Simulation Modelling Practice and Theory 105 (2020) 102156

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

# Towards simulating non-lane based heterogeneous road traffic of less developed countries using authoritative polygonal GIS map

Kazi Hasan Zubaer[a], Quazi Mishkatul Alam[a], Tarik Reza Toha[b], Saiful Islam Salim[a,*], A.B.M. Alim Al Islam[a]

[a] Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh
[b] Department of Textile Machinery Design and Maintenance, Bangladesh University of Textiles, Bangladesh

ARTICLE INFO

*Keywords:*
Computational geometry
Transportation
Spatial-temporal systems
Modeling and simulation
Simulator
Interpreter
Traffic engineering algorithms

ABSTRACT

Microscopic traffic simulators have become efficient tools to conduct different analytic studies on roads, vehicles, behavior of drivers, and critical intersections, which lead towards a well-planned traffic solution. Devising a realistic and sustainable traffic solution requires replication of the real traffic scenario in a simulator. For example, to simulate the traffic stream of less developed countries, we need to simulate non-lane based heterogeneous traffic stream, i.e., motorized and non-motorized vehicles, with their on road behaviors such as irregular pedestrian, illegal parking, violation of laws pertaining lanes, etc. However, most of the existing traffic simulators are unable to mimic the unstructured road traffic stream of less developed countries with their diversified behaviors. We also need to represent the corresponding geographical maps as a list of nodes and edges with relevant geographic information such as length and width of the edges to feed the simulator. However, we lack a Geographic Information System (GIS) protocol to construct an effective simulator readable road network from authoritative polygonal GIS maps. State-of-the-art GIS protocol often converts the geographical maps using polyline data collected from less-credible OpenStreetMap. Although polyline data has length information (for example, length of a street), it lacks width information limiting its capability of being an effective simulator readable road network map. Therefore, in this work, we propose a new microscopic traffic simulator to handle unstructured road traffic stream of less developed countries with their diversified behaviors. To feed the simulator, we also propose a novel parsing algorithm to construct an effective simulator-readable road network from GIS maps through extracting both length and width information of the edges and develop parser based on the algorithm. Our parsing algorithm realizes a delicate blending of geometry with computation over road points to eventually generate the simulator-readable road network. Our simulator receives the network topology generated by our parser, traffic routes, and traffic demand flow rates as input, visualizes the traffic flows, and provides relevant traffic statistics. To evaluate sustainability of our proposed parser and simulator in real-life scenarios, we evaluate our parser by parsing GIS street maps of four different areas resulting in 86% accuracy on an average and evaluate our simulator after necessary calibration using real traffic data resulting in 99% accuracy in terms of travel time.

* Corresponding author.
*E-mail addresses:* zubaer.kh@gmail.com (K.H. Zubaer), 1105076.qma@ugrad.cse.buet.ac.bd (Q.M. Alam), toha@butex.edu.bd (T.R. Toha), 1018052067@grad.cse.buet.ac.bd (S.I. Salim), alim_razi@cse.buet.ac.bd (A.B.M.A.A. Islam).

https://doi.org/10.1016/j.simpat.2020.102156
Received 11 April 2020; Received in revised form 23 July 2020; Accepted 28 July 2020
Available online 30 July 2020
1569-190X/ © 2020 Elsevier B.V. All rights reserved.

## 1. Introduction

Microscopic traffic simulation has become an effective tool for evaluating and developing road traffic management and control systems all over the world [1]. It facilitates different analytic studies on roads, vehicles, behavior of drivers, critical intersections to develop a well-planned traffic solution for both urban and rural areas. Consequently, traffic simulation over different real-life traffic scenarios allows us to foresee potential consequences of taking any measure at infrastructure level or at policy level. Hence, in recent times, the usage of microscopic traffic simulators is increasing to investigate different transport planning and policy initiatives before actual field implementations. In order to devise a realistic traffic solution, we need to model real-world traffic stream to the simulation environment, which varies from country to country. The traffic stream of less developing countries such as Bangladesh, India, etc., contain both motorized vehicles and human powered vehicles on same road way, which results in an unstructured traffic system, commonly characterized as non-lane based and heterogeneous traffic system [1,2]. The fast moving motorized vehicles include bus, cars, auto rickshaws (Compressed Natural Gas), trucks, etc. On the other hand, the slow moving human powered vehicles include rickshaws, vans, and bi-cycles, which are different in driving behavior and exhibit lack of discipline following lane in particular. Furthermore, irregular road crossing by pedestrians and hawkers, illegal parking of vehicles, etc., are very common phenomena in the road networks of less developed countries. In addition to model traffic stream, we also need to model real road network, i.e., geographical maps in the simulator rather than hand-made road network. Normally, geographic maps are stored by Geographical Information System (GIS) with their position, shape, depth, color, etc. The geographic data can be represented in both raster format and vector format. Raster based data formats represent data of each pixel of the map, which is memory expensive. On the other hand, vector based data formats use geometric shapes such as points, lines, polylines, polygons, etc., to represent data, which is easily editable, maintainable, and analyzable. To model real geographical maps in a microscopic traffic simulator, we need to convert these maps to simulator readable road network, i.e., in graph format as a list of nodes and edges with their capacity.

In order to mimic the realistic outcome of any existing or future development on the road network of developing countries, we need to replicate the real traffic scenario of the developing countries in the simulator with their corresponding geographical maps. However, existing traffic simulation tools such as VISSIM [3], MITSIM [4], etc., primarily focus on simulating structured traffic. Hence, we fail to use them to mimic diversified traffic scenario of developing countries. Besides, there exist some open source simulators [5,6], which support only lane-based homogeneous traffic. However, we cannot extend them to support the real traffic scenario of less developed countries due to architectural differences, i.e., non-lane based concept is quite different to lane based concept. In order to model geographical map in a simulator, we have no straightforward mechanism to convert geographical maps to simulator readable maps, i.e., weighted graph. There exist several popular GIS data handling software such as ArcGIS, QGIS, etc., which provide some tools for extracting different features from standard GIS data. These software help us to find information such as nearest neighbors of a point, intersecting areas and points, union of different areas, etc. However, they do not provide any mechanism to represent the GIS data as a graph network. Existing literature addresses this lacking to solve real world problems such as vehicle routing problem [7,8], real-time localization [9], facilitating distribution of utility to the customers [10], characterizing urban road network [11], etc. Most of them use polyline data that represents the map as a list of unclosed points, i.e., end and start points are disconnected. Polyline data contains the length information of the edges, however, it lacks the width information of the edges that is required to make an effective weighted graph. For example, we cannot model the flow of vehicles in a road without knowing the capacity, i.e., width of the road. Moreover, they use the crowd-sourced GIS data of OpenStreetMap [12], which is not reliable [13–15].

As a remedy of the aforementioned problems in a sustainable manner, we have proposed a new microscopic traffic simulator in our previous study [16], which simulates the non-lane based heterogeneous traffic stream and on-road traffic behavior that generally occurred in the road networks of less developed countries. Our simulator takes network topology, traffic routes, and traffic demand flow rates as input, then it visualizes the traffic flows, and finally, provides traffic statistics such as average vehicle speed, total number of vehicles per road, and total number of accidents per road. Besides, in order to create readable road network by our simulator, in this work, we propose a new algorithm to abstract the authoritative GIS map as a graph structure as an extension of our previous study. We use polygonal vector data that represents the map as a list of closed points, i.e., the end and start points are connected. Polygonal data enables us to extract the approximate width and length information of the edges, which makes the graph workable for our simulator. Based on our work, we make the following set of contributions:

- We develop a traffic simulator to address non-lane based traffic by dividing a road into some thinner strips than lanes. Here, one vehicle can cover multiple strips and move along a random strip. We incorporate irregular movement of pedestrians and hawkers such as crossing the roads, walking along the roads instead of foot-path, etc., in our simulator.
- We devise a novel algorithm to convert the authoritative polygonal GIS data to a graph structure that is readable to the simulator. Our algorithm realizes a delicate blending of geometry with computation over road points to eventually generate the simulator-readable road network. We analyze time complexity of our algorithm and validate the theoretical run time against actual run time obtained through real experimentation. We also analyze the effects of different parameters of our algorithm on its performance and tune those parameters to get better accuracy.
- We develop a parser based on our devised algorithm and use it to parse the authoritative GIS street maps of four random areas in Dhaka city, the capital of Bangladesh, which reveals 86% accuracy on average. After that, we calibrate our simulator using real traffic data, i.e., traffic flow rate and average vehicular speed, and real geographical maps as weighted graph produced by our parser to evaluate the performance of our simulator. It results in 99% accuracy on average in terms of travel time between two intersections of Dhaka city.

## 2. Related work

Microscopic traffic simulation tools are increasingly being used around the world to test different transport planning and policy initiatives before their actual field implementations. For effective planning over a road network of a particular area, we need a simulator that meets the diverse behavior of road traffic of that area. In addition to mimic realistic traffic behavior, we also need to incorporate realistic road network maps in the traffic simulator for effective transportation planning. Now, we present some of the existing studies along with their drawbacks related to the realistic traffic simulation.

### *2.1. Literature survey on simulator*

There exist several simulators in literature and over the year they are updated to mimic the real-world traffic. Ferrer et al., [17] propose an interactive microscopic simulator named as AIMSUN for urban and non-urban networks. The latest version of AIMSUN includes mesoscopic and microscopic simulators and dynamic traffic assignment models based on either user equilibrium or stochastic route choice. However, it supports only lane-based and homogeneous traffic. Yang et al., [18] propose a microscopic traffic simulation model named as MITSIM that evaluates the impacts of alternative traffic management system designs, traveler information systems, public transport operations, and various ITS strategies at the operational level and assists in their subsequent refinement. However, MITSIM supports lane-based and homogeneous traffic only. Although an open-source version of MITSIM is available since 2004 [19], we cannot extend it to support non-lane based heterogeneous traffic due to the architectural difference, i.e., non-lane based concept is quite different to lane based concept. Fellendorf et al., [20] propose a microscopic behavior-based multi-purpose traffic simulator named as VISSIM to analyze and optimize traffic flows. It offers a wide variety of urban and highway applications, integrating public and private transportation. It is increasingly being used for state-of-the-art transportation planning and operations analysis [2]. The latest version of VISSIM supports heterogeneous traffic, however, it follows lane-based discipline. It also models car parking, moving pedestrian, etc., in a structured manner.

### *2.2. Literature survey on parser*

To model real geographical map of a road network in the traffic simulator, we need a conversion process to convert the geographical maps to simulator readable graph as a list of nodes and edges. There exist several conversion methods in literature that convert GIS map to graph in order to solve different real-world problems. Bennett et al., [10] propose a system to construct a directed graph from GIS data of utility distribution network such as electricity, gas, water, telecommunication, cable television, etc., to facilitate distribution of the utilities to customers. Gupta et al., [9] propose a method to construct a graph structure of GIS map and Building Information Modeling (BIM) to localize an individual or a vehicle that is moving inside (BIM) or outside a building (GIS) in real-time. Payne et al., [8] propose an algorithmic procedure for transforming GIS road map data into a topographical network for generating customized multi-day bicycle routes. In another study [7], they introduce a topographical network that enables minimum cost routing based on the changes in road gradient. Wilkie et al., [21] construct a lane-centric graph representation of roads to produce large-scale traffic simulations. Karduni et al., [11] propose a protocol to convert the geographical data to a graph network with relevant geographical information such as length and develop a ArcGIS tool to perform this protocol on GIS data of 80 urban ares of the world. However, most of the studies use polyline data to construct a graph, which lacks width information of edges that is required to build an effective simulator readable graph. For example, we cannot represent capacity of a road, i.e., amount of traffic supported by a road, without width information of an edge. Moreover, all of the studies collect their GIS data from OpenStreetMap (OSM), one of the most popular and well supported examples of Volunteered Geographic Information (VGI). The credibility of this crowd-sourced data is not enough to date, hence, researchers are devising new methods to assess the accuracy and consistency of OSM data [13–15].

