Intelligent Automation And Soft Computing, 2019
Copyright © 2019, TSI® Press
Vol.25,no.1,65–77

Simulation of Real-Time Path Planning for Large-Scale Transportation Network Using Parallel Computation

Jiping Liu[a,b], Xiaochen Kang[a,*], Chun Dong[a] and Fuhao Zhang[a]

[a]Chinese Academy of Surveying and Mapping, NO. 28 Lianhuachi West Road, Beijing 100830, China;
[b]Institute of Geographical Sciences, Henan Academy of Sciences, No. 64 Longhai Middle Road, Zhengzhou 450052, Henan, China

ABSTRACT
To guarantee both the efficiency and accuracy of the transportation system, the real-time status should be analyzed to provide a reasonable plan for the near future. This paper proposes a model for simulating the real-world transportation networks by representing the irregular road networks with static and dynamic attributes, and the vehicles as moving agents constrained by the road networks. The all pairs shortest paths (APSP) for the networks are calculated in a real-time manner, and the ever-changing paths can be used for navigating the moving vehicles with real-time positioning devices. In addition, parallel computation is used to accelerate the shortest path searching and vehicle navigation. The testing results suggest that considerable time reduction can be realized in comparison with the non-real-time computations. This finding demonstrates that the proposed model is useful in improving the efficiency of a large-scale transportation system.

KEY WORDS: Real-time path planning; Simulation; Urban transportation network; Parallel computation

1 INTRODUCTION

IT is well-known that point of interest (POI) and urban road networks have become the key data sources for vehicle navigation applications and location-based services (LBS) (Bakillah, et. al., 2014; B. Yang, et. al., 2014). POIs are often used to represent geographical entities, which are usually static locations (e.g., hospitals, restaurants, and hotels). The urban street networks play a great role in forming the city's structural skeleton and directly affect the city's transportation efficiency (Levinson, 2012). To a large degree, POIs are closely inter-related with the road networks. POIs are often used as the origin or destination locations in path finding and deemed as the data carriers of local life activities, whereas road networks are often used as routing system for searching the avenues represented by the POIs (B. Yang, et. al., 2014).

Identifying the optimal routes in the urban transportation network have always been an indispensable task for city people in their daily lives. Most people have to select an optimal route between home (or another starting place) and the office (or another target place). The optimal route is ever changing: an optimal route for a fixed position at this point in time may change at the next point in time. This situation occurs for many people every day. As a whole, the evolution of urban transportation is a group behavior that results from the interactions of the participating pedestrians and vehicles, instead of the simple aggregation of them. This finding means that the mutual influence between the participants is a basic property, and a congested road with more vehicles should be given less attention when walking by time (Shirabe, 2014). In addition, Martijn, et. al. (2007) suggests that the drivers' preferences should not be negligible. In general, these pervasive relations between vehicles can be understood based on Tobler's first law geography: all the locations on a geographic surface are related to each other, but the closer locations are more strongly related than more distant ones (Kim, et. al., 2014; Tobler, 1970). In other words, the dynamic interactions should be considered when collaborating the activities from the participants (Kang, 2015). Therefore, a local optimal route at any given time should take into consideration the surrounding road conditions.

The present study aims to develop an agent-based dynamic transportation model that is capable of capturing both the individual behaviors of the vehicles and the real-time states of the road segments involved

CONTACT Xiaochen Kang kxc2005@126.com

in the urban transportation network. To update the traffic status continually, the shortest path for each pair of locations is computed iteratively. Next, each vehicle can obtain the most optimal path dynamically at any given location. In this study, we adopt the classical Floyd-Warshall algorithm, which has been widely used in graph-searching algorithms for solving shortest-path problems for a network (Cormen, et. al., 2001) . In addition, the proposed model employs the vector geographic information system (GIS) (both data model and operations) at map level. This simulation was performed on an Nvidia GeForce GTX Titan Black card for the transportation network of Suzhou, which is a megacity in China. To assess the performance, the outcomes were compared to those of a non-real-time model in terms of the average traveling time.

2 RELATED WORK

AS stated by Wegener (1994), operational urban models are often built around the inter-relations between land use and transportation. The dynamic transportation network is an important component of the urban dynamics, and it plays a role in physical distribution of the moving vehicles. Therefore, an efficient transportation network is the key to improving urban efficiency. This section reviews the conventional technologies for modeling urban dynamics and the commonly used methods in searching for the optimal paths.

2.1 *Modeling of Urban Dynamics*

In a GIS, models can be static or dynamic. A model is static if the input and output both correspond to the same point in time; a model is dynamic if the output represents a later point in time than the input (Longley, et. al., 2005). Static models provide indices or indicators that can be used to predict impacts, sensitivities, or vulnerabilities. Dynamic models go further by attempting to project quantifiable impacts into the future and are used to assess different management or development scenarios (Castle and Crooks, 2006). Cellular automata (CA) and agent-based modeling (ABM) are two commonly employed techniques, both of which have significantly advanced the role of modeling and simulation in geography and regional science (Clarke, 2014). In particular, these two methods have been used in exploiting many aspects of urban dynamics, including urban land-use (Dahal and Chow, 2014), urban residential dynamics (Y. Chen, et. al., 2012), urban expansion (Batty and Xie, 1994), real-time vehicle navigation (Huang, et. al., 2007), etc. Many applications appeal to the idea of representing the spatial system on a regular lattice such as a grid, and then the dynamic change is thus conceived as change in the state of a cell. In other words, the function of a cell's states is designed by referencing the states of the nearest neighbors (Batty, et. al., 1999). The immediate association between the regular lattice and the raster-based representation in GIS suggests that the dynamics resulting from the spatial interactions between the cells can be easily realized in GIS. However, there are some limitations due to the irregular geometries in the real-world environments. Instead, the agent-based representation has a similar association between the discrete agents and the vector features (such as point, linear, polygonal features) , and it provides a natural method for describing and simulating a system composed of real-world entities (Castle and Crooks, 2006). In recent years, ABM has become a prevalent approach for modeling complex urban systems (Y. Chen, et. al., 2012). In comparison with CA, agent-based representations are centered around human actions, rather than landscape and transitions (An, et. al., 2005; Parker, et. al., 2003), and focus on how to determine agents' behavior using spatial information. Agent technologies and methods have been applied to many fields of traffic and transportation systems, including modeling and simulation, dynamic routing and congestion management, robotic moving, and intelligent traffic control (Alfaro and Riff, 2008; B. Chen and Cheng, 2010).

