Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Towards adaptable and tunable cloud-based map-matching strategy for GPS trajectories*#

Aftab Ahmed CHANDIO†‡[1,2], Nikos TZIRITAS[1], Fan ZHANG†[1], Ling YIN[1], Cheng-Zhong XU[1,3]

([1]*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China*)

([2]*Institute of Mathematics and Computer Science, University of Sindh, Jamshoro 70680, Pakistan*)

([3]*Department of Electrical and Computer Engineering, Wayne State University, Detroit 48202, USA*)

†E-mail: chandio.aftab@usindh.edu.pk; zhangfan@siat.ac.cn

Received Jan. 19, 2016; Revision accepted Apr. 11, 2016; Crosschecked Nov. 8, 2016

Abstract: Smart cities have given a significant impetus to manage traffic and use transport networks in an intelligent way. For the above reason, intelligent transportation systems (ITSs) and location-based services (LBSs) have become an interesting research area over the last years. Due to the rapid increase of data volume within the transportation domain, cloud environment is of paramount importance for storing, accessing, handling, and processing such huge amounts of data. A large part of data within the transportation domain is produced in the form of Global Positioning System (GPS) data. Such a kind of data is usually infrequent and noisy and achieving the quality of real-time transport applications based on GPS is a difficult task. The map-matching process, which is responsible for the accurate alignment of observed GPS positions onto a road network, plays a pivotal role in many ITS applications. Regarding accuracy, the performance of a map-matching strategy is based on the shortest path between two consecutive observed GPS positions. On the other extreme, processing shortest path queries (SPQs) incurs high computational cost. Current map-matching techniques are approached with a fixed number of parameters, i.e., the number of candidate points (N_{CP}) and error circle radius (ECR), which may lead to uncertainty when identifying road segments and either low-accurate results or a large number of SPQs. Moreover, due to the sampling error, GPS data with a high-sampling period (i.e., less than 10 s) typically contains extraneous datum, which also incurs an extra number of SPQs. Due to the high computation cost incurred by SPQs, current map-matching strategies are not suitable for real-time processing. In this paper, we propose real-time map-matching (called RT-MM), which is a fully adaptive map-matching strategy based on cloud to address the key challenge of SPQs in a map-matching process for real-time GPS trajectories. The evaluation of our approach against state-of-the-art approaches is performed through simulations based on both synthetic and real-world datasets.

Key words: Map-matching, GPS trajectories, Tuning-based, Cloud computing, Bulk synchronous parallel
http://dx.doi.org/10.1631/FITEE.1600027 **CLC number:** TP399; U495

1 Introduction

In recent years, many organizations are gradually migrating their applications to cloud environments to take advantage of computing and storage elasticity combined with the pay-as-you-go model. The advent and rapid growth of information and communication technologies (ICTs) has led to a new emerging concept called 'urban computing', wherein sensors, vehicles, devices, buildings, people, and roads are accessed as a compound to probe city dynamics (Zheng *et al.*, 2014). The data presented in the

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (No. 2015CB352400), the National Natural Science Foundation of China (Nos. 61100220 and U1401258), and the US National Science Foundation (No. CCF-1016966)

A preliminary version of this paper was presented at the 2nd International Conference on Internet of Vehicles (IOV 2015), Chengdu, China, Dec. 19, 2015

ORCID: Aftab Ahmed CHANDIO, http://orcid.org/0000-0002-5752-0520; Fan ZHANG, http://orcid.org/0000-0002-4974-3329

aforementioned components is usually obtained in the form of Global Positioning System (GPS) data. Due to the live nature of the above GPS-embedded components, their data volume can have an exponential growth, ranging from a few dozens of terabytes to petabytes (i.e., Big Data) (Chandio *et al.*, 2015a). To achieve a better quality of service (QoS), information in GPS data is often used in location-based services (LBSs) and intelligent transportation systems (ITSs), i.e., traffic flow analysis (Kühne *et al.*, 2003), route planner (Gonzalez *et al.*, 2007), geographical social network (Zheng *et al.*, 2008), and hot route finder (Li *et al.*, 2007). Because such applications need to process a massive amount of data in an effective way, there is an imperative need for adopting cloud. For instance, recent cloud-based services within the transportation domain are agent-based urban transportation systems (Li *et al.*, 2011a), urban intelligence transportation (Wang and Wang, 2013), cloud-enabled intensive FCD computation (Li *et al.*, 2011b), and traffic flow forecasting (Chen *et al.*, 2013).

In this paper, we address the problem of map-matching, which is playing a pivotal role in ascertaining the quality of many trajectory-based applications (e.g., driving directions, road guidance, moving object management, and traffic flow analysis). Basically, map-matching is a fundamental process of the above applications to align the observed GPS positions accurately onto a road network, which is in the form of a digital map (Lou *et al.*, 2009). However, in terms of accuracy, Lou *et al.* (2009), Newson and Krumm (2009), Yuan *et al.* (2010), and Goh *et al.* (2012) suggested that the best performance of a map-matching process is determined by the transition probability which incorporates the shortest path between two consecutive GPS points observed. On the other extreme, the execution of the shortest path queries (SPQs) in the map-matching process requires a high computational cost which subsequently makes map-matching unaffordable for real-time processing (Lou *et al.*, 2009).

Moreover, the map-matching process becomes a critical step when processing infrequent and imprecise sampling GPS data. Particularly, GPS data may suffer from two typical errors: (1) measurement error and (2) sampling error (Fang and Zimmermann, 2011). The measurement error is caused by the limitations of GPS who generates noisy GPS data, while the sampling error arises from a high-sampling period that generates extraneous GPS data. To handle the measurement error, the state-of-the-art map-matching techniques use two map-matching parameters, (1) the number of candidate points (N_{CP}) and (2) the error circle radius (ECR), to consider a number of road segments. Unfortunately, the state-of-the-art map-matching approaches (Lou *et al.*, 2009; Newson and Krumm, 2009; Yuan *et al.*, 2010; Goh *et al.*, 2012) are designed with fixed parameters (i.e., N_{CP} and ECR), which may lead to uncertainty and identifying either no road segments or a large number of road segments. In the case of no road segments, low accurate results may be produced, while in the case of a large number of SPQs, many SPQs may have to be processed.

In terms of the sampling error, GPS data with a high-sampling period (i.e., less than 10 s) typically contains extraneous datum (e.g., a vehicle stops many times, moves slowly, is trapped in a traffic jam, waits for the green signal, and moves on a high-way link), which also incurs an extra number of SPQs. If such a kind of GPS data is eliminated, a trajectory may suffer from discontinuity. Therefore, a reasonable technique is required that can intelligently adjust the sampling rate to reduce unnecessary GPS data and provide accurate trajectories before applying the map-matching process. Nevertheless, some approaches can execute SPQs by pre-computing the shortest path distances and partition a large network graph into small regions, such that the required partition can fit the memory constraints (Kolahdouzan and Shahabi, 2004; Hu *et al.*, 2006; Wang *et al.*, 2008; Liu *et al.*, 2012; Thomsen *et al.*, 2012; Tiwari and Kaushik, 2013). Since SPQs are executed in a sequential way, the state-of-the-art approaches experience high pre-computation and storage costs (Thomsen *et al.*, 2012).

Due to the aforementioned facts, the state-of-the-art approaches lead to both low accurate results and high running time. Therefore, such approaches are not suitable for real-time map-matching of GPS trajectories. Particularly, real-time traffic information plays a vital role in dynamic traffic control and management systems (Goh *et al.*, 2012; He *et al.*, 2013; Chen *et al.*, 2014). Consequently, it is of paramount importance for the map-matching of real-time GPS trajectories to optimize the problem of SPQs in an adaptive and efficient way. To deal with the above challenges, in this paper we present a real-time

map-matching (called RT-MM) approach, which is fully adaptive and based on cloud for real-time GPS trajectories. The proposed map-matching strategy is approached as follows:

1. We present a systematic model of the map-matching strategy for real-time GPS trajectories.

2. We introduce a tuning-based strategy that fine-tunes adaptively the interior and exterior parameters of a map-matching process. The interior parameters (N_{CP} and ECR) are tuned based on the locality of road networks. Basically, tuning the interior parameters based on the locality of the road networks works similarly as our previous approach, LB-MM (Chandio *et al.*, 2015b). Specifically, the LB-MM approach selects an apt number of values for interior parameters (N_{CP} and ECRs), based on different classes of the locality of a road network for each GPS sampling point. Since LB-MM considers only interior parameters, we incorporate it in tuning exterior parameters. The exterior parameters are relevant to a sampling rate according to a sliding window adjusted intelligently based on the feedback information of the monitored parameters at runtime. The technique for adjusting the sampling rate of GPS data provides a highly accurate trajectory after eliminating extraneous data without affecting the map-matching accuracy.