To fill the gap in literature, we have proposed a new simulator in our previous study [16], which can model realistic traffic behavior of less developing countries such as non-lane based unstructured traffic with their on-road diversified behavior including irregular pedestrians on road, unwise parking on road, mobile market, etc. In order to represent corresponding geographical road network in the simulator, in this work, we propose a novel algorithm to construct a graph with width and length information of the edges from authoritative polygonal vector GIS data as an extension of the previous study. We develop a parser based on our devised algorithm to parse geographical network.

### *2.3. Literature survey on other related studies*

In addition to simulator and parser perspectives, there remain other related perspectives such as urbanization. Rapid urbanization and population growth have become a serious global issue in recent times. Rapid urbanization has caused significant changes in land use and urban growth patterns in metropolitan areas. Therefore, urban growth modeling has become an essential tool for the decision-makers to understand characteristics, assess sustainability, and find properties of current urban forms and population growth. In recent times, spatiotemporal modeling for urban applications has received especial attention.

Pooyandeh et al., [22] investigate different viewpoints in spatiotemporal modeling for urban applications such as CA-based modeling, agent-based modeling, Artificial Neural Networks modeling, fractal-based modeling, and temporal GIS modeling. They also review the applications of these approaches in modeling complex urban systems. Shen et al., [23] improve the spatiotemporal dynamic models for urban growth management by integrating Forretter's Urban Dynamics and Cellar Automation. Xu et al., [24]

perform an empirical study in the region of Munich. They test an integrated urban growth model to explain the current urban growth by combining a range of driving factors using auto logistic regression with a transition probability matrix from the Markov chain method in a cellular automata model simulation. Dadashpoor et al., [25] analyze the urban growth patterns and predict future growth using the SLEUTH model for Tabriz Metropolitan Area, Iran. They show that the edge growth pattern was dominant over the entire region while the dispersed growth pattern appears to be greater in some sub-regions. Terama et al., [26] present the methods and outcomes of integrating population projections across multiple spatial scales with an urban growth model. Xia et al., [27] use geospatial big data to measure urban relatedness and spatial interaction among city groups from a macro-level perspective. Chen et al., [28] propose to integrate an urban growth simulation model with the multi-region input-output (MRIO) model, and thereby illustrate how urban land consumption in one region can cause ecosystem services' degradation in another. They explore declination in ecosystem services due to urban land teleconnections under five shared socioeconomic pathways (SSP) scenarios. Fu et al., [29] propose a method to retrieve Impervious Surface Percentage (ISP) and to characterize urban growth patterns using time-series Landsat images. Zullo et al., [30] examined the relationship between Land Surface Temperature (LST) variations and increasing urbanized areas during the period 2001-2011 in Po Valley, utilizing different urban growth spatial patterns (UGSP). The implementation and calibration of key factors such as accessibility and suitability are very important in such urban growth modeling. Gonzalez et al., [31] propose a fully calibrated model to simulate urban growth namely MUGICA (Model for Urban Growth simulation using Irregular Cellular Automata).

With the beginning of rapid urban growth in the last few decades, severe climatic and environmental changes occur in developing countries such as Bangladesh, India, etc. The pace of rapid urban growth depends on several key driving economic factors such as industrial development, job opportunities, and promising a better lifestyle. Chandan et al., [32] attempt to correlate the dynamic land-use change, land cover, land surface temperature, and future urban growth scenario for one of the most systematically planned city of India, Chandigarh. To alleviate the problems of urban sprawl, Urban Growth Boundaries (UGBs) have been applied in many rapidly urbanizing areas. Chen et al., [33] propose a method that is capable of simulating UGB alternatives with economic and ecological constraints. Liang et al., [34] propose an urbanization economic growth model and a simultaneous equation model.

Transports play a vital role in the growth of such urbanization. Therefore, selection of the right transport impact indicators of urban growth is challenging. Besides the impacts vary both geographically and temporally. Perveen et al., [35] find 23 transport impact indicators from the literature, evaluate them, and classify their suitability in terms of space-time pair through a two-round of Delphi survey.

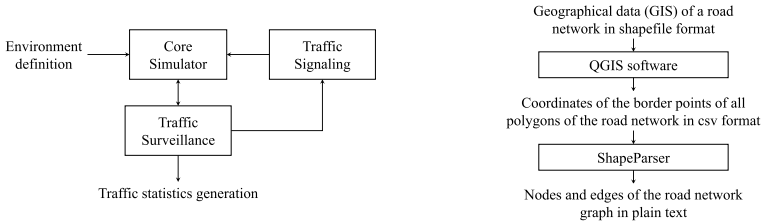
Besides, noise pollution possess a critical problem especially in the urban areas, where traffic noise is the foremost source of this pollution. Urban road networks significantly influence traffic noise. Paiva et al., [36] have performed a noise assessment with the creation of noise maps over the neighborhood in the city of Sao Paulo, Brazil. They perform a survey to ascertain the perception of the residents of the neighborhood concerning the effects of this exposure. Lu et al., [37] develop a structural equation model to explore the mediated effect of road characteristics on traffic noise through traffic flow using data from field measurement in Dalian City, China. For further analysis, they have also performed the paired comparisons of the scenarios through microscopic and macroscopic traffic simulations. Benocci et al., [38] propose Dynamap (DYNamic Acoustic MAPping) whose main scope is to implement real-time acoustic maps in a district of the city of Milan. Dynamap is based on the idea of finding a suitable set of roads displaying similar traffic noise behavior so that one can group them into single noise maps. Wang et al., [39] present an integrated geometric acoustics method to calculate both indoor and outdoor traffic noise distributions. Ahmed et al., [40] propose a neural network (NN) model to predict and simulate the propagation of vehicular traffic noise in a dense residential area at the New Klang Valley Expressway (NKVE) in Malaysia. Cai et al., [41] propose a method to assess population exposure to road traffic noise based on high-resolution population and grid-level noise data. Kuehnel et al., [42] propose to add scenario sensitivity for noise emissions in microscopic integrated land use/transport models. Bakowski et al., [43] simulate the changes in the noise measure and variation using the vehicle traffic measurements in both Nordic Prediction Method and Cnossos-EU models.

Traffic plays a significant impact in both urbanization and noise pollution, where several research studies have been conducted as mentioned above. The impact is more severe in developing countries such as Bangladesh, India, etc. Therefore, a dedicated traffic simulator in assessing the impact for the developing countries is required, where our simulator can play a baseline role. Methodology adopted in our simulator is amenable for such enhancements. We describe our proposed methodology behind the development of our simulator and parser in the next section.

## 3. Our proposed methodology

Our proposed simulator consists of three modules such as core simulator, surveillance, and signaling. We delineate the modules and their interactions in Fig. 1a. At first, we define the simulator environment, which describes the road network such as junctions and links, traffic flow rate, and routes and feed the definition to the core simulator. After that, the core simulator creates the environment according to given definition and generates heterogeneous traffic including pedestrians, motorized vehicles, and human-powered vehicles. Then, it simulates the traffic using their real-world behavior. While running the simulation over the user defined environment, the surveillance module collects data of current traffic from different parts of the road network and forwards it to signaling module, which imposes traffic signals to the simulator. When simulation time is over, surveillance module generates different statistics on simulated road network such as average number of vehicles, average speed of vehicles, accident count, etc., for each link.

In order to make the simulator environment realistic, we convert GIS data of a road network to simulator readable graph. Fig. 1b shows the high level overview of the conversion process. We devise a parser based on this process and we name it as ShapeParser. Our




(a) Modules of our proposed microscopic traffic simulator

(b) Abstract conversion process of polygonal GIS vector data of a road network to a graph

**Fig. 1.** Block diagram of our proposed methodology.

parser requires polygonal data of a road network in comma separated value (csv) format as its input. Here, polygonal data means the coordinates of the border points of all polygons. We use QGIS software [44], an open-source GIS application, to extract the co-ordinates of the border points in csv format from polygonal GIS vector data of a road network in shapefile (shp) format [45]. A shapefile contains geographical data such as coordinates, color, depth, etc., for Geographic Information System (GIS). QGIS extracts the border points in a closed loop for each polygon with their coordinates. Then, we feed the polygonal data to our parser and get nodes and edges of the road network graph with their coordinates in plain text as its output, which represent the intersections and corridors of the road network, respectively. Our parser also provides width of the edges as road capacity. In the next subsequent sections, we describe the detail architecture of our proposed simulator and parser.

## 4. Simulator implementation

We implement our simulator in Java language and we visualize the simulation using Swing library of Java. At first, we create the simulation environment by using some configuration files. Then, we simulate the non-lane based heterogeneous traffic stream with their on-road behavior in a loop until simulation time is over. The logical flow chart of our simulator is showed in Fig. 2. Now, we describe the main components of our simulator in detail.

### *4.1. Modeling road network*

Our simulator loads the simulation environment from five configuration files, which describe the information of junctions, links, traffic demand, routes, and some configuration parameters of the simulator. Since the road network is represented as a graph data structure in the simulator, the junctions are nodes and links are edges of the graph. Therefore, junctions and links define the road network. Besides, traffic route and demand describe the traffic flow rate for a particular path in the road network. We create a road using a link, which can be subdivided by some segments to represent the curvature of a road. Each segment is further divided laterally into some thin strips, which act as virtual lanes for non-lane based heterogeneous traffic stream. A vehicle occupies a number of strips based on its type. Each link is sandwiched by a foot-path on each side which also occupies some strips. Fig. 3 demonstrates the arrangement of strips in a curved road. Now, we describe the input configuration files briefly.

#### *4.1.1. Junctions*

Junctions represent the intersections of the roads. In the configuration file of junctions, we describe a junction with its unique id, Cartesian co-ordinate, and a list of adjacent links. The input configuration file of junctions is showed in Fig. 4a. The first line denotes the number of junctions of the given road network.

#### *4.1.2. Links*

Links define the actual roads or corridors in the simulation environment. In the configuration file of links, we describe a link with its unique id, id of two adjacent nodes, and number of constituent segments. After each link description, we describe the information of all segments that belong to a link. We describe a segment with its unique id, Cartesian co-ordinates of two end points, and width of the segment. Fig. 4b shows the format of a link file. The first line denotes the number of links of the given road network.

#### *4.1.3. Traffic routes*

Traffic routes are the paths between source and destination nodes in the graph. We describe a traffic route with the source node, destination node, and the list of links. We demonstrate a route file in Fig. 4c. Here, the first line denotes the number of routes in the file. We use bidirectional traffic, hence, each entry is repeated in reverse direction. Traffic generates from source node and starts from the first link of the route. Conversely, it arrives at the last link of the route and enters into destination (sink) node.

