Real-Time Routing with OpenStreetMap data

Dennis Luxen*
Karlsruhe Institute of Technology
Institute of Theoretical Computer Science
76128 Karlsruhe, Germany
luxen@kit.edu

Christian Vetter
Nokia gate5 GmbH
10115 Berlin, Germany
christian.vetter@nokia.com

ABSTRACT

Routing services on the web and on hand-held devices have become ubiquitous in the past couple of years. Websites like Bing or Google Maps allow users to find routes between arbitrary locations comfortably in no time. Likewise on-board navigation units belong to the off-the-shelf equipment of virtually any new car.

The amount of volunteered spatial data of the OpenStreetMap project has increased rapidly in the past five years. In many areas, the data quality already matches that of commercial map data, if not outright surpass it.

We demonstrate both a server and a hand-held device based implementation working with OpenStreetMap data. Both applications provide real-time and exact shortest path computation on continental sized networks with millions of street segments.

We also demonstrate sophisticated real-time features like draggable routes and round-trip planning.

Categories and Subject Descriptors

H.3.5 [**Information Systems Applications**]: Web-based services; G.2.2 [**Mathematics of Computing**]: Graph Theory—*Graph algorithms*

General Terms

Algorithms

1. INTRODUCTION

Routing has become ubiquitous with popular web services like Bing or Google Maps and on-board navigation devices in virtually every new car. Finding shortest paths in a road network is a problem that was solved in the early ages of computation. Unfortunately Dijkstra's seminal algorithm does not scale to large graphs and most schemes that have been advised for compensation of that fact do not produce optimal routes. In the past five to ten years the algorithm engineering community developed algorithms and data structures that provide substantial speedups over Dijkstra's algorithm and guaranteed optimal routes.

Albeit available in a technical sense, these highly efficient algorithms are not yet applied in a wide range. The most important applications for fast routing are server-based web services and hand-held based navigation. While both use-cases look different at first, the technical challenges to design scalable systems are mostly the same. Project OSRM [13] is a server based implementation, while MoNav [12] is an implementation for hand-held devices such like tablet computers or smart phones.

*Partially supported by DFG Grant 933/5-1.

ACM SIGSPATIAL GIS '11 November 1-4, 2011. Chicago, IL, USA

2. RELATED WORK

Substantial work has been done by the algorithm engineering community regarding speedup-techniques to Dijkstra's original algorithm. The road network is modelled as a graph $G = (V, E)$ where the junctions correspond to nodes and the street segments to edges. *Goal direction* provides a sense of guidance to Dijkstra's algorithm while hierarchical techniques exploit the natural hierarchy of road networks. A sophisticated version of an earlier method is *A* with landmarks and triangle inequality (ALT)* [6]. One of the first techniques that provides optimal results with substantial speedups is the method of *arc flags* [11]. Hierarchical techniques follow the notion that sufficiently long routes will enter the long distance network at some point in time, i.e. enter a highway or national road. *Contraction Hierarchies (CH)* [4] have a very convenient trade-off between preprocessing and query time. Road networks of continental size can be preprocessed within a matter of minutes and queries run in the order of about a hundred microseconds. CH heuristically orders the nodes by some measure of importance and shortcuts them in this order. Here, shortcutting means that a node is removed from the graph and as few shortcut edges as possible are inserted to preserve shortest path distances. The resulting data structure consists of the union of the original edges and of all generated shortcuts. A query, which is essentially a bidirected Dijkstra, only needs to follow those edges that lead to more important nodes. This means the algorithm works on a directed acyclic graph (DAG), which usually limits the search space to a few hundred nodes. Bauer et al. investigate combining goal-directed techniques and hierarchy-based methods [1]. Their *CHASE* algorithm even computes shortest paths in road networks in the order of ten microseconds.

Sanders et al. [15] show that it is feasible to efficiently im-

plement CH on mobile devices with very limited resources and still achieve substantial speedups over a run Dijkstra's algorithm on server machines. Kieritz et al. [9] solve the technical challenges on how to implement the CH preprocessing on a cluster of commodity hardware. Though their description mostly refers to a time-dependent version of CH, all the techniques are also applicable to a static road network. We refer the interested reader to [2] for an overview of the the research on route planning algorithms.

3. IMPLEMENTATION

Both implementations have been done in C++ and are open-source. Thus the code is available to any one on the world-wide web [12, 13]. Even though OpenStreetMap is used as a main data source, it requires only little work to add further data sources. The server runs on Linux while the hand-held implementation is available for a range of devices running e.g., Windows (Mobile), Linux variants and Symbian OS.

The main algorithmic components for both implementations are the same. The mobile version computes routes off-line, in contrast to many solutions that just connect to a routing server to achieve acceptable query speed. The mobile implementation does use a different graph data structure though, which will be detailed in a later paragraph.

