Generating Traffic Data

Thomas Brinkhoff
Institute for Applied Photogrammetry and Geoinformatics
FH Oldenburg/Ostfriesland/Wilhelmshaven (University of Applied Sciences)
Ofener Str. 16/19, D-26121 Oldenburg, Germany
http://www.fh-oow.de/institute/personen/brinkhoff/

Abstract

Experimental investigations of spatiotemporal algorithms and data structures demand for generators that produce realistic data sets. Especially Location-Based Services (LBS) require the simulation of traffic. In this case, the data sets consist of objects that move within a given infrastructure. In this paper, two different approaches—the Network-based Generator and the City Simulator—are reviewed. Both generators for traffic data have a great deal in common, but are different in certain points. In addition, a short overview on projects using these generators is given.

1 Introduction

Comprehensible performance evaluations are one of the most important requirements in the field of spatiotemporal algorithms and data structures. This demand covers the preparation and use of well-defined test data and benchmarks enabling the systematic and comprehensible evaluation and comparison of data structures and algorithms.

In experimental investigations, *synthetic data* following statistical distributions as well as *real data* from real-world applications are used as test data or as query sets. The use of synthetic data allows testing the behavior of an algorithm or of a data structure under exactly specified conditions or in extreme situations. In addition, for testing the scalability, synthetic data sets are often suitable. However, it is difficult to assess the performance of real applications by employing synthetic data. The use of real data tries to solve this problem. In this case, the selection of data is crucial. For non-experts it is often difficult to decide whether a data set reflects a “realistic” situation or not.

For testing spatiotemporal algorithms and data structures, moving objects are required. Such objects should model moving persons or driving vehicles. Especially for Location-Based Services, data sets are useful that simulate traffic. One suitable approach for getting *traffic data* consists of (1) a definition of an infrastructure, which restricts the movement of the objects, and (2) the computation of the moving objects within this infrastructure. If the infrastructure models a real-world environment, such an approach can be understood as a simulation that generates synthetic data on top of real data.

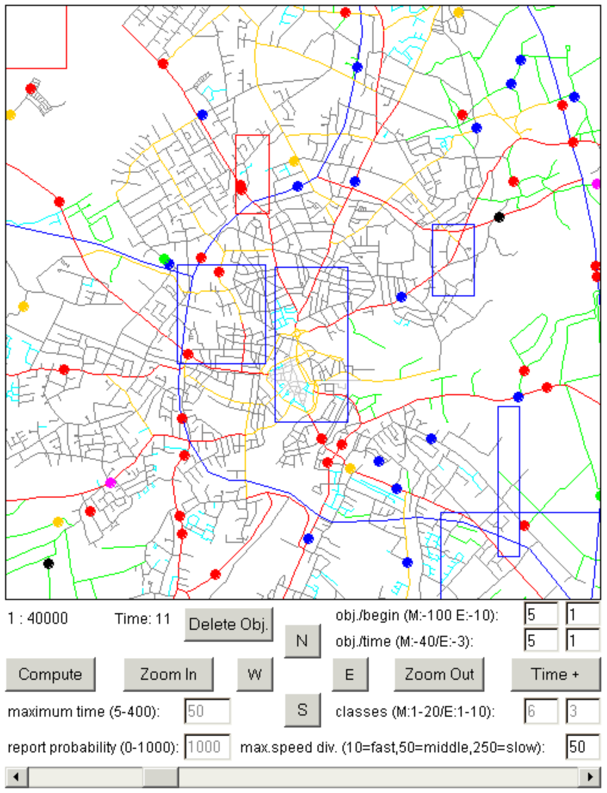
In the last few years, several generators for producing spatiotemporal data have been developed [10, 8, 9, 11]. Section 2 of this paper presents two proposals that generate traffic data according to the approach mentioned before: the Network-Based Generator by Thomas Brinkhoff and the City Simulator by J. Kaufman, J. Myllymaki, and J. Jackson from IBM. Section 3 gives a short overview on projects that make use of data generated by these generators. The paper concludes with a summary and some suggestions for future work.