3. To compute the shortest path distances and temporal/speed constraint, we propose an extension of the single source shortest path (SSSP) function (Seo *et al.*, 2010) following the bulk synchronous parallel (BSP) paradigm (Malewicz *et al.*, 2010) in the cloud environment. We implement the SSSP function following the BSP paradigm in the Hama environment which is deployed on top of Hadoop. Taking advantage of a cloud environment, the proposed strategy drastically reduces the precomputation time and storage cost. By the above strategies (i.e., tuning- and cloud-based), we propose a viable solution to the issues posed earlier regarding efficient handling of SPQs.

The above approach is empirically evaluated using real-world and synthetic datasets. The results reveal that our proposed strategy reduces the overall running time of the map-matching process by reducing the number of SPQs and candidate points (CPs), while maintaining accuracy.

2 Related work

In this section, we first discuss the basic steps of a map-matching strategy and then address the problems of map-matching strategies found in state-of-the-art approaches. Particularly, all map-matching strategies follow three major steps: (1) initialization, (2) weight calculation, and (3) weight aggregation (Fig. 1). The initialization step of the map-matching strategy prepares a number of CPs projected on the candidate road segments within an ECR range. The second step concerns finding a path between two consecutive points in a trajectory by calculating weight scores regarding CPs. The last step concerns the aggregation of weight scores.

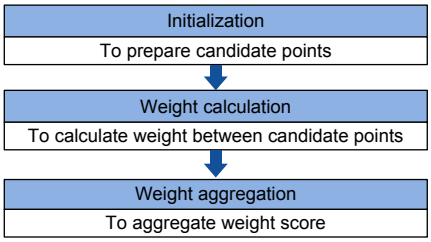


Fig. 1 Basic steps in a map-matching process

Map-matching strategies can be classified mainly into: (1) incremental (Greenfeld, 2002; Wenk *et al.*, 2006), (2) global (Brakatsoulas *et al.*, 2005; Lou *et al.*, 2009; Yuan *et al.*, 2010), and (3) statistical (Hummel and Tischler, 2005; Pink and Hummel, 2008; Newson and Krumm, 2009; Goh *et al.*, 2012) methods. Incremental map-matching strategies endeavor to find a local match of geometries for each GPS sample and suppose the small portion of space of the road network close to the GPS sample. In this approach, the weight score is aggregated based on the previous result for each GPS sampling point. This approach performs well in terms of accuracy when the sampling frequency is very high, i.e., 2–5 sampling intervals between GPS points. Regarding global map-matching approaches, the algorithms match an entire trajectory onto a road network. The global map-matching algorithms produce better results in terms of accuracy when applying low-sampling-rate GPS data. On the other hand, incremental approaches are very fast but suffer from low accurate results, while global approach algorithms achieve better accuracy at the expense of high computational cost. The

approaches in the last category leverage on statistical methods, i.e., Bayesian classifiers, hidden Markov model (Pink and Hummel, 2008), Kalman filter, and cubic spline interpolation (Hummel and Tischler, 2005), to match GPS points onto a road network. These approaches are particularly effective in handling GPS measurement errors.

Most current map-matching algorithms aim to achieve better QoS such as (1) high accuracy and (2) fast response time. Due to the tradeoff between high accuracy and fast response in map-matching strategies, providing a better QoS is difficult. It is a common approach to reduce the sampling rate of GPS data to save the cost of energy consumption, communication, and computation. For example, Fang and Zimmermann (2011) minimized the cost of energy by reducing the sampling rate of GPS data. Unfortunately, low-sampling-rate GPS data increases uncertainty and leads to less accurate results. On the other hand, because SPQs are time-consuming, handling SPQs in a sequential way makes the overall running time of many ITS applications unaffordable for real-time processing. Thus, considering the tradeoff between accuracy and response time in the map-matching strategy becomes a big challenge.

2.1 Spatial and temporal matching

Our proposed map-matching strategy is inspired by the spatial and temporal matching (ST-M) global map-matching strategy (Lou *et al.*, 2009) that aims to provide high accuracy. Spatial (i.e., geometric structure) and temporal (i.e., speed) constraints are incorporated in ST-M to solve the problem of low-sampling-rate GPS trajectories. In ST-M, first the weight function based on spatial and temporal constraints is defined with respect to two consecutive GPS points and their CPs. A candidate graph is created in ST-M where each node in the graph is a set of CPs and their edges represent a set of road segments regarding the shortest path between two neighbouring points associated with the spatial and temporal weight scores. The algorithm then generates a true path based on the largest summation of the weight functions.

Because the ST-M algorithm uses SPQs, through experimental evaluation we found that the execution time of the map-matching process increases dramatically when increasing the number of GPS sampling points per trajectory, violating QoS (Chandio *et al.*, 2014). For instance, Fig. 2a shows the number of total SPQs when varying N_{CP} and the number of GPS points per trajectory. Fig. 2b shows the total number of CPs found less than the fixed value of the CP parameter when varying ECR and the number of GPS points per trajectory.

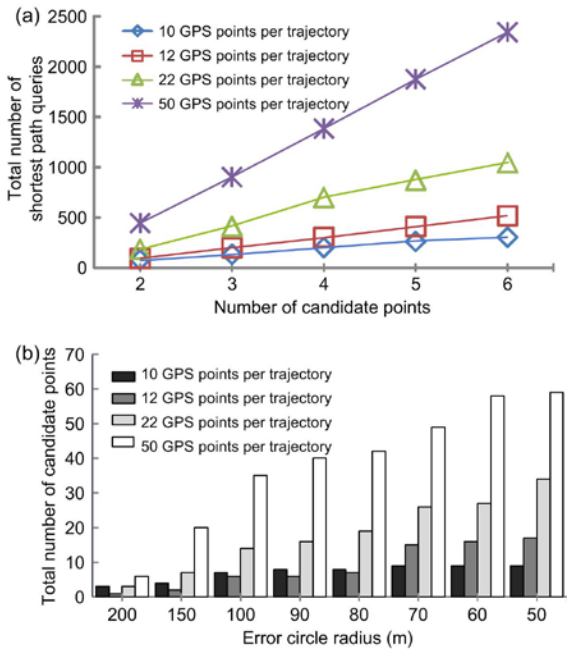


Fig. 2 Simulation results of spatial and temporal matching for real-world trajectories with different numbers of GPS trajectory points: (a) the relationship between the total number of SPQs and the number of candidate points (ECR=100 m); (b) the relationship between the total number of CP and the error circle radius

The time cost of ST-M is high because ST-M executes a large number of SPQs. On the other extreme, fixed N_{CP} and ECR can produce less accurate results when ECR does not provide the required N_{CP} (Quddus *et al.*, 2007) (Fig. 2b). For example, Fig. 3 shows a snapshot of two fixed ECR (i.e., r_1 is small and r_2 is large). As shown, for point p_i, two CPs can be considered if ECR is equal to r_1 (small); otherwise, four CPs are considered if ECR is equal to r_2 (large).

Therefore, in this study, we introduce a novel approach of applying an adaptive strategy that addresses the aforementioned challenges. Our tuning-based strategy adaptively adjusts the sampling rate of GPS data and fine-tunes the map-matching parameters (e.g., N_{CP} and ECR). The proposed strategy is

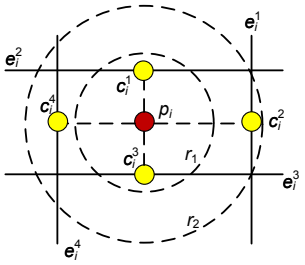


Fig. 3 The interior setting for considering candidate points (CPs) for a sampoing point

conceptually similar to the tunable job scheduling strategy in Tang *et al.* (2013). Moreover, we enhance our map-matching strategy with the strategy of pre-computing the shortest path distances and road network partitioning for real-time GPS trajectories. Previous approaches deal with the execution of SPQs by pre-computing the shortest path distances and partitioning a large network graph to small regions such that the required partition could be fit in memory (Kolahdouzan and Shahabi, 2004; Hu *et al.*, 2006; Wang *et al.*, 2008; Liu *et al.*, 2012; Thomsen *et al.*, 2012; Tiwari and Kaushik, 2013). However, the aforementioned techniques deal with the execution of SPQs in a sequential way, which comes at the expense of high pre-computation and storage costs (Thomsen *et al.*, 2012). In contrast, we propose an extension of the SSSP function following the BSP parallel paradigm in cloud environments that reduces pre-computation time and storage cost significantly.

