How to Cross Open Spaces?

Jakob Miksch[1,2], Stefan Hahmann[2], Bernd Resch[1], Johannes Lauer[2]

[1]Z_GIS, Department of Geoinformatics, Salzburg University, Austria

[2]GIScience Research Group, Department of Geography, Heidelberg University, Germany

Contact: `jakob.miksch@stud.sbg.ac.at`

Introduction

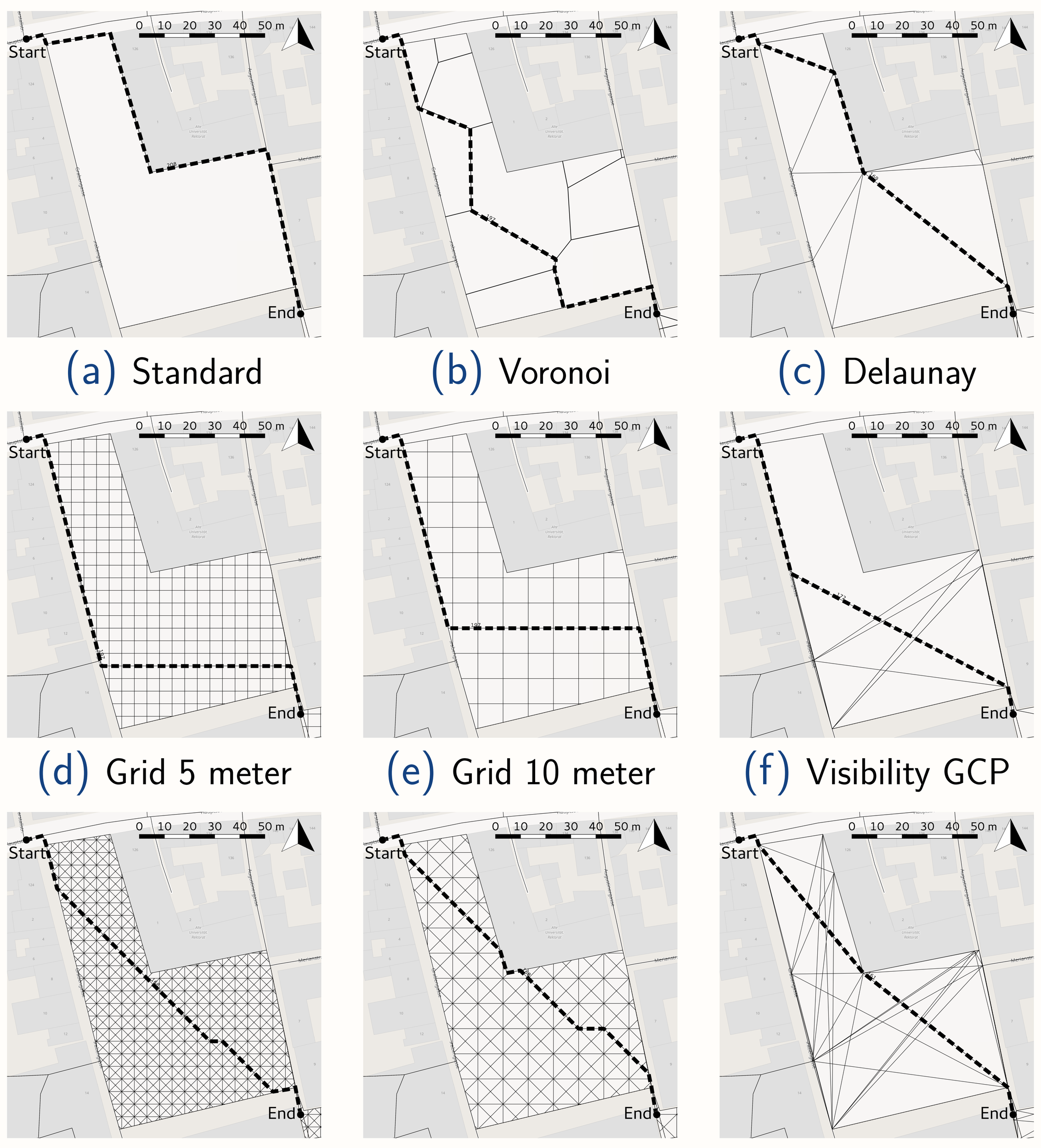
Finding the shortest path in open space is a well known challenge for pedestrian routing engines [1, 2]. A common solution is routing on the bounded polygon edges, which causes in most cases an unnecessarily long route (figure 1a). A possible solution is to create a subgraph crossing the open space. This research project assesses this approach and investigates its implications for routing engines.

