clustering structure of the data set. Hierarchical clustering algorithms are more robust and less influenced by cluster shapes; they are less sensitive to largely differing point densities of clusters, and they can represent nested clusters (Sander et al., 2003). Since hierarchical clustering algorithms can partition a data set with no prior knowledge of the number of clusters (e.g., distribution of road segments in a road network), they are seen as an alternative to spatial indexing techniques in this research. Clustering road segments in a hierarchical clustering tree by using an average-linkage method is described in the next section.

3.2. *Clustering Road Segments*. In graph theory, a graph is formally represented by the term $<V, E>$, where V represents vertices in the graph and E represents edges in the graph. A general adjacency matrix in graph theory is $A = \{a_{ij}\}$, where a_{ij} represents the weight (e.g., distance) between i^{th} vertex and j^{th} vertex. Consisting of intersections and segments, a spatial network can be represented as a graph where intersections are the vertices and segments are the edges. To better perform map matching for navigation applications, segments are clustered instead of intersections, because intersections only represent geometric information, whereas segments represent both geometrical and topological information.

To build the hierarchical clustering tree of a spatial network, a new matrix A′ is introduced in this algorithm. Generalizing a spatial network as a non-directional graph clustered by distance, A′ becomes a symmetric matrix. In order to define A′ two matrices are defined first. One matrix represents the topology of a spatial network, denoted by $T = \{t_{ij}\}$, where t_{ij} is 0 if the i^{th} and the j^{th} vertices are on the same segment (this occurs when i and j represent the same vertex or i and j are the two vertices of a segment) and t_{ij} is 1 when the i^{th} vertex is not directly connected to the j^{th} vertex. The other matrix is an adjacency matrix $A = \{a_{ij}\}$, which computes the Euclidian distance between any two vertices to represent the closeness of the intersections in geometric space. As a result, a new matrix $A' = A * T = \{a_{ij} * t_{ij}\}$ is defined to combine the two factors, geometrical distance and topological relationship from matrices A and T. The weight of each element in the matrix A′ not only considers the distance between the vertices (i.e, intersections), but also considers the topology of the spatial network.

A road network can be very large but sparse, since a road link is only connected to few links in topology. Therefore, topological relationship matrix T is sparse and could be built in a more compact data format by traversing only physical connected links. In addition, although real world road networks may contain one way segments, the road network is treated as a symmetric graph, which can simplify the clustering calculation and make the matrix A′ compact. A 'one-way' attribute can be added in the process of map matching.

Based on matrix A′, we build a hierarchical clustering tree from the bottom-up. Segments on the lowest level of this tree, as the smallest units in the structure, are

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

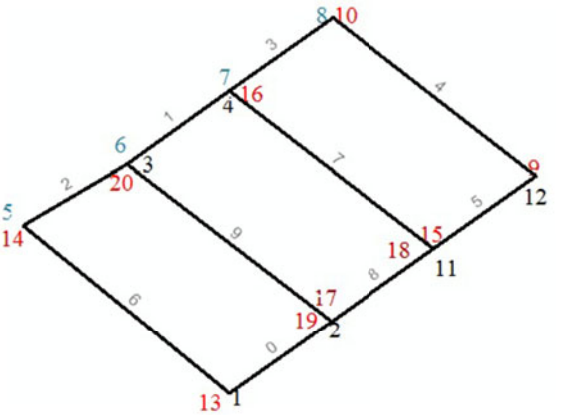

Figure 1. An example of road network.

grouped together based on the distances, one from another, and the clustering process is continued until the root of the tree is reached.

To illuminate the clustering procedure, Figure 1 shows 10 segments, labelled by their identities from 0 to 9. Every segment has two vertices, so 10 segments have twenty vertices, from 1 to 20. Matrix A′, therefore, becomes a 20-by-20 matrix. Since it is symmetrical, only it's upper or lower triangular (off diagonal) needs to be stored. The weights in the upper triangular and the lower triangular are all set as 0 s, as shown in Figure 2.

Given the matrix A′, a hierarchical clustering tree is built using the average-linkage clustering method. The average linkage clustering is based on measuring the proximity between two groups of vertices. Here we use the average distance between all pairs of vertices in cluster r and cluster s as the measurement. Its definition is as follows:

$$d(r, s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} \text{dist}(x_{ri}, x_{sj}) \tag{1}$$

where n_r is the number of vertices in cluster r and n_s is the number of vertices in cluster s, and x_{ri} is the i^{th} vertex in cluster r, and x_{sj} is the j^{th} vertex in cluster s.