#### *4.1.4. Traffic demand*

Traffic demand represents the traffic flow rate of a route between a source and sink pair of nodes in the network. We describe

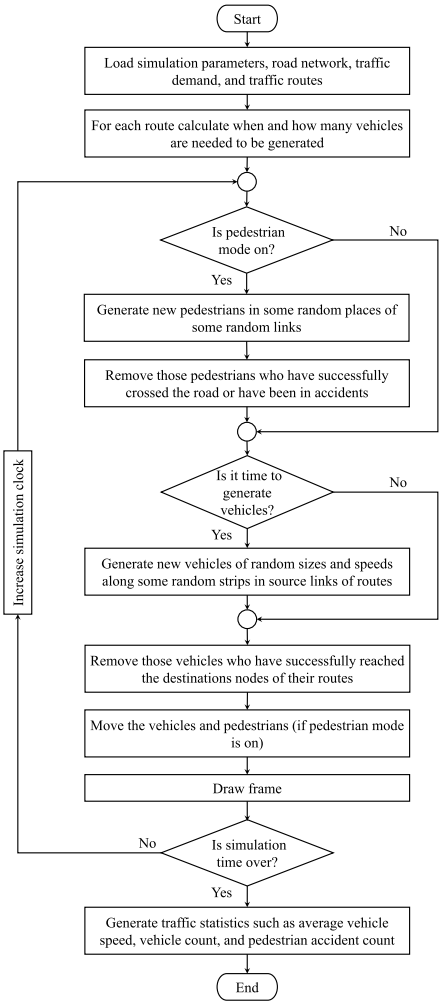



**Fig. 2.** Flow chart of our proposed microscopic traffic simulator.

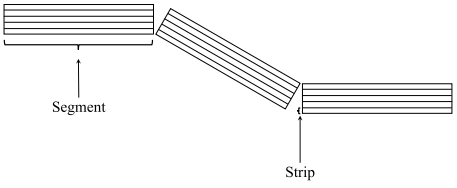



**Fig. 3.** Strip arrangement of a curved road (link).

```
21
0 0 0 16 18 17 14
1 317 26 4
2 436 20 19
3 675 14 20
4 990 0 18
5 1204 95 17
6 1138 413 13
7 955 767 22
8 643 750 0
9 428 766 6
10 126 760 2
11 55 419 1
12 0 0 15 14 13 12
13 0 0 10 15 11
```

(a) Format of the junction file

```
23
0 8 16 1
0 618 730 591 567 10
1 11 18 1
0 67 403 161 409 10
2 10 18 2
0 110 735 134 656 10
1 137 647 175 440 10
3 18 19 1
0 185 410 289 260 10
4 19 1 1
0 295 225 293 46 10
5 19 15 2
0 356 257 469 358 10
1 484 368 563 377 10
```

(b) Format of the link file

```
12
11 9 1 21 6
9 11 6 21 1
1 10 4 3 2
10 1 2 3 4
7 4 22 12 14 18
4 7 18 14 12 22
8 3 0 8 9 20
3 8 20 9 8 0
6 1 13 15 10 5 4
1 6 4 5 10 15 13
2 5 19 16 17
5 2 17 16 19
```

(c) Format of the route file

```
12
11 9 1800
9 11 1800
1 10 1500
10 1 1500
7 4 1200
4 7 1200
8 3 2000
3 8 2000
6 1 3600
1 6 3600
2 5 1000
5 2 1000
```

(d) Format of the demand file

```
SimulationSpeed 100
SimulationEndTime 10000
PixelPerStrip 3
PixelPerFootpathStrip 3
PixelPerMeter 6
EncounterPerAccident 1
StripWidth 0.5
FootpathStripWidth 0.5
MaximumSpeed 50.0
PedestrainMode Off
```

(e) Format of the parameter file

**Fig. 4.** Different input configuration files of our simulator.

traffic demand of each route with its source node, sink node, and flow rate in vehicles per hour. Fig. 4d shows a demand file. Here, the first line denotes the number of demand entries in the file. As we use bidirectional traffic, each demand entry is repeated in reverse direction.

#### *4.1.5. Simulation parameters*

We provide some additional simulation parameters to customize the environment such as simulation speed, simulation end time, pixel per strip, pixel per footpath strip, pixel per meter, encounter per accident, strip width, footpath strip width, maximum speed of the vehicles, and state of pedestrian mode. Here, one unit linear distance is equivalent to one meter and one simulation step is equivalent to one minute. Besides, encounter per accident is the probability percentage of being an accident. Fig. 4e shows the parameter file. These parameters can also be given as input from GUI. After creating simulation environment, we calculate when and how many vehicles are needed to be generated from a probability distribution. We can mimic a real road environment by providing geographical road network and real survey data of traffic demand, routes, accident probability, maximum speed of vehicles, etc. We can also add time dependency on these parameters to make the environment more realistic.

### *4.2. Modeling pedestrians*

After modeling road network, we have to model the pedestrians and their behavior in our simulator. In developing country, pedestrians move across the roads from anywhere. Besides, a pedestrian may have an accident with a running vehicle while crossing the road. We generate new pedestrians in some random places of random links and the probability distribution of pedestrian generation is completely uniform. We remove those pedestrians who have successfully crossed the road or have been in accidents. Note that, our simulator has an option to turn off the pedestrian model, which we provide as input during road network modeling.

Actually, we create the pedestrian as a generic object to provide additional functionality to make the simulator more realistic. For example, we can make some static pedestrians on the border sides of a road to mimic the street hawkers, crowds around mobile markets, etc., on the footpaths.

### *4.3. Modeling vehicles*

This component is the core part of our simulator. In developing countries, different types of vehicles run in the roads such as motorized and non-motorized, i.e., human powered. As they are different in terms of width, length, speed, etc., they cannot maintain a lane structure while moving along the roads. We model the non-lane based heterogeneous traffic by using different sizes of vehicles with different speeds. We model the real-world drivers' behavior by imitating real-world behavior, i.e., any vehicle can move along any strip of the roads. According to traffic demand, we generate different types of vehicles on different source nodes of the routes. Vehicle can switch between strips according to real-world scenario. If one vehicle wants to overtake another vehicle, first it checks whether its right strips have any space to overtake. If it finds empty, it overtakes another vehicle. Otherwise, it again checks whether its left strips have any space to overtake. Thus, the vehicles show real-world behavior while moving along the strips. As soon as the vehicles reach the destination nodes of their routes successfully, we remove them from the network. In order to mimic the parked vehicles on the roads, we can create some static vehicles on the border sides of some roads.

### *4.4. Graphical user interface*

The usability of a traffic simulator depends on interactivity. Hence, in order to make our simulator more user friendly, we provision for graphical interactions along with the configuration files. Our simulator visualizes the simulation in a graphical user interface (GUI). One such simulation is illustrated in Fig. 5. Here, the colored rectangles and green dots represent the vehicles and pedestrians respectively. We can see that different sizes of vehicles are congested in the three source links. They are waiting for green signals from traffic signaling module. We can also see that the vehicles are highly congested in the middle of the roads and the congestion decreases with the increase of lateral distance across the road. There is no lane-structure in the roads and vehicles try to

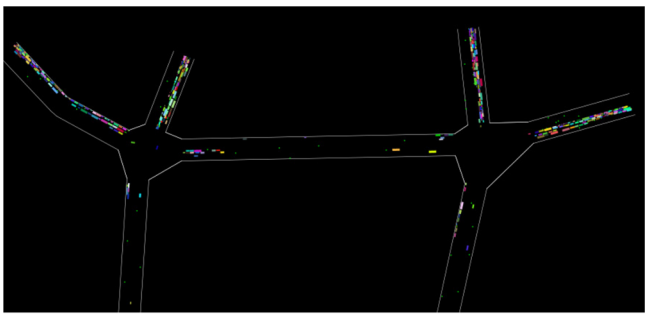


**Fig. 5.** Visual representation of our simulator (colored rectangles are vehicles, green dots are pedestrians). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

```
start
object
4692 8365 false
4944 7675 false
vehicle
933 984 931 983 4847 4851 4868 4871
4782 4731 4808 4757 9098 9010 9083 8995
3669 3670 3693 3694 583 653 583 653
1351 1367 1371 1387 8728 8676 8734 8682
11124 11120 11140 11136 8887 8856 8885 8854
11772 11775 11781 11784 152 177 151 176
7414 7405 7434 7426 8727 8676 8723 8673
7853 7841 7867 7856 395 425 401 430
13209 13158 13209 13158 4784 4784 4805 4805
3684 3685 3705 3706 567 622 567 622
5665 5700 5650 5685 312 350 326 364
13874 13853 13876 13855 1083 1088 1089 1095
end
```

**Fig. 6.** Format of the simulator log file describing the state of pedestrians and vehicles for each simulation step.

move everywhere when they get spaces. Hence, this scenario delineates the real non-lane based heterogeneous traffic stream, which resembles a common scenario in the road networks of cities in developing and underdeveloped countries. Since GUI mode consumes much CPU and memory, we provide an option to simulate the traffic in console mode. The console mode is faster than GUI mode. Consequently, at the end of simulation, different traffic statistics are generated.

### 4.5. Simulator log

We save the attributes of the pedestrians and vehicles for each simulation step in a log file to replay the simulation. A typical log file is shown in Fig. 6. For each step, the log starts and ends a simulation step with a *start* and an *end* keywords respectively. The information about pedestrians and vehicles is started with an *object* and a *vehicles* keywords respectively. The log describes the information regarding a pedestrian with the co-ordinate of the pedestrian and status of the pedestrian whether (s)he is in accident or not. On the other hand, the information regarding a vehicle is described with the co-ordinates of four corner points as following format:

$$x_1 \quad x_2 \quad x_3 \quad x_4 \quad y_1 \quad y_2 \quad y_3 \quad y_4$$

Here, $(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$, and $(x_4, y_4)$ represent the top-left, top-right, bottom-left, and bottom-right corner points of the vehicle.

### 4.6. Statistics generation

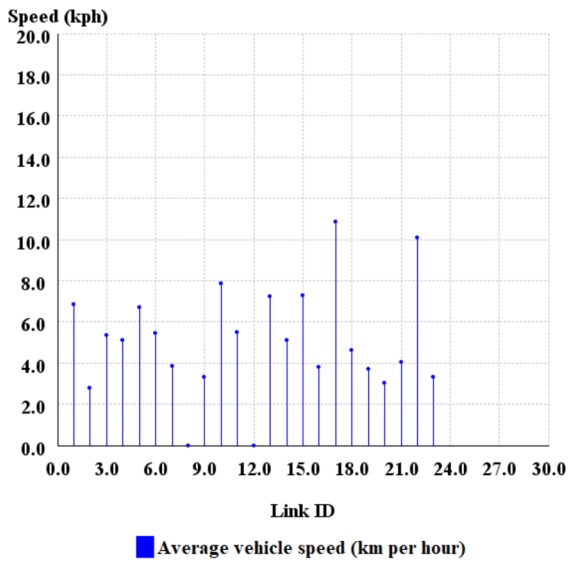
In order to simulate the surveillance system, we place one sensor in a random place of per segment. The sensors gather data of the number of vehicles that have passed over the sensor and measure the speed of the passed vehicles. It also collects accident information on every segment. When the simulation time is over, the surveillance module plots the graphs of average number of vehicles, average speed of vehicles, and average accident count for every link. The traffic statistics of a small road network is shown the Fig. 7. Here, we use traffic demand of 100 vehicles per hour (vph), maximum allowable speed of the vehicles 20 km per hour (kph), and accident probability of 0.1%. Now, we describe the implementation of our parser to convert GIS data to graph.