2.2 *Shortest Path Algorithms in the Transportation Network*

The shortest path problem lies at the heart of network flows that seeks for the paths with minimum cost from source node to sink node in networks (Liu, et. al., 2016; Sever, et. al., 2013; Xing and Zhou, 2011; Xu, et. al., 2007), and it is viewed as an issue of concern in computer science, operations research, distributed computing, bioinformatics and so on (Bonifaci, 2013; Y.-L. Chen and Yang, 2000; Kim, et. al., 2014; Wu, et. al., 2013). The shortest-path problem can be described as follows: given a weighted, directed graph and two special vertices, a shortest path from one vertex to another is a directed path with the property that no other such path has a lower weight. Consider a directed weighted network G = (V,E), where V is the set of vertices and E is the set of arcs. Each arc is denoted by an ordered pair (i,j), where (i,j) ⊆V. Suppose that there is only one directed arc (i,j) from i to j. Let the vertex s be the source and assume t as the destination. We define a path $p(i,j)$ as a sequence of alternating vertices and arcs. The shortest-path weight, also called the distance, from vertex s to t, denoted $d(s,t)$, is the minimum weight of all possible directed paths with origin s and destination t. Let $s \xrightarrow{p} t$ denote that t is reachable from s through the directed path p. We have

$$d(s,t) = \min \{ (p):\ s \xrightarrow{p} t \} \tag{1}$$

In this equation, the vertices in the graph correspond to the origin and destination venues for path finding in a road network.

When finding the optimal path for a traveling vehicle in the urban network, many classical algorithms can be used, such as the Dijkstra (Dijkstra, 1959), Bellman-Ford (Bellman, 1958; Ford and Fulkerson, 1963), Johnson (Johnson, 1977), and Floyd-Warshal (Floyd, 1962; Warshall, 1962) algorithms. These algorithms can solve three kinds of problems: one-to-one (or one-to-some) shortest paths, one-to-all shortest paths and all-to-all shortest paths (Zhan, 1997). The Dijkstra algorithm is suitable for solving the one-to-one and one-to-all shortest-path problems (Skiena, 1990), and the time complexity is $O(n^2)$ in general (Johnson, 1973). Herein, n is the number of vertices. The Bellman-Ford algorithm is also suitable for solving one-to-all shortest-path problems, and the time complexity is $O(mn)$. Herein, m is the number of edges. The Johnson algorithm and the Floyd-Warshal algorithm are both suitable for solving the All Pairs Shortest Paths (APSP) problem, and their time complexities are $O(n^2 lg(n)+mn)$ and $O(n^3)$, respectively. For the urban transportation network, the number of vehicles is often many times the number of roads and intersections. Therefore, performing a round of path finding for each vehicle is infeasible. Comparatively, the Floyd-Warshall and Johnson algorithms are more effective. The former is most effective for dense graphs (many edges), while the latter is most effective for sparse graphs (few edges).

2.3 Parallel Computing in Searching for the Optimal Paths

Most of these path searching algorithms exhibit high computational complexity for networks that are changing in real time. To cope with large-scale transportation network, parallel computing is widely used, which supports decomposing the task into many lightweight subtasks and then assigning them to the CPU or GPU cores. Chang, et. al. (1994) presented a user-optimum route assignment model for real-time navigation, and the results showed that parallel model significantly outperforms the sequential model in terms of computing speed, especially in a larger network. Rego and Roucairol (1996) described a parallel Tabu search algorithm for the vehicle routing problem; it was shown to be feasible when the number of tasks was fixed at compilation time. However, the degree of parallelism was not related to the computing resources and load imbalance could happen (Talbi, 2002). Alexandre and Gabriel (2005) proposed a parallel cooperative multi-search method based on the solution warehouse, which enabled several search threads to cooperate by asynchronously exchanging information on the best solutions that have been identified. This method achieved a linear acceleration, but was limited in terms of the number of vehicles and the total distance traveled. Z. Yang, et. al. (2007) presented an optimization model for a bus network design based on the coarse-grained parallel ant colony algorithm (CPACA), which used the parallelization strategies of an ant colony algorithm (ACA) to improve the calculation time. Zhang and Wu (2011) introduced a novel algorithm based on the Pulse Coupled Neural Network (PCNN) to solve the APSP problem, and it required no iteration and maintained low computational complexity. Wu, et. al. (2013) presented a hybrid programming model that combined MPI (Message Passing Interface) and CUDA (Compute Unified Device Architecture) to take full use of the GPU cluster, and a speedup of hundreds of times was achieved. However, the number of vertices and links is limited and real-time navigation, which requires large-scale shortest-path calculations and analysis, cannot be realized (Wu, et. al., 2013; Zhang and Wu, 2011).

2.4 Issues to Be Addressed

For a transportation network with tens of thousands road intersections and segments, millions of POIs play the roles of origin and destination venues for millions of vehicles. This property means that almost infinitely many computations should be performed in a very short period of time for determining the optimal path between each pair of interesting points. Therefore, two issues on model designing and performance optimization should be considered.

(1) The road conditions are always changing, and dynamic shortest paths that consider both internal and external factors perform better than static shortest paths (Chang, et. al., 1994). A way to represent the road networks, moving vehicles, and their relationships should be determined.

(2) Searching for optimal paths for millions of vehicles is still a big challenge due to the very high complexity. Path searching and vehicle navigation should be performed with very high efficiency through the use of HPC facilities.

In this paper, we propose an agent-based model for representing the real-time status of the road and the changing positions of moving vehicles. The road agents are static in locations, but dynamic in attributes. The vehicle agents are always moving along the road segments during the lifetimes, and serve as the driving forces for the transportation system. Moreover, the shortest path is used to calibrate the route choice for each vehicle, and the GPU-based Floyd-Warshall algorithm is adopted for its simplicity and obvious performance advantage in comparison with the CPU implementation (Katz and Kider Jr, 2008).

3 SIMULATION OF DYNAMIC TRANSPORTATION

In general, when to simulate a certain phenomenon using agent-based modeling, the agents and their interaction rules should both be identified. In this section, a vehicle movement model under road network constraints is presented. This model includes three kernel components: a network-based representation for modeling the road attributes and real-time conditions, an agent-based model for the moving vehicles, and the GPU-based Floyd-Warshall algorithm, which is aimed at searching for the shortest paths via road vertices for millions of origins and destinations.

