Evaluating Contraction Hierarchies on Real and Generated Road Networks

Arun Mahendra Paul Butler Yang Xu

December 2014

Abstract

One commonplace application of shortest-path graph search is navigational routing. Given that the road network is relatively static, a family of algorithms have been invented for shortest-path queries in applications where pre-processing the data is possible. One such algorithm is Contraction Hierarchies, which is optimal and does not rely on localized search. We investigate the graph properties of road networks by evaluating Contraction Hierarchies on real-world and generated road networks. Our goal is to gain a better understanding of why the algorithm works and whether it can be applied successfully to graphs that are not road networks.

1 Introduction

The task of finding the best[1] path on a road map reduces to finding the shortest path on a weighted graph. Shortest path finding is a classic problem in computer science, and early route-finding systems have improved on classical approaches through heuristics and approximation.

Route finding in practice lends itself well to decentralized search algorithms[SS12]. It's rare that the shortest route from point A to point B involves a long drive in the opposite direction, and intelligently pruning such routes has been used in the past to speed up path finding.

Additionally, some systems have built-in heuristics based on the observation that transportation infrastructure is designed in a hierarchical fashion: a typical trip might start on a small residential street, travel to a major road, then a highway, back to a major road, and finally onto another residential street[SS05]. Heuristics can take advantage of this to avoid searching for residential streets in the middle of a long interstate trip.

In recent years, these heuristic approaches have inspired a family of more formal algorithms, beginning with Highway Hierarchies and more recently Contraction Hierarchies. These algorithms involve first preprocessing the graph into an implicit hierarchy, and then using a modified, bidirectional version of Dijkstra's Algorithm[Dij59] to find the shortest path. This is well-suited for the online routing model, where the underlying map data changes infrequently.

With the success of these algorithms on road network data, it's natural to wonder whether they would improve shortest-path finding performance on non-road networks. While we suspect that these algorithms work in part because of properties of road networks, an understanding of what these properties are could both inform future work on Contraction Hierarchies and help find potential applications of the algorithm outside of road networks.

[1] "best" usually means "quickest", but any metric that is a linear combination of its parts can be used

2 Prior Work

2.1 Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks

In [Gei+08], Geisberger et al. build on previous approaches to preprocessing for shortest-path search in order to introduce a method called Contraction Hierarchies. In addition to showing that the empirical performance of this method is an improvement over the previous state-of-the-art, a considerable contribution is the simplicity of the method over previous work, most notably Highway Hierarchies[Bas12].

Like previous work, the Contraction Hierarchies algorithm consists of two distinct stages: preprocessing and querying. Motivated by the real world use case of low latency shortest path searches, the performance goal is oriented towards minimizing query time at the expense of preprocessing performance.

In the **preprocessing phase**, an ordering over the nodes is chosen. The ordering does not impact correctness of the algorithm, but it is vital for performance of both the preprocessing and query phases. An optimal ordering of nodes is APX-hard[Mil12], but simple heuristics have been shown to work quite well.

For each node in the chosen order, that node and associated edges are removed from the graph. "Shortcut" edges are added between pairs of nodes in order to preserve the shortest path between every remaining pair of nodes. Another data structure maps from this new edge to the original set of edges that represent the actual shortest path between those nodes.

This process is continued until all nodes in the graph have been "contracted". A new graph is constructed by adding all of the original and "shortcut" edges back to the original graph, and annotating the nodes with their ordering.

In the **query phase**, a complete Dijkstra search is initiated from the source and target nodes. The search is restricted in that from the source, it only visits nodes that appear later in the ordering. From the source, only nodes with a lower order are visited. If a good ordering was chosen during preprocessing, this restriction greatly limits the search and can be done quite quickly.

From these Dijkstra searches, the nodes that are settled in both searches are considered, and their shortest distances to the source and target are summed to get the total distance. Whichever has a shortest total distance belongs to the path, and their paths are expanded to get the final shortest path.