## 5. Parser implementation

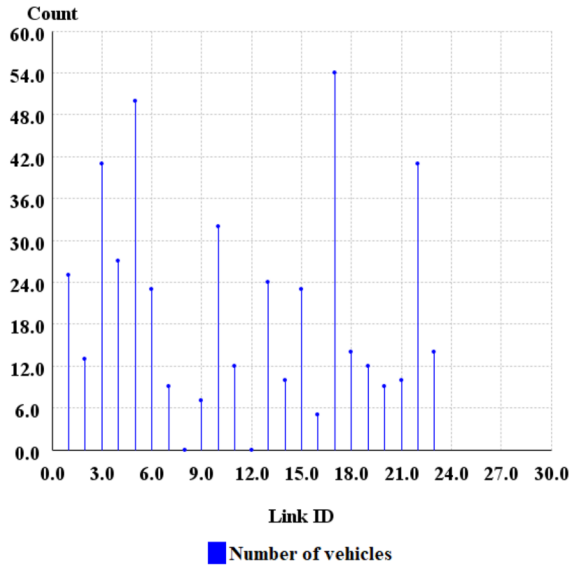
We implement our parser in Java language by adopting modular approach. Fig. 8 shows the logical flow of our parser. Our parser parses the polygonal data to graph over five sequential phases. At first, our parser takes the coordinates of border points as input and produces valid polygons after removing unnecessary polygons such as road divider. After this pre-processing step, we generate all points inside the roads. Then, we detect all points in junctions and dead ends of the road network. After that, we calculate the coordinates of nodes including junctions and dead ends, which are the nodes of our desired graph. Next, we detect the connections between the nodes, which produce the edges of the graph. Finally, we calculate the coordinates of these edges and write the nodes and edges in a plain text file with their coordinates. Thus, we get our desired graph of the given road network. Now, we describe the working process of our parser in detail.

### 5.1. Valid polygon extractor

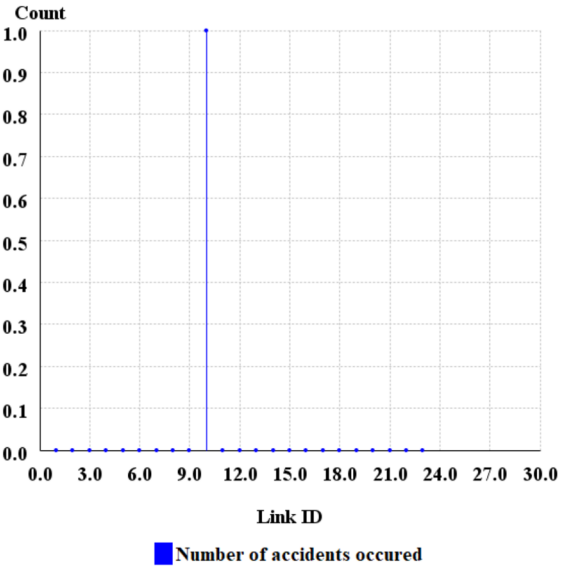
This module reads polygonal input data that contains the coordinates of border points of all polygons in a road network. The border points of a polygon appear sequentially in the input file (Fig. 9a), however, the polygons do not follow any order. The main task of this module is to perform some pre-processing tasks such as eliminating close points and road dividers. The distance between border points are non-uniform. For example, a lot of points is needed to represent a curvature (Fig. 9b), whereas only two points are needed to represent a straight line. In order to reduce complexity of our algorithm, we consider uniform spaced points only. On the other hand, all polygons do not necessarily represent roads, some represent area not included in any road (Fig. 9c). Besides, some




(a) Average vehicle speed in km per hour over all links




(b) Total number of vehicles over all links




(c) Total number of accidents over all links

**Fig. 7.** Traffic statistics for each link during simulation time.

narrow polygons represent road dividers (Fig. 9d) that are irrelevant to our desired graph. Therefore, we need to remove the dividers from our road network.

In order to detect narrow polygon, we use surveyor's area formula [46], also known as shoelace formula, to calculate area of the polygon by given Cartesian coordinates of the border points. Then, we divide the area by the maximum distance between the points. Thus, we get the approximate width of the polygons. If the width is less than an empirical threshold, we mark it as narrow polygon. If a narrow polygon is closed under another polygon, this polygon is actually a divider and we remove it. Note that, there is no intersecting polygon in GIS data. Thus, this pre-processing module cleans the polygonal data of road network and returns valid polygons only. We also calculate local and global minimums and maximums coordinates from all valid polygons for future usage.

The main tasks of this module are border points extraction, close points elimination, and divider elimination. The time complexity of border points extraction depends on the number of points in input file ($n$) and eliminating close points depends on the number of

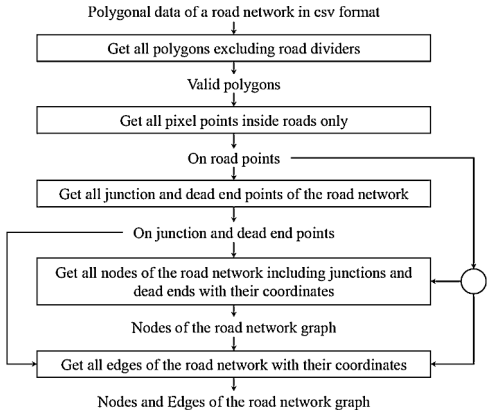
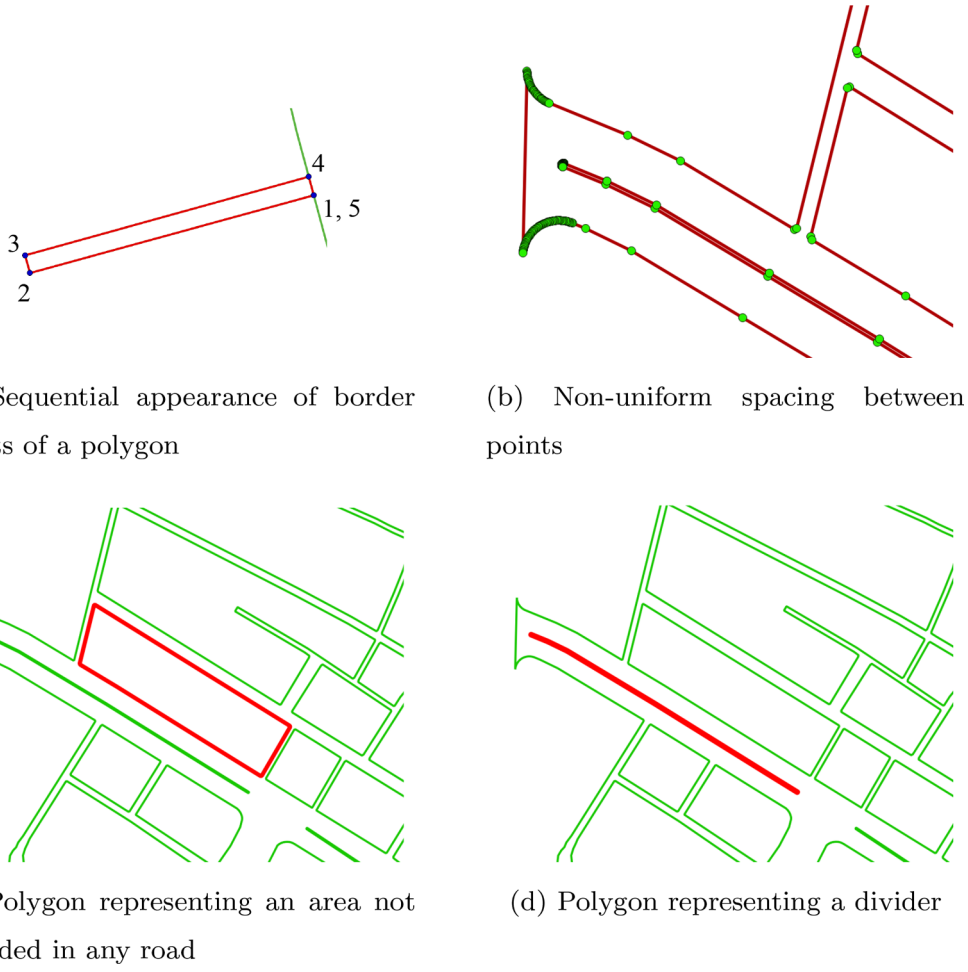



**Fig. 8.** Detail conversion process of polygonal data of a road network to nodes and edges with their coordinates.




(a) Sequential appearance of border points of a polygon

(b) Non-uniform spacing between points

(c) Polygon representing an area not included in any road

(d) Polygon representing a divider

**Fig. 9.** Different characteristics of polygonal data. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

extracted points for the given road network area. Hence, their combined time complexity is $O(n)$. On the other hand, divider elimination requires comparison between all pair polygons. Each comparison needs Shoelace formula for area calculation and its time complexity is $O(s^2)$, where $s$ denotes average number of border points for each polygon. Besides, the time complexity of determining whether one point is inside in a polygon or not is $O(p + s)$, where $p$ denotes the number of polygons in the given road network area. As one border point can be inside in maximum one polygon, the determination process rejects majority polygons ($O(p)$) and one

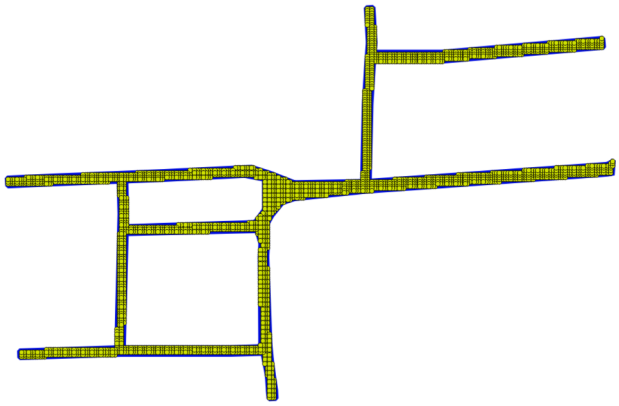


**Fig. 10.** All road points in a road network.

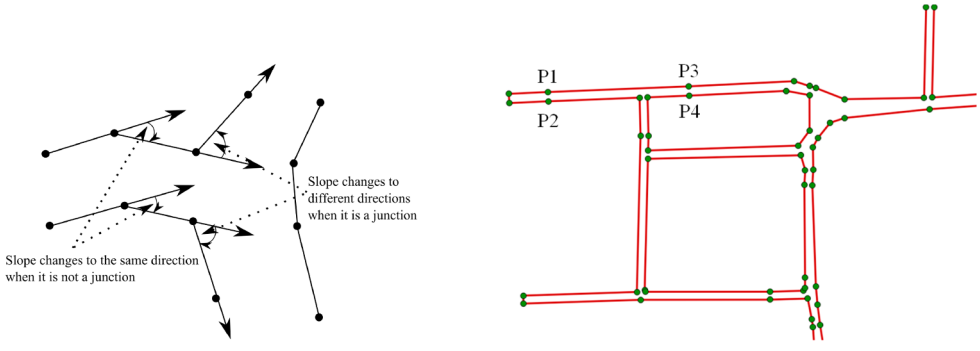
successful determination needs $O(s)$ time. Hence, the divider elimination process needs $O(ps^2(p+s))$. If the number of input points is much greater than extracted points, the overall time complexity is $O(n)$. Otherwise, divider elimination process will dominate.