Method

There are many algorithms for creating subgraphs [1]. Figure 1 shows some examples. They were created with *JTS Topology Suite*. The dotted black line shows the shortest route from the start to the destination while the graph is visualized by the thin black lines.

Algorithms

Delaunay, *Voronoi* and *Visibility All* take all nodes of the polygon into account. In contrast, *Visibility GCP* uses just points which are connected to other parts of the graph (**G**raph **C**onnection **P**oints). *Spider* is an extension of *Grid* with additional diagonals [2]. Table 1 shows that the standard route is 29% longer than the direct route with the *Visibility All* algorithm.



(a) Standard (b) Voronoi (c) Delaunay
(d) Grid 5 meter (e) Grid 10 meter (f) Visibility GCP
(g) Spider 5 meter (h) Spider 10 meter (i) Visibility All

Figure 1 : Open space algorithms.

Table 1 : Lengths of the routes and difference to *Visibility All*. Additional graph edges compared to *Standard*.

Algorithm	Length	Graph Edges
Visibility All	161 m (+ 0%)	+ 27
Visibility GCP	172 m (+ 7%)	+ 9
Delaunay	169 m (+ 5%)	+ 8
Grid 5 meter	197 m (+22%)	+ 504
Grid 10 meter	197 m (+22%)	+ 131
Standard	208 m (+29%)	+ 0
Voronoi	197 m (+22%)	+ 19
Spider 5 meter	168 m (+ 4%)	+ 1034
Spider 10 meter	168 m (+ 4%)	+ 277

Analysis

In order to compare these algorithms on a larger scale, we implemented them in the *GraphHopper* routing engine. Afterwards we computed the routing graph for each algorithm. The test area was the *OpenStreetMap* dataset of Baden-Württemberg and Austria. The processing time and the count of additional edges were compared with the respective values of the standard graph.

However, only the polygons stored as ways (closed linestrings) were considered. Polygons that are stored as relations (a data type which consists of a set of OpenStreetMap features) were not considered due to technical difficulties. Nevertheless, the analysis is still representative, since most of the open spaces are stored as ways (table 2).