3 Definitions

Definition 1 (GPS trajectory) A vehicle's trajectory T is measured in a sequence of GPS sample points, i.e., T: $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, where $0<p_{i+1}.t-p_i.t<\Delta T$, $p_i.t$ is the timestamp of p_i ($1\leq i<n$) and ΔT is the time spent between two consecutive GPS points. Besides, each GPS point $p_i \in T$ contains GPS position information, including latitude p_i.lat and longitude p_i.long.

Definition 2 (Strategy event) Map-matching examines trajectory T at pre-scheduled times O. Pre-scheduled time has a regular time interval and is also called event time and denoted by ΔO.

Definition 3 (Meta-points) Meta-points (denoted by M) represent the recent GPS sampling points that are ready for the map-matching process, i.e., M: $p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n$, where $p_i \in M$, $M \subset T$, and $0<p_{i+1}.t-p_i.t<\Delta M$ ($1\leq i<n$, $0\leq \Delta T<\Delta M$). The time spent between two consecutive GPS points of a meta-point list is denoted as ΔM.

Definition 4 (Sliding window) Sliding window represents the set of GPS points in M considered for map-matching.

Definition 5 (Road graph) A directed road network graph $G(V, E)$ is called a road graph, where V denotes the set of points intersecting the road segments, called vertices, and E signifies a set of road segments, called edges. A directed edge e is associated with (1) the unique identification e.gid, (2) the average travel speed $e.v$, (3) the road length $e.l$, (4) the starting point e.start, (5) the ending point e.end, and (6) the intermediate points comprising the road polyline.

Definition 6 (Path) A path P is a list of connected road segments between two vertices (V_i, V_j) in a road network G, i.e., P: $e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_n$, where, for $1\leq k<n$, e_1.start=V_i, e_n.end=V_j, and e_k.end=e_{k+1}.start.

4 Fully adaptive map-matching strategy

In this section, we introduce our proposed technique called RT-MM (real-time-map-matching), which is a fully adaptive map-matching strategy for real-time GPS trajectories based on cloud. The primary goal of this study is to improve the map-matching strategy that can provide a better trade-off between accuracy and response time. This section discusses the architecture and major components of the proposed map-matching strategy.

4.1 System architecture

A complete systematic model of our proposed technique consists of two main steps: off-line and online efforts. The architecture of the system (Fig. 4) is explained as follows.

Off-line efforts are composed of partitioning the road network graph and pre-computing the shortest path distance and temporal/speed constraints. The road network graph is split into small sub-graphs, with each sub-graph keeping its boundary values such as the maximum and minimum longitude and latitude (Wang *et al.*, 2008). Our partition approach guarantees that each sub-graph contains approximately an equal number of nodes for the purpose of load balancing. The reason for splitting a road network graph

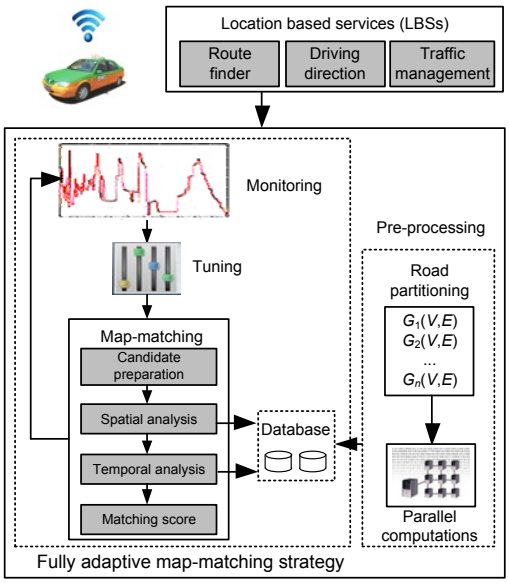


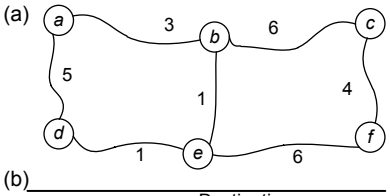
Fig. 4 An overview of the proposed system

into small partitions is due to memory constraints. However, splitting a road network into partitions may degrade the accuracy. To maintain accuracy, we expand the boundary of each partition by adding a certain value to the latitude and longitude coordinates of the border line. In that way, a process of the shortest path distance between two consecutive GPS points can be processed within the same partition. Regarding the offline efforts, the shortest path distance and temporal/speed constraints are computed by following the parallel computing paradigm, i.e., BSP (Malewicz *et al.*, 2010), in a cloud environment to reduce the pre-processing time. We provide an extension of the SSSP function following the BSP paradigm. The aforementioned implementation takes place in Hama (Seo *et al.*, 2010) on top of the Hadoop environment. Besides computing the shortest path distance, the proposed SSSP function computes the number of edges and the speed constraints of all edges contained in the shortest path. We briefly discuss the modified SSSP function employing the BSP parallel paradigm in Section 4.2.

In terms of online efforts, our proposed map-matching strategy monitors periodically the GPS sampling data in real time to adaptively fine-tune the interior and exterior parameters of the map-matching process. The interior parameters (i.e., N_{CP} and ECR) are tuned based on the locality of the corresponding road network. The exterior parameters are relevant to the sampling rate and the number of GPS points contained in the sliding window, and are tuned based on the feedback information of monitored parameters at runtime. In this phase, the map-matching strategy checks the queue periodically and accepts real-time GPS trajectory T for map-matching on the digital map. Nevertheless, the online phase can work independently using SPQs without the aforementioned off-line efforts. A complete map-matching process for real-time GPS trajectories is further discussed in Section 4.3.

4.2 Modified SSSP algorithm following the BSP parallel paradigm in cloud environments

In this section, we discuss our modified SSSP algorithm that follows the BSP parallel paradigm to generate all (source, destination) shortest path pairs together with the temporal/speed constraint of all the nodes in the graph. The SSSP function computes the shortest path distances between a single source node and all-pairs in the graph. Fig. 5 shows an example of the SSSP process for a network graph consisting of six nodes and seven edges. Each edge is weighted with a value (Fig. 5a). The SSSP results of each node in the graph are presented in Fig. 5b. Basically, SSSP is a classical function which has been well solved based on the Dijkstra algorithm (Dijkstra, 1959).



(b)

Source	Destination					
	a	*b*	*c*	*d*	*e*	*f*
a	-	3	9	5	4	10
b	3	-	6	2	1	7
c	9	6	-	8	7	4
d	5	2	8	-	1	7
e	4	1	7	1	-	6
f	10	7	4	7	6	-

Fig. 5 An example of SSSP computation: (a) a network graph; (b) SSSP results of all nodes in the graph

Because the calculation of the shortest path distances contains graph computations demanding large data processing, we use a parallel paradigm to reduce the pre-computation time. The implementation of the SSSP function in a parallel paradigm makes the SSSP function more efficient against other shortest path techniques following sequential processing. The SSSP function has recently been studied into two

well-known parallel paradigms (Kajdanowicz *et al.*, 2014), i.e., MapReduce (MR) (Dean and Ghemawat, 2008) and BSP (Malewicz*, et al.*, 2010). Due to the graph and the iterative processing nature of the problem, MR is deemed not suitable for the shortest path computations (Kajdanowicz *et al.*, 2014).