### 5.2. Road point generator

This module takes valid polygons as input given by previous pre-processor module and generates all pixel points inside the roads only. We check every pixel line by line whether it is inside any road or not. We scan every pixel inside the global minimum and maximum coordinates to reduce search space and test whether it is inside one or two polygons [47]. If a pixel is inside exactly one polygon, it is situated in a road. On the other hand, if a point is in exactly two polygons, this point is situated in non-road area. Obviously, a point cannot be situated in more than two polygons. If our working resolution is greater than one, we consider its four neighbors, i.e., east, west, north, and south points, whether any of them is situated in any road or not. Since this line by line pixel checking is computationally very expensive operation, we parallelize this module by using Java threads. We use one thread per scan-line and every thread returns the valid points, i.e., points situated in roads, not in non-road area. Later, we accumulate all valid points and save in a csv file. Fig. 10 illustrates all on-road points. Thus, we generate all road points from all valid polygons in the given road network. The time complexity of this module depends on number of polygon checks against each point, which requires $O(m(p+s))$ time, where $m$ denotes the number of discrete points inside the rectangle covering all the polygons.

### 5.3. Junction and dead-end points filter

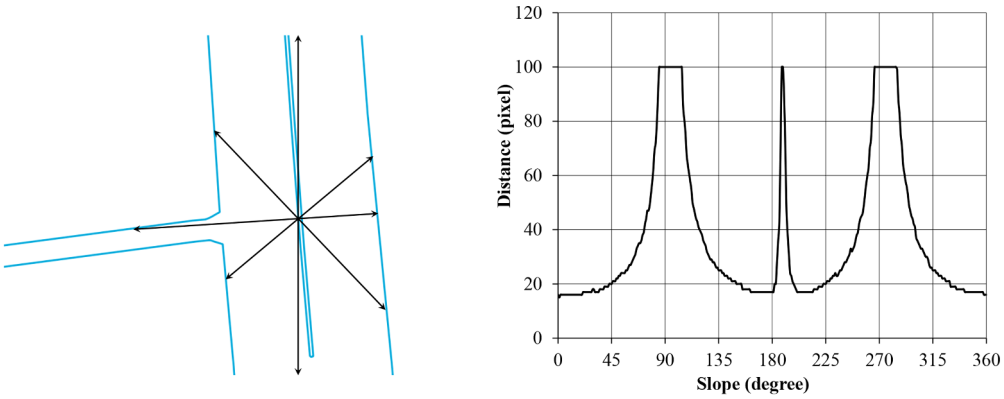
This module takes all valid points as input and returns only those valid points, which are situated inside junction of roads or dead-end of any road in the network. We can detect a junction by calculating slope changes in different points across a road. For a curve not representing a junction, the slope change will be towards the same direction for both sides of the road. However, for a curve representing a junction, the slope change will be towards different directions for two sides of the road (Fig. 11a). The border points of the polygons do not have relations with each other. For example, close points do not necessarily indicate points across a road. It can




(a) Calculation of slope changes for junction detection

(b) Border points across a road

**Fig. 11.** Determining junction with calculation of change of slope. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)




(a) Extents of a point for different directions (b) Graph of extents of a point for different directions

**Fig. 12.** Determining junction with extents of a point for different directions.

be observed from Fig. 11b. We can see that $P1$, $P2$ and $P3$, $P4$ points are situated across a road, however, $P3$ and $P4$ points are situated in different polygons. Since we cannot find the points that situated on the opposite side of a road, we cannot use this approach to detect a junction rather we follow an empirical approach.

From previous module, we get all valid points that are situated on the roads. Now, we generate rays in different directions for each valid point and check in how many directions there are roads present. Depending on the number of directions, we decide if the point is situated on a junction, or at the middle of a road, i.e., at corridor, or at the end of a road, i.e., at dead end. Let, $p$ is a valid point on a road and we generate a ray $v_\theta$ from point $p$ along a direction vector with slope $\theta$. After some time, ray $v_\theta$ goes out of the road and it needs $e_\theta$ distance to reach the border of the road. We vary the slopes from 1 to 360 °, i.e., $\theta = 1, 2, \ldots, 360$. If the ray $v_\theta$ passes along a road, the distance $e_\theta$ will be higher. We show this approach in Fig. 12a and corresponding distances are plotted in Fig. 12b. We can see there are three peaks in the curve at 95, 188, and 270 °. These peaks imply that we can go along three roads from this point. Therefore, it is a junction of three roads. We identify the peaks by comparing with an empirical threshold, which depends on the distances. If there is one peak in the curve, this point is situated in the dead-end of a road. On the other hand, if we find exactly two peaks, this point is neither a junction nor dead-end point rather it is situated in the corridor. We can calculate a road's width $w_p$ by calculating the minimum distance around all points using this equation:

$$w_p = e_\theta + e_{180+\theta} \tag{1}$$

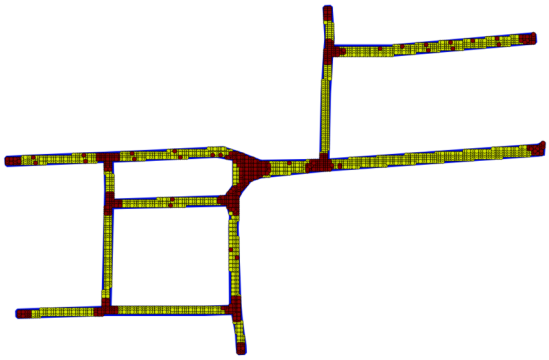
We apply this peak detection algorithm for all valid points to get the junction or dead-end points. It is also a computationally expensive task, hence, we distribute each point to a thread and later we accumulate all junction or dead-end points.

After filtering all valid points, we get on-junction and dead-end points. However, some neighbors of the filtered points get ignored by our algorithm. We manually add these false negative points with the filtered points and save all filtered points in a csv file. After running this module, we arrive at the following state illustrated in Fig. 13a. We can see that there are some false positive points filtered from this module. We clean them in the next phase. The time complexity of this module is $O(ra\log_2 a)$, where $r$ and $a$ denote number of road points and number of directions around a point to be checked respectively. Here, we have to sort the distance array of a point to find the threshold to check whether a peak exists. Hence, it requires $O(a\log_2 a)$ time.

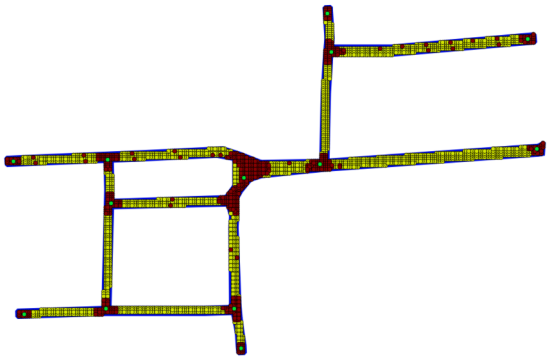
### 5.4. Node generator

We feed all points of roads, junctions, and dead-ends to this module in order to generate nodes of our desired graph. At first, we run Breadth First Search (BFS) algorithm among the junction and dead-end points to find connected components using four neighbors around each point. Thus, we get cluster of points and each cluster is a single node in our desired graph, as a node is situated in junction or dead-end in a road network. We use centroid of a cluster as the coordinate of a node. We also calculate the area of a node as a circle by using the number of points of the respective cluster. From this area, we compute radius of nodes for future usage.

From previous stage, we get some false positive points (Fig. 14a). In order to eliminate those points, we detect the adjacent points of each cluster. We again run BFS algorithm among the adjacent points using eight neighbors around each adjacent point. If they form single or double cluster, the respective filtered points are false-positive (Fig. 14b), since the adjacent points around true-positive points form exactly same number of clusters as number of roads in a junction (Fig. 14c). Note that, we bypass the dead-end points from the false detection, since adjacent points of dead-end points form a single cluster, which will make these dead-end points as false-positive. After eliminating false-positive points, we get nodes of our desired graph and save the nodes in a csv file. Fig. 13b shows the sequential output of our parser, i.e., road points, junction and dead-end points, and nodes. The time complexity of this module is $O(d)$, where $d$ denotes the number of junction and dead-end points.

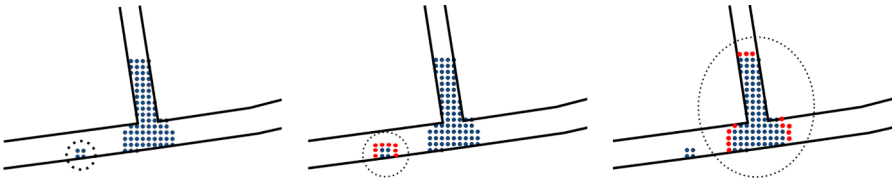


(a) Junction and dead-end points (red pixels) and road points (yellow pixels)



(b) Nodes (green pixels), junction and dead-end points (red pixels), and road points (yellow pixels)

**Fig. 13.** The intermediate output of our parser for a road network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Some false-positive points along with true-positive points

(b) Adjacent points of false-positive points form a single cluster

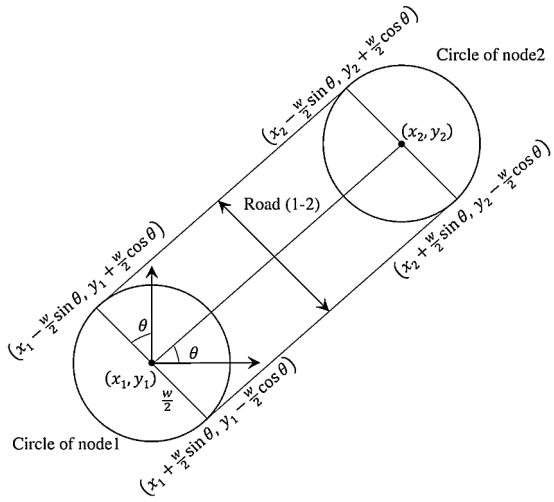
(c) Adjacent points of true-positive points form exactly same number of clusters as number of roads in a junction

**Fig. 14.** Identifying false-positive points from junction and dead-end points. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

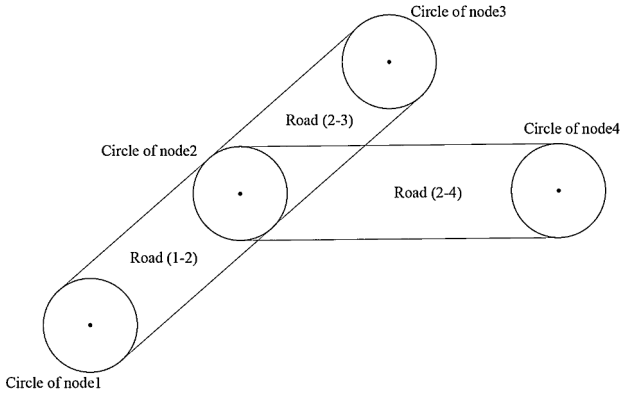
### *5.5. Edge generator*

After generating nodes of our desired graph, we use this module to generate edges from road points, junction points, dead-end points, and nodes. Edges represent the corridors or links between junctions. For simplicity, we do not consider the curvature information of edges. Hence, we use one segment per edge. In order to find the edges, we run BFS algorithm over all road points. Each road point will be adjacent to exactly two clusters of junction or dead-end points, i.e., two nodes. Since the both nodes are adjacent to each other, there is an edge in between them. Thus, we can build the adjacency list for all the nodes. We calculate width of an edge by taking the minimum of all width of points $w_p$, which we have calculated previously using Eq. 1.