The contraction hierarchies algorithm is a definite improvement over previous methods, but the authors do not touch on its applicability to non-map data. In fact, despite working with weighted graphs as an abstraction, the literature has by and large ignored any use of the algorithms outside of mapping. The reader with a graph theory background must wonder what properties of road networks allow the algorithm to work, and whether there are other graphs for which the algorihtm is useful.

2.2 Graph generation models

Early attempts at modeling road networks were characterised by a simple planar model based on the assumption that real world road networks have very few crossings (intersections). These assumptions have been shown to be inaccurate by Eppstein and Goodrich's empirical analysis on US road network data and concludes that there is a substantial number of road crossings in the real world. Abraham et al. introduce a new model that uses an input set of points to seed the generation of local road networks. Multiple scaled versions these local networks are generated and connected to each other to form a large road network. Importantly, this model captures the low highway dimension property of real world road networks. This model, although a step forward from the planar model, still fails to describe certain properties essential to real world road networks such as varying road density in urban and rural areas.[Kal+06]

Kalapala et al[Kal+06] studied road networks of United States, England and Denmark and empirically showed that navigation through real world road networks is scale invariant. They observed that for most navigation, the path from source to destination goes through hierarchies in a symmetric fashion and this pattern was invariant of the total distance travelled. This observation was formalized using a fractal model that follows a power law dual degree distribution [Eis10]. Although the model was an improvement over the previous ones, this model is too symmetrical when compared to a real world road network and therefore was not good a representative model route planning studies.

Eisenstat in [Eis10] introduces a generative model for road networks based on Quad Trees. Although similar to the fractal model, this model relies on two parameters to control the random generation of network: the number of intersections and density of sprawl. The network is formed by randomly dividing squares in iterations into quadrants to form a Quad Tree structure. High density regions of the network contains large number small squares. Appropriate choice of control parameters will yield networks with edge density comparable to that of a real road network.

3 Method

Open Source Routing Engine[LV11] (OSRM) is an open source application for route-finding on OpenStreetMap data. We chose OSRM for our project because it contains a high-quality open source implementation of Contraction Hierarchies, and its modular design allows us to plug in an arbitrary graph without major changes to the OSRM source code.

To measure the efficacy of Contraction Hierarchies on a given dataset, we used a metric called "expansion" which is the number of shortcuts added normalized by the number of edges in the original graph. This is a reasonable metric because the goal of the heuristic in the CH algorithm is to minimize the shortcuts added[Gei+08].

We modified the data pipeline used by OSRM to capture attributes of the graph before and after the CH algorithm runs. Additionally, we implemented a tool to convert SNAP[LS14] graphs and generic weighted edge lists to the custom binary formats used by OSRM so that we could run the algorithm on generated graphs.

4 Real Datasets

4.1 OpenStreetMap

OpenStreetMap[HW08] is an extensive worldwide map dataset including road-level details. It is distributed in full under the liberal Open Data Commons Open Database License. The dataset is compiled from government institutions as well as individual volunteer contributors. We use sections of OpenStreetMap as a baseline for the performance of the CH algorithm, and compare it to the performance of our generated graph models. We used two OSM datasets: the entire state of Delaware, and the District of Columbia.

5 Generated Datasets

5.1 GNM Model

Real-world road networks are subject to physical constraints. Each node (intersection) is a point in three-dimensional Cartesian space and although road speed limits differ, the weight between two nodes is highly correlated with their distance.

In order to determine whether this alone was a sufficient property to make the CH algorithm effective on the data, we extended the G_{NM} graph generation procedure. After assigning a latitude and longitude to each node uniformly at random, we generate random edges as in G_{NM}. Then every edge is assigned a weight based on the Euclidean distance between the nodes it connects.

Additionally, two variations were constructed where edges were chosen randomly but discarded if they exceeded a given distance. In the graph GNM(1) this distance is 10,000 and in GNM(2) it is 5,000, where the nodes in both are placed uniformly at random in a space of 100,000 $\times$ 100,000.

