# Computing the least complex path for vehicle drivers based on classified intersections

Robert S. Sladewski
University of Augsburg,
Institute of Geography
Alter Postweg 118
86159 Augsburg, Germany
robert.sladewski@gmx.de

Andreas Keler
University of Augsburg,
Applied Geoinformatics,
Institute of Geography
Alter Postweg 118
86159 Augsburg, Germany
andreas.keler@geo.uni-augsburg.de

Andreas Divanis
University of Augsburg,
Institute of Geography
Alter Postweg 118
86159 Augsburg, Germany
andreasdivanis@web.de

**Abstract**

Recent services for car navigation include selections for computing the shortest, fastest or most economic route between origin and destination. Our idea is to compute the least complex route, which might be challenging, since there is no consistent definition of complexity. We focus on the complexity of road intersections, which is experienced by vehicle drivers with the turning motion itself in selected crossroads. Therefore, we define weights for different turning possibilities. In a case study in Le Havre, we compute the least complex path on the underlying road network. First results show differing routes comparing to shortest path based on Dijkstra's Algorithm.

*Keywords*: Least Complex Path, Shortest Path, Vehicle Routing, Car Navigation Systems, Location Based Services (LBSs).

## 1 Computing Routes

Most car navigation systems compute the shortest path, the fastest path or the most economic one and calculate with the attributes of the underlying graph. Some of the attributes allow path calculations, which avoid toll roads or highways based on the personal preferences of the driver. Additionally, "real-time" information about traffic jams, for example, can be taken in account in the route calculation to provide the user with some alternatives. The automatic positioning function (e.g. GPS) allows the navigation system to locate the user constantly and to compare the driving behaviour with the proposed route.

However, there are a few more navigation strategies, which comply with a purpose. In case of human navigation, explaining the best route to a tourist requires the simplest route with a minimal set of instructions. This Simplest Path concept proposed by Duckham and Kulik (2003) reduces the amount of information by negotiation of intersections, where users go straight. For navigating through an unknown environment towards a visible target (landmark), the Least Angle Strategy of Hochmair (2000) may be the best choice. At an intersection, the agent takes the road with the least deviation from the direct target direction. In case of bicycle navigation, the scenery is a desirable route criteria but not essential and hard to find. Hochmair and Navratil (2008) show a method of finding scenic routes using standard shortest path computation and the reducing of costs of edges by traversing buffers around attractive places. Since the user is interested in a compromise of a nice scenery and the common route selection criteria shortest, fastest and simplest path, Hochmair and Fu (2013), i.e., show a more complex computational method with the additional criteria "scenic route" and "route with least bicycle-car interactions".

Other routing strategies include the previous generation of a cost surface. Subsequently, these continuous fields are constrained by road networks (Karrais et al., 2014). There are numerous possibilities of connecting surface information with road network segments. Examples for the latter routing strategies are computations of the safest paths for motorised tourists based on recent crime reports (Keler and Mazimpaka, 2016) or of paths with the best available air quality (Karrais et al., 2014).

Another issue is the complexity of the underlying transportation infrastructure. Krisp and Keler (2015) provide routing options for driving beginners in Munich that avoid complicated crossings by defining them using the number of nodes they contain. Crossings with high complexity are represented as obstacle polygons that are avoided by driving beginners.

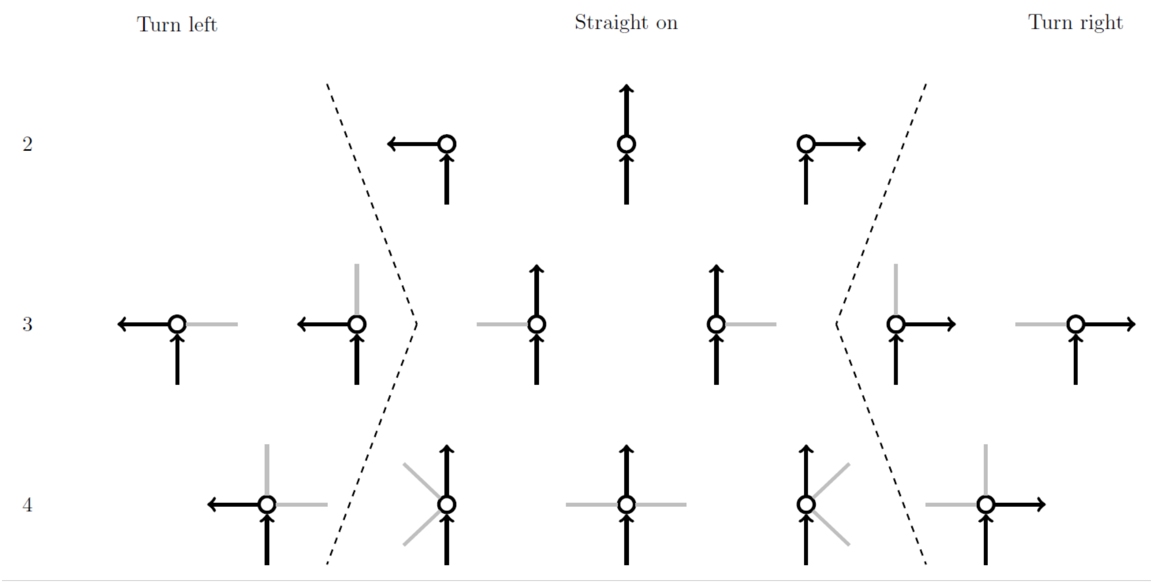
The implementation of the mentioned strategies mostly relies on the Algorithm of Dijkstra (1959) as a subroutine, which iterates through a network calling the values of edges, nodes or pairs of edges.