Malewicz *et al.* (2010) in Google Inc. introduced an alternative model, called Pregel, which is based on the BSP parallel paradigm. Pregel has been implemented in Hama (Seo *et al.*, 2010) and we chose this tool to perform the shortest path computations. The BSP algorithm will generate a series of supersteps, and each superstep executes a user-defined function in parallel asynchronously (Yin *et al.*, 2003; Kajdanowicz *et al.*, 2014) (Fig. 6). At the end of each superstep, the BSP algorithm uses a synchronization barrier to synchronize computations in the system. The synchronization barrier is responsible for the state wherein each superstep waits for the remaining supersteps running in parallel. By default, the SSSP function accepts an input of the network graph and a single source node. As discussed previously, the proposed SSSP function computes the shortest paths, the number of edges, and the total speed of all edges contained in the shortest path. The output of the function is used to calculate spatial and temporal weight scores of two consecutive GPS sample points for the final map-matching process.

Algorithm 1 shows the modified SSSP function following the BSP parallel paradigm. It takes a road network graph as the input. The format of the input of Algorithm 1 is modified as follows:

Format:

Node\tEdge$_1$:Length,Speed\tEdge$_2$:Length,Speed\...
\tEdge$_n$:Length,Speed

Examples:

84\t192:89.4,180
85\t82:7.3,260\t81:8.3,90\t176:162.3,90
86\t48:107.1,90\t164:120.2,260\t306:203.4,260
87\t105:24.6,260

(Default format is used only for computing the shortest distance, explained into the Hama SSSP page, http://wiki.apache.org/hama/SSSP)

Each example represents a node and its adjacent edges. Each adjacent edge is represented by three variables: edge identification, edge length, and speed.

At the start of the proposed SSSP function, the variable of the actual value (at each vertex except the source) that corresponds to the minimum distance is initialized with infinity (lines 2–6). Basically, in the modified SSSP, each vertex reads a message from its adjacent edges, which contains three parts: distance, total number of edges, and total speed of the edges between the source and current vertices. If the distance in a message at a given vertex is smaller than the actual value associated with the current vertex, then the function updates the above three parameters associated with the current vertex (lines 1–13). Finally,

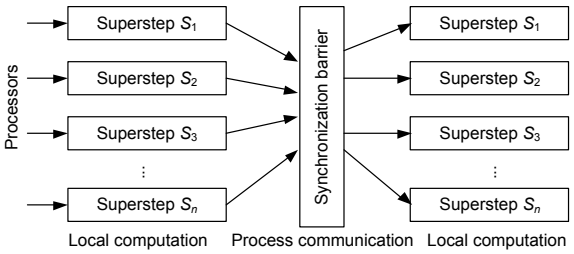


Fig. 6 A snapshot of bulk synchronous parallel (BSP) processing

Algorithm 1 Modified SSSP algorithm in BSP

Input: a road network graph $G(V, E)$ and source node
Output: the shortest paths // The output record provides the
// shortest distance, a number of edges, and total speed of
// roads in each shortest path

```
 1: function COMPUTE(vertex v, message
            m<distance, edgeCount, speedCouns>)
 2: if isStartVertex(v) then
 3:    minDistanceDefault=0
 4: else
 5:    minDistanceInDefault=∞
 6: endif
 7: foreach message m do
 8:    if m.distance<minDistanceInDefault then
 9:       minDistanceInDefault=m.distance
10:       edgeCountInDefault=m.edgeCount
11:       speedCountInDefault=m.speedCount
12:    endif
13: endfor
14: if v.distance>minDistanceInDefault then
15:    v.minDistance=minDistanceInDefault
16:    v.edgeCount=edgeCountInDefault+1
17:    v.speedCount=speedCountInDefault
18:    foreach e in the neighbor of v
19:       m.distance=e.distance+minDistanceInDefault
20:       m.edgeCount=m.edgeCount+1
21:       m.speedCount=e.speed+speedCountInDefault
22:       sendMessage(e.id, m)
23:    endfor
24: endif
25: voteToHalt()
26: end function
```

the current vertex sends a message with the updated values to all its adjacent edges (lines 14–24) and becomes an inactive vertex by calling voteToHalt() (line 25). When a vertex receives a message, it becomes active. The process terminates when there are no active vertices to be considered.

4.3 Map-matching strategy for real-time GPS trajectories

The proposed map-matching strategy is based on window- and tuning-based techniques. In the window-based approach, recent GPS sampling points of the vehicle are chosen for the map-matching process. Regarding the tuning-based approach, the interior and exterior parameters of map-matching are fine-tuned. The steps used in our proposed map-matching strategy are depicted in Fig. 4: (1) monitoring, (2) tuning, (3) candidate preparation, (4) spatial analysis, (5) temporal analysis, and (6) score matching.

In the monitoring phase, our map-matching strategy periodically monitors the flow of real-time GPS workloads and the parameters of the sliding window in each strategy event E. Based on the monitored parameters, we determine the current driving state of a vehicle to help the next phase fine-tune the sampling rate of GPS data. Our intuition is that if a vehicle is driven on a high way or a long road segment, then the sampling rate can be reduced by filtering the noisy and extraneous data. Otherwise, a high sampling rate of GPS data is needed to ensure the accuracy of the map-matching if a vehicle is traveling inside the city. To estimate the current driving state of the vehicle, we choose the average speed of the vehicle extracted from the previous sliding window (Fang and Zimmermann, 2011), which is defined as follows:

$$\overline{S}_{i-1}=\frac{\sum_{u=1}^{k} e'_u.l}{\Delta W_{(p'_i.t-p'_1.t)}}, \tag{1}$$

where $e'_u.l_1 \rightarrow e'_u.l_2 \rightarrow \cdots \rightarrow e'_u.l_k$ is the total length of the matched shortest path and $\Delta W_{(p'_i.t-p'_1.t)}$ is the total time spent on path P in the previous sliding window W_{i-1}. If the vehicle is traveling with an average speed $\overline{S}_{i-1}$ of 40 km/h or above, then the vehicle is thought to be on high way; otherwise, we consider that the vehicle is moving inside the city. We set this threshold in our experiment because, in the road network, the road segments inside the city permit a 40 km/h maximum driving speed.

The tuning phase is responsible for amending the sampling rate ΔT of GPS sampling data based on monitored parameter $\overline{S}_{i-1}$. A new sampling rate ΔM of the GPS sampling data is obtained by

$$\Delta M=\frac{\Delta E}{W}, \tag{2}$$

where W represents the number of GPS sampling points in the sliding window. The exterior parameters considered in this study are ΔE and W. Both parameters are tunable with pre-defined values such that ΔE=10 and W=12 imply that the vehicle is in high-way driving state, while ΔE=5 and W=10 imply that the vehicle is in inside-city driving state. In this way, the sampling rate of GPS data can be adjusted reasonably.

To maintain the quality of our map-matching strategy, we incorporate the weight score of the previous mapped GPS sampling point as the source of the path in the current sliding window. A schematic model is shown in Fig. 7, which shows three sliding windows. As observed, the second and third sliding windows follow the most recent results (i.e., last mapped GPS point) of the previous sliding window to maintain trajectory continuity.

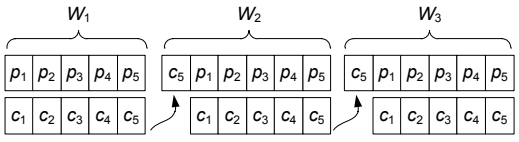


Fig. 7 An example of the proposed window-based map-matching scheme for real-time trajectory (W=5), where p_1, p_2, …, p_5 denote the GPS sampling points and c_1, c_2, …, c_5 denote the corresponding correct candidate points

Besides the exterior settings, the tuning phase is also responsible for tuning the interior parameters, i.e., N_{CP} and ECR, for the determination of the most likely road segments. The tuning strategy decides the interior settings based on the locality of the road network. We characterized the locality of the road network into a grid format. The number of grids varies according to the memory capacity of the underlying system (e.g.,

we use 200×200 grids in our experiment). For the characterization of the locality of the road network, we create the relationship in Table 1 for considering N_{CP} and ECR settings.