5.2 Fractal Model

[Kal+06] introduced an approach called ”Fractal Model” to generate road networks that simulates the structure of real world road networks. It starts with a unit square as shown in the Figure 1.

5.2.1 Road network generation

Divide the square evenly into 9 smaller squares by placing 4 roads: two paralleled vertical roads and two paralleled horizontal roads. Then choose certain subsets of the smaller squares and do the placing roads and dividing as stated in previous step. In our implementation, we choose each square with an index of odd number to do the dividing. Repeat selecting subsets and dividing until we have the expected number of roads.

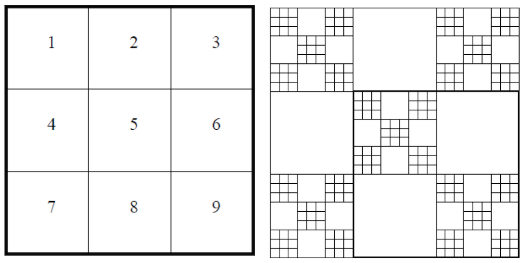


Figure 1: Fractal Model for road network [Kal+06]

5.2.2 Dual model representation of road network

Also this paper introduced a novel representation of road network called dual model. In the traditional transition of road network to graph, intersections and roads are represented by nodes and edges respectively. As a result of this primal representation, almost all vertices have degree of 4, which does not agree with the fact that one intersection (node), in most cases, is a crossing of two roads as opposed to 4 roads. To better understand the structure of the road network, it employed the dual model to generate graph based on road network: a road in the network is denoted as a node in the graph while one edge connects two nodes if corresponding two roads have one intersection. In our implementation, we give each point in the road network a coordinate. Then each node in the final graph has a coordinate which is set to be the coordinate of the mid-point of the road. The distance of the edge is calculated as the Euclidean distance based on the coordinates of the two nodes. The following table shows the results of graph generated by different subdivison schemas.

Index	Nodes	Edges	Eff. Diameter	Closed Triangles
odd	1000	5984	5.46	0
even	1000	5984	6.81	0
all	1000	5984	4.86	0

Table 1: Fractal Model basic graph statics

5.3 QuadTree Model

Eisenstat introduces random road network generation using QuadTree data structure in [**Eis10**]. We implemented this model in python and generated random road network of varying sizes.

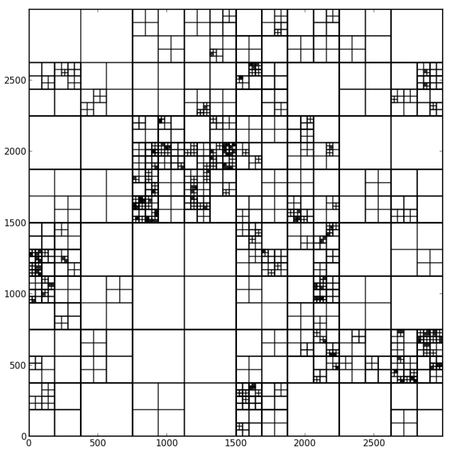
5.3.1 QuadTree

The main component of the model is the Quadtree data structure. Quadtree is a tree data structure similar to a binary tree with the difference that each node could have up to four children instead of two. In our implementation each node either has four children or no children. The first node in the tree forms the initial plane. Each node in the tree represents a square and its four children represent the division of the square into four quadrants. Squares are added to the tree by recursively dividing squares into four quadrants.

5.3.2 Road network and coordinate generation

Each node in the tree represents a square in a plane. For a given number n of nodes, we randomly divide squares to four quadrants, i.e. add four children to leaf nodes in the QuadTree. Each edge of the square forms a road connecting two points. We implemented a Point class that represents a point in the 2 dimensional space. The Point class implements Python's standard comparison, hash methods and also a method to calculate euclidean distance to another point in the same plane. Once the desired number of squares are generated, we recursively go through the nodes of the tree and generate Points based on the box formed by the squares in the tree. From these Points we subsequently generate a list of edges and the euclidean distance between them. Additionally, we implemented utility methods to perform Breadth First Search, add nodes to specific nodes in the tree and print statistics. To visualize the generated road network we created a utility to plot the network using Matplotlib [Hun07]
The following table and figures illustrate two random road networks generated using the QuadTree model.