After applying the average-linkage clustering method, Figure 3 shows how the hierarchical clustering tree is built from the matrix A′ in Figure 2. The clustering process starts from the bottom where average distances of two vertices on the same segment as a cluster are set to zero in A′. Segments are grouped together in an order that clusters those with closer distances until all the clusters are grouped together and the root is reached. The root of the clustering tree represents the entire spatial network as one group.

4. ADAPTIVE SEARCHING ALGORITHM.

In this section, a binary tree structure and its corresponding adaptive searching algorithm is discussed. The search will be conducted on a binary tree corresponding to the clustering tree that has been built. For map matching, the search starts from the entire road network and stops when a group of segments is found to meet a given criterion. With the movement of a user, relevant road segments change with respect to the received GPS positions.

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0	0	0.2062	0.25	0.2126	0.2062	0.25	0.3202	0.3	0.3202	0.2	0.3	0	0.2126	0.2	0.25	0.1	0.2	0.1	0.2062
2	0	0	0.2062	0.2062	0.253	0.2062	0.2062	0.25	0.2	0.25	0.1	0.2	0.1	0.253	0.1	0.2062	0	0.1	0	0.2062
3	0	0	0	0	0.0943	0	0.1	0.2	0.3202	0.2	0.25	0.3202	0.2062	0.0943	0.25	0.1	0.2062	0.25	0.2062	0
4	0	0	0	0	0.1942	0.1	0	0.1	0.25	0.1	0.2062	0.25	0.25	0.1942	0.2062	0	0.2062	0.2062	0.2062	0.1
5	0	0	0	0	0	0	0.1942	0.2941	0.402	0.2941	0.3206	0.402	0.2126	0	0.3206	0.1942	0.253	0.3206	0.253	0.0943
6	0	0	0	0	0	0	0.1	0.2	0.3202	0.2	0.25	0.3202	0.2062	0.0943	0.25	0.1	0.2062	0.25	0.2062	0
7	0	0	0	0	0	0	0	0	0.25	0.1	0.2062	0.25	0.25	0.1942	0.2062	0	0.2062	0.2062	0.2062	0.1
8	0	0	0	0	0	0	0	0	0.2062	0	0.2062	0.2062	0.3202	0.2941	0.2062	0.1	0.25	0.2062	0.25	0.2
9	0	0	0	0	0	0	0	0	0	0	0.1	0	0.3	0.402	0.1	0.25	0.2	0.1	0.2	0.3202
10	0	0	0	0	0	0	0	0	0	0	0.2062	0.2062	0.3202	0.2941	0.2062	0.1	0.25	0.2062	0.25	0.2
11	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0.3206	0	0.2062	0.1	0	0.1	0.25
12	0	0	0	0	0	0	0	0	0	0	0	0	0.3	0.402	0.1	0.25	0.2	0.1	0.2	0.3202
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0.25	0.1	0.2	0.1	0.2062
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.3206	0.1942	0.253	0.3206	0.253	0.0943
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0	0.1	0.25
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2062	0.2062	0.2062	0.1
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.2062
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	0.25
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2. Corresponding matrix (20-by-20).