In order to calculate the coordinates of the endpoints of an edge, we draw two circles around the nodes using the width of the




(a) Geometry behind determining the coordinates of the endpoints of an edge




(b) Overlapping edges while determining the coordinates

**Fig. 15.** Determining the coordinates of the endpoints of an edge.

connecting edge as radius. After that, we compute two pairs of points on the circles of adjacent nodes such that the connecting line of each pair becomes parallel to connecting line of nodes. The geometry behind this computation is showed in Fig. 15a. This approach may produce overlapping edges (Fig. 15b). In order to solve this problem, we determine the intersection of the edges [48] while meeting at a single node. We shorten the intersecting edges up-to the intersection point. Thus, we get the coordinates of edges, i.e., roads. We save them in a csv file. The time complexity of edge finding is $O(r)$ and coordinates calculation is $O(e)$, where e denotes the number of edges. Therefore, this module needs $O(r)$ time.

After passing these five stages, we get our desired graph from polygonal GIS vector data in terms of coordinates of nodes and edges. We delineate our final output in Fig. 16. We can see that our road network graph is somewhat similar to original road network. If we consider the curvature information of a road, the graph will be more realistic. The overall run time complexity of our parser is dictated by road points generation. Hence, the overall time complexity is $O(m(p + s))$.

### 5.6. Output generation

After getting the coordinates of nodes and endpoints of edges, we save the information in two plain text files respectively. We save nodes with their unique ids and adjacent nodes in one text file. Another text file contains the information regarding the edges of the graph. We save links with their unique ids, two adjacent nodes, and number of segments they have. Then, we write all segments information such as segments id, coordinates of two end points, and width of the segments. The output data can be visualized from

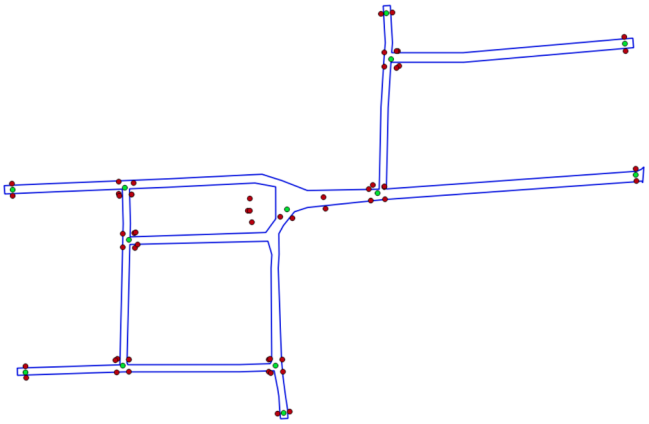


**Fig. 16.** Nodes (green pixels) and endpoints of edges (red pixels) in a road network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

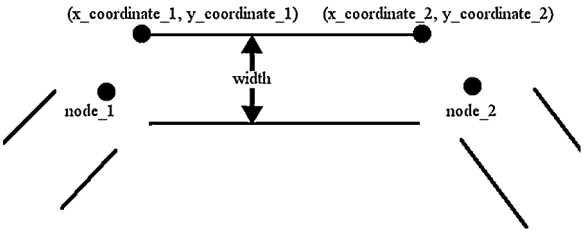



**Fig. 17.** The output data for polygonal GIS vector data of a road network.

Fig. 17. The information generated here are the basic information required to generate the graph of a road network. In the next section, we present the experimental evaluation.

## 6. Experimental evaluation of simulator

In this section, we present some preliminary results of our simulator. We evaluate our simulator in both virtual and real-world experiments. In virtual test, we analyze the effect of a particular link over the whole network. On the other hand, in real testing, we measure travel time of the simulated vehicles and compare it with real data. Note that, we use real geographic map of a particular road network in real testing, which is generated by our parser. We evaluate the performance of our simulator in a single machine and its configuration is showed in Table 1.

### *6.1. Virtual testing*

We consider a particular link of a road network and analyze its effect over the whole network. Fig. 18 shows the map of the road network, which is used for testing purpose. We analyze the effect of the red colored link (link 10) by inclusion and exclusion respectively in two simulations. The traffic statistics of both simulations such as average vehicle speed and total number of vehicles are showed in Fig. 19. We can see that the number of vehicles has increased significantly in some links such as link 9, 14, and 16 due to link 10 exclusion (Fig. 19a). Besides, the average speed of these congested links have decreased (Fig. 19b). In these experiments, we use demand flow rate of 100 vph, maximum allowable speed is 10 kph, and pedestrian mode is turned on with accident probability of 0.01%.

**Table 1**
Configuration of our Machine .

| Parameter | Value |
|---|---|
| OS | Windows 10 Pro-(64 bit) |
| CPU | Intel Core i5-3470 @ 3.20GHz |
| Cores | 4 |
| RAM | 4 GB |
| Heap | 2 GB |

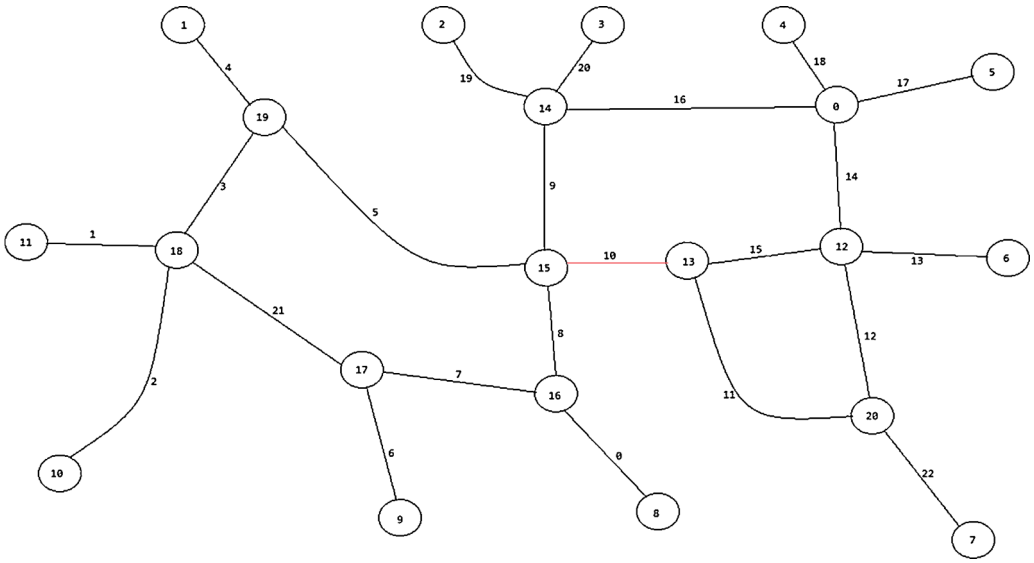
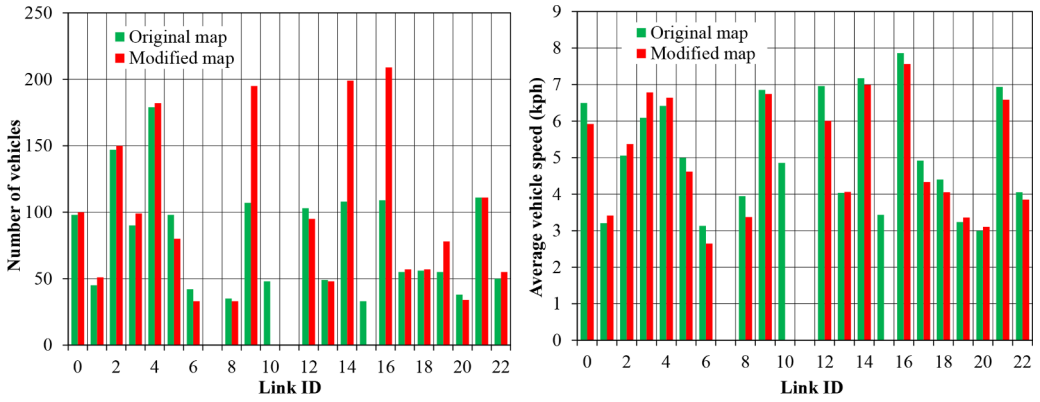



**Fig. 18.** Sample map of a road network used in analyzing link effects by removing the red colored link. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)




(a) Total number of vehicles over all links (b) Average speed of vehicles over all links

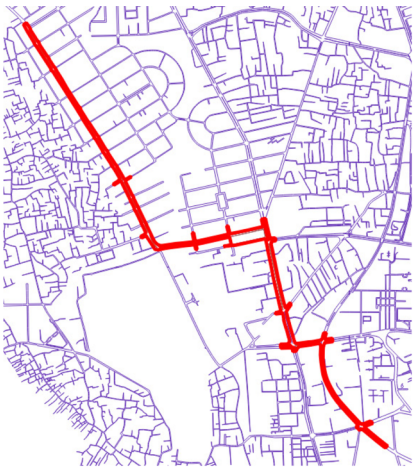
**Fig. 19.** Traffic statistics for original map and modified map. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

### *6.2. Real testing*

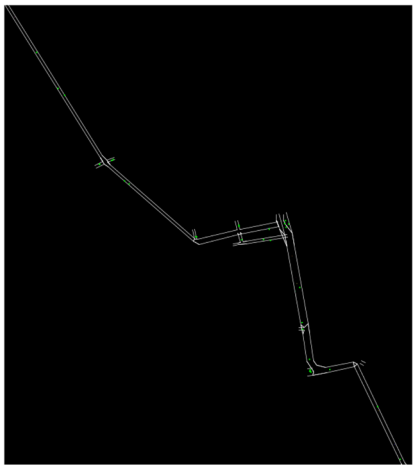
In addition to evaluation in virtual environment, we evaluate our simulator with real-world experiments. We compare the simulated travel time against real data in a real road network. We use the real road map of Dhaka city (capital of Bangladesh) and measure the travel time between Sankar and Palashi intersections in both simulation (Fig. 20b) and real-world (Fig. 20a). In the simulation, we use demand flow rate of 124 vph and maximum allowable vehicle speed of 17.3 kph in both intersections [2]. We provide these data through two files of the simulator, i.e., demand file and parameter file. We also turn pedestrian mode on. Note that, we collect real data of bi-directional traffic (37 samples over one month) including different vehicles, i.e., from Sankar to Palashi and vice-versa, in different times of days. Thus, our real data covers various conditions that can be happened in the roads. We perform 10 iterations in simulation by creating same environment of real world and log travel time of different vehicles in both directions. Then, we calculate the average values with standard deviations. The resultant average real travel time and simulated travel time is showed in Fig. 20c. We can see that our simulator closely mimics the real travel time (99% accuracy) from Sankar to Palashi and vice-versa. Next, we describe the performance of our conversion algorithm that is used to build the parser.

