7 Tikhomirov V.M. *Results of science and technology: Modern problems in mathematics: Fundamental direction*. VINITI, Moscow, 1987, 14, p. 103–270.
8 DeVore R.A., Konyagin S.V., Temlyakov V.N. *Contruc. Approx.*, 1998, 14, p. 1–26.
9 Schmeisser H.-J., Sickel W. *Journal of Approximation Theory*, 2004, 128, 2, p. 115–150.
10 Sickel W., Ullrich T. *Journal of Approximation Theory*, 2009, 161, 2, p. 748–786.
11 Romanyuk A.S. *Ukrain. Math. Journal*, 1991, 43, 1, p. 1297–1306.
12 Romanyuk A.S. *Ukrain. Math. Journal*, 1997, 49, 9, p. 1409–1422.
13 Sun Yongsheng, Wang Heping. *Trudy Math. Inst. Steklov*, 1997, 219, p. 356–377.
14 Stasyuk S.A. *Matem. zametki*, 2010, 87, 1, p. 108–121.
15 Akishyev G. *East Journal of Approx*, 2008, 14, 2, p. 193–214.
16 Akishyev G. *Mathem. Sb.*, 2006, 197, 8, p. 17–40.
17 Bekmaganbetov K.A. *Ufimskiy Matem. Journal*, 2009, 1, 2, p. 9–16.
18 Bary N.K., Stechkin S.B. *Trudy Moskov Mat. Obshch.*, 1956, 5, p. 483–522.
19 Akishyev G. *Mathematical Journal*, Almaty, 2008, 8, 4, p. 10–19.
20 Akishyev G. *Uchenie Zapiskiy Kazan Univer.*, 2006, 148, 2, p. 5–17.

UDC 004.9

T.Bakibayev, K.Abeshyev

al-Farabi Kazakh National University, Almaty (E-mail: timurbakibayev@gmail.com)

Adaptation of GIS Open Source Maps to Lanes Graph

This article describes the algorithm for automatic adaptation of a graph of streets taken from open source maps to a graph of traffic lanes. During the work, we encountered several problems, some of which have been left open. They key idea is to generate new edges «to the right» of the edges given in a street map and to connect them together. All solutions are described with a text followed by some programming code.

Key words: GIS, maps, streets, lanes, graphs, traffic simulation.

Introduction

Given a set of vertices and edges that represent streets with their properties like «one-way», «number of lanes» etc., our goal is to produce a proper graph of traffic lanes with connected edges so that it would be possible to find the exact ways of traffic movement.

What are the general problems with the graph? First, all neighbor edges must be connected in a sense that they really need to have a common vertex. Second, the graph must obey some basic traffic rules like it is not allowed to turn right from the second lane or to turn left from the right lane.

The basic approach is as follows:

1. If the street is not one-way, we copy it and swap the direction. Both streets become one-way.
2. Having only one-way streets, we build vertices and edges to the right of this street, as many as the stated number of lanes. By default, we build three lanes.
3. Correcting the graph.

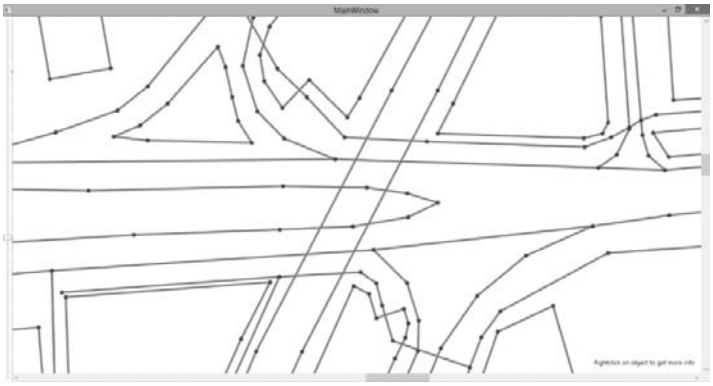
The last step is the most difficult and hence the most interesting part.

Data from Openstreetmap.org. The data available on openstreetmap.org is divided into two basic parts:

1. Node. This object represents a vertex with geographic coordinates.
2. Way — a sequence of nodes and some properties. The ordered sequence of nodes can represent a road (street), building or a park depending on its properties. Moreover, each type of these objects has its own unique properties. For instance, a street has a property «One-way»; a building has a property «Levels».

Reading a way from the database we build a set of edges in our graph. Of course, all the original data is stored in each edge so we can find the corresponding original way in the database in just one step. We skip the simple function that reads an XML file downloaded from openstreetmap.org and creates an array of

nodes and edges. As the result we get two lists named «mapWay» and «mapNode». So by the moment we only have loaded the map without lanes. This would be insufficient for a GPS navigator or traffic simulation software. Picture 1 demonstrated the data we received.



Picture 1. Abai street, Almaty

Here you can see not only the streets but also a small park and buildings. Some of these streets are actually not visible because they are doubled with the opposite direction.