Table 1 Parameters based on locality information*

Class	GPS point density	ECR (m)	Number of CPs
Class-I	$\sum e \geq 21$	60	3
Class-II	$16 \leq \sum e < 21$	80	4
Class-III	$12 \leq \sum e < 16$	100	5
Class-IV	$9 \leq \sum e < 12$	120	6
Class-V	$\sum e < 9$	150	7

* Chandio *et al.* (2015b)

Based on the tuning phase, the candidate preparation phase retrieves a possible number of candidate road segments for each GPS sampling point p in sliding window M. The CPs c on candidate road segment e are generated by geometry projection within ECR r with respect to the Euclidean distance between the GPS sampling points. A set of the closest CPs c to the GPS sampling point are chosen such that $c = \text{argmin}_{\forall c_i \in e} \text{dist}(c_i, p)$ and $\text{dist}(c_i, p)$ is the distance between candidate point c_i and GPS sampling point p. In consequence, the map-matching process considers a number of the most likely CPs matching to the respective GPS sampling point. Fig. 3 illustrates the strategy of interior settings.

The next step concerns spatio-temporal analysis, which evaluates the retrieved candidate points to generate the candidate graph $G'(V', E')$. In this study, the GPS measurement error is assumed to follow a normal distribution $N(\mu, \sigma^2)$ of the distance GPS point and candidate point $x_i^j = \text{dist}(c_i^j, p)$ (c_i^j is the jth candidate point of the ith GPS sampling point). Like Lou *et al.* (2009), for GPS measurement, a zero-mean normal distribution with 20 m standard deviation is used, which is described by

$$N(c_i^j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[\frac{(x_i^j - \mu)^2}{2\sigma^2}\right]. \quad (3)$$

The spatial-temporal analysis function in our proposed map-matching strategy is adopted from Lou *et al.* (2009), which is denoted as

$$F(c_{i-1}^t \to c_i^s) = F_s(c_{i-1}^t \to c_i^s) \times F_t(c_{i-1}^t \to c_i^s), \ 2 \leq i \leq n, \quad (4)$$

where F_s is a spatial analysis function calculated by

$$F_s(c_{i-1}^t \to c_i^s) = N(c_i^s) \times V(c_{i-1}^t \to c_i^s), \ 2 \leq i \leq n, \quad (5)$$

and V is a likelihood method in transmission probability, which defines a true path from $p_{i-1} \to p_i$ and follows the shortest path from $c_{i-1} \to c_i$, calculated by

$$V = \frac{d_{i-1 \to i}}{w_{(i-1,t) \to (i,s)}}, \ 2 \leq i \leq n.$$

The temporal analysis function F_t is defined by

$$F_t(c_{i-1}^t \to c_i^s) = \frac{\sum_{u=1}^{k} e_u'.v \cdot \overline{V}_{(i-1,t) \to (i,s)}}{\sqrt{\sum_{u=1}^{k} (e_u'.v)^2} \times \sqrt{\sum_{u=1}^{k} \overline{V}^2_{(i-1,t) \to (i,s)}}}, \quad (6)$$

where $e_1'.v \to e_2'.v \to \cdots \to e_k'.v$ is the speed constraint of the road segments belonging to the shortest path. The temporal analysis phase creates a score for each candidate point by considering the average speed of the road segments between two neighboring candidate points. For details of spatial and temporal analysis, we refer readers to Lou *et al.* (2009). Our proposed strategy incorporated with the spatial and temporal analysis function is enhanced by replacing SPQs and the speed constraint pre-computed in BSP parallel paradigm (Fig. 4).

In the matching score phase, the candidate graph $G'(V', E')$ is evaluated. Each node V' in G' is a set of candidate points associated with the observation probability $N(c_i^j)$, while E' in G' is a set of road segments in the shortest path between two neighboring points which are associated with the spatial $F_s(c_{i-1}^t \to c_i^s)$ and temporal $F_t(c_{i-1}^t \to c_i^s)$ scores. Finally, a sequence of real candidate points is selected from a candidate graph $G'(V', E')$, considering the largest summation of the scores.

To configure the adaptive map-matching strategy for real-time GPS trajectories, we first give some notations before describing the fully adaptive map-matching strategy. T_E and T_W are the tunable sampling rate and the number of GPS points in sliding window

W, respectively. Δ_E and Δ_W are used for tuning T_E and T_W at each time interval event, respectively. P_m states the monitored parameters in real time. S_i and S_d refer to the events triggering the increment and decrement of the tunable parameters, respectively. $\text{Check}_{\text{Time}}$ denotes the time instance at which the runtime parameters are checked and the adaptive strategy is tuned.

Algorithm 2 describes our adaptive map-matching strategy for real-time GPS trajectories. At each time interval, the strategy collects the parameter values P_m of the previous sliding window (line 3). Then the parameter values are compared with the predefined thresholds to decide whether a strategy tuning event should be triggered (lines 4–13). In consequence, the tunable parameters are adjusted at each time interval (line 14). A set of candidate points within an ECR for each GPS sampling point is generated based on new tuning parameters by invoking the populateCandidates() function (lines 15–17). Algorithm 3 shows the pseudocode for preparing the candidate points based on the locality of a road network. It initially finds the grid for each GPS point and then returns the best N_{CP} and ECR. Getting back to Algorithm 2, generateGraph() and findSequence() follow the same way as described in Lou *et al.* (2009) (lines 20–21). generateGraph() generates a candidate graph (in the matching phase) for the GPS points in each sliding window, while findSequence() calculates the weight scores to identify a sequence of real CPs by considering the largest summation of the weight scores assigned to CPs.

Algorithm 2 Fully adaptive map-matching strategy for real-time GPS trajectories

Input: a road network graph $G(V, E)$ and a trajectory
Output: a matched sequence of CPs for a given trajectory

1: **do**
2: **if** $\text{Current}_{\text{Time}}-\text{LastWin}_{\text{Time}}>\text{Check}_{\text{Time}}$ **then**
3: P_m=collectMonitoredParas() // collect parameters
4: e=eventNeed(P_m) // feedback for tuning
5: **if** $e=S_i$ **then**
7: $T_E=T_E+\Delta_E$ // increment for tuning
8: $T_W=T_W+\Delta_W$ // increment for tuning
9: **endif**
10: **if** $e=S_d$ **then**
11: $T_E=T_E-\Delta_E$ // decrement for tuning
12: $T_W=T_W-\Delta_W$ // decrement for tuning
13: **endif**
14: ΔM=tuneSR(T_E, T_W) // re-tune a sampling ratio
15: **foreach** i in M **do**
16: $C_{i.[]}$.add=populateCandidates(i) // locality-based
17: **endfor**
18: $\text{LastWin}_{\text{Time}}=\text{Current}_{\text{Time}}$
19: $\text{Check}_{\text{Time}}=\Delta E$ //reset time interval
20: G'=generateGraph(C) // generate a candidate graph
21: findSequence(G') // find matched candidate points
22: **endif**
23: **while TRUE**

Algorithm 3 populateCandidates()

Input: a GPS point
Output: candidate points (CPs)

1: gc=findGridCells(p.long, p.lat) // find grid cells
2: ge=findMaxEdges(gc) // find a number of roads in gc
3: cp=bestCandidateNumber(ge) // find a number for CPs
4: rp=bestRadiusNumber(ge) // find a number for ECR
5: **foreach** i in gc **do**
6: gids[]=getGids(i) // collect all roads in the grid cells
7: **endfor**
8: return c[]=getCandidatePoints(gids, cp, rp) // get CPs

5 Experimental evaluation

In this section, we give a discussion about the experimental setup as well as the results obtained from the comparison between our proposed approach and state-of-the-art techniques.

5.1 Computation environments

To carry out the experiments, we employed two computation environments. For the pre-processing step based on the BSP parallel paradigm, the Hadoop (version 1.1.2) cloud environment was created. Three physical machines were dedicated, with tone for master node and the rest for worker nodes. Each of the node was composed of eight 2.20 GHz Intel® Xeon® CPUs and 32 GB RAM. For BSP parallel processing, we employed Hama (version 0.6.2) on the top of Hadoop. Next, one of the three physical machines was dedicated to provide real-time map-matching processing. For a fair comparison, we implemented all of the studied map-matching strategies in Java (version 1.7) programming language. We stored all of the datasets in the PostgreSQL (version 9.1) database together with the PostGIS (version 2.0) spatial tool and pgRouting tool for executing SPQs.

5.2 Datasets

1. Road network: We used a real-world road network graph of Shenzhen city, China. The real-world road network graph contains a total of 86 335 vertices and 133 397 road segments.