Preprocessing. We support both main (compressed) OpenStreetMap formats like PBF and OSM XML, and make it easy to use different routing profiles, e.g., motorcar or bicycle routing. The preprocessing is optimized to run on normal desktop computers: We use external memory algorithms to only consume a modest amount of memory. The most time intensive parts of the preprocessing are parallelized, making full use of the multi-core processors present in most modern computers.

Mobile Data Structures. The greatest limitation of hand-held devices comes from a limited I/O bandwidth to the flash memory. Special data structures have been devised for the hand-held implementation [15]. The fact that the Contraction Hierarchies graphs forms a DAG, is used to construct a topological node ordering that greatly increases data locality. The data structure is then compressed into blocks of 4 kilobytes, allowing for random access while optimizing for flash memory accesses. Since retrieving information about the route, i.e., geometry and maneuver annotation, takes up a large amount of time, we precompute this information for important shortcut edges. These pre-unpacked shortcuts speed up long distance queries considerably. Altogether, this makes the routing procedure very effective I/O wise, while using only a limited amount of RAM.

A low-depth spatial tree structure is used to facilitate reverse geocoding. In contrast to a normal quad-tree where each node has four potential children, we subdivide the space into 1024 parts. This way we can load whole blocks of data at once while reducing the amount of random I/Os.

Server Architecture. The architecture of the server implementation was chosen to be scalable with regard to the number of users and the amount of data. For example, components are independent of each other. Sub-services like serving map tiles, geocoding, and of course routing, are realized as plugins and can be changed with a few lines of code. The server engine also does not save any information on the session, i.e. it is stateless. The state is only saved at the client, which is a JavaScript-based browser application. This makes it easy to distribute the service across several servers and data centers.

So far the server has to rely on external sources to provide geocoding. Users enter free-form location descriptions and expect the server to be error-correcting. While the basic research, e.g. [8], is published, it has yet to be implemented in a production-grade quality.

Table 1: Statistics on several OpenStreetMap data sets.

		Berlin	Germany	Europe
	nodes	52 781	4 182 094	19 432 484
	edges	139 558	10 141 120	48 604 198
mobile	processing[s]	20	380	1 934
	space[MB]	6	500	2 517
	query[ms]	35	83	128
server	processing[s]	34	597	2 486
	space[MB]	11	1 128	6 137
	query[ms] 5	8	12	27

Experimental Results. We tested the performance of both implementations on several OpenStreetMap extracts (c.f., Table 1). The server implementation was tested on an Intel Core2Duo running at 2.67GHz, whereas a Nokia N900 ARMv7 clocked at 600MHz was used for the mobile version. Note that the times of server and mobile implementation are only barely comparable to each other, since each implementation performs tasks the other doesn't. Also, routing makes up only a fraction of the query times.

It becomes obvious that our solution is fast enough for the user not to notice any delay. Compared with contemporary solutions it is noteworthy that we are able to produce exact results, while being as fast or faster as top-grade heuristics. Furthermore the query times scale very well with the map size, from metropolitan maps up to continental ones. Note that the number for the server do not include network transfers because this highly depends on the connection a user has.

Vanishing Bottlenecks. We have seen that the actual routing algorithm runs in the order of a few (server) to a hundred milliseconds (hand-held) on data covering the European continent. Thus, routing is not a bottleneck anymore, and other components become obstacles. For example, the hand-held-based variant spends most of its I/O operations to retrieve the route's geometry and compute driving maneuvers, rather than computing the route itself.

For the server-based implementation on the other hand, bandwidth and network latency are now the biggest concern. Most of the I/O is done to transfer the geometry of the route. Since, the client does not have any map data, but only graphic tiles, this data must be transferred. To cope with the problem we apply two algorithms that effectively and efficiently reduce this data by at least an order of magnitude. First, the geometry is generalized according to the zoom level using a heuristic variant of the classic Douglas-Peucker algorithm [3] that can be implemented to run in $O(n \log n)$

Map data (c) OpenStreetMap contributors, CC-BY-SA

Figure 1: Screenshots of the hand-held (left) and server-based implementation online (right).

time [7]. Second, a string of longitude and latitude coordinate pairs is highly compressible. We implement the Google Maps Encoded Polyline Algorithm Format[1] which is simple and offers great compression rates. Our implementation is well-tuned and also exploits SIMD-instructions available on modern processors. Last but not least, gzip compression as defined in the HTTP 1.1 standard is applied to further reduce the amount of data. A long-distance route of 3 000 kilometers can be encoded in full details occupying less than 60 kilobytes of space.

Latency on the Internet is tens of milliseconds and users usually expect instantaneous answers from a website. Answering queries below a threshold of 100 milliseconds is what web-services usually aim for. A low algorithmic overhead means that the scalability of the server engine is excellent, i.e., the server's efficiency is mostly limited by its network connection. A demo installation on a virtual server with 4GB of RAM and 1800 (virtual) MHz is able to serve more than a thousand queries per minute including overhead for network transfer.