3.1 *Representation of the Road Networks*

In the real world, the road network is made up of road segments and the road intersections. A segment has two categories of attributes: static attributes and dynamic attributes, as shown in

Table 1. The static attributes include the road identifier, names, grades, lengths, widths, directions and vitality. The dynamic attributes include the number of moving vehicles on a road segment, the real-time vehicle density, and traffic velocity.

In our daily lives, the road network connects interesting points with possible routes, and an origin venue and a destination venue are indispensable for defining a trip. Figure 1 is a road network layer with the affiliated POIs. Obviously, the POIs have enhanced the semantics of a particular road segment, and they can be integrated with a geometric-based method (B. Yang, et. al., 2014). Herein, each point is assigned to a road network by searching for the nearest road segment. For example, a primary school and a family home were assigned to corresponding nearest road segments. Then, two possible routes could be found. After integrating the road network and the POIs, searching for an optimal path in navigation is easily transformed into a search for the shortest path between each pair of sites.

Table 1. Attributes of a road segment in the transportation network

Attribute Type	Attribute List	Comments
Static Attributes	Road Identifier	The unique identifier of a road segment
	Road Name	The road name of a road segment
	Road length	The length of a road segment
	Road Width	The width of a road segment
	Road Area	The product of the length and the width of a road segment
	Road Vertices	The vertices that define the segment
	Number of POIs	The number of the POIs along a road segment
Dynamic Attributes	Number of vehicles	The number of vehicles on a road segment at a given time
	Vehicle Density	The number of vehicles per square meter on a road segment at a given time
	Vitality (Interval)	Being blocked or unblocked
	Traffic Velocity	The average running speed of vehicles
	Necessity	An particular road segment that must be passed
	Default option	Searching for the shortest path based only on the road conditions

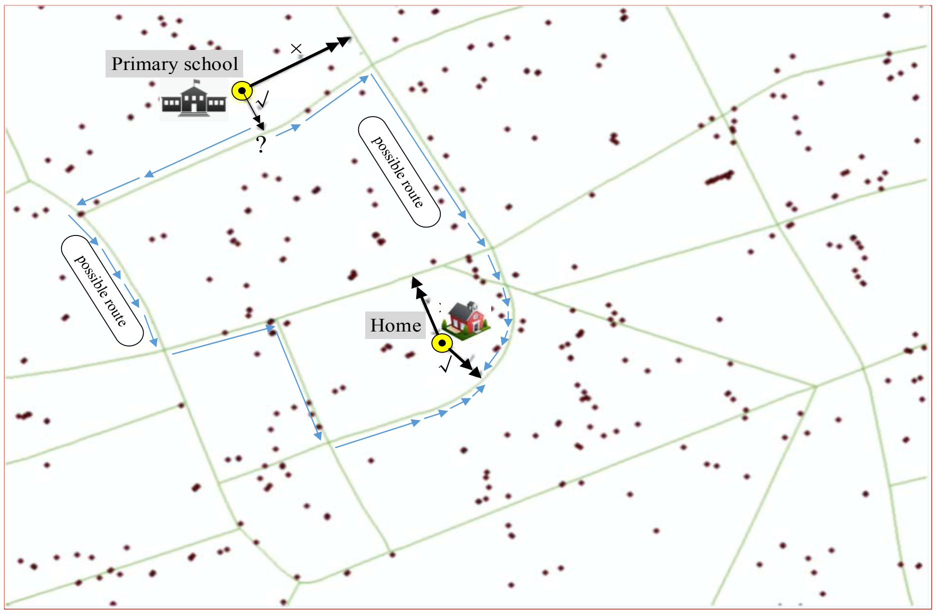

Figure 1. Integration of the POIs and the network

The streets are often crowded at peak times. As a result, the graph-based shortest path might be stuck due to the traffic jams. To simulate the road conditions in real time, the dynamic attributes listed in Table 1 should be given more attention. The static attributes are those factors that are constant during the entire simulation, while the dynamic attributes are those factors that change over time. In addition, the drivers' preference may also change during travel. For example, certain drivers may make temporary decisions regarding going someplace, which could potentially influence the road conditions and then result in some changes in other drivers' routes. In summary, taking the real-time conditions into account can contribute to obtaining a better traveling route (Mi and Liu, 2016).

3.2 *Agent-Based Representation of the Moving Vehicles*

A highly efficient transportation system requires the vehicles to reach their destinations in as little time as possible. Each vehicle' traveling time is determined by the static attributes and the dynamic attributes, and agent-based representation of the moving vehicles is proposed based on some major factors.

3.2.1 Modeling of the Network-Constrained Moving Vehicles

According to the car-following models, the average speed of the vehicles is a function of the vehicle density (Rothery, 2008; Takashi, 2002). As stated by Artimy, et. al. (2005), three main quantities are closely related: vehicle density, flow and speed. The density represents the number of vehicles per unit distance, the flow measures the number of vehicles that pass an observer per unit time, and the speed is the distance a vehicle travels per unit time. The proposed model focuses on how to utilize the real-time vehicle density to reduce the traveling time. Specifically, the road length l, the road width w, and the real time number of the vehicles n on a road segment are combined to approximate the density.

$$k = n/(l \times w) \tag{2}$$

where $l*w$ is the area of the road segment.

In the model, each vehicle is represented as an agent with the following static and dynamic attributes, which are listed in Table 2. For each vehicle, a unique identifier is needed. The origin and the destination are both determined by the road identifier and the distance to a road vertex (from which the distance to another vertex can be calculated). As the road conditions are changing with time, the present activity for each vehicle is determined by the previous conditions. Therefore, an update interval is set for updating the simulation result. The vitality attribute is used to mark the vehicle as dead (arrived) or alive (moving), and the real-time position indicates where the vehicle is located.

3.2.2 Interaction Rules among the Moving Vehicles

Except for the vehicle's performance and the driver's preferences, the speed of a vehicle is largely determined by the surrounding vehicles. According to our daily experiences, the speed of a vehicle is positively proportional to the vehicle density in a small local area and limited by the highway speed limits. Figure 2 shows the interaction radius, which defines the density units for the vehicles of interest (the black ones); their density units are highlighted. Each vehicle's interaction radius can be recognized by dividing the road segment into zones, such as the region *A*, *B*, *C* and *D* in the figure. In each unit, the vehicle of interest can directly interact with near vehicles that are located in the same unit. The resulting density is used to approximate the speed, and distant vehicles are not considered when computing the density.