2. Real-world GPS-embedded vehicle trajectory: For testing the map-matching strategies, we used a real-world trajectory dataset of GPS-embedded taxi-cab traveling around Shenzhen city on Oct. 10, 2013. A total of 5128 GPS points were collected within 24 h. We also chose a set of trips obtained in the dataset, which are varied according to different numbers of GPS sampling points and the sampling rate.

3. Synthetic trajectory dataset: We used a synthetic trajectory dataset generated randomly by our own simulator. The simulator first creates a shortest path between two random vertices in the road network. Then, a set of edges returned by the shortest path is considered as a ground truth. The sampling points are generated on the respective edges according to the sampling rate. To encapsulate the GPS sampling error, each GPS sampling point follows a zero-mean 20 m standard deviation normal distribution. The method for creating the synthetic trajectory dataset was verified and used in Lou *et al.* (2009) to evaluate the ST-Matching algorithm. A similar way is reasonably considered in this study to evaluate our proposed strategies against the ST-Matching algorithm.

6 Experimental results and discussions

To evaluate the performance of the proposed strategies, we used two metrics: running/computation time and accuracy. The running time is the overall execution time of the map-matching process, and the accuracy represents the percentage of the correct matching GPS points (CMP). Specifically, CMP is calculated according to Eq. (7) (Yuan *et al.*, 2010):

$$\text{CMP}=\frac{\text{Number of correctly matched points}}{\text{Total number of points to be matched}}\times 100\%. \tag{7}$$

We evaluate the performance of the proposed fully adaptive map-matching strategy against the ST-M algorithm and the proposed strategy with static parameters (N_{CP}=5 and ECR=100 m). We call the proposed map-matching strategy with static parameters RT-M-I and the fully adaptive map-matching strategy RT-M-II. Moreover, we analyze the proposed strategy by using the total numbers of SPQs and CPs required.

6.1 Comparisons between RT-MM and ST-M

First, we analyze the proposed RT-MM strategy against ST-M regarding running time and accuracy. Fig. 8a shows the average running times of five experiments when varying the number of GPS points per trajectory.

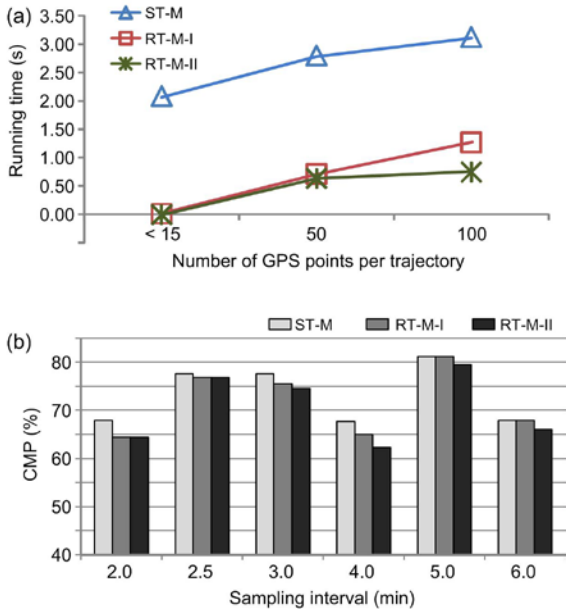


Fig. 8 The running time with different numbers of GPS points per trajectories (a) and the accuracy with different sampling intervals for the synthetic dataset (b)

The results shown in Fig 8a reveal that our proposed strategies took much less computation time as compared to the ST-M algorithm. Another remark is that the adaptive strategy, RT-M-II, outperforms RT-M-I when the trajectories consist of more than 50 GPS sampling points. Fig. 8b shows the comparison of the average CMP of synthetic trajectories when varying the sampling rate from 2 to 6 min per trajectory. Our proposed strategies produced almost the same results as those of ST-M. The reason that the accuracy of RT-MM is lower than that of ST-M is that, the road network is partitioned into small regions and the trajectory is traveling on more partitions of the road network. If a trajectory crosses more than one

partition, the accuracy of our proposed map-matching strategy will be degraded slightly. Such an intuition is shown in Fig. 8b; i.e., the trajectory with a sampling rate of 4.0 min crosses on a total of five partitions of the road network, and the accuracy is degraded.

6.2 Analyzing RT-MM when varying the static parameters

We examined the execution time of our proposed map-matching strategy (RT-M-I) when varying the static parameters (N_{CP} and ECR). Fig. 9a shows the execution time of RT-M-I for different N_{CP} (from 2 to 5) when varying the number of GPS points per trajectory. Specifically, we fixed ECR to 100 m. We can observe that trajectories with a smaller N_{CP} take less running time for mapping against the ones with a larger N_{CP}. Moreover, we evaluated the proposed strategy in terms of running time when varying ECR. Particularly, we used a full-day real-world trajectory (described in Section 5.2) for map-matching when varying ECR from 100 m to 300 m as well as N_{CP} from 3 to 10. Fig. 9b shows the overall running time of the proposed strategy when processing the full-day real-world trajectory. Fig. 9b reveals that higher ECR and N_{CP} increase the running time. Next, we analyzed the impact of the size of sliding windows with respect to the static parameters values. Fig. 9c shows the average running time of each sliding window when matching the full-day real-world trajectory. We can find that the running time is significantly affected when varying N_{CP} and ECR.

6.2.1 Impact of adaptive parameters based on locality of the road network

In this section, we evaluated the proposed adaptive strategy in terms of CMP and the distributions of GPS sampling points. Fig. 10a shows the average CMP for synthetic trajectories when varying the sampling rate from 2.0 to 6.0 min per trajectory. The proposed adaptive strategy produced almost the same results as compared to the ST-M algorithm.

Fig. 10b shows the distributions of GPS sampling points and reveals that Class-I, II, and III are dominant against the remaining classes. By tuning exterior parameters, the proposed strategy selected reasonably29% GPS sampling points of the full-day real-world trajectory. The rest of them were excluded as extraneous GPS sampling points. Table 2 shows

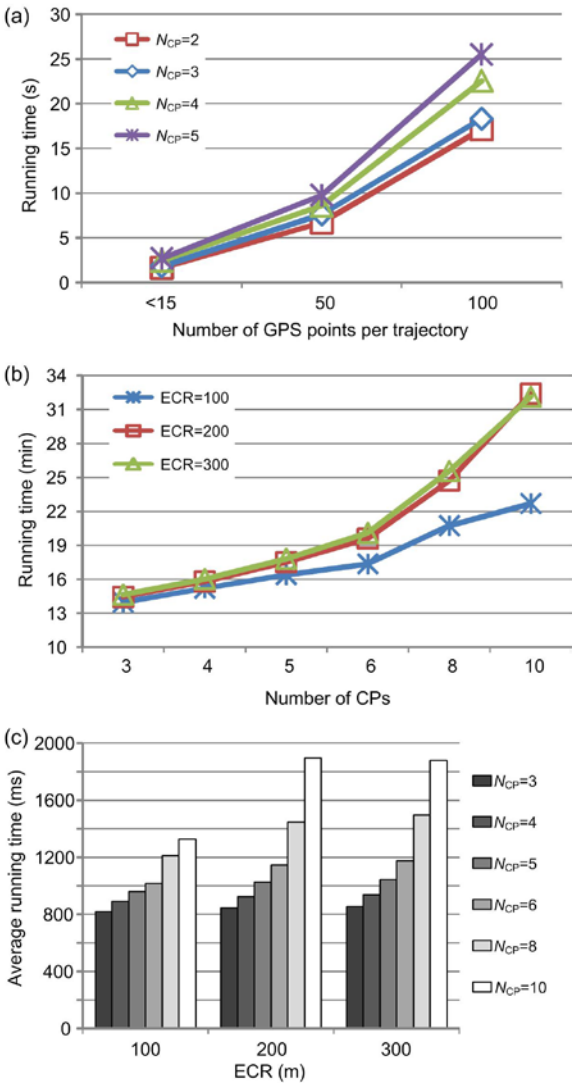


Fig. 9 Running time when varying the number of GPS points of trajectories (a) and the number of candidate points (N_{CP}) of a full-day trajectory (b), and the average running time of different numbers of candidate points of full-day trajectories with respect to different circle radii and numbers of candidate points

the percentage of total GPS sampling points.