The aim of this work is to define driving situations at intersections and respectively to define the terms "turning" and "straight ahead". We want to classify these situations according to the composition of the intersection and to give them values to make a computation of routes possible. This classification leads to an algorithm we implemented in java.

## 2 Definitions of *turn* and *straight*

In this section, we define the more complex process of turning and the less complex situation of going straight ahead. The definitions base not on absolute values like the angle of turning.

Figure 1: Intersections with 1 node and its degree from 2 to 4.



We define the situations depending on the composition of the current intersection respectively the conditions of the situation. In a first step, we make rules of turning and driving straight ahead. We summarize the possible situations in a table and give them values, which classify the situations at its best.

## 2.1 *Turn* situation

There are 3 conditions that describe a turn situation, which are presented in Table 1.

Table 1: The conditions of turn.

| | |
|---|---|
| **Cond. 1** | The road to follow is on the left/right side of the road on which the driver is currently driving |
| **Cond. 2** | There has to be at least a second path option, at an angle greater than that of the first junction, in the direction of the traffic |
| **Cond. 3** | Only the junction with the smallest angle is the turn junction |

## 2.2 *Straight* situation

There are 3 conditions that describe a straight on situation, which are presented in Table 2.

Table 2: The conditions of straight.

| | |
|---|---|
| **Cond. 1** | Going straight on is using the junction with the greatest angle to the road the driver is currently driving |
| **Cond. 2** | Neither turn left or turn right applies to straight ahead |
| **Cond. 3** | At the same obtuse angles that satisfy the second condition, the right arm is made |

## 2.3 Intersection diagram

The previous definitions lead to the diagram in Figure 1 that summarizes the possible situations constructed by one single node and its degree from 2 to 4.

The left and the right situation in the first row cannot be called turning because they infringe condition 2 of Table 1.

## 2.4 Intersection classification

Finally, the presented intersections in Figure 1 get a value as it is illustrated in Table 3.

Table 3: Classification of the intersections.

| Degree | Value |
|---|---|
| Degree 2 | 0 |
| Degree 3 +straight | 1 |
| Degree 3 + turning | 2 |
| Degree 4 + straight | 3 |
| Degree 4 + turning | 4 |

## 3 Case study

We implemented an algorithm, which computes routes based on this intersection classification. For this implementation, the GraphStream (2016) java library was used and applied on a dataset of the French city Le Havre. The dataset consists of a connected graph in DGS format, which is routable.

The algorithm of Dijkstra (1959) is the subroutine of this implementation, which is able to compute the shortest path in the light of the lengths of edges, nodes, or pairs of edges. Considering the complexity of intersections for this work, it iterates through pairs of edges and summarizes the values of their classification.

We choose a small study area in Le Havre and first compute the shortest path (Figure 2) and then the least complex path (Figure 3) between two selected points. We want to compare the generated routes by total length and by accumulated complexity values.

Figure 2: Shortest path with the aid of algorithm of Dijkstra.

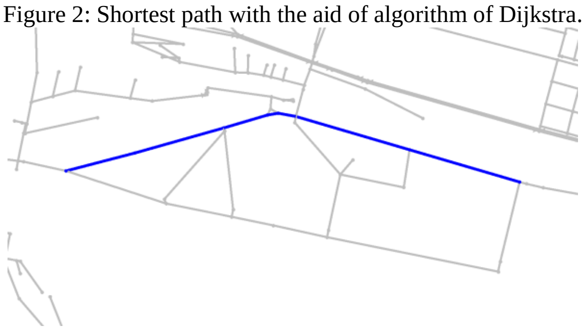
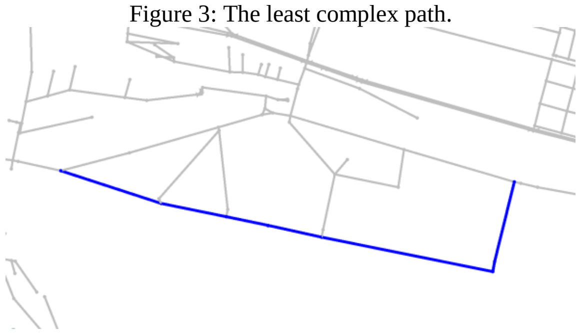


Figure 3: The least complex path.



The shortest path in Figure 2 implies a length of about 3484 m and the least complex path in Figure 3 a length of about 4040 m. This means the least complex path in this case is nearly 16 % longer than the shortest path.