2 Generators for Traffic Data

2.1 Network-based Generator by Brinkhoff

The *Network-based Generator* by Thomas Brinkhoff [1, 2] is based on the observation that objects often move according to a network. This observation holds, e.g., for road traffic as well as for railway traffic. Air traffic also follows a network of air corridors and shipping is strongly influenced by rivers, channels, and other waterways. Herds of animals often follow a (invisible) network during their migration. In consequence, (almost) no objects can be observed outside of the network. Figure 1 illustrates the graphical interface of the generator.

The generator uses a discrete time model: the whole period is divided by *n* time stamps. At each time stamp, new moving objects are generated and existing objects are moved or are deleted because they have reached their destination. Each moving object belongs to a class that specifies the behavior of the object. For example, the (maximum) speed is defined by such a class.



1 : 40000 Time: 11
obj./begin (M:-100 E:-10): 5 1
obj./time (M:-40/E:-3): 5 1
maximum time (5-400): 50
classes (M:1-20/E:1-10): 6 3
report probability (0-1000): 1000
max.speed div. (10=fast,50=middle,250=slow): 50

Figure 1: Visualization of the Network-based Generator. The points are the moving objects. The rectangles visualize the external objects. The colors indicate the classes of roads, of moving objects, and of external objects.

Each edge of the network belongs to an edge class, which defines the speed limit and the capacity of an edge. If the number of objects traversing an edge at a time stamp exceeds the specified capacity, the speed limit on this edge will be decreased.

Furthermore, so-called external objects can be generated in order to simulate the impact of weather conditions or similar influences. There are external objects, which exist over the whole period, and others, which are created in the course of the simulation and are deleted later. External objects may change their position and their (rectangular) shape over the time. If a moving object is in the catchment area of an external object, its speed is influenced according to the parameters of the class the external object belongs to.

The computations of the number of new objects per time stamp, of the start location, of the length of a new route and of the location of the destination are time-dependent. This feature allows modeling daily commuting and rush hours. In order to speed up the computation, the route of an object is computed once at the time of its creation. However, the fastest path may change over the time by the motion of other objects and of external objects. Therefore, a re-computation is triggered by events depending on the travel time (in order to simulate messages of radio traffic services) and on the deviation between the current speed and the expected speed on an edge (in order to simulate the reaction of drivers in a traffic jam).

The Network-based Generator is written in Java 1.1. Its behavior can by influenced by a parameter file as well as by extending or modifying a set of well-chosen Java classes that are provided as source code. A graphical user interface allows the setting of parameters and the visualization of the network and of the generated objects.

The network used by the generator is specified by simple text files or by spatial data stored in Oracle Spatial. The same holds for the output: the reported objects are written to a text file or into a database. The following example shows a selected part of such output file; each line consists of the type of event, the object ID, the class of the moving object, the index of the time stamp and the x,y coordinates:

```
newpoint    0   3   0   20435   19558
point   0   3   1   20455   19688
newpoint    5   0   1   13858   10979
point   0   3   2   20475   19818
point   5   0   2   13800   11627
newpoint    10  1   2   5079    18012
point   0   3   3   20496   19948
point   5   0   3   13504   12223
point   10  1   3   5334    17822
newpoint    15  0   3   13566   20167
disappearpoint  0   3   4   20493   20078
point   5   0   4   13258   12841
point   10  1   4   5981    17832
point   15  0   4   13612   19876
```

The Network-based Generator can be downloaded from following web site: *http://www.fh-oow.de/institute/iapg/personen/brinkhoff/generator.shtml*

2.2 City Simulator by IBM

The City Simulator by J. Kaufman, J. Myllymaki, and J. Jackson [4, 5] is a scalable, three-dimensional model city that enables the creation of dynamic spatial data simulating the motion of up to 1 million moving objects (persons). The data space of the city is divided into different types of places that influence the motion of the objects: roads, intersections, lawns and buildings are such places that define together a *city plan*. Each building consists of an individual number of floors for modeling the third dimension. Figure 2 illustrates a part of a city plan and the moving objects. The dark areas are buildings that are connected by a grid of roads. The small points are moving objects; their colors indicate the current floor.