On the other hand, Fig. 10c clearly shows that the proposed strategy reduces the numbers of SPQs and CPs required. Specifically, by employing our strategy, the total numbers of SPQs and CPs are 27% and 15% less than those of ST-M, respectively. The reason of the reduced number of SPQs is that the proposed strategy with interior tuning selects different classes of the locality of the road network when matching each GPS point, which unquestionably provides a faster processing.

On the other hand, we examined the distributions of GPS sampling points in each class when processing the full-day real-world trajectory (Table 3).

Table 3 clearly discloses that the classes based on the locality of the road network are adaptively selected by the proposed strategy for each GPS sampling point of the trajectory. The distributions of GPS sampling points in Class-I and II are dominant against the remaining classes. Because Class-I and II have small N_{CP} and ECR, the proposed adaptive strategy significantly reduced the number of SPQs and consequently the running time. Next, we analyzed the impact of the size of the sliding window on the proposed adaptive strategy. Fig. 11a shows the average running time of each sliding window size. Most of the sliding windows took less than 500 ms. Moreover, we statistically analyzed the impact of the size of sliding windows regarding the Box-and-Whiskers plot (Fig 11b). The first and third quartiles in the plot are between 0.4 and 0.8 s.

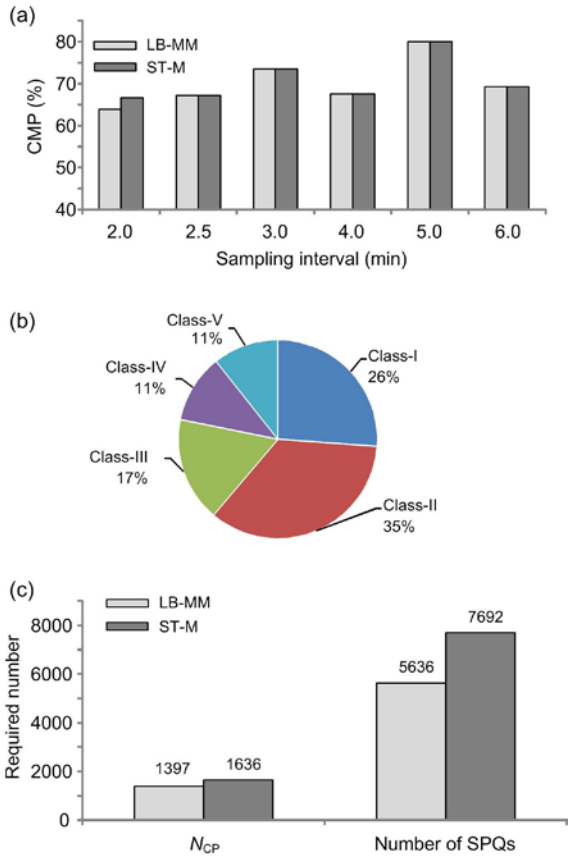


Fig. 10 Comparison evaluation of locality of the road network: (a) CMP with respect to different GPS sampling rates; (b) distribution of GPS sampling points in each class; (c) number of CPs and SPQs required

Table 2 The percentage of GPS sampling points

GPS sampling points	Number	Percentage
GPS sampling points used	1513	29%
GPS sampling points excluded	3801	71%
Total	5314	100%

Table 3 Distributions of GPS sample data in each class for matching a full-day real-world trajectory

Class	Number of GPS sampling points			Percentage
	Inside-city	High-way	Total	
Class-I	223	414	637	42.10%
Class-II	255	315	570	37.67%
Class-III	99	113	212	14.01%
Class-IV	24	23	47	3.11%
Class-V	15	32	47	3.11%
Total	616	897	1513	100.00%

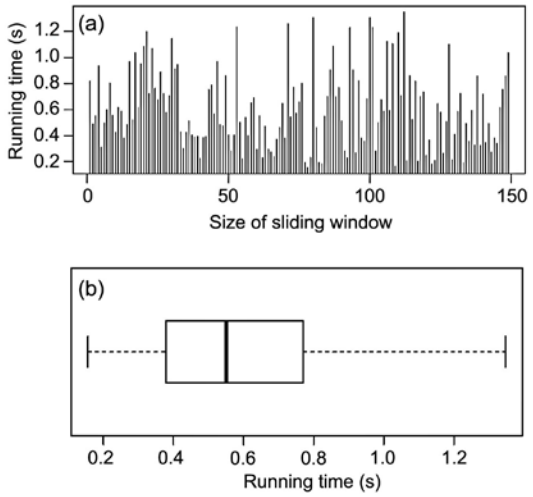


Fig. 11 Average running time for different sizes of the sliding window (a) and running time of each sliding window in the Box-and-Whiskers plot (b)

6.2.2 A comparison of static and adaptive parameters

Furthermore, we evaluated the performance of the adaptive strategy RT-M-II against the strategy RT-M-I with N_{CP} equal to 5 and ECR equal to 100 m. Both strategies were compared in terms of three metrics: (1) the total number of CPs, (2) the total number of SPQs, and (3) the total number of CPs found less than the fixed value of the CP parameter. The results are shown in Table 4, which indicates that RT-M-II outperformed RT-M-I. Specifically, the adaptive strategy significantly reduced the number of SPQs

Table 4 Performance comparisons between the adaptive strategy RT-M-II against the static strategy RT-M-I for full-day real-world trajectory map-matching

GPS point type	N_{CP}		N_{SPQ}		N_{CLF}	
	I	II	I	II	I	II
In-city	3145	1942	22 414	12 255	385	224
High-way	3350	2704	19 237	11 726	295	346
Total	6495	4646	41 651	23 981	680	570
Ratio	1.398		1.737		1.193	

N_{CP}: number of CPs; N_{SPQ}: number of SPQs; N_{CLF}: number of CPs less than the fixed value of the CP parameter. I: RT-M-I strategy; II: RT-M-II strategy

and CPs required. This is because the adaptive strategy considers different parameter classes for mapping each GPS sampling point of the full-day real-world trajectory. Particularly, assigning an appropriate class for mapping each GPS point can significantly reduce the total number of CPs and SPQs required.

7 Conclusions

It is evident that location-based services (LBSs) become more and more data hungry. In this paper, we proposed a fully adaptive map-matching strategy for real-time GPS trajectories based on the cloud environment. The proposed strategy adaptively fine-tunes the interior and exterior parameters of the map-matching process. The interior parameters, i.e., the number of candidate points (N_{CP}) and the error circle radius (ECR), are tuned based on the locality of a road network, while the exterior parameters depend on the sampling rate which is intelligently adjusted based on the feedback information of the monitored parameters at runtime. Furthermore, unlike traditional approaches, the shortest path distance and the speed constraint of road segments are pre-computed by following the bulk synchronous parallel (BSP) paradigm. In that way, the pre-computation time is reduced drastically. We analyzed the performance of our strategies in terms of (1) running time, (2) accuracy, (3) total number of SPQs, and (4) total number of candidate points (CPs) when applying real-world GPS data and synthetic data. Results revealed that, by assigning an appropriate class of the locality of a road network for mapping each GPS sampling point, the total number of CPs and SPQs can be significantly reduced, thus considerably decreasing the execution time. In our future work, we intend to extend our map-matching strategy to leverage on machine-learning techniques to adjust the interior and exterior parameters.

References

Brakatsoulas, S., Pfoser, D., Salas, R., *et al.*, 2005. On map-matching vehicle tracking data. Proc. 31st Int. Conf. on Very Large Data Bases, p.853-864.

Chandio, A.A., Zhang, F., Memon, T.D., 2014. Study on LBS for characterization and analysis of big data benchmarks. *Mehran Univ. Res. J. Eng. Technol.*, **33**(4):432-440.

Chandio, A.A., Tziritas, N., Xu, C.Z., 2015a. Big-data processing techniques and their challenges in transport domain. *ZTE Commun.*, **13**(1):50-59.