By counting the values of the intersections, it becomes apparent that the shortest path in Figure 2 would have a complexity value sum of 7 and the least complex path in Figure 3 a complexity value sum of 3 (compare values in Table 3).

The source node is set to 0. The least complex path algorithm works in both directions and, respectively, gets the same result.

## 4 First findings and outlook

Our first results of the proposed routing algorithm show reasonable least complex paths in many parts of Le Havre. This highlights the usefulness of the intersection scheme in Figure 1.

The further steps in our approach would be the testing of the algorithm with randomly selected points within the road network. The Euclidean point distances between start and destination points should then vary between short and very long distances that go across the whole investigation area. The resulting generated random routes calculated by the algorithm of Dijkstra and our algorithm might then reveal eventual problems or challenges of the proposed technique.

Computing the least complex path in a road network might be realized in different ways and can be adapted to selected investigation areas. Therefore, we need test results of other road networks from other cities. This facilitates also reasoning on our introduced intersection classification in Table 3 that might be then less suitable for Asian megacities than for the road network of Le Havre.

Additionally, we want to include OpenStreetMap (OSM) road network information for testing our routing algorithm. The road network of OSM has one of the best qualities of accessible road network geodata (Stanica et al., 2013). Its quality was already evaluated for its suitability for vehicle routing by Graser et al. (2014). Additionally, OSM and in general VGI can help improving the quality of already established routing services (Bakillah et al., 2014), despite the fact that the accuracy and validity of OSM data is difficult to detect (Helbich et al., 2010). Nevertheless, we need to test the connectivity of selected OSM road networks for guaranteeing reliable applications. Therefore, we need to convert OSM road networks into routable connected graphs.

Another motivation of the latter is to compare the proposed least complex path method with the technique by Krisp and Keler (2015) for computing the easiest to drive route. The latter technique is dependent on the node density of extracted OSM road segments. It might be interesting to observe if the produced results of both methods are comparable or fundamentally different.

## References

Bakillah, M., Lauer, J., Liang, S., Zipf, A., Arsanjani J. J., Mobasheri, A. and Loos, L. (2014) Exploiting Big VGI to Improve Routing and Navigation Services. In: Karimi, H. (eds.) *Big Data Techniques and Technologies in Geoinformatics*. Boca Raton, CRC Press, pp. 175-190.

Dijkstra, E. W. (1959) A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1): 269-271.

Duckham, M. and Kulik, L. (2003) “Simplest” Paths: Automated Route Selection for Navigation. In: Kuhn, W., Worboys, M. F. & Timpf, S. (eds.) *Spatial Information Theory. Foundations of Geographic Information Science: International Conference*, International Conference, COSIT 2003, Kartause Ittingen, Switzerland, September 24-28, 2003. Proceedings, pp. 169-185.

GraphStream (2016) GraphStream. A Dynamic Graph Library. http://graphstream-project.org/.

Graser, A., Straub, M. and Dragaschnig, M. (2014) Is OSM Good Enough for Vehicle Routing? A Study Comparing Street Networks in Vienna.” In: Gartner, G. & Huang, H. (eds.) *Progress in Location-based Services*, Lecture Notes in Geoinformation and Cartography. Berlin, Springer, pp. 3-18.

Helbich, M., Amelunxen, C., Neis, P. and Zipf, A. (2010) Investigations on Locational Accuracy of Volunteered Geographic Information Using OpenStreetMap Data. In: *GIScience 2010 Workshop on the role of Volunteered Geographic Information using OpenStreetMap Data*. Zurich, Switzerland.

Hochmair, H. H. (2000) "Least Angle" Heuristic: Consequences of Errors During Navigation. In: *GIScience 2000*, pp. 282-285.

Hochmair, H. H. and Navratil, G. (2008) Computation of Scenic Routes in Street Networks. In: *Geospatial Crossroads @ GI Forum '08: Proceedings of the Geoinformatics Forum Salzburg*, pp. 124-133.

Hochmair, H. H. and Fu, Z. J. (2009) Web Based Bicycle Trip Planning for Broward County, Florida. *GIS Center*, 2, http://digitalcommons.fiu.edu/gis/2.

Karrais, N., Keler, A. and Timpf, S. (2014) Routing through a continuous field constrained by a network. In: Stewart, K., Pebesma, E., Navratil, G., Fogliaroni, P. & Duckham, M. (eds.) *Extended Abstract Proceedings of the GIScience 2014*, pp. 57-61. Vienna, Austria.

Keler, A. and Mazimpaka, J. D. (2016) Safety-aware routing for motorised tourists based on open data and VGI. *Journal of Location Based Services*, 10(1), 64-77.

Krisp, J. M. and Keler, A. (2015) Car navigation – computing routes that avoid complicated crossings. *International Journal of Geographical Information Science*, 29(11), 1988-2000.

Stanica, R., Fiore, M. and Malandrino, F. (2013) Offloading floating car data. In: *Proceedings of the 2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pp. 1-9.