When a vehicle crosses a road segment, the driver changes the speed according to the surrounding vehicle density. In other words, when updating the status of each vehicle, the neighboring vehicles should be continually monitored. In the simulation, each vehicle agent maintains a combined variable that records the real-time position (p) in a binary form.

$$p = [r, d] \tag{3}$$

where r is the road identifier, and d is distance from the road intersection.

Table 2. **Attributes of each vehicle agent in an update interval**

Attribute Type	Attribute List		Comment
Static Attribute	Vehicle Identifier		The unique identifier of a vehicle
	Origin	Road Identifier	The road identifier where the origin is located
		Linear Distance	The distance from the origin to one of the road vertices
	Destination	Road Identifier	The road identifier where the destination is located
		Linear Distance	The distance from the destination to one of the road vertices
	Update Interval		Time in seconds for performing an iteration in the simulation
	Interval Identifier		The number of iterations in the simulation
Dynamic Attributes	Vitality (Interval)		Being dead or active
	Position	Road Identifier	The road identifier where the vehicle is located at the current time

		Distance	The distance to the next road vertex in the optimal path

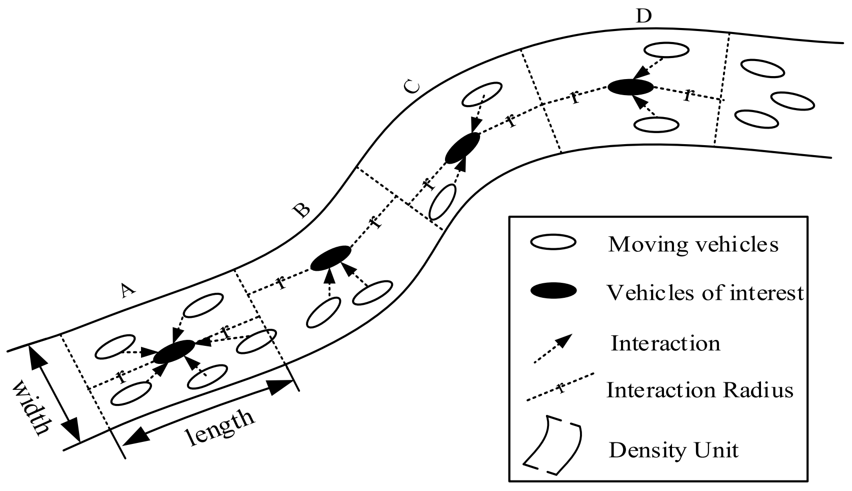

Figure 2. **The interactions among the vehicles**

3.3 *Modeling the Origins and Destinations*

A transportation network is naturally a graph with weights on the edges, and the numbers of road intersections and road segments usually range from several hundreds to tens of thousands. The number of vehicles that are active at the same time may range from tens of thousands to millions. For such a network, millions of paths should be computed when navigating all the vehicles at the same time. However, computing the shortest paths for all the vehicles in a short period of time is a great challenge. Each vehicle on a road segment has only two exits at the road intersections. In other words, all the vehicles on the same road segment will share the two road intersections as the origin or destination when searching for the optimal paths. In this way, the size of the problem can be reduced to the scale of the transportation network.

Figure 3 shows the two possible routes from the origins or towards the destinations. As the origin and the destination are both located in the road segments, four cases are derived through computing the shortest paths between road intersections: (1) *o→1→2→d*, (2) *o→1→2→4→3→d*, (3) *o→4→2→d*, and (4) *o→4→3→d*. When computing the shortest paths for millions of vehicles, multiple vehicles on the same road segment will share the two constrained vertexes as the exits. In this way, the problem of route planning for millions of vehicles is transformed into the APSP problem of a transportation network, and each shortest path will be one of the four possible paths.

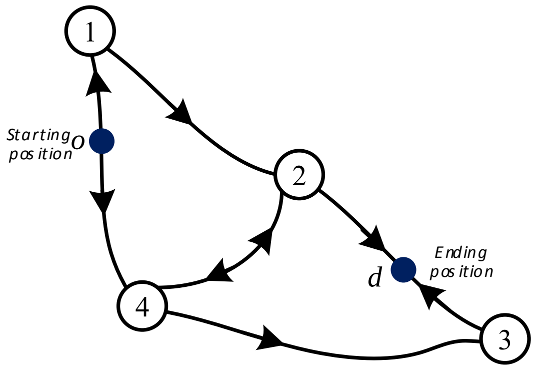

Figure 3. **Possible paths from origin (o) to the destination (d) in a road network**

3.4 *Using Floyd-Warshall Algorithm to Solve APSP Problem*

As the route planning problem for millions of vehicles is transformed into the APSP problem, each road segment may carry many vehicles for a period of time. In this way, the shortest path between each pair of vertices can be reused for the corresponding vehicles. Table 3 shows the pseudo-code of the commonly used APSP algorithm, namely, the Floyd-Warshall algorithm.

4 IMPLEMENTATION AND EXPERIMENT

In this section, we discussed the design and implementation of the proposed simulation model. In addition, the experimental result are introduced with detailed discussions.

Table 3. **Pseudo-code of the Floyd-Warshall algorithm**

Initialization	**let** dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity) **let** next be a |V| × |V| array of vertex indices initialized to null

APSP Computation	**Procedure Sequential_**
	    for each edge (u,v)
	        dist[u][v] ← w(u,v) , next[u][v] ← v
	    for k from 1 to |V|
	        for i from 1 to |V|
	            for j from 1 to |V|
	                if dist[i][k] + dist[k][j] < dist[i][j] **then**
	                    dist[i][j] ← dist[i][k] + dist[k][j], next[i][j] ← next[i][k]

Path Construction	**Procedure** Path(u, v)
	    if next[u][v] = null **then**
	        return []
	    path = [u]
	    while u ≠ v
	        u ← next[u][v]
	        path.appends
	    return path

4.1 *Implementation*

4.1.1 Matrix and Path Initialization

The popular ESRI shapefile format is a simple, non-topological format for storing the geometric location and attribute information of geographic features. The road network data in this format should be preprocessed for building adjacency relations. Moreover, a topological matrix is used for storing the relations by referencing the GPU computing model. The four steps are as follows.