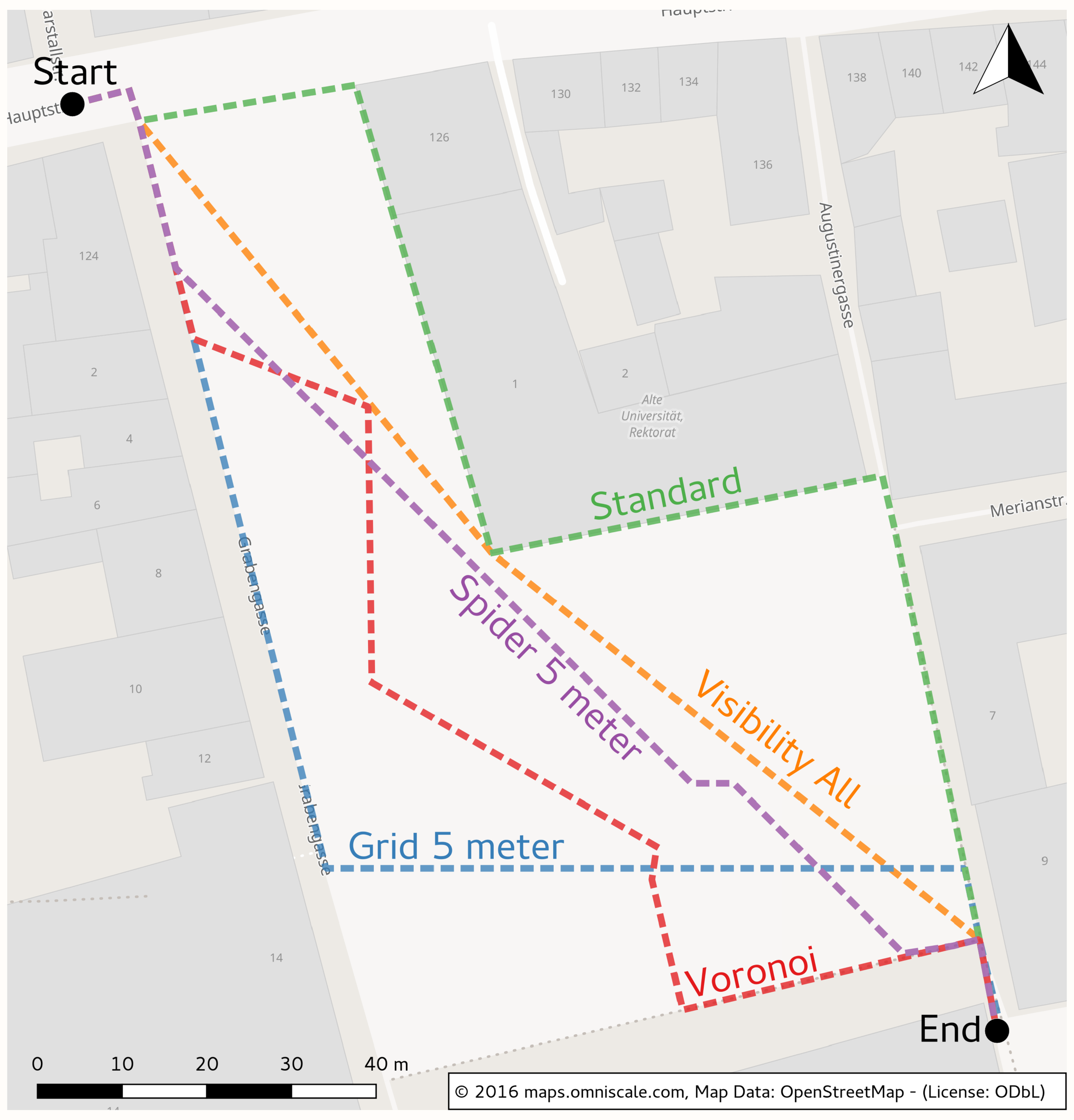


Figure 2 : Routing possibilities through the open space.

Results

The diagrams (figure 3) show that both the edge count and the computation time increases for most of the algorithms. The additional edge count of the algorithms shows a similar pattern for both regions. In contrast, the additional computation time for the *Visibility All* is much higher in Austria than in Baden-Württemberg.

Table 2 shows that there are less open spaces in Austria than in Baden-Württemberg. This explains why the relative increase of the edge count in Austria is lower. The quality of the routing result can be assessed with figure 1 and table 1. The *Visibility All* algorithm generates the most natural route and is moreover the shortest one. However, its computation time might take very long (figure 3 (b)).

Table 2 : Facts about test areas.

	Austria	Baden-Württemberg
standard comp. time	8 min 51s	13 min 50s
graph edges	2.5 million	2.7 million
open spaces *total*	2693	5641
open spaces *ways*	2459	5225
open spaces *relations*	234	416
highway ways	1.44 million	1.46 million
share open spaces	0.19 %	0.38 %