Chandio, A.A., Tziritas, N., Zhang, F., *et al.*, 2015b. An approach for map-matching strategy of gps-trajectories based on the locality of road networks. 2nd Int. Conf. on Internet of Vehicles, p.234-246. http://dx.doi.org/10.1007/978-3-319-27293-1_21

Chen, B.Y., Yuan, H., Li, Q., *et al.*, 2014. Map-matching algorithm for large-scale low-frequency floating car data. *Int. J. Geograph. Inform. Sci.*, **28**(1):22-38. http://dx.doi.org/10.1080/13658816.2013.816427

Chen, C., Liu, Z., Lin, W.H., *et al.*, 2013. Distributed modeling in a MapReduce framework for data-driven traffic flow forecasting. *IEEE Trans. Intell. Transp. Syst.*, **14**(1): 22-33. http://dx.doi.org/10.1109/TITS.2012.2205144

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. *Commun. ACM*, **51**(1): 107-113. http://dx.doi.org/10.1145/1327452.1327492

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. *Numer. Math.*, **1**(1):269-271. http://dx.doi.org/10.1007/BF01386390

Fang, S.K., Zimmermann, R., 2011. EnAcq: energy-efficient GPS trajectory data acquisition based on improved map matching. Proc. 19th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems, p.221-230. http://dx.doi.org/10.1145/2093973.2094004

Goh, C.Y., Dauwels, J., Mitrovic, N., *et al.*, 2012. Online map-matching based on hidden Markov model for real-time traffic sensing applications. 15th Int. IEEE Conf. on Intelligent Transportation Systems, p.776-781. http://dx.doi.org/10.1109/ITSC.2012.6338627

Gonzalez, H., Han, J., Li, X., *et al.*, 2007. Adaptive fastest path computation on a road network: a traffic mining approach. 33rd Int. Conf. on Very Large Data Bases, p.794-805.

Greenfeld, J.S., 2002. Matching GPS observations to locations on a digital map. Proc. 81st Annual Meeting of the Transportation Research Board, p.1-13.

He, Z.C., She, X.W., Zhuang, L.J., *et al.*, 2013. On-line map-matching framework for floating car data with low sampling rate in urban road networks. *IET Intell. Transp. Syst.*, **7**(4):404-414. http://dx.doi.org/10.1049/iet-its.2011.0226

Hu, H., Lee, D.L., Lee, V., 2006. Distance indexing on road networks. Proc. 32nd Int. Conf. on Very Large Data Bases, p.894-905.

Hummel, B., Tischler, K., 2005. Robust, GPS-only map matching: exploiting vehicle position history, driving restriction information and road network topology in a statistical framework. GIS Research UK Conf., p.68-77.

Kajdanowicz, T., Kazienko, P., Indyk, W., 2014. Parallel processing of large graphs. *Fut. Gener. Comput. Syst.*, **32**: 324-337. http://dx.doi.org/10.1016/j.future.2013.08.007

Kolahdouzan, M., Shahabi, C., 2004. Voronoi-based K nearest neighbor search for spatial network databases. Proc. 30th Int. Conf. on Very Large Data Bases, p.840-851.

Kühne, R., Schäfer, R., Mikat, J., *et al.*, 2003. New approaches for traffic management in metropolitan areas. Proc. IFAC CTS Symp.

Li, Q., Zhang, T., Yu, Y., 2011a. Using cloud computing to process intensive floating car data for urban traffic surveillance. *Int. J. Geograph. Inform. Sci.*, **25**(8):1303-1322. http://dx.doi.org/10.1080/13658816.2011.577746

Li, X., Han, J., Lee, J.G., *et al.*, 2007. Traffic density-based discovery of hot routes in road networks. Int. Symp. on Spatial and Temporal Databases, p.441-459. http://dx.doi.org/10.1007/978-3-540-73540-3_25

Li, Z.J., Chen, C., Wang, K., 2011b. Cloud computing for agent-based urban transportation systems. *IEEE Intell. Syst.*, **26**(1):73-79. http://dx.doi.org/10.1109/MIS.2011.10

Liu, K., Li, Y., He, F., *et al.*, 2012. Effective map-matching on the most simplified road network. Proc. 20th Int. Conf. on Advances in Geographic Information Systems, p.609-612. http://dx.doi.org/10.1145/2424321.2424429

Lou, Y., Zhang, C., Zheng, Y., *et al.*, 2009. Map-matching for low-sampling-rate GPS trajectories. Proc. 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems, p.352-361. http://dx.doi.org/10.1145/1653771.1653820

Malewicz, G., Austern, M.H., Bik, A.J., *et al.*, 2010. Pregel: a system for large-scale graph processing. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.135-146. http://dx.doi.org/10.1145/1807167.1807184

Newson, P., Krumm, J., 2009. Hidden Markov map matching through noise and sparseness. Proc. 17th ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems, p.336-343. http://dx.doi.org/10.1145/1653771.1653818

Pink, O., Hummel, B., 2008. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. Proc. 11th Int. IEEE Conf. on Intelligent Transprotation Systems, p.862-867. http://dx.doi.org/10.1109/ITSC.2008.4732697

Quddus, M.A., Ochieng, W.Y., Noland, R.B., 2007. Current map-matching algorithms for transport applications: state-of-the art and future research directions. *Transp. Res. Part C*, **15**(5):312-328. http://dx.doi.org/10.1016/j.trc.2007.05.002

Seo, S., Yoon, E.J., Kim, J., *et al.*, 2010. HAMA: an efficient matrix computation with the MapReduce framework. IEEE 2nd Int. Conf. on Cloud Computing Technology and Science, p.721-726. http://dx.doi.org/10.1109/CloudCom.2010.17

Tang, W., Ren, D., Lan, Z., *et al.*, 2013. Toward balanced and sustainable job scheduling for production supercomputers. *Parall. Comput.*, **39**(12):753-768. http://dx.doi.org/10.1016/j.parco.2013.08.007

Thomsen, J.R., Yiu, M.L., Jensen, C.S., 2012. Effective caching of shortest paths for location-based services. Proc. ACM SIGMOD Int. Conf. on Management of Data, p.313-324. http://dx.doi.org/10.1145/2213836.2213872

Tiwari, S., Kaushik, S., 2013. Scalable method for k optimal meeting points (k-omp) computation in the road network databases. Int. Workshop on Databases in Networked Information Systems, p.277-292. http://dx.doi.org/10.1007/978-3-642-37134-9_21

Wang, J.Z., Wang, Z.J., 2013. Architecture design of urban intelligent transportation using cloud computing. *Adv. Mater. Res.*, **605-607**:2549-2552. http://dx.doi.org/10.4028/www.scientific.net/AMR.605-607.2549

Wang, S., 2010. A cyberGIS framework for the synthesis of cyberinfrastructure, GIS, and spatial analysis. *Ann. Assoc. Am. Geograph.*, **100**(3):535-557. http://dx.doi.org/10.1080/00045601003791243

Wang, S., Liu, Y., 2009. TeraGrid GIScience gateway: bridging cyberinfrastructure and GIScience. *Int. J. Geograph. Inform. Sci.*, **23**(5):631-656. http://dx.doi.org/10.1080/13658810902754977

Wang, Z.Y., Du, Y., Wang, G., *et al.*, 2008. A quick map-matching algorithm by using grid-based selecting. Int. Workshop on Education Technology and Training and Geoscience and Remote Sensing, p.306-311. http://dx.doi.org/10.1109/ETTandGRS.2008.217

Wenk, C., Salas, R., Pfoser, D., 2006. Addressing the need for map-matching speed: localizing global curve-matching algorithms. 18th Int. Conf. on Scientific and Statistical Database Management, p.379-388. http://dx.doi.org/10.1109/SSDBM.2006.11

Yin, G.G., Xu, C.Z., Wang, L.Y., 2003. Optimal remapping in dynamic bulk synchronous computations via a stochastic control approach. *IEEE Trans. Parall. Distr. Syst.*, **14**(1): 51-62. http://dx.doi.org/10.1109/TPDS.2003.1167370

Yuan, J., Zheng, Y., Zhang, C., *et al.*, 2010. An interactive-voting based map matching algorithm. 11th Int. Conf. on Mobile Data Management, p.43-52. http://dx.doi.org/10.1109/MDM.2010.14

Zheng, Y., Wang, L., Zhang, R., *et al.*, 2008. GeoLife: managing and understanding your past life over maps. 9th Int. Conf. on Mobile Data Management, p.211-212. http://dx.doi.org/10.1109/MDM.2008.20

Zheng, Y., Capra, L., Wolfson, O., *et al.*, 2014. Urban computing: concepts, methodologies, and applications. *ACM Trans. Intell. Syst. Technol.*, **5**(3):38. http://dx.doi.org/10.1145/2629592