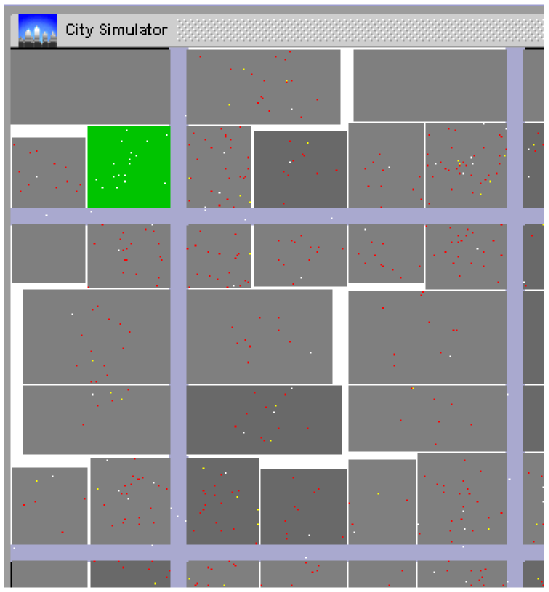


Figure 2: Visualization of a city plan.

Moving objects being on a road enter with a user-defined probability the first floor of a building. Objects on the first floor may leave the building if they are near a door. They perform random walks on a building floor or they may move up or move down from one floor to the next floor depending on user-defined probabilities if they are near to specific points (stairs). An object on a road moves with a linear combination of random walk and the drift velocity of the road; the influence of the drift velocity increases as a moving object gets closer to the center of a road.

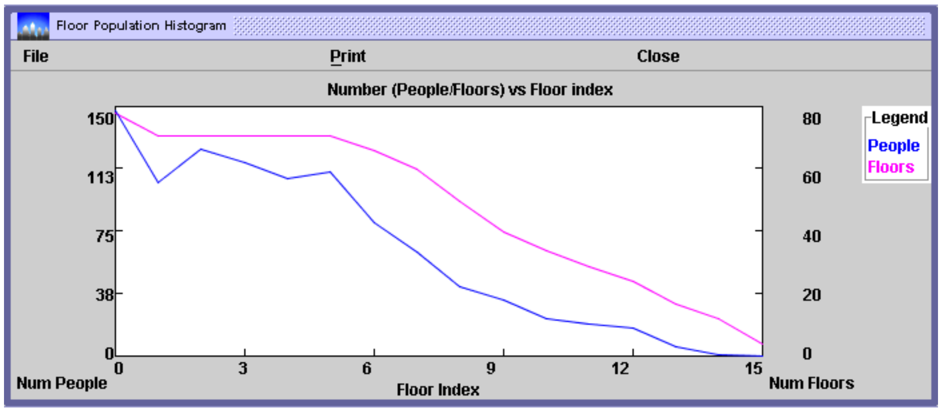


Figure 3: The floor population histogram visualizes the number of floors and the number of moving objects in respect to a floor index.

The City Simulator is written in Java 2.0 using the Xerces class library. Several parameters allow controlling the simulator. The number of objects and the number of time stamps are examples for such parameters. The parameters are defined by a parameter file and may be re-defined by the input from a graphical user interface. Other parameters are defined by the city plan. The user interface allows the visualization of the city plan, of the generated objects, and of the number of objects being on the different floors (see Figure 3).