## 7. Experimental evaluation of parser

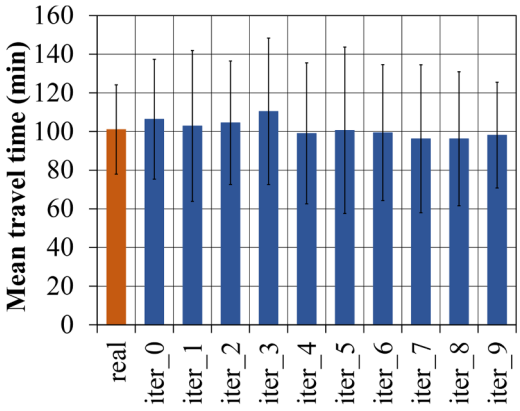
In this section, we present experimental run time analysis of our algorithm for different size of inputs. Here, we use GIS map of road network of Dhaka city, capital of Bangladesh. After that, we present the accuracy of our algorithm followed by the effect of



(a) Real GIS road map of Sankar to Palashi intersections (red colored)



(b) Simulated road map of Sankar to Palashi intersections




(c) Travel time measuring in simulator and real-world

**Fig. 20.** Comparison between simulated and real average travel times. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

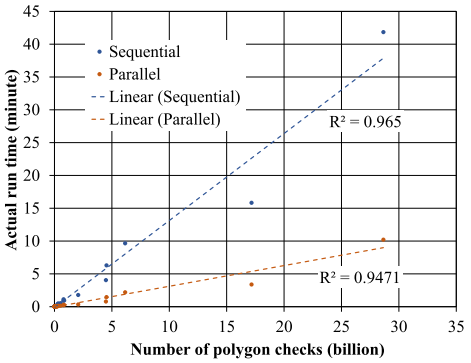
resolution changes on its performance. We evaluate the performance of our parser in a single machine and its configuration is showed in Table 1.

### 7.1. Run time analysis

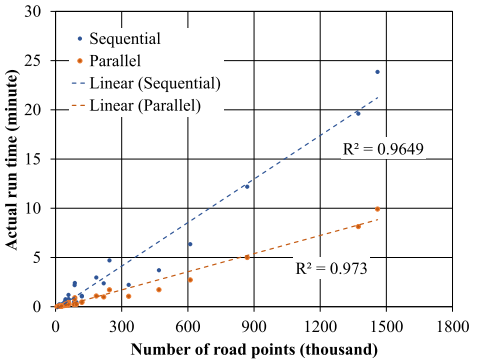
We evaluate our parser for road network of different areas of Dhaka city and measure the run time of each phase. Among five phases, road points generation is the most dominant phase followed by junction and dead-end points filtration. We apply both sequential and parallel approach for computational expensive modules. We demonstrate the actual run time of road points generation phase in both approaches in Figs. 21a against the number of polygon checks. The correlation coefficient $R^2$ confirms that the approximate worst case boundary for road points generation can be expressed by $O(m(p+s))$. Fig. 21b delineates the actual run time to find the junction and dead-end points against the total number of road points. We can see that the correlation coefficient indicates the correctness of the time complexity of junction and dead-end points filter module, i.e., $O(ra\log_2 a)$. Finally, we present over all run time against the number of polygon checks in Fig. 21c. The correlation coefficient confirms that the number of polygon checks is the most dominant factor for the run time of our algorithm and the worst case boundary can be expressed by $O(m(p+s))$.

### 7.2. Accuracy of our algorithm

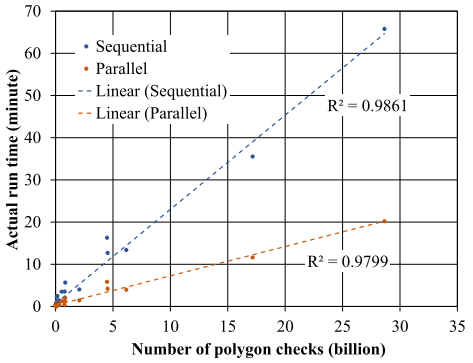
Our algorithm works perfect for equal width and linear road network (Fig. 16). However, its accuracy varies for heterogeneous width




(a) Run time of road points generator module




(b) Run time of junction and dead-end points filter module




(c) Run time of whole parsing process

**Fig. 21.** Actual run time of the most dominant two modules and whole parsing process. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

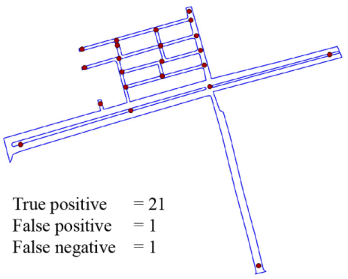
and curved road. We parse four random areas of Dhaka city and show the final output in terms of nodes in Fig. 22. We use QGIS Desktop 2.18.14 software to pre-process these areas and feed the csv output of QGIS to parser. The accuracy of our algorithm for these particular four areas are summarized in Table 2. We cannot calculate false negative rate from the data, since we do not know the number of true negative nodes. Hence, we calculate the accuracy as the percentage of true positive minus false negative as penalty over total nodes.

### *7.3. Effect of resolution changes*

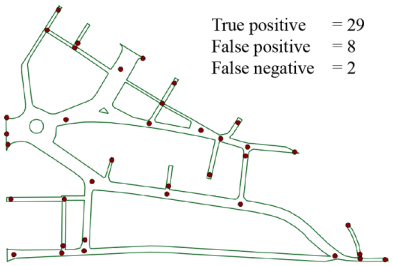
We have several parameters in our algorithm such as linear resolution, angular resolution, empirical threshold to check divider, threshold for peak detection, etc. We validate our parser by tuning these parameters to improve the accuracy. Here, we present the effect of resolution changes, i.e., linear and angular, on the performance of our parser for Agargao area. Linear resolution indicates the minimum closest point we take, i.e., we discard the points which is situated less than the minimum distance to reduce the input data set. On the other hand, angular resolution is the number of angles that we check around a point to junction and dead-end detection. The effects are showed in Table 3 with their detection statistics. We can see that both run time and accuracy decrease with the decrease of both linear and angular resolution. However, less linear resolution reduces much both time and accuracy. Since our algorithm cannot detect nodes properly in low resolution, the resolution should be compatible with the roads' width. Not only resolution but also all other parameters should be compatible with roads width. Therefore, the parameters should be tuned up empirically. Note that, we use linear as one and angular resolution as 360 degrees in all evaluation except this testing. In order to reduce run time, we use hash set data structure in our algorithm to store the points set, since hash set has constant access time on average. We also tune up the hash code to improve the run time.

## 8. Future work

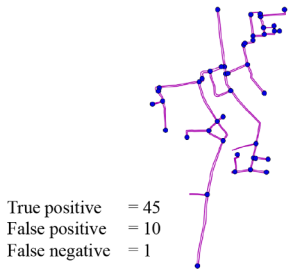
The purpose of this traffic simulator was to mimic unstructured, non-ideal road traffic scenarios present in the developing and under developed countries. Therefore, we used a simple model to realize these diverse behavior. However, there are a lot of factors



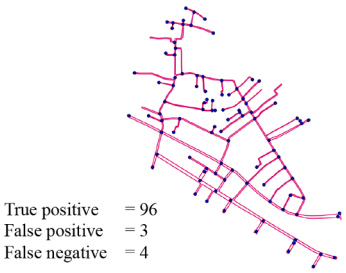

(a) Map of Agargao area




(b) Map of Dilkusha area




(c) Map of Nakhalpara area




(d) Map of Fakirapool area

**Fig. 22.** Node detection from GIS map of four random areas in Dhaka city. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

**Table 2**
Summary of node detection from GIS map of four random areas in Dhaka city .

| Location | Road length (km) | True positive (TP) | False positive (FP) | False negative (FN) | Accuracy (%) |
|---|---|---|---|---|---|
| Agargao | 2 | 21 | 1 | 1 | 91 |
| Dilkusha | 3 | 29 | 8 | 2 | 73 |
| Nakhalpara | 3 | 45 | 10 | 1 | 80 |
| Fakirapool | 6 | 96 | 3 | 4 | 93 |
| Total | 14 | 191 | 22 | 8 | 86 |

**Table 3**
Effect of resolution change while detecting node from GIS map of Agargao area .

| Linear resolution (pixel per meter) | Angular resolution (# of angles around a point) | Run time (sec) | TP | FP | FN | Accuracy (%) |
|---|---|---|---|---|---|---|
| 1 | 360 | 9 | 21 | 1 | 1 | 91 |
| 0.5 | 360 | 5.4 | 19 | 0 | 3 | 84 |
| 0.25 | 360 | 4.5 | 14 | 1 | 7 | 47 |
| 1 | 180 | 8.4 | 20 | 1 | 2 | 86 |
| 1 | 120 | 6.9 | 21 | 4 | 1 | 80 |

that we omitted for the sake of simplicity, such as, driver behavior, vehicle characteristics, modal split, etc. We envision to incorporate these factors in our future models. Moreover, owing to the lack of data, travel time between only a pair of points in the Dhaka city is simulated and verified. In future, we plan to cross check our results by collecting travel time data among different pair of points in Dhaka city to demonstrate efficacy of our simulator. We will use Google Map's historical traffic data in this purpose. Our parser also needs some advancements such as incorporating the curvature information of the roads to make the graph more realistic. We also plan to improve our algorithm such that it can detect nodes from heterogeneous road width and curved roads with better accuracy.

Nonetheless, our future plan also includes extension of our simulator for simulating diversified related phenomena. For example, we plan to incorporate different types of traffic noise models in our simulator. Besides, as transports play a vital role in the growth of urbanization, we plan to integrate urban growth models in our simulator.

## 9. Conclusion

The presence of non-ideal conditions such as the mixture of human powered and motor powered vehicles, illegal parking, violation of rules pertaining lanes, etc., are worsening the traffic condition of less developed countries in the world. A traffic simulator addressing the above-stated issues could be a potential answer towards designing an effective and sustainable transportation system for these countries. However, none of the existing studies on traffic simulator mimics all the aspects of the road traffic behavior of developing countries. Besides, in order to model the system realistic, we need to use the geographical map (GIS) of the system, which must be transformed to a list of nodes and edges with their capacity to make the map readable to the simulator. Existing literature lack a standard algorithm to convert the geographical network to graph network. Therefore, in this work, we design a simulator that meets the above-specified issues in a sustainable manner. We provide different calibration parameters that can be used to mimic the real-world scenario. We also devise a novel algorithm that transform the given geographical network into a simulator readable graph network. We use polygonal vector GIS data collected from an authoritative source, which is more sustainable than OpenStreetMap. We develop a parser based on our algorithm to parse geographical network. Real-world deployment of our parser confirms us an average accuracy of 86% while parsing road maps of four random areas of Dhaka city. On the other hand, real-world deployment of our simulator confirms us an average accuracy of 99% in terms of travel time while simulating real map provided by our parser after necessary calibration using real data of Dhaka city.

## Acknowledgment

This research was funded by ICT Division, Government of People's Republic of Bangladesh. We thankfully acknowledge to the Department of Urban and Regional Planning, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh for providing us the GIS map of road network of Dhaka city.