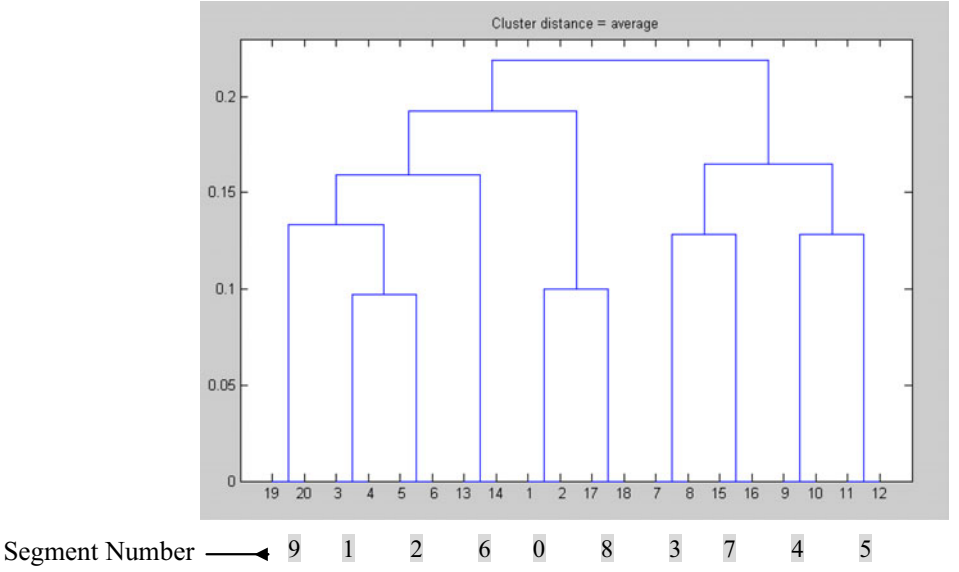

Figure 3. Corresponding clustering tree.

Instead of fixing a search window size for candidate road segments, grouped segments are dynamically obtained from the clustering tree.

4.1. *A Binary Tree Structure from the Clustering Tree.* Since each group only joins one of the two members (one of which may be compound) in the clustering tree shown in Figure 3, the clustering tree can be structured as a binary tree. Therefore, the map matching process will search a binary tree to identify candidate segments. Considering that road segments each have a different position, length and orientation, Minimum Bounding Rectangles (MBR) are used to delineate cluster boundaries. Each MBR in upper levels of the tree represents a group of segments where sibling nodes may cover overlapping areas. Construction of the binary tree is from bottom to top, so in the end, the root node represents a MBR covering the entire road network. Figure 4 shows a portion of the binary tree. Figure 5 shows the MBR of a group of road segments, segments 1, 2 and 9.

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

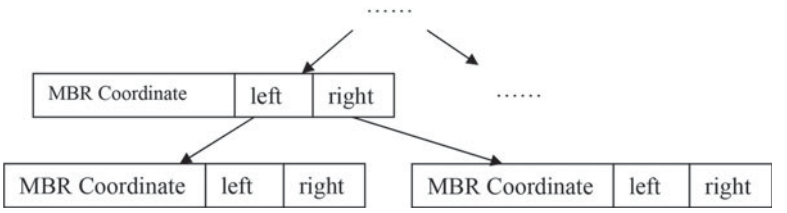

Figure 4. Data structure of a binary tree for road segment clustering.

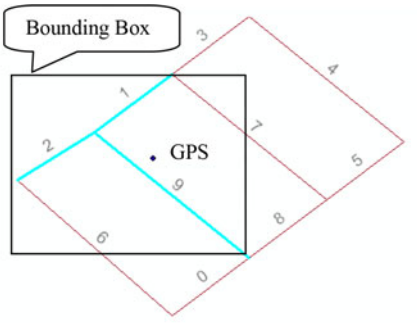

Figure 5. Distance from a GPS point to a MBR.

4.2. *Searching Algorithm*. As shown in Figure 4, the range of MBRs is narrowed down from top to bottom. Therefore, in order to determine the search window for map matching, the binary tree is traversed from top to bottom and will stop at a group node where a halt criterion is met. The halt criterion is important for determining the size of the search window. The searching algorithm is as follows:

1. Input the binary tree and a GPS point.
2. Set a criterion to halt search.
3. If the GPS point is within the boundary of the current group node and the halt criterion is not met, then search its sub-trees. Otherwise, search the sibling node.
4. If the halt criterion is met, then traverse all the leaf nodes of this sub-tree for the candidate road segments.

The halt criterion is critical in this algorithm. Since our purpose in the algorithm is to define the set of candidate road segments by a given GPS position, the halt criterion must consider the relationship between the GPS position and the segment groups. We describe a GPS position as p(x, y), and a bounding box as B(minx, miny, maxx, maxy). Then, c is noted as the centre of the bounding box. Distance (p, c) calculates the distance between the GPS position (p) and the centre point (c) and within (p, B) evaluates whether p is within the bounding box or not. With this, the criterion could be:

$$\text{IF Distance (p, c)} < \text{threshold and Within (p, B) THEN stop search.}$$

The smaller the threshold is, the closer the GPS position is to the centre of a selected group. In order to avoid misidentifying road candidates if the GPS position is located

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

on the boundary of two clustered sub-groups, the threshold is set relatively small in order to guarantee that the GPS position is close enough to the centre of a selected group. With the same threshold, road segments in one cluster would be selected as candidates more often in dense urban areas than in less dense, rural areas.