New Features. The small search space of the speedup-technique we employ does not only enable us to handle requests quickly, but also empowers us to implement new features in real-time. The most prominent feature is planning of round-trips. This includes computing a distance table containing distances between all pairs of (intermediate) stops as well as solving a traveling salesman problem (TSP) instance. Distance tables can be efficiently computed [10] with our algorithm and since we allow 25 intermediate stops, the TSP instance can be solved by heuristics with good quality. Without a speedup-technique this would not be feasible since computing the distance table on a continental network would easily take minutes or even more.

4. DEMONSTRATION DESCRIPTION

The demonstration will highlight the real-time capabilities of both routing engines. Special focus will be on running *shortest path queries*, *draggable routes*, and of course computation of *round-trips* on OSRM. Also, we will show the respective capabilities of MoNav with a hands-on presentation of a device running the latest version of the software.

[1]http://code.google.com/apis/maps/index.html

Users will be able to use the web service with their computers right away. A public demo installation of the server engine is running online at:

- http://map.project-osrm.org

The hand-held implementation is available as download for several devices at:

- http://code.google.com/p/monav/

Note that the project is in rapid progress and that functionality may change over time.

5. FUTURE WORK

The development of the server version will focus on several features. Most prominently, the server only features one cost metric currently. In the near future, the server should be enhanced by metric for at least bicycle and pedestrian routing. Semi-time-dependency, like roads that can be accessed only at certain times of the day or like ferries, need the development of further algorithms. Turn costs can be modeled by an edge based graph, a graph theoretic modelling, that allows to use the existing algorithms and just changes the input data. Since the amount of data the OpenStreetMap Project provides still has very high growth rates, the implementation of a fully distributed system is an aim for the mid-term.

The mobile version will also support turn costs and restrictions. However, since data size is more important on mobile devices, we will implement an optimized Contraction Hierarchies variant [5], that can handle turn costs without transforming the graph into an edge based one. Once QtQuick [14] is supported by a wide variety of devices we will switch the GUI implementation to QML.

A nice-to-have feature for the OpenStreetMap project itself would be the development of a data quality tool based on routing, because quality and consistency will become one of the most important factors to distinguish map data providers. A goal for the long-term future is the incorporation of live traffic data, which will provide many interesting algorithmic challenges.

6. REFERENCES

[1] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm. *ACM Journal of Experimental Algorithmics*, 15(2.3):1–31, January 2010. Special Section devoted to WEA'08.

[2] D. Delling, P. Sanders, D. Schultes, and D. Wagner. Engineering Route Planning Algorithms. In J. Lerner, D. Wagner, and K. A. Zweig, editors, *Algorithmics of Large and Complex Networks*, volume 5515 of *Lecture Notes in Computer Science*, pages 117–139. Springer, 2009.

[3] D. H. DOUGLAS and T. K. PEUCKER. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112, 1973.

[4] R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In C. C. McGeoch, editor, *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, June 2008.

[5] R. Geisberger and C. Vetter. Efficient routing in road networks with turn costs. In *SEA'11*, pages 100–111, 2011.

[6] A. V. Goldberg and C. Harrelson. Computing the Shortest Path: A* Search Meets Graph Theory. In *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 156–165. SIAM, 2005.

[7] J. Hershberger and J. Snoeyink. An o(nlogn) implementation of the douglas-peucker algorithm for line simplification. In *Proceedings of the tenth annual symposium on Computational geometry*, SCG '94, pages 383–384, New York, NY, USA, 1994. ACM.

[8] C. Jung, D. Karch, S. Knopp, D. Luxen, and P. Sanders. Engineering Efficient Error-Correcting Geocoding. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM Press, 2011.

[9] T. Kieritz, D. Luxen, P. Sanders, and C. Vetter. Distributed Time-Dependent Contraction Hierarchies. In P. Festa, editor, *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*, volume 6049 of *Lecture Notes in Computer Science*. Springer, May 2010.

[10] S. Knopp, P. Sanders, D. Schultes, F. Schulz, and D. Wagner. Computing Many-to-Many Shortest Paths Using Highway Hierarchies. In *Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07)*, pages 36–45. SIAM, 2007.

[11] E. Köhler, R. H. Möhring, and H. Schilling. Acceleration of Shortest Path and Constrained Shortest Path Computation. In *Proceedings of the 4th Workshop on Experimental Algorithms (WEA'05)*, volume 3503 of *Lecture Notes in Computer Science*, pages 126–138. Springer, 2005.

[12] MoNav. http://code.google.com/p/monav/.

[13] Project OSRM. http://project-osrm.org.

[14] QtQuick. http://qt.nokia.com/qtquick.

[15] P. Sanders, D. Schultes, and C. Vetter. Mobile Route Planning. In *Proceedings of the 16th Annual European Symposium on Algorithms (ESA'08)*, volume 5193 of *Lecture Notes in Computer Science*, pages 732–743. Springer, September 2008.