(1) Vertex extraction and matrix building. When traversing each linear feature (road segment), the two vertices are first recorded. As the adjacent road segments have common vertices, some duplicate vertices should be removed. After locating all the incoming vertices in the set of vertices that have already been extracted, an adjacent matrix can be constructed with the road segments. In this study, an open-source library Geometry Engine - Open Source (GEOS) was used to computing the topological relations, and the value for any pair of vertices is set to true or false.

(2) Path construction. A road-segment-based network is constructed, which involves the static and dynamic attributes that were introduced in

Table 1. The static attributes are only related to the road, while the dynamic attributes are also related to the vehicles. Therefore, the dynamic attributes should be initialized and updated according to the moving vehicles.

(3) Vehicle generator construction. Vehicle generator is an interface layer that supports the generation of some simulated vehicles or actual vehicles along the road. The simulated vehicles are used to test the system, while the actual ones are used in real-time path planning through the reception of continuous GPS Signals from the actual vehicles. The attributes of the vehicles are introduced in

Table 2.

(4) Matrix initialization. The next step is to initialize the adjacency matrix according to the attributes of the road segments and initial transportation conditions, which serve as the initial weights of the matrix.

4.1.2 GPU-Based APSP Path Searching

After initialization, the raw network data and vehicle information are transformed into the adjacency matrix. To accelerate the path solving procedure, the matrix is first copied from host memory to GPU memory. Next, the operations on the data will be performed in parallel with the help of threads and blocks. After this, the resulting path matrix will be sent again from the GPU to the CPU. Table 4 shows the source code for the GPU-based Floyd-Warshall algorithm.

4.1.3 Iterative Framework

To support the simulation, a framework is implemented with three components: an initializer, an iterator and a display window (Figure 4). The initializer takes charge of the initialization of the program with all the initial conditions; the iterator drives the iteration; and the display window implements the visualization of the real-time status in each iteration.

Table 4. **The source code for the Floyd-Warshall algorithm**

Source code | **Note**

```
__global__ void _floyd_kernel (int k, int *d_g,int *d_p, int n)
{
      int thread_idx=blockIdx.x*blockDim.x + threadIdx.x;
      if(thread_idx >=n)  return;
      int idx=n*blockIdx.y+ thread_idx;
      __shared__ int best;
      if(threadIdx.x==0) tok=d_g[n*blockIdx.y+k];
           __syncthreads();
      if(tok ==INF) return;
      int fromk=d_g[k*n+ thread_idx];
      if(tmp_b==INF) return;
      int newdis= fromk + tok;
      if(newdis<d_g[idx]) {
          d_g[idx] = newdis;
          d_p[idx] = k;
      }
}
```

The kernel function of the Floyd-Warshall algorithm on GPU.
Input:
   d_g: adjacency matrix
   k: from 1 to n
   n: number of the vertices
Output:
   d_p: path matrix
Variables
   thread_idx: from 1 to n
   blockIdx.y: from 1 to n
   d_g[idx]: distance from thread_idx to blockIdx.y
   tok: distance from blockIdx.y to k
   fromk: distance from k to thread_idx
   newdis: the potential

```
dim3 dimGrid((n+1024-1)/1024,n);
for(int k=0;k<n;k++)
{
      _floyd_kernel <<<dimGrid, 1024>>>(k, d_g, d_p, n);
      err = cudaThreadSynchronize();
}
```

(n+1024-1)/1024: number of blocks
n: number of threads in a CUDA block

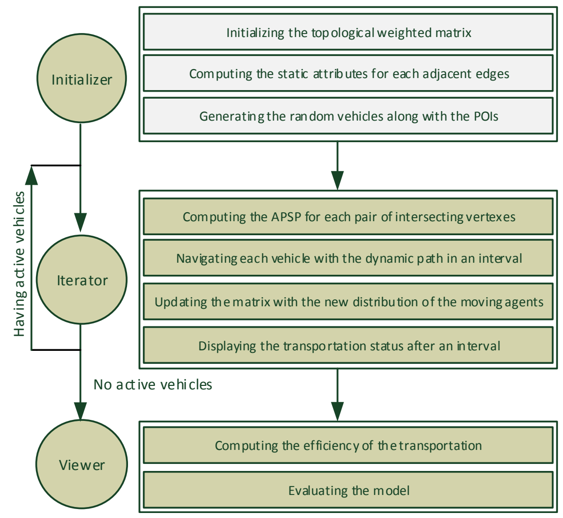
Initializer
Initializing the topological weighted matrix
Computing the static attributes for each adjacent edges
Generating the random vehicles along with the POIs
Having active vehicles
Iterator
Computing the APSP for each pair of intersecting vertexes
Navigating each vehicle with the dynamic path in an interval
Updating the matrix with the new distribution of the moving agents
Displaying the transportation status after an interval
No active vehicles
Viewer
Computing the efficiency of the transportation
Evaluating the model

Figure 4. **The iterative framework of the simulation**

The iterator works by looping through four steps: computing the APSP for the network, navigating each vehicle with the paths, updating the matrix with the changing attributes, and displaying the results. Among these steps, computing the APSP and the navigation of all the vehicles in their paths are computationally intensive. First, the GPU-based Floyd–Warshall algorithm is implemented on NVIDIA GPUs utilizing the CUDA programming API. When navigating all the vehicles in the same interval, different vehicles have no inter-independence. Therefore, the positions of the vehicles on the road segments can be updated simultaneously. Open Multi-Processing (OpenMP) is used to exploit the power of the multi-core architectures by adding the directive "#pragma omp parallel for schedule (dynamic)". In each interval, the display window only changes once. The results for each interval are stored as an independent file, which is directly visualized in GIS software.

4.2 *Testing and Discussions*

4.2.1 Testing Data

The study area (see Figure 5) is located in Suzhou City, China and it covers approximately 8488 km^2. The transportation network consists of 8,447 intersections (i.e., vertices) and 13,924 road segments (i.e., links). The study data are stored in ESRI Shapefile format.

4.2.2 Testing Environment

The experiment was conducted on a workstation running Microsoft Windows 7 (×64) with a GTX Titan Black GPU card (2880 CUDA cores and 6 GB GDDR5 memory), two Six-Core Intel Xeon E5645 processors (2.40 GHz in each core) and 16 GB DDR3-1333 ECCSDRAM. In addition, CUDA was used for solving the APSP problem in GPU, and OpenMP was used for accelerating the navigation of the vehicles.