The output is produced as a text file, which contains a unique object ID, a time stamp, and x,y,z coordinates. The next example shows some selected lines such a file:

```
# index, time,  x,y,z
1, 4840.658295, 819.6,251.4,4.6     # cycle = 320
1, 5266.601442, 817.2,247.8,0.0     # cycle = 350
1, 5879.725107, 807.2,245.6,0.0     # cycle = 390
1, 6176.540532, 804.2,249.0,4.6     # cycle = 410
1, 6327.931518, 805.6,249.6,9.3     # cycle = 420
```

The city plan is described by an XML file. It contains information about geometries and about the probabilities of movements:

```
<Road Angle="0.0" Length="470.0" Width="100.0">
    <EndPoint1 x="0" y="1050"/>
    <EndPoint2 x="470" y="1050"/>
    <MotionRules EnterProb="0.1" ExitProb="0.1" VelGradient="2.4"/>
</Road>
<GrassyField Angle="0.0" Length="518.0" Width="506.0">
    <EndPoint1 x="481" y="746"/>
    <EndPoint2 x="999" y="746"/>
    <MotionRules ExitProb="0.1"/>
</GrassyField>
<Building Angle="0.0" Length="518.0" Width="394.0">
    <EndPoint1 x="481" y="1298"/>
    <EndPoint2 x="999" y="1298"/>
    <MotionRules ExitProb="0.1" UpProb="0.03"/>
    <Floor Altitude="18" Angle="0" FloorNum="1" Length="518" Width="394">
        <EndPoint1 x="481" y="1298"/>
        <EndPoint2 x="999" y="1298"/>
        <MotionRules DownProb="0.03" UpProb="0.03"/>
    </Floor>
    <Floor Altitude="36" Angle="0" FloorNum="2" Length="518" Width="394">
        <EndPoint1 x="481" y="1298"/>
        <EndPoint2 x="999" y="1298"/>
        <MotionRules DownProb="0.03" UpProb="0.03"/>
    </Floor>
</Building>
```

The Java class representing the city plan can be replaced by a user-specific class fulfilling a specified Java interface. The download address of the City Simulator is: *https://secure.alphaworks.ibm.com/aw.nsf/techs/citysimulator*.

2.3 Discussion

Comparing both generators for traffic data, we can observe several similarities:

- A discrete time model is used.
- The generators are implemented in Java. Compiled class files are provided plus some source code.
- The infrastructure is specified by user-defined files.
- The motion of the objects is computed by a simulation.
- The output is produced as simple text files.

There are some significant differences: While the Network-based Generator is limited to two-dimensional data sets, the City Simulator supports three-dimensional city plans and computes 3D points. Another difference concerns the map: The first approach limits the movements of the objects by edges of width 0 whereas the City Simulator defines areas. As a result, spatial clusters in the shape of lines can by expected by the Network-based generator, and polygonal clusters using the City Simulator. In the case of the City Simulator, the movement of the objects is influenced by the rules of the place they are in. In contrast, the Network-based Generator also considers the possible influence of other moving objects.

3 Applications

This section gives a short overview on projects that are using the presented generators for traffic data.

LOCUS LOCUS [5] is a testbed for dynamic spatial indexing. It supports the DynaMark benchmark specification [6]. The City Simulator is a component of the architecture of LOCUS for generating location trace files. Each record in a location trace file is used for updating the spatiotemporal index. After *n* updates, *m* queries like proximity queries, k-nearest neighbor queries, and sorted-distance queries are executed in respect to the location of a user. Figure 4 depicts the architecture of LOCUS.

MOX Applications tracking and presenting mobile objects require to be kept informed about new, relocated, or removed objects fulfilling a given query condition. Consequently, the spatiotemporal database system must trigger its clients by transmitting the necessary information about such update operations. The query, which causes this process, is called *continuous query*. MOX is an architecture for querying XML-represented moving objects [3]. It especially supports continuous queries. Figure 5 illustrates the architecture of MOX. For testing the system and for investigating continuous queries, the Network-based Generator has been integrated into MOX. The objects produced by the generator are inserted into the database, which triggers the affected continuous queries.