4.3. *Adaptive Search Window Set*. In spatial indexing techniques, a search window for candidate segments is normally a fixed-size rectangle centred at a given GPS position. In order to retrieve those segments within the search window, searching algorithms need to pass through the indexed tree several times. In contrast to such indexing techniques, a clustering tree provides an adaptive search window by clustering spatially close segments into groups and only passing through the tree once. The output of the searching algorithm is a list of candidate segments.

The number of segments selected in the list of candidate segments depends on three factors. The first factor is the density of the road network. For example, in a rural area, road networks are sparse and candidate segments may only include one or two segments of highways, whereas in urban (downtown) areas, more segments are included in the candidates list. Second is the relevant dynamic position of GPS positions in a clustered group. The final factor is the threshold in the halt criterion, which is set based on GPS error range.

In common with spatial indexing techniques, nodes in the clustering tree have overlapping MBRs, which is the reason why we set a threshold to justify whether the map matching of a GPS position is related to this clustered segment group or not. If a GPS position is located between groups, then the search window will be adaptively adjusted for a higher-level clustered group that includes the clustered lower-level groups. For instance, in Figure 5, if a GPS position is located on the corner of Segment 2 and Segment 6, the search window should cover the higher group that includes segments 9, 1, 2 and 6. Instead of only searching the group corresponding to the MBR shown in Figure 5, the search should be stopped on the group that contains segments 9, 1, 2 and 6. This search procedure starts from the root of the tree, as shown in Figure 3, and once the halt criterion is met it will stop at the level that includes segments 9, 1, 2 and 6, rather than searching the lower level.

4.4. *Adaptive Search Window Update*. To track the movement of a vehicle, a map matching search window must follow its motion. Whether to update a search window or keep searching in the previous window depends on the direction and speed of the vehicle's movement. With the previous candidate road segments in memory, and by knowing the successive positions to estimate the vehicle's movement, searching in the same segment group is continued. However, when the vehicle is moving out of a known segment group, a new sub-tree has to be initiated. Under this circumstance, the search window must be updated by repeating the initial search procedure with new parameters.

5. PERFORMANCE ANALYSIS.

To evaluate this proposed algorithm, we used two datasets, the University of Pittsburgh's main campus road map and the Allegheny County road map, and built two clustering trees. By using simulated GPS positions, we analyzed the performance of the algorithm.

5.1. *Datasets*. We employed road maps in the Pittsburgh area from the US TIGER data files and collected some GPS positions on the University of Pittsburgh's main campus and then tested our algorithm on different GPS locations, simulating a

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

Table 1. Tree features of two road networks.

Road network	Road segments (leaf nodes)	Maximum depth of hierarchical clustering tree	Minimum depth of hierarchical clustering tree
University of Pittsburgh's campus	171	14	10
Oakland area	1643	20	10

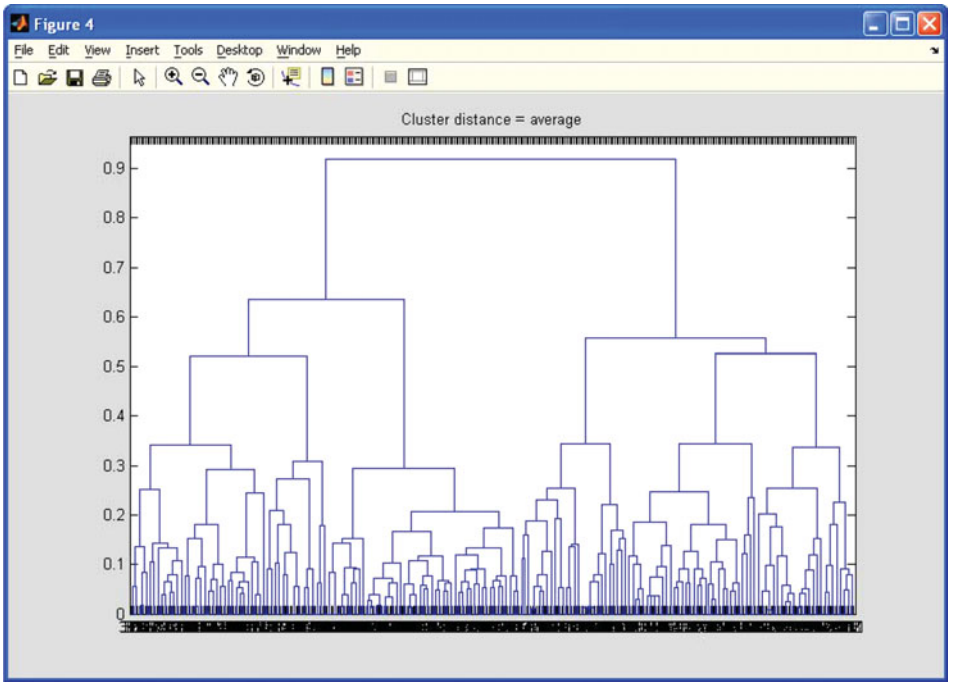

Figure 6. The clustered tree for the University of Pittsburgh's main campus.