Initial Plane Size	Nodes	Edges	Eff. Diameter	Closed Triangles
3000 pts	2468	4181	32.6	0
5000 pts	2982	4993	34.7	0



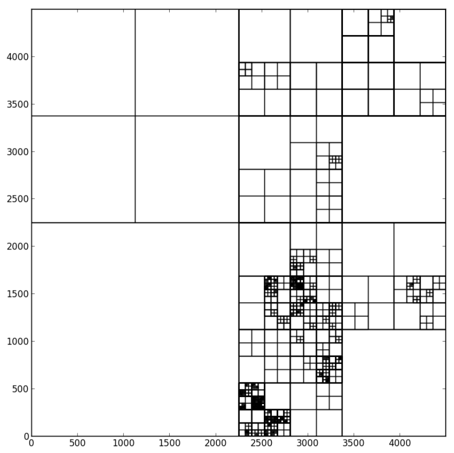


Figure 2: QuadTree Model generated random road networks. Left: Initial plane with 3000 points with 4181 roads. Right: Initial plane with 5000 points and 4993 roads.

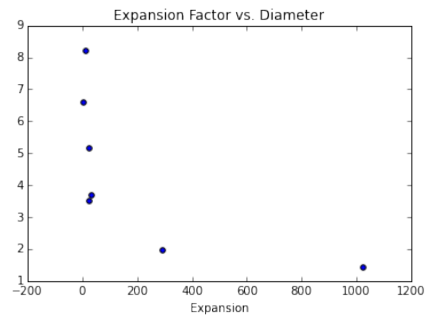
From the networks in Figure 2 it is noticeable that density distribution of roads depend on the initial plane size. An interesting point to note here is that unlike real road network graphs the generated road network has no closed triangles, this means that road networks generated using QuadTree model will have zero clustering coefficient.

6 Results

Input	Nodes	BiDir. Edges	Contracted Edges	Expansion	Clust. Coef.	Eff. Diam.	Runtime
OSM Delaware	261389	519812	746953	1.4370	0.0005	1023.3366	37.7454
OSM D.C.	88574	171238	333443	1.9472	0.0010	291.9773	38.7574
GNM	1000	2000	16410	8.2050	0.0033	11.6833	15.1291
GNM(1)	1000	2000	10304	5.1520	0.0188	24.3440	4.3801
GNM(2)	1000	2000	7393	3.6965	0.0836	32.0596	4.0288
QuadTree	1385	4696	32957	3.5091	0	26.0538	8.2300
Fractal	1000	5984	78959	6.5975	0	5.4649	299.6593

The expansion of the real-world graphs were lower than our generated models, which indicates that the CH algorithm is dependent on graph properties which are latent to real road networks that are absent in the networks generated from models. Additionally, the effective diameter was much higher for the real-world data.

CH performed worst on the GNM model, which indicates that weights that are bounded by Euclidean distance are not a sufficient property to CH to work effectively. The distance-limited variations do perform better, and there is an apparent inverse relation between effective diameter and expansion.



The QuadTree model performed better than the Fractal model with an expansion factor of 3.5 compared to 6.6. This is apparently due to the fact that the diameter of the QuadTree is much larger due to its non-uniform node density.