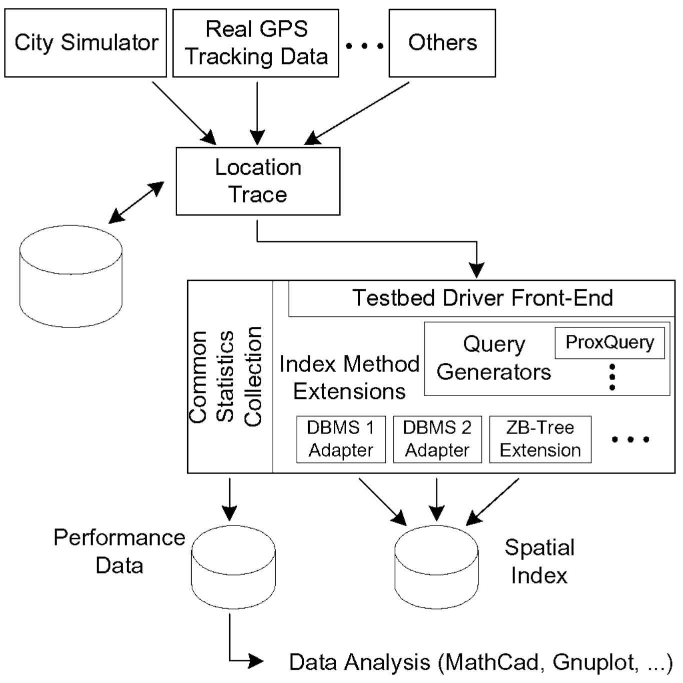


Figure 4: Architecture of LOCUS [5]. The City Simulator is one possible component for generating location trace files.

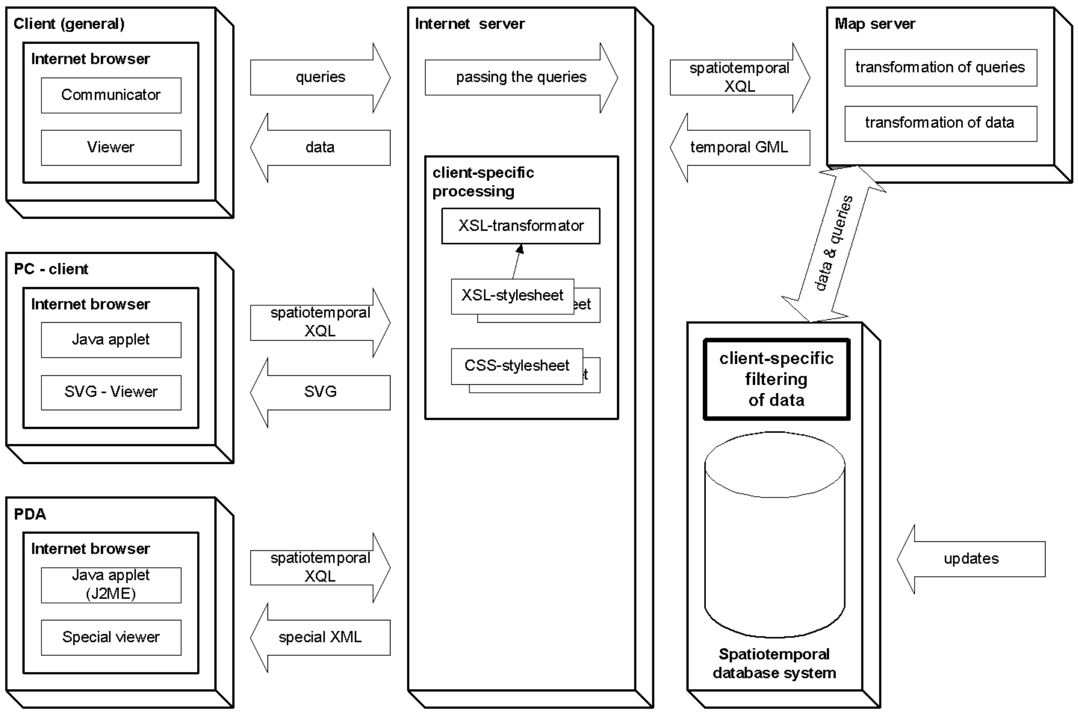


Figure 5: Architecture of MOX [3]. The Network-based Generator is used for performing the updates in the database system.