4.2.3 Testing Results and Discussions

It is hypothesized that a shorter time interval will yield path planning results that are closer to the real-time results and, thus, a more efficient navigation strategy. However, shorter time intervals will require more computationally expensive simulations. Thus, the path planning work will be more expensive in terms of the time required to compute the paths and update the dynamic attributes, even when optimized with the parallel paradigms. The average traveling times in simulation with different updating intervals are shown in Table 5. The results include two groups of experiments: iterative updating without real-time computation and iterative updating with real-time computation. A more straightforward illustration is shown in Figure 6.

As shown in Table 5, the minimum time interval is set to 15 seconds, this is because performing a simulation for the testing data consumes about 14 seconds.

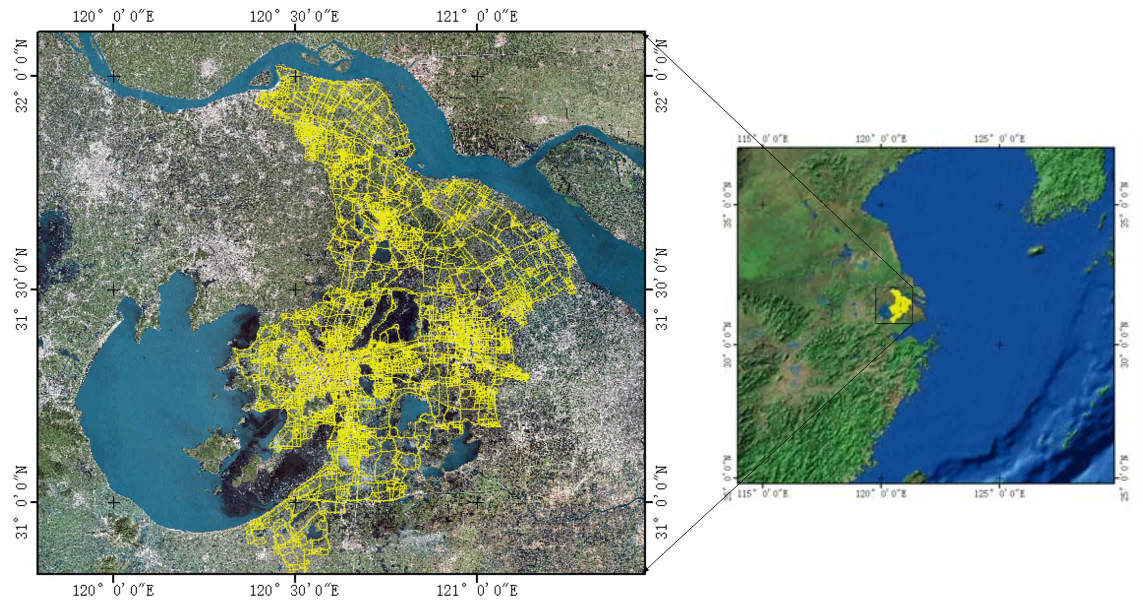

Figure 5. The study area: the road network of Suzhou City, China

Table 5. Average traveling times with different intervals: non-real-time computation and real-time computation

Interval (s)	Non-real-time computation (s)	Real-time computation (s)	Efficiency improvements
3840	2465	2456	0.4%
1920	3490	2642	24.3%
960	5832	3077	47.2%
480	6705	3586	46.5%
240	7151	3880	45.7%
120	7407	3988	46.2%
60	7551	4043	46.5%
30	7640	4110	46.2%
15	7714	4120	46.6%

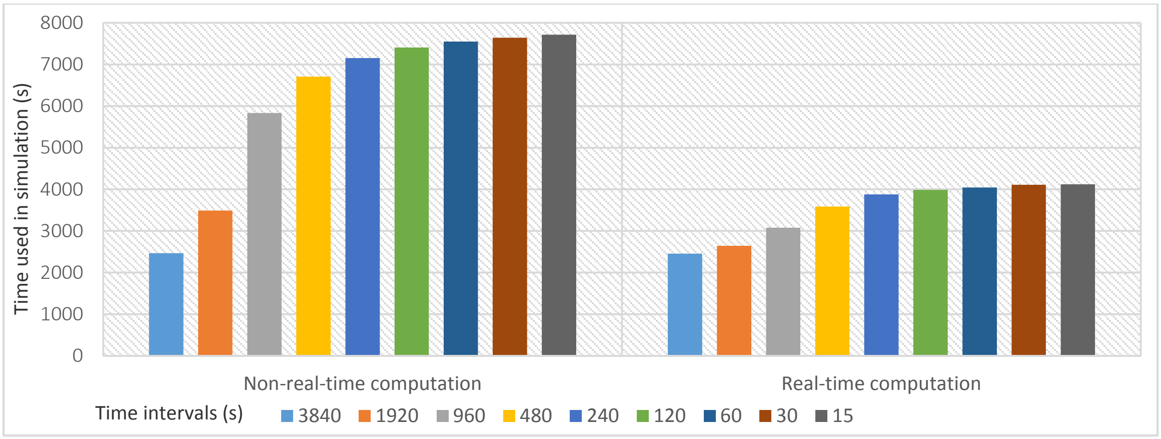

Figure 6. Illustration of the non-real-time computation and real-time computation

Due to the ever changing road conditions, more iterations with a shorter interval can contribute to a more realistic simulation in comparison with fewer iterations with a longer interval. In the non-real-time computation, the optimal paths were only calculated once according to the initial conditions. Then, all the vehicles traveled from the origins to the destinations in accordance with the established routes. In one iteration, each vehicle only updated its speed once according to the conditions after its last iteration. Therefore, with the decrease of the interval time, the simulation time increased gradually. When the interval time reached 120 seconds, the time consumption tended to a stable value. In the real-time computation, the optimal paths were calculated in each iteration, and the ever changing road conditions were used for updating the road weights in the next iteration. Similarly, when the interval time reached 120 seconds, the time consumption tended to a stable level.

In comparison, the real-time computation simulation only costs a smaller amount of time in each iteration, and an efficiency improvement of more than 45% can be achieved. As the result from each iteration was stored in a file, the vehicles in the simulation can be displayed in the developed viewer, as shown in Figure 7. The small yellow dots in each interval represent the vehicles. For the same interval, there were fewer vehicles in the real-time computation. This indicates that the average traveling time for real-time computation is shorter than that for non-real-time computation.