## References

[1] S.M. Mashrur, M.S. Hoque, Development of Calibrating Microscopic Simulation Model for Non-Lane based Heterogeneous Traffic Operation, Proceedings of the 3rd International Conference on Civil Engineering for Sustainable Development (ICCESD), Department of Civil Engineering, Khulna University of Engineering and Technology (KUET), Khulna, Bangladesh, 2016, pp. 1075–1085.
[2] S.M. Muniruzzaman, M. Hadiuzzaman, M.M. Rahman, M.M. Hasnat, N. Haque, F. Rahman, T. Hasan, Calibration and validation of microscopic simulation model for non-Lane based heterogeneous traffic stream of developing country, J. Built Environ. Technol. Eng. (JBETE) 1 (2016) 244–251.
[3] F. Huang, P. Liu, H. Yu, W. Wang, Identifying if VISSIM simulation model and SSAM provide reasonable estimates for field measured traffic conflicts at signalized intersections, Acci. Analy. Prevent. 50 (2013) 1014–1024.
[4] B. Ciuffo, C.L. Azevedo, A Multi-Step Approach for the Global Sensitivity Analysis of Complex Traffic Simulation Models: Application to the MITSIM Model, Technical Report, Transportation Research Board, Washington, DC, USA, 2013.
[5] Multi-Agent Transport Simulation MATSim, in: A. Horni, K. Nagel, K. Axhausen (Eds.), Ubiquity Press, London, UK, 2016.
[6] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, SUMO - Simulation of Urban MObility, Proceedings of the 3rd International Conference on Advances in System Simulation (SIMUL), ThinkMind, Barcelona, Spain, 2011, pp. 55–60.
[7] K.C. Payne, M. Dror, The Development of a Smart Map for Minimum "Exertion" Routing Applications, Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS), University of Hawai'i at Manoa, Hawaii, USA, 2017, pp. 1200–1205.
[8] K.C. Payne, M. Dror, Construction of a topographical road graph for bicycle tour routes, Sport. Technol. 8 (1–2) (2015) 1–11.
[9] A. Gupta, A. Yilmaz, Ubiquitous Real-time Geo-spatial Localization, Proceedings of the 8th SIGSPATIAL International Workshop on Indoor Spatial Awareness (ISA), ACM, California, USA, 2016, pp. 1–10.
[10] J. Bennett, K. A. Heater, M. L. Braun, System and methods for creating a geospatial network model in a client environment, uS Patent No. 15/398,611, Filed January 01, 2017, Issued July 06, 2017 (July 2017).
[11] A. Karduni, A. Kermanshah, S. Derrible, A protocol to convert spatial polyline data to network formats and applications to world urban road networks, Sci. Data 3 (2016).
[12] M. Haklay, P. Weber, Openstreetmap: user-Generated street maps, IEEE Pervasive Comput. 7 (4) (2008) 12–18.
[13] A. Kashian, A. Rajabifard, Y. Chen, K. Richter, OSM POI Analyzer: A Platform for assessing position of POIs in openstreetmap, Int. Arch. Photogram. Remote Sens. Spat. Inform. Sci. (ISPRS) 42 (2/W7) (2017) 497–504.
[14] S.S. Sehra, J. Singh, H.S. Rai, Assessing openstreetmap data using intrinsic quality indicators: an extension to the QGIS processing toolbox, Futu. Internet 9 (2) (2017).
[15] P. Mooney, P. Corcoran, A.C. Winstanley, Towards Quality Metrics for OpenStreetMap, Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS), ACM, California, USA, 2010, pp. 514–517.
[16] Q.M. Alam, B. Sarker, B. Biswas, K.H. Zubaer, T.R. Toha, N. Nurain, A.B.M.A.A. Islam, Towards Simulating Non-Lane Based Heterogeneous Road Traffic of Less Developed Countries, Proceedings of the 5th International Conference on Information and Communication Technology for Sustainability (ICT4S), 52 EasyChair, Toronto, Canada, 2018, pp. 37–48.
[17] J.L. Ferrer, J. Barceló, AIMSUN2: Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks, Technical Report, Departamento de Estadística e Investigación Operativa, Facultad de Informática, Universidad Politécnica de Cataluña, Barcelona, Spain, 1993.
[18] Q. Yang, H.N. Koutsopoulos, A microscopic traffic simulator for evaluation of dynamic traffic management systems, Transport. Res. Part C 4 (3) (1996) 113–129.
[19] M. Ben-Akiva, H.N. Koutsopoulos, T. Toledo, Q. Yang, C.F. Choudhury, C. Antoniou, R. Balakrishna, Traffic Simulation with MITSIMLab, in: J. Barceló (Ed.), Fundamentals of Traffic Simulation, 145 Springer, New York, USA, 2010, pp. 233–268.
[20] M. Fellendorf, VISSIM: A microscopic Simulation Tool to Evaluate Actuated Signal Control including Bus Priority, Technical Report, Institute of Transportation Engineers, Dallas, USA, 1994.
[21] D. Wilkie, J. Sewall, M.C. Lin, Transforming GIS data into functional road models for large-Scale traffic simulation, IEEE Trans. Vis. Comput. Graph. 18 (6) (2012) 890–901.
[22] M. Pooyandeh, S. Mesgari, A. Alimohammadi, R. Shad, A Comparison Between Complexity and Temporal GIS Models for Spatio-temporal Urban Applications, in: O. Gervasi, M.L. Gavrilova (Eds.), Computational Science and Its Applications – ICCSA 2007, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 308–321.
[23] T. yan SHEN, W. dong WANG, M. HOU, Z. cheng GUO, L. XUE, K. zhong YANG, Study on spatio-Temporal system dynamic models of urban growth, Syst. Eng. Theory Pract. 27 (1) (2007) 10–17, https://doi.org/10.1016/S1874-8651(08)60002-2.
[24] C. Xu, D.O. Pribadi, D. Haase, S. Pauleit, Incorporating spatial autocorrelation and settlement type segregation to improve the performance of an urban growth model, Environment and Planning B: Urban Analytics and City Science (2019), https://doi.org/10.1177/2399808318821947.

[25] H. Dadashpoor, P. Azizi, M. Moghadasi, Analyzing spatial patterns, driving forces and predicting future growth scenarios for supporting sustainable urban growth: evidence from tabriz metropolitan area, iran, Sustain. Citi. Soc. 47 (2019) 101502, https://doi.org/10.1016/j.scs.2019.101502.
[26] E. Terama, E. Clarke, M.D.A. Rounsevell, S. Fronzek, T.R. Carter, Modelling population structure in the context of urban land use change in europe, Reg Environ Change 19 (3) (2019) 667–677, https://doi.org/10.1007/s10113-017-1194-5.
[27] C. Xia, A. Zhang, H. Wang, B. Zhang, Y. Zhang, Bidirectional urban flows in rapidly urbanizing metropolitan areas and their macro and micro impacts on urban growth: a case study of the yangtze river middle reaches megalopolis, china, Land use policy 82 (2019) 158–168, https://doi.org/10.1016/j.landusepol.2018.12.007.
[28] Y. Chen, X. Li, X. Liu, Y. Zhang, M. Huang, Tele-connecting china's future urban growth to impacts on ecosystem services under the shared socioeconomic pathways, Sci. Total Environ. 652 (2019) 765–779, https://doi.org/10.1016/j.scitotenv.2018.10.283.
[29] Y. Fu, J. Li, Q. Weng, Q. Zheng, L. Li, S. Dai, B. Guo, Characterizing the spatial pattern of annual urban growth by using time series landsat imagery, Sci. Total Environ. 666 (2019) 274–284, https://doi.org/10.1016/j.scitotenv.2019.02.178.
[30] F. Zullo, G. Fazio, B. Romano, A. Marucci, L. Fiorini, Effects of urban growth spatial pattern (ugsp) on the land surface temperature (lst): a study in the po valley (italy), Sci. Total Environ. 650 (2019) 1740–1751, https://doi.org/10.1016/j.scitotenv.2018.09.331.
[31] P. Barreira-Gonzalez, F. Aguilera-Benavente, M. Gomez-Delgado, Implementation and calibration of a new irregular cellular automata-based model for local urban growth simulation: the mugica model, Environ. Plann. B 46 (2) (2019) 243–263, https://doi.org/10.1177/2399808317709280.
[32] M.C. Chandan, G. Nimish, H.A. Bharath, Analysing spatial patterns and trend of future urban expansion using sleuth, Spat. Inform. Res. 28 (1) (2020) 11–23, https://doi.org/10.1007/s41324-019-00262-4.
[33] Y. Chen, X. Li, X. Liu, H. Huang, S. Ma, Simulating urban growth boundaries using a patch-based cellular automaton with economic and ecological constraints, Int. J. Geograph. Inform. Sci. 33 (1) (2019) 55–80, https://doi.org/10.1080/13658816.2018.1514119.
[34] W. Liang, M. Yang, Urbanization, economic growth and environmental pollution: evidence from china, Sustain. Comput. 21 (2019) 1–9, https://doi.org/10.1016/j.suscom.2018.11.007.
[35] S. Perveen, M. Kamruzzaman, T. Yigitcanlar, What to assess to model the transport impacts of urban growth? a delphi approach to examine the space–time suitability of transport indicators, Int. J. Sustain. Transport. 13 (8) (2019) 597–613, https://doi.org/10.1080/15568318.2018.1491077.
[36] K.M. Paiva, M.R.A. Cardoso, P.H.T. Zannin, Exposure to road traffic noise: annoyance, perception and associated factors among brazil's adult population, Sci. Total Environ. 650 (2019) 978–986, https://doi.org/10.1016/j.scitotenv.2018.09.041.
[37] X. Lu, J. Kang, P. Zhu, J. Cai, F. Guo, Y. Zhang, Influence of urban road characteristics on traffic noise, Transport. Res. Part D 75 (2019) 136–155, https://doi.org/10.1016/j.trd.2019.08.026.
[38] R. Benocci, A. Molteni, M. Cambiaghi, F. Angelini, H.E. Roman, G. Zambon, Reliability of dynamap traffic noise prediction, Appl. Acoust. 156 (2019) 142–150, https://doi.org/10.1016/j.apacoust.2019.07.004.
[39] H. Wang, H. Gao, M. Cai, Simulation of traffic noise both indoors and outdoors based on an integrated geometric acoustics method, Build. Environ. 160 (2019) 106201, https://doi.org/10.1016/j.buildenv.2019.106201.
[40] A.A. Ahmed, B. Pradhan, Vehicular traffic noise prediction and propagation modelling using neural networks and geospatial information system, Environ. Monit. Assess. 191 (3) (2019) 190, https://doi.org/10.1007/s10661-019-7333-3.
[41] M. Cai, Z. Lan, Z. Zhang, H. Wang, Evaluation of road traffic noise exposure based on high-resolution population distribution and grid-level noise data, Build. Environ. 147 (2019) 211–220, https://doi.org/10.1016/j.buildenv.2018.08.037.
[42] N. Kuehnel, R. Moeckel, Impact of simulation-based traffic noise on rent prices, Transport. Res. Part D 78 (2020) 102191, https://doi.org/10.1016/j.trd.2019.11.020.
[43] A. Bkowski, V. Dek, L. Radziszewski, Z. Skrobacki, Validation of traffic noise models, AIP Conference Proceedings 2077 (1) (2019) 020005. URL https://doi.org/10.1063/1.5091866.
[44] Q.D. Team, QGIS Geographic Information System, Open Source Geospatial Foundation Project, 2002.
[45] ESRI, Shapefile technical description, An ESRI White Paper, Technical Report, Environmental Systems Research Institute, Inc., California, USA, 1998.
[46] B. Braden, The Surveyor's area formula, Coll. Math. J. 17 (4) (1986) 326–337.
[47] W. R. Franklin, PNPOLY - Point Inclusion in Polygon Test (April 2017). URL https://wrf.ecse.rpi.edu/Research/Short_Notes/pnpoly.html.
[48] R. Goldman, Intersection of Two Lines in Three-space, in: A.S. Glassner (Ed.), Graphics Gems, Academic Press Professional, Inc., California, USA, 1990, p. 304.