vehicle's movement. Our algorithm was implemented in Matlab and tested in a PC environment with Intel Core 2, 2·13 GHz CPU and 2 GB memory.

5.2. *Construction Cost*. Memory usage and the time complexity involved in constructing a clustering tree is dependant upon the extent of a road network. In order to save both memory capacity and time, our strategy was to divide a large-scale road network into a set of sub-networks and then build a corresponding clustering tree for each sub-network. Thus the original road network corresponds to a forest structure. Indexing each tree in the forest is straightforward. As shown in Figure 4, intermediate memory is needed to build matrices to calculate the average distance of clusters. Each matrix, before compression, is n by n, so the memory usage is n^2, where n is the number of intersections.

In our algorithm, average-link clustering merges, in each iteration, the pair of clusters with the highest cohesion. Based on this recursive computation of cohesion, the time complexity of average-link clustering is $O(n^2 \log n)$. However, Murtagh (1992) compared various hierarchical clustering approaches in computational time

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

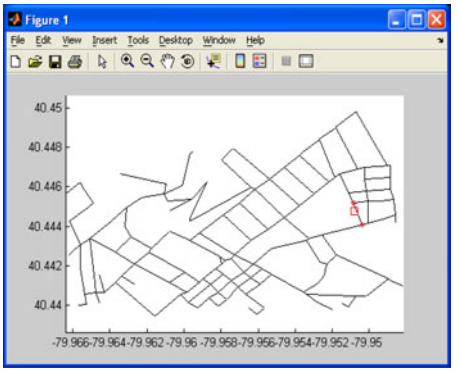

Figure 7-1. Moving on a relatively long road segment.

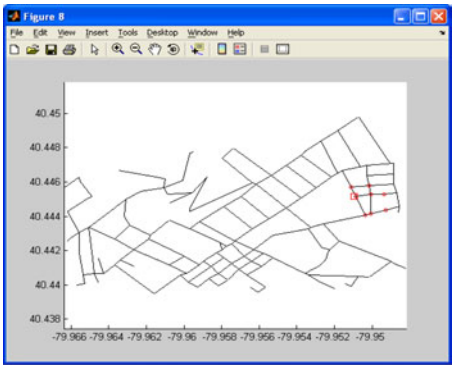

Figure 7-2. Approaching an intersection.

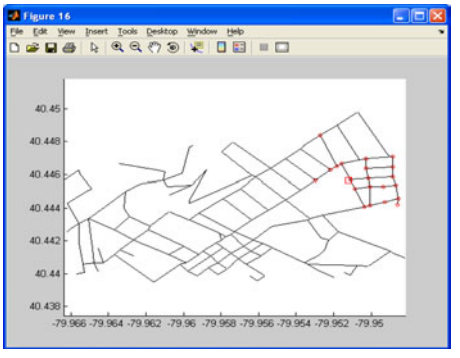

Figure 7-3. At another intersection.

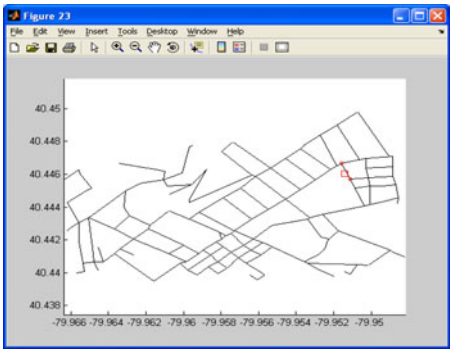

Figure 7-4. Moving in the middle of a segment.