5 CONCLUSIONS

Transportation network is the foundation of the modern metropolis, and it determines the urban distribution, travel activities and development of the urban system. As a complex system, ubiquitous spatial associations exist among the participants, and spatial conflicts (e.g. traffic congestion) are commonly encountered. To improve the transportation efficiency and reduce the average traveling time, an agent-based model was proposed to represent the moving vehicles constrained by the networks. The road segments and the moving vehicles were modeled as interrelated objects with both static and dynamic attributes. Next, the mutual interactions among the vehicles were automatically analyzed in the individual units defined by the interaction radius. Based on the attributes, which are updated in each iteration, the vehicles could dynamically choose the optimal paths, and the CUDA-enabled Floyd-Warshall algorithm and OpenMP-based vehicle navigation were iteratively used to accelerate the simulation. Finally, the results of the experiment suggested that the average traveling time could be reduced by over 45%.

The primary contributions of this study include two aspects: (i) a representation of the moving vehicles and the road networks with static and dynamic attributes and (ii) agent-based model for simulating and navigating millions of vehicles that are constrained by transportation networks using parallel computation. In the near future, we will further investigate even larger networks from such cities as Beijing and Shanghai with GPS signals from the real environment.

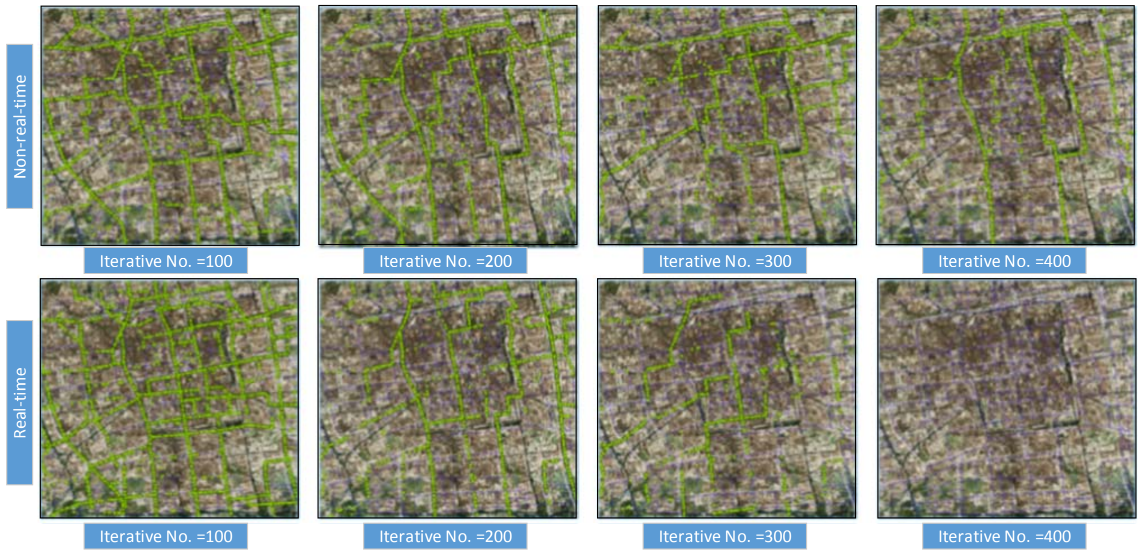

Figure 7. Displays of the non-real-time computation and real-time computation

6 ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China (Grant No. 41701461), the National Key Research and Development Program of China (Grant No. 2016YFC0803101) and the Basic Research Fund of the Chinese Academy of Surveying and Mapping (Grant No. 7771718). We thank the editors and the anonymous reviewers for their insightful comments which have helped to improve the quality of the paper.

7 REFERENCES

L. B. Alexandre, and Gabriel, C., Teodor. (2005). A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Computers and Operations Research, 32(7), 1685-1708.

T. Alfaro, and Riff, M.-C. (2008). An Evolutionary Navigator for Autonomous Agents on Unknown Large-Scale Environments. Intelligent Automation and Soft Computing, 14(1), 105-116.

L. An, Linderman, M., Qi, J., Shortridge, A., and Liu, J. (2005). Exploring complexity in a human–environment system: an agent-based spatial model for multidisciplinary and multiscale integration. Annals of the association of American Geographers, 95(1), 54-79.

M. M. Artimy, Robertson, W., and Phillips, W. J. (2005). Assignment of dynamic transmission range based on estimation of vehicle density. Paper presented at the International Workshop on Vehicular Ad Hoc Networks, Vanet 2005, Cologne, Germany, September.

M. Bakillah, Liang, S., Mobasheri, A., Jokar Arsanjani, J., and Zipf, A. (2014). Fine-resolution population mapping using OpenStreetMap points-of-interest. International Journal of Geographical Information Science, 28(9), 1940-1963.

M. Batty, and Xie, Y. (1994). From cells to cities. Environment and Planning B: Planning and Design, 21(7), 31-48.

M. Batty, Xie, Y., and Sun, Z. (1999). Modeling urban dynamics through GIS-based cellular automata. Computers, environment and urban systems, 23(3), 205-233.

R. Bellman, (1958). On a Routing Problem. Quarterly Appl Math, 16(1), 87-90.

V. Bonifaci, (2013). Physarum can compute shortest paths: A short proof. Information Processing Letters, 113(1–2), 4-7.

C. J. Castle, and Crooks, A. T. (2006). Principles and concepts of agent-based modelling for developing geospatial simulations. Paper presented at the Ucl Centre for Advanced Spatial Analysis, University College London: London, UK.

G. L. Chang, Junchaya, T., and Zhuang, L. (1994). A user-optimum route navigation model with a massively parallel computing architecture. Transportation Planning and Technology, 18(2), 107-129.

B. Chen, and Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. Intelligent Transportation Systems, IEEE Transactions on, 11(2), 485-497.

Y.-L. Chen,, and Yang, H.-H. (2000). Shortest paths in traffic-light networks. Transportation Research Part B: Methodological, 34(4), 241-253.

Y. Chen, Li, X., Wang, S., and Liu, X. (2012). Defining agents' behaviour based on urban economic theory to simulate complex urban residential dynamics. International Journal of Geographical Information Science, 26(7), 1155-1172.