Now our goal is to build a graph of lanes for each street on its right side. In order to do this we check how many lanes does it have and then we take a gap of lane width multiplied by lane number to the right from each point of the street. Actually, moving to the right means we have to check the angle of the street and then subtract 90 degrees (90 degrees clockwise). Here come a couple of very simple geometrical functions.

For finding the angle we use arctan function [1]:

$$Angle = \arctan\left(\frac{endPoint.Y - startPoint.Y}{endPoint.X - startPoint.X}\right) * \pi / 180.$$

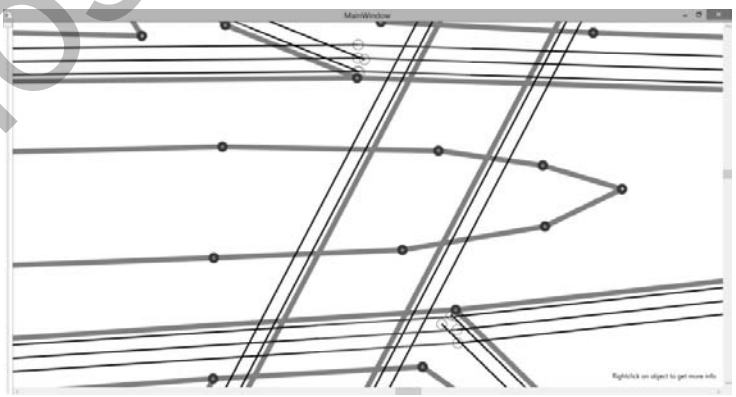
But the arctan function sometimes leads us to unexpected results so we have to do some manipulations that you can see below the formula in the above given programming code.

Now as we know how to find points to the right from endpoints of our streets, we add the nodes and edges to the graph that we want to see.

Now we introduce the problems we faced and the solutions we found.

Processing the Road Map

From the previous chapter we get the following picture (Picture 2).



Picture 2. Lanes without processing

The thin lines are the edges we generated, and the thin circles are the corresponding vertices. One third of each line is marked with a green color and shows the direction of the edge. As you can see there are some visible problems in our new built graph. First, the lanes on the joined streets are crossed and if you want to work with the graph it would be impossible to have a turn to the joining streets (since the vertices do not

match). Second, on this graph it seems to be possible to turn to the right from the second lane. Third, we have to get rid of the unnecessary tails.

Since finding the cross points of two lines is very simple (see [2]) we only mention that the function is named «is Crossed» and it returns true if the given two lines have a common point and this common point is saved to the «cross Point» parameter.

Having this function the idea to solve the problems is to find the cross points, cut the ending edges in these points, and to divide the cutting lane into two lanes so that this graph would be proper.

Now, to go deeper into details we need to introduce a function called «moveToAngle» that works as follows. Having as input a point to start with, an angle and a distance, it finds a new point with the same *z*-coordinate:

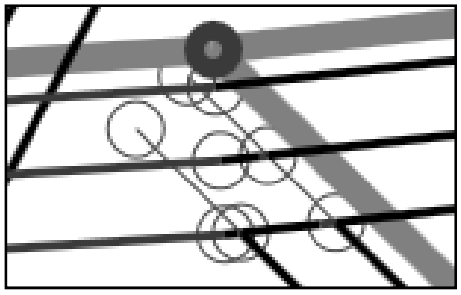
$$x = inPoint.X + \cos(angleInRadian) * Distance;$$
$$y = inPoint.X + \sin(angleInRadian) * Distance;$$
$$z = inPoint.\mathrm{Z}.$$

The algorithm of the solution is as follows.

1. For each lane[i] do
   a. If lane[i] is an ending lane, For each lane[j] do
      if lane[j].node1 = lane[i].node2 or lane[j].node2 = lane[i].node2
      if is Crossed(lane[i],lane[j])
        lane[i].node2 = cross Point;
        lane[j].node2 = cross Point;
        newLane.node1 = lane[j].node2; //Break lane[j] into 2 parts
        newLane.node2 = lane[j].oldNode2;
   b. If lane[i] is a starting lane, For each lane[j] do
      if lane[j].node1 = lane[i].node1 or lane[j].node2 = lane[i].node1
      if is Crossed(lane[i],lane[j])
        lane[i].node1 = cross Point;
        lane[j].node2 = cross Point;
        newLane.node1 = lane[j].node2; //Break lane[j] into 2 parts
        newLane.node2 = lane[j].oldNode2;

This ensures that nodes at cross points are created and the graph becomes proper in a sense that all intersections are marked as nodes.