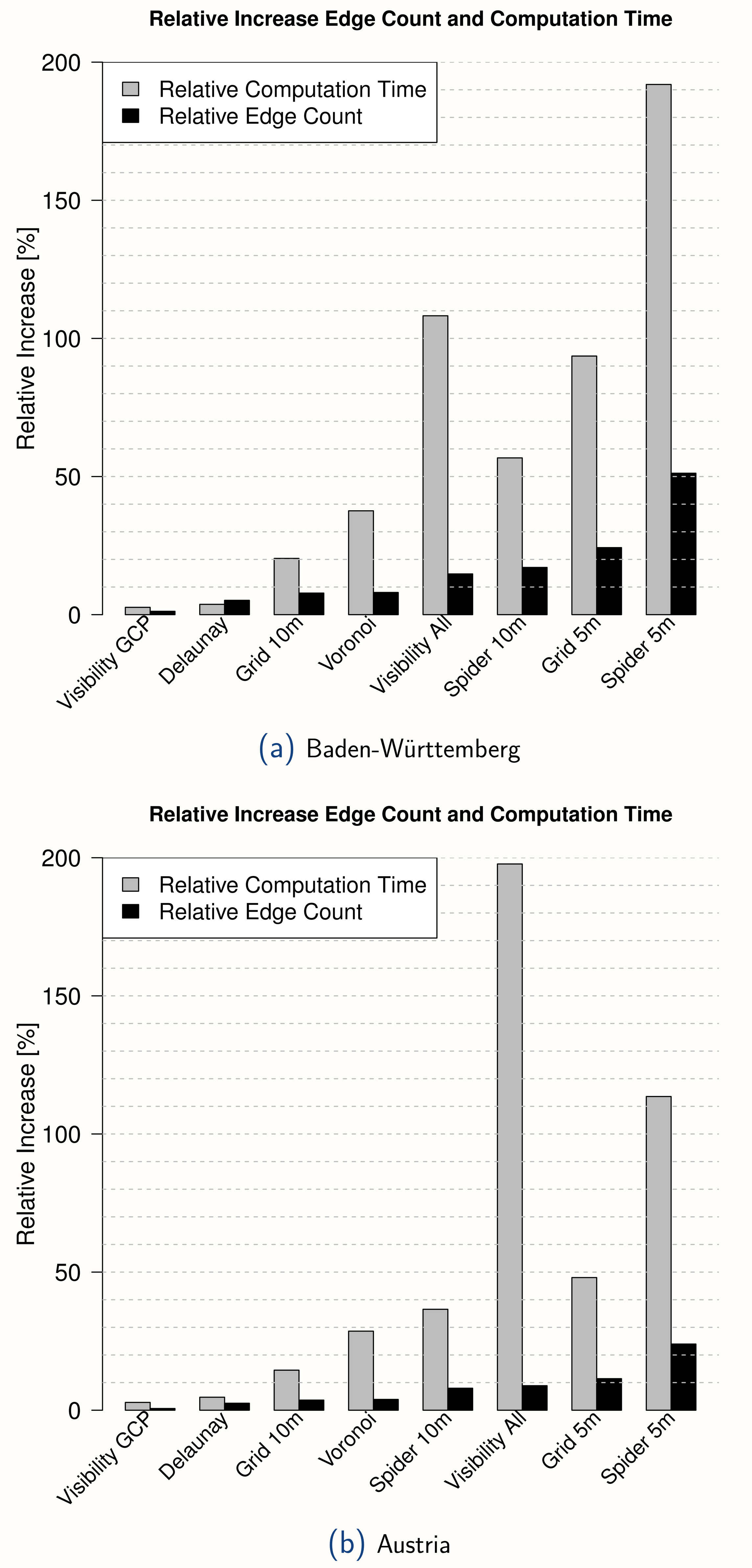


Figure 3 : Processing results (sorted by edge count).

Outlook

This work has done a pre-assessment for implementing algorithms into a routing engine. The *Visibility All* algorithm looks most promising so far. However, it should be improved by removing unnecessary connections [1]. This would decrease the additional edge count and possibly the computation time. In a next step, areas modeled as relations should be taken into account as well. Moreover it should be tested if open space routing has an effect on the performance of routing requests.

References

[1] Graser, Anita (2016). Integrating Open Spaces into OpenStreetMap Routing Graphs for Realistic Crossing Behaviour in Pedestrian Navigation, *GI_Forum Journal for Geographic Information Science*.

[2] Dzafic, Dzenan et al. (2015). Routing über Flächen mit SpiderWebGraph, *Symposium für Angewandte Geoinformatik (AGIT)*.

MyAccessible.EU

A part of the research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 612096 (CAP4Access).