Figure 7. Query results changing with a vehicle's movement.

complexity and concluded that $O(n^2)$ time implementations exist for most of the widely known hierarchical clustering methods, and some methods can even perform close to O(n) expected time for hierarchical clustering. Therefore, the construction cost of our hierarchical clustering tree can be further reduced by more sophisticated techniques. In this paper, we use a recursive approach to construct the hierarchical clustering tree. Table 1 provides features of the constructed clustering trees corresponding to the two different network scales. Constrained by memory limitation in MATLAB®, we could not conduct experiments with large-size road networks. In spite of this, our algorithm can be expanded to any large-scale network by splitting it into sub-networks and organizing it as a forest structure rather than a large tree structure. For example, Allegheny County geographically includes Pittsburgh City, so the city road network can be structured as a sub-network of the county road network.

A balanced binary tree has a depth, $\log_2 n$, but the hierarchical clustering trees in our algorithm are not balanced, which is why maximum and minimum depths are considered. Figure 6 shows the result of the tree construction by using, as an example, the University of Pittsburgh's main campus which has the maximum depth 14 and minimum depth 10 as shown in the first row in Table 1.

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

Table 2. Results of the searching cost.

Road network	Maximum hierarchical level	Median search depth	Median search depth (Intersection)	Median search depth (on-segment)	Median search depth (boundary)
University of Pittsburgh's campus	14	8	7	10	5
Oakland area	20	9	6	10	6

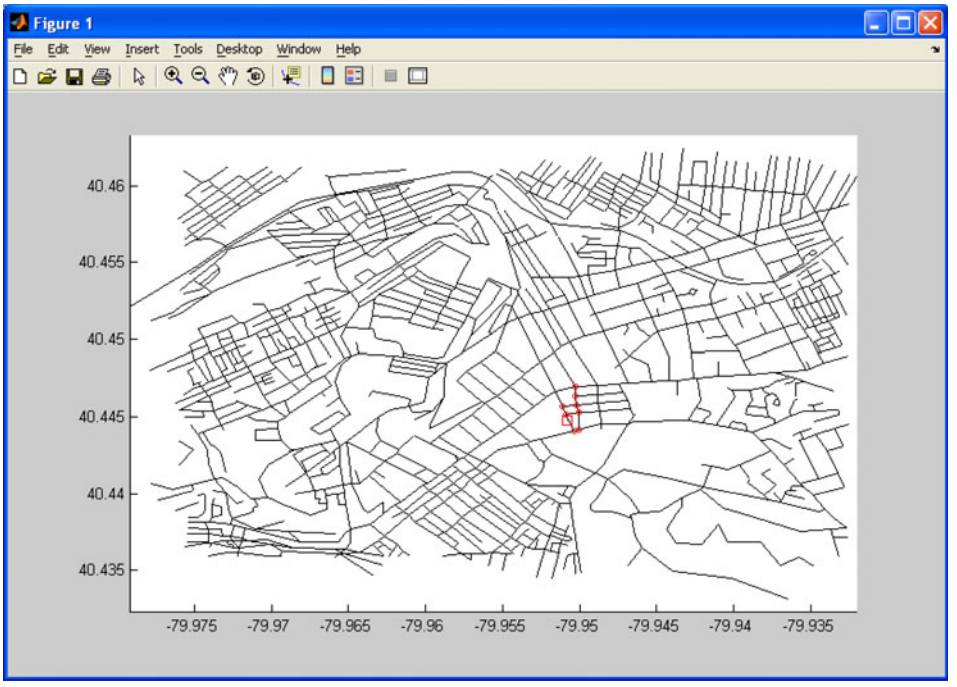

Figure 8. Example scenario on a large-scale network.

5.3. *Searching Cost*. In theory, if n is the number of segments, then the time complexity of searching binary trees is $O(\log_2 n)$. However, two factors influence query efficiency. One is the density of a road network and the other is the changing of a moving object from an intersection to its linked road segments. Although geometry and topology of road networks decide the tree clustering, once a road network is built, its corresponding graph morphology hardly changes. Thus, the structure of a clustering tree corresponding to the road network is essentially fixed, based on the clustering methods. Therefore, for a given road network, we mainly consider the query cost as influenced by the location of GPS positions.