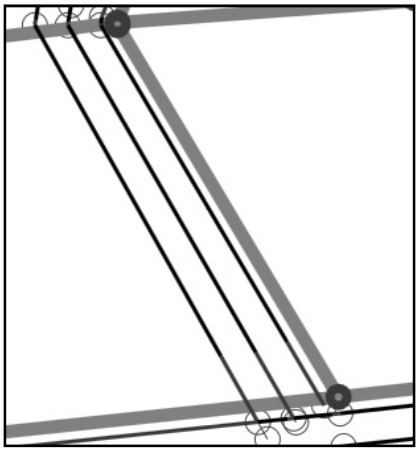
Picture 3 shows the result of this algorithm.



Picture 3. The algorithm for getting rid of excessive tails

Here the red lines are the lines that do not exist anymore. In the end we can turn right only from the right most lanes. The same works with the left turn (obvious from the programming code above). Note that the nodes (the green circles on the picture) still exist. But the fact is that no edges (ways, streets) are connected to these nodes, so they do not really hurt and can easily be eliminated. We still keep them in order to show the red lines in this project.

Besides of these three problems there is another problem that is visible from the Picture 4.



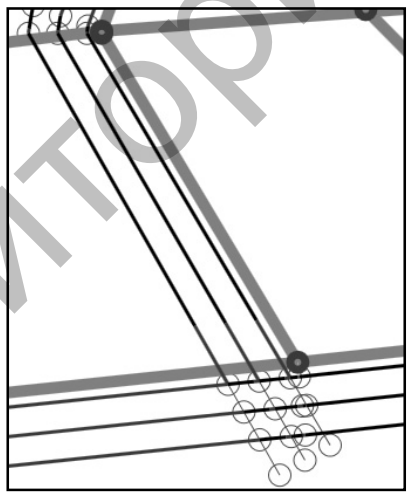
Picture 4. Insufficient edge length

Here the problem is that the left most lane does not touch another left most lane. This occurs because the angle between streets is almost 90 degree and the «receiving» street is one way. So the lane length is insufficient to touch the other street and for this we introduce another solution.

Since this can occur only in the beginning or in the ending of the complete street (not in the middle), we are sure that making these ending lanes longer would not hurt at all. And for this we have written the function that makes the lines longer. It takes two points and a distance as an input, calculates the angle using the abovementioned method and uses the «moveToAngle» function to make the line longer:

$$new\ point = moveToAngle(point2, calculateAngle(point1, point2), distance);$$

In addition, we find the lanes that are the beginning or the ending of streets and use this function. The result is as in Picture 5.



Picture 5. Solved graph

Now the lanes are much longer and we are sure they will be cut if there is any street that is supposed to receive the traffic.

Conclusion

As the result of our work, we got the nice algorithms for building a new graph based on only a street map that allows analyzing the map on lanes level. We introduce the mathematical problems we faced and show how to solve them, not only the ideas but also some programming code samples. This may be useful if you design a GPS navigator that would assist the driver with lane switching and that would find better (faster) ways to reach to the destination point. This work is a part of a project for building a traffic simulation software for mathematical analysis of traffic flow in Kazakhstan cities.

References

1 Weisstein Eric W. *Angle. From MathWorld — A Wolfram Web Resource*. [ER]. Access mode: http://mathworld.wolfram.com/Angle.html

2 Weisstein Eric W. *Line-Line Intersection.* From MathWorld — A Wolfram Web Resource. [ER]. Access mode: http://mathworld.wolfram.com/Line-LineIntersection.html

Т.Бәкібаев, Қ.Әбешев

Ашық ГИС карталарының қозғалыс сызықтарының графына бейімделуі

Мақалада «Open street maps» сайтынан алынған көше графтарының қозғалыс сызықтарының графтарына автоматты бейімделуінің алгоритмі ұсынылған. Жұмыс барысында бірнеше қиындықтар кездесті және олардың бірқатары шешімсіз қалды. Мақаланың негізгі мақсаты көше графтары қабырғаларының «оң жағынан» жаңа қабырғалар құру және оларды біріктіру болып табылады. Барлық шешім жолдары мәтінде сипатталған, сондай-ақ бағдарлама кодтары берілген.

Т.Бакибаев, К.Абешев

Адаптация открытых ГИС карт к графу полос движения

В статье описан алгоритм автоматической адаптации графа улиц, взятого с «Open street maps», к графу полос движения. Во время работы мы столкнулись с несколькими проблемами, некоторые из которых остались открытыми. Основная идея заключается в создании новых рёбер «справа» от рёбер графа улиц и соединении их. Все решения описаны текстом, затем следует программный код.

UDC 510.67

B.Poizat

Claude Bernard Lion1 University, Camille Jordan Institute, France (E-mail: poizat@math.univ-lyon1.fr)

Fundaments of Positive Logic

The main purpose of this paper is to give the basic concepts of positive logic. The content of this work is the essence of the course of lectures given by me in the Academican Ye.A.Buketov Karaganda State University in the months of April and May 2013.

Key words: Homomorphisms, positive formulae, inductive limits, inductive classes, universal sentences, compactness, companion theories, model-complete theories.

Outlet