6.1 Conclusion

Based on previous literature, we used the resultant size of the contraction hierarchy as a proxy for the performance of the algorithm on each dataset. We found that on a GNM random graph, the CH algorithm resulted in a vastly larger contraction hierarchy. When we used graph generation algorithms - QuadTree model and the Fractal model, we found that the contraction hierarchies were more manageable and usable, within a factor of two of real-world road data. Therefore we conclude that it is not enough to merely have distances that are bounded by a factor of their distance in Euclidean space; it is also important that the graph be hierarchal in shape. We observe an empirical relationship between diameter of a graph and its expansion factor. This is unfortunate because it limits the applicability of the CH algorithm outside of road network applications, but answers an question previously unanswered by the literature.

7 References

References

[Dij59] E.W. Dijkstra. "A note on two problems in connexion with graphs". English. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. ISSN: 0029-599X. DOI: `10.1007/BF01386390`. URL: `http://dx.doi.org/10.1007/BF01386390`.

[SS05] Peter Sanders and Dominik Schultes. "Highway Hierarchies Hasten Exact Shortest Path Queries". In: *Proceedings of the 13th Annual European Conference on Algorithms*. ESA'05. Palma de Mallorca, Spain: Springer-Verlag, 2005, pp. 568–579. ISBN: 3-540-29118-0, 978-3-540-29118-3. DOI: `10.1007/11561071_51`. URL: `http://dx.doi.org/10.1007/11561071_51`.

[Kal+06] Vamsi Kalapala et al. "Scale invariance in road networks". In: *Phys. Rev. E* 73 (2 Feb. 2006), p. 026130. DOI: `10.1103/PhysRevE.73.026130`. URL: `http://link.aps.org/doi/10.1103/PhysRevE.73.026130`.

[Hun07] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95.

[Gei+08] Robert Geisberger et al. "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks". In: *Proceedings of the 7th International Conference on Experimental Algorithms*. WEA'08. Provincetown, MA, USA: Springer-Verlag, 2008, pp. 319–333. ISBN: 3-540-68548-0, 978-3-540-68548-7. URL: `http://dl.acm.org/citation.cfm?id=1788888.1788912`.

[HW08] Mordechai (Muki) Haklay and Patrick Weber. "OpenStreetMap: User-Generated Street Maps". In: *IEEE Pervasive Computing* 7.4 (Oct. 2008), pp. 12–18. ISSN: 1536-1268. DOI: `10.1109/MPRV.2008.80`. URL: `http://dx.doi.org/10.1109/MPRV.2008.80`.

[Eis10] David Eisenstat. "Random road networks: the quadtree model". In: *CoRR* abs/1008.4916 (2010). URL: `http://arxiv.org/abs/1008.4916`.

[LV11] Dennis Luxen and Christian Vetter. "Real-time routing with OpenStreetMap data". In: *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS '11. Chicago, Illinois: ACM, 2011, pp. 513–516. ISBN: 978-1-4503-1031-4. DOI: `10.1145/2093973.2094062`. URL: `http://doi.acm.org/10.1145/2093973.2094062`.

[Bas12] Prof. Dr. Hannah Bast. *Efficient Route Planning Lectures 6 & 7*. 2012. URL: `http://ad-wiki.informatik.uni-freiburg.de/teaching/EfficientRoutePlanningSS2012`.

[Mil12] Nikola Milosavljević. "On Optimal Preprocessing for Contraction Hierarchies". In: *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. IWCTS '12. Redondo Beach, California: ACM, 2012, pp. 33–38. ISBN: 978-1-4503-1693-4. DOI: `10.1145/2442942.2442949`. URL: `http://doi.acm.org/10.1145/2442942.2442949`.

[SS12] Peter Sanders and Dominik Schultes. "Engineering Highway Hierarchies". In: *J. Exp. Algorithmics* 17 (Sept. 2012), 1.6:1.1–1.6:1.40. ISSN: 1084-6654. DOI: `10.1145/2133803.2330080`. URL: `http://doi.acm.org/10.1145/2133803.2330080`.

[LS14] Jure Leskovec and Rok Sosič. *Snap.py: SNAP for Python, a general purpose network analysis and graph mining tool in Python*. `http://snap.stanford.edu/snappy`. June 2014.