Performance Tests The use of the presented generators for experimental evaluations of spatiotemporal algorithms and data structures is just starting. A first example is the investigation of approximations for trajectory segments by Zhu, Su, and Ibarra [12].

4 Conclusions

Two approaches for generating traffic data – the Network-based Generator and the City Simulator – have been presented. Both generators allow the simulation of the motion of a huge number of moving objects. They have been integrated into more complex architectures for testing spatiotemporal queries. However, the number of investigations using these generators is still quite low. In the next years, the presented generators must prove their usefulness.

For improving the usability of generators for spatiotemporal data, a (more or less) standardized XML output format (e.g. defined by using GML 3 [7]) would be helpful. Furthermore, an open exchange forum for providing maps (i.e. network files and city plans), parameter settings, location trace files, and so on could help to boost the use of the generators.

References

[1] T. Brinkhoff. Generating Network-Based Moving Objects. In: *Proc. 12th International Conference on Scientific and Statistical Database Management*, Berlin, Germany, 2000, pp. 253–255.

[2] T. Brinkhoff. A Framework for Generating Network-Based Moving Objects. *GeoInformatica*, 6(2):155–182, June 2002.

[3] T. Brinkhoff and J. Weitkämper. Continuous Queries within an Architecture for Querying XML-Represented Moving Objects. In: *Proc. 7th International Symposium on Spatial and Temporal Databases*, Redondo Beach, CA, pp. 136–154, 2001.

[4] J. Kaufman, J. Myllymaki, and J. Jackson. City Simulator. *alphaWorks Emerging Technologies*, November 2001, *https://secure.alphaworks.ibm.com/aw.nsf/techs/citysimulator*.

[5] J. Myllymaki and J. Kaufman. LOCUS: A Testbed for Dynamic Spatial Indexing. *Bulletin of the Technical Committee on Data Engineering*, 25(2):48–55, June 2002.

[6] J. Myllymaki and J. Kaufman. DynaMark: A Benchmark for Dynamic Spatial Indexing. In: *Proc. 4th International Conference on Mobile Data Management*, Melbourne, Australia, pp. 92–105, 2003.

[7] Open GIS Consortium Inc. *OpenGIS Geography Markup Language (GML) Implementation Specification, Version 3.0*, January 2003, *http://www.opengis.org/techno/implementation.htm*.

[8] D. Pfoser and Y. Theodoridis. Generating Semantics-Based Trajectories of Moving Objects. In: *Proc. International Workshop on Emerging Technologies for Geo-Based Applications*, Ascona, Switzerland, pp. 59–76, 2000.

[9] J.-M. Saglio and J. Moreira. Oporto: A Realistic Scenario Generator for Moving Objects. *GeoInformatica* 5(1):71–93, March 2001.

[10] Y. Theodoridis, J.R.O. Silva, and M.A. Nascimento. On the Generation of Spatiotemporal Datasets. In: *Proc. 6th International Symposium on Large Spatial Databases*, Hong Kong, China, pp. 147–164, 1999.

[11] T. Tzouramanis, M. Vassilakopoulos, and Y. Manolopoulos. On the Generation of Time-Evoling Regional Data. *GeoInformatica*, 6(3):207–231, September 2002.

[12] H. Zhu, J. Su, and O.H. Ibarra. Trajectory Queries and Octagons in Moving Object Databases. In: *Proc. ACM International Conference on Information and Knowledge Management*, McLean, Virginia, pp. 413–421, 2002.