As vehicles move on the road, they transfer from one road segment to another. As vehicles approach an intersection, or move on a relative short segment, a search window will cover more road candidates than those covered when moving on a long

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

segment. Query performance is most inefficient when a vehicle moves on a boundary between two large-scale clusters. Figure 7 shows how candidate road segments change with different positions when a vehicle is moving. The red square in each figure shows the received GPS position and the red points on the map show the intersections of candidate segments as the result of searching the clustering tree. Furthermore, by testing the same points in a larger area, e.g. in Oakland, which covers the entire University of Pittsburgh's main campus, it can be seen that the nature of the road network itself determines similar candidate roads, as shown in Figure 8, although different clustering trees for different road networks were searched. Since our average-linkage clustering approach is based on the average difference of groups, and the same area has constant road network structure, the searching algorithm produces results with certainty. Given a GPS position, the experiment shown in Figure 8 indicates that selected candidates in the large-scale network are consistent with the searching results in small-scale network, as shown in Figure 7.

As discussed previously, in order to evaluate various map matching situations, experiments were conducted in three scenarios: approaching an intersection, moving on a relatively long segment, and moving around a boundary of two clusters. Table 2 shows the average searching cost for the two situations.

In summary, since the hierarchical clustering tree built for the road network is not a balanced binary tree, we tested on real road networks to observe the actual construction structures and computed average search costs in different situations. Compared with multi-pass searching an object in spatial indexing techniques, searching on the clustering tree of road segments requires only one pass.

6. CONCLUSION AND FUTURE RESEARCH.

This paper proposes a new algorithm to search for candidate road segments given a GPS position for map matching. Rather than fixing a search window, this paper provides an adaptive window based on obtained GPS positions and a real road network. Considering that the clustering technique can represent the relationship of road segments based on the density of a road network well, we used a hierarchical clustering algorithm to cluster road segments. By building a hierarchical clustering tree, a binary tree is achieved. We analysed memory usage and time complexity of the tree construction and the computational efficiency of the searching algorithm. Based on the results of the experiments, it can be concluded that this approach adaptively finds candidate road segments based on GPS positions for areas with different geometry and topology. The binary tree was designed to group segments and speed up the search time and, as shown in Table 2, the algorithm can efficiently find road segments.

In our future research, we will address the opportunity for further savings in memory usage by decreasing the redundant intermediate points in map databases when building the clustering tree.

REFERENCES

Alborzi, H. and Samet, H. (2007). Execution time analysis of a top-down R-tree construction algorithm. *Information Processing Letters*, **101**, 6–12.

Bercken, J. V. D. and Seeger, B. (2001). An Evaluation of Generic Bulk Loading Techniques. *VLDB'2001*, 461–470.

https://doi.org/10.1017/S0373463313000076 Published online by Cambridge University Press

Chen, J. D., Meng, X. F., Guo, Y. Y. and Xiao, Z. (2009). Update-efficient Indexing of Moving Objects in Road Networks. *GEOINFORMATICA*, **13**(4), 397–424.

Guttman, A. (1984). R-tree: A dynamic index structure for spatial searching. *Proceedings of the ACM SIGMOD Conference*, 47–57.

Huang, Z. (1998). Extensions to the K-means Algorithm for Clustering Large Datasets with Categorical Values. *Data Mining and Knowledge Discovery*, (2), 283–304.

Kalashnikov, D. V., Prabhakar, S., Hambrusch, S. and Aref, W. (2002). Efficient evaluation of continuous range queries on moving objects. *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, 731–740.

Kotsiantis, S. and Pintelas, P. (2004). Recent Advances in Clustering: A Brief Survey. *WSEAS Transactions on Information Science and Applications*, **1**(1), 73–81.

Lin, H. Y. (2008a). Efficient and compact indexing structure for processing of spatial queries in line-based databases. *Data & Knowledge Engineering*, **64**(1), 365–380.

Lin, H. Y. (2008b). Using B+-trees for processing of line segments in large spatial databases. *Journal of Intelligent Information Systems*, **31**, 35–52.

Murtagh, F. (1992). Comments on "Parallel Algorithms for Hierarchical Clustering and Cluster Validity". *IEEE Transactions on PA analysis and machine intelligence*, **14**(10).

Sander, J., Qin, X., Lu, Z. Y., Niu, N. and Kovarsky, A. (2003). Automatic Extraction of Clusters from Hierarchical Clustering Representations. *Advances in Knowledge Discovery and Data Mining*, **2637**, 567–579.

Zhao, Y. L. (1997). Vehicle location and navigation systems: Intelligent Transportation Systems, *Artech House*, 83–103.

Zhao, J. L. and Cheng, H. K. (2001). Graph indexing for spatial data traversal in road map databases. *Computers & Operations Research*, **28**(3), 223–241.