K. C. Clarke, (2014). Cellular Automata and Agent-Based Models Handbook of Regional Science (pp. 1217-1233): Springer.

T. H. Cormen, Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to algorithms (Vol. 2): MIT press Cambridge.

K. R. Dahal, and Chow, T. E. (2014). An agent-integrated irregular automata model of urban land-use dynamics. International Journal of Geographical Information Science, 28(11), 2281-2303.

E. W. Dijkstra, (1959). A Note on Two Problems in Connection with Graphs. Numerische Mathematics, 1(1), 269--271.

R. W. Floyd, (1962). Algorithm 97: Shortest path. Communications of the Acm, 5(6), 345.

D. R. Ford, and Fulkerson, D. R. (1963). Flows in networks. Physics Today, 16(7), 54-56.

B. Huang, Wu, Q., and Zhan, F. B. (2007). A shortest path algorithm with novel heuristics for dynamic transportation networks. International Journal of Geographical Information Science, 21(6), 625-644.

D. B. Johnson, (1973). A Note on Dijkstra's Shortest Path Algorithm. Journal of the Acm, 20(3), 385-388.

D. B. Johnson, (1977). Efficient Algorithms for Shortest Paths in Sparse Networks. Journal of the Acm, 24(1), 1-13.

X. Kang, (2015). Graph-based synchronous collaborative mapping. Geocarto International, 30(1), 28-47.

G. J. Katz, and Kider Jr, J. T. (2008). All-pairs shortest-paths for large graphs on the GPU. Paper presented at the Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware.

J. Kim, Han, W.-S., Oh, J., Kim, S., and Yu, H. (2014). Processing time-dependent shortest path queries without pre-computed speed information on road networks. Information Sciences, 255, 135-154.

D. Levinson, (2012). Network structure and city size. PloS one, 7(1), e29721.

J. Liu, Xu, S., Zhang, F., and Wang, L. (2016). A hybrid genetic-ant colony optimization algorithm for the optimal path selection. Intelligent Automation and Soft Computing, 1-8.

P. A. Longley, Goodchild, M. F., Maguire, D. J., and Rhind, D. W. (2005). Geographic information systems and science: John Wiley and Sons: New York, NY, USA.

M. Martijn, Matthieu, V. D. H., and Aart, V. H. (2007). Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. European Journal of Operational Research, 181(1), 59-75.

B. Mi, and Liu, D. (2016). A fuzzy neural approach for vehicle guidance in real-time. Intelligent Automation and Soft Computing, 23(1), 13-19.

D. C. Parker, Manson, S. M., Janssen, M. A., Hoffmann, M. J., and Deadman, P. (2003). Multi-agent systems for the simulation of land-use and land-cover change: a review. Annals of the association of American Geographers, 93(2), 314-337.

C. Rego, and Roucairol, C. (1996). A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem: Springer US.

R. W. Rothery, (2008). Car following models. Trac Flow Theory, 199(3), 287–291.

D. Sever, Dellaert, N., van Woensel, T., and de Kok, T. (2013). Dynamic shortest path problems: Hybrid routing policies considering network disruptions. Computers and Operations Research, 40(12), 2852-2863.

T. Shirabe, (2014). A path that buys time to decide where to go. International Journal of Geographical Information Science, 28(2), 314-325.

S. Skiena, (1990). Dijkstra's algorithm: Reading, MA: Addison-Wesley.

N. Takashi, (2002). The physics of traffic jams. Reports on progress in physics, 65(9), 1331.

E. G. Talbi, (2002). A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics, 8(5), 541-564.

W. R. Tobler, (1970). A computer movie simulating urban growth in the Detroit region. Economic geography, 234-240.

S. Warshall, (1962). A Theorem on Boolean Matrices. Journal of the Acm, 9(1), 11-12.

M. Wegener, (1994). Operational urban models state of the art. Journal of the American Planning Association, 60(1), 17-29.

Q. Wu, Tong, C., Wang, Q., and Cheng, X. (2013). All-pairs Shortest Path Algorithm based on MPI+ CUDA Distributed Parallel Programming Model. Journal of Networks, 8(12), 2797-2803.

T. Xing, and Zhou, X. (2011). Finding the most reliable path with and without link travel time correlation: A Lagrangian substitution based approach. Transportation Research Part B: Methodological, 45(10), 1660-1679.

M. H. Xu, Liu, Y. Q., Huang, Q. L., Zhang, Y. X., and Luan, G. F. (2007). An improved Dijkstra's shortest path algorithm for sparse network. Applied Mathematics and Computation, 185(1), 247-254.

B. Yang, Zhang, Y., and Lu, F. (2014). Geometric-based approach for integrating VGI POIs and road networks. International Journal of Geographical Information Science, 28(1), 126-147.

Z. Yang, Yu, B., and Cheng, C. (2007). A Parallel Ant Colony Algorithm for Bus Network Optimization. Computer-Aided Civil and Infrastructure Engineering, 22(1), 44–55.

F. B. Zhan, (1997). Three fastest shortest path algorithms on real road networks: Data structures and procedures. Journal of geographic information and decision analysis, 1(1), 69-82.

T. Zhang, and Wu, L. (2011). A novel algorithm for APSP problem via a simplified delay pulse coupled neural network. Journal of Computational Information Systems, 7(3), 737-744.

8 NOTES ON CONTRIBUTORS

Jiping Liu received the Ph.D. degree in Cartography and GIS from the PLA Information Engineering University, China, in June 2004. Since 1999, he has been a professor in Chinese Academy of Surveying and Mapping. His current research interests include emergency geographical information service, geospatial big data analysis and spatial decision-making.

Xiaochen Kang received the Ph.D. degree in Cartography and GIS from Wuhan University, China, in June 2015. He is currently a research assistant at Chinese Academy of Surveying and Mapping. His principal areas of research are high performance geo-computation and geographical conditions monitoring.

Chun Dong received the Ph.D. degree in Population, Resources and Environmental Economics from Fudan University, China, in June 2006. Since 2012, she has been a professor in Chinese Academy of Surveying and Mapping. Her research interests include spatial data mining and geographical conditions monitoring.

Fuhao Zhang received the Ph.D. degree in Geographical Information System from Liaoning Technical University in June 2009. His research interests include e-government GIS, spatial data fusion and